UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

TESIS DOCTORAL

Herramientas Basadas en IA para el Diseño y Control de Calidad
para el Usuario Final en Experiencias Interactivas

AI-Enabled Tools for End-User Design and Quality Assurance
in Interactive Experiences

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Pablo Gutiérrez Sánchez

DIRECTORES

Pedro Antonio González Calero
Marco Antonio Gómez Martín

UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

TESIS DOCTORAL

Programa de Doctorado en Ingeniería Informática

Herramientas Basadas en IA para el Diseño y Control de Calidad
para el Usuario Final en Experiencias Interactivas

AI-Enabled Tools for End-User Design and Quality Assurance
in Interactive Experiences

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Pablo Gutiérrez Sánchez

DIRECTORES

Pedro Antonio González Calero
Marco Antonio Gómez Martín

AI-Enabled Tools for End-User Design and Quality Assurance in Interactive Experiences

UNIVERSIDAD
COMPLUTENSE
MADRID

PhD THESIS

Dissertation submitted to obtain the degree of Doctor in Computer Science and Engineering by
Pablo Gutiérrez Sánchez

Supervised by
Pedro Antonio González Calero
Marco Antonio Gómez Martín

Facultad de Informática
Universidad Complutense de Madrid
Madrid, 2025

Herramientas Basadas en IA para el Diseño y Control de Calidad para el Usuario Final en Experiencias Interactivas

UNIVERSIDAD
COMPLUTENSE
MADRID

TESIS DOCTORAL

Memoria presentada para obtener el grado de doctor en Ingeniería Informática por
Pablo Gutiérrez Sánchez

Dirigida por los profesores
Pedro Antonio González Calero
Marco Antonio Gómez Martín

Facultad de Informática
Universidad Complutense de Madrid
Madrid, 2025

UNIVERSIDAD COMPLUTENSE MADRID

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D./Dña. Pablo Gutiérrez Sánchez,
estudiante en el Programa de Doctorado en Ingeniería Informática,
de la Facultad de Informática de la Universidad Complutense de Madrid, como autor/a de la tesis presentada para la obtención del título de Doctor y titulada:

Herramientas Basadas en IA para el Diseño y Control de Calidad para el Usuario Final en Experiencias Interactivas

En inglés: AI-Enabled Tools for End-User Design and Quality Assurance in Interactive Experiences

y dirigida por: Pedro Antonio González Calero y Marco Antonio Gómez Martín

DECLARO QUE:

La tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la Ley de Propiedad Intelectual (R.D. legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, modificado por la Ley 2/2019, de 1 de marzo, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita.

Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad del contenido de la tesis presentada de conformidad con el ordenamiento jurídico vigente.

En Madrid, a 2 de octubre de 2025

Fdo.: _____

Esta DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD debe ser insertada en la primera página de la tesis presentada para la obtención del título de Doctor.

I have hated words
and I have loved them,
and I hope I have made them right.
Markus Zusak, The Book Thief

Agradecimientos

Who can say
if I've been changed for the better?
But because I knew you
I have been changed for good

Wicked: A New Musical. Stephen Schwartz

Madre mía, aquí estamos. Después de cuatro años de tesis, a punto de cerrar una etapa de la vida. Y sí, suena a cliché, pero de verdad se siente como estar cerrando un ciclo. Un ciclo con muchos altibajos, también hay que decirlo: de recibir más de una hostia de la vida, pero también de muchísimo aprendizaje y de un crecimiento personal enorme. Como diría mi amiga Cristina: *character development*.

Y dentro de todo, he tenido una suerte inmensa. Lo diré siempre por activa y por pasiva: si hay algo por lo que estoy agradecido en esta vida es por la gente que me he ido encontrando en el camino. Y en el contexto de la tesis... ya que hablamos de altibajos, ha habido muchas personitas que empujaron el carro cuando tocaba subir una cuesta imposible o que tiraron de él con cariño cuando yo me quedaba atascado en un barrizal. Así que, sin más rodeos, esta sección va para todas esas personas que, de una forma u otra, han sido mis compañeras de viaje estos últimos años.

Cómo no, tengo que empezar por los culpables de meterme en este lío y de llevarme, poco a poco, al mundo de la universidad. Pedro Antonio, Marco, gracias por darme la oportunidad de recorrer este camino y, sobre todo, por confiar en mí en cada pequeño tramo hasta llegar aquí... a menudo mucho más de lo que yo mismo confiaba en mí. Ya sé que soy un pesado, un poco llorica y que me meto en demasiados berenjenales, pero gracias por aguantarme. Y, en especial este último año, gracias por tener siempre la puerta abierta y por intentar hacer esta recta final lo más llevadera posible. Me habéis hecho sentir muy arropado, y eso no tiene precio. Gracias, Pedro, por esas charlas de corazón en Londres, por tus consejos y por haber sido tanto un "papi de tesis" como un segundo padre en más de un momento vital.

No puedo olvidarme de Pedro Pablo, que para mí siempre será mi "tercer

tutor en las sombras", como me gusta llamarle cuando no me oye. Porque siempre ha estado dispuesto a echarme un cable en todo momento, a tener una charla en su despacho, en el pasillo, con un café de lo que hiciera falta en cualquier momento. Gracias por dejarte engañar para venirte a Japón, porque era un viaje en el que me hacía especial ilusión que estuvieras, y porque los recuerdos contigo siempre son más bonitos. Y gracias por tener siempre esa ilusión por todo y ese cariño en lo que haces. Eres increíble.

Si mis tutores me dieron la oportunidad de hacer esta tesis, mis amigos del 409 se aseguraron de que pudiera disfrutarla, de una forma u otra, en cada momento del proceso. Al principio, cuando estaba más solo que la una entre teletrabajo y escuchar el eco de la última planta de la biblioteca, fue Ale quien se movió para que pudiera tener un sitio en el 409 con todos los demás. Desde entonces no he dejado de sentirme acompañado. Y aquí tengo que multiplicar por mil las gracias a Toni y a Ale: por estar ahí siempre, por ser tan auténticos y con tan buen corazón, por hacerme sentir que pertenecía, por esos abrazos top tier, por las risas a diario... y hasta por esa indiferencia inicial de Toni en el metro, que hoy es una anécdota con la que me río siempre. Gracias por la terapia gratuita (y por los salseos, marujeos y frikeos mil) que me habéis regalado. No sé qué habría hecho sin vosotros.

A Toni le agradezco especialmente haberme despertado la pasión por la docencia: tienes una energía contagiosa, de mayor quiero ser como tú. Y por haberme hecho sentir que podía confiar en él para cualquier cosa y en todo momento, no sabes lo mucho que he apreciado tenerte al lado estos años. Y a Ale, por ser como eres: un marujón y a veces un poco pesado, sí, pero con un corazón de oro. Gracias por todas las conversaciones, por tenerme siempre en cuenta, por esas sesiones conjuntas de terapia. Hablaba con Clara, después de Japón, de cómo has sido la persona que se ha colado de forma más inesperada en nuestros corazoncitos. Hace cuatro años nunca hubiera imaginado que llegarías a ser uno de mis grandes apoyos y alguien a quien tengo tantísimo cariño. A los dos, gracias por ser mis amigos. Os quiero.

Marta, amigui, gracias por ser como eres. Aunque no lo creas, abrir la puerta del despacho por las mañanas y verte lanzarme una de tus sonrisas era suficiente para ponerme en un chip positivo para todo el día. Eres un sol y vales mucho más de lo que crees. Cris, mi novia de la facultad, contigo me pasó algo curioso: cuando nos vimos por segunda vez, ya tú con plaza en la Complu, y sin saber muy bien dónde o cuándo nos habíamos conocidos en la primera, ya me trataste como si me conocieras de toda la vida, con un cariño que no puedo hacer más que agradecerte con todo mi corazón. Un cariño que has continuado desde entonces y por el que me siento tremendamente afortunado, eres la mejor. Y Miguel... ya sabes lo que te toca, que eres un solete y un gamberro, que te preocupas todo el rato por los demás y que tienes un corazón enorme aunque nos metamos contigo llamándote egocéntrico por

picarte. Es nuestra forma de decirte que robas toda la atención con tu forma de ser. Me encanta tenerte aquí. Hemos reído, hemos llorado, hemos vivido de todo... ¡pues ya está! Y por supuesto, Antonio, Isma, Dani, Jorge, gracias por endulzar esos cafés, esas comidas y tardes de juegos de mesa o charlas sobre escape rooms o recetas de cocina.

Clara, no imaginas lo mucho que agradezco haberte conocido. En ti he encontrado una amiga de verdad, de las que están en las buenas y en las malas, y de las que no te juzgan por nada. Un alma súper afín, que para mí ha sido una zona segura, un lugar donde refugiarme cuando no sabía dónde meterme. Gracias por hacerme sentir comprendido, por esa empatía desbordante y por esa sonrisa capaz de inspirar a cualquiera a mover montañas. Y Álex, creo que poco puedo escribir aquí que no sepas ya. Eres alguien súper importante en mi vida, que ha estado ahí siempre. Siempre. En cada derrumbe, en cada alegría, te quiero muchísimo y te admiro un montón. Ojalá poder seguir compartiendo muchos años más a tu lado, porque eres una persona increíble y fundamental para mí.

Miguel, ahora sí, te toca. Has sido mi pilar fundamental en estos años, mi compañero de vida, la persona que se ha estado tragando mis tardes de lloros, mis crisis existenciales, mis ataques de ansiedad, ausencias y paranoias varias, probablemente más que nadie, y con una paciencia y un cariño que no tienen precio. Eres una persona increíble y te mereces todo lo bueno que te pase en la vida. Gracias. Gracias. Gracias. Te quiero muchísimo.

Sarah, Irene, ya ni me molesto en contar los años. Quizá no nos veamos tanto como quisiéramos, pero sois mi primera familia elegida. Personas con las que, pase lo que pase, siempre es como si no hubiera pasado el tiempo. Veros siempre me carga las pilas, me hacéis sentir en casa, como un hermanito mimado y a veces hasta malcriado. Os quiero. Chema, Cris, Enrique, Guille, Ming, Rafa, Rubén... sois mi segunda familia elegida. Con vosotros el tiempo y la distancia tampoco importan. Estoy convencido de que ni la tesis ni la carrera hubieran llegado a buen puerto sin esas tardes de estudio conjunto, desahogos, quedadas y lloros compartidos. Habéis definido mi vida universitaria y todo lo que vino después. Os quiero un montón.

Y a Cristina, simplemente gracias por existir y, citando a aquel poemario que guardo como un tesoro, "I think you were always meant to know me a little better than anyone else." Eres de las personas más importantes de mi vida.

Pablo, Paburo, lo creas o no, probablemente de no haberte conocido en aquel año de Erasmus en Berlín, no estaría escribiendo estas líneas ahora. Porque fuiste tú quien despertó esa pasión por el querer hacer más con videojuegos, por el querer crear, por querer volverme loco y hacer lo que me pedía el corazón.

Y, por último, pero no menos importante, mi familia. Mamá, nunca olvidaré esas conversaciones en el coche, de vuelta a casa desde la facultad,

cuando me escuchabas llorar diciendo que no valía para esto y tú me dabas ánimos para seguir. Ahora es una anécdota graciosa, pero en todos esos momentos el saber que tenía un espacio seguro en ti era un regalo del cielo. Papá, gracias por confiar siempre en mí, por creer en mí más que yo mismo, por quererme incondicionalmente contra viento y marea. Y mamá, aunque me grabaras a fuego que en esta vida hay que ser humilde, no puedo evitar sentir un orgullo enorme por todo lo que habéis hecho por mí, por lo que hacéis cada día, por el amor y la dedicación que mostráis día, tras día, tras día... Lo siento, pero tengo que presumir, porque no conozco a nadie capaz de dar tanto amor y dedicación a su familia como vosotros: sois increíbles. Os quiero con todo mi corazón.

María, a día de hoy, más de 12 años más tarde, todavía sigo recordando lo que me escribiste en la última página de la agenda de Estados Unidos, y que me ha acompañado siempre en cada paso de la vida. Tenías razón, las cosas han ido pasando muy deprisa desde entonces, tan frenéticamente que ahora da vértigo ponerse a recordarlo. Pero, ¿sabes? con todas los logros y cagadas por el camino, creo que podemos sentirnos orgullosos de haber ido poco a poco tomando los mandos de control de este tren y haber ido tomando las decisiones que nos salían del corazón en cada momento, tal y como escribías entonces. Gracias por estar ahí siempre, por tu dedicación, por tu cariño, por ser como eres. Por ser la mejor hermana mayor que podría haber tenido nunca. Y como decía de papá y de mamá, siento tener que presumir de ti, porque eres increíble y una persona a la que admiro muchísimo, por luchadora, por currante, por tanto amor como das a los demás. Te quiero un montón. Y por supuesto, a Santi, Sofía, Nicolás y Olivia, sois maravillosos.

En conclusión, soy un enorme afortunado. Me he repetido hasta la saciedad, pero es la verdad: he tenido la suerte de encontrarme con personas increíbles en mi vida, y no puedo hacer más que dar gracias por ello. Gracias a todos y cada uno de vosotros por haber hecho que este viaje fuera tan especial. Os quiero un montón. Gracias. Gracias. Y, cuando se apague el eco del último "gracias", mil gracias más.

Abstract

Video game development has become an increasingly complicated and multidisciplinary process requiring tight coordination between programmers, artists, and designers. Among these roles, designers face their own set of unique challenges: they must create compelling mechanics that take into account the diversity of player styles and preferences, ensure consistency with the game's objectives, and balance their creative vision with technical and budgetary constraints. As games continue to grow in scale and complexity, verifying and iterating on design decisions becomes increasingly difficult, leading to a need for tools and methodologies that support designers without demanding advanced technical or programming expertise on their side.

This thesis explores the intersection of artificial intelligence, end-user programming, and game design, with the aim of developing accessible and usable tools that empower designers in two primary domains: automated quality assurance (QA) and player modeling, and the creation and deployment of interactive experiences for educational and cultural purposes. The contributions include theoretical frameworks, algorithmic strategies, and practical tools, all grounded in real-world applications and validated through empirical studies and/or collaborations with institutional partners.

The first line of research focuses on the development of AI-driven methodologies to support QA tasks in commercial video games, particularly through automated regression testing and gameplay validation. We propose machine learning techniques that enable autonomous agents to detect anomalies in gameplay and design by comparing their performance across different game versions. A key contribution is the design of visual, user-friendly tools aimed at non-technical profiles, including node-based editors that allow the construction of reinforcement learning agents capable of autonomously identifying potential errors or unwanted gameplay variations—without requiring manual reward specification. These ideas are implemented and validated in Liquid Snake, a 3D stealth game prototype developed in Unity, conceived as a realistic and reusable testbed for AI-powered QA. The environment has been iteratively expanded to explore new testing algorithms and reward generation strategies based on linear temporal logic task specifications (LTL), allowing designers to define and evaluate testing tasks in a formal and inter-

pretable way.

Building upon these notions, the LTL toolkit is then extended toward player modeling, describing a set of methods and algorithms to automatically extract behavioral models from player traces and represent them in accessible formats. These models can lead to a deeper understanding of player behavior and provide support for data-informed design iterations that stay aligned with user expectations and play styles. The proposed clustering and specification distillation techniques demonstrate the feasibility of generating understandable behavioral explanations, while highlighting areas for design improvement.

The second research axis addresses the design and deployment of interactive experiences beyond the entertainment context, particularly in education and cultural heritage. A major contribution is the development of the EnigMachine Editor, an authoring tool that enables non-programmers to create augmented reality (AR) adventures inspired by treasure hunts and escape rooms, via a web-based interface. This tool has been successfully used by designers and educators to build AR-enabled interactive narratives hosted in museums such as the García Santesmases Computer History Museum and the National Museum of Natural Sciences in Madrid. Extensive user studies demonstrate the tool's effectiveness in supporting both creative expression and educational goals, while also identifying usability challenges and opportunities for refinement.

Further evaluation is provided through Enigma Bio, a pedagogical game deployed at the National Museum of Natural Sciences. An A/B study with schoolchildren shows that while the game fosters engagement and improves understanding of biodiversity-related concepts, its educational impact increases significantly when combined with in-person instruction. These findings underscore the value of integrating digital and traditional approaches in serious game design.

Finally, the thesis explores the integration of generative AI—specifically large language models (LLMs)—into the authoring pipeline. We present and evaluate a system capable of producing structured, semantically coherent drafts of treasure hunt-style experiences based on artifact descriptions and narrative goals provided by designers. This system illustrates the potential of LLMs to act as creative co-authors in the early stages of interactive experience design, while also revealing key challenges related to factual accuracy, stylistic consistency, and control over design intent.

Keywords: game design, game testing, authoring tools, machine learning.

Resumen

El desarrollo de videojuegos se ha convertido en un proceso cada vez más laborioso y multidisciplinar que exige una estrecha coordinación entre programadores, artistas y diseñadores. Entre estas figuras, los diseñadores se enfrentan a una serie de retos únicos: por un lado, deben crear mecánicas atractivas que tengan en cuenta la diversidad de estilos y preferencias de los jugadores; por otro, se ocupan de garantizar la coherencia con los objetivos del juego y de equilibrar su visión creativa con las limitaciones técnicas y presupuestarias del proyecto. Conforme los juegos van creciendo en escala y complejidad, verificar e iterar sobre dichas decisiones de diseño se vuelve cada vez más difícil, lo que lleva a una necesidad de herramientas y metodologías que respalden a los diseñadores sin exigirles conocimientos técnicos o de programación avanzados.

Esta tesis explora la intersección entre la inteligencia artificial, la programación para el usuario final (end-user programming) y el diseño de videojuegos, con el fin de implementar herramientas accesibles que doten de autonomía a los diseñadores en dos ámbitos principales: el control de calidad (QA) y el modelado de jugadores automatizados, y la creación y desplegado de experiencias interactivas con fines educativos y culturales. Entre las contribuciones de este trabajo se incluyen modelos teóricos, estrategias algorítmicas y diversas herramientas prácticas, todos ellos construidos sobre la base de aplicaciones prácticas reales y validados mediante estudios empíricos y/o colaboraciones con instituciones colaboradoras.

La primera línea de investigación se enfoca en el desarrollo de métodos basados en inteligencia artificial para proporcionar asistencia en tareas de QA en videojuegos comerciales, en particular a través de pruebas de regresión y de validación de la jugabilidad automatizadas. Se proponen metodologías de aprendizaje automático que permiten construir agentes autónomos capaces de detectar potenciales anomalías en la experiencia de juego y las especificaciones de diseño, mediante el análisis comparativo de su comportamiento a lo largo de distintas versiones del juego. Una de las principales aportaciones en este marco consiste en el diseño de herramientas visuales orientadas a usuarios no técnicos, que permiten construir este tipo de agentes mediante editores basados en nodos. Estos son capaces de identificar automática-

mente errores o variaciones indeseadas en la jugabilidad, sin necesidad de configurar manualmente funciones de recompensa o configurar los parámetros del algoritmo. Estas ideas se implementan y validan en LIQUID SNAKE, un prototipo de juego de sigilo en 3D desarrollado en Unity como banco de pruebas realista y reutilizable para QA automatizado con IA. Este entorno ha sido expandido iterativamente a lo largo de la tesis y ha servido para explorar nuevos algoritmos de prueba y estrategias de generación de recompensas basadas en especificaciones de diseño en lógica lineal temporal (LTL), lo que permite a los diseñadores definir y evaluar tareas de manera formal e interpretable.

Sobre estas bases, se extiende el marco LTL hacia el modelado de jugadores, proponiendo una vía para extraer automáticamente modelos de comportamiento a partir de trazas de juego y representarlos en formatos comprensibles para diseñadores. Dichos modelos permiten una comprensión más profunda del comportamiento de los jugadores y facilitan iteraciones de diseño informadas por datos, manteniéndose alineadas con sus expectativas y estilos. Las técnicas de agrupamiento y destilación de especificaciones desarrolladas demuestran la viabilidad de generar explicaciones comprensibles sobre el comportamiento del jugador, al tiempo que identifican oportunidades de mejora en el diseño.

La segunda línea de investigación se orienta al diseño y despliegue de experiencias interactivas en contextos más allá del entretenimiento, con especial atención al ámbito educativo y al patrimonio cultural. Una contribución destacada es el desarrollo del ENIGMACHINE EDITOR, una herramienta de autoría que permite a personas sin conocimientos de programación crear aventuras de realidad aumentada inspiradas en búsquedas del tesoro y escape rooms, a través de una aplicación web. Esta herramienta ha sido utilizada con éxito por diseñadores y educadores para construir narrativas interactivas con realidad aumentada en contextos museísticos, como el Museo de Historia de la Informática García Santesmases o el Museo Nacional de Ciencias Naturales de Madrid. Estudios de usuario exhaustivos muestran que la herramienta resulta efectiva tanto para fomentar la expresión creativa como para alcanzar objetivos educativos, y al mismo tiempo revelan retos de usabilidad y posibles mejoras.

Una evaluación complementaria se lleva a cabo mediante ENIGMA BIO, un juego pedagógico implementado en el Museo Nacional de Ciencias Naturales. Un estudio A/B con escolares demuestra que, aunque el juego mejora el interés y la comprensión de conceptos sobre biodiversidad, su impacto educativo se incrementa significativamente cuando se combina con instrucción presencial. Estos resultados subrayan el valor de integrar enfoques digitales y tradicionales en el diseño de juegos serios.

Finalmente, la tesis explora la incorporación de inteligencia artificial generativa —específicamente grandes modelos de lenguaje (LLMs)— en el pro-

ceso de autoría. Se presenta y evalúa un sistema capaz de generar borradores estructurados y semánticamente coherentes de búsquedas del tesoro, a partir de descripciones de artefactos y objetivos narrativos proporcionados por diseñadores. Este sistema demuestra el potencial de los LLMs como coautores creativos en las etapas iniciales del diseño de experiencias interactivas, a la vez que pone de relieve importantes desafíos relacionados con la precisión factual, la coherencia estilística y el control sobre las decisiones de diseño.

Palabras clave: diseño de videojuegos, testing de videojuegos, herramientas de autoría, aprendizaje automático.

Contents

List of Figures

List of Tables

Part I

Contents of the Thesis

Chapter 1

Introduction. Objectives of the Thesis

You can't connect the dots looking forward; you can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future.

Steve Jobs

1.1 Introduction

Video games have evolved significantly over the last decades, becoming increasingly complex software products. This growth is manifested both in the extension of their worlds and the length of their experiences, as well as in the vast number of elements and assets that make up their creation. The production of a video game is a multidisciplinary effort that traditionally involves three main roles: programmers, artists, and designers. Programmers are responsible for the development of the underlying tools, components, and systems. Artists are responsible for the creation of the audiovisual content that brings the game to life, including scenarios, characters, animations, and audio. The designer, a unique role in the video game industry, has the crucial task of defining the gameplay—the mechanics that govern player interaction—that the programmers must implement. In addition, the designer establishes the layout and dynamics of each level, deciding when and where challenges appear, using the tools provided by the programmers and the visual resources of the artists to build the setting, aesthetics, and narrative of the game.

As of today, the responsibilities of designers in the development of a commercial video game are very diverse, focusing on the following key areas:

- **Game mechanics design and testing**. Designers are responsible for defining game mechanics and balancing difficulty, often through a process of trial and error. In complex games, the many interacting components can make it difficult to spot issues, sometimes leading to softlocks or game-breaking bugs. As games evolve and assets are reused across levels, changes in one area can have unintended effects elsewhere. Detecting and resolving these subtle design flaws requires ongoing testing, which becomes increasingly costly as the game expands.

- **Alignment with the game's audience**. More subtly, another major challenge in design comes from the need to account for the diversity of player styles and preferences, which can mean that a design that works well for one type of player may not be equally effective for another. Therefore, designers must be able to adapt their mechanics and levels to different play styles. This often leads to a level being designed with multiple player types in mind, allowing, for example, for both aggressive and stealthy strategies to complete the same level. While this does not mean that the design must be generalistic and applicable to every kind of player, it is crucial to understand the product's target audience and how they are expected to interact with the game. During closed testing phases or after launch, designers can gather valuable insights from player interactions — qualitatively, by watching recordings or observing players, and quantitatively, by analyzing telemetry data — and use these insights to refine the design and enhance the player experience. This analysis process is often referred to as player modeling.

- **Purpose-driven design and narrative integration**. When games are created with goals beyond entertainment—such as education, training, or social impact—designers must ensure that gameplay, narrative, and user experience align with the project's intent. This involves crafting mechanics that support learning or reflection, integrating accurate and relevant content, and developing compelling narratives that engage the target audience while reinforcing key messages. Balancing factual accuracy with engaging gameplay can introduce additional complexity, often requiring collaboration with domain experts. Whether used in classrooms, museums, or health contexts, these purpose-driven games demand careful design to maintain consistency, relevance, and player immersion.

Effective collaboration between programmers and designers is a fundamental aspect of professional game production. There is often a dialogue, even a debate, between the designer's vision of what is enjoyable and the programmer's technical feasibility of implementing those ideas. Game design

has an inherent creative dimension, where the designer must explore various alternatives to achieve the desired player experience. In parallel, programmers must provide designers with an intuitive "interface" — which may take the form of configurable parameters, visual editors, or scripting languages — to enable them to carry out this creative exploration. Simultaneously, it is crucial to identify and correct errors that designers may introduce during their work. As designers gain access to more expressive tools, their creative freedom increases—but so does their responsibility for the final result and the risk of introducing errors. This challenge of enabling users without programming skills to configure complex software systems falls within the research field of what is known as "end-user programming."

1.2 Objectives of the thesis

The primary objective of this thesis is to develop methodologies and end-user tools that assist game designers in creating and maintaining high-quality interactive experiences, without requiring extensive technical expertise. Building on the challenges identified in previous work, this research concentrates on two main areas:

Objective 1. Develop accessible artificial intelligence techniques for automatic quality assurance, game design validation and testing, and player modeling in video games.

Objective 2. Design intuitive tools that support the creation and deployment of engaging interactive experiences across various domains, including education and cultural heritage.

These high-level research directions define the core focus of this thesis and are expanded upon through the following specific objectives.

Supporting objective 1: AI for game design quality assurance

Objective 1.1. Develop machine learning-based methodologies that enable the creation of autonomous agents capable of performing automated regression testing in video games. These tools should support designers and QA professionals by reducing the need for manual testing, while remaining easy to use and accessible to users without prior experience in artificial intelligence or programming.

Objective 1.2. Construct a testing environment that is realistic, sufficiently complex, and representative of game development scenarios involving diverse mechanics, genres, and interaction patterns. This environment should serve as a testbed for evaluating and demonstrating the capabilities of autonomous testing agents in varied gameplay situations.

Objective 1.3. Explore the definition of a formal, yet intuitive and designer-friendly way of specifying testing goals and design intentions. The objective is to identify a specification approach that aligns closely with designers' existing workflows and cognitive models, allowing them to express expected behaviors and gameplay constraints without requiring deep technical knowledge.

Objective 1.4. Investigate the use of these specification mechanisms for building interpretable models of player behavior. These models should help designers better understand how players interact with their games and provide actionable insights for refining game mechanics and improving overall user experience.

Supporting objective 2: Tools for interactive experience design

Objective 2.1. Construct a platform for interactive experiences that is sufficiently generalistic, expressive, and representative of real-world purpose-driven game scenarios, such as educational or cultural heritage applications. This system should undergo a series of design iterations, incorporating feedback from designers and end-users to ensure it meets the needs of both groups effectively, and should serve as a testbed for evaluating the effectiveness of the authoring tools developed in subsequent objectives.

Objective 2.2. Design and develop authoring tools that empower designers to create interactive experiences for the platform described in Objective 2.1 without requiring advanced technical expertise. These tools should prioritize accessibility and ease of use, while offering sufficient expressiveness to support diverse styles of interaction and gameplay.

Objective 2.3. Conduct a thorough user experience evaluation for both the authoring tools developed in Objective 2.2 and the interactive experiences generated through them. This evaluation should assess the usability, effectiveness, and overall satisfaction of both designers and end-users, providing insights into how well the tools support the design process and how engaging the resulting interactive experiences are.

Objective 2.4. Investigate the integration of artificial intelligence methods into design tools, enabling the automated generation of early-stage drafts of interactive experiences based on high-level design ideas. The focus is on supporting the creative process while ensuring that the generated content aligns with the designer's intent and remains factually accurate.

While these two objectives focus on different stages and contexts of interactive experience development, they are linked by the common motivation of empowering designers in some of their most common workflows without

requiring advanced technical knowledge. Objective 1 centers on supporting designers within the entertainment-oriented production pipeline, where verifying and validating complex systems is a major bottleneck. By introducing accessible AI-driven methodologies for testing and player modeling, this line of work reduces the technical overhead of quality assurance and facilitates data-informed iteration.

Objective 2, in turn, addresses the complementary challenge of enabling designers and educators to create and deploy interactive experiences in domains such as education and cultural heritage. Although this context differs from commercial game QA, it shares the same underlying need of providing expressive, intuitive tools that extend designers' creative reach without enforcing programming expertise. In this sense, the move from QA to creation does not reflect a departure but a broadening of scope—showcasing how similar principles of accessibility, interpretability, and AI assistance can be applied across the lifecycle of interactive experiences, from ensuring their quality to enabling their conception and deployment.

1.3 Structure of the thesis

This thesis is presented in *thesis by articles* format. It consists of two main parts: the first part provides an overview of our research context, objectives, methods and contributions, while the second part contains a collection of publications supporting this thesis that detail the technical aspects and experimental results of the research conducted.

1.3.1 Part I: contents of the thesis

The first part of the thesis is structured as follows:

1. Chapter 2 provides a comprehensive review of the state of the art in video game testing, purpose-driven design and end-user tools for interactive experiences. It also discusses the challenges faced by designers and programmers in these areas, contextualizing the contributions made in this thesis.

2. Chapter 3 introduces the contributions made toward Objective 1, focusing on the development of an automated, agent-based regression testing framework for video games. The main contributions presented in this chapter are:

    - A general workflow for automatic agent-based regression testing in video games to perform systematic quality assurance tasks.
    - The Liquid Snake testbed for AI-based quality assurance, developed to evaluate the viability of the proposed techniques in a realistic and complex scenario.

- An algorithm and methodology for defining testing goals using temporal logic specifications, enabling designers to express their intentions in a format that closely resembles natural language. These specifications are then used to automatically train reinforcement learning agents to carry out regression testing and verify the validity of those goals throughout development.
- A visual, node-based tool in Unity 3D that allows designers to define the specifications of the testing goals and train testing agents in a way that is intuitive and close to their cognitive model, without requiring them to have prior knowledge of temporal logics or programming.
- A set of methods for building interpretable models of player behavior based on temporal logics, which can be used to analyze player interactions with the game and provide insights into how players engage with the game mechanics and levels, as well as to better understand the behavior of bots in testing tasks.

3. Chapter 4 presents the contributions made toward Objective 2, focusing on the development of a platform for interactive experiences and the tools to support their design. The main contributions presented in this chapter are:

   - A case study that illustrates the complexities and requirements of designing and deploying interactive experiences within real-world educational and cultural contexts, namely an educational escape room held over several iterations in a natural science museum in Madrid, Spain.
   - The EnigMachine ecosystem consisting of intuitive authoring tools that enable non-technical professionals to create their own interactive adventures, thereby bridging the gap between domain expertise and technical implementation.
   - A thorough player evaluation of game experiences created using our tools with museum visitors, discussing methodologies for assessing the effectiveness of interactive content in achieving engagement goals.
   - A usability evaluation of the EnigMachine Editor Tool, providing insights into how well it meets the needs of its intended users and the issues encountered during the validation process.
   - A pilot study exploring the potential of AI-assisted adventure design, discussing how artificial intelligence, and particularly large language models, can be integrated into the adventure design process to bootstrap the creation of different game elements, such as puzzles, narratives, and question blocks.

4. Lastly, Chapter 5 concludes the first block by summarizing the main contributions and findings, and outlining potential future research directions.

1.3.2 Part II: papers of the thesis

The second part of the thesis is structured as a collection of articles that have been published or accepted for publication in peer-reviewed conferences and journals. Each article is presented as a self-contained chapter, with its own introduction, methodology, results, and discussion sections. The list of articles that constitute this thesis is presented below. Each article is referenced with its corresponding chapter, where all technical details and experimental results are provided.

Paper 1: Gutiérrez-Sánchez et al. (2022) Gutiérrez-Sánchez, P., Gómez-Martín, M. A., González-Calero, P. A. and Gómez-Martín, P. P. Liquid Snake: a test environment for video game testing agents. In *Actas del I Congreso Español de Videojuegos*, Madrid, Spain, December 1-2, 2022 (edited by R. Lara-Cabrera and A. J. F. Leiva), volume 3305 of CEUR Workshop Proceedings. CEUR-WS.org, 2022. Included in Chapter 6.

Paper 2: Sagredo-Olivenza et al. (2024) Sagredo-Olivenza, I., Gutiérrez-Sánchez, P., Gómez-Martín, M. A. and González-Calero, P. A. Liquid Gym: An Open Stealth Game Test-bed for AI Techniques Controlling Complex Characters. In *Actas del II Congreso Español de Videojuegos (CEV'24)*, pages 1-12. A Coruña, Spain, 2024. Included in Chapter 7.

Paper 3: Gutiérrez-Sánchez et al. (2023b) Gutiérrez-Sánchez, P., Gómez-Martín, M. A., González-Calero, P. A. and Gómez-Martín, P. P. Reinforcement Learning with Temporal Logic Specifications for Regression Testing NPCs in Video Games. In *2023 IEEE Conference on Games (CoG)*, pages 1-8. 2023. Included in Chapter 8.

Paper 4: Gutiérrez-Sánchez et al. (2024) Gutiérrez-Sánchez, P., Gómez-Martín, M. A., González-Calero, P. A. and Gómez-Martín, P. P. A Progress-Based Algorithm for Interpretable Reinforcement Learning in Regression Testing. *IEEE Transactions on Games*, volume 16(4), pages 844-853, 2024. Included in Chapter 9.

Paper 5: Gutiérrez-Sánchez et al. (2023a) Gutiérrez-Sánchez, P., Gómez-Martín, M. A., González-Calero, P. A. and Gómez-Martín, P. P. AI Behavior Graphs: A Visual Toolkit for Defining NPC Specifications for Regression Testing. In *CEUR Workshop Proceedings*, volume 3599. 2023. Included in Chapter 10.

Paper 6: Gutiérrez-Sánchez et al. (2025b) Gutiérrez-Sánchez, P., Pérez-Liébana, D. and Gaina, R. D. Explaining and Clustering Playtraces Using Temporal Logics. In *Proceedings of the 20th International Conference on the Foundations of Digital Games, FDG '25.* Association for Computing

Machinery, New York, NY, USA, 2025. ISBN 979-8-4007-1856-4. Included in Chapter 11.

Paper 7: Gonzalez-Calero et al. (2024) Gonzalez-Calero, P., Camps-Ortueta, I., Gutiérrez-Sánchez, P. and Gómez-Martín, P. P. On the Importance of Contextualizing an Educational Escape Room Activity. In *Electronic Journal of e-Learning*, volume 22, pages 43-56, 2024. Included in Chapter 12.

Paper 8: Gutiérrez-Sánchez et al. (2025a) Gutiérrez-Sánchez, P., Gómez-Martín, M. A., González-Calero, P. A. and Gómez-Martín, P. P. Evaluating Usability for an AR enabled Escape Room Authoring Ecosystem. In *Entertainment Computing*, volume 55, page 100982, 2025. ISSN 1875-9521. Included in Chapter 13.

Paper 9: Gutiérrez-Sánchez et al. (2025) Gutiérrez-Sánchez, P., Gómez-Martín, M. A., González-Calero, P. A. and Gómez-Martín, P. P., Thawonmas, Ruck. Initializing Interactive Treasure Hunts in Cultural Heritage Sites: An LLM-Based Approach. In *Entertainment Computing - ICEC 2025* (edited by M. Sugimoto, A. Di Iorio, P. Figueroa, R. Yamanishi and K. Matsumura), pages 151-165. Springer Nature Switzerland, Cham, 2025. ISBN 978-3-032-02555-5. Included in Chapter 14.

Chapter 2

State of the Art

"Then why do you want to know?"
"Because learning does not consist only
of knowing what we must or we can do,
but also of knowing what we could do
and perhaps should not do."

Umberto Eco, The Name of the Rose

2.1 Game development and design tools

Since their first appearances in the 1960s, video games have continued to grow and improve in scope and sophistication, driven by the creativity of developers working within the technical limitations of their time. Today, the video game industry is a global economic powerhouse that generates substantial global revenue and employs workers all over the world. While a 2021 report by Price Waterhouse Coopers projected the industry to reach USD 172.4 billion in 2024, more recent analyses indicate even stronger growth. Estimates for 2025 suggest the global games market is on track to reach approximately USD 237 billion according to Midia Research[1], with some projections going as high as USD 303 billion as per Precedence Research[2]. These updated figures further underscore the industry's position as a multidisciplinary creative field with artistic ramifications, comparable to media such as cinema, theater, or literature. In fact, video game development involves collaboration among professionals from diverse domains—such as programming, art, sound design, and writing—who must integrate their expertise to produce a cohesive and engaging experience.

The end result of this collaborative process is a software application comprising one or more executable files and multiple data files. The creation of

[1] `https://www.gamesindustry.biz/midia-research-global-games-industry-estimated-to-reach-2369bn-in-2025`

[2] `https://www.precedenceresearch.com/video-game-market`

the executable is the responsibility of the programmers, while while artists and sound designers handle the data files, which include 2D and 3D models, textures, music, sound effects, and animations—collectively known as *game assets*.

The internal structure of the executable—which constitutes its software architecture—is defined by programmers and determines how different systems within the game interact. The architecture, along with its underlying technology stack, determines key development decisions such as how much of the development can be outsourced (i.e., licensed from third parties), which must me custom-built, and how much of it could be reused in subsequent games. Generally, most video games are structured in:

- **Game Engine**: responsible, among other tasks, for loading and managing all assets created by artists, handling user input, performing physical simulations, supporting network connectivity, and rendering both graphics and audio. It is usually relatively stable throughout development and often determines the technical capabilities of the game—whether it can run on a PC or console, whether it supports online play, whether it can use controllers, and so on.

- **Game logic**: in charge of translating the designers' game specifications into a format that the engine can interpret and execute. It defines the rules, mechanics, and behaviors that govern gameplay. At its core are game entities—self-contained logic components capable of performing specific tasks. Common examples include enemies, interactive objects, and environmental triggers. Because this layer is closely tied to the game design—which often evolves through iterative experimentation—it tends to change frequently throughout development, often right up to the final stages of the project.

Until relatively recently, developing a video game required significant technical expertise, primarily due to the need to build a custom game engine for each project. However, in recent years, the industry has undergone a process often referred to as the "democratization" of game development, driven by the rise of powerful, accessible commercial game engines. These engines—such as Unreal Engine[3], Unity3D[4], and Godot[5]—support the development of a wide range of modern games and include comprehensive toolsets that simplify the creation and management of game assets.

By streamlining tasks such as asset integration, user interface design, and physics simulation, these engines reduce the workload for programmers and lower the barrier to entry for designers and artists. As a result, teams with

[3] `https://www.unrealengine.com/`
[4] `https://unity.com/`
[5] `https://godotengine.org/`

limited technical backgrounds or smaller budgets can engage in game development more feasibly than ever before. This shift has been further amplified by the advent of digital distribution platforms like Steam[6], Epic Games Store[7], and Nintendo eShop[8], which eliminate the need for physical media and significantly reduce publishing costs. Together, these advancements have made game development more accessible to independent developers and small studios by lowering both technical and financial entry barriers.

In addition to accelerating development timelines, commercial game engines provide out-of-the-box solutions for traditionally complex programming challenges—such as physics systems, rendering pipelines, pathfinding algorithms, lighting models, and more. Consequently, small teams can now produce high-quality games that rival those developed by large, resource-rich studios. Today, the main tasks of the programmers of these studios consist of the implementation of the game logic block mentioned above, as the rest of the functionality is in the hands of the game engine, which is often common to all titles.

This reliance on game engines, while democratizing development, also highlights the critical role of custom tools in optimizing the game development process, particularly for designers. As emphasized in a popular 2020 GDC talk[9], the primary goal of these tools is to save time and improve efficiency, allowing designers to maintain their creative flow and iterate more rapidly on game logic and content. These tools can range from simple reusable code snippets to complex level editors tailored to the specific needs of a project. For instance, a level editor that allows for intuitive manipulation of game environments or a system that streamlines the integration of branching narrative and music can significantly reduce the time spent on repetitive tasks, freeing up designers to focus on the core creative aspects of the game.

The development of such custom tools often follows an iterative approach, where basic game mechanics and content are prototyped first before investing in more sophisticated tooling. This ensures that the tools address actual needs and avoid the pitfall of over-engineering features that may not be ultimately useful. The decision to create a tool is often driven by an "annoyance threshold"—the point at which the time wasted on a manual or inefficient process outweighs the time and effort required to build an automated solution. A classic example is the management of localized text within a game. If a designer needs to update a specific piece of UI text that appears in multiple menus and game screens, manually finding and editing each instance can be tedious and error-prone. A custom tool could provide a centralized interface

[6] `https://store.steampowered.com/`

[7] `https://store.epicgames.com/`

[8] `https://www.nintendo.com/en-gb/Nintendo-eShop/Nintendo-eShop-1806894.html`

[9] `https://www.youtube.com/watch?v=_mbOM06A5sA`

where all in-game text is listed with its corresponding IDs and locations. The designer could then edit the text once, and the tool would automatically propagate the changes across all relevant parts of the game, ensuring consistency and saving significant time, especially during localization efforts.

As game development evolves, a trend emerges with custom third-party tools. Often, these tools arise to fill specific gaps or provide more tailored mechanisms for tasks that commercial game engines do not yet address comprehensively. When a particular third-party solution gains significant traction and demonstrates a widespread need among practitioners, the developers of popular game engines often take notice. Recognizing the value and demand, these engines frequently integrate the core functionality of such successful third-party tools natively into their platforms in subsequent iterations. This process streamlines workflows for a broader user base and solidifies industry best practices. Notable examples of this phenomenon within Unity include its Localization modules, which was inspired by the need for robust localization solutions often addressed by external assets; Visual Scripting tools like *Bolt* (now owned by Unity and integrated as Unity Visual Scripting), which addressed the desire for more accessible, code-free logic implementation similar to that of Unreal's Blueprints; and the native Behavior Tree system, which responded to the widespread use of third-party AI behavior tree solutions. This cycle of third-party innovation leading to native engine integration underscores the dynamic and collaborative nature of the game development ecosystem, ultimately benefiting developers by providing increasingly comprehensive and user-friendly toolsets.

Ultimately, while commercial game engines provide a robust foundation for game development, the strategic creation and utilization of various tools remain vital for maximizing efficiency and empowering designers. We can classify these tools based on their primary function within the development pipeline. For level design, visual editors are paramount, allowing designers to directly manipulate game environments. Popular examples include the built-in level editors of the abovementioned commercial engines, which offer drag-and-drop functionality and real-time feedback, as well as specialized third-party solutions like the *Tiled* Map Editor for 2D games[10]. For crafting dialogues and narrative flows, tools often provide visual scripting interfaces to structure conversations and branching storylines without extensive coding. Examples include Articy:Draft, Twine, and the dialogue editors integrated within some game engines. When it comes to AI design, visual tools help create complex behaviors through systems like behavior trees and state machines. Both Unity and Unreal Engine have visual behavior tree editors, and there are also standalone tools and plugins like Behavior Designer for more advanced AI logic.

While the specific interfaces and functionalities of these tools differ sig-

[10] `https://www.mapeditor.org/`

nificantly based on their tasks, a common thread, especially for those aimed at designers, is their emphasis on visual interaction and minimizing the need for direct programming or scripting. This focus on user-friendliness and visual feedback ensures that individuals with varying technical backgrounds can effectively contribute to the game development process, fostering greater creative collaboration and accelerating the realization of complex game designs.

2.2 Quality assurance. Game design testing

Designers are responsible for defining the mechanics and rules that govern the player's interaction with the game, as well as balancing its difficulty and progression. This often involves a laborious process of trial and error, where practitioners experiment with different configurations and settings to find the right combination that delivers a satisfying experience and aligns with their vision for the game. In complex games, this task is far from trivial, as designers must account for a wide range of factors and interacting components, making it difficult to identify errors or design flaws and often requiring the repeated testing of the game by several professionals to evaluate the quality of the product and detect potential problems. These issues can lead to unexpected interactions that affect gameplay, ranging from softlock situations—where the player gets stuck and can make no further progress—to game-breaking scenarios, where players might become able to exploit a bug to gain unfair advantages or disrupt the game experience.

Moreover, it is crucial to consider the constantly evolving nature of the application. Video games are complex software projects involving many professionals who continuously add, remove, and modify resources and files. A design that worked well in an earlier version may not function as intended in a later build, especially when assets are reused across different parts of the game (e.g., enemy characters that appear in multiple levels or missions, collectible items, obstacles). For example, if a designer adjusts the AI of an enemy to improve the experience in one level, it may inadvertently cause problems in other environments where that same enemy is used. These can be thought of as "silent" design errors—they change the player's experience without making the game crash or become unplayable. Identifying and fixing such issues becomes an exponentially costly process as the game grows, requiring thorough and ongoing testing across all critical areas. Before we present explicit examples of design errors, let us first contextualize the testing process within the game development life cycle.

2.2.1 Testing in the game development life cycle

Within the game development process, testing falls under the umbrella of quality assurance (QA) (Bethke, 2003). The QA team ideally consists of a test lead, who is responsible for planning and organizing the tests throughout development and for proposing the procedures and standards to follow, and several testers, who carry out the tests, analyze the results, and prepare a report for the development team. While most game developers and publishers use their own testing methodologies, adapted to their specific infrastructure, it is possible to identify a set of common tests across most companies:

- **Functional or black-box tests**. Possibly the most common and costly type of test, functional testing involves repeated interactions by a group of testers with the game to verify whether a given feature behaves as expected (a player's weapon, menu navigation, the ability to complete a level or unlock an achievement, etc.). The term black-box testing refers to the fact that these tests are conducted without considering the internal implementation of these mechanics, focusing instead on verification based solely on the player's perceived experience, thus providing results that closely reflect what a potential end user would encounter.
- **Smoke tests**. Smoke testing involves running basic, quick checks that can detect serious errors early on—errors severe enough to halt further development and testing. These procedures can be seen as small, simple tests (for example, launching the game and loading a level) which, if they fail, immediately reveal a problem that prevents continued testing and requires prior code review.
- **Regression tests**. Every time a change is made to a version of the game, there is a risk that a previously successful test will start to fail. This can happen either because the new change affects previous functionality or because an undetected bug only emerges after the change. To mitigate this, regression tests are used, which involve re-running tests that were executed in the past to ensure the application still works as expected.

The testing process typically begins with the QA team developing a series of test cases. These cases describe the different functionalities to be verified, the steps required to perform the tests, and the expected outcomes. For example, in a platformer game, a test case might pose the question of whether it is possible to reach the end of a specific level. The steps needed to perform this test would include launching the game, loading the level, and using the player controls to navigate the environment in an attempt to move the character to the end point of the level. The expected result would be that the player is able to reach the end, thus completing the level.

Once a version of the game is available, testers can proceed to execute it and perform the tests by following the previously specified steps. If the test results match the expected outcomes, the test is marked as passed. Otherwise, the tester records the observed issues in a report, usually including detailed information about the test execution, such as video captures, input traces, and actions. This report is then forwarded to the part of the team responsible for addressing the issue (programmers, designers, etc.), who will resolve it within a given timeframe before sending it back to QA, where the fix must be validated by re-running the relevant test.

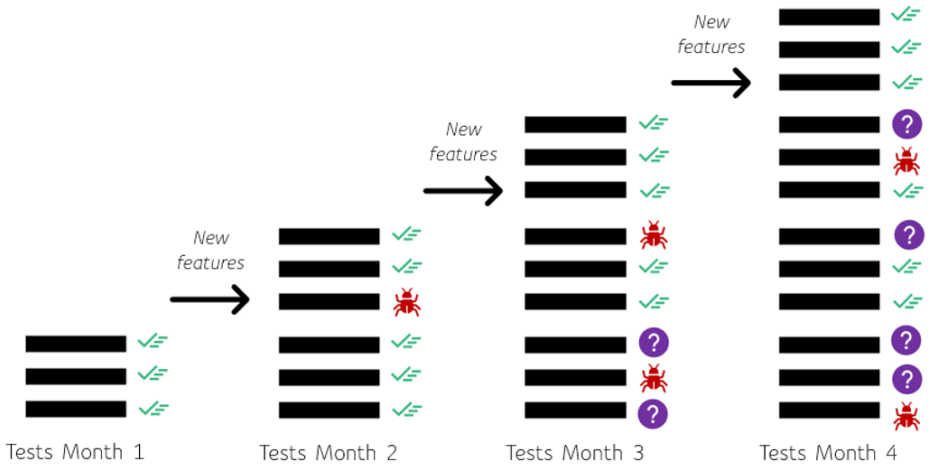


Figure 2.1: Game test coverage issues during development.

However, as game development progresses, the number of components increases—implemented mechanics, code modules, designed levels, introduced abilities, and so on. This means that the number of tests to be run and, consequently, the human resources required also grow, often exponentially. It becomes necessary not only to verify the correct functioning of new elements but also to ensure that they have not invalidated earlier ones (for example, through regression tests). This means that, ideally, every previously run test should be run again to guarantee sufficient coverage of the current version of the product. In practice, however, the testing process tends to lean towards a workflow akin to that of Figure 2.1, where old features slowly become neglected in favor of newer ones and any excess testing effort is directed towards more obvious issues. This creates a significant challenge for both small and large studios, where economic and human constraints often lead to situations where guaranteeing exhaustive test coverage becomes unfeasible.

2.2.2 Design errors and specification issues

We can define a *design specification* as a property that is expected to hold in the game. This is a deliberately broad definition, as the concept of design is generally quite wide-ranging and can encompass gameplay properties (e.g., that a level can be completed within a certain amount of time or under a set of given constraints), balancing aspects (e.g., that it is possible to win a combat encounter with any available character, or that all player choices are viable in some context), or even temporal dependencies (e.g., that a key must be obtained before opening a door leading to the exit, and that no other way exists to achieve this goal). Therefore, a design specification can refer to any aspect of the game that the designer considers relevant to achieving their vision. Later, we will focus on design specifications with a more technical and formally defined emphasis.

Given a design specification, we can think of a design error as a situation in which the behavior of the game or its target players does not align with what the designer intended. These errors can arise in various ways, and their impact can range from minor issues to critical failures that severely hinder gameplay or user experience. In this sense, it is important to distinguish these problems from implementation errors, which refer to failures in the coding or execution of a specific software feature and can, in turn, lead to design errors if the induced problem directly affects a specification (for example, a programming bug might allow the player to reach the level exit without obtaining the key, thereby violating the last specification mentioned above). Thus, design errors can often be more subtle than implementation errors, as they may not be apparent in the code or during game execution, nor easily detectable through traditional testing mechanisms such as unit tests, but they can have a significant impact on the player experience.

A simple example can be seen in a classic platformer like *Super Mario Bros.* Here, the player controls Mario, who must jump over enemies and obstacles to progress. In such a game, if the designer were to increase Mario's speed to make the game more challenging, this change might cause the character to jump too far, making it difficult for players to land on narrow platforms and leading to frustration. Similarly, a decision to alter Mario's jump height—making it either too high or too low—could break the game's level design. For instance, a higher jump might allow players to reach areas they were not meant to access at that stage, trivializing puzzles or bypassing challenges. A lower jump, on the other hand, might make it impossible to reach critical regions needed to complete a level. In both cases, a seemingly small change would require a thorough review of all levels to ensure the player experience remains intact.

A second example can be found in 3D survival horror or stealth games such as *Silent Hill* or *Metal Gear Solid.* In these titles, the player explores a complex environment while solving puzzles and avoiding enemies. In a given

development iteration, the designer may request an update to an enemy's AI so it can react to sounds the player makes—like moving, shooting, or breaking a window—making the experience more intense and realistic. However, if at that time the game already contained an earlier level in which the player was required to break windows to access certain areas and solve puzzles, with the new AI behavior, an enemy from another room might unexpectedly pursue the player into a space originally designed to be a safe, pressure-free zone, drastically altering the intended tension and pacing. Even small adjustments to AI behavior—like reaction speed or detection range—can have ripple effects in multiple parts of the game.

A third and even subtler example involves changes to shared AI modules or code components, which might be modified not only by designers but also by programmers updating the game's codebase, adding features, or refining behaviors. Enemy AI, for instance, is often built using control structures like behavior trees or state machines, with reusable modules that encode common behaviors (such as moving to a location, attacking, or patrolling). While this modularity typically accelerates development by building a shared library of interoperable components, it also increases the risk of widespread design errors when changes are made. The more generic and widely used a module is, the harder it becomes to detect and resolve unintended design consequences when it is updated.

Given these challenges, there has been a growing interest in developing automated testing solutions to assist designers and quality assurance teams in detecting and preventing such issues efficiently. This shift reflects a broader trend toward leveraging automation and artificial intelligence to manage the increasing complexity of modern games.

2.2.3 State of the art in automatic testing in video games

Automated testing of video games has garnered significant attention in both academia and industry in recent years, motivated by the need to maintain adequate coverage as games grow increasingly large and complex (Politowski et al., 2022). Within the game development process, testing is part of the broader quality assurance (QA) effort, and its importance has escalated alongside the rising complexity, scale, and multi-platform nature of modern games.

In general software development, automated testing technologies play a central role in practices such as test-driven development (TDD) (Beck, 2002) and continuous integration/continuous delivery (CI/CD) (Humble and Farley, 2010). Some notable examples of widely studied automated testing techniques are search-based testing (Alshahwan et al., 2018; Fraser and Arcuri, 2011) or symbolic execution (Cadar et al., 2008). However, strategies like these are challenging to deploy in game testing, as game playing is typically a continuous interaction process with complex graphical user interfaces

(GUIs) between the game and the player. Consequently, game testing has traditionally relied heavily on *functional* or *black-box* testing, typically carried out by teams of human testers interacting with the game to validate specific features like mechanics, menus, or level completion. This manual process is highly resource-intensive, both in time and personnel, particularly because tests must often be repeatedly run as regression tests across a large number of reused assets and levels.

A variety of tasks within the testing process are susceptible to automation (Garousi and Mäntylä, 2016), including test case design, scripting, execution and result evaluation, reporting, and test management. However, test automation cannot fully replace a human tester in all cases. For example, if we design a test to determine whether a given level in a video game is sufficiently challenging or fun, it will be very difficult to develop an automatic system that is able to provide a reliable answer to that question in the absence of a human component to support it. Taking this limitation for granted, automated tests can be conceived not as a substitute but as a complementary tool within the QA process that can lead to significant savings in terms of time and human resources.

One of the simplest options for automating the process of running a test is the use of game segments recorded manually by humans, which are then used to check that the sequences played back are capable of completing the set objective (Ostrowski and Aroudj, 2013). In these methodologies, a human tester performs a test as they normally would, but with the peculiarity that the traces of actions taken are stored for use in future regression tests. Thus, if a manually performed test is passed successfully, the sequence of tasks executed to arrive at that result can be replayed as many times as necessary as the development process progresses, resulting in what can be regarded as a regression test based on repeating a previous test exactly. If the test produces an unexpected result after changes in the application, it becomes necessary to once again resort to human testers to re-record new sequences in the modified environments, while a favorable result can be interpreted, roughly speaking, as an absence of induced errors in the corresponding functionality. While effective in some cases, these methods struggle in dynamic or non-deterministic environments; even small environmental changes can invalidate a recorded trace, requiring new human-generated playthroughs. This limitation naturally leads to the exploration of more adaptable, AI-driven solutions.

At the other end of the spectrum, we can find techniques such as *monkey testing*, an automatic black box testing mechanism aimed at applications with graphical user interfaces, based on injecting a continuous flow of input events into the system (pressing a button, touching the screen at a given point, etc.) with the aim of trying to "break" the application (Bécares et al., 2017). In this sense, the technique corresponds to a test case that asks the

question of whether it is possible to find a sequence of user actions that leads to reaching an error state in the software. In contrast to the use of recorded game segments, this technique executes a game "blindly," without any specific goal beyond breaking it, but is useful in detecting general bugs and has the advantage of being easily applied at any point in the development process without reliance on human testers. Traces that result in critical states can be included in the testing report and sent to the development team for further analysis and correction. More sophisticated variants, like evolutionary algorithms, treat input sequences as populations evolving toward increasingly destructive behaviors (Gandini et al., 2010). In this way, the test moves from a purely random methodology to a process that is more driven by the ability of traces to expose errors in the system.

As for testing methods based on agents generated by artificial intelligence algorithms, we can find numerous projects in the past few years that allow us to exemplify the techniques currently employed in this line of research. One of the most prominent uses of these methods is in what is known as *game state coverage*. This type of test aims to explore the possible game states as exhaustively as possible, asking the question of how they relate to each other. Included here are questions such as whether it is possible to find a sequence of actions that allow the player to navigate between two given states, or whether there is a risk of reaching a critical or error state from certain starting configurations, among other examples. These questions are especially prominent when the object of the test is a game composed of 3D environments with complex navigation schemes and sophisticated physics engines that may introduce unexpected failures (e.g., clipping through walls, excessive force from jumps). The fundamental idea of these tests is to develop an agent to explore a level, trying to reach all possible states and configurations and recording game traces for later analysis. Purely randomized tests are particularly poorly conditioned in this context: as the complexity of the environments increases, the execution of certain critical action sequences to progress through the levels, like chaining precise jumps across a danger zone, becomes increasingly improbable through unguided exploration.

To address these challenges, Gordillo et al. (2021) and Bergdahl et al. (2020) proposed a series of approaches based on the use of deep reinforcement learning techniques to train agents capable of exhaustively exploring game states. These papers follow the idea of what is known as count-based exploration, where agents receive rewards inversely proportional to the frequency of visited states. These works consider an agent to have visited a new state only when the distance between its current position and any previously visited state exceeds a given threshold. The reward assigned to the agent for being in a state is thus given by

$$R_t = R_{\max} * [1 - \frac{N_i}{max_{counter}}],$$

where $R_{\max}$ is the maximum reward that can be obtained in the state, N_i is the number of times the agent has visited the state, and $max_{counter}$ is the maximum number of times visiting that state will provide a non-zero reward. Additionally, this methodology makes use of an agent initialization technique strongly reminiscent of algorithms such as RRTs (Rapidly Exploring Random Trees (LaValle, 1998)), where the player is randomly placed in previously explored positions at the beginning of each training episode instead of always starting exploration from scratch. With this, trained agents achieved around 95% coverage rates in state space, as opposed to a maximum of 48% when a randomized policy was deployed. Bergdahl et al. (2020) similarly compare navigation mesh strategies with RL-trained agents for reaching arbitrary positions, demonstrating the advantage of learning-based methods in complex spaces.

Methods like these enable the detection of navigation exploits and physics bugs that would otherwise go unnoticed (e.g., walls lacking physical properties or points where the player received excessive force when jumping), as well as navigation strategies not contemplated in the original design of the environments (such as "shortcuts" to reach the goal from the initial state). These are examples of design errors that can be more easily detected with the help of AI agents, as they are capable of exploring the environment in a more exhaustive and unpredicatable way than a human player. In most cases, these design errors are linked to situations in which the player becomes able to traverse sections of their environment in ways that were not originally contemplated by the designer.

Reinforcement learning has also been used to try to model the interactions of different types of players with game environments in order to provide support in the design process. Agarwal et al. (2020) generate agents with different behaviors to play levels from a 2D platform game, *Sonic the Hedgehog 2*, using a genetic deep reinforcement learning (DRL) strategy and offer an intuitive methodology for visualizing the trajectories of the bots. Meanwhile, Ariyurek et al. (2021) use agents with different personas (explorers, pacifists, etc.) to discover different ways of approaching levels, a strategy conceptually similar to that of Holmgard et al. (2019) where a variation of Monte Carlo Tree Search (MCTS) is used to demonstrate how generative player models can be leveraged to replicate archetypal play styles across different levels. Zheng et al. (2019) introduced the *Wuji* framework, using Deep Reinforcement Learning (DRL) and evolutionary algorithms to balance between winning the game and exploring the game space for error detection, showcasing the efficacy of their method to discover bugs on sample sections of two commercial games.

While not as prominent as RL in the game testing literature, imitation learning (IL) is starting to gain some popularity in recent years. By IL we refer to the generation of agents capable of mimicking the behavior of an

expert demonstrator that provides a set of state-action pair traces as a reference for the bot to be trained. Some of the most widespread strategies are behavioral cloning (Bain and Sammut, 1995), which makes use of supervised learning techniques from the dataset of demonstrations to predict the most likely actions for a given state, and generative adversarial imitation learning (GAIL, (Ho and Ermon, 2016)), based on an adversarial approach in which the rewards of the actuator agent are proportional to how much it manages to deceive a second discriminator agent trained in parallel to learn to distinguish between human and synthetic behaviors.

In game testing, Sestini et al. (2022b, 2024) introduce the curiosity-conditioned proximal trajectories algorithm for testing complex environments, alternating between policies similar to and distant from those of a human via a weighting system of signals based on IL, RL and curiosity. Sestini et al. (2022a) propose a methodology for trace validation based on the DAGGER algorithm (Ross et al., 2011) in which the designer interactively demonstrates their desired behavior in real time. A similar approach is used in the Google Research project *Falken*[11]. Tucker et al. (2018) use inverse reinforcement learning to generate rewards from human traces for agents learning to play Atari games.

Currently in the industry, a majority of game companies adopt some sort of ad-hoc manual testing pipeline without making use of systematic or automated solutions (Aleem et al., 2018). Even when they do, most of the testing techniques used are based on semi-automatic script-based testing (Lovreto et al., 2018), classical AI, such as manually programmed agents via behavior trees or state machines, model-based approaches (Iftikhar et al., 2015; Stahlke et al., 2019), or randomized exploration methods to complement manual testing tasks[12,13]. While all of these can improve the efficiency of the testing process, they still require substantial manual efforts to develop the corresponding scripts and agents. On top of that, the ad-hoc scripting approach is often limited to specific game scenarios that follow a pre-defined strategy, which can lead to a lack of coverage in the testing process.

One of the main reasons behind the slow adoption of machine learning techniques in the video game industry is the lack of trust and understanding of these techniques by industry professionals. Most game designers and testers lack technical training or background in machine learning, which makes it difficult for them to understand how these algorithms work and how they can be applied to their work. This problem is compounded by the opaque nature of many of these algorithms, which are often viewed as "black boxes" of limited interpretability and transparency, requiring blind faith in

[11] `https://github.com/google-research/falken`

[12] `https://www.gdcvault.com/play/1027049/-Final-Fantasy-VII-Remake`

[13] `https://www.gdcvault.com/play/1026366/Automated-Testing-of-Gameplay-Features`

their results (El-Nasr and Kleinman, 2020). As a result, industry professionals may be reluctant to adopt techniques that they cannot fully understand or control (Jacob et al., 2020).

In addition to the trust gap, integrating ML into existing development pipelines presents significant technical and organizational challenges (Gillberg et al., 2023). These include the complexity of integrating ML systems into current game engines, as well as the need for substantial computational resources, all of which can be, on the one hand, prohibitively expensive and, on the other hand, difficult to justify in production environments where timelines are tight and human resources are limited (Zhao, 2024). Ethical considerations further complicate adoption, particularly concerns over algorithmic bias (El-Nasr and Kleinman, 2020) and the potential displacement of creative roles through automation (Merchant, 2024), both of which may lead to resistance to change on the part of developers. To overcome these challenges, we consider it critical to build tools and models that are accessible and understandable to designers, programmers, and testers, emphasizing the usability, transparency, and ease of implementation of these technologies. Recent work, such as Sestini et al. (2024), has shown how the introduction of visual explanations in the reporting interfaces presented to designers can result in a significant increase in user confidence and acceptance of automatic playtesting, as well as increased confidence in the use of machine learning techniques in the development process.

2.3 Purpose-driven design. Authoring tools

2.3.1 Serious games and applications

While the primary goal in commercial games is usually to entertain the user, game design can also pursue broader objectives such as education, training, or raising social awareness. This distinction forms the basis of what are commonly referred to as *serious games*, defined as "games that do not have entertainment, enjoyment or fun as their primary purpose" (Djaouti et al., 2011). In these cases, designers must ensure that the game fulfills its intended purpose and that the player experience aligns with the project's objectives. This may involve creating gameplay mechanics that encourage learning or reflection, as well as integrating narrative elements that reinforce the game's message. A crucial aspect of these purpose-driven games is factual accuracy, requiring designers to ensure that the content is both relevant and correct—often through collaboration with subject matter experts or through extensive research. For instance, video games are increasingly being used in classroom settings to teach a wide variety of academic subjects, such as mathematics (Chorianopoulos et al., 2014; Barbieri et al., 2021; Fraga-Varela et al., 2021), languages (Sørensen and Meyer, 2007; Alyaz et al., 2017), and

computer programming (Miljanovic and Bradbury, 2018; Yallihep and Kutlu, 2020). Numerous systematic reviews further highlight the wide application of these tools in educational contexts (Verschaffel et al., 2019; Miljanovic and Bradbury, 2018).

Outside traditional education, serious games are frequently used in high-stakes fields where real-world practice may involve significant risk, such as medicine, therapy and rehabilitation, military training, or aerospace. In medicine, for example, games have been designed for various purposes (Gorbanev et al., 2018), such as improving technical skills in medical students (Olgers et al., 2021), teaching clinical reasoning (Chon et al., 2019), or providing trainees with surgical simulators as safe practice environments prior to performing real interventions (Carr et al., 2024). Applications in therapy and rehabilitation include virtual reality games for individuals with mobility impairments learning to use wheelchairs (Gerling et al., 2020), games to support social interaction in people on the autism spectrum (Gorbanev et al., 2018), or children with chronic diseases (Holtz et al., 2018). In the aerospace sector, games have been explored as flight simulators for professional training (Nisansala et al., 2015), in a similar fashion to the surgical ones. Meanwhile, military applications make extensive use of serious games in e-learning for simulating military environments in training and educational systems (Samčović, 2018). Balancing playability with educational or professional value often introduces additional design challenges, especially when addressing highly specialized or sensitive domains.

Another expanding application area for serious games—where educational relevance, interpretative transparency, and factual accuracy become paramount—is the cultural sector. This includes not only games focused on cultural heritage and historical reconstruction but also those developed in collaboration with museums and cultural institutions to engage visitors with fields such as natural sciences, anthropology, or environmental education (Anderson et al., 2009; Mortara et al., 2014; Albadawi, 2021; Kara, 2021). These environments apply serious games as a tool to stimulate curiosity, enhance learning, and provide a more accessible representation of complex topics via interactivity and discovery (Rowe et al., 2020). Such initiatives often require close collaboration between designers, educators, curators, and domain experts to ensure the content is both accurate and pedagogically effective (Anderson et al., 2009). Leaning on game mechanics and storytelling, these tools aim to foster deeper understanding and emotional connection with users.

2.3.2 State of the art in gaming applications for cultural heritage

Mobile technology has become a critical component of interactive initiatives for cultural sites, integrating effectively with other approaches, such as the

physical objects and artefacts found in museums and exhibitions (Koutsabasis, 2017). Additionally, there is a growing emphasis on "gamification" as a method to increase the appeal of cultural material to learners, as showcased in recent literature reviews (Mortara et al., 2014; Paliokas and Sylaiou, 2016; Malegiannaki and Daradoumis, 2017; Khan et al., 2020; DaCosta and Kinsell, 2022).

The widespread adoption of AR and VR technology is noteworthy in this regard. According to Kahn et al. (Khan et al., 2020), 46.66% of the 45 publications assessing gamification strategies between 2015 and 2020 propose the use of AR or VR components to explore cultural heritage. Despite often being regarded as more captivating, serious games pose an array of difficulties for creators in that they are required to reach a careful balance between enjoyment and education (Camps-Ortueta et al., 2019). While several general guidelines and heuristics exist for educators, researchers, instructional designers, and developers (DaCosta and Kinsell, 2022), crafting games that guarantee meaningful learning and present a genuine and respectful image of culture remains a challenging problem (Mortara et al., 2014).

The notion of design patterns was first presented by Christopher Alexander in the context of architecture (Alexander, 1977) and later adopted in the field of computer science by Gamma et al. (1994). Formalized in the game design domain by Björk and Holopainen (2005) in the mid-2000s, design patterns have recently found applications in AR-driven experiences for museums and exhibitions (Rau et al., 2022). Namely, Rau et al. (2022) focus on patterns that contain knowledge and functions for interactions like spatial connections or general overviews. These patterns enable content creators to focus on templates that learners want to adapt for particular exhibits rather than on devising elaborate templates from scratch. This strategy is consistent with our methods at EnigMachine, where we design educational escape rooms using escape patterns and templates.

Based on the treasure hunt concept, escape rooms target players wishing to seek clues, solve puzzles, and complete certain tasks within a particular period of time, usually with the goal of escaping the room (Nicholson, 2015; Wiemker et al., 2015). In the recent past, this genre of entertainment has gained immense popularity in major cities and has begun to gain traction in academia (Fotaris and Mastoras, 2019). Furthermore, the use of escape rooms for educational purposes has become increasingly popular across different levels of education, including higher education institutions (Gonzalez-Calero et al., 2024). When properly designed, escape rooms can provide an engaging environment that motivates learners to complete the tasks at hand (Clarke et al., 2017) and also can provide a way to facilitate learning through failure (Rawlinson and Whitton, 2024). EnigMachine has also adopted this format, progressively moving the focus away from simple treasure hunts to allowing for the design of rich, fully fledged educational escape

rooms where learners solve tasks in more holistic contexts.

2.3.3 State of the art in authoring tools for interactive experiences

Despite the potential of these technologies and concepts, authoring remains a significant challenge in the medium to this day (Green et al., 2021; Hargood and Green, 2022). The technical skills required to develop games for both cultural heritage and educational institutions can act as a major barrier to entry, preventing creatives or educators from accessing the field, while the production process can prove prohibitively expensive for institutions that do not possess the financial resources for it. The accessibility of the medium can thus be considered a major challenge for this field, especially given the already existing concerns and difficulties in the adoption of new technologies, both in the cultural heritage (Jones et al., 2019; Green et al., 2021) and educational domains (Garzón-Artacho et al., 2021).

The collective solution adopted by the research community appears to be focused on the development of authoring tools that enable users such as educators or curators, who are generally non-technical, to design interactive activities and narratives and translate them onto digital platforms. These efforts fall within the research line of what is known as end-user development, i.e., the creation of authoring tools aimed at professionals who may not be technically savvy enough to operate complex systems during the content authoring process (Lieberman et al., 2006). Included here are tools geared towards the creation of augmented reality mobile applications and interactive stories in cultural heritage (Fidas et al., 2015; Sintoris et al., 2014), multi-platform guides and experiences for museums (Ghiani et al., 2009; Linaza et al., 2008; Roussou et al., 2015), or narratives with GPS-based interaction mechanisms in urban environments (Hansen et al., 2012).

Recent work about educational escape rooms (Sánchez, 2023; Wolf et al., 2024) mention the use of commercial authoring tools for creating digital educational escape rooms (DEER), which use digital environments and may be played by participants not present on site, such as Genially (Genially Web S.L., 2025), Telescape Live (Buzzshot, 2025), Breakout EDU (Schaffhauser, 2017), Room Escape Maker (ROOM ESCAPE MAKER - INPI, 2025), and even open-source platforms such as Escapp (Grupo de Internet de Nueva Generación, 2025). Nevertheless, at the time of writing, we are not aware of any authoring tool specifically designed for creating physical educational escape rooms such as our EnigMachine ecosystem.

These efforts do, however, demand careful attention to interface design and user experience across individual authoring tools. While numerous works and heuristics exist that focus on deriving design principles and evaluations for the games or interactive experiences themselves (Desurvire and Wiberg, 2008, 2009; Korhonen, 2016; Andreoli et al., 2017; Raptis et al.,

2019; Pujol-Tost, 2019), whether for commercial games, serious games, or other educational or cultural heritage activities, a comparatively small fraction of this work addresses a similar analysis of the actual authoring tools. A number of state-of-the-art reviews on the issue explicitly highlight this limitation (Bekele et al., 2018; Konstantakis and Caridakis, 2020), stressing how the current spotlight lies primarily on the audience space as opposed to that of the authors, resulting in a landscape where few guidelines or studies exist examining what constitutes an effective end-user authoring tool. This scarcity could potentially stem from the complex nature of this kind of analysis, which often requires long experimental user sessions in order to meaningfully probe the characteristics of the system (Kuniavsky et al., 2012).

Chapter 3

AI Techniques for Automatic QA and Player Modeling

"But then suppose you stepped into one of those rooms," he said, "and discovered another room within it. And inside that room, another room still. Rooms within rooms within rooms. Isn't that how it might be, trying to learn Josie's heart? No matter how long you wandered through those rooms, wouldn't there always be others you'd not yet entered?"

Kazuo Ishiguro, Klara and the Sun

Video games are increasingly complex and ambitious software products in terms of their scope and the number of elements or assets involved in their creation. In this context, and as described in Sections 2.2.1 and 2.2.2, there exist a number of quality control tasks related to gameplay validation. This involves ensuring that the way players interact with the game remains consistent with the designer's preconceived idea of how each challenge should be tackled. Play-testing is a common method of assessing game quality, usually requiring a human tester to answer questions about the degree of difficulty and perceived enjoyment during play, the presence of bugs and glitches, or the feasibility of successfully completing various tasks enforced by the design. The problem with this approach to quality testing is that as levels, shared assets between game scenes, non-playable character AIs (NPCs), or functionalities are created or modified throughout development, the answers to these questions may change unexpectedly in an increasing number of scenarios. For instance, modifying the configuration of a shared enemy AI could alter the difficulty of a level, prevent it from being solved using the originally designed strategy, or even enable unexpected new solutions.

The tests involved in the task of determining whether a feature or a design that was correct in the past continues to work properly after some progress in the development of the project are known as *regression tests*. Their cost of execution grows dramatically as the volume of the code base and elements included in the game increases, as they essentially entail the recurring repetition of previously executed test batteries or checks. These tests may raise functional questions ("does the enemy continue to approach the player when it encounters them at a distance below a certain threshold?"), or more abstract questions linked to the game design ("is it possible to complete the level in less than 10 minutes and without losing health points?" or "is it possible to traverse the level while picking up all the collectibles and without being detected by enemies in the process?", to name a few examples). While the first types of questions can often be addressed through the use of tools such as unit tests, for which most commercial engines offer good support, this is typically not the case for the second type of problems, which generally require humans playing the game repeatedly trying to figure out an answer to the question posed by the test.

In order to maintain sufficiently thorough coverage of the range of scenarios and checks to be performed within the game, design validations should be performed as frequently as possible. This is not logistically feasible if these tasks are reserved for human testers, as they require an unsustainable effort in terms of time and money. By design validations, we will narrow down here to the type of test that receives a specification, typically given by a series of steps or instructions to be performed sequentially by the tester, and reports whether these are feasible in the environment in which they are proposed, together with comments on what makes them invalid in the case where they are deemed non-viable. Having an automated specification validation methodology in the development process does not replace the human tester but allows redirecting their efforts to verify only those sections of the project where it is truly suspected that unexpected changes have occurred, thus increasing efficiency and coverage in QA.

In this chapter, we will focus on defining and expanding on the different components comprising an automated and general game testing methodology based on machine-learning agents. We will start with a description of the general framework for automatic testing of game design specifications in Section 3.1, focusing on introducing the main building blocks of the methodology, its suggested workflow, and the different components involved in the process. We will then expand on each of these components in Sections 3.2, and 3.3.

3.1 A workflow for automatic testing of game design specifications

Following the notions of testing and QA in video games that we introduced in Section 2.2.1, to be able to replicate the process of a QA practitioner in a commercial game under development, it is vital that we are able to address the following fundamental steps:

1. **Definition of the Design Specification**: First, it is necessary to have a design specification that clearly defines the requirements and objectives of the level or section of the game to be evaluated. While the nature of this specification is deliberately vague and open to different definitions depending on the type of evaluation to be performed, in any case it will be essential to have, on the one hand, a formal description language that allows expressing design requirements precisely and, on the other hand, a method that allows us to compute an approximation of whether the specification in question is satisfied or not in our environment. Since in most cases these requirements must be expressed by the game designer, it is important that the chosen language is accessible and understandable to them, ideally through a high-level language as close as possible to natural language that does not require advanced technical knowledge, and that is integrated into the game engine used for project development by means of some type of tool or plugin.

2. **Training of Automated Agents**: Once a design specification is available, the next step is to train an automated agent capable of interacting with the game environment and evaluating whether the specification is met. This can be done in several ways. The first consists of resorting to agents built using manual scripting techniques where the programmer imperatively invokes different actions and commands on the bot to achieve its goals. These can be robust with a sufficient investment of resources, but generally end up being costly as requirements grow in complexity. The second approach considers the use of machine learning techniques, which allow building agents capable of declaratively learning to play the game; that is, instead of imperatively launching actions as in the previous case, training bots that build a valid policy solely based on the described objectives.

3. **Test Execution**: Once an automated agent has been trained, the next step is to execute a series of tests in the game environment to evaluate whether the design specification continues to be respected. This involves executing the agent in the game environment and collecting various metrics on its performance, as well as reports justifying the observed behavior. Additionally, it is important to have a mechanism

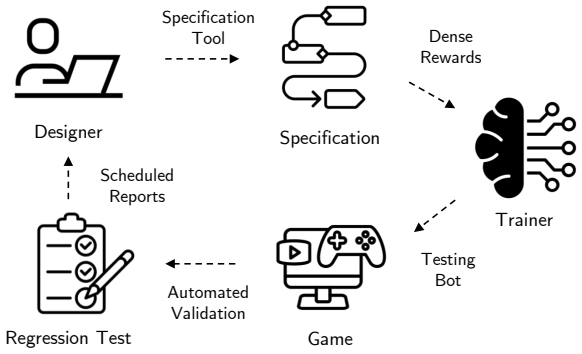


Figure 3.1: Sample diagram for this section's workflow for RL agents: the designer provides a plan on how to interact with the level using some sort of specification tool, from which a dense reward function is generated to train a bot to periodically validate the feasibility of the solution throughout development.

> that allows comparing the results obtained in the execution of the tests with the expected results, so that it can be determined whether the design specification has experienced significant variations compared to a previous checkpoint. Since tests are generally not necessarily deterministic (neither in the case of automated agents nor with human testers), executions must be repeated a sufficient number of times to obtain a good approximation of the distribution of results.

The connection flow of these components in the testing process is illustrated in the diagram in Figure 3.1.

While not directly linked as a paper of this thesis, we first introduced a draft of this workflow in (Gutiérrez-Sánchez et al., 2021) (published in the *Conference on Artificial Intelligence and Interactive Digital Entertainment 2021*) as a general approach to automatic testing of game design specifications. This preliminary paper presented the application of reinforcement learning together with behavior trees to automatically test if modifications to the AIs of a stealth game have an impact on the user experience, and focused on establishing first approach for steps 2 and 3 above, thereby laying the groundwork for the rest of the contributions of this chapter.

Before going on to describe each of these steps in detail, however, a key point to address is the environment in which the testing process will take place. In this regard, it is important to have a testbed that resembles as much as possible the development environments of modern video game teams, so that the proposed workflow can be truly useful and applicable to the reality of the studios. In the following section, we contextualize the current situation of existing game testbeds and present our own, LIQUID SNAKE, expressly designed to facilitate the implementation of automatic testing techniques for

the rest of the chapter, and to serve as an architectural reference for its integration into commercial projects.

3.2 Liquid Snake: a testbed for AI-based QA

The landscape of game testbeds for AI research has seen significant contributions from various organizations. For instance, OpenAI has been a major player in disseminating Reinforcement Learning (RL), developing testing environments for RL-driven agents, initially known as Gym (Brockman et al., 2016) and now rebranded as Gymnasium [1]. This platform enables AI practitioners to test different machine learning algorithms across a large collection of sample environments, ranging from classic Atari games to fundamental problems like Cart Pole, Pendulum, and Mountain Car.

Gymnasium and its API have become a benchmark for evaluating and comparing various RL algorithms. Beyond its built-in environments, numerous third-party implementations based on Gymnasium exist, such as *Flappy Bird* [2], *Slime Volleyball* Gym Environment [3], and stable-retro [4]. These include different renditions of classic games like *Super Mario Bros 2 Japan* (*Lost Levels* in the West) from the Nintendo Entertainment System (NES), *Punch Out* (NES), *Tetris* (Game Boy), *Virtua Fighter* (32X), or *Mortal Kombat* (Genesis), among others.

However, most of these implementations involve relatively small games (like *Slime Volley* or *Flappy Bird*) or rely on emulators for titles developed for older consoles, as seen with stable-retro. This framework is therefore suitable for training and evaluating agents engaging with a selection of simple games, but it falls short when assessing the ability of RL-based strategies and other machine learning and AI techniques to be valuable in other major areas of the video game industry. This includes controlling the behavior of non-playable characters (NPCs) or performing automated testing tasks. The limitations stem from the lack of complexity or closeness to modern games in terms of mechanics and scale, as well as the inability to access and modify the source code of these environments.

One of the most successful implementations compatible with the Gymnasium API, which partially addresses the limitations of other Gymnasium test environments, is ML-Agents (Juliani et al., 2020). This tool allows developers to create games built on Unity3D and use them as test environments for RL and Imitation Learning (IL) algorithms. The ML-Agents Toolkit has been used in several research works (Almeida et al., 2024; Majumder and Majumder, 2021), inspiring the creation of challenges and agent

[1] https://gymnasium.farama.org/
[2] https://github.com/markub3327/flappy-bird-gymnasium
[3] https://github.com/hardmaru/slimevolleygym
[4] https://github.com/Farama-Foundation/stable-retro

creation competitions in complex environments such as Obstacle Tower (Juliani et al., 2019), as well as in commercial games like *Source of Madness* [5], where ML-Agents was used to generate enemy behaviors.

At the core of ML-Agents is a comprehensive machine learning package specifically designed for Unity environments. This package includes implementations of state-of-the-art reinforcement learning algorithms, such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Soft Actor-Critic (SAC) (Haarnoja et al., 2018), which are widely recognized in the research community for their efficiency, stability, and ability to handle continuous action spaces. For imitation learning, ML-Agents offers Behavioral Cloning (Bain and Sammut, 1995), where agents learn a task by directly replicating an expert's actions, treating the expert's demonstrations as a supervised learning dataset of (state, action) pairs, and Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016), a model-free imitation learning algorithm that learns to mimic expert behaviors by treating imitation as a distribution matching problem between recorded demonstrations and agent responses.

Beyond these core algorithms, the ML package supports several advanced extensions to enhance agent training. These include intrinsic motivation mechanisms, such as Random Network Distillation (RND) (Burda et al., 2018), which encourages exploration in sparse-reward environments, and curriculum learning frameworks, which progressively increase task difficulty or variety to accelerate learning. The toolkit also provides support for dynamic neural architectures, such as Hyper Networks (Chauhan et al., 2023), which can modulate agent policies based on external signals, and transformer-based architectures (Vaswani et al., 2023), which allow agents to process variable-length observations efficiently and capture long-range dependencies in sequential tasks.

Although this toolkit offers many virtues, it does come with certain limitations. Just like Gymnasium, it is closely aligned with the use of RL, as well as specific Python versions and frameworks (Keras or Pytorch). Despite the recent incorporation of Trainer Plugins to support custom implementations of the training interfaces provided by ML-Agents, these are still very limited, making it difficult to test other algorithms beyond those included in the tool.

While these standardized testbeds and benchmarks are valuable for guiding research efforts and testing new algorithms, their direct application to development cycles within commercial studios faces significant hurdles. The reality is that the implementation of these methodologies in commercial game development currently suffers from several integration problems that discourage developers from employing them in their projects. Firstly, generating autonomous agents capable of naturally playing a given game is complex and

[5] `https://sourceofmadness.com/`

requires non-trivial knowledge in machine learning. While new tools like the abovementioned ML-Agents in Unity3D or MindMaker [6] in Unreal Engine help by offering more developer-friendly ways to train policies, these are mostly aimed at academic research or creating small-scale proofs-of-concept. Secondly, to our knowledge, there is a scarcity of accessible benchmarks and environments that truly serve as showcases and examples for automatic testing techniques in games undergoing continuous development. While many existing platforms — such as OpenAI Gym, DeepMind Lab (Beattie et al., 2016), MineRL (Guss et al., 2021), and the StarCraft II Learning Environment (Vinyals et al., 2019) — are designed to evaluate machine learning algorithms, they are all based on closed, fully developed games with fixed objectives or performance metrics. This contrasts with the problem that concerns us: starting from a game in open development and using agents to perform regression tests as it is continually modified.

It is therefore relevant, from our point of view, to propose a test-bed oriented to serve as a testing framework not only for machine learning algorithms, but also for their applications on quality control strategies in commercial games and the development of support tools for automated testing. Having a common environment also enables a shared vocabulary in the community: for instance, it becomes possible to compare the results of two ways of approaching a regression testing problem on specific and shareable modifications (such as "does my method detect that it is more difficult to evade level 2 enemies after altering a node of its behavior tree?").

On the other hand, from the developers' point of view, having a simple reference environment where they can try out automatic testing mechanisms allows them to experiment with different strategies in a smaller external project and validate them before taking the step of integrating them into their own games. At the same time, this test-bed can be used as an architectural reference for those teams that are considering undertaking automatic testing but are held back by the complexities and technical unknowns associated with the problem.

With this, here we present Liquid Snake - a third-person 3D prototype belonging to the stealth genre and developed in Unity3D intended to act as a common test-bed for regression testing and automatic quality control, as well as an architectural reference for the integration of such methods in commercial projects. Originally introduced in a preliminary version in Gutiérrez-Sánchez et al. (2022) and later expanded on in Sagredo-Olivenza et al. (2024), Liquid Snake tasks the player with guiding the main character through a series of rooms and locations in search of the level's exit and solving various puzzles, all while avoiding detection by robotic guards patrolling the area, security cameras and other surveillance mechanisms. Figure 3.2 provides a screenshot of the game's appearance in a sample level.

[6] `https://github.com/krumiaa/MindMaker`

Figure 3.2: Top-down capture of a level section from LIQUID SNAKE.

Enemies in this game exhibit patrol-based behavior, where they move sequentially across a predefined set of points established by the level designer and pause at each location to simulate scouting the environment—a mechanic frequently used in stealth games to structure player-enemy interactions. If a guard detects the player within its cone of vision (reflected in Figure 3.2 using a pink robotic enemy), they will visit the spot where they last saw them and look around for a short time. As long as the guard keeps noticing the player, or senses them in their vicinity, a suspicion meter will gradually build up. If the suspicion meter reaches a maximum level, the guard will enter an aggressive mode and chase the protagonist until they are captured. The player can use the environment to hide, crouch behind mid-height objects to avoid detection by vision cones, and run to quickly dash through sections of the level before the enemies' suspicion gauges can grow too high. Certain levels also require the player to momentarily allow themselves to be spotted by the robot guards and draw their attention to a secluded spot, forcing them to inspect that location while the protagonist advances through the areas the guards have left unprotected in the process.

In turn, the player has a number of movement options and actions to interact with the environment. Apart from basic movement on the XZ axis by walking or running, the character can jump to overcome low obstacles or reach high platforms that would otherwise be inaccessible, as well as assume a crouched position to hide from enemies behind mid-height obstacles or maneuver under low-ceiling sections. Additionally, the character can interact with certain objects in the scene by approaching their vicinity (as in the vending machine in the figure surrounded by a circular yellow area), triggering different effects on the environment. One of the most typical examples of these interactions is the presence of triggers that enable or disable laser

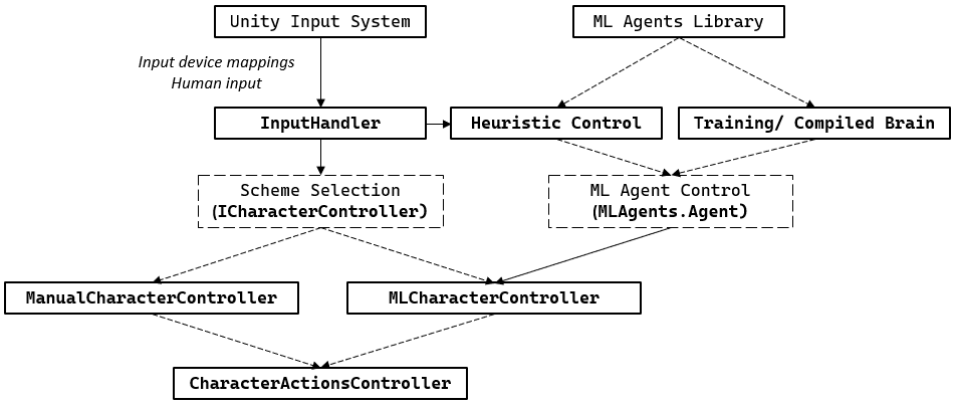


Figure 3.3: Control scheme in Liquid Snake.

barriers such as those on the right side of the figure, which serve to block areas.

Architecturally, one of the main bottlenecks when integrating a machine learning model with a test environment is the difficulty to conveniently switch between control schemes. The most classic examples of this situation are found in the need to alternate between manipulating the character manually, using a policy trained by reinforcement learning, executing traces prerecorded by human demonstrators, or applying hand-scripted routines to specify desired behaviors. Liquid Snake is designed to allow straightforward switching between control systems according to the needs of each use case. To this end, the character object exhibits a series of methods that enable the different actions described above to be invoked, which are then called from different controllers depending on the situation. If the intention is for a human to play in a traditional way, the input is provided by Unity's input system, mapping each keyboard or controller entry with one of the available actions; on the other hand, when the intention is for an AI controller to manipulate the character, a series of specific actuators are provided for each controller type to translate the intentionality of the agent and model used into calls to the corresponding actions. Thus, the decision of how an agent's output is converted into a character's behavior is entirely in the hands of the implementer. The proposed scheme is summarized in figure 3.3, for the case in which it is intended to alternate between manual control and control through an ML Agents library agent. This facilitates working with a wide variety of models such as neural network-based systems, rule-based approaches, behavior trees or planners, to name a few, each of which can choose to define actions with different degrees of granularity (discrete, continuous, high- or low-level).

Currently, Liquid Snake comes integrated with the Unity ML-Agents library, described earlier, which provides the necessary support to start test-

ing reinforcement and imitation learning methodologies on our environment. In addition, ML-Agents provides its own wrapper on top of the typical Gymnasium environment interfaces, allowing interaction with environments that make use of the tool through an API well known to developers familiar with Gymnasium. Additionally, we include the Behavior Bricks tool[7] for programming and designing the behavior of both NPCs and gameplay bots through Behavior Trees, as well as Unity's native support for state machine programming through Visual Scripting. Also available as an optional module for this approach is our behavior generation tool via reinforcement learning and formal task specifications, AI Behavior Graphs, which will be introduced in detail in Section 3.4.

Once a trained model is available using any of the resources listed above, it is possible to perform simulations on the level of interest while collecting different customizable metrics related to the performance of the agents. At the moment, the project provides a set of Unity play-mode tests that can be used as a guideline to load scenes automatically, instantiate a standalone controller, associate it to the character object and then run the corresponding scene for a fixed number of times collecting event-driven metrics. This allows to gather simulation metrics from a fair number of levels in a convenient way, avoiding manual switches and executions.

These features make this an attractive testing and experimentation environment not only from the viewpoint of game AI practitioners, but also from a pedagogical perspective, allowing students of AI or game programming disciplines to apply the skills they have learned in scenarios close to real games.

As mentioned at the beginning of the section, Paper 1 (published in *Actas del II Congreso Español de Videojuegos, 2024*) and Paper 2 (published in *Actas del I Congreso Español de Videojuegos, 2022*) detail the first two iterations of this environment in order to address Objective 1.2 of the thesis. These works set the stage for the contributions of this chapter, providing the necessary infrastructure to implement our proposed workflows for automatic testing of game design specifications.

3.3 RL with temporal logic specifications for regression testing

Once a suitable testing environment is established and we have the capability to generate automated agents that can interact with it, we can begin to tackle the challenge of creating these agents for validating design specifications. As discussed in Section 3.1, this process starts with defining a design specification that allows us to express the requirements and objectives of the

[7] `https://bb.padaonegames.com/`

game level or section being evaluated. This highlights the need for a formal description language capable of precisely and accessibly articulating design requirements for the designer, along with a method to compute an approximation of whether the given specification is met within our environment.

A promising initial approach to this problem involves using reinforcement learning (RL) techniques to generate agents that learn to fulfill a design specification. Reinforcement learning is one of the three core machine learning paradigms, focusing on how intelligent agents should interact with their environment to maximize some form of reward. In RL, an agent takes actions within its environment, which in turn provides feedback and "rewards" through positive or negative reinforcement. RL algorithms strive to create agent behaviors that maximize this cumulative reward. They typically adopt the formulation and formalism of Markov Decision Processes (MDPs) as a starting point.

If we consider $s_t \in S$ as the state of the system at a given instant t, and $a_t \in A_{s_t}$ as the action that the agent executes at that instant (where A_{s_t} is a potentially infinite set of admissible actions for the agent in state s_t), the agent's goal is to learn a mapping between states and actions, known as a *policy* $\pi : S \to A = \cup_{s \in S} A_s$. This policy aims to maximize the expected long-term total reward from each state s:

$$G_t = E[\sum_{i=0}^{\infty} \gamma^i R_i]. \tag{3.1}$$

Here, $\gamma \in (0, 1)$ is a *discount factor* that prioritizes rewards earned in the short term over those collected in the distant future, and R_i represents the reward the agent receives at the i-th instant.

3.3.1 Preliminaries: the reward shaping problem

Within this paradigm, the reward function serves as the fundamental input for defining the learning task. It is responsible for translating the agent's behavior into a numerical signal that the chosen RL algorithm will attempt to maximize during training. This immediate reward signal, denoted as R_t, is the concrete value the agent receives at a particular time step, emphasizing its role in the sequential experience the agent optimizes. While R_t is the observed reward, its underlying generation can be formalized in various ways: as $R(s)$ (dependent solely on the state), $R(s, a)$ (dependent on the state and action taken), or $R(s, a, s')$ (dependent on the state, action, and resulting next state). While the reward function offers a declarative way to encode tasks, eliminating the need to manually implement behaviors or even explain how an agent should act beyond indicating desirable and undesirable outcomes, the reality is that defining an appropriate reward function for the problem at hand can be a complex and costly process.

For instance, in many RL tasks, especially within game environments, the agent receives a reward signal only at the very end of a long sequence of actions or upon achieving a final objective. This is an example of what is known as *sparse rewards*. The fundamental issue here is the difficulty the agent faces in attributing credit or blame to individual actions that occurred much earlier in the sequence, making learning excruciatingly slow and inefficient (Sutton and Barto, 2018).

As an example, consider training an RL agent to play a game like *Pong*, where the goal is to win the game by being the first to reach a certain score (e.g., 11 points). The primary reward in this case could be extremely sparse, for instance +1 for winning the entire game, and 0 otherwise. In this extreme sparse reward setup, the agent would face significant hurdles:

- **Excessively Slow Learning:** The agent might execute thousands or even millions of random actions before accidentally winning a point or, even more rarely, an entire game. With such infrequent positive reinforcement, the learning algorithm struggles to establish a meaningful connection between specific paddle movements and the distant positive reward.

- **Severe Credit Assignment Problem:** When a rare win occurs, it becomes nearly impossible for the agent to discern which of the myriad previous actions contributed to that success (i.e. "Was it the last paddle movement, an earlier defensive block, or a perfectly angled shot?"). The credit assignment problem is a cornerstone challenge in RL, and sparse rewards significantly exacerbate it (Sutton and Barto, 2018).

- **Exploration Challenges and Local Optima:** Without intermediate rewards, the agent might never discover effective strategies. For instance, in *Pong*, an agent might learn to simply keep its paddle stationary in the center, as any random movement initially yields no immediate benefit. It could get stuck in this suboptimal local optimum, never exploring movements that could lead to hitting the ball and eventually scoring.

Beyond the sparsity issue, even when rewards are given more frequently, designing a reward function that perfectly encapsulates the desired behavior for complex tasks is incredibly challenging. Hand-crafting such a function is prone to errors, often leading to unintended consequences or requiring an unreasonable amount of human effort. The core of the "reward shaping problem," as formally explored by Ng et al. (1999), is how to provide effective intermediate guidance without altering the optimal policy of the original task.

As a second game example, consider the problem of training an RL agent to autonomously drive a car in a complex, open-world environment such as *GTA V*. The primary task is to navigate from Point A to Point B while adhering to traffic laws and ensuring safety. The primary reward could be a simple +100 for reaching the destination, but defining a suitable shaping reward function for this task quickly becomes daunting due to the multitude of factors involved. The agent must learn to balance several objectives, including:

- Quantifying "Safety" (i.e. "How do we numerically reward nuanced safe driving?"). While penalties for collisions (e.g., -1000 for a major crash, -50 for a fender bender) are relatively straightforward, rewarding preventive safety is not particulary trivial. Assigning precise numerical values to behaviors like maintaining a "safe following distance," anticipating hazards, or smoothly handling curves can become highly subjective and difficult to implement consistently.

- Balancing "Efficiency" with Other Objectives. If the practitioner wishes to enforce swift completion of the task at hand, it is common to introduce continuous "existential penalty" signals to encourage speed (e.g., -0.1 per second). By doing so, the intention is to drive the agent to reach the desired goal quickly to prevent the accumulation of negative reward, but this can conflict with safety and legality in different scenarios. If speed is over-rewarded, the agent might learn to drive recklessly, ignore traffic lights, and dangerously weave through traffic to minimize travel time. In some extreme cases, if the agent is unable to find sources of positive reward, it might eventually try to find ways of prematurely ending the episode to avoid the negative penalty, such as crashing into a wall or driving off a cliff, which would be counterproductive to the task. Conversely, overly penalizing speed might lead to an agent that drives excessively slowly, never reaching the destination efficiently.

All of these unintended behaviors lead to what is commonly known as the *reward hacking problem* (Amodei et al., 2016). Agents, being powerful optimizers, will always take the path of least resistance to maximize their cumulative reward, even if that choice is not aligned with what the human designer intended. If there is a heavy penalty for crashing, for instance, the agent may learn to simply stop the car immediately and never move, effectively "solving" the collision avoidance problem but failing the primary task of reaching the destination. If "staying on the road" is rewarded, but sidewalks are penalized, a poorly designed function might inadvertently encourage the agent to swerve into oncoming traffic to avoid crossing over a minor sidewalk encroachment, as the penalty for a collision may be delayed

or not as severe as a sidewalk penalty. In complicated tasks like driving, there may be multiple complimentary or conflicting objectives: safety, efficiency, legality, and even passenger comfort. Assigning appropriate relative weights for the different properties of a reward that would ultimately be contained under a single scalar reward function can be notoriously difficult, typically involving a process of trial and error and fine-tuning of function parameters.

Lastly, sequential and conditional tasks add further complexity to reward function design. These tasks require the agent to complete a set of actions in a specified order, or to react to certain conditions before proceeding. The reward function must not only guide the agent through the correct sequence but also manage dependencies correctly, prevent the agent from skipping essential steps, and avoid rewarding incomplete or irrelevant actions. A typical literature example comes from training RL agents to perform complex crafting tasks in games like *Minecraft* or *Rust*, where the agent must collect resources, craft items, and follow specific sequences of objectives to achieve a given goal. In such cases, the reward function must ensure that the agent gathers the necessary materials in the proper order, crafts intermediate items before the final product, and avoids behavioral detours that do not contribute to the task at hand.

An example of this is the 2019 MineRL competition (Guss et al., 2021), which challenged participants to create agents that could play *Minecraft* and eventually obtain a diamond, an infamously complex task requiring long-term planning, hierarchical control and efficient exploration methods to gather prerequisite resources, craft tools, and navigate the game's world. A naive first approach to tackle this task would be to grant simple, fixed positive rewards for each completed sub-step (e.g., +10 for collecting wood, +10 for crafting planks, etc.), but this comes with several issues. First, the agent might learn to endlessly collect wood, maximizing the easier "collect wood" reward, even after it has gathered more than enough for the task. Consequently, it could get stuck in this local optimum, never progressing to the later crafting steps. Second, the agent might ignore crafting dependencies and start collecting stone even before it has crafted sticks. That is, if the reward for "collecting Stone" is purely additive, the agent has no incentive to adhere to the required sequence of dependencies (sticks are needed before the final pickaxe can be crafted). Lastly, the agent might incur in suboptimal resource management, crafting excessive amounts of planks or sticks to maximize the reward for "crafting planks," even if only a single set is needed for the final pickaxe. This leads to inefficient resource usage that is not penalized by the reward function.

A more robust approach, rooted in the theory by Ng et al. (Ng et al., 1999), is to use potential-based reward shaping, where the shaping reward $F(s, a, s') = \gamma\Phi(s') - \Phi(s)$. Here, $\Phi(s)$ is a "potential function" that assigns a value to the current state representing its "progress" towards the goal. For in-

stance, $\Phi(s)$ could increase as the agent completes sequential sub-goals (e.g., higher potential after wood collected, even higher after planks crafted, etc.). Unfortunately, defining an accurate and comprehensive potential function $\Phi(s)$ for a complex crafting system with multiple recipes, intermediate products, and inventory management becomes incredibly difficult, and human error in designing $\Phi(s)$ can still lead to suboptimal policies or unintended behaviors.

In order to design a reward signal that effectively guides the agent towards the desired behavior without introducing unintended side effects or demanding prohibitive engineering effort, in the following section we will introduce a method that eliminates the need to specify a manual reward function in the reinforcement learning loop by automatically constructing a dense function from a staged representation of what is expected to be validated in the level.

3.3.2 Temporal logics for RL

As described in the previous section, one of the key disadvantages of RL is that the user must manually translate the requirements of the task to be performed by the agent into a numerical reward function, which often requires significant mathematical background, considerable expertise in machine learning and domain knowledge of the target environment, making these strategies inaccessible to the general user. For complex problems, and especially for those involving temporal dependencies (goals that must be fulfilled in a certain order, for example), these functions can be very challenging to design; furthermore, the user is typically confronted with the need to manually shape the rewards to guide the agent during learning. Reward shaping can become very error-prone and is frequently accompanied by tedious trial-and-error iterations to produce reward functions that induce the expected behavior. It is in this context that temporal logics (TLs) have attracted increasing interest in the last decade as a means of supporting the generation of reward functions in RL frameworks from requirements formally modeled in these languages, particularly in the field of robotics (Liao, 2020).

By TLs we refer to any system of rules and symbols that allows reasoning about propositions in terms of their evolution over time. TLs play an important role in the formal verification of requirements in both hardware and software (Buzhinsky, 2019), with Linear Temporal Logic (LTL) (Li et al., 2017) being one of the most widely adopted examples of these logics. LTL formulas are defined based on a series of core logical and temporal operators and, more importantly, predicates of the form $f(s) > 0$, where $f : \mathcal{S} \to \mathbb{R}$ represents a function applied to the system's state $s \in \mathcal{S}$. The latter are the ones that are truly responsible for incorporating a certain knowledge base into our specifications, as well as conditions related to the world and context in which we operate, such as "being close to a point" or "perceiving

a high temperature." These conditions are highly domain-dependent and consequently, for each game we contemplate, we will inevitably encounter the need to define a specific set of predicates that will determine the expressiveness of our tasks. Having said this, an LTL specification $\phi \in \Phi$ adheres to the following syntax:

$$\begin{gathered}\phi := \top \mid f(s) > 0 \mid \neg\phi \mid \phi_A \wedge \phi_B \mid \phi_A \vee \phi_B \mid \\ \phi_A \Rightarrow \phi_B \mid \Diamond\phi \mid \Box\phi \mid \phi_A \,\mathcal{U}\, \phi_B \mid \bigcirc \phi\end{gathered} \tag{3.2}$$

Here, $\top$ is the True boolean constant, $f(s) > 0$ is a check on a domain function on the system state, $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction) and $\Rightarrow$ (implication) are Boolean connectives, and $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), and $\bigcirc$ (next) are temporal operators.

From a design perspective, $\Diamond$ indicates a requirement that must be met at some point during the game, and is usually referred to as a liveliness condition, whereas $\bigcirc$ mandates that the associated condition be met in the very next simulation time step. For instance, if our predicate is $\bigcirc \phi$, we anticipate ϕ to be true in the second time step of the run. Typically, designers would need to use a combination of $\Diamond$ and $\bigcirc$ operators to ensure that the specified condition is met at a later time. This allows for sequences of conditions that must be satisfied one after the other by adding blocks of the form $\phi_A \wedge \bigcirc(\Diamond \phi_B)$. For global constraints that must be continuously in effect (which are often called safety conditions), the $\Box$ operator can be employed, followed by the condition we want to persist. This is especially useful for modeling level constraints, such as maintaining a specific item or preventing health points from falling below a certain threshold.

Among TLs, the focus has been mostly on those that allow the formulation of quantitative semantics, commonly known as robustness metrics. As opposed to qualitative semantics, which simply indicate whether or not a system trace satisfies a logical predicate, these robustness measures make it possible to provide a numerical assessment of how well a state trace $\mathcal{T} = (s_1, \ldots, s_n) \in \mathcal{S}^n$ adheres to the given specification through a robustness function $\rho : \mathcal{S}^n \times \Phi \to \mathbb{R}$, which assigns a real-valued number to each predicate ϕ from the predicate space Φ and system trace $\mathcal{T}$.

Once a quantitative semantics is available (and this need not be unique), the obvious strategy to guide learning in a RL setting on the basis of TL specifications is to assign a reward to the agent at the end of each training episode given according to the robustness of the trace generated during the episode, $\rho(\mathcal{T}, \phi)$ (Pant et al., 2017). This strategy suffers however from a problem of reward sparsity during training, as reinforcement can only be provided for a whole trajectory, hampering the learning of complex or long-lasting tasks in a significant way.

One general approach to alleviate this problem is provided by the notion

of *Reward Machines* (RMs) (Toro Icarte et al., 2022). RMs are a type of finite state automaton that encode reward functions over propositional abstractions of the environment. Instead of producing a single episodic reward, an RM progresses through its own finite set of states in parallel with the environment, outputting at each step either a numeric reward or a reward function. This makes it possible to specify temporally extended, non-Markovian objectives such as sequential goals, loops, or safety conditions, while still exposing a structured, automaton-based decomposition that the agent can exploit. Formally, given propositional symbols P, states S, and actions A, a reward machine is a tuple

$$\mathcal{R}_{P,S,A} = \langle U, u_0, F, \delta_u, \delta_r \rangle,$$

where U is a finite set of states with initial state u_0, F is a finite set of terminal states (with $U \cap F = \emptyset$), $\delta_u : U \times 2^P \to U \cup F$ is the transition function, and $\delta_r : U \to (S \times A \times S \to \mathbb{R})$ maps each machine state to a reward function. At each time step, the RM consumes the current truth assignment over P, updates its internal state according to δ_u, and outputs a reward given by δ_r.

In practice, simple reward machines are often used, in which the reward is provided directly as a number, $\delta_r : U \times 2^P \to \mathbb{R}$. An MDP augmented with an RM then yields an *MDPRM*, whose state space is the cross product $S \times U$ and whose rewards are Markovian with respect to this augmented state, making it possible to apply standard RL algorithms while retaining the temporal structure of the original objective.

In response to the same reward sparsity problem, Li et al. (2019) introduce the concept of *Finite State Predicate Automata* (FSPAs) to abstract away the temporal dependencies of the original predicate and provide dense rewards based on robustness variations between automaton transitions. Conceptually, FSPAs are closely related to RMs: both are finite state automata where transitions are guarded by logical predicates over environment observations. The distinction lies in their output: FSPAs attach robustness-based rewards to transitions, whereas RMs provide a more general framework in which rewards can be arbitrary functions of environment dynamics. In this sense, FSPAs can be viewed as a particular instantiation of the broader reward machine paradigm, where the predicates are derived from temporal logic robustness and the outputs are defined as robustness variations.

Thus, an FSPA can be thought of as a finite state automaton where transitions are associated with TL predicates, such that a transition occurs when the robustness of its linked predicate is both positive and maximal (if multiple edges meet this criterion). This allows the global problem to be converted into small sub-steps localized in states of the automaton where the progress conditions are well defined. As any given transition in FSPAs comes with an associated robustness variation, these can be used as a step-

based reward instead of the episodic one described earlier. Additionally, trap states are introduced in these FSPAs to represent task failure.

An additional advantage of this construction is that it eliminates the need to both define and compute the robustness of temporal operators in the predicates of our language, which is often a complex and costly process. In the case of the grammar from Equation 3.2, here we define the robustness function $\rho_s(\phi) := \rho(s, \phi)$ that outputs $\rho_s(\top) = 1$, $\rho_s(f(s) > 0) = f(s)$, $\rho_s(\neg\phi) = -\rho_s(\phi)$, $\rho_s(\phi_A \wedge \phi_B) = \min(\rho_s(\phi_A), \rho_s(\phi_B))$ and $\rho_s(\phi_A \vee \phi_B) = \max(\rho_s(\phi_A), \rho_s(\phi_B))$, aiming to keep the range of all the domain functions we define in the same interval $f(s) \in [-1, 1]$ so that they are comparable to each other and no conditions outweigh others in composite predicates. Note how here we drop any mention of traces in favor of single states (now $\rho : \mathcal{S} \times \Phi \to \mathbb{R}$), which are the only evaluations that we will need to perform when relying on a FSPA structure.

In the literature, these automata are usually implemented by means of Deterministic Rabin Automata (DRAs) or Limit-Deterministic Büchi Automata (LDBAs). External libraries like Rabinizer (Komárková and Křetínský, 2014) offer convenient and optimized solutions for automatically carrying out these conversions in the case of DRAs, making the process easily embeddable with game engines and highly practical, which is why in this work we opt for DRAs instead of LDBAs. Whenever we refer to the automata used in our approach, we will be talking about DRAs. Further conversion options are described by Rozier (2013) and Esparza et al. (2022).

3.3.3 A game environment example

To illustrate these concepts, we will now provide an example from LIQUID SNAKE, showcasing how LTL can be used to formulate a design specification for one of its levels, together with a possible FSPA generated from the requirements to be used later in a RL framework.

For the example presented here, we define $\eta : \mathcal{S} \to [0, 1]$ as the proportion of the player's remaining health, with $\pi := \eta(s) > 0$ representing the predicate "the character is still alive". For the notion of proximity of the character to a target *tar*, we define the function $\gamma_{\text{tar}} : \mathcal{S} \to [-1, 1]$:

$$\gamma_{\text{tar}}(s) = \begin{cases} 1 - \frac{\delta_{\text{tar}}(s)}{b}, & 0 \leq \delta_{\text{tar}}(s) < b \\ \frac{\delta_{\text{tar}}(s)^2 - 2B\delta_{\text{tar}}(s) + 2bB - b^2}{(b-B)^2}, & b \leq \delta_{\text{tar}}(s) \leq B \\ -1, & B < \delta_{\text{tar}}(s) \end{cases} \tag{3.3}$$

where $\delta_{\text{tar}}(s)$ is the euclidean distance between the character and the target in s, b is the lower bound below which we consider proximity to be achieved, and B is the upper bound beyond which we assume the distance to be too large to be considered close. This is nothing more than the second-degree polynomial that evaluates to -1 when the distance between objects is at its maximum

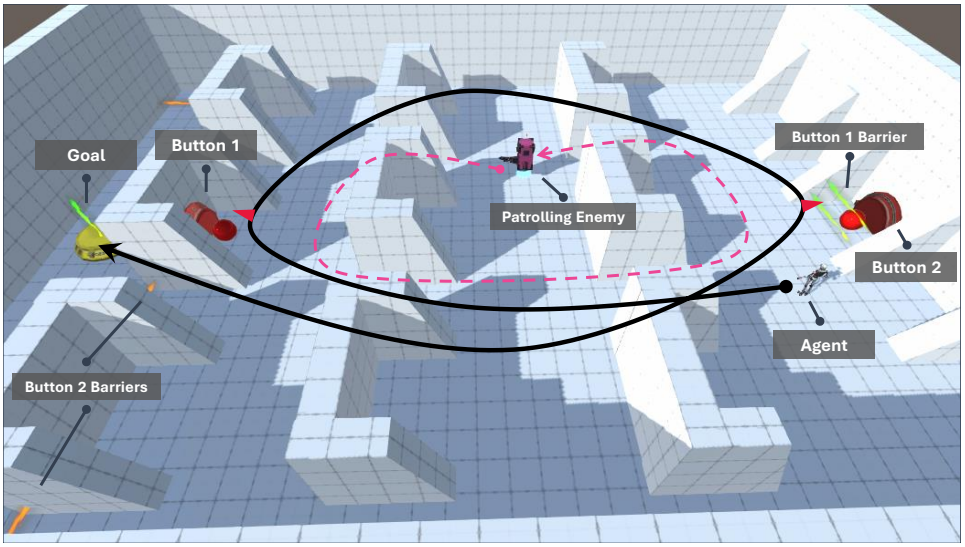


Figure 3.4: Screenshot of the "Cover and Hide" environment.

($\delta_{\text{tar}}(s) = B$) and 0 when it equals the proximity threshold ($\delta_{\text{tar}}(s) = b$), which turns into a linear function approaching 1 as the distance gets closer to 0. Empirically, we choose B as the size of the level's bounding box, $b = 1$, and write $\gamma_{\text{tar}} := \gamma_{(\text{tar})}(s) > 0$ to denote the player's proximity predicate to a target in the scene. For instance, if we set $B = 10$ and are currently 5 units away from the target, then $\gamma_{\text{tar}}(s) \approx -0.69$. This function has a steeper gradient as $\delta_{\text{tar}}(s)$ approaches b and is only positive when proximity has been reached.

One of the levels included in the environment, depicted in Figure 3.4, introduces an enemy that patrols an environment filled with multiple grid-based obstacles, obstructing visibility and movement for both the enemy and the player. The objective here is to visit and activate the button to the left, s_1 ("Button 1" in the figure), to unlock the barrier guarding the button on the right side of the environment, s_2 ("Button 2"). After that, the player must visit this second button to unlock the barriers protecting the area in the leftmost section of the level, and then reach the goal located there, e. All of this must be done while remaining alive. In terms of LTL the specification for this level can be expressed as follows:

$$\Diamond(\gamma_{s_1} \wedge \bigcirc(\Diamond(\gamma_{s_2} \wedge \bigcirc\Diamond\gamma_e))) \wedge \Box\,\pi. \tag{3.4}$$

From the designer's point of view, this translates into listing a series of conditions that must be met sequentially through blocks of "and next eventually" operators as mentioned earlier, in this case visiting three level points in order, along with a clause with task restrictions ("without dying").

This predicate can then be translated into a DRA using a translation library such as Rabinizer, outputting an FSPA like the one in Figure 3.5.

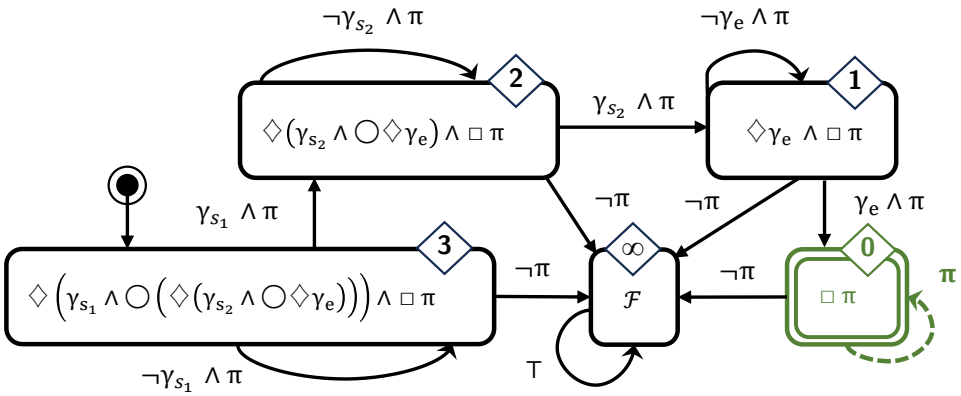


Figure 3.5: FSPA translation of the predicate from formula 3.4.

In this figure, the connections between states are labeled with the predicates that must be satisfied to trigger the transition. These automata are constructed so that there is always an edge equipped with a true formula, meaning that there will always be a transition to take at any given step. Furthermore, each of the states is labeled with a predicate representing what part of the specification remains to be fulfilled at that node, with the initial state holding the complete original formula, and subsequent states containing increasingly smaller formulas. We will discuss later how this property can be of help in supplementing the information provided by regression tests. Also note here the trap state that can be reached from any point of the automaton by violating the safety condition (when the player loses all health points), labeled with a false predicate indicating that it is no longer possible to satisfy the specification from it. The final state, on the other hand, is still accompanied by the original safety condition and outlined in green to denote that the automaton will accept any trace that ends in this state. Note that although this example produces a fairly simple automaton for the sake of readability, the FSPAs generated can be arbitrarily complex, especially when conditional clauses are introduced and multiple nested temporal operators are combined.

3.3.4 A progress-based algorithm for reward generation

Our approach to the reward allocation strategy, which is formally stated in Algorithm 1, is based on the concept of tracking progress within the task specification, in a way reminiscent of the potential functions mentioned earlier. We heavily rely on the structure of the reward automaton to measure this progress. In simple terms, at each step of the simulation, we calculate a value between 0 and 1, representing the proportion of the specification that has been satisfied up to that point, in a way reminiscent of the potential functions mentioned earlier in this chapter. The agent is then rewarded

based on how much it has improved compared to the best value of this metric during the episode.

We start with a design specification provided via a LTL predicate and translate it into an equivalent DRA-based FSPA. For each state as in the automaton, we then determine the minimum number of transitions required to reach a goal state. This metric is denoted by $d_g(as)$. This also allows us to identify "trap" states from which it is impossible to reach a goal (i.e. $d_g(as) = \infty$). This is valuable for determining when we can end the episode due to a specification violation. In the example from Figure 3.5, $d_g(as)$ is indicated by a diamond in the upper right corner of each state. Note how this is ∞ for the trap state and 0 for the single goal state outlined in green. With these distances, we can define SpecSize, as $\max_{as}(d_g(as))$ (line 4). This helps us reason about our progress based on an upper limit. In our example, SpecSize = 3.

With this information, during each agent step t (not necessarily linked to the game's update step), we apply the action chosen by the agent's current policy to the game state $gs^t \xrightarrow{Action} gs^{t+1}$ and update the automaton's state $as^t \xrightarrow{gs^{t+1}} as^{t+1}$. To do that, we evaluate the robustness of each outgoing edge ϕ_e from as^t, $\rho_{gs}(\phi_e)$, and take the transition with the highest value. We then estimate the base progress in the specification from as^{t+1} as $\frac{\text{SpecSize}-d_g(as^{t+1})}{\text{SpecSize}}$. This is stated in lines 7-11 and tells us how many steps we have taken toward a goal state relative to the maximum number of steps required to reach it. Continuing with our example, the state as with label $\diamondsuit\gamma_e \wedge \square\, \pi$ has $d_g(as) = 1$, and so we can estimate that we have completed at least $\frac{2}{3}$ of the task and initialize the progress at that value. This way of estimating progress assumes that all sub-tasks are equal in size, which might be an oversimplification depending on the context.

Now, if the transition would keep us in the current state, the reward is adjusted based on the best robustness that would improve the distance (lines 12-14). This indicates how much progress has been made in the current task. For instance, if we are at the same as from before, the only edge that would bring us closer to the goal would be the one with predicate $\phi_e = \gamma_e \wedge \pi$. If at the current time-step we are at full health and $d_e = \frac{t+T}{2}$ (half of the maximum distance to the level target), then we would have that $\rho_{gs}(\gamma_e) = -0.75$ (see Equation 3.3), $\rho_{gs}(\pi) = 1$ and $\rho_{gs}(\phi_e) = \min(-0.75, 1) = -0.75$. As the base progress was $\frac{2}{3}$ this means that our actual progress at the current step is $\frac{2}{3} + \frac{1-0.75}{SpecSize}$, which is approximately 0.75. If this value is higher than the last maximum recorded up to that point, the difference is added to the step reward (lines 25-28).

However, it could be the case that no transition would allow us to improve the distance to the goal. This means we are either in a trap state (line 19) from which we can never reach a goal or are located in a goal state from which we cannot improve further because it is already as good as possible

Algorithm 1 Algorithm for Reward Generation - Rabin FSMA

Initialize game state as gs^0
Initialize automaton state as as^0
Initialize episode progress as 0
SpecSize $\leftarrow \max_s(d_g(as))$
while $t < t_{max}$ **do**
  Initialize step reward as 0
  Action $\leftarrow$ policy(gs^t, as^t)
  Step Action in game state $gs^t \xrightarrow{Action} gs^{t+1}$
  Use gs^{t+1} and as^t to tick the Automaton, computing $\rho_{gs^{t+1}}(\phi_e)$ for all predicates ϕ_e in edges attached to as^t
  Select edge from as^t with the highest $\rho_{gs^{t+1}}(\phi_e)$ and transit Automaton using that edge $e : as^t \rightarrow as^{t+1}$
  step progress $\leftarrow \frac{\text{SpecSize} - d_g(as^{t+1})}{\text{SpecSize}}$
  if $(as^t = as^{t+1})$ **then**
    if other e' would improve distance to goal **then**
      step progress += $\frac{1+\max_{e' \in impEdges}(\rho_{gs^{t+1}}(\phi_{e'}))}{\text{SpecSize}}$
    else
      if e leads to a goal state **then**
        step progress = 1
        terminate the episode (success)
      else
        step reward -= terminal penalty
        terminate the episode (failure)
      end if
    end if
  end if
  if step progress > episode progress **then**
    step reward += step progress - episode progress
    episode progress = step progress
  end if
  step reward += existential penalty
  grant step reward to agent
end while

(line 16). Typically, when we reach a trap state, it is common to end the episode directly. In an acceptance state, it might sometimes be interesting to let the episode continue beyond the original goal for the agent to learn how to stay in it and improve the specification's quality. Nevertheless, for now, we are only interested in tasks that end after reaching a goal state, so we will not delve into the optimization details in that case.

Considering this structure, it is easy to see that the total accumulated reward throughout the episode always falls between 0 and 1. This implies that there is no difference between completing the specification sooner or later, as long as it eventually gets fulfilled. Because of this, it is often useful to introduce a penalty in each simulation step to encourage the agent to optimize the time required to complete the tasks (line 29). This also provides an initial estimate of how long it might take to fulfill the design intent. To maintain the interpretability of the results, we choose the value of the existential penalty in such a way that if the agent fails to complete the specification within the provided time, the total penalty amounts to -1. This changes the total reward range to [-1, 1]. With this decision, a negative value indicates a failure to meet the specification, with values closer to 0 representing progress towards task completion, while a positive value corresponds to successful specification completion, with higher values indicating "faster" policies. If existential penalties are applied, it is often advisable to apply a terminal penalty upon reaching a trap state, so as to prevent the agent from attempting to break the specification on purpose (for instance, by letting themselves die) in order to avoid further negative rewards.

3.3.5 Experimental evaluation

Our experimental evaluation sought to assess the ability of the methodology introduced in the previous section to develop agents for automatic testing. This not only translates into training bots that are able to complete the assigned LTL specifications in reasonable times, but also that these can be used a posteriori for the detection of significant variations in the gameplay, and more specifically in the possibility of satisfying these design requirements as different modifications are introduced to the game.

Our evaluation was structured as follows:

1. We defined the RL training setup, selecting PPO as the training algorithm of choice for its relative generality and lack of over-reliance on hyper-parameter tuning. Test specifications were set using LTL predicates to be evaluated on a training-step basis on game environment entities, and compiled into DRA-based FSPAs for reward generation. The agent's input comprised a one-hot-encoding of the index of the current automaton state, a set of user-defined vector scene elements and variables, and a semantic map given by a discrete categorization of the

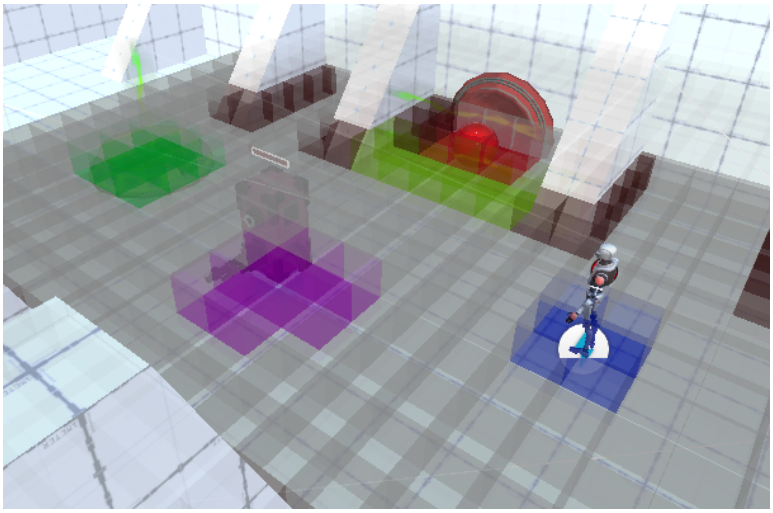


Figure 3.6: Semantic map used by the agent to perceive the environment.

space and the elements around the agent to grant it a certain level of spatial perception, following the approaches of Sestini et al. (Sestini et al., 2022a,b). The latter is defined by a three-dimensional grid of cube-shaped regions centered on the agent, with each of these regions describing the type of the object included in it by means of an integer value, as seen on Figure 3.6. The neural networks used to encode the policy had two fully connected layers with 512 hidden units each.

2. We established experimental parameters for the PPO algorithm (learning rate $= 0.0003$, $\beta = 0.01$, $\epsilon = 0.2$, $\lambda = 0.95$, $\gamma = 0.99$), with steps per episode and total training steps being level-dependent. We compared our approach against an imitation baseline, where agents were trained via behavioral cloning (BC) and GAIL from a set of human demonstrations completing the task at hand. Performance was measured using LTL task success rate, specification progress, and automaton reward, all evaluated over 25 simulations. Training stopped upon consistent specification satisfaction or after 20 hours.

3. We applied our approach to three distinct use cases in LIQUID SNAKE, each addressing common testing challenges. For each case, agents were trained using Algorithm 1 and the BC + GAIL approach, and evaluated on the corresponding task under three different setups: the original configuration of the environment prior to any alterations (i.e. the one used in training), a first variation of the level with minor edits, and a second version with major, potentially specification-breaking changes. Test cases are as follows:

   (a) **Validating Navigation.** This involved a tutorial level requiring

agents to perform a sequence of simple navigation actions, such as unlocking a passage by pressing a button, crouching to get accross low ceilings, and jumping to reach floating platforms. Modifications included both subtle obstacle changes and complete path obstructions.

(b) **Validating Multi-Stage Environments.** This corresponds to the example from Equation 3.4, where agents navigated multi-stage tasks with patrolling enemies. Modifications focused on enemy AI adjustments.

(c) **Validating Branching Environments.** This last complex scenario forced agents to take branching paths with fundamentally different tasks (a corridor with moving obstacles and a stealth maze with patrolling enemies) based on the random outcome of a button press at the beginning of the level. Modifications included minor obstacle path adjustments and further enemy AI changes.

Where edits do not completely invalidate the original specification, we expect agents to continue to be able to satisfy their predicates, possibly informing of meaningful variations in their performance. That is, if the agent is still able to validate the specification, but its reward metrics are noticeably altered, we can issue a warning that there is a chance that the gameplay experience of the level has been compromised. Predictably, there will be situations where the agent is no longer able to complete the task after the change, in which case we would like the test to be as informative as possible as to what may have caused this failure to meet the objective. Our intention will therefore be to ensure exhaustive coverage of the game design specifications, running periodic automated checks and reporting suspicious scenarios, these being the only ones that need to be manually reviewed by a human tester. Note that in the testing process the same agent is always used without retraining until a significant change is found.

Our primary findings for this experimental evaluation are listed below:

1. Our proposed method consistently achieved higher success rates and specification progress compared to the imitation baseline across all original and modified levels. For instance, in Case 1, our model maintained a 1.0 pass rate for the first battery of changes and 0.90 for the second set of modifications, significantly outperforming the imitation baseline's 0.92 and 0.32 respectively.

2. Training times were substantially lower for our approach. In Case 1, our model trained in 3.5 hours compared to the imitation baseline's 20 hours. Similar efficiencies were observed in Case 2 (2.7h vs. 20h) and Case 3 (6.8h vs. 20h). This demonstrates that our method can generate effective testing agents in a much more reasonable timeframe.

3. Our approach proved more robust to subtle environmental changes, indicating fewer false positives in regression tests. While both models detected significant breaking changes, our method provided more informative results through FSPA logs, allowing identification of where exactly the agent failed within the specification. This is a key advantage for debugging and targeted human review. The imitation model, in contrast, consistently struggled with performance degradation across modifications, rendering it unreliable for continuous testing.

Paper 4 (published in *IEEE Transactions on Games*) presents the details of the work introduced in this section to address Objective 1.3, including a comprehensive analysis of the results summarized above, a full description and screenshots of the environments used alongside the LTL predicates encoding the design specifications to be checked, and complete tables with quantitative results of our experiments.

3.4 A visual tool for defining LTL specifications in Unity

While temporal logics offer a powerful means to generate dense reward functions from formal specifications, the process of defining these specifications can be cumbersome and inaccessible for designers without a technical background. Several factors contribute to these challenges:

- Temporal logics, though more akin to natural language than other formal specification mechanisms, programming languages, or reward function definitions for complex tasks, are still formal languages. Their effective use requires a basic technical and mathematical understanding, which can be a significant barrier for game designers without such training. Furthermore, requiring a designer to define a formal specification textually using temporal logic operators is error-prone and, more importantly, deviates significantly from a typical video game designer's workflow, which often revolves around visual tools and graphical environments.
- Defining atomic predicates that represent world and context conditions (e.g., "being close to a point" or "having low health points") must be accessible from the game engine. This accessibility requires an explicit mapping between these expressions and a numerical evaluation of their robustness. Ultimately, all calculations related to a predicate's robustness depend on evaluating the quantitative semantics of its atomic predicates, which must be manually defined and are domain-specific.
- Linking the inputs of these atomic predicates to various game variables and entities is crucial. This allows the agent to evaluate world

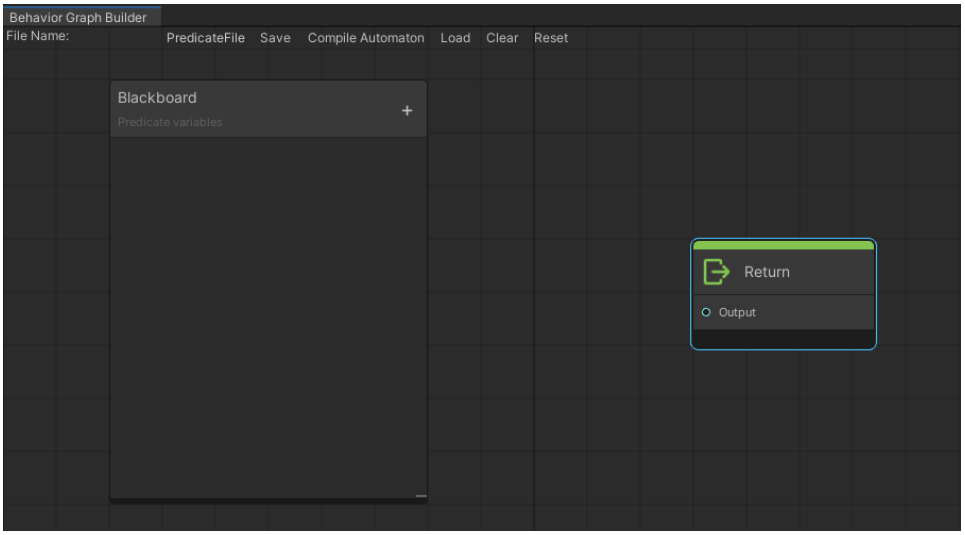


Figure 3.7: Default Editor Window.

conditions at each simulation step. These inputs can be tied to arbitrary environment variables and their `GameObjects`, and ideally, they should be connectable via drag-and-drop mechanisms to spare the designer from dealing with source code.

- Managing the training process and algorithm, as outlined in the previous section, involves translating the specification into an automaton, evaluating transitions, and assigning rewards at each simulation step. All of these technical details should be hidden from the designer, requiring no manual intervention beyond visually introducing their specification, linking variables from the editor, and initiating the training process with a sequence of clicks.

To address these challenges, in (Gutiérrez-Sánchez et al., 2023a), we introduce *The Unity AI Behavior Graphs Toolkit*, a dedicated tool engineered to simplify the creation and training of our LTL-driven NPC behaviors in Unity 3D. This toolkit enables designers to outline NPC behaviors through an intuitive visual node editor and subsequently generate reward functions for training reinforcement learning agents, streamlining the process described in Section 3.3.

3.4.1 Basic usage

Once the tool has been added to the Unity project, designers can begin interacting with the graphical behavior window via a dedicated inspector menu option in the engine editor. This opens a window similar to the one shown in Figure 3.7, presenting an empty default graph that includes a

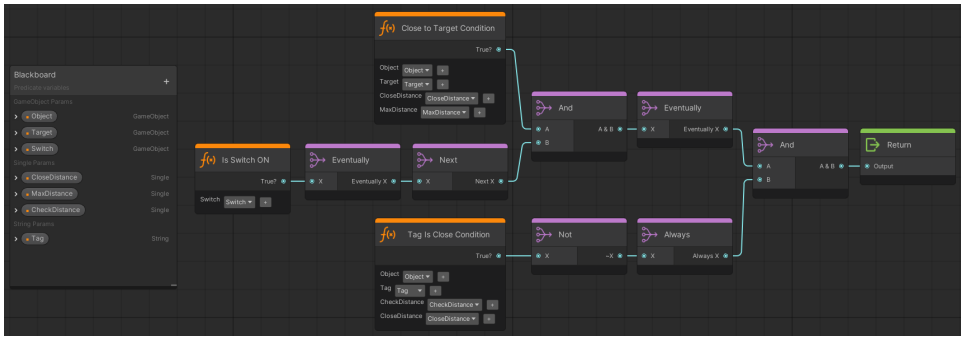


Figure 3.8: Example editor graph for the specification from Formula 3.5.

blackboard with no variables and the predicate *Return* node. Within the graph view, two fundamental elements are present:

- **Blackboard**: Used for storing parameters that can be referenced by nodes in the graph. Parameters are categorized by type (e.g., `GameObject`, `Vector3`, `String`).

- **Node Editor**: A graphical editor facilitating the addition, deletion, editing, and management of connections between nodes. To introduce a new node, users can right-click in the editor and select the "Create Node" option to access the node exploration panel. This explorer is structured into nested categories, including `Flow` (for core temporal logic operators) and `Conditions` (for basic condition operators). New categories and nodes can also be included, as detailed in Section 3.4.2.

Node input parameters can be other nodes or predicates within the graph, represented by input ports that receive connections via edges, or blackboard variables. In the latter case, the parameter is listed in a dedicated section at the bottom of the node, accompanied by a dropdown selector and a "+" button. The dropdown allows users to select the blackboard variable linked at runtime, while the button enables the creation of a new variable of the appropriate type on the blackboard.

It is important to note that while a behavior graph can contain multiple disconnected node groupings, only the component linked to the special return node (green, always unique per graph) will be considered at runtime.

As an example, Figure 3.8 illustrates an instance of a behavior graph and its blackboard, representing the specification from the following LTL formula, requiring the player to eventually approach a switch and next activate it at some point, while ensuring that no enemy ever gets too close to the character:

$$\phi := \Diamond\,(\texttt{switchClose} \wedge \bigcirc\,(\Diamond\,\texttt{switchOn})) \wedge \Box\,\neg\texttt{enemyClose} \tag{3.5}$$

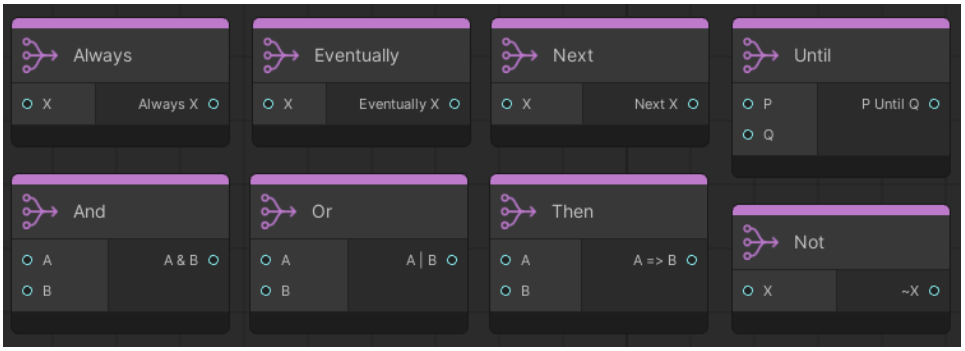


Figure 3.9: Currently available flow nodes.

Notice how the original predicates have been expressed using more general nodes (e.g., `switchClose` becomes `targetClose`). This tree-like representation is often more intuitive and easier for non-technical users to understand than the original formula.

3.4.2 Adding custom editor nodes

The AI Behavior Graphs Toolkit differentiates between two types of nodes based on their role in constructing the specification:

- **Flow Nodes** correspond to standard operators in Linear Temporal Logic that accept a set of LTL predicates as parameters (e.g., eventually, always, until). This is a closed set of operators; currently, no other composite nodes are supported, meaning user-created nodes cannot introduce references to other predicates as parameters. A visual overview of the available flow nodes is provided in Figure 3.9.
- **Condition Nodes** correspond to robustness functions in LTL. They represent a condition on a set of parameters whose truth value is not boolean (`true` or `false`) but rather a numerical value indicating its robustness.

Although some default simple conditions are included, users can manually define new condition nodes for inclusion in behavior graphs. To do so, users must extend the abstract class `LTLPredicate`, as shown in Figure 3.10 for a predicate that computes the robustness of $A < B$. The most crucial function here is `EvaluateRobustness`, which returns a numerical value representing the extent to which the predicate is satisfied at the current moment. While there are no strict upper or lower bounds for the returned value, it is recommended to keep it within a normalized range of $[-1, 1]$ to ensure consistent interactions with other operators and conditions and prevent it from being overwhelmed by the magnitude of nearby predicates.

```
[Predicate("Conditions/Less Than", "A < B")]
public class LessThanPredicate : LTLPredicate {
    [InParam("A")]
    public float A { get; set; }
    [InParam("B")]
    public float B { get; set; }
    public override float EvaluateRobustness() {
        return B - A;
    }
}
```

Figure 3.10: Defining a custom node for the $A < B$ predicate.

Since a predicate of this type is essentially a function that returns a value based on inputs, these parameters are also important. Input parameters available in the predicate's robustness calculation must be declared in the class extending `LTLPredicate` using the annotation `[InParam("paramName")]`, specifying the name by which the user wishes to refer to the parameter from the associated node in the graphical editor. This name does not necessarily have to match the variable name in the class.

Finally, for the predicate to be available as a node in the editing tool, the implemented class must be accompanied by the annotation `[Predicate("path/to/condition", "predicateOutputName")]`. Recalling now the structure of the node explorer in the graphical editor, each node can be included in a different category, potentially nested in a chain of subcategories, as seen with `LessThanPredicate`, which appears under the `Conditions` category (see Figure 3.11). The first input parameter of the preceding annotation corresponds to this category "path" and must be manually defined by the class implementer. The second parameter corresponds to the text that will appear in the graphical editor to represent the node's output value, which can further clarify the condition's nature to the user.

3.4.3 Compiling automata

Once a satisfactory graph for an NPC is ready, the next step is to compile it into an automaton for runtime use. In reality, the visual tool generates an LTL specification and metadata for its graph representation. Neither can be directly used to reward an agent during training. To enable this, we convert the original specification into a structure that allows local reasoning about the predicate, abstracting all dependencies and temporal operators into states where the current position in the process is known. This is achieved through the use of FSPAs, as introduced in Section 3.3.2, specifically constructed using Rabin Automata in our case.

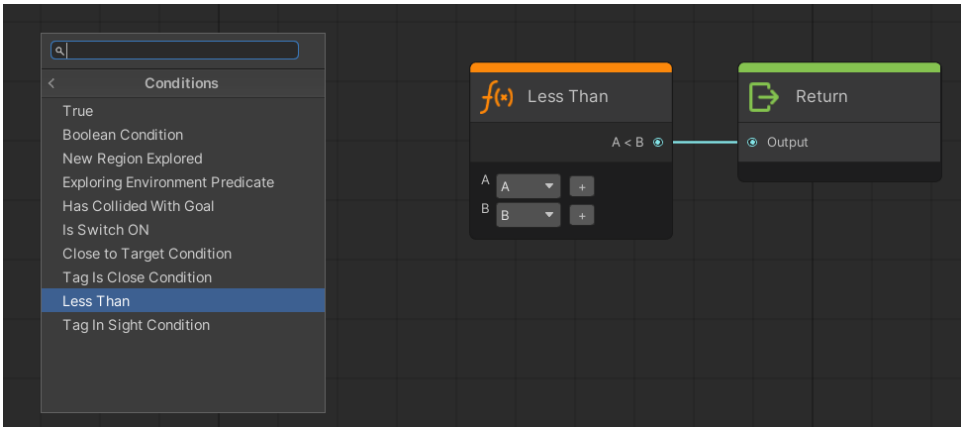


Figure 3.11: Less Than Node.

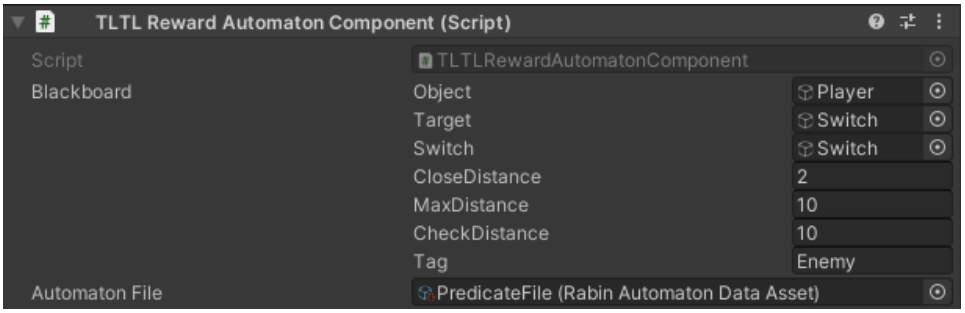


Figure 3.12: Reward Automaton Component as seen in the Unity Editor.

However, converting a Linear Temporal Logic specification into a Rabin automaton is a complex challenge. To facilitate this, we currently employ Rabinizer 3 (Komárková and Křetínský, 2014), an external open-source Java library, executed via a C# Process from the Unity Editor. Selecting the "Compile Automaton" option from the editor menu triggers the conversion process. This begins by transforming the current predicate into a format acceptable to the Rabinizer library. Once the output automaton is ready, the process parses the result into a Rabin Automaton Data Asset, which can be used for runtime inference. Note that the generated asset contains only references to the necessary classes for instantiating different node types in the specification at runtime, as well as the blackboard variables they reference. These variables remain without actual values until assigned to an entity in the scene.

The graph in Figure 3.8 does not precisely reflect the intended specification. The predicates used are mostly generic (e.g., "object near the target" instead of "player near the switch") and require specific entities to convey the designer's real intent. To perform this assignment, the toolkit provides the `LTLRewardAutomatonComponent`, which acts as a bridge between

the automaton-format specification asset provided as a parameter, its blackboard variables, and the elements in the scene. This component is typically added to the NPC's game object that should satisfy the specification. An example after assigning the asset compiled from Figure 3.8 with the necessary input parameters can be seen in Figure 3.12. Here, some parameters are literals, while others are direct references to game objects.

At runtime, this component instantiates the automaton's transitions with specific instances of the predicates to be executed during the game to evaluate conditions and user-introduced references, resulting in what we call a `LTLRewardAutomaton`. This acts as a data container that does not update itself during execution, delegating this task to other components that can trigger such updates by calling the automaton's `Tick` method.

Paper 5 (published in *Actas del II Congreso Español de Videojuegos, 2023*) presents the details of the work introduced in this section to further address Objective 1.3, aiming to tackle the part of the goal concerned with providing a designer-friendly and intuitive framework that alligns with their existing cognitive models.

3.5 Player modelling via temporal logics

Once we have the necessary infrastructure to evaluate LTL predicates on a game's execution environment, we can then address the inverse problem to the one developed in the previous chapter. In this scenario, instead of starting with an LTL design specification and producing a reward function to train a model capable of satisfying it, we begin with a set of game traces (recorded gameplay sessions) and seek an LTL specification that can explain the observed behavior within these traces. This approach has multiple applications:

- **Player Analysis and Modeling**: By analyzing a collection of game traces from a single player or a specific playstyle, we can uncover patterns that explain their observed behavior. This is invaluable for understanding how players interact with the game and what strategies they employ.
- **Clustering**: Given an unclassified set of game traces, we can identify latent patterns that differentiate between various playstyles present in the data. This helps in categorizing players into distinct groups or styles, leading to a better understanding of player diversity and preferences, and enabling the design of more personalized gaming experiences.
- **Inverse Reinforcement Learning**: If we start with a set of game traces that serve as demonstrations from an expert player, and we

distill an LTL predicate that explains the observed behavior, we can then apply the techniques from the previous section. This allows us to transform that predicate into a reward function that can be used to train a reinforcement learning agent. In doing so, we achieve a configuration similar to the inverse reinforcement learning paradigm, where the primary goal is to learn a reward function directly from a set of demonstrations.

- **Explaining Testing Traces**: When we have a set of traces generated by a regression testing bot, we can search for patterns that provide temporal properties about the agent's behavior. This adds an additional layer of analysis to the testing process results, enriching the information provided by the bot.

3.5.1 Inferring temporal characterisations of play-traces

In this section, we present our algorithms for learning temporal predicates to characterise play styles through a set of examples and counterexamples. We begin by formally defining the problem at hand.

Let $\mathcal{P}_g$ be the (possibly infinite) set of LTL predicates that can be generated from an LTL grammar g. We define a trace evaluator σ as a function that takes a predicate $p \in \mathcal{P}_g$ and a real-valued trace of length $n \in \mathbb{N}$, $t = \{s_1, \ldots, s_n\}$, in a given trace space Σ^* and outputs an array of fitness values representing how well the predicate models the trace according to the function's criteria in a series of dimensions. More formally, if $m \in \mathbb{N}$ is the number of fitness dimensions (or alternatively, the number of metrics that are computed for each trace), this is a function with signature $\sigma : \Sigma^* \times \mathcal{P}_g \to \mathbb{R}^m$:

$$\sigma(t, p) = (\sigma_1(t, p), \ldots, \sigma_m(t, p)),$$

where $\sigma_i(t, p) \in \mathbb{R}$, $0 \leq i \leq m$, is the i-th fitness value of the trace t with respect to the predicate p.

This allows us to convert a set of traces into a real vector representation that can be used to compare traces and predicates with each other according to a set of user-defined heuristics. On top of this, we can define additional functions to aggregate the results of each evaluator considered in the analysis into a single goodness value accounting for evaluators' outputs across sample sets (usually a combination of a positive examples set and a negative one). A typical aggregation function has the signature $\gamma_\sigma : (\Sigma^*)^* \times (\Sigma^*)^* \times \mathcal{P}_g \to \mathbb{R}$.

With these notions in mind, we are ready to formally state this section's problem.

Problem 1. Given a set of positive traces or examples P, a second set of negative traces or counterexamples N for a game, a list of trace evaluators σ_i, $i = 1, \ldots, m$, and a metrics aggregator γ_σ, learn an LTL predicate ϕ

such that: (1) $t \models \phi$, $\forall t \in P$; (2) $t \not\models \phi$, $\forall t \in N$; and (3) $\gamma_\sigma(P, N, p) >= \gamma_\sigma(P, N, p'), \forall p' \in \mathcal{P}_g$.

The first two conditions indicate that we aim to find predicates that fit the considered examples and counterexamples, while the third one states that we are also interested in the chosen predicates being optimal with respect to the established metric aggregator. In practice, we will use approximate strategies to address this problem, so the above conditions will not always be met exactly.

3.5.1.1 Brute force tree expansion

The first search method we will consider to approximate Problem 1 can be seen in Algorithm 2, and is a brute-force approach in which all possible solutions within certain constraints are explored in search of the optimal one in terms of fitness.

First of all, it is to be noted that, from this point onwards, whenever we speak of a grammar, we will assume that we are working with one in Backus-Naur form (commonly BNF), a meta-syntax notation for context-free grammars, often used to describe the structure of language specifications. We will also assume when referencing clauses representing functions over the state of the system ($f(s) > 0$ in Equation 3.2) that the grammar has been extended to represent all variables of interest in the context where the language is used. This implies that the $f(s) > 0$ symbol can be expanded to any of the possible literals recorded in a game as an implicit rule.

That stated, the brute-force algorithm starts from the initial node of the grammar considered, and builds a production tree $\mathcal{T}_g$, in which each node is expanded according to all the possible derivations allowed from it. Since this tree is usually infinitely deep (one need only consider the recursive rules that reference a predicate within another), here we opt to prevent this construction from exceeding a maximum depth specified by the parameter $D \in \mathbb{N}$. Once the pruned tree is available, it is possible to perform any traversal of its leaves, collecting all the predicates that have been completely expanded within the imposed limit. For each of the predicates gathered in this way, it is enough to apply the chosen metrics aggregation function, and keep the individual with the best value according to this heuristic.

Algorithm 2 Brute Force Tree Expansion with Fixed Depth D.

Generate a production tree $\mathcal{T}_g$ from input grammar g, allowing a maximum depth of D.
For each leaf node in the tree that results in a fully expanded LTL predicate p, compute $\gamma_\sigma(P, N, p)$.
return $p \mid \gamma_\sigma(P, N, p)$ is maximal among terminal nodes.

3.5.1.2 Grammatical evolution search

Grammatical Evolution (GE) (O'Neill and Ryan, 2003) is a type of genetic programming (GP; (McKay et al., 2010)) that evolves solutions by generating computer programs or expressions, guided by a predefined grammar. A common tool used in GE is precisely the Backus-Naur Form grammar, which provides a formal way to specify the syntactical structure of the programs or expressions to be evolved. GE encodes solutions as sequences of integers, which are then mapped to syntactically correct programs using the BNF grammar.

The adaptation of our problem to a grammatical evolution model is straightforward by setting up the search grammar we are interested in alongside the heuristic function given by the corresponding metric aggregator, and is relatively simple to implement with genetic programming libraries such as the MOEA Framework (Hadka, 2025).

3.5.1.3 Grammar-based Monte Carlo tree search

Monte Carlo Tree Search (MCTS) is a highly-selective best-first search method that is often used for decision-making. This algorithm balances between exploitation and exploration to balance the search tree growth towards the most promising parts of the search space. This iterative method combines a tree policy (e.g. UCB1 (Gelly and Wang, 2006)) with Monte Carlo simulations or *rollouts*, which are trajectories sampled at random in the decision space. For more information about this algorithm and its variants, please refer to (Browne et al., 2012).

In our particular case, we model the search as a sequential decision problem on which production of the reference grammar to apply at each step of a derivation. That is, starting from a first state given by the initial symbol of the grammar, in each run of the MCTS we are interested in first deciding which of the symbols of our current state we wish to expand and, secondly, which of its production rules to use to perform this expansion. As MCTS heuristics, we make use of the metric aggregation function common to all the algorithms in this section, assuming a value of 0 for predicates that have not yet been fully expanded. This means that only rollouts that result in final states can yield useful information.

3.5.2 Tree clustering of play-traces

The second problem that was undertaken here was to propose a classification or clustering approach to distil general behavioral patterns within samples from a potentially diverse set of traces. This is a purely exploratory analysis problem, and as such more complex to define than the task in the previous section. In this case, we shall frame it in terms of looking for predicates that

Algorithm 3 Grammar-Based MCTS.

Initialise base predicate s_p state with the base symbol from the chosen search grammar.
while s_p not fully expanded **do**
  Apply MCTS to s_p with allocated budget to select the seemingly best symbol to expand from the set of unexpanded symbols in s_p and production rule from the given grammar.
  Update s_p after applying the suggested expansion.
end while
return s_p.

are capable of "separating" traces as much as possible in the sense that the sets of samples that do and do not satisfy the predicate are cohesive and at the same time distant from each other.

More specifically, starting from the previously defined trace evaluators and some form of standard clustering metric, our problem is to construct a predicate that seeks to optimise this metric for the partition given by the sets of traces that it accepts and rejects, respectively.

More formally, given a predicate $p \in \mathcal{P}_g$ and a set of traces T, we define the sets $C^p_{\models}(T) = \{t \in T, t \models p\}$ and $C^p_{\not\models}(T) = \{t \in T, t \not\models p\}$ of traces accepted and rejected by that predicate. By now applying a trace evaluator σ on each member of these clusters we obtain two new sets of individuals $\sigma(C^p_{\models}(T)), \sigma(C^p_{\not\models}(T)) \subset \mathbb{R}^m$, on which it is now possible to apply a clustering metric to heuristically evaluate how good the partition of the samples into these two sets is. With this, we can formally formulate our second problem.

Problem 2. Given a set of unclassified traces or examples T, a list of trace evaluators $\sigma_i, i = 1, \ldots, m$, a clustering quality score function q, and a quality threshold $\alpha > 0$, learn a sequence of predicates $(p_j)_{j=0}^n$ such that:

$$P_j = C^{p_{j-1}}_{\not\models}(P_{j-1}), N_j = N_{j-1} \cup C^{p_{j-1}}_{\models}(P_{j-1}), P_0 = T, N_0 = \emptyset, \tag{3.6}$$

$$q(\sigma(C^{p_j}_{\models}(P_j)), \sigma(C^{p_j}_{\not\models}(P_j)), \sigma(N_j)) >= \alpha, j = 0, \ldots, n-1, \tag{3.7}$$

$$q(\sigma(C^{p_n}_{\models}(P_n)), \sigma(C^{p_n}_{\not\models}(P_n)), \sigma(N_n)) < \alpha. \tag{3.8}$$

Intuitively, we are describing an iterative task. We start from a set of traces T to be partitioned based on predicates p_j that explain a subset of samples with sufficient confidence $\alpha > 0$. In the first iteration, we are simply trying to find a predicate p_0 that splits T into two cohesive sets that are sufficiently separated from one another according to our clustering criterion in the real space induced by our selected metrics. N_1 will then become the set of traces that were modelled by p_0 and that as such do not need further classification, while P_1 becomes populated with the traces discarded in the first round. The next iterations follow this pattern, with the subtlety that

we keep expanding N_j with the traces modelled by the sequence of splitting predicates, and we then try to find subsequent predicates in such a way that the groups $\sigma(C^{p_j}_{\models}(P_j)), \sigma(C^{p_j}_{\not\models}(P_j))$ and $\sigma(N_j)$ maintain a good clustering score. Therefore, at each iteration we ask ourselves the question "which trace groups can we find that appear to behave differently to all previous explanations?". In other words, we look for predicates that can explain unique and meaningful behavioral subsets, and progressively more specific ones. These iterations continue until it becomes infeasible to find predicates that induce a reasonable partitioning according to the quality threshold.

Algorithm 4 Tree Clustering of Trace Set T with Threshold α.

Choose a list of trace evaluators $\{\sigma_i\}_{i=1}^m$.
Choose a clustering quality score q.
Choose a search algorithm S_{alg} from Problem 1, selecting $\gamma_\sigma(P, N, p) = q(\sigma(C^p_{\models}(P)), \sigma(C^p_{\not\models}(P)), \sigma(N))$ as metrics aggregator.
$P_0 = T$.
$N_0 = \emptyset$.
$j = 0$.
$p_0 \leftarrow S_{\text{alg}}(P_0, N_0)$.
while $\gamma_\sigma(P_j, N_j, p_j) >= \alpha$ **do**
  $j = j + 1$.
  $P_j = C^{p_{j-1}}_{\not\models}(P_{j-1})$.
  $N_j = N_{j-1} \cup C^{p_{j-1}}_{\models}(P_{j-1})$.
  $p_j \leftarrow S_{\text{alg}}(P_j, N_j)$.
end while
return $\{p_j\}$, sequence of splitting predicates.

Algorithm 4 specifies a procedure by which to heuristically and approximately construct a sequence of predicates with the above conditions. Leveraging the constructs developed in the previous section for Problem 1, the tree clustering algorithm aims to model the trace separation steps as predicate searches that optimise the result of the metric aggregator given by:

$$\gamma_\sigma(P, N, p) = q(\sigma(C^p_{\models}(P)), \sigma(C^p_{\not\models}(P)), \sigma(N)). \tag{3.9}$$

That is, at each step we look for a predicate that induces the best possible separation between the traces we have already explained through previous predicates, the traces that become explained by the new predicate, and the traces that remain to be modelled.

3.5.3 Experimental evaluation

A preliminary experimental evaluation was performed to validate the effectiveness of the algorithms proposed in the previous sections on a complex

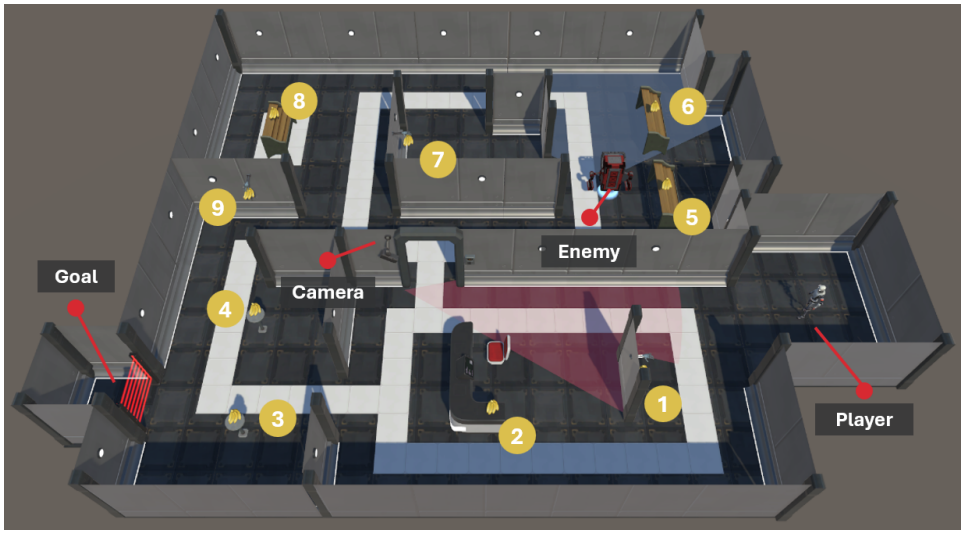


Figure 3.13: Top View of Reference Level.

level of LIQUID SNAKE, which can be seen in Figure 3.13. In this level, the player must collect objects 1 to 9 marked in the image to unlock the exit door, while avoiding detection by the security mechanisms and enemies present on the stage. This level is governed by stealth rules, where the enemies accumulate suspicion for the player's actions, and if it reaches a certain threshold, the player is captured and the level ends.

For this pilot study, we gathered 15 player traces across five distinct play styles labelled A-E. These styles ranged from aggressive approaches (e.g., Style A, where the player is detected and captured after trying to rush through the first area) to stealthy completion (e.g., Style C, where the player avoids detection by slowly following the robot from behind). We recorded system state variables (i.e., the value of quantitative semantics for available environmental predicates) every 0.1 seconds, including player-object proximity using the metric from Equation 3.3, object collection status, goal achievement, enemy visibility and suspicion levels, and player actions (e.g., crouching). The traces were collected directly by the authors of the study, who played the level multiple times, attempting to follow the play styles described above. It should be noted here that in this preliminary stage of the research, issues of size, diversity, or representativeness of the traces were not yet considered. Instead, we focused on the viability of the proposed algorithms for finding interesting patterns in relatively small and simple sets of traces, leaving a more exhaustive analysis of these aspects for future work.

We then defined our set of trace evaluators σ_i to assess the quality of the predicates generated by our algorithms. These evaluators were designed to capture different aspects of player behavior and the effectiveness of the predicates in characterizing it:

- **Progress**. We define progress following the notions introduced in Section 3.3.4, as the proportion of the predicate task that was completed by the trace based on the distance to a goal state in the associated automaton (note how progress is 1 if and only if $t \models p$):

$$\rho_{\text{FSPA}}(t) = \frac{D_{\text{FSPA}} - d_{\text{FSPA}}(s^t_{\text{FSPA}})}{D_{\text{FSPA}}} + \frac{1 + \max_{e \in impEdg}(\rho_{s_n}(\phi_e))}{D_{\text{FSPA}}}. \tag{3.10}$$

- **Exploitation**. Let i be the instant where the FSPA associated with the predicate accepts or rejects the trace t; we define exploitation $e(t,p)$ as $e(t,p) = 1 - \frac{|t|-i}{|t|}$, or the proportion of trace steps that are processed before the underlying FSPA reaches an acceptance decision. Intuitively, we are interested in a predicate showing high values of exploitation in positive examples, where this metric can often be interpreted as implying that the predicate contains relevant information about the reference trace.

- **Reward**. We define the reward $r(t,p)$ as the sum of variations in progress for each of the automaton's execution steps:

$$r(t,p) = \sum_{i=1}^{|t|-1} (\rho_{\text{FSPA}_p}(t^{i+1}) - \rho_{\text{FSPA}_p}(t^i)). \tag{3.11}$$

  This value allows us to identify segments in which a player is actively striving to achieve a sub-goal through local variations in the robustness of certain state variables.

These metrics are combined in the following aggregation function:

$$\begin{aligned}\gamma(P,N,p) =& \frac{\sum_{t \in P}(e(t,p) + r(t,p) + \rho(t,p))}{|P|} \\ &+ \frac{|\{t \in P \mid t \models p\}|}{|P|} + \frac{|\{t \in N \mid t \not\models p\}|}{|N|}.\end{aligned} \tag{3.12}$$

We apply the algorithms from Section 3.5.1 to Problem 1 to try to infer predicates that characterise each of the play styles relative to the rest of the styles. That is, if the styles are given by $S = [A, B, C, D, E]$, then we are considering the problems that take $P = S_i$, $N = \bigcup_{j \neq i} S_j$, $i = 1, \ldots, 5$. In all of them we use the metrics aggregator from Equation 3.12 and the grammar from Equation 3.2. All experiments were conducted on an Intel Core i7-9750H PC with 16GB RAM. Our main findings for each of the algorithms are as follows:

- The Brute-force Algorithm, set to $D = 6$, explored all predicates up to that depth (428,750 nodes), consistently taking approximately 6 minutes per search. While stable in execution time, it produced broad,

general predicates (e.g., `G(!playerCrouching)` for Style A) that, due to their global nature, often resulted in low reward values. This means that the predicates were not very informative about the player's behavior, as they did not capture specific actions or sequences of interest, but were always the best in terms of fitness for the given maximum depth. An important consideration is that, since the brute force algorithm explores all possible combinations of production rules to a fixed depth, the computational complexity grows exponentially with the maximum depth allowed. This means that, although exhaustive results are obtained for small depths, increasing this depth to capture more complex patterns can quickly become unfeasible due to the time and computational resources required. The choice of $D = 6$ in this case is taken as the greatest depth for which the algorithm can run in a reasonable time (less than an hour) in this case study.

- The Grammatical Evolution Algorithm, with a population size of 100, individual size of 15, and 100 generations, took between mere seconds to about 4 minutes per search for the reference problems. It produced predicates that were generally less informative and of lower fitness than those from the brute-force method, with predicates often focusing on single events (e.g. `F(goalReached)` for Style C). This indicates once again that the algorithm struggled to capture the complexity of player behavior, resulting in overly simplistic predicates that did not provide a comprehensive understanding of the play styles, albeit being faster than the brute-force method.

  While the choice of individual and population sizes reflects a trade-off, it is important to note that increasing the individual size can exacerbate one of the known limitations of Grammatical Evolution: low locality. In GE, small changes in the genotype can lead to disproportionately large or unpredictable changes in the phenotype due to the mapping process from codons to grammar rules. Larger individuals often increase this variance, as more codons provide more opportunities for such disruptive changes. In this sense, while bigger individuals can theoretically represent more complex patterns, they can also lead to highly erratic search behavior, making convergence less predictable and the algorithm more prone to producing low-fitness or overly simplistic solutions. Similarly, a larger population allows exploration of a wider range of solutions but requires more computational resources per generation. The values selected here represent a compromise based on preliminary testing, balancing solution complexity, search stability, and computational efficiency in the context of our problem.

- The MCTS Algorithm, with a budget of 300 iterations per expansion, rollout length of 15, and an exploration constant of $K = \sqrt{2}$, resulted

in predicates that were generally more elaborate than in the other two cases, but at the same time more informative, while remaining relatively fast (generally in the same ranges as the grammatical evolution approach). For instance, in the case of style E, the proposed explanation starts by specifying that the level is not completed, followed by a characteristic notion of order: object 3 is collected first, and then at some point object 8 is collected. This corresponds to the observation that this type of player traverses the top of the level in reverse order to the order in which the guard traverses. In the case of style B, on the other hand, it is indicated that there always comes a point in these traces where the player becomes detected by the camera until the end of the game, which again is a representative explanation of the reason for the defeat of this group. The algorithm generally featured significantly higher values for the reward metric, intuitively indicating its ability to capture consistently positive variations in progress across the traces, and thus to identify segments in which the player is actively striving to achieve a sub-goal.

The choice of MCTS parameters reflects a balance between exploration and exploitation. A budget of 300 iterations per expansion allows the algorithm to moderately explore the search space without incurring excessive computational cost. The rollout length was preliminarily set to 15 to allow reasonably informative derivations from the root node. However, one issue with this direct adaptation of MCTS is that the rollout length remains constant throughout all stages of the search, which is less appropriate in our use case than in typical game-decision applications. In a game, it may be useful to capture as much future information as possible at all times, but in our case, once a significant number of derivations has been reached, continuing to consider predicates that accept many more derivations can overly affect both the readability of the results and the evaluation time of associated fitness metrics. In light of this, in future iterations we plan to allow a dynamic reduction of this parameter during the algorithm's execution. The exploration constant $K = \sqrt{2}$ is a common and standard choice in practice that balances the need to explore new areas of the search space while exploiting known high-fitness areas. These parameters were selected based on preliminary experiments to optimize performance in the context of our specific problem.

The above results suggest that the MCTS algorithm is the most effective for this task, as it balances exploration and exploitation, leading to more informative predicates that capture the nuances of player behavior. The brute-force method, while thorough, tends to produce overly general predicates that do not provide much insight into specific play styles. The grammatical evolution approach, although faster, also fails to capture the

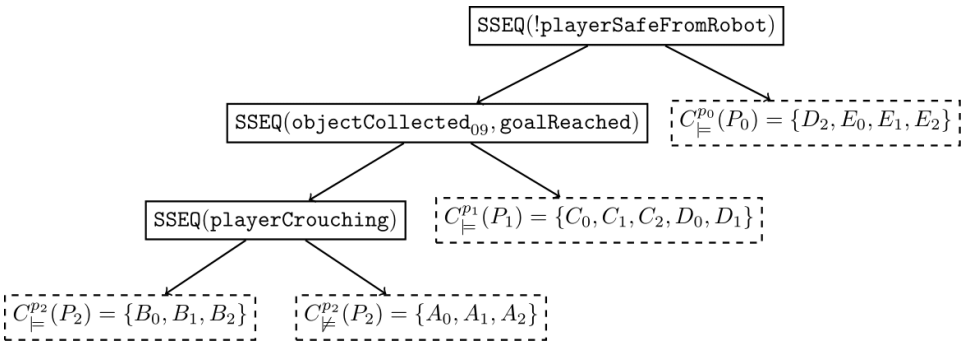


Figure 3.14: Clustering Diagram for Player Traces.

complexity of player behavior effectively.

We then applied Algorithm 4 to identify prominent behavioral patterns in the input set $T = \bigcup_i S_i,\ i = 1, \ldots, 5$ including our original 15 traces, unlabelled. Based on the results of the previous experiment, MCTS was used as the search algorithm, with the same parameters as before, as were trace evaluators. The clustering quality score q was defined as the **Silhouette Score** (Rousseeuw, 1987), and the threshold α was set to 0.51. A new grammar was used for this task, in order to demonstrate the flexibility of our system to adapt to different types of questions. In this case, we used a grammar that allows us to express sequences of events that happen in a strict time-ordered manner:

$$\begin{aligned} \phi &:= \texttt{SSEQ}(\texttt{lit-seq}) \\ \texttt{lit-seq} &:= \texttt{lit} \mid \texttt{lit}, \texttt{lit-seq} \\ \texttt{lit} &:= f(s) > 0 \mid \neg(f(s) > 0), \\ \texttt{SSEQ}(l) &= \mathcal{F}(l), \\ \texttt{SSEQ}(l_1, \ldots, l_n) &= \mathcal{F}(l_1 \wedge \mathcal{X}(\texttt{SSEQ}(l_2, \ldots, l_n))) \wedge !l_2 \mathcal{U} l_1. \end{aligned} \tag{3.13}$$

The execution of our algorithm with this setup resulted in the tree clustering diagram shown in Figure 3.14. The groups established in this way, represented as dashed boxes in the figure, are defined on the basis of a negation of all previous properties plus an additional constraint (represented by solid boxes) that refines their characterization when contrasted with the remaining traces. These clusters closely resembled our original play styles, suggesting that the algorithm effectively identified meaningful patterns.

It is important to note, however, that the interpretability of these results depends partly on the reader's familiarity with this type of structure and, more fundamentally, with temporal logics themselves. In addition, the choice of search grammar directly affects readability: a more complex or less constrained grammar may generate predicates that are obscure or difficult to interpret. By contrast, the grammar used in our experiments imposed a

strict sequential structure, which favored readability by producing properties that can be understood as sequences of events occurring in a specific temporal order. This design choice makes it easier to interpret the actions and behaviors characterizing each player group, making the results more accessible for analysis—though this advantage would not necessarily hold if a more general grammar were used.

Paper 6 (published in *Proceedings of the 20th International Conference on the Foundations of Digital Games (FDG '25)*) presents the details of the work introduced in this section to address Objective 1.4, including an in-depth analysis of the results summarized above, alongside predicate examples for each algorithm and play style, and statistics for fitness, execution time, trace evaluator metrics, and acceptance and rejection rates for positive and negative sets in each problem.

Chapter 4

Tools for Creating and Deploying Interactive Experiences

That's what fiction is about, isn't it, the selective transforming of reality? The twisting of it to bring out its essence?

Yann Martel, Life of Pi

Interactive experiences are rapidly transforming how we engage with information and entertainment across diverse fields, from education and cultural heritage to corporate training and marketing. Museums are no longer static repositories of artifacts; they are becoming dynamic spaces where visitors can interact with exhibits through augmented reality (AR), virtual reality (VR), and mobile games. Educational institutions are using interactive simulations to teach complex scientific concepts, and businesses are employing gamified training modules to enhance employee engagement. The common thread is the power of interactivity to deepen understanding, foster curiosity, and create memorable experiences.

The widespread availability of mobile devices has been a key driver of this transformation. Smartphones and tablets have become ubiquitous tools for accessing and interacting with digital content, making it easier than ever for individuals to participate in personalized, interactive journeys. Instead of passively reading text or listening to lectures, users can actively explore virtual environments, solve puzzles, and collaborate with others in real-time. This shift towards active learning and engagement has profound implications for how we design and deliver information across various domains.

While the potential for rich, engaging interactive experiences is immense, the development process often presents significant challenges. Creating com-

pelling AR applications, immersive VR simulations, or even sophisticated mobile games requires a unique blend of creative vision and technical expertise. Developers often rely on powerful game engines like Unity, Unreal Engine, or Godot, which, while incredibly versatile, can be daunting for professionals without a strong background in software engineering. These tools demand proficiency in programming languages, 3D modeling, animation, and a host of other technical skills.

This technical expertise gap frequently excludes domain experts who possess invaluable knowledge but might lack the programming proficiency to translate their ideas into interactive digital formats. Consider the example of a museum curator who wants to create an interactive exhibit to showcase a collection of ancient artifacts. The curator might have a deep understanding of the historical context, cultural significance, and artistic details of the artifacts. They might envision a treasure hunt that guides visitors through the museum, revealing clues and supplementary information via AR overlays on their mobile devices. However, without access to user-friendly authoring tools, the curator's vision might remain unrealized due to the complexities of game engine development.

Similarly, an educator who wants to design an escape room to teach complex scientific concepts might struggle to bring their ideas to life using traditional game development tools. The educator might have a strong grasp of the pedagogical principles and learning objectives, but they might lack the technical skills to create the interactive puzzles, 3D environments, and narrative elements that make an escape room engaging and effective. In conventional development workflows, bringing such ideas to life involves a painstaking process of manually configuring numerous interconnected game elements, writing code, and managing complex digital assets. This labor-intensive approach makes iterative design difficult and can significantly slow down the entire production cycle, especially when content needs frequent updates or adaptations for different audiences or temporary exhibits.

Recognizing this critical need, a central focus of our work has been to develop intuitive tools that support the creation and deployment of engaging interactive experiences across various domains, including education and cultural heritage. We aim to empower non-technical professionals to design and build their own digital adventures, allowing their domain expertise to drive the content without being bottlenecked by technical complexities. Our goal is to democratize the creation of interactive content, making it accessible to a wider range of creators and fostering innovation in how we communicate and engage with information.

In this context, our efforts have focused on developing systems that streamline the authoring process, allowing creators to concentrate on narrative design, educational objectives, and interactive elements rather than technical implementation. This has led us to investigate how such interactive

experiences can be systematically evaluated for their educational effectiveness, particularly in settings where specific learning outcomes are prioritized. We are especially interested in how well these games communicate complex concepts and in identifying complementary strategies to ensure players meaningfully achieve their learning goals. To this end, we have begun integrating artificial intelligence methodologies into game design tools, with the aim of supporting designers in the agile creation of structured interactive experiences from simple input specifications. Our focus lies in ensuring that these AI-generated elements remain factually accurate, controllable, and structurally coherent throughout the design process.

In this chapter, we will delve into the challenges inherent in traditional interactive experience development, present our solutions designed to overcome these barriers, and explore how these tools can empower creators to deliver impactful and engaging digital content. We will start with a case study that illustrates the complexities and requirements of designing and deploying interactive experiences within real-world educational and cultural contexts in Section 4.1. This case study will serve as a foundation for understanding the practical issues faced by creators and motivating the solutions we propose to address them. Following this, in Section 4.2 we will describe EnigMachine, our ecosystem to develop intuitive authoring tools that enable non-technical professionals to create their own interactive adventures, thereby bridging the gap between domain expertise and technical implementation. Section 4.3.1 will focus on the evaluation of game experiences created with our tools with museum visitors, discussing methodologies for assessing the effectiveness of interactive content in achieving engagement goals. In turn, Section 4.3.2 will focus on analysing the usability of the EnigMachine Editor Tool, providing insights into how well it meets the needs of its intended users and the issues encountered during the validation process. Lastly, in Section 4.4, we will explore the potential of AI-assisted adventure design, discussing how artificial intelligence, and particularly large language models, can be integrated into the adventure design process to bootstrap the creation of different game elements, such as puzzles, narratives, and question blocks.

4.1 Designing interactive experiences for educational and cultural settings: the Enigma Bio case study

This section outlines the development of Enigma Bio, an educational escape room created in this thesis for the National Museum of Natural Sciences in Madrid (see Figure 4.1). As an initial phase of our broader research, this project offered valuable insights into the challenges of integrating advanced

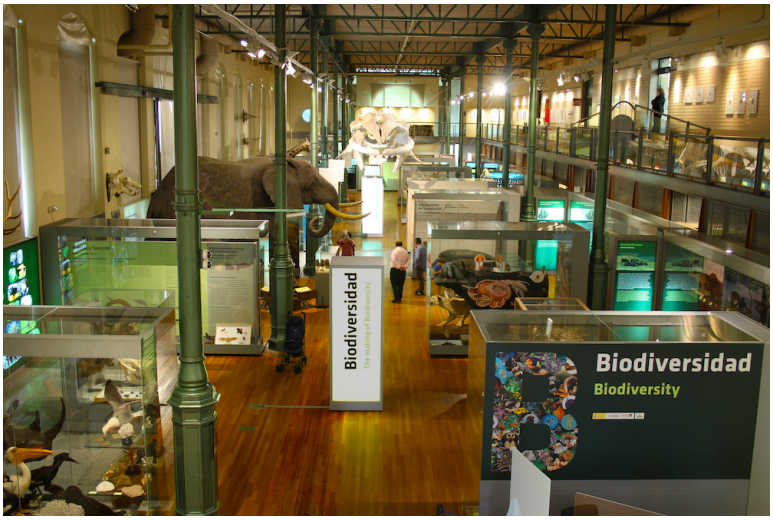


Figure 4.1: General view of the museum exhibition.

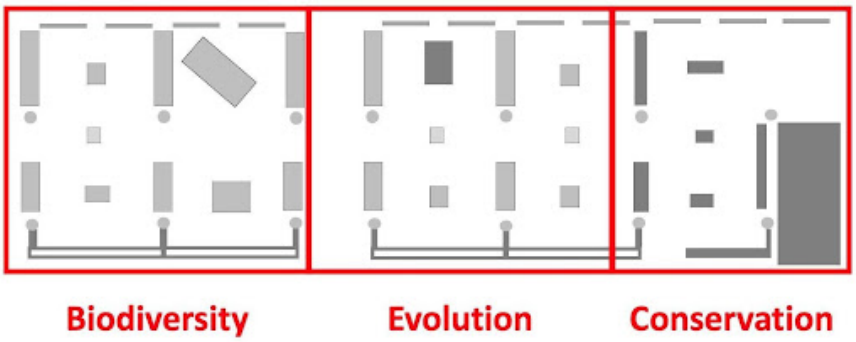


Figure 4.2: Exhibition floor map.

interactive technologies into museum-based educational activities. The experiences and lessons gained from ENIGMA BIO informed the subsequent development of more intuitive tools for crafting interactive experiences.

ENIGMA BIO was conceived as an educational escape room aimed at teaching children aged 11 to 13 about the abstract concept of biodiversity. The objective was to raise awareness of climate change and its impact on biodiversity. Designed for groups of 20 to 30 children, the one-hour experience ran on mobile devices using augmented reality (AR) technology. The game followed a structured thematic progression: beginning with the diversity of life forms, then linking these to natural selection and adaptation, and ultimately addressing threats to biodiversity—especially those related to human-induced climate change and biodiversity loss. Within a limited timeframe, the game sought to enhance children's understanding by leveraging group dynamics in a physical museum setting, guided by exhibition

Figure 4.3: Children interacting with the exhibition content.

content and the active presence of an educator as mediator.

As shown in Figure 4.2, while the areas in the exhibitions are not physically separated, the display cases define spaces that can be perceived as connected, with "doors" enabling passage between zones. In ENIGMA BIO each of these three "rooms" adhere to an open structure, allowing players to solve various puzzles simultaneously and without a specific order. ENIGMA BIO's core game mechanic centers on using a mobile device's camera to search for artifacts, using image recognition technology. Players advance through a narrative adventure by following clues, answering questions, and engaging in different stages. Throughout these stages, participants encounter narrative elements, dialogues, artifact hunts, and blocks that regulate player advancement.

An artifact hunt prompts participants to use their device's camera to locate an object indicated by a game clue. Successfully identifying the artifact completes the search. Usually, an augmented reality image or text will overlay the object, offering hints or additional information for the game. To encourage students to read the content of the explanatory panels in the exhibition, many of the object searches in ENIGMA BIO refer to details on those panels, as shown in Figure 4.3 where several children try to find a detail on the panel "the shapes of biodiversity".

During a room stage, the participant has the freedom to interact with all the sub-sequences of artifact hunts and puzzles positioned within the room in any order they choose. To progress to the next stage from a room, the user must solve a final puzzle, which will only be possible to solve successfully

Figure 4.4: Explanations of the educator during the game.

after having completed the rest of the quests hidden in the room. At the end of the room, the player will find a control stage in the game designed to prevent participants from progressing until they are given a keyword known only to the museum educator. This will allow to run the escape room in a synchronized manner for the group of participants, by allowing to wait until all participants have completed one room before granting passage to the next. And, more important, this will bring full attention of the children to the explanations of the museum educator, as shown in Figure 4.4.

4.1.1 Evolution of design and research questions

Earlier versions of our "Enigma saga" games, also developed for museums, adopted a treasure hunt format. These games used image recognition and AR on mobile devices to connect with museum exhibits. While successful in capturing children's attention, they often reduced the educator's role to a brief interaction at the beginning and end. This raised concerns about maintaining a balance between gameplay and educational guidance. Although the AR-enhanced treasure hunts were engaging, they frequently prioritized entertainment over learning, with children focusing more on solving puzzles than absorbing exhibit content—leading to the underutilization of the educator.

To address this, ENIGMA BIO transitioned from a treasure hunt to an escape room format in which the educator played a more central role by managing "gates" that controlled progression. This shift was in response to museum educators' request for a format that would allow the game to pause, creating moments for explanation and discussion. As players completed challenges within each zone, they regrouped at the educator's station, where connections between their discoveries and broader concepts could be

clarified. The strategically placed gates ensured consistent engagement with the educator, allowing for contextual explanations, question-and-answer opportunities, and reinforcement of key learning goals. Additionally, the escape room format fostered teamwork and collaboration, encouraging children to work together as they solved puzzles and advanced through the experience.

Our research within the Enigma Bio project was guided by a series of evolving questions that emerged as the project progressed through multiple iterations over several months:

- How could we introduce children to complex topics like climate change and biodiversity in a way that is both engaging and educational, and that integrates naturally into a museum visit using available technologies?
- How could the game serve as a tool for museum educators, enabling their explanations to be interwoven naturally between gameplay phases while maintaining the children's attention and interest?
- How could the game be made flexible enough to adapt to groups of children with diverse knowledge levels and skills, and how could it be usable by museum educators without requiring advanced technical expertise?
- How could we effectively collect data on gameplay and the impact of educator interventions to iteratively improve both the game and the associated educational activities?

Although formulated in the context of educational games, the research questions above resonate with the themes of Objective 1. In particular, the need to iteratively evaluate whether design goals are being met parallels the focus on quality assurance in entertainment games, while the collection and interpretation of gameplay data to refine educational activities closely aligns with player modeling. Similarly, the emphasis on creating tools that educators can use without technical expertise reflects the same end-user empowerment that drives both objectives. These shared concerns highlight the continuity between the two research axes, despite their application to different domains.

4.1.2 Design iterations and findings

4.1.2.1 First iteration

In November 2021, we collaborated closely with museum educators to co-design the game's script, including questions related to consumption habits. We worked with six school groups—almost 120 participants aged 10-12 (fifth

and sixth grade) and 13-14 (second year of secondary school). To foster greater engagement from museum staff, we invited them to select the game's protagonists. They chose two of their colleagues, Fernando Valladares and Pilar López, to be featured within the experience. Key takeaways from this iteration included:

- We introduced a mechanic that required players to locate the educator and wait for a code to progress. This proved effective in coordinating group focus and channeling attention toward the educator's explanations.
- While the concept of biodiversity was successfully introduced through the educator's input, we found it too complex to convey fully within the limited duration of the game.
- Multiple-choice questions were embedded during game pauses—after educator explanations and before providing the next code. These questions successfully captured children's attention and enabled us to gather useful feedback.

4.1.2.2 Second iteration

In November 2022, we extended the co-design process to include schoolteachers, creating a more holistic learning experience. We worked with three school groups, comprising nearly 60 participants aged 10-11 (fifth grade). This iteration introduced a structured three-part experience:

1. **Pre-visit Session**: A few days before the museum visit, we conducted a talk introducing biodiversity, its link to evolution, and the current threats contributing to biodiversity loss. This addressed a key research objective: determining whether a pre-session could enhance the educational impact of escape room activities with limited on-site duration. We hypothesized that without this introduction, the game's playful elements might overshadow its educational intent.
2. **Museum Visit and Game**: During the visit, the children played the treasure hunt game, alternating between puzzle-solving phases and educator-led explanations. The educator contextualized topics such as biodiversity, evolution, and extinction, concluding the visit with two contrasting species case studies: the Iberian lynx—a conservation success—and the thylacine—an extinct species lost to hunting and habitat destruction.
3. **Post-visit Activity**: Returning to school, students completed a reflective exercise. Each child chose either the lynx or the thylacine and wrote and illustrated the animal's story.

This integrated approach proved highly effective. The pre-visit session laid a solid conceptual foundation, which the museum visit reinforced, while the post-visit activity encouraged reflection and retention.

From a technical standpoint, however, this iteration exposed limitations in the game's development workflow. Editing game elements or adjusting content through the Unity editor was time-consuming and inefficient, especially when managing multiple game versions for different institutions or participant groups. This underscored the need for more user-friendly authoring tools that would allow educators to customize content without technical expertise.

4.1.2.3 Third iteration

In February 2023, our focus shifted toward refining the game mechanics, particularly for older players. We collaborated with four school groups — approximately 100 participants aged 12-15 (first and third year of secondary school). The goal was to introduce more complex challenges and to begin testing our first authoring tools, which are described in later sections.

Within the span of a week, we developed and tested three game versions. The final version featured an enhanced gameplay structure integrated with a 15-question survey on energy use, dietary habits, and broader consumption behaviors. Notably, this version replaced the traditional sequential treasure hunt with an escape room format consisting of four parallel puzzles. Solving these provided clues required to "escape" the room, allowing for increased flexibility and collaboration.

Results from the A/B testing used in the second iteration showed a moderate overall increase in environmental awareness among both control and experimental groups. However, the data indicated that students who received the pre-visit session demonstrated a stronger grasp of the intended messages. This supported our hypothesis that while the game could independently promote learning, an introductory session significantly enhanced comprehension and engagement.

These findings underscored a broader insight: educational games can be powerful tools for learning, but they are most effective when embedded within a well-considered pedagogical framework. A blended approach — combining gameplay with preparatory instruction and post-game reflection — proved more impactful than relying on the game alone. As such, Enigma Bio is best positioned not as a standalone learning activity, but as part of a broader educational experience designed to maximize both engagement and understanding.

Paper 7 (published in *Electronic Journal of e-Learning*) presents the details of the design iterations introduced in this section to address Objective 2.1, including a more in-depth description of the findings listed above,

alongside the statistical analysis supporting our claims. The following constitutes a summary of the main contributions of this work:

1. We introduce ENIGMA BIO as an enhanced version of the enigma saga's treasure hunt games, infused with escape room elements to enrich the puzzle-solving experience. By incorporating educators into the gameplay mechanics, we have transformed the game into a valuable tool for engaging students.

2. We gather preliminary evidence suggesting that prior exposure to relevant concepts, such as biodiversity, could amplify the game's impact. Notably, the experiment highlighted the positive influence of a presession introduction on the efficacy of short educational games in helping primary school children understand complex concepts.

3. We describe the design and development process of ENIGMA BIO, detailing the iterative approach taken to refine the game mechanics and educational content. This includes the challenges faced in balancing gameplay with educational objectives, and how these were addressed through careful design iterations, which provided a set of guidelines for future serious game development.

4.1.3 From Unity to custom authoring tools

Unity has long been our tool of choice for creating these applications. To separate the technical aspects, handled by programmers, from the creative effort of crafting adventures, handled by designers, we rely on scriptable objects. These mechanisms allow Unity programmers to define new types of assets—beyond maps, textures, models, or sounds—that designers can then manipulate to shape the entire interactive experience.

Despite being a highly valuable tool, the Unity editor is not particularly user-friendly when the number of these scriptable objects grows and their relationships become complex. While conceptually assets from an execution point of view, scriptable objects are often very close to how programmers view the game, frequently exposing superfluous technical details. When designers devise an interactive experience like a treasure hunt or escape room, their approach is at a much higher level. Having to specify it through a myriad of small, interrelated scriptable objects in an environment not tailored for this purpose makes their task much more daunting. Additionally, this forces them to have a Unity installation and to become minimally proficient in an environment that provides a vast array of functionality. All of this generates a significant amount of noise for their more restricted task.

Lastly, in our context, this way of using Unity for creating these experiences introduced an additional logistical problem. Over time, we have created many versions of our different adventures to adjust them to specific

museum initiatives (e.g., science weeks, nights at the museum, school visits) or to temporary changes in exhibitions due to artifact loans or restorations. Most of the core foundations of our games (i.e., programming and art) are common to all of them and are collectively known as EnigMachine. With an increasing number of adventures (created using different scriptable object configurations), version control of so many slightly different applications quickly became a struggle.

To address these problems, in recent years, we have developed an authoring tool external to Unity, which we call EnigMachine Editor. This tool facilitates the creation of these interactive experiences. It is essentially a web application where designers can write and specify an entire adventure, providing a clear, distraction-free interface directly within the browser, making it usable from any device. Each adventure is managed in a distinct project within the application and follows a lifecycle completely independent of the Unity project that will later run the experience. The web application also provides an export mechanism that creates the scriptable objects EnigMachine expects, enabling the automatic creation of the final mobile application.

Aside from significantly improving workflow, facilitating version control, and removing the need for designers to interact with Unity, the time required to create each experience has also been greatly reduced. An informal analysis with designers familiar with the nature and inner workings of the experiences created determined that, once an adventure is specified, entering it into the system went from requiring a few days to mere hours in most cases. This is especially valuable given that a substantial portion of the testing and quality control process for these applications must be carried out on-site, outside the development space, with the consequent logistical limitations this entails.

4.2 The EnigMachine ecosystem

EnigMachine is a system for developing mobile applications that support augmented reality-enhanced treasure hunts and escape rooms in museums and other spaces. We collectively refer to these applications as *adventures*. Curators and enthusiasts can specify a set of museum tour phases in a web environment, which visitors can then enjoy on-site via a mobile app. At a high level, the whole process can be summarized as follows:

- Content creators log into the web application and, using their user account, build an adventure using the tools provided by the platform.
- Once the adventure is completed, the platform generates a mobile application that the user can download.
- The application is installed on the visitors' mobile devices (either per-

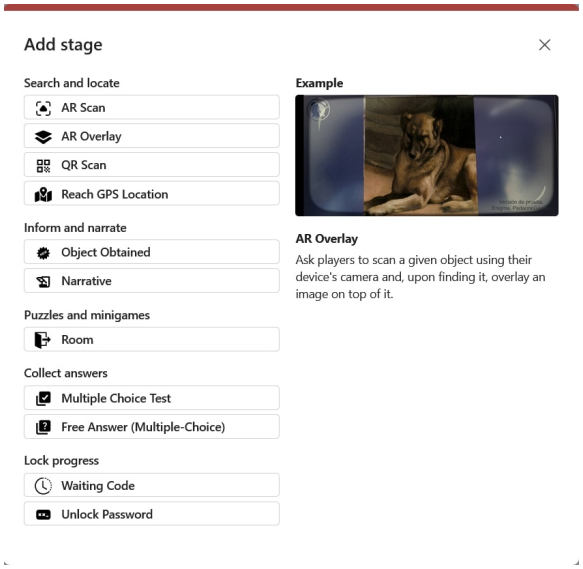


Figure 4.5: Selection menu listing currently available phases and their categories.

sonal or museum-owned).

- Visitors use it during their tour.

The model of creating content through the web to be used by other people is not new and is being applied to platforms such as *Kahoot!*[1] and *Canva*. EnigMachine distinguishes itself by being specifically designed for the creation of treasure hunts using augmented reality in museum environments, adapting its components to the needs of this context.

Our web editor application allows the creation of interactive adventures in an accessible way for users of all levels, including those with no technical or programming knowledge. Following the same approach as some of the above-mentioned content creation tools, our tool organizes the structure of the visit into sequences of building blocks depicted by "slides," representing the phase layout of the game. In the following, we use the terms *phase* and *stage* interchangeably to refer to these building blocks of an adventure. The 11 types of stages available are grouped into different categories based on their underlying design intent (e.g., finding an artifact from the exhibit, providing information, collecting participant responses, etc.), and can be found in Figure 4.5 in a screenshot of the "Add stage" selection menu.

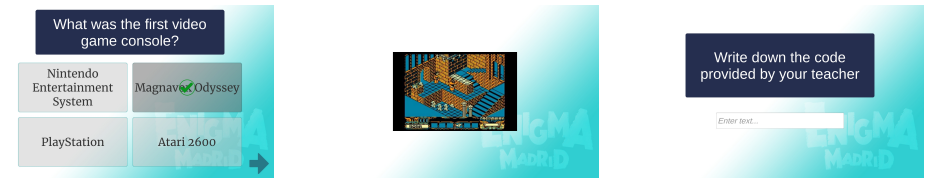
As an example, within the *Collect Answers* category, adventure designers can promt users with questions whose answers are recorded for later analysis. These questions can be related to the adventure itself, such as awarding

[1] https://kahoot.com/

points to encourage competition among participants, or they can be more open-ended questions designed for opinion or learning analysis. This approach is particularly useful when the treasure hunt is created by a teacher for a group of students or by curators for activities with specific educational objectives.

4.2.1 Phases and structure

(a) *Narrative* Phase (b) *AR Scan* Phase (c) *AR Overlay* Phase



(d) *Multiple Choice* Phase (e) *Object Obtained* Phase (f) *Waiting Code* Phase

Figure 4.6: In-game screenshots of most common phase types in EnigMachine

As previously mentioned, in EnigMachine a treasure hunt is known as an *adventure*. The basic meta-information for an adventure includes its name, a brief description of its content, its author, and information specific to the mobile application itself, such as the icon to use, the id of the generated application, or its version number.

It also defines the *virtual characters* that appear in it, specified via their name and an image that represents them. Beyond that, an adventure is, in principle, a linear succession of *phases*, specified by the designer, that users have to overcome.

To create a common thread in the experience, one of the phases is "narrative." In this stage, virtual characters provide explanations or present the next challenges to solve. Except for this type of phase, which is mainly passive from the user's point of view (only having to read), the remaining phases generally require the player to engage in more elaborate actions.

Typically, the design guideline is to ensure that interacting with museum artifacts or the environment is necessary to progress through the phases. This can be achieved either directly by blocking progress after a seeking and

scanning stage of some kind of physical element of the exhibition (artwork, panel, QR, etc.) or in more subtle ways, for instance, by means of quiz-like questions in which the user must give the correct answer to score points, ideally by having to search for the relevant information by carefully studying a physical element of the museum. Ultimately, the intention is to enable designers to make use of the phases available to guide visitors to look for answers in the vicinity of the current location in the adventure.

We will now provide a more detailed description of the most important phase types.

4.2.1.1 Narrative

As indicated earlier, narrative-type phases aim to provide textual information to the player, leading to or setting the next challenges for them. To specify a narrative phase, the author lists the different sentences that comprise it and which character says each of them. This makes it possible to convey the feeling of a dialogue between two characters conversing with each other. Most often, however, these are used so that the character acting as a "virtual guide" speaks directly to the player, pointing out the last goal that they have accomplished or the next challenge that they will need to overcome.

During the course of the adventure, the game displays each of these lines in sequence, together with the image of the character who is speaking them, as seen in Figure 4.6a. Meanwhile, the player simply moves on to the next sentence once the previous one has been read.

4.2.1.2 Augmented reality scan

To clear these stages, the player is prompted to point their device at a certain object in their environment. The application then switches to search mode while displaying a reminder message of what needs to be done. Upon aiming at the right object, the game plays an effect indicating that the target has been detected and moves on to the next phase, as depicted in Figure 4.6b.

The creator of the adventure chooses what needs to be targeted by uploading a picture of the item and a hint to be displayed within the web application. Ideally, the objective of what is to be done should be explained with a narrative prior to this phase. For instance, in order to encourage the visitor to carefully analyze the content of a given artwork, the application could request that they point to a character with some specific characteristics or to the largest animal that appears on the canvas. This can also be used to enforce a path through several works by asking visitors to locate the paintings of a certain author fulfilling a certain condition, such as including a person with a sword.

Our experience in creating treasure hunts has consistently shown that

this is one of the phases that provides the most engagement for users.

4.2.1.3 Augmented reality overlay

The previous phase type does not take full advantage of the possibilities of augmented reality, as it is limited to recognizing the item specified by the designer. The *AR Overlay* type phases (Figure 4.6c) go a step further in that once the target element has been found (a picture, a poster, etc.) the application superimposes an image on top of the physical element that is being displayed on the mobile screen captured by the camera.

The adventure designer can choose the image to be superimposed on the physical element. A common approach is to reveal a piece of text, providing a clue regarding the next step to be taken. It is also possible to magnify a section of the artwork being targeted, to show hidden content, such as characters previously removed by the author of a painting that have been discovered when analyzing the picture, or to add speech bubbles with text spoken by the characters that appear in the artwork so as to integrate them with the adventure's narrative.

4.2.1.4 Quizzes

Collect Answer type phases, available in formats with or without correct answers (*Multiple Choice Test*, shown in Figure 4.6d, and *Free Answer*, respectively), are designed to encourage participants to look for information on the museum information panels, artifacts, and artworks. Additionally, they can be used to gather feedback from visitors. The submitted responses are recorded and stored for future retrieval, although the infrastructure for this service is still in the development stages and is not yet generally available.

4.2.1.5 Object obtained

Treasure hunts are, in essence, games "in which organizers prepare a list defining specific items that participants must collect or complete" (Wise and Forrest, 2003). Creating the sense of progressing through stages and obtaining items is crucial for fostering the illusion of advancement.

Although EnigMachine currently does not feature a built-in inventory system, it provides designers with *Object Obtained* phases. During these, the player is shown an effect that suggests that "an achievement has been unlocked" or "a goal has been achieved," as seen in Figure 4.6e. This phase is passive from the player's perspective, who only receives feedback indicating that a new item has been obtained. Players may not even regard this as a true phase, as it is executed automatically and without interaction. However, from the designer's perspective, achieving such an effect requires deliberately adding this dedicated phase to emulate progress.

Figure 4.7: Structure of a puzzle block within a *Room* type phase.

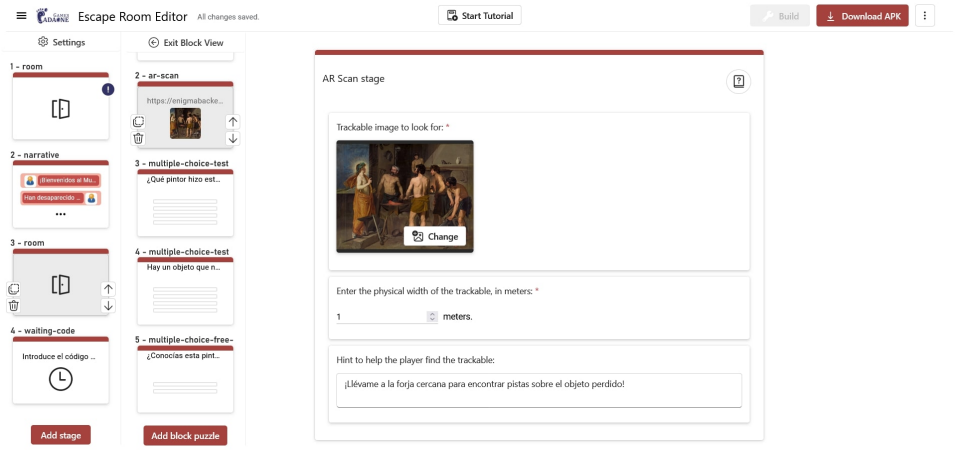


Figure 4.8: Screenshot of the *Room* type phase editor.

Despite the absence of an inventory, these phases give the player the impression of gathering items. Designers can then incorporate the use of these items into later narrative stages by having characters mention and suggest their use. For example, an adventure could start with a quiz phase that, when completed, awards the user an "X-ray camera" to discover hidden messages in paintings, thereby justifying subsequent AR overlay phases. This approach exemplifies how designers can craft engaging experiences that appear dynamic to players by blending passive feedback with narrative integration to maintain the flow of the adventure.

4.2.1.6 Rooms

EnigMachine features a structure where adventures primarily progress through a linear sequence of stages, albeit with some flexibility introduced through phases of the *Room* type. These stages emulate the mechanics of *escape rooms*, a popular game format where players must "discover clues, solve puzzles, and accomplish tasks in one or more rooms in order to accomplish a specific goal in a limited amount of time" (Hall, 2021). Such gameplay

dynamics naturally complement treasure hunts and hold significant cultural heritage value.

In EnigMachine, *Room* type phases simulate an escape room scenario, challenging players to solve multiple blocks of puzzles to advance and eventually exit the associated space. The order in which puzzle blocks in the room are tackled is generally non-sequential and not relevant by design; however, all puzzle blocks must be solved to progress to a final challenge that holds the key to unlock the room's exit.

Each puzzle block in one of these phases represents a linear series of any of the previously described stages available in EnigMachine, effectively branching the adventure through these nested blocks into smaller sub-adventures (as depicted in Figure 4.7 with a preview of the 8 sub-stages that constitute one of the 4 blocks in a sample room). This branching is nonetheless limited in depth, as room puzzle sets are not allowed to incorporate additional room phases within them. The web editor handles this nested stucture via an additional column of slides representing the sub-stages in the selected room block, as seen in Figure 4.8.

From a design perspective, each block (sequence of stages) is encouraged to culminate with an "object obtained" stage to signify progression in the room escape scenario. The final puzzle block unlocking the room can integrate narrative elements informing players that their use of acquired objects has allowed them to escape successfully.

A notable advantage of *Room* type stages is their ability to induce player dispersion. In scenarios involving groups, such as children in school visits, the inherent linearity of adventure phases often leads to a degree of collective exploration for items and clues in which players may opt to observe others' actions before proceeding similarly.

Room type stages disrupt this linearity by empowering players to determine the order in which they tackle blocks within a room. As an additional dispersion strategy, EnigMachine offers the option to randomize the order of available room blocks. This feature ensures that even if players choose the default left-to-right, top-to-bottom approach, they are likely to disperse across the environment, particularly when the puzzles included target distinctly located objectives and artifacts.

4.2.1.7 Waiting code

In this type of phase, the application displays a short message to the player, who must simply type in a secret code, usually a single word (Figure 4.6f). While this mechanism can be used to motivate the player to solve puzzles by searching for information in the environment, its primary use is to control the application when it is being used by a group of schoolchildren on a visit to the museum. In this context, it is common for the visit to be supervised

by an educator who, at specific times, may wish to regroup all the students to provide additional information or encourage some form of group debate. This can be accomplished by incorporating one of these phases asking for a code known only to the educator, who will openly disclose it only after all the students have reassembled and engaged in the educator-led activity outside of the adventure.

When the application is used with museum devices, it is important to disable the predictive keyboard, as this will remember the most frequently used words, which could make discovering the secret code trivial after a number of visits.

4.2.2 Resulting application

Starting from an adventure specified through the web editor, the platform generates an application that the designer can then download and install on a mobile device (i.e., a phone or tablet). This application is fully self-contained in the sense that it does not require an Internet connection in order to be run and includes both the adventure and all of its associated resources.

When the application is opened, the adventure is immediately executed in the first phase specified by its author. Throughout its use, players progress through each of the stages, reading the texts in the narratives, searching for the physical artifacts in the AR stages, or answering the questions in the quizzes. The game collects usage data and logs events to an analytics server that allows for further analysis of, for instance, average times in solving a given phase or answer success rates. If the device is not connected to the Internet, the events are buffered locally and submitted as soon as the Internet connection is restored. Data sent in this way do not contain any device-specific information, meaning that they are kept anonymous and it is virtually impossible to identify the specific user who generated them.

When the player completes the adventure, the application offers the choice to start the adventure over again, thus initiating a new session as far as the analytics are concerned. This facilitates the use of applications developed for museum-loaned devices; when, for example, a student returns a borrowed tablet to a museum educator, the next student who takes the same tablet can proceed to play the adventure immediately.

4.2.3 Ecosystem and architecture

As noted at the beginning of the section, curators and hobbyists can specify their adventure within a web application that allows for the download of an installable app on mobiles and tablets once completed.

The web application featuring the adventure editor (described in more detail in section 4.2.3.2) is comprised of a *backend* developed in Node.js that handles the management of both users and the adventures created, and

a *frontend* leveraging the React framework to build the application's user interface in the browser.

Adventures are serialized in JSON format and stored in the backend. Each `.json` file consists of a first section containing the meta-information of the adventure and its characters. This is followed by the sequence of game phases with all their relevant parameters, following a format specifically adapted to the requirements of each stage. The adventure resources (i.e., character images, photos of the physical objects to be found in the AR phases, etc.) are hosted on the server with an assigned unique identifier, which the JSON file references via URL.

The installable mobile application is built with Unity3D, a multiplatform game development engine with support for augmented reality and geolocation. For the development of video games on Unity3D, a development platform known as *Unity Editor* is used. This is where programmers, designers, and artists work together to create the content of the game and, once finished, make the executable files for the different platforms. The Unity Editor can be further extended by means of editor code. Despite the fact that this does not have an effect on the final builds, many development teams rely on this feature to create, for example, automation tools to be integrated into the development environment in order to increase their productivity.

In the context of EnigMachine, the objective is for the final application created with Unity3D to be built automatically without the need for human intervention. Furthermore, the final application should be self-sufficient in such a way that, once installed, it does not require a connection to the network. This means that the application must fully contain the created adventure, along with all of its resources.

To achieve this, in addition to the web application, the infrastructure around EnigMachine incorporates a *builder* responsible for the assembly of the final applications (section 4.2.3.3). This acts as a daemon that comes into operation when a user makes a request from the web application to build the final application for their adventure. The builder then retrieves the adventure's `.json` file from the backend and initiates the generation of the mobile app. Once ready, the app is sent back to the backend, which in turn makes it available for download to the user. As the entire build process can potentially last for several minutes, users are shown a dynamic feedback message on the web indicating that their application is in the process of being built during this time.

All the applications built from the different adventures are created from a Unity3D template project that contains the *Adventure Player*. This contains core resources shared among all adventures (i.e., scenes, UI images, augmented reality plugins) and the source code that is responsible for providing the underlying execution logic. It also constitutes what is known as the EnigMachine *Engine*, which interprets the phases included in the

adventure and manages their execution (Section 4.2.3.1). For the EnigMachine Engine, an adventure is seen as *data*, meaning that running a different adventure does not require any kind of additional coding or adaptation of the Unity project (i.e., creating new scenes or adding new scripts). The only configuration required is to specify the sequence of phases to be used in the adventure.

Although, as previously mentioned, the backend serializes the adventures in JSON format, Unity3D provides a native mechanism for serializing resources that is better tailored to its internal needs in the form of what are termed *scriptable objects*. EnigMachine Engine does in fact expect adventures as resources serialized through these scriptable objects and not through JSON. As a result, one of the tasks of the builder is to perform the conversion from the JSON format used by the backend to the scriptable objects used by the Unity project. Once the transformation is completed, the Unity Editor is run as a batch process to generate the project's build.

While the EnigMachine Engine could easily handle the conversion from JSON format to native Unity resources at runtime in the final application, delegating this step to an offline process performed by the builder offers the advantage of making the end game more efficient and decoupling the EnigMachine engine from the rest of the infrastructure. Notably, it is possible to create adventures directly from Unity using the template project with its entire infrastructure and relying on the tools provided by Unity itself to handle assets and resources stored in the native scriptable object format. Even though this process does require some Unity and programming knowledge, it provides greater customizability of the adventures by allowing the modification of elements that cannot currently be changed from the web adventure editor, such as background images, fonts, or visual components, or even the addition of new types of phases not included among the standard ones described in section 4.2.1.

Figure 4.9 illustrates the general operation of EnigMachine with its three main building blocks: the editor, the builder, and the final application. In the following, each element is detailed separately.

4.2.3.1 EnigMachine engine

As noted earlier, EnigMachine Engine can be understood as a Unity3D project that contains the code and resources necessary to be able to run an adventure game. These include, among others, an augmented reality plugin to activate the camera and detect the *trackables* specified in the *AR scan* type phases, or UI elements with position markers to render character portraits and the sentences they say in narrative phases.

Adventures are specified as scriptable objects. In Unity, scriptable objects are data containers that can be used to store data independent of class instances. From the engine's point of view, they are resources in the same

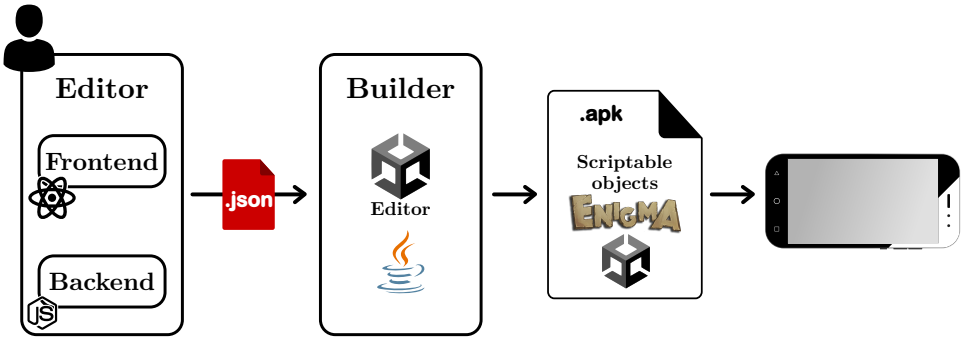


Figure 4.9: General architecture and workflow.

way as a scene, a 3D model, or a texture. Moreover, the Unity Editor is designed to provide a user-friendly approach to the specification of scriptable objects.

From an implementation point of view, each type of phase supported by the EnigMachine Engine requires the specification of a type of scriptable object with the parameters that define it and the scripting of a *player* that interprets and runs it. Most of those players need visual resources. For instance, the player of the *Narrative* type phases requires UI elements including position markers for the character images and dialog text, as discussed previously. To maintain the modularity of the Unity project, the requirements and dependencies of each player are kept in an independent scene, so that, in order to incorporate a new type of phase, one must set up the scriptable object that serializes it, the player that executes it, and the scene that contains the graphic resources that it needs. This modularity allows, when appropriate, to eliminate from the Unity project the triplets of scriptable object, player, and resources associated with those phases that an adventure does not use, thus reducing the size of the final build.

The adventure as a whole is also stored in a scriptable object that contains references to the scriptable objects of all the game's phases and the meta-information related to the characters. Following the pattern discussed above, there exists a global player responsible for the execution of a complete adventure, which is tasked with moving it forward by activating the specific player that understands how to interpret the next phase's scriptable object. The scenes with the resources of each player are loaded additively when required, with the scenes that are used the most usually being kept preloaded to speed up the execution.

Scriptable objects keep *the model* of the adventure, while scenes and their resources serve as *the views* and, finally, the players function as *the controllers*, keeping a loose relationship with the well-known model-view-controller software design pattern. Views are loaded from the Unity scenes and reused across phases of the same type, whereas controllers are created

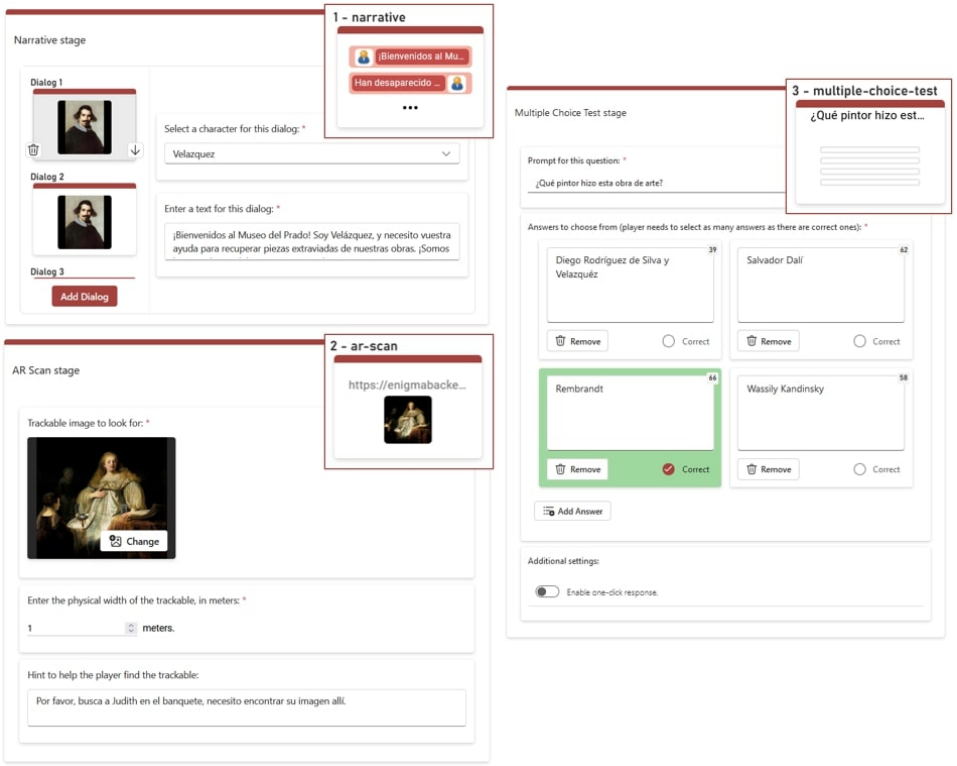


Figure 4.10: Configuration components and preview slides for different phase types.

on demand by the adventure global player. When controllers are created, they must find their views. Dependency injection is used to avoid having to look for them while browsing through Unity scenes.

EnigMachine Engine is, as can be seen, an independent element from the rest of the web-based adventure creation ecosystem. As already discussed, it is possible to create adventures entirely from Unity and create end builds directly with it without ever interacting with the web editor or the builder. This naturally requires knowledge of programming, Unity, and how the EnigMachine Engine works internally, but in return, it provides customization or extension options that are not otherwise supported in the web editor. This is how early adventures were created and continues to be used to this day in the development of the engine.

4.2.3.2 Editor

While the infrastructure described above is quite powerful and allows for great flexibility in adventure creation, in practice, it is often a time-consuming and error-prone process in which a designer with technical knowledge of

Unity iteratively assembles the adventure from within the engine. The technical aspect of this process takes away some of the emphasis from the design, and the lack of a graphical planner makes it difficult to get a general sense of the structure and flow of the adventures. Furthermore, the EnigMachine Engine does not provide native version control or user management systems, making it difficult to create new variations of earlier adventures that coexist with previous ones but include modifications specifically designed for new use cases.

Due to these concerns, we decided to create a web editor designed specifically to address these issues and simplify the design and development cycle of new adventures, drawing inspiration from other popular content editing tools such as *Power Point*, *Kahoot!*, or *Canva*. Our web application was conceived as a single-page solution with React as the primary frontend framework and a supporting server implemented in NodeJS for managing adventure data, users, and metrics. The EnigMachine Engine's modularity extends naturally to the editor side, where each type of phase supported by the EnigMachine Engine is materialized into three fundamental pieces that must be specified independently to enable its edition on the web application: a data schema, an editing component, and a preview component.

A JSON data model or schema defines each of the existing phases of the engine for which we wish to provide support in the web editor. It is important to note here that this does not automatically imply a one-to-one correspondence with the definition of the scriptable objects that specify phases at runtime. For one, not all parameters supported by an EnigMachine Engine player need to be exposed on the web interface. During engine development, it is not uncommon to allow the configuration of parameters that are too low-level and would cause too much noise for the designers. In that case, the builder (Section 4.2.3.3) is responsible for "filling the gaps" and setting the values of scriptable objects that are not specified in the JSON. In fact, the builder may make decisions during the conversion process based on the sequence of adventure stages, adding specific settings for one stage based on what the next stage is, such as when an AR scan is followed by an AR overlay with the same recognized physical objects, where the transition settings are explicitly modified to produce a more natural feel between stages.

On the other hand, some engine phases with a high degree of customizability can be represented by multiple types of editors in the web application, improving system usability and providing designers with more intuitive abstractions. This is the case, for instance, of the multiple-choice question phases, which, although on the EnigMachine engine side they are implemented as a single-phase type with a large number of parameters (correct answers, maximum and minimum number of answers that the player should select, etc.), on the web editor side they end up translated into two con-

ceptually different phase types with complementary functions: the Multiple-Choice Test phase (a question with a correct answer) and the Free-Answer phase (an opinion or reflection prompt). Although this process sacrifices some customizability in favor of more rigid structures with automatic parameters, we have seen in practice how this can prevent errors and reduce the gap between the designer's mental model and the final result in the adventure.

Moving on to the second point, for each JSON schema that is registered as supported in the web editor, the application structure requires the programmer to implement a React editor component that accepts an object that conforms to that schema as an input parameter, as well as a function that allows changes to be made to the state of that object based on the user's interactions. Again, these editors can be one-to-one representations of JSON fields in simple cases, but they are free to provide alternative mappings to conceptually simpler representations. Returning to the multiple-choice question example, while the associated JSON includes arrays containing the indices of the correct answers, the editor component represents what is correct as a property of the answer itself. Similarly, each editor can impose different constraints on the states they manage, such as text lengths or item counts in a sequence. Lastly, each phase schema must be accompanied by a slide component capable of displaying its preview in the application's layout panel. Figure 4.10 depicts some examples of editing components and their preview slides.

4.2.3.3 Builder

The *builder* is the daemon in charge of collecting the requests triggered by the users from the web editor to create the final applications for their adventures and to generate the downloadable builds that can later be installed on mobile devices.

It consists of a Java application that periodically polls the backend to check if any new request for the creation of a build has been received. Thus, the backend is the system in charge of monitoring all pending requests and assigning a priority to them. When a request is received, the daemon launches the Unity Editor from the command line in *Desktop Headless Mode*, handles the construction of the build, and sends it back to the backend along with the *log* information of the whole process.

In order to generate the end applications, the builder needs the EnigMachine Engine project as a template into which to incorporate the scriptable objects converted from the `.json` obtained from the backend. This translation process is performed by the Unity Editor itself using editor code. In particular, the template project used by the builder includes EnigMachine Engine along with editor code that provides a number of automation tools to particularize the project with an adventure read from a JSON. When the

backend informs the builder Java daemon that there is a new application build request, it *does not* itself do the translation from JSON to scriptable objects but instead runs the Unity Editor in batch mode and launches some of the EnigMachine built-in automation tools as editor code to perform the job. More specifically, the build creation involves three separate runs of the Unity Editor from the Java daemon on the EnigMachine Engine template project:

1. The editor code receives the backend URL from which the adventure's JSON can be downloaded. The Unity Editor itself, running in desktop headless mode, retrieves this JSON and creates the scriptable objects by leveraging the editor API provided by Unity itself, which simplifies the creation of resources. Since this is editor code, the translation tools do not wind up in the final builds, and the process remains entirely offline. Upon completion, the template project places the adventure's scriptable objects in specific directories, and the downloaded JSON and associated resources are deleted.

2. The Unity Editor is restarted on the extended template project, and a script is executed to begin the process of producing the final application in the same way that it could be done manually in the editor. When finished, the daemon retrieves this build and returns it to the backend.

3. In a third iteration of the Unity Editor, all the resources created in the initial stage are eliminated from the project template to tidy it up and get it ready for the subsequent build.

For simplicity, all of the editor code run in the three executions is in fact embedded in the EnigMachine Engine so that it can also be invoked manually when using the Unity Editor in the manner described in section 4.2.3.1. Having the process of converting the JSON into scriptable objects be done directly from within Unity Editor rather than being done from the Java daemon itself offers multiple advantages:

- It is possible to use the Unity Editor API to generate the scriptable objects instead of having to reverse engineer the way in which Unity Editor writes the objects to files in order to create them.

- When expanding the EnigMachine Engine to incorporate a new phase type, the structure of both the scriptable objects and the associated JSON is designed in parallel, and the translation process takes place directly within the Unity project without the need to modify the Java daemon of the builder. Naturally, this also requires extending the web editor accordingly.

- It is possible to manually request that the scriptable objects of an adventure defined in the web editor be constructed from a local copy of the EnigMachine Engine and then perform additional customizations that are not available from the web editor. This eliminates the need to specify the entire adventure from within the Unity editor, which is far more labor-intensive and error-prone than specifying it through the web application.

The builder thus bundles the Java daemon code, the EnigMachine Engine template, and the version of Unity Editor, along with its plugins, that is used to construct the build. These three elements are packaged in a Docker container that is regenerated when any of them need to be updated.

While not tied to a specific research publication, the work in this section directly addresses Objective 2.2 of the thesis, acting as a foundation for our subsequent research contributions. The EnigMachine platform provides a comprehensive framework enabling non-technical practitioners to create treasure hunt and escape room games in educational and cultural heritage contexts, integrating various types of interactive phases and leveraging augmented reality to enhance user engagement. This work sets the stage for further exploration of user experience evaluation and the development of interactive experiences, as detailed in the following sections.

4.3 Evaluating the EnigMachine ecosystem

As established in Objective 2.3, a key aim of this work was to conduct a thorough user experience evaluation of both the authoring tools developed and the interactive AR games generated with them. To confirm that our authoring tool is sufficiently accessible to novice users, we proposed a study to test and validate the following preliminary hypotheses.

Hypothesis 4.3.1. EnigMachine enables the creation of adventures that offer a rich gameplay experience, in which players perceive high levels of competence, fluency, and satisfaction, and low overall stress.

Hypothesis 4.3.2. EnigMachine Editor is generally suitable for non-technical users wishing to build adventures from a previous design concept.

Hypothesis 4.3.3. EnigMachine Editor is also generally usable by non-technical users when designing adventures directly from the platform itself over an extended period of time.

Each of these hypotheses was tested through a series of experiments, which are described in the following sections. The first experiment, detailed in Section 4.3.1, focused on evaluating the player experience of games created with EnigMachine in a museum setting, addressing Hypothesis 4.3.1.

The second experiment, described in Section 4.3.2.1 and linked to Hypothesis 4.3.2, assessed the usability of the authoring tool when translating design intentions to actual adventures on the system by having participants implement games based on a given design document. Lastly, Section 4.3.2.2 presents a long-term evaluation of the authoring tool, where participants were tasked to work on their own adventures over an extended period of time to validate Hypothesis 4.3.3.

4.3.1 Game experience evaluation

The evaluation of the user experience was conducted through a series of sessions with participants who played a set of games created with EnigMachine in a museum setting. The primary goal was to assess how well the games engaged users, facilitated exploration, and provided an enjoyable experience. To this end, we employed a mixed-methods approach that combined quantitative measures of user experience with qualitative feedback from participants.

The quantitative evaluation was conducted using the in-game version of the Game Experience Questionnaire (GEQ) (IJsselsteijn et al., 2013), which is a widely recognized tool for assessing user experience across a wide spectrum of genres, interactive experiences and user types (Janßen et al., 2016; Marín-Lora et al., 2024). The GEQ captures various dimensions of gameplay experience, including immersion, fluency, positive and negative affect, challenge, tension, and anxiety, and was administered to participants after they completed the adventure assigned to them. Additionally, information was collected on users' previous experience with escape rooms along with basic demographic data.

The qualitative evaluation involved collecting open-ended feedback from participants regarding their experiences with the game. Participants were encouraged to share their thoughts on the most and least enjoyable aspects of the game, as well as any suggestions for improvement. This qualitative data provided valuable insights into user preferences, challenges faced during gameplay, and areas where the game could be enhanced.

Regarding the design of the experiment itself, two variations of an adventure game set on the third floor of the García Santesmases Museum, dedicated to the history of computer science and located at the Faculty of Computer Science of the Complutense University of Madrid, Spain, were constructed. These were then administered to different groups of participants working in teams. The adventures designed comprised a total of 8 blocks of puzzles, making use of most of the types of phases available in the authoring tool. Each of these puzzle blocks required the careful inspection of at least one artefact or collection of artefacts in the museum and were distributed in two *Room*-type stages, each confined to one of the wings of the area where the activity took place. As mentioned in Section 4.2.1.6, the

4 blocks in each room could be completed in any order, being shown to the players in a random arrangement in each game.

The experiment was carried out across multiple sessions with a total of 62 participants, including 17 females, 44 males, and one person who chose not to answer the question. The mean age was 24.58 years (standard deviation = 3.05; minimum age: 19 years, maximum age: 34 years). Among the participants, 61.3% had little or no experience with escape rooms, 29% had moderate experience, and 9.7% considered themselves skilled players. Advertisements targeting our university's design, programming, production, and art students were used to recruit participants, and their involvement was scheduled into one-hour blocks of museum visits.

In each of these sessions, participants were randomly split into teams of 2 to 4 people, each receiving a tablet with the pre-installed game, as well as sheets of paper and writing pens to take notes if they deemed it necessary. Initially, given that the museum can receive several groups of visitors at the same time, the simultaneous participation of multiple groups in the same time slot was allowed in cases where time restrictions required it. However, to avoid overcrowding of players around certain key points in the museum or having groups blindly following others to copy their answers, alternative versions of the game were administered in these sessions, with randomised sections and distinct starting points for each team.

Once the equipment and materials had been prepared, players were free to interact with the game in the corresponding section of the museum, with a one-hour time limit, after which they were instructed to return to their starting point to conclude the activity. These sessions were designed with competition in mind, by announcing that the first players to complete the escape room would receive a small reward in order to encourage immersion and rivalry between teams. After finishing the adventure, users were asked to submit the 14 items of the In-Game GEQ questionnaire.

Participants evaluated each item using a five-point Likert scale (0 = complete disagreement, 4 = complete agreement). Additionally, they were invited to provide open-ended comments regarding the most and least enjoyable aspects of the experience, as well as a qualitative assessment of the session overall. Participation was primarily driven by the opportunity to engage with an augmented reality escape room in a museum setting, with small prizes awarded to the first teams to complete the activity.

Responses generally indicated a positive reception. The majority of participants reported enjoyment, with low levels of stress and frustration, and consistently high feelings of accomplishment and satisfaction. Nonetheless, certain aspects emerged as areas for potential improvement—most notably, immersion and narrative engagement. Several players expressed a preference for more concise textual content and a greater emphasis on puzzle-solving over storytelling. Others recommended the integration of multimedia ele-

ments, such as video and audio, to enrich the narrative dimension.

Some issues were also associated with the physical deployment of the experience. When multiple teams converged on nearby puzzles, crowding occasionally disrupted the flow and focus of play. In terms of usability, participants noted limitations such as the inability to revisit previous narrative messages. A few users also commented on the simplistic nature of the visual and auditory design, suggesting that this slightly diminished the overall sense of immersion.

Despite these observations, the overall experience was viewed as engaging and enjoyable. The findings suggest that the core design effectively achieved its aims, while also identifying specific opportunities for future refinement.

Lastly, an exploratory analysis was conducted to assess the influence of prior escape room experience on players' perceptions. Results revealed a significant difference in the perceived level of challenge: experienced participants found the activity somewhat less demanding than those with no prior experience.

Hypothesis 4.3.1 was confirmed. Adventures created with the EnigMachine system exhibited strong outcomes in terms of positive affect, flow, and competence, while maintaining moderate challenge and low levels of stress and negative affect, suggesting that the system successfully supported enjoyable and meaningful experiences. The most prominent areas for improvement related to enhancing sensory and imaginative engagement—either by reducing spatial congestion or through more sophisticated multimedia elements.

4.3.2 Usability evaluation

Authoring tools are a critical component in the creation and study of interactive experiences, shaping not only the design process but also the resulting artefacts. Although a wide range of tools exist for producing various multimedia outputs, comparatively limited attention has been paid to evaluating the user experience of these tools—particularly in relation to the effort invested in evaluating the outputs they produce. In the field of interactive digital narrative (IDN) authoring tools, for example, a 2023 literature review (Hargood and Green, 2022) found that only 7 out of the 37 tools surveyed included a dedicated UX evaluation. In contrast, 26 tools were introduced solely through usage demonstrations and examples, without further analysis of their usability from the perspective of intended end users.

The same study proposed several hypotheses to explain this disparity, emphasizing the following core challenges:

- The difficulty of breaking down a complex creative process into discrete tasks suitable for traditional usability testing.
- The extended timeframes typically required for authoring tasks, which

are not well-represented by short-term, lab-based evaluation sessions.

- The limited insight that purely quantitative metrics offer when attempting to capture the nuanced experience of creative practitioners.

Completing narrowly defined mechanical tasks with clear goals is not directly comparable to engaging in a long-term, open-ended creative process. While conventional usability testing remains valuable, these limitations highlight the need for complementary, more holistic studies that examine authoring tools in both structured and exploratory contexts.

For these reasons, the following sections present a two-part evaluation of the user experience in EnigMachine Editor. The first, introduced in Section 4.3.2.1, involves a short-term usability study conducted with a modest number of participants performing predefined design tasks. The second, discussed in Section 4.3.2.2, expands the scope through a month-long exploration of a more open-ended creative project, involving a larger participant group and longer engagement with the tool.

4.3.2.1 Usability in design implementation tasks

Hypothesis 4.3.2 raised the question of whether a user without a technical background or prior familiarity with the tool's concepts is generally able to use EnigMachine Editor for what we refer to as "design implementation tasks." By this, we denote tasks in which a user "translates" a set of design ideas into a working prototype, without these being necessarily devised by the user, assuming that everything included in these ideas is feasible to implement with the tools provided. Conceptually, this corresponds to the "short-term" system usage discussed at the beginning of this section. To assess this property, a study was conducted to analyse the ability of non-programmer users to convert a design document into a complete adventure using EnigMachine Editor.

This evaluation aims, on the one hand, to observe user feedback to understand the possible frictions and mechanical complexities perceived when attempting to translate a design concept into a functional element on the platform and, on the other hand, to inspect the games created by these users to determine the degree of success achieved in the implementation process. For this purpose, the System Usability Scale (SUS) questionnaire (Brooke, 1996), administered to the participants after the end of the testing sessions, was used in tandem with Single Ease Question (SEQ) questionnaires (Sauro and Dumas, 2009) after each task completed by the users in a first tutorial stage at the beginning of the experiment. Both of these evaluation systems have been repeatedly demonstrated to be valid, sensitive and reliable for UX evaluation. Key properties such as ease of use, complexity, consistency, confidence in use, and the learning curve of the system were assessed.

The experiment was designed as a practical session in the context of the Master's Degree in Video Game Design at our educational centre, with the participation of 14 individuals, comprising 5 women and 9 men, with a mean age of 24.88 years (standard deviation = 2.31; minimum age: 22 years, maximum age: 34 years). The participants were university design students, most with little or no programming experience, and with non-technical profiles. All data collected during the session were anonymised prior to analysis.

The session began with a brief introduction aimed at contextualising the activity and the tool. This was followed by a self-guided tutorial integrated into the web application, consisting of a battery of tasks to introduce the most important sections of EnigMachine. This tutorial was comprised of 13 tasks presented through textual prompts, allowing the user the option to quit any of them if they felt unable to complete it successfully. Once a task was completed, and before proceeding to the next task, the system displayed a single-item SEQ questionnaire with the prompt, "Overall, how difficult or easy was it to perform this task?". Responses to this question were given on a 7-point Likert scale, where a value of 1 indicated great difficulty in completing the task and a value of 7 indicated great ease.

Once the tutorial phase was completed, participants were given a draft design document for the game used in the experiments from Section 4.3.1, outlining the structure of the game flow and the different puzzles to be incorporated, referencing the museum artefacts needed to overcome each challenge. In addition, they were provided with a reference adventure, which was blank save for having the necessary assets pre-included to build the game (i.e., images required for AR events or to define characters, for instance). After the material was provided, participants were given 2 hours to translate the design document into a working implementation using the authoring tool. Subsequently, and for each participant, the number of essential phases (i.e., ignoring cosmetic or split narrative phases) correctly translated into EnigMachine Editor from the design document in the time allotted was computed. We consider that a minimum implementation of the draft script includes at least 18 stages, including those nested in room-type phases.

Lastly, and briefly before concluding the session, participants were asked to fill in the SUS questionnaire designed to assess the overall usability perceived by users during the experiment. This questionnaire consists of 10 items to assess components such as ease of use, complexity, consistency, confidence in use, and the learning curve of the system.

Participants rated each item on a 5-point Likert scale, where a value of 1 indicated complete disagreement and a value of 5 indicated complete agreement. In addition to analysing the results for each of the questions independently, the SUS score of each user was computed following the formula proposed in the original SUS manuscript (Brooke, 1996). Additionally, and

on a voluntary basis, each participant could write a free text commentary elaborating in more detail how they had felt about the tool, both positively and negatively.

The analysis of the SUS responses revealed a high level of satisfaction with the usability of EnigMachine Editor for design implementation tasks. The average SUS score obtained was 80.83, placing the tool in the 90-95th percentile according to established usability benchmarks (Admin, 2018). This corresponds to a Grade A, which is widely considered an indicator of excellent usability. These results suggest that users with no prior technical background were able to effectively understand and operate the system when working within a predefined and well-scoped design space.

Complementary results from the SEQ responses during the tutorial phase reinforce these findings. Participants consistently rated individual tasks as easy to complete, with most values clustering around the upper end of the 7-point scale. The few instances of lower scores were generally associated with tasks involving modification of nested blocks within room-type stages, such as inserting substages within parent containers or reordering blocks. This is to be expected, given that dealing with a nested structure is often more complex than working with individual stages and suggests that, while the overall interaction model was accessible, certain advanced or composite editing actions may require further simplification or scaffolding to improve learnability.

In terms of implementation success, the average completion rate of essential stages from the draft design document was 85.8%, indicating that most users were able to correctly replicate the intended game flow within the allotted time. Participants rarely required assistance during the implementation phase, and no participant abandoned the task before the deadline. Informal observations during the session further confirmed that most users navigated the interface fluidly and were able to identify the necessary tools to construct the different game mechanics prescribed in the script.

The optional free-text comments provided additional qualitative insights. A number of participants praised the intuitive structure of the interface and the visual clarity of the editing environment. Several users noted that the self-paced tutorial was particularly helpful in acclimating them to the tool's logic, reducing the initial intimidation they might have felt from facing a new digital platform. On the other hand, a few users mentioned that while the overall experience was smooth, they would have appreciated more visual cues or affordances to distinguish between different types of blocks (e.g., narrative vs. functional) and their impact on the resulting gameplay.

4.3.2.2 UX in creative authoring

Our final Hypothesis (Hypothesis 4.3.3) raised the question of whether EnigMachine Editor is a generally usable tool for non-technical audiences to

design interactive experiences directly and from scratch from the web platform itself over a longer period of time. The main difference between this use case and the one from the previous experiment is that we are now considering situations where the user is not a priori handed a set of design ideas for which a translation is guaranteed to exist in the tool but a much more ambiguous and creative task for which they are responsible for crafting a comprehensive design solely employing the features provided by EnigMachine.

Our expectation prior to the start of the experiments was that this type of evaluation would yield overall weaker usability results than those observed in the design implementation tasks, as a consequence of the potential inability to translate certain ideas from the creative process into the palette of features provided by the system, and the issues with traditional usability evaluation assumptions outlined at the beginning of this section. Here we considered hypothetical situations such as users wishing to include conditional flows, reskinning assets from the final game, or making more complex puzzles than those available in EnigMachine.

This experiment was set up as a practical project in the video game design subject of the video game development undergraduate degree at our educational centre, with the participation of 50 individuals. The participants were first-year undergraduate students, most of whom had little or no programming experience at the time the experiment was carried out.

In this context, the participants, in groups of 3 to 4 students, and over a period of time of roughly one month, were tasked with using the web editor to build their own adventures based on specific areas of the García Santesmases Computer Science Museum, which was the same as the one used for the experiments in Section 4.3.1. During this time, participants were granted access to the museum venue to take photographs and gather information about the different artefacts available in the exhibitions, with the aim of constructing rooms, narratives, and puzzles that would motivate visitors to inspect the museum's objects carefully during the game.

In the final stages of construction of these adventures, and on the occasion of a series of visits to the museum in the context of open days for secondary schools, a set of immersive tours was organised using the games created by those students interested in having live beta testers for their games. These were organised in 3 sessions of 37, 45, and 23 visitors, respectively, coming from a total of 6 educational centres in our city and mostly between 15 and 18 years old.

The usability questionnaires employed, as well as the tutorial and tool contextualization sessions in this experiment, were identical to those in Section 4.3.2.1. Thus, the initial phase of both groups was the same, with both contributing to the tutorial data with individual tasks linked to SEQ surveys, with the main differences being located in the second phase of the experiment. In this latter case, all participants were administered the SUS

questionnaire after the end of the period allotted to complete their adventures.

The average System Usability Scale (SUS) score obtained from the group was 63, which falls slightly below the industry benchmark and corresponds to a Grade C in the standard SUS grading scale. This result supports our initial hypothesis that usability performance would be lower in more open-ended, creative authoring contexts compared to the more constrained design implementation tasks described in Section 4.3.2.1.

The qualitative and quantitative data collected throughout the experiment indicate several contributing factors to this decrease in perceived usability. First, participants consistently reported lower confidence and higher perceived inconsistency when interacting with the system. Unlike in the previous study, where the task was more structured and the translation of design intent to implementation was more direct, the creative nature of this project often required students to reinterpret or compromise their original ideas to fit within the boundaries of EnigMachine Editor's available features. This mismatch between user expectations and system affordances was especially pronounced in cases involving conditional logic, custom scoring mechanisms, or more elaborate puzzle structures.

Secondly, a recurring theme in the feedback was a desire for more extensive customization options, especially in the visual and auditory layers of the game. Participants frequently expressed frustration at not being able to modify background visuals, adjust the interface color palette, or include audio elements such as ambient music or sound effects. This lack of flexibility was seen as a limitation in achieving the level of immersion they were aiming for in their designs.

Moreover, participants articulated the need for the tool to go beyond acting as a passive implementation platform. A significant number suggested that creative support mechanisms—such as templates, reusable patterns, or suggestions for interactive elements—could have helped guide the design process, particularly in early stages where ambiguity was high. This request aligns with patterns seen in other creative tools such as *PowerPoint* or *Kahoot!*, where scaffolding features play a central role in easing the design experience for non-technical users.

Lastly, several groups working collaboratively highlighted challenges related to co-authoring and workflow coordination. The absence of built-in support for multiple authors or version control made teamwork more cumbersome, leading to workflow inefficiencies and, in some cases, redundant effort.

Hypothesis 4.3.2, which stated that non-technical users would be able to use the tool to transform a design script into a playable adventure, was confirmed. This was not necessarily the case in the context of Hypothesis 4.3.3, which considered the situation where designers would have to use the tool to

create complex designs over an extended period of time and without a reference script. In the latter, general usability metrics, without being too poor, were found to show room for improvement when compared to commonly accepted industry standards.

The technical details of the work introduced in this section, including an in-depth analysis of the results summarized above, item-wise aggregates and answer distributions for the SUS, GEQ, and SEQ questionnaires, and statistical analyses comparing the results of different experiments conducted among distinct groups of participants, can be found in Paper 8 (published in *Entertainment Computing*), which addresses objective 2.3. Here we have the following further contributions:

1. We confirmed that our authoring tools enable non-technical users to create playable adventures from structured design scripts, with usability results indicating overall positive user experience and effectiveness in multiple authoring scenarios.

2. We evaluated the player experience of the generated adventures using the GEQ, finding strong results in dimensions such as positive affect, flow, and competence, with moderate challenge and low levels of stress and negative affect.

3. We assessed the user experience of the authoring process using SUS and SEQ. Short-term design tasks showed clearly positive usability results, while long-term, open-ended tasks without a reference script revealed opportunities for improving creative expressivity and better supporting complex design work, showcasing more modest usability scores.

4. We identified specific areas for improvement, particularly in sensory and imaginative immersion for game experience, recommending enhanced support for audio-visual design, and the need for better flow and aesthetic customization controls for the design experience.

These findings support the EnigMachine ecosystem as a low-cost, accessible, and engaging solution for immersive cultural heritage experiences, while outlining critical considerations for the future development of authoring platforms targeting non-technical users.

4.4 Towards AI-assisted adventure design

The preceding sections highlighted the various technical challenges involved in developing serious games within cultural contexts. While tools like EnigMachine Editor facilitate the creation of these experiences from a technical standpoint, the game design process itself remains complex due to several factors:

- Often, content curators at cultural heritage sites are not experts in game design. Their intention in creating such experiences is to motivate visitors to explore the space and learn about its objects, but they may lack experience in game design or interactive narrative creation. This can lead to experiences that are not as engaging or effective as they could be with a professional game designer, or curators may feel overwhelmed by the "blank canvas" effect and not know where to begin.

- In particular, a content curator is typically interested in selecting a set of museum artifacts to include in a visit and building some thematic thread to relate them. When creating an escape room experience based on a selection of objects, however, the content curator must not only define the objects to be included but also design a series of puzzles and challenges that guide players through the game's narrative. This requires a level of creativity and design that can be difficult to achieve without the aid of appropriate tools or a professional designer.

- Furthermore, the fact that these games are developed in a cultural context implies that the content must be factually accurate and historically or scientifically precise. This can be an additional challenge, as designers must ensure that the narrative and puzzles are consistent with available information about the objects and their context, which may require exhaustive research and a deep understanding of the subject. The large number of artifacts and texts that constitute a museum's knowledge base can make this process even more complicated, as the designer must be able to effectively select and synthesize relevant information and ensure that feedback given to the player is coherent and clear (e.g., avoiding ambiguities that could arise when giving a clue valid for multiple artworks).

- The museum's physical layout adds a final layer of complexity to the design process, as the designer must consider the physical arrangement of objects and how players will interact with them. This may require careful planning and an understanding of the space's dynamics, as well as the ability to design puzzles that effectively integrate into the museum's physical environment (e.g., a common type of clue refers to the relative position of an object to others).

To ease the design process, generative artificial intelligence models can provide assistance in brainstorming (Yu-Han and Chun-Ching, 2023) or generating initial versions of these games. This aligns with commercial interactive content creation platforms such as *Kahoot!* or *Quizizz AI*[2], which offer simple text-based mechanisms supported by generative AI as a source of inspiration and novel ideas.

[2]https://quizizz.com/quizizz-ai

In particular, the rise of Large Language Models (LLMs) has spurred interest in using these AIs for content generation across various domains, including video games. Trained on extensive text corpora, LLMs generate responses based on user prompts, usually accompanied by a set of rules or conversation history. Without prompt-embedded domain knowledge, their responses rely entirely on input and training data, known as a zero-shot paradigm, whereas adding examples enables few-shot reasoning (Yu et al., 2020). These models are not free of reasoning and action errors, however, which has given rise to different strategies to improve the generation quality by means of patterns such as chain-of-thought prompting (Wei et al., 2023) or ReAct (Reasoning and Acting) (Yao et al., 2023).

In video games, the use of LLMs has been explored for a variety of tasks including narrative creation, interactive stories, and scene generation (Buongiorno et al., 2024; Kumaran et al., 2024; Short, 2024). This includes the production of games in the form of visual novels with complex branching that propagate over time (Taveekitworachai et al., 2024), although some work reports that LLM-generated dialogues are not always perceived as positively as those written by humans (Akoury et al., 2023). They have also been applied in level design (e.g., Mario Bros (Sudhakaran et al., 2023) or Sokoban levels (Todd et al., 2023)) or for game environments and layouts (Gallotta et al., 2024; Hu et al., 2024).

For cultural heritage sites, LLMs have been used for guided tours and personalized narratives rather than traditional games. Helmy et al. (Helmy et al., 2024) combine a 3D scene generation tool (NeRF) with the LLaMa 3 language model to power a virtual tour guide for Egyptian heritage sites. In turn, Trichopoulos et al. (Trichopoulos, 2023) use OpenAI's GPT model to generate personalized narratives and recommendations for museum visitors. These approaches use conversational interfaces to encourage user exploration and engagement.

When developing content for cultural heritage sites, however, it is essential that the material produced remains factually grounded on the venue's knowledge base, often comprising numerous texts with information regarding the artifacts of its distinct exhibits. Given the tendency of LLMs to hallucinate, ensuring factual accuracy is paramount, particularly in learning-centered experiences.

With this background in mind, we went on to implement an adventure bootstraping system that uses an LLM to generate initial game ideas based on a museum and its object descriptions, as well as a set of parameters defining the desired game type. Specifically, we focused on creating a subset of Enigma adventures included to the following three core game flow phases:

- *Narrative phases*: virtual characters interact via dialogues to tell a story or elaborate on concepts and events that take place in the game.

- *Artefact search phases*: players are prompted to explore the area to locate a museum item based on a treasure hunt-style clue.
- *Multiple-choice question phases*: the system presents questions with a correct answer and distractors to assess the player's understanding.

Typically, these three kinds of phases are introduced in this same order, giving rise to thematic blocks focusing on each of the artifacts to be discovered by the player: (1) A narrative stage introduces or continues the story and motivates the search for an artifact; (2) a search phase prompts the player to reason about the given clue and explore the area to locate the artifact; and (3) a question phase encourages further scrutiny of the work in search of the correct answer.

Building on the broader goals outlined in Objective 2.4, this section investigates how artificial intelligence techniques can be integrated into design tools to support the early stages of creating interactive experiences. The aim is to enable the automated generation of initial content drafts from high-level design concepts—enhancing creativity while ensuring that the resulting output aligns with the designer's intent and remains grounded in factual accuracy.

Several key challenges arise in planning such an integration:

- How can we ensure that the generated content remains coherent, relevant to the museum's artifacts and exhibits, and consistent with the designer's narrative style and intent?
- How can we preserve narrative focus and cohesion across the game's multiple phases? Designers typically aim for logical continuity between clues, questions, and the underlying artifacts, which means that generating each element in isolation is insufficient. A unifying narrative flow must span the entire experience.
- How can we prevent factual inaccuracies in the generated content, particularly given the known limitations of LLMs, such as their tendency to hallucinate information not present in the source material?

To address these concerns, we propose a two-step method that combines fact distillation with a plan-and-execute strategy. The first step involves extracting the most relevant facts from the institutional descriptions provided by the museum—typically curated by the designer—so that content generated by the LLM is both consistent with the museum's knowledge base and resistant to factual drift. The second step focuses on constructing a high-level action plan for the adventure, outlining the sequence of phases and their intended content. This narrative scaffolding helps maintain coherence across stages and reinforces the model's understanding of what it is expected to produce at each point in the experience.

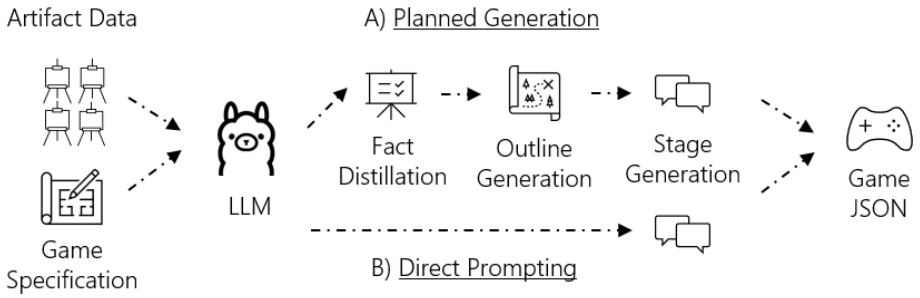


Figure 4.11: Treasure Hunt generation flow for both approaches.

To evaluate the effectiveness of this approach, we formulated the following hypotheses, each addressing a distinct component of the proposed pipeline:

Hypothesis 4.4.1. Distilling the most relevant facts from institutionally-provided descriptions of the artifacts included in an adventure improves the consistency and accuracy of the clues and questions generated by the LLM when compared to using the original descriptions directly.

Hypothesis 4.4.2. Developing an action plan prior to phase generation supports narrative focus and coherence across the game's stages and improves the quality of the generated content, compared to generating each phase in isolation.

In what follows, we will illustrate our approach using a case study based on the "Time of Fables" itinerary[3] from the Prado Museum in Madrid. This thematic tour explores myths and legends depicted in classical artworks and includes a curated selection of paintings, each accompanied by its own rich narrative and historical context.

4.4.1 Approach

Our approach starts from a set of treasure-hunt configuration details provided by the game designer or content curator and generates a JSON file with the definition of an Enigma game based on the information provided (see Figure 4.11).

4.4.1.1 Game parameters

To build an adventure based on a solid knowledge of the museum's objects, the first step is to list the artifacts the designer wants to include in the game. If the venue already displays panels with textual descriptions of the works, these can be directly used as input. However, depending on the

[3] `https://www.museodelprado.es/en/itinerary/tiempo-de-fabulas-los-mitos-y-los-dioses-radio-3/8c345c2a-2bd8-4cf7-9bc1-09fa38b73c0a`

Table 4.1: Sample input for a "Mythology Mystery" specification.

Parameter	**Input**
`General Description`	A mythology-based adventure where players uncover the legendary stories hidden in paintings and sculptures.
`Intention`	Provide a deep dive into mythology through artwork.
`Fact Focus`	Myths and legends behind classical artworks.
`Narrative Style`	A mysterious quest where players act as seekers of lost mythological knowledge.
`Clue Style`	Riddles based on myths, e.g., "Seek the god who..."
`Question Style`	Questions exploring the accuracy of myths in art.
`Target Audience`	Mythology and history enthusiasts.
`Difficulty`	Hard

use case, it may be beneficial to enrich this data with other remarks such as information about nearby objects, metadata regarding the artifact's location, or any relevant facts that could help generate clues in the treasure hunt. An example of an artifact description from our case study is shown below in Sample 1 for the artwork "Apollo in the Forge of Vulcan".

> **"Apollo in the Forge of Vulcan"**: "Diego Velázquez's painting *Apollo in the Forge of Vulcan* (c. 1630) depicts the Roman god Vulcan at work in his forge, surrounded by assistants hammering metal in the intense heat. Vulcan, engaged in forging, represents both creation and destruction. Velázquez contrasts the warm glow of fire with cooler tones..." (1)

Once this information is provided, the rest of the parameters refer to stylistic and focus considerations, both at the overall domain level and for the different phases of the adventure, as seen in the example from Table 4.1. Notably, parameters are provided to control the target audience (e.g., "Mythology and history enthusiasts"), the design intent, the expected difficulty, and a general description of the application's goal. Numerical inputs to control other aspects of the game, such as the length of narrative blocks or the number of answers per multiple-choice question, are also included.

4.4.1.2 Fact distillation and plan-and-execute strategy

Once the designer has entered this information, the game generation process begins. This starts by taking the texts for each of the artifacts and producing a reduced set of succinct facts with the focus indicated by the designer in the `Fact Focus` parameter. The number of facts output per artifact is controlled by the `Number of Facts` input parameter. We preliminarily argue that this step may contribute to focusing the attention of the model on the pieces of information that the designer considers most pertinent in the given context, especially given that the descriptions provided by cultural heritage institutions can be quite lengthy. An example of the facts suggested by the model for the artwork "Apollo in the Forge of Vulcan" using the sample inputs above is shown below.

1. Velázquez's painting 'Apollo in the Forge of Vulcan' humanizes the Roman god Vulcan, depicting him as a labourer rather than an idealized deity, making it an allegory of craftsmanship, creativity, and transformation.
2. Velázquez masterfully contrasts the warm glow of fire with cooler tones, creating a dramatic atmosphere that blends mythology with naturalism.
3. Velázquez enhances the realism and depth of the scene through his use of light and shadow, showcasing his technical mastery.

During all stages of generation, we make use of Microsoft's Guidance library[4] to control the output of the LLM in a structured way. This ensures that it is never necessary to perform parsing operations on the data generated by the model and that, in general, there are always mechanisms in place to guarantee the structural integrity of the proposed solutions. For instance, it is possible to restrict the output of the LLM during narrative generations to pick from only valid characters available in the adventure metadata or to impose the generation of an exact number of single-sentence answers in the question stages.

Given that the generated adventures may become excessively long, once the most relevant facts of each work have been distilled, we prompt the model to prepare a structured draft in which the points to be followed throughout the adventure are briefly defined. This strategy is aligned with established prompt engineering methodologies, such as *Plan-and-Execute*, where the model first outlines a solution strategy before sequentially addressing each subtask. This planning stage is not only a useful design approach but also addresses a known limitation of large language models: their tendency to lose track of crucial information in extended contexts. Recent studies have shown that LLMs often exhibit reduced performance when key details are situated in the middle of long prompts (Liu et al., 2023), and that memory retention generally degrades as input length increases (Liu et al., 2024). By requiring the model to articulate and store a coherent plan at the outset, we aim to mitigate these issues by ensuring that the overall structure

[4] `https://github.com/guidance-ai/guidance`

and progression of the adventure remain accessible to the model during the generation of each individual phase.

More specifically, this planning phase involves constructing a narrative-clue-question triplet for each artifact, following the template introduced at the end of Section 4.4. For every item included in the itinerary, the LLM is tasked with (1) selecting a distilled fact to serve as the foundation for a clue leading the player to the artifact, (2) defining a focal point to be emphasized in the multiple-choice question presented after discovery, and (3) summarizing the narrative events that bridge the interval between the current artifact and the next stage in the game. This preparatory step is intended to preserve narrative coherence and thematic continuity across the entire adventure.

Continuing with the example above, the general points generated as a sketch of the artifact block would be:

- **Narrative summary:** "as Kaida and Lyra delve into the world of ancient mythology, they discover that the mysterious collector's cryptic letter is linked to Velázquez's enigmatic painting "Apollo in the Forge of Vulcan", which holds the key to understanding the transformative power of art and the connection between creativity and craftsmanship, setting the stage for their quest to uncover the secrets hidden within the masterpieces."
- **Focus of the clue:** "the transformative power of art and the connection between creativity and craftsmanship."
- **Focus of the question:** "the accuracy of myths in art by asking about the depiction of Vulcan in the painting."

Once this outline is available, and with it the general narrative, the model makes a proposal of the characters to be used, together with a brief description of each of them. In the ongoing example, proposed characters are **Kaida Asteria**, "a brilliant and enigmatic art historian with a passion for uncovering the hidden secrets of ancient mythologies," and **Lyra Flynn**, "a charismatic and resourceful adventurer with a knack for deciphering cryptic clues and unravelling mysteries."

4.4.1.3 Stage generation

Having devised the plan to be followed, our approach then moves on to individually generate the stages of the game according to the general descriptions in the draft. Here, each of the designed stages is created from a sequence of prompts to the LLM in which the style settings, the focus to be followed, and the rules that must be generally respected in any stage of the corresponding type are injected. For instance, in question-type stages, one of the basic restrictions is that answers must not be repeated, while in clue-type stages, a clue must not explicitly reveal the name of the artifact to be searched for.

Structural and design constraints are a key aspect to take into account during the generation process. By the former, we refer to constraints on the shape of the produced content from a formatting point of view:

- A multiple-choice question must have exactly the requested number of possible answers, all of them consisting of a single sentence. The latter is also a restriction in the case of clues.
- A narrative stage can only reference characters previously defined in the game, must not exceed the maximum number of interactions per story block, and each of the interventions must be made up of a single sentence.

By design constraints, we denote rules that cannot generally be verified automatically as they are based on respecting the design intent of the person who specifies the adventure. Some examples for this paper's use case are:

- A narrative stage should never give overly obvious clues to the next artifact to be encountered in the game (e.g., the name of the artifact itself). Dialogues should not be repeated either directly or paraphrased, must respect the specified storytelling style, and must be cohesive and smoothly link the different stages of the game, especially in terms of providing the necessary context so that subsequent stages are sufficiently grounded.
- A clue stage should not explicitly reveal the name of the artifact to which it leads, but neither should it be so ambiguous or broad that the player has insufficient information to locate it. The level of difficulty, focus, and style should respect the design intentions.
- A multiple-choice question stage should not explicitly reveal the answer in the question, nor list the name of the artifact as one of the possible choices. Responses should not be repeated, either verbatim or in paraphrased but essentially equivalent versions. In addition, the correct answer should not be made more emphatic than the distractors (e.g., with an exclamation mark at the end). The level of difficulty, focus, and style should respect the design intentions. Most importantly, questions must always be correct and respect the factual data provided by the user's descriptions. The latter is particularly important in any adventure with a pedagogical focus.

With this, a final stage sequence for the running example would be:

- **Narrative:**

  Kaida: The symbol seems to be a variant of the ancient Greek character for the Muses.

Lyra: That's a great observation, Kaida.

Kaida: I think it's more than just a coincidence.

Lyra: Then we need to find out what Velázquez's paintings have to do with it.

Kaida: The paintings are said to hold the key to understanding the power of art.

- **Clue:** Seek the divine craftsman where fiery passion forges the very essence of art itself.
- **Question:** What is the primary representation of Vulcan in Velázquez's "Apollo in the Forge of Vulcan", according to the mythological context depicted in the artwork?
  - **Correct Answer:** Vulcan is portrayed as a labourer rather than an idealized deity, humanizing him and making the painting an allegory of craftsmanship, creativity, and transformation.
  - **Distractor 1:** The painting shows Vulcan as a powerful, godlike figure, emphasizing his divine nature and mythological significance.
  - **Distractor 2:** The artwork depicts Vulcan as a mere mortal, struggling to control the fiery forge, highlighting his vulnerability and humanity.
  - **Distractor 3:** The painting presents Vulcan as a symbol of destruction, emphasizing his role in the mythological context and the chaos he brings to the world.

While we present these examples here in a simplified form to improve readability, note that the system's output is delivered as a JSON file containing the definitions of each phase, metadata, and character listings. This structure is repeated as many times as artifacts included in the game, with a final additional narrative block between the point at which the player finds the last artifact and the end of the game. All the prompts and Guidance code used for the generation can be found in the project repository[5].

4.4.2 Evaluation

To assess the effectiveness of the proposed generation pipeline, we conduct a structured evaluation based on a series of design bootstraping experiments. The goal is to assess the validity of Hypotheses 4.4.1 and 4.4.2 considering whether incorporating explicit planning steps improves the quality and consistency of the generated adventure content when compared to a direct prompting approach that skips these steps in favor of generating each stage in isolation with the previously generated content as prompt context. We evaluate this by comparing both methods on 10 use cases available in the project repository, similar to the one in Table 4.1. These test cases simulate realistic scenarios for designing an interactive treasure hunt, varying in theme, focus, difficulty, and descriptions of the target audience and styles

[5]Repository Link: `https://github.com/pgutierrez858/enigma-llms`

to be followed. A fixed sequence of 4 popular artworks from the abovementioned "Time of Fables" itinerary from the Prado Museum was set as input for all use cases.

Our experimental evaluation is structured as follows:

1. In the first step, we generate complete adventure narratives using both the pre-planning and direct prompting generation pipelines. The pre-planned pipeline includes an initial fact extraction and strategy formulation phase before content generation as described above. Generated adventures consist of an intial narrative introducing the main characters of the story and its setup, a sequence of narrative-clue-quiz triplets with the main plot of the game, and a final narrative block closing the adventure. For each approach and generated adventure, we evaluate performance and efficiency metrics, such as the number of words included in each of the stage types, or the time taken to output each block by the LLM.

2. For each of the created stages, we then define three possible creation errors: *correctness failure* (i.e., the phase includes false information), *style failure* (i.e., the phase does not follow the style stated in the input, or presents some form of generation error), and finally *domain failure* (i.e., the phase does not conform to the general design constraints). Note that the failure cases do not necessarily have to be mutually exclusive. We manually annotate the generated adventures to identify these errors. We then conduct a detailed analysis of the nature and frequency of the recorded issues.

3. Lastly, we perform a qualitative evaluation of the generated adventures by manually revising them and assessing their overall quality, coherence, and narrative flow. This evaluation is based on a set of criteria that includes the relevance of the generated content to the specified theme, the logical progression of the narrative, and the overall engagement level of the adventure. We also consider how well each approach adheres to the design parameters provided by the user. This qualitative assessment is yet to be performed with a group of experts in game design and cultural heritage, being for now limited to our own early impressions, and will be reported in future work.

All experiments were conducted on a PC equipped with a Llama 3.1 8B model, an Intel Core i7-9750H processor at 2.60GHz, and 16GB of RAM, using Guidance for structured prompting. Full logs and annotated outputs are publicly available in the project repository. The primary findings at each stage of the experimental evaluation are summarized below.

- Our initial analysis of performance and efficiency metrics revealed clear distinctions between the planned and direct generation pipelines.

While the planned approach took approximately 10 minutes longer on average, this additional time is due to its fact extraction and preplanning stages. Since real-time generation is not required, this overhead is likely acceptable. In terms of content volume, both approaches were remarkably consistent: narrative, question, and clue word counts were closely aligned, indicating that the generation method does not significantly affect output length.

- A detailed review of creation errors across stage types highlighted the strengths and limitations of each strategy. Notably, both approaches performed flawlessly in clue-type stages, showing no errors across all defined categories. However, differences emerged in other content areas. The planned method showed a higher rate of domain errors in narrative stages, largely due to prematurely revealing the next artifact's name—an issue present in a majority of planned adventures. Other issues included repetitive content blocks and sentence truncation caused by early stopping rules. In question stages, planned generation occasionally produced distractors that were overly similar to the correct answers or offered factually invalid correct options.

  In contrast, the direct approach exhibited more stylistic issues, such as inconsistent adherence to input style and frequent looping, sometimes spanning multiple stages. Its domain errors were often more disruptive, including narrative inconsistencies (e.g., children addressing each other as "mum" in a family-themed tour) and references to irrelevant artworks. Direct generation questions also suffered from excessive verbosity, occasionally revealing too much information or exceeding token limits, leading to sentence truncation.

- Our qualitative evaluation, though preliminary and yet to be performed with domain experts, revealed a clear advantage for planned generation in narrative coherence. Planned adventures tended to follow more structured and focused storylines, and their domain errors—such as early artifact naming—were typically easy to fix. Direct generation, by contrast, was more prone to minor hallucinations, referencing works not included in the input. While both methods occasionally produced such errors, they were more frequent and intrusive in the direct approach.

  For clue stages, both methods produced generally valid and creative outputs, comparable to what a human designer might propose. However, because the model lacked awareness of the full artifact set, it could not fully account for potential distractors—highlighting the need for playtesting to validate clue clarity. As for questions, both pipelines yielded mostly accurate and usable content, adhering to the source material and design specifications. That said, some questions appeared to draw on external training data rather than the provided inputs, which

can introduce factual mismatches when artifacts share names.

Paper 9 (published in *International Conference on Entertainment Computing 2025*) presents the details of the work introduced in this section to address Objective 2.4, including a more in-depth analysis of the results summarized above, alongside distributions and aggregates for the different types of error counts and generation metrics. Here we have the following further contributions:

1. We introduce a novel approach for the automatic generation of treasure hunt games for cultural heritage sites using large language models, transforming user-specified artefact selections and textual design parameters into structured quest drafts suitable for further refinement.

2. We propose and evaluate a two-step generation strategy that combines fact distillation and pre-planning with phase-wise generation, demonstrating improvements in narrative focus, clue and question consistency, and overall content structure.

3. We confirm Hypothesis 4.4.1, finding that the distillation of key facts from institutional descriptions enhances the factual correctness and clarity of generated clues and questions when compared to the use of raw descriptions.

4. We partially confirm Hypothesis 4.4.2, observing that pre-planning supports narrative coherence and structural consistency across game stages, though it also introduces a distinct pattern of domain-specific errors, highlighting trade-offs between planning depth and generative flexibility.

5. We identify strengths and limitations across both planned and direct generation pipelines, including error typologies and stylistic variations, offering practical guidance for the integration of LLMs into semi-automated authoring workflows in cultural heritage contexts.

Chapter 5

Conclusions, Contributions and Future Work

But we have no idea what'll happen in the future. If we know nothing, we can do anything. Maybe that's what freedom means.

Tomihiko Morimi, The Tatami Time Machine Blues

In this chapter, we explore the motivations behind this thesis, detailing how they have driven our contributions and shaped the resulting approach. We highlight the evolution from our initial proofs-of-concept to perform automated quality assurance tasks in simple environments, to the development of Liquid Snake as a more realistic testing environment, and AI Behavior Graphs as a complete user tool to support the visual input of design specifications and automated training of regression testing bots for them. We then underscore the role of Enigma Bio as a first pilot from whose development iterations in cultural heritages sites we were able to learn a plethora of valuable lessons. These were then applied to the creation of EnigMachine as a generalized ecosystem for the creation of interactive AR-enabled experiences by non-technical profiles such as museum curators and educators. Finally, we conclude by outlining future research directions we are planning to take.

5.1 Conclusions and contributions

This thesis aims to advance the development of methodologies and tools that support game designers in the creation and maintenance of high-quality interactive experiences, without requiring advanced technical expertise. The central goal is to democratize the use of artificial intelligence, end-user sys-

tems, and formal methods in game design and development workflows, enabling broader access to quality assurance and interactive storytelling tools across different domains such as entertainment, education, and cultural heritage.

The work has been structured into two clearly differentiated lines of research, each corresponding to one of the global objectives outlined in Section 1.2:

- **Objective 1:** The development of accessible AI and formal methods-based techniques for automatic quality assurance, game design validation and testing, and player modeling in video games.
- **Objective 2:** The design of intuitive authoring tools that support the creation and deployment of interactive experiences across various domains, including educational and cultural applications.

We now present the conclusions and scientific contributions derived from the specific objectives that support each of these lines of work.

Contributions toward objective 1: AI for game design quality assurance

Objective 1.1. The first objective of this thesis was the development of machine learning methodologies for the creation of autonomous agents capable of performing automated regression testing in video games. Our goal was to equip game designers and QA professionals with tools that enhance traditional manual testing by introducing bots for automating various regression tests. These tools were to be designed to be affordable, easy to use, and require no prior knowledge of artificial intelligence or programming. We believe that this approach was necessary in order to to address the increasing complexity of modern video games and the need to maintain adequate testing coverage, especially in a context where design changes are frequent and can affect multiple aspects of gameplay.

The foundations of the techniques described in this first block were initially introduced in (Gutiérrez-Sánchez et al., 2021), a paper published in the *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (2021)* prior to the start of the thesis. In that work, we proposed a generic approach using reinforcement learning agents for automated regression testing across different levels of a simple video game. Our focus was on the early detection of significant gameplay variations caused by changes in enemy AI design. The system generated automatic alerts when bots detected statistically significant changes in their performance metrics—but only for agents trained with rewards tied to general goals, such as completing a level or collecting items. Despite the simplicity of the strategies

used, the results demonstrated that the agents could detect changes similar to those observed by human testers and provided valuable insights for debugging unexpected design issues.

Objective 1.2. Given the lack of suitable environments for assessing AI algorithms in the context of ongoing video game development, our second objective was to create a realistic, sufficiently complex, and varied testing environment to demonstrate the feasibility of the proposed techniques—particularly for regression testing and automatic error detection in commercial games.

Paper 1 and Paper 2, both presented at the *Spanish Video Game Conference (2022) and (2024)*, respectively, addressed this goal. In the first paper, we introduced a first version of Liquid Snake, a 3D third-person stealth game prototype designed to conveniently integrate autonomous agent-driven quality control mechanisms into the development life cycle of a video game, based on the open-source ML-Agents library in Unity3D. Focusing on the problem of regression testing for unintended changes caused by modifications to enemy AI, we argued that this environment serves well as a sample testbed for automated QA methodologies, due to the inherent complexity and behavioral variety of NPCs typically found in stealth games. The second paper described a more advanced version of the environment, including new game mechanics and level types, as well as support for new types of algorithms and AI techniques. In both cases, the goal was to maintain a project that enables realistic validation of the techniques and tools developed in this thesis while also serving as a resource for researchers and game developers exploring new automated testing methodologies. The evolution of Liquid Snake across the studies comprising this thesis supports its use as a research testbed for both academia and industry.

Objective 1.3. Our third objective was to generalize the problem of automated regression testing in video games to a formal framework that allows the specification of testing tasks using a designer-friendly input system. With this formalization, we aimed to provide tools that are closely aligned with the needs of human practitioners, enabling them to define design specifications in a precise, comprehensible, and visual manner. These specifications were used to guide the learning process of autonomous agents without the need to manually define complex task encodings. This adaptation also aimed to facilitate the interpretation and debugging of the generated agents, allowing designers and QA professionals to understand how design specifications relate to agent behavior.

This objective was addressed in a modular and progressive way in Paper 3, Paper 4, and Paper 5. Paper 3, presented at the *IEEE Conference on Games (2023)*, introduced an initial approach for specifying testing tasks using a temporal logic language and presented a reinforcement learning algorithm that leverages these specifications to automatically generate reward

functions, thereby guiding the agent's learning process.

Following this, Paper 4, presented at the *Spanish Video Game Conference (2023)*, described a visual node tool integrated in the Unity editor that allows designers to intuitively specify testing tasks in the specification language of Paper 3 and use these specifications to automatically launch the training process of the associated agent, without the need for prior knowledge in AI or programming.

Lastly, Paper 5, published in the *IEEE Transactions on Games (Special Issue on Human-Centered AI in Game Evaluation, 2024, Q2)*, presented an improved version of the reinforcement learning algorithm of Paper 4, which includes improvements in the quality of the generated reward functions and in the efficiency of the training process. This paper also included a comprehensive evaluation of the applicability of the proposed methodology to a set of testing tasks in the LIQUID SNAKE environment and a comparison with other automated testing methodologies based on imitation learning and reinforcement learning.

Objective 1.4. The fourth and final goal in this quality assurance block was to leverage the formal task specification framework developed in Objective 1.3 for the creation of player modeling tools. Building on the premise that it is possible to generate autonomous agents from design specifications, here we seeked to extend this idea to allow designers to extract models and explanations of player behaviors in the same formalisms that are used to define testing tasks. This approach aimed to help designers better understand how players interact with the game and how game mechanics can be adjusted to improve the user experience.

This objective was addressed in Paper 6, presented in *Foundations of Digital Games (2025)*. In this paper, we introduced a series of methodologies that, starting from a set of game traces, explore LTL specifications capable of explaining the behavior observed in the traces. We also presented a tree clustering algorithm that allows grouping similar traces and generating design specifications that describe the observed behavior in each group. We applied this methodology to a set of traces obtained from human players in LIQUID SNAKE and evaluated its ability to identify behavior patterns and produce understandable explanations for designers.

Contributions toward objective 2: Tools for interactive experience design

Objective 2.1. Shifting focus to the goals related to the development of interactive experience design tools, the first objective was to construct a generalistic, expressive, and purpose-driven platform for interactive experiences, incorporating feedback from designers and end-users across design iterations. This objective was primarily addressed in Paper 7, published in

the *Electronic Journal of e-Learning (2024, Q1).*

In this paper, we introduced Enigma Bio as an initial use case for an AR-enabled serious game aimed at teaching complex concepts, specifically biodiversity, to children aged 11 to 13 at the National Museum of Natural Sciences of Madrid. Our analysis focused on the adventure's capacity to communicate these concepts independently and the value of complementing the game with a preceding face-to-face session with an educator. To evaluate this, we conducted an A/B test with groups of students participating in the museum game, some with and some without an introductory session. Our findings suggested that while the adventure game could convey concepts to some extent, the introductory session significantly improved students' understanding, indicating that the game serves as a valuable complementary tool for reinforcing learning in educational contexts.

We detailed three main design iterations, spanning from late 2021 to early 2023, that guided the development of a game suitable for educational and cultural heritage applications. This iterative process, which integrated feedback from designers and players, yielded several crucial lessons that informed our subsequent objectives:

- We identified the need for in-game mechanisms that empower museum curators and educators to manage the game flow and direct visitors' attention to their explanations when necessary. This was implemented through the inclusion of a "waiting codes" mechanic.

- These games proved highly effective at reinforcing the understanding of concepts but less so at teaching them from scratch, as the playful component could sometimes overshadow the learning objective. However, combining the game with direct input from educators, either during or prior to the museum session, successfully supported concept acquisition.

- Embedding multiple-choice questions during game pauses proved an effective method for gathering feedback when children's attention was at its peak.

- Creating and editing adventures directly within the Unity Editor was found to be highly impractical for rapid iteration and prone to errors. This approach also prevented non-technical designers from independently testing their ideas, requiring programmer support. Furthermore, version control was cumbersome, and even programmers found it inconvenient to manipulate scriptable objects for new game construction.

- Linearly structured games presented difficulties when accommodating large groups, leading to issues such as groups blindly following others to

solve puzzles, or overcrowding at critical points. This could hinder AR scans, reduce player immersion, and disturb other museum visitors. To address this, escape-room-inspired blocks were incorporated, allowing players to naturally explore the museum, solve challenges in their own order, and distribute themselves more evenly.

Objective 2.2. The second objective of this block was the development of tools that enable game designers to create interactive experiences, such as those from Objective 2.1, without requiring advanced technical knowledge, thereby addressing the issues encountered by designers and developers during the aforementioned design iterations.

While not directly tied to one of the publications in this thesis, this objective was thoroughly addressed in Section 4.2. This section detailed our solution: an ecosystem of tools designed for the easy creation and deployment of interactive experiences in a data-driven, modular, and user-friendly manner. Specifically, we described the general phases supported by our system, which were distilled from the most common blocks developed for ENIGMA BIO and other Enigma Saga games. We further explained how these can be created within the web-based ENIGMACHINE EDITOR and subsequently automatically converted by a builder module into an Android Package (APK) that can be directly run on mobile devices. The underlying software architecture enabling this workflow was also detailed.

Objective 2.3. The tools developed in Objective 2.2 were intended to be both intuitive and accessible to designers, while simultaneously being powerful enough to facilitate the creation of interactive experiences that meet high design standards in terms of gameplay. Achieving this required a careful balance between ease of use and flexibility, empowering designers to explore diverse approaches and play styles without technological constraints.

This objective was addressed in Paper 8, a paper accepted to the Journal Track of the *International Conference on Entertainment Computing (2025, Q2)*, and published in the corresponding Special Issue of *Entertainment Computing*. In this work, we presented a set of comprehensive evaluations of the ENIGMACHINE ecosystem, assessing both its usability as a platform for designers and its capability to generate high-quality interactive experiences. This evaluation was conducted through various experimental sessions involving both short- and long-term users of the tool, as well as players of the adventures deployed at the García Santesmases Computer History Museum. The results demonstrated that the tool enables designers to generate quality interactive experiences without advanced technical knowledge, while also identifying specific areas for improvement in future iterations.

Objective 2.4. Our final objective was to integrate artificial intelligence methodologies into game design tools to assist designers in building initial drafts of interactive experiences from high-level specifications and preliminary design ideas. Our aim was to ensure these AI strategies were factually

accurate, controllable, and capable of consistently generating valid output.

This objective was addressed in Paper 9, a paper presented at the Conference Track of the *International Conference on Entertainment Computing (2025)*. In this paper, we proposed an approach utilizing large language models (LLMs) to automate the initial drafting of treasure hunts, adhering to the format used in EnigMachine Editor. We focused on generating cohesive sets of characters, narratives, clues, and multiple-answer questions, based on an initial description of the museum artifacts designated to host the adventure, the treasure hunt's objective, and various stylistic parameters. We compared two distinct generation methodologies, evaluating the quality of each in terms of correctness, consistency, and stylistic coherence. Our results showcased how LLMs can generate initial drafts of sufficient quality to serve as a starting point for designing interactive experiences, while also highlighting future challenges that need to be addressed for their realistic integration into game designers' workflows.

5.2 Future work

As future work, there are multiple lines of research we are considering, which mainly involve further extending the methodologies and frameworks introduced in this thesis. Let us enumerate the most relevant ones:

Initialize Behavior Graphs from Demonstrated Traces or Written Descriptions. Our approach from Section 3.5 provided preliminary examples showcasing how temporal logic predicates can be distilled from user behavior traces to model player goals, action flows, and constraints. Drawing inspiration from inverse reinforcement learning(Ng and Russell, 2000)—where the goal is to infer an agent's reward function from input demonstrations—we plan to explore a related method in which temporal logic predicates, rather than reward functions, are derived and used to bootstrap AI behavior graphs. As discussed in Sections 3.3.1 and 3.3.2, LTL predicates are often more interpretable than opaque reward functions, making it possible for designers to adjust and extend them within our visual editor. This way, instead of specifying rules, predicates, or reward functions directly, designers could provide demonstrations of desired behavior, enabling more intuitive and direct forms of agent training without sacrificing the formality and interpretability of logic-based training. That said, predicate complexity would need to be managed carefully; overly intricate logic formulae can quickly become unmanageable, much like decision trees, whose comprehensibility sharply decreases with increasing size, despite generally being considered interpretable (Freitas, 2014).

We further intend to deploy LLMs to provide an alternative specification initialization approach. Designers could describe desired behaviors in natural language, and an LLM—provided with some form of semantical an-

notation from the environment describing available entities, relationships between them, and possible predicates that could be exposed to the learning algorithm—could translate these descriptions into formal LTL predicates. Similar approaches have been explored in other domains with moderate success, where LLMs assist in generating reinforcement learning policies and skeletons (e.g., LLM-Guided RL policy modulation (Tan et al., 2025), LLM-MARL for strategic multi-agent gameplay (Li, 2025)). Embedding LLM translation into the derivation of predicates could help practitioners rapidly create interpretable behavior graphs that reflect intended behavior.

Address the Challenges of ML-based QA in the Industry. Recently, Sestini et al. (2024) gathered a set of insights from expert practitioners in the context of game validation and testing to identify the key challenges and opportunities for machine learning techniques in practical settings. Their findings highlight several concrete directions for future work, most of which intersect directly with our goals for AI BEHAVIOR GRAPHS's design and agent training systems and align with our perceptions on designer-centric QA:

- **Generalization.** Designers express the need for agents that go beyond reproducing optimal or previously demonstrated behavior (if using IL strategies), instead emulating the diverse and unpredictable nature of real players. ML models trained on expert traces or fixed environments often fail to generalize to novel contexts or unexpected player strategies, with a general tendency to overfitting their policies to the samples seen during training. As exemplified in our experiments from Section 3.3.5, this ability to generalize to changes is important to minimize false positives: situations where an agent is unable to adapt to non-breaking modifications in the game configuration and wrongly issues a design specification alert. A possible way to address this problem is through Procedural Level Generation (Justesen et al., 2018), enabling agents to be trained in a larger set of levels instead of a fixed one. While effective, this raises several practical issues: on the one hand, implementing procedural generators is far from trivial for complicated games, and on the other, generating valid levels that require intricate sequences of actions to be solved and are sound design-wise and close to real tasks is also quite challenging and not generally applicable.

  Approaches such as offline reinforcement learning and data augmentation have also shown promise in this space (Sinha et al., 2021). Offline reinforcement learning, similarly to Behavioral Cloning (Fujimoto and Gu, 2021), learns a policy using a pre-defined dataset without directly interacting with the environment. Assuming the provided dataset is sufficiently exhaustive, this methodology can lead to trained agents being more general and undergo deeper exploration. While our prelim-

inary experiments showed how our models were more generally more adaptable than baseline imitation learning bots, we are still striving to further validate them in more thorough use cases. We are also researching possible tactics, such as the ones just mentioned, to provide designers with more robust testing and lower false positive rates.

- **Personas.** A recurring request among surveyed professionals was support for training multiple behavioral personas—agents that reflect different playstyles such as cautious, exploratory, or collectionist. While we argue that our work can partly support this requirement via the definition of alternative specifications encoding a variety of approaches to deal with the task at hand, it is true that a single predicate can be fulfilled in multiple ways and conditioned by personality. Prior research has explored persona modeling in RL via reward shaping (Holmgard et al., 2019; Barthet et al., 2022), which we also did to an extent when introducing our preliminary testing workflow in Gutiérrez-Sánchez et al. (2021). Continuing this line of work, in the future we intend to perform further experiments using personality-oriented reward modules like the ones from the abovementioned works to ensure that designers are able to explore how different types of archetypal players can fulfill the reference specifications.

- **Exploration.** Another point for improvement in our methods is the tension between our algorithm's goal of maximizing specification reward and the need for exploratory behavior. Exploration is especially valuable during QA and level design, where agents might identify edge cases or gameplay flaws overlooked by designers. While combining IL with RL (as explored by Sestini et al. (2022b)) offers a path that balances exploitation of a known solution via the IL module and exploration via a RL module favoring novelty, current approaches remain sample-inefficient and are not trivial to integrate within our workflow. Future AI Behavior Graphs iterations might explore targeted exploration modules (Burda et al., 2018) or curiosity-based mechanisms (Pathak et al., 2017) during agent replay to foster novel trajectories without derailing overall goal completion.

- **Usability.** Perhaps most critically, the usability of our testing tools remains as a core point for future work. Designers need interpretable agent behaviors, clear model feedback, and low-effort training workflows. While our approach is able to enrich testing results with information related to the state of the specification's FSPA where the test failed, or computed progress reports, other techniques from explainable RL and game analytics—such as trajectory visualization, state-action heatmaps, and policy summarization—can be repurposed here to provide additional accessible insights into an agent's learning and

its performance in tests.

Furthermore, one-shot (Duan et al., 2017) and few-shot imitation learning approaches (Hakhamaneshi et al., 2022) offer a path to reducing the burden on designers by enabling learning from a small number of high-quality demonstrations. However, their applicability to incomplete or evolving game builds remains an open challenge. Integrating IL with the existing trace-recording infrastructure in AI BEHAVIOR GRAPHS could further reduce overhead by reusing natural gameplay data for agent training, but how to combine these techniques with our framework's training approach is not entirely clear.

Lastly, the AI BEHAVIOR GRAPHS tool itself has yet to undergo a thorough usability evaluation like the one performed for ENIGMACHINE, which is something that we intend to work on in the near future.

Extend ENIGMACHINE from User Insights and Related Research. ENIGMACHINE is an evolving ecosystem, iteratively shaped by user feedback to enhance its features and workflows. Our usability analysis highlighted several areas for improvement, which inform the direction of future development.

A key area of frustration for users was the limited ability to customize the aesthetic presentation of their generated games. This aligns with recent findings on the relationship between immersion and visual quality. For example, a study on mobile games with crisp, colorful 2.5D graphics found strong correlations between visual appeal (i.e. high-quality visual design) and all of focused attention, perceived reward experience, and perceived usability, suggesting that aesthetics can meaningfully enhance engagement (Kokil and Harwood, 2022). Similarly, Alexiou et al. (2020) showed that visual and narrative design significantly influence psychological flow, which in turn increases perceived learning in serious games. While care must be taken to avoid overwhelming users with excessive design options, we aim to introduce mechanisms that allow creators to adjust visuals and sound in a streamlined, accessible manner.

Another commonly reported limitation was the lack of support for branching narratives. Current escape room blocks only enable players to move through sections in non-linear but non-conditional ways, lacking the expressive power to support causal relationships between choices and outcomes. Both practitioners[1] and research (Moser and Fang, 2012, 2014) indicate that when players predict and perceive that their choice matters, their emotional involvement and narrative immersion increase. In response, we plan to explore and adapt branching paradigms from existing end-user platforms.

[1] `https://www.gamedeveloper.com/design/meaningful-decisions-in-branching-narratives`

Tools like *Twine*[2] offer expressive branching through variables and conditional logic, while *Genially*[3] provides basic "go to if..." structures suitable for educational narratives. These systems provide models for enabling branching while maintaining accessibility for non-programmers, and will be used as valuable reference to guide the design of EnigMachine's future editor capabilities.

By taking the above directions to enhance the system, we also hope to resolve the narrative and aesthetic immersion problems that our game experience evaluations highlighted as the weakest points in the tested games.

Include Stronger Mechanisms to Ensure Fact Accuracy. To move beyond our current reliance on reasoning chains and post-hoc fact distillation, future work should focus on establishing more formal and structured grounding methods for LLM-generated cultural adventures. A promising direction is the integration of SPARQL-based ontologies, which can anchor every narrative element—such as hints, characters, and questions—to verified knowledge sources. Barros et al. (2016)'s *Data Adventures* demo demonstrates an early version of this approach by generating mystery missions through SPARQL queries over DBpedia, ensuring that locations, events, and NPCs are derived from real-world data. Similarly, Jiayang et al. (2024) describe the *EventGround* framework, which grounds free-text narratives in knowledge graphs centered on temporal and causal relations, providing structure, consistency, and interpretability in generated storylines.

To translate user-authored mission descriptions into structured queries at scale, we envision adopting hybrid prompting pipelines. For instance, Jiang et al. (2025) introduce the *OntoSCPrompt* system, which produces initial SPARQL skeletons and incrementally fills in schema-specific details, demonstrating strong transferability across domains. Complementary techniques such as COT-SPARQL use chain-of-thought prompting to help LLMs produce more accurate and interpretable queries (Zahera et al., 2024). These methods can be combined with retrieval-augmented generation (RAG, (Lewis et al., 2021)) pipelines to form robust, hybrid systems—where RAG retrieves both textual and structured data, SPARQL ensures factual grounding, and the LLM generates coherent, context-aware narratives. Together, these strategies can significantly enhance the factual integrity of generated adventures without sacrificing creativity or flexibility.

Consider AI Co-Design Strategies. Alongside the methodologies in our work and in the literature, there is a developing interest in the notion of *co-design* or *mixed-initiative* approaches, or turning the human designer into an active actor in the generation algorithm, requesting their insight and participation at different points throughout the applied method (Yannakakis et al., 2014). As an example, Ross et al. (2023) implemented this

[2] `https://twinery.org/`
[3] `https://genially.com`

principle in *Programmer's Assistant*, a multi-turn conversational interface where users guide the LLM across iterative refinement stages. Following a similar approach, Gallotta et al. (2024) present a chat-based interface that enables designers to collaboratively create levels for a Darkest Dungeon-inspired game with a backing LLM module. The system processes natural language commands, then performs operations on a level model to fulfill user requests. This interaction model aims to offer a deeper, less intrusive, and more supportive user experience compared to traditional one-shot generation.

Bringing this paradigm into EnigMachine would allow users to engage with the system at any point in the creative workflow, from outlining to debugging or expanding adventures. However, realizing this vision is non-trivial. Studies have highlighted usability barriers, particularly for non-programmers. Feldman and Anderson (2024) found that users often faced challenges in intent specification and communication when interacting with code-generating LLMs, pointing to a need for clearer feedback loops and interaction patterns. Similarly, Vaithilingam et al. (2022) reported that while Copilot was effective for idea generation, users struggled with understanding, editing, and debugging code, emphasizing the importance of supporting prompt construction, comprehension, and iterative refinement in these efforts to bridge the gap between AI-systems and human practitioners.

Building on Norman's HCI framework, Bhat et al. (2023) highlight key challenges such as coherence, trust, predictability, and ownership in LLM-powered writing assistants. Similarly, Macias et al. (2024) underscore the importance of a user-centered design approach for educational authoring tools in their Moodle plugin, underscoring the need for intuitive UI and robust content control when integrating LLMs for non-technical users. Furthermore, the work by Zi et al. (2025) reveals significant usability hurdles for beginner programmers using LLMs for code generation, citing low comprehension rates for generated code (a mere 32.5% success rate), struggles for non-native English speakers, unfamiliarity with syntax, and automation bias. Collectively, these studies suggest that future efforts must focus on making LLM-powered tools more comprehensible, controllable, and reliable for a diverse range of users, ensuring that the technology genuinely enhances rather than hinders the authoring process.

These findings align with our own concerns about the integration of LLMs in the EnigMachine ecosystem. While, as mentioned earlier, we have a clear plan to leverage these technologies to expedite the authoring process, it remains crucial to ensure users perceive the newly added AI components as non-intrusive, fact-based, transparent, and controllable. We seek to integrate these components into a co-design loop that grants users a high degree of agency and ownership over the adventures and experiences they create. In order to accomplish this, each step of development and research will require

thorough evaluation to confirm these objectives are met.

Bibliography

"I'm wondering what to read next."
Matilda said. "I've finished all the
children's books."

Roald Dahl, Matilda

Admin. Item Benchmarks for the System Usability Scale - JUX. 2018. Publication Title: JUX - The Journal of User Experience.

Agarwal, S., Herrmann, C., Wallner, G. and Beck, F. Visualizing AI Playtesting Data of 2D Side-scrolling Games. In *2020 IEEE Conference on Games (CoG)*, pages 572–575. IEEE, Osaka, Japan, 2020. ISBN 978-1-7281-4533-4.

Akoury, N., Yang, Q. and Iyyer, M. A Framework for Exploring Player Perceptions of LLM-Generated Dialogue in Commercial Video Games. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2295–2311. Association for Computational Linguistics, Singapore, 2023.

Albadawi, B. I. The Virtual Museum VM as a Tool for Learning Science in Informal Environment. *Education in the Knowledge Society (EKS)*, volume 22, pages e23984–e23984, 2021. ISSN 2444-8729.

Aleem, S., Capretz, L. F. and Ahmed, F. Critical Success Factors to Improve the Game Development Process from a Developers Perspective. 2018.

Alexander, C. *A pattern language: towns, buildings, construction*. Oxford University Press, 1977.

Alexiou, A., Oshri, M., Schippers, M. and Angelopoulos, S. Narrative and Aesthetics as Antecedents of Perceived Learning in Serious Games. *Information Technology & People*, 2020. ISSN 0959-3845. Publisher: Emerald.

ALMEIDA, P., CARVALHO, V. and SIMÕES, A. Reinforcement Learning as an Approach to Train Multiplayer First-Person Shooter Game Agents. *Technologies*, volume 12(3), page 34, 2024. Publisher: MDPI.

ALSHAHWAN, N., GAO, X., HARMAN, M., JIA, Y., MAO, K., MOLS, A., TEI, T. and ZORIN, I. Deploying Search Based Software Engineering with Sapienz at Facebook. In *Search-Based Software Engineering* (edited by T. E. Colanzi and P. McMinn), pages 3–45. Springer International Publishing, Cham, 2018. ISBN 978-3-319-99241-9.

ALYAZ, Y., SPANIEL-WEISE, D. and GURSOY, E. A Study on Using Serious Games in Teaching German as a Foreign Language. *Journal of Education and Learning*, volume 6(3), page 250, 2017. ISSN 1927-5269, 1927-5250.

AMODEI, D., OLAH, C., STEINHARDT, J., CHRISTIANO, P. F., SCHULMAN, J. and MANÉ, D. Concrete Problems in AI Safety. *CoRR*, volume abs/1606.06565, 2016. ArXiv: 1606.06565.

ANDERSON, E. F., MCLOUGHLIN, L., LIAROKAPIS, F., PETERS, C., PETRIDIS, P. and DE FREITAS, S. Serious games in cultural heritage. 2009.

ANDREOLI, R., COROLLA, A., FAGGIANO, A., MALANDRINO, D., PIROZZI, D., RANALDI, M., SANTANGELO, G. and SCARANO, V. A Framework to Design, Develop, and Evaluate Immersive and Collaborative Serious Games in Cultural Heritage. *J. Comput. Cult. Herit.*, volume 11(1), 2017. ISSN 1556-4673. Place: New York, NY, USA Publisher: Association for Computing Machinery.

ARIYUREK, S., BETIN-CAN, A. and SURER, E. Automated Video Game Testing Using Synthetic and Humanlike Agents. *IEEE Transactions on Games*, volume 13(1), pages 50–67, 2021. ISSN 2475-1502, 2475-1510.

BAIN, M. and SAMMUT, C. A Framework for Behavioural Cloning. In *Machine Intelligence 15*, pages 103–129. 1995.

BARBIERI, G. G., BARBIERI, R. and CAPONE, R. Serious Games in High School Mathematics Lessons: An Embedded Case Study in Europe. *Eurasia Journal of Mathematics, Science and Technology Education*, volume 17(5), page em1963, 2021. ISSN 13058215, 13058223.

BARROS, G., LIAPIS, A. and TOGELIUS, J. Playing with Data: Procedural Generation of Adventures from Open Data. In *Proceedings of DiGRA/FDG 2016 Conference*. 2016.

BARTHET, M., KHALIFA, A., LIAPIS, A. and YANNAKAKIS, G. Generative Personas That Behave and Experience Like Humans. In *Proceedings of*

the 17th International Conference on the Foundations of Digital Games, FDG '22. Association for Computing Machinery, New York, NY, USA, 2022. ISBN 978-1-4503-9795-7. Event-place: Athens, Greece.

BEATTIE, C., LEIBO, J. Z., TEPLYASHIN, D., WARD, T., WAINWRIGHT, M., KÜTTLER, H., LEFRANCQ, A., GREEN, S., VALDÉS, V., SADIK, A., SCHRITTWIESER, J., ANDERSON, K., YORK, S., CANT, M., CAIN, A., BOLTON, A., GAFFNEY, S., KING, H., HASSABIS, D., LEGG, S. and PETERSEN, S. DeepMind Lab. 2016. _eprint: 1612.03801.

BECK, K. *Test-Driven Development by Example*. Addison Wesley, 2002.

BEKELE, M. K., PIERDICCA, R., FRONTONI, E., MALINVERNI, E. S. and GAIN, J. A Survey of Augmented, Virtual, and Mixed Reality for Cultural Heritage. *J. Comput. Cult. Herit.*, volume 11(2), pages 7:1–7:36, 2018. ISSN 1556-4673.

BERGDAHL, J., GORDILLO, C., TOLLMAR, K. and GISSLEN, L. Augmenting Automated Game Testing with Deep Reinforcement Learning. In *2020 IEEE Conference on Games (CoG)*, pages 600–603. IEEE, Osaka, Japan, 2020. ISBN 978-1-7281-4533-4.

BETHKE, E. *Game Development and Production*. Wordware game developer's library. Wordware Publishing Inc., 2003.

BHAT, A., SHRIVASTAVA, D. and GUO, J. L. C. Approach Intelligent Writing Assistants Usability with Seven Stages of Action. 2023. _eprint: 2304.02822.

BJÖRK, S. and HOLOPAINEN, J. *Patterns in Game Design*. Charles River Media, 2005.

BROCKMAN, G., CHEUNG, V., PETTERSSON, L., SCHNEIDER, J., SCHULMAN, J., TANG, J. and ZAREMBA, W. OpenAI gym. *arXiv prepr int arXiv:1606.01540*, 2016.

BROOKE, J. SUS – a quick and dirty usability scale. pages 189–194. 1996.

BROWNE, C. B., POWLEY, E., WHITEHOUSE, D., LUCAS, S. M., COWLING, P. I., ROHLFSHAGEN, P., TAVENER, S., PEREZ, D., SAMOTHRAKIS, S. and COLTON, S. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, volume 4(1), pages 1–43, 2012.

BUONGIORNO, S., KLINKERT, L., ZHUANG, Z., CHAWLA, T. and CLARK, C. PANGeA: Procedural Artificial Narrative Using Generative AI for Turn-Based, Role-Playing Video Games. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 20(1), pages 156–166, 2024. ISSN 2334-0924, 2326-909X.

BURDA, Y., EDWARDS, H., STORKEY, A. and KLIMOV, O. Exploration by Random Network Distillation. 2018. _eprint: 1810.12894.

BUZHINSKY, I. Formalization of natural language requirements into temporal logics: a survey. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 400–406. 2019.

BUZZSHOT. Telescape Live. 2025.

BÉCARES, J. H., VALERO, L. C. and GÓMEZ-MARTÍN, P. P. An Approach to Automated Videogame Beta Testing. *Entertainment Computing*, volume 18, pages 79–92, 2017.

CADAR, C., DUNBAR, D. and ENGLER, D. KLEE: unassisted and automatic generation of high-coverage tests for complex systems programs. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 209–224. USENIX Association, USA, 2008. Event-place: San Diego, California.

CAMPS-ORTUETA, I., GONZÁLEZ-CALERO, P. A., QUIROGA, M. A. and GÓMEZ-MARTÍN, P. P. Measuring Preferences in Game Mechanics: Towards Personalized Chocolate-Covered Broccoli. In *Entertainment Computing and Serious Games - First IFIP TC 14 Joint International Conference, ICEC-JCSG 2019, Arequipa, Peru, November 11-15, 2019, Proceedings* (edited by E. D. V. d. Spek, S. Göbel, E. Y.-L. Do, E. Clua and J. B. Hauge), volume 11863 of *Lecture Notes in Computer Science*, pages 15–27. Springer, 2019.

CARR, L., MCKECHNIE, T., HATAMNEJAD, A., CHAN, J. and BEATTIE, A. Effectiveness of the Eyesi Surgical Simulator for ophthalmology trainees: systematic review and meta-analysis. *Canadian Journal of Ophthalmology. Journal Canadien D'ophtalmologie*, volume 59(3), pages 172–180, 2024. ISSN 1715-3360.

CHAUHAN, V. K., ZHOU, J., LU, P., MOLAEI, S. and CLIFTON, D. A. A Brief Review of Hypernetworks in Deep Learning. 2023. _eprint: 2306.06955.

CHON, S.-H., TIMMERMANN, F., DRATSCH, T., SCHUELPER, N., PLUM, P., BERLTH, F., DATTA, R. R., SCHRAMM, C., HANEDER, S., SPÄTH, M. R., DÜBBERS, M., KLEINERT, J., RAUPACH, T., BRUNS, C. and KLEINERT, R. Serious Games in Surgical Medical Education: A Virtual Emergency Department as a Tool for Teaching Clinical Reasoning to Medical Students. *JMIR Serious Games*, volume 7(1), page e13028, 2019. ISSN 2291-9279.

Chorianopoulos, K., Giannakos, M. N. and Chrisochoides, N. Design Principles for Serious Games in Mathematics. In *Proceedings of the 18th Panhellenic Conference on Informatics*, pages 1–5. ACM, Athens Greece, 2014. ISBN 978-1-4503-2897-5.

Clarke, S., Peel, D., Arnab, S., Morini, L., Keegan, H. and Wood, O. EscapED: A Framework for Creating Educational Escape Rooms and Interactive Games to For Higher/Further Education. *International Journal of Serious Games*, volume 4(3), 2017.

DaCosta, B. and Kinsell, C. Serious Games in Cultural Heritage: A Review of Practices and Considerations in the Design of Location-Based Games. *Education Sciences*, volume 13(1), page 47, 2022. ISSN 2227-7102.

Desurvire, H. and Wiberg, C. Master of the game: assessing approachability in future game design. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, pages 3177–3182. ACM, Florence Italy, 2008. ISBN 978-1-60558-012-8.

Desurvire, H. and Wiberg, C. Game Usability Heuristics (PLAY) for Evaluating and Designing Better Games: The Next Iteration. In *Online Communities and Social Computing* (edited by A. A. Ozok and P. Zaphiris), pages 557–566. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-642-02774-1.

Djaouti, D., Alvarez, J. and Jessel, J.-P. Classifying serious games: the G/P/S model. In *Handbook of Research on Improving Learning and Motivation through Educational Games*, Advances in Game-Based Learning, pages 118–136. IGI Global, 2011.

Duan, Y., Andrychowicz, M., Stadie, B. C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P. and Zaremba, W. One-Shot Imitation Learning. 2017. _eprint: 1703.07326.

El-Nasr, M. S. and Kleinman, E. Data-Driven Game Development: Ethical Considerations. 2020. _eprint: 2006.10808.

Esparza, J., Křetínský, J., Raskin, J.-F. and Sickert, S. From linear temporal logic and limit-deterministic Büchi automata to deterministic parity automata. *International Journal on Software Tools for Technology Transfer*, volume 24(4), pages 635–659, 2022. ISSN 1433-2787.

Feldman, M. Q. and Anderson, C. J. Non-Expert Programmers in the Generative AI Future. In *Proceedings of the 3rd Annual Meeting of the Symposium on Human-Computer Interaction for Work*, CHIWORK '24. Association for Computing Machinery, New York, NY, USA, 2024. ISBN 979-8-4007-1017-9. Event-place: Newcastle upon Tyne, United Kingdom.

FIDAS, C., SINTORIS, C., YIANNOUTSOU, N. and AVOURIS, N. A survey on tools for end user authoring of mobile applications for cultural heritage. In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–5. 2015.

FOTARIS, P. and MASTORAS, T. Escape Rooms for Learning: A Systematic Review. In *European Conference on Games Based Learning*, pages 235–243. 2019.

FRAGA-VARELA, F., VILA-COUÑAGO, E. and RODRÍGUEZ-GROBA, A. Serious Games and Mathematical Fluency: A Study from the Gender Perspective in Primary Education. *Sustainability*, volume 13(12), page 6586, 2021. ISSN 2071-1050.

FRASER, G. and ARCURI, A. EvoSuite: automatic test suite generation for object-oriented software. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, pages 416–419. Association for Computing Machinery, New York, NY, USA, 2011. ISBN 978-1-4503-0443-6. Event-place: Szeged, Hungary.

FREITAS, A. A. Comprehensible classification models: a position paper. *SIGKDD Explor. Newsl.*, volume 15(1), pages 1–10, 2014. ISSN 1931-0145. Place: New York, NY, USA Publisher: Association for Computing Machinery.

FUJIMOTO, S. and GU, S. S. A Minimalist Approach to Offline Reinforcement Learning. 2021. _eprint: 2106.06860.

GALLOTTA, R., LIAPIS, A. and YANNAKAKIS, G. Consistent Game Content Creation via Function Calling for Large Language Models. In *2024 IEEE Conference on Games (CoG)*, pages 1–4. IEEE, Milan, Italy, 2024. ISBN 979-8-3503-5067-8.

GAMMA, E., HELM, R., JOHNSON, R. and VLISSIDES, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

GANDINI, S., RUZZARIN, W., SANCHEZ, E., SQUILLERO, G. and TONDA, A. A Framework for Automated Detection of Power-related Software Errors in Industrial Verification Processes. *Journal of Electronic Testing*, volume 26(6), pages 689–697, 2010. ISSN 1573-0727.

GAROUSI, V. and MÄNTYLÄ, M. V. When and what to automate in software testing? A multi-vocal literature review. *Information and Software Technology*, volume 76, pages 92–117, 2016. ISSN 0950-5849.

Garzón-Artacho, E., Sola-Martínez, T., Romero-Rodríguez, J.-M. and Gómez-García, G. Teachers' perceptions of digital competence at the lifelong learning stage. *Heliyon*, volume 7(7), page e07513, 2021. ISSN 2405-8440.

Gelly, S. and Wang, Y. Exploration exploitation in go: UCT for Monte-Carlo go. In *NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop*. 2006.

Genially Web S.L. Genially. 2025.

Gerling, K., Dickinson, P., Hicks, K., Mason, L., Simeone, A. L. and Spiel, K. Virtual Reality Games for People Using Wheelchairs. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–11. ACM, Honolulu HI USA, 2020. ISBN 978-1-4503-6708-0.

Ghiani, G., Paternò, F. and Spano, L. D. Cicero Designer: An Environment for End-User Development of Multi-Device Museum Guides. In *End-User Development* (edited by V. Pipek, M. B. Rosson, B. de Ruyter and V. Wulf), pages 265–274. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-642-00427-8.

Gillberg, J., Bergdahl, J., Sestini, A., Eakins, A. and Gisslen, L. Technical Challenges of Deploying Reinforcement Learning Agents for Game Testing in AAA Games. 2023. _eprint: 2307.11105.

Gonzalez-Calero, P., Camps-Ortueta, I., Gutiérrez-Sánchez, P. and Gómez-Martín, P. On the Importance of Contextualizing an Educational Escape Room Activity. *Electronic Journal of e-Learning*, volume 22, pages 43–56, 2024.

Gorbanev, I., Agudelo-Londoño, S., González, R. A., Cortes, A., Pomares, A., Delgadillo, V., Yepes, F. J. and Muñoz, O. A systematic review of serious games in medical education: quality of evidence and pedagogical strategy. *Medical Education Online*, volume 23(1), page 1438718, 2018. ISSN 1087-2981.

Gordillo, C., Bergdahl, J., Tollmar, K. and Gisslen, L. Improving Playtesting Coverage via Curiosity Driven Reinforcement Learning Agents. In *2021 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, Copenhagen, Denmark, 2021. ISBN 978-1-6654-3886-5.

Green, D., Hargood, C. and Charles, F. Use of Tools: UX Principles for Interactive Narrative Authoring Tools. *J. Comput. Cult. Herit.*, volume 14(3), 2021. ISSN 1556-4673. Place: New York, NY, USA Publisher: Association for Computing Machinery.

GRUPO DE INTERNET DE NUEVA GENERACIÓN. Escapp. 2025.

GUSS, W. H., CODEL, C., HOFMANN, K., HOUGHTON, B., KUNO, N., MILANI, S., MOHANTY, S., LIEBANA, D. P., SALAKHUTDINOV, R., TOPIN, N., VELOSO, M. and WANG, P. The MineRL 2019 Competition on Sample Efficient Reinforcement Learning using Human Priors. 2021. _eprint: 1904.10079.

GUTIÉRREZ-SÁNCHEZ, P., GONZÁLEZ-CALERO, P. A., GÓMEZ-MARTÍN, M. A., GÓMEZ-MARTÍN, P. P. and THAWONMAS, R. Initializing interactive treasure hunts in cultural heritage sites: An llm-based approach. In *Entertainment Computing – ICEC 2025* (edited by M. Sugimoto, A. Di Iorio, P. Figueroa, R. Yamanishi and K. Matsumura), pages 151–165. Springer Nature Switzerland, Cham, 2025. ISBN 978-3-032-02555-5.

GUTIÉRREZ-SÁNCHEZ, P., GÓMEZ-MARTÍN, M., GONZALEZ-CALERO, P. and GÓMEZ-MARTÍN, P. A Progress-Based Algorithm for Interpretable Reinforcement Learning in Regression Testing. *IEEE Transactions on Games*, volume PP, pages 1–10, 2024.

GUTIÉRREZ-SÁNCHEZ, P., GÓMEZ-MARTÍN, M. A., GONZÁLEZ-CALERO, P. A. and GÓMEZ-MARTÍN, P. P. Reinforcement learning methods to evaluate the impact of AI changes in game design. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 17, pages 10–17. 2021. Issue: 1.

GUTIÉRREZ-SÁNCHEZ, P., GÓMEZ-MARTÍN, M. A., GONZÁLEZ-CALERO, P. A. and GÓMEZ-MARTÍN, P. P. Liquid Snake: a test environment for video game testing agents. In *Actas del I Congreso Español de Videojuegos, Madrid, Spain, December 1-2, 2022* (edited by R. Lara-Cabrera and A. J. F. Leiva), volume 3305 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022.

GUTIÉRREZ-SÁNCHEZ, P., GÓMEZ-MARTÍN, M. A., GONZÁLEZ-CALERO, P. A. and GÓMEZ-MARTÍN, P. P. AI Behavior Graphs: A Visual Toolkit for Defining NPC Specifications for Regression Testing. In *Actas del II Congreso Español de Videojuegos, Madrid, Spain, November 9-10, 2023*, CEUR Workshop Proceedings (In Press). 2023a.

GUTIÉRREZ-SÁNCHEZ, P., GÓMEZ-MARTÍN, M. A., GONZÁLEZ-CALERO, P. A. and GÓMEZ-MARTÍN, P. P. Reinforcement Learning with Temporal Logic Specifications for Regression Testing NPCs in Video Games. In *2023 IEEE Conference on Games (CoG)*, pages 1–8. 2023b.

GUTIÉRREZ-SÁNCHEZ, P., GÓMEZ-MARTÍN, P. P., GONZÁLEZ-CALERO, P. A. and GÓMEZ-MARTÍN, M. A. Evaluating usability for an ar-enabled

escape room authoring ecosystem. *Entertainment Computing*, volume 55, page 100982, 2025a. ISSN 1875-9521.

GUTIÉRREZ-SÁNCHEZ, P., PÉREZ-LIÉBANA, D. and GAINA, R. D. Explaining and Clustering Playtraces Using Temporal Logics. In *Proceedings of the 20th International Conference on the Foundations of Digital Games*, FDG '25. Association for Computing Machinery, New York, NY, USA, 2025b. ISBN 979-8-4007-1856-4.

HAARNOJA, T., ZHOU, A., ABBEEL, P. and LEVINE, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. 2018. _eprint: 1801.01290.

HADKA, D. Moea framework: A free and open source java framework for multiobjective optimization. 2025. Version 5.1 [Computer software].

HAKHAMANESHI, K., ZHAO, R., ZHAN, A., ABBEEL, P. and LASKIN, M. Hierarchical Few-Shot Imitation with Skill Transition Models. In *International Conference on Learning Representations*. 2022.

HALL, L. E. *Planning your escape: strategy secrets to make you an escape room superstar*. Tiller Press, New York, first tiller press trade paperback edition edition, 2021. ISBN 978-1-9821-4034-2.

HANSEN, F. A., KORTBEK, K. J. and GRØNBÆK, K. Mobile Urban Drama: interactive storytelling in real world environments. *New Review of Hypermedia and Multimedia*, volume 18(1-2), pages 63–89, 2012. ISSN 1361-4568, 1740-7842.

HARGOOD, C. and GREEN, D. The Authoring Tool Evaluation Problem. In *The Authoring Problem: Challenges in Supporting Authoring for Interactive Digital Narratives* (edited by C. Hargood, D. E. Millard, A. Mitchell and U. Spierling), pages 303–320. Springer International Publishing, Cham, 2022. ISBN 978-3-031-05214-9.

HELMY, M., EL-DIN, Y. S., MOHAMED, O. T., KADER, O. S. A., RAMADAN, S. A., KAMAL, A. E. and SELIM, M. R. M. Navigating the World with an Intelligent Tourist Guide Using Generative AI. In *2024 International Telecommunications Conference (ITC-Egypt)*, pages 1–6. IEEE, Cairo, Egypt, 2024. ISBN 979-8-3503-5140-8.

HO, J. and ERMON, S. Generative Adversarial Imitation Learning. *arXiv:1606.03476 [cs]*, 2016.

HOLMGARD, C., GREEN, M. C., LIAPIS, A. and TOGELIUS, J. Automated Playtesting With Procedural Personas Through MCTS With Evolved Heuristics. *IEEE Transactions on Games*, volume 11(4), pages 352–362, 2019. ISSN 2475-1502, 2475-1510.

HOLTZ, B. E., MURRAY, K. and PARK, T. Serious Games for Children with Chronic Diseases: A Systematic Review. *Games for Health Journal*, volume 7(5), pages 291–301, 2018. ISSN 2161-7856.

HU, S., HUANG, Z., HU, C. and LIU, J. 3D Building Generation in Minecraft via Large Language Models. 2024. _eprint: 2406.08751.

HUMBLE, J. and FARLEY, D. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Signature Series (Fowler). Addison-Wesley, 2010.

IFTIKHAR, S., IQBAL, M. Z., KHAN, M. U. and MAHMOOD, W. An automated model based testing approach for platform games. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 426–435. 2015.

IJSSELSTEIJN, W. A., DE KORT, Y. and POELS, K. *The Game Experience Questionnaire*. Technische Universiteit Eindhoven, 2013.

JACOB, M., DEVLIN, S. and HOFMANN, K. "It's Unwieldy and It Takes a Lot of Time." Challenges and Opportunities for Creating Agents in Commercial Games. 2020. _eprint: 2009.00541.

JANSSEN, D., TUMMEL, C., RICHERT, A. and ISENHARDT, I. Towards Measuring User Experience, Activation and Task Performance in Immersive Virtual Learning Environments for Students. In *Immersive Learning Research Network* (edited by C. Allison, L. Morgado, J. Pirker, D. Beck, J. Richter and C. Gütl), pages 45–58. Springer International Publishing, Cham, 2016. ISBN 978-3-319-41769-1.

JIANG, L., HUANG, J., MÖLLER, C. and USBECK, R. Ontology-Guided, Hybrid Prompt Learning for Generalization in Knowledge Graph Question Answering. 2025. _eprint: 2502.03992.

JIAYANG, C., QIU, L., CHAN, C., LIU, X., SONG, Y. and ZHANG, Z. EventGround: Narrative Reasoning by Grounding to Eventuality-centric Knowledge Graphs. 2024. _eprint: 2404.00209.

JONES, C. E. K., THEODOSIS, S. and LYKOURENTZOU, I. The Enthusiast, the Interested, the Sceptic, and the Cynic: Understanding User Experience and Perceived Value in Location-Based Cultural Heritage Games Through Qualitative and Sentiment Analysis. *J. Comput. Cult. Herit.*, volume 12(1), pages 6:1–6:26, 2019. ISSN 1556-4673.

JULIANI, A., BERGES, V.-P., TENG, E., COHEN, A., HARPER, J., ELION, C., GOY, C., GAO, Y., HENRY, H., MATTAR, M. and LANGE, D. Unity: A General Platform for Intelligent Agents. *arXiv:1809.02627 [cs, stat]*, 2020.

JULIANI, A., KHALIFA, A., BERGES, V.-P., HARPER, J., TENG, E., HENRY, H., CRESPI, A., TOGELIUS, J. and LANGE, D. Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning. 2019. _eprint: 1902.01378.

JUSTESEN, N., TORRADO, R. R., BONTRAGER, P., KHALIFA, A., TOGELIUS, J. and RISI, S. Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation. 2018.

KARA, N. A Systematic Review of the Use of Serious Games in Science Education. *Contemporary Educational Technology*, volume 13(2), page ep295, 2021. ISSN 1309-517X.

KHAN, I., MELRO, A., CARLA, A. and OLIVEIRA, L. Systematic Review on Gamification and Cultural Heritage Dissemination. *Journal of Digital Media & Interaction*, volume vol. 3 n.º 8, pages 19–41 Páginas, 2020.

KOKIL, U. and HARWOOD, T. The interplay between perceived usability and quality in visual design for tablet game interfaces. *J. Usability Studies*, volume 17(3), pages 89–116, 2022. Place: Bloomingdale, IL Publisher: Usability Professionals' Association.

KOMÁRKOVÁ, Z. and KŘETÍNSKÝ, J. Rabinizer 3: Safraless Translation of LTL to Small Deterministic Automata. In *Automated Technology for Verification and Analysis* (edited by F. Cassez and J.-F. Raskin), pages 235–241. Springer International Publishing, Cham, 2014. ISBN 978-3-319-11936-6.

KONSTANTAKIS, M. and CARIDAKIS, G. Adding Culture to UX: UX Research Methodologies and Applications in Cultural Heritage. *J. Comput. Cult. Herit.*, volume 13(1), pages 4:1–4:17, 2020. ISSN 1556-4673.

KORHONEN, H. Evaluating Playability of Mobile Games with the Expert Review Method. 2016.

KOUTSABASIS, P. Empirical Evaluations of Interactive Systems in Cultural Heritage: A Review. *International Journal on Computational Methods in Heritage Science*, volume 1(1), pages 100–122, 2017.

KUMARAN, V., ROWE, J. and LESTER, J. NarrativeGenie: Generating Narrative Beats and Dynamic Storytelling with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 20(1), pages 76–86, 2024. ISSN 2334-0924, 2326-909X.

KUNIAVSKY, M., GOODMAN, E. and MOED, A. *Observing the user experience: a practitioner's guide to user research*. Morgan Kaufmann, Amsterdam ; Boston, 2nd ed edition, 2012. ISBN 978-0-12-384869-7.

LaValle, S. M. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S. and Kiela, D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. 2021. _eprint: 2005.11401.

Li, X., Serlin, Z., Yang, G. and Belta, C. A formal methods approach to interpretable reinforcement learning for robotic planning. *Science Robotics*, volume 4(37), page eaay6276, 2019. ISSN 2470-9476.

Li, X., Vasile, C.-I. and Belta, C. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3834–3839. IEEE, Vancouver, BC, 2017. ISBN 978-1-5386-2682-5.

Li, Z. Language-Guided Multi-Agent Learning in Simulations: A Unified Framework and Evaluation. 2025. _eprint: 2506.04251.

Liao, H.-C. A Survey of Reinforcement Learning with Temporal Logic Rewards. 2020.

Lieberman, H., Paternò, F. and Wulf, V. *End user development*. Number volume 9 in Human-computer interaction series. Springer, Dordrecht, 2006. ISBN 978-1-4020-4220-1.

Linaza, M. T., Torre, I., Beusing, R., Tavernise, A. and Etz, M. *Authoring Tools for Archaeological Mobile Guides*. The Eurographics Association, 2008. ISBN 978-3-905674-14-9.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F. and Liang, P. Lost in the Middle: How Language Models Use Long Contexts. 2023. _eprint: 2307.03172.

Liu, X., Zhao, R., Huang, P., Xiao, C., Li, B., Wang, J., Xiao, T. and Zhu, J. Forgetting Curve: A Reliable Method for Evaluating Memorization Capability for Long-Context Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing* (edited by Y. Al-Onaizan, M. Bansal and Y.-N. Chen), pages 4667–4682. Association for Computational Linguistics, Miami, Florida, USA, 2024.

Lovreto, G., Endo, A. T., Nardi, P. and Durelli, V. H. S. Automated Tests for Mobile Games: An Experience Report. In *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 48–488. IEEE, Foz do Iguaçu, Brazil, 2018. ISBN 978-1-5386-9605-7.

Macias, M. V., Kharlashkin, L., Huovinen, L. E. and Hämäläinen, M. Empowering Teachers with Usability-Oriented LLM-Based Tools for Digital Pedagogy. In *Proceedings of the 4th International Conference on Natural Language Processing for Digital Humanities* (edited by M. Hämäläinen, E. Öhman, S. Miyagawa, K. Alnajjar and Y. Bizzoni), pages 549–557. Association for Computational Linguistics, Miami, USA, 2024.

Majumder, A. and Majumder, A. Case Studies in ML Agents. *Deep Reinforcement Learning in Unity: With Unity ML Toolkit*, pages 513–552, 2021. Publisher: Springer.

Malegiannaki, I. and Daradoumis, T. Analyzing the educational design, use and effect of spatial games for cultural heritage: A literature review. *Computers & Education*, volume 108, pages 1–10, 2017. ISSN 03601315.

Marín-Lora, C., Chover, M., Martín, M. Y. and García-Rytman, L. Creating a treadmill running video game with smartwatch interaction. *Multimedia Tools and Applications*, volume 83(19), pages 57709–57729, 2024. ISSN 1573-7721.

McKay, R. I., Hoai, N. X., Whigham, P. A., Shan, Y. and O'neill, M. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, volume 11, pages 365–396, 2010. Publisher: Springer.

Merchant, B. AI Is Already Taking Jobs in the Video Game Industry. *Wired*, 2024. ISSN 1059-1028.

Miljanovic, M. A. and Bradbury, J. S. A Review of Serious Games for Programming. In *Serious Games* (edited by S. Göbel, A. Garcia-Agundez, T. Tregel, M. Ma, J. Baalsrud Hauge, M. Oliveira, T. Marsh and P. Caserman), pages 204–216. Springer International Publishing, Cham, 2018. ISBN 978-3-030-02762-9.

Mortara, M., Catalano, C. E., Bellotti, F., Fiucci, G., Houry-Panchetti, M. and Petridis, P. Learning cultural heritage by serious games. *Journal of Cultural Heritage*, volume 15(3), pages 318–325, 2014. ISSN 12962074.

Moser, C. and Fang, X. Effects of narrative structure and salient decision points in role playing games. *18th Americas Conference on Information Systems 2012, AMCIS 2012*, volume 3, pages 2375–2381, 2012.

Moser, C. and Fang, X. Narrative Structure and Player Experience in Role-Playing Games. *International Journal of Human-Computer Interaction*, volume 31, pages 146–156, 2014.

NG, A. and RUSSELL, S. Algorithms for Inverse Reinforcement Learning. *ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

NG, A. Y., HARADA, D. and RUSSELL, S. J. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 278–287. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. ISBN 1-55860-612-2.

NICHOLSON, S. Peeking behind the locked door: A survey of escape room facilities. Technical report, 2015.

NISANSALA, A., WEERASINGHE, M., DIAS, G. K. A., SANDARUWAN, D., KEPPITIYAGAMA, C., KODIKARA, N., PERERA, C. and SAMARASINGHE, P. Flight Simulator for Serious Gaming. In *Information Science and Applications* (edited by K. J. Kim), pages 267–277. Springer, Berlin, Heidelberg, 2015. ISBN 978-3-662-46578-3.

OLGERS, T. J., BIJ DE WEG, A. A. and TER MAATEN, J. C. Serious Games for Improving Technical Skills in Medicine: Scoping Review. *JMIR Serious Games*, volume 9(1), page e24093, 2021. ISSN 2291-9279.

OSTROWSKI, M. and AROUDJ, S. Automated Regression Testing within Video Game Development. *GSTF Journal on Computing (JoC)*, volume 3(2), page 10, 2013. ISSN 2010-2283.

O'NEILL, M. and RYAN, C. Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language, volume 4 of Genetic programming. 2003.

PALIOKAS, I. and SYLAIOU, S. The Use of Serious Games in Museum Visits and Exhibitions: A Systematic Mapping Study. In *2016 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*, pages 1–8. IEEE, Barcelona, Spain, 2016. ISBN 978-1-5090-2722-4.

PANT, Y. V., ABBAS, H. and MANGHARAM, R. Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1235–1240. IEEE, 2017.

PATHAK, D., AGRAWAL, P., EFROS, A. A. and DARRELL, T. Curiosity-driven Exploration by Self-supervised Prediction. 2017. _eprint: 1705.05363.

POLITOWSKI, C., GUÉHÉNEUC, Y.-G. and PETRILLO, F. Towards automated video game testing: still a long way to go. In *Proceedings of the 6th International ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation*, pages 37–43. ACM, Pittsburgh Pennsylvania, 2022. ISBN 978-1-4503-9293-8.

PUJOL-TOST, L. Did We Just Travel to the Past? Building and Evaluating With Cultural Presence Different Modes of VR-Mediated Experiences in Virtual Archaeology. *J. Comput. Cult. Herit.*, volume 12(1), pages 2:1–2:20, 2019. ISSN 1556-4673.

RAPTIS, G. E., FIDAS, C. and AVOURIS, N. Do Game Designers' Decisions Related to Visual Activities Affect Knowledge Acquisition in Cultural Heritage Games? An Evaluation From a Human Cognitive Processing Perspective. *J. Comput. Cult. Herit.*, volume 12(1), pages 4:1–4:25, 2019. ISSN 1556-4673.

RAU, L., BITTER, J. L., LIU, Y., SPIERLING, U. and DÖRNER, R. Supporting the creation of non-linear everyday AR experiences in exhibitions and museums: An authoring process based on self-contained building blocks. *Frontiers in Virtual Reality*, volume 3, 2022. ISSN 2673-4192.

RAWLINSON, R. E. and WHITTON, N. Escape Rooms as Tools for Learning Through Failure. *The Electronic Journal of e-Learning (EJEL)*, volume 22(4), 2024.

ROOM ESCAPE MAKER - INPI. ROOM ESCAPE MAKER. 2025.

ROSS, S., GORDON, G. J. and BAGNELL, J. A. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. 2011.

ROSS, S. I., MARTINEZ, F., HOUDE, S., MULLER, M. and WEISZ, J. D. The Programmer's Assistant: Conversational Interaction with a Large Language Model for Software Development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, IUI '23, pages 491–514. ACM, 2023.

ROUSSEEUW, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, volume 20, pages 53–65, 1987. ISSN 0377-0427.

ROUSSOU, M., PUJOL, L., KATIFORI, A., CHRYSANTHI, A., PERRY, S. and VAYANOU, M. The Museum as Digital Storyteller: collaborative participatory creation of interactive digital experiences. 2015.

ROWE, J. P., LOBENE, E. V., MOTT, B. W. and LESTER, J. C. Play in the Museum: Design and Development of a Game-Based Learning Exhibit for

Informal Science Education. In *Natural Language Processing: Concepts, Methodologies, Tools, and Applications*, pages 214–231. IGI Global Scientific Publishing, 2020. ISBN 978-1-7998-0951-7.

ROZIER, K. Y. *Explicit or Symbolic Translation of Linear Temporal Logic to Automata*. Thesis, Rice University, 2013.

SAGREDO-OLIVENZA, I., GUTIÉRREZ-SÁNCHEZ, P., GÓMEZ-MARTÍN, M. A. and GONZÁLEZ-CALERO, P. A. Liquid Gym: An Open Stealth Game Test-bed for AI Techniques Controlling Complex Characters. In *Actas del II Congreso Español de Videojuegos (CEV'24)*, pages 1–12. A Coruña, Spain, 2024.

SAMČOVIĆ, A. Serious games in military applications. *Vojnotehnicki glasnik*, volume 66(3), pages 597–613, 2018. ISSN 0042-8469, 2217-4753.

SAURO, J. and DUMAS, J. S. Comparison of three one-question, post-task usability questionnaires. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1599–1608. Association for Computing Machinery, New York, NY, USA, 2009. ISBN 978-1-60558-246-7.

SCHAFFHAUSER, D. Breakout! Gaming to Learn. *T.H.E. Journal Technological Horizons in Education*, volume 44, page 6, 2017.

SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A. and KLIMOV, O. Proximal Policy Optimization Algorithms. 2017. _eprint: 1707.06347.

SESTINI, A., BERGDAHL, J., TOLLMAR, K., BAGDANOV, A. D. and GISSLÉN, L. Towards Informed Design and Validation Assistance in Computer Games Using Imitation Learning. 2022a.

SESTINI, A., GISSLEN, L., BERGDAHL, J., TOLLMAR, K. and BAGDANOV, A. D. Automated Gameplay Testing and Validation with Curiosity-Conditioned Proximal Trajectories. *IEEE Transactions on Games*, pages 1–14, 2022b. ISSN 2475-1502, 2475-1510.

SESTINI, A., GISSLÉN, L., BERGDAHL, J., TOLLMAR, K. and BAGDANOV, A. D. Automated Gameplay Testing and Validation With Curiosity-Conditioned Proximal Trajectories. *IEEE Transactions on Games*, volume 16, pages 113–126, 2024.

SHORT, A.-R. Designing Fictional Worlds for Play Through Large Language Models. In *Volume 3B: 50th Design Automation Conference (DAC)*, page V03BT03A051. American Society of Mechanical Engineers, Washington, DC, USA, 2024. ISBN 978-0-7918-8837-7.

Sinha, S., Mandlekar, A. and Garg, A. S4RL: Surprisingly Simple Self-Supervision for Offline Reinforcement Learning. 2021. _eprint: 2103.06326.

Sintoris, C., Yiannoutsou, N., Ortega-Arranz, A., López-Romero, R., Masoura, M., Avouris, N. and Dimitriadis, Y. TaggingCreaditor: A tool to create and share content for location-based games for learning. In *2014 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL2014)*, pages 280–284. 2014.

Stahlke, S. ., Nova, A. and Mirza-Babaei, P. Artificial Playfulness: A Tool for Automated Agent-Based Playtesting. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, pages 1–6. Association for Computing Machinery, New York, NY, USA, 2019. ISBN 978-1-4503-5971-9. Event-place: Glasgow, Scotland Uk.

Sudhakaran, S., González-Duque, M., Glanois, C., Freiberger, M., Najarro, E. and Risi, S. MarioGPT: Open-Ended Text2Level Generation through Large Language Models. 2023.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. Bradford Books, 2nd edition, 2018.

Sánchez, A. M. Using digital educational escape rooms as a motivational review tool for Economics. *The International Journal of Management Education*, volume 21(3), page 100852, 2023. ISSN 1472-8117.

Sørensen, H. B. and Meyer, B. Serious Games in language learning and teaching – a theoretical perspective. In *Proceedings of DiGRA 2007 Conference: Situated Play*. 2007.

Tan, H., Yan, H. and Yang, Y. LLM-Guided Reinforcement Learning: Addressing Training Bottlenecks through Policy Modulation. 2025. _eprint: 2505.20671.

Taveekitworachai, P., Nimpattanavong, C., Gursesli, M. C., Lanata, A., Guazzini, A. and Thawonmas, R. Multiverse of Greatness: Generating Story Branches with LLMs. 2024.

Todd, G., Earle, S., Nasir, M. U., Green, M. C. and Togelius, J. Level Generation Through Large Language Models. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, pages 1–8. ACM, Lisbon Portugal, 2023. ISBN 978-1-4503-9855-8.

Toro Icarte, R., Klassen, T. Q., Valenzano, R. and McIlraith, S. A. Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning. *Journal of Artificial Intelligence Research*, volume 73, pages 173–208, 2022. ISSN 1076-9757.

TRICHOPOULOS, G. Large Language Models for Cultural Heritage. In *Proceedings of the 2nd International Conference of the ACM Greek SIGCHI Chapter*, pages 1–5. ACM, Athens Greece, 2023. ISBN 979-8-4007-0888-6.

TUCKER, A., GLEAVE, A. and RUSSELL, S. Inverse reinforcement learning for video games. 2018.

VAITHILINGAM, P., ZHANG, T. and GLASSMAN, E. L. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI EA '22. Association for Computing Machinery, New York, NY, USA, 2022. ISBN 978-1-4503-9156-6. Event-place: New Orleans, LA, USA.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L. and POLOSUKHIN, I. Attention Is All You Need. 2023. _eprint: 1706.03762.

VERSCHAFFEL, L., DEPAEPE, F. and MEVARECH, Z. Learning Mathematics in Metacognitively Oriented ICT-Based Learning Environments: A Systematic Review of the Literature. *Education Research International*, volume 2019, pages 1–19, 2019. ISSN 2090-4002, 2090-4010.

VINYALS, O., BABUSCHKIN, I., CZARNECKI, W. M., MATHIEU, M., DUDZIK, A., CHUNG, J., CHOI, D. H., POWELL, R., EWALDS, T., GEORGIEV, P. and OTHERS. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, volume 575(7782), pages 350–354, 2019. Publisher: Nature Publishing Group.

WEI, J., WANG, X., SCHUURMANS, D., BOSMA, M., ICHTER, B., XIA, F., CHI, E., LE, Q. and ZHOU, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2023.

WIEMKER, M., ELUMIR, E. and CLARE, A. Escape Room Games: "Can you transform an unpleasant situation into a pleasant one?". In *Game based learning* (edited by J. Haag, J. Weißenböc, M. W. Gruber, M. Christian and F. Freisleben-Teutscher), pages 55–68. Fachhochschule st Pölten GmbH, Austria, 2015.

WISE, D. and FORREST, S. *Great big book of children's games: over 450 indoor and outdoor games for kids*. McGraw-Hill, New York, 2003. ISBN 978-0-07-142246-8.

WOLF, M., MONTAG, M., SÖBKE, H., WEHKING, F. and SPRINGER, C. Low-Threshold Digital Educational Escape Rooms Based on 360VR and Web-Based Forms. *The Electronic Journal of e-Learning (EJEL)*, volume 22(4), 2024.

Yallihep, M. and Kutlu, B. Mobile Serious Games: Effects on Students' Understanding of Programming Concepts and Attitudes towards Information Technology. *Education and Information Technologies*, volume 25(2), pages 1237–1254, 2020. ISSN 1360-2357.

Yannakakis, G. N., Liapis, A. and Alexopoulos, C. Mixed-initiative co-creativity. 2014.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. and Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. *International Conference on Learning Representations (ICLR)*, 2023.

Yu, S., Liu, J., Yang, J., Xiong, C., Bennett, P., Gao, J. and Liu, Z. Few-Shot Generative Conversational Query Rewriting. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1933–1936. ACM, Virtual Event China, 2020. ISBN 978-1-4503-8016-4.

Yu-Han, C. and Chun-Ching, C. Investigating the Impact of Generative Artificial Intelligence on Brainstorming: A Preliminary Study. In *2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, pages 193–194. 2023.

Zahera, H. M., Ali, M., Sherif, M. A., Moussallem, D. and Ngomo, A.-C. N. Generating SPARQL from Natural Language Using Chain-of-Thoughts Prompting. In *SEMANTICS*, pages 353–368. 2024.

Zhao, Z. Application and Problems of AI in Game Development. *Applied and Computational Engineering*, volume 110, pages 13–21, 2024.

Zheng, Y., Xie, X., Su, T., Ma, L., Hao, J., Meng, Z., Liu, Y., Shen, R., Chen, Y. and Fan, C. Wuji: Automatic Online Combat Game Testing Using Evolutionary Deep Reinforcement Learning. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 772–784. IEEE, 2019.

Zi, Y., Li, L., Guha, A., Anderson, C. J. and Feldman, M. Q. "I Would Have Written My Code Differently": Beginners Struggle to Understand LLM-Generated Code. 2025. Publication Title: arXiv.org.

Part II

Papers of the Thesis

Chapter 6

Liquid Snake: a Test Environment for Video Game Testing Agents

Liquid Snake: a test environment for video game testing agents

Pablo Gutiérrez-Sánchez[1,*], Marco A. Gómez-Martín[1,*], Pedro A. González-Calero[1,*], Pedro P. Gómez-Martín[1,*]

[1]*Complutense University of Madrid, Madrid, Spain*

Abstract

In recent years, a number of benchmarks and test environments have been proposed for research on AI algorithms that have made it possible to evaluate and accelerate development in this field. There exists, however, an absence of environments in which to evaluate the feasibility of such algorithms in the context of games intended for continuous development, in particular in regression testing and automatic error detection tasks in commercial video games. In this paper we propose a new test-bed - Liquid Snake: a 3D third-person stealth game prototype, designed to conveniently integrate autonomous agent-driven quality control mechanisms into the development life cycle of a video game, based on the open source ML-Agents library in Unity3D. Focusing on the problem of regression testing on the potential unexpected changes induced in a game by altering the AI of enemies, we argue that this environment lends itself to be used as a sample test environment for automated QA methodologies thanks to the complexity and variety in the behaviors of NPCs naturally present in stealth titles.

Keywords

Benchmark, QA, regression testing

1. Introduction

In the continuous development of a commercial title, it is common to find numerous enemies and NPCs being reused in different sections of the game that evolve over time. In this context, the evolution of the AI controlling these agents can end up inducing non-negligible modifications in the gameplay of a level, contrary to the intentions of the original designs. These changes are not always straightforward to detect: on the one hand development teams often do not have the resources to perform sufficiently exhaustive testing tasks, and on the other hand these modifications do not necessarily have to "break" sections of the game, simply altering the user experience in a more or less subtle way and thus may go unnoticed by testers with extensive experience in the game.

The tests involved in the task of determining whether a feature or a design that was correct in the past continues to work properly after some progress in the development of the project

I Congreso Español de Videojuegos, December 1–2, 2022, Madrid, Spain

*Corresponding author.
pabgut02@ucm.es (P. Gutiérrez-Sánchez); marcoa@fdi.ucm.es (M. A. Gómez-Martín); pagoncal@ucm.es (P. A. González-Calero); pedrop@fdi.ucm.es (P. P. Gómez-Martín)
0000-0002-6702-5726 (P. Gutiérrez-Sánchez); 0000-0002-5186-1164 (M. A. Gómez-Martín); 0000-0002-9151-5573 (P. A. González-Calero); 0000-0002-3855-7344 (P. P. Gómez-Martín)

are known as regression tests, and their cost of execution grows dramatically as the volume of the code base and elements included in the game increases, as they essentially entail the recurring repetition of previously executed test batteries or checks. These tests may raise functional questions ("does the enemy continue to approach the player when it encounters them at a distance below a certain threshold?"), or more abstract questions linked to the game design ("is it possible to complete the level in less than 10 minutes and without losing health points?" or "is it possible to traverse the level while picking up all the collectibles and without being detected by enemies in the process?", to name a few examples). While the first types of questions can often be addressed through the use of tools such as unit tests, for which most commercial engines offer good support, this is typically not the case for the second type of problems, which generally require humans playing the game repeatedly trying to figure out an answer to the question posed by the test.

Since the latter poses both an economic and logistical bottleneck, different strategies have been proposed in recent years in an attempt to automate these checks and alleviate the burden they can place on a QA team. These include ideas such as replaying game traces recorded by human players during testing [1] or training autonomous agents based on AIs capable of interacting with the game in specific ways, each of them being more or less feasible to implement in a real development context [2, 3, 4].

While it is true that nowadays there are numerous standardized test-beds and benchmarks for machine learning techniques that are intended to act as "collective challenges" in the community to guide research efforts towards solving specific problems and to serve as environments for testing new algorithms, the same cannot be said for applications of such techniques to development cycles within commercial studios. Regression testing techniques described in the literature are often obtuse or inaccessible from the perspective of development teams, with proprietary use examples or technologies that are difficult to transfer to new environments.

It is therefore relevant, from our point of view, to propose a test-bed oriented to serve as a testing framework not only for machine learning algorithms, but also for their applications on quality control strategies in commercial games and the development of support tools for automated testing. Having a common environment also enables a shared vocabulary in the community: for instance, it becomes possible to compare the results of two ways of approaching a regression testing problem on specific and shareable modifications (such as "does my method detect that it is more difficult to evade level 2 enemies after altering a node of its behavior tree?").

On the other hand, from the developers' point of view, having a simple reference environment where they can try out automatic testing mechanisms allows them to experiment with different strategies in a smaller external project and validate them before taking the step of integrating them into their own games. At the same time, this test-bed can be used as an architectural reference for those teams that are considering undertaking automatic testing but are held back by the complexities and technical unknowns associated with the problem.

With this, in this paper we present Liquid Snake - a third-person 3D prototype belonging to the stealth genre and developed in Unity3D intended to act as a common test-bed for regression testing and automatic quality control, as well as an architectural reference for the integration of such methods in commercial projects. The rest of the paper is structured as follows. Section 2 discusses related work of interest in this field. Section 3 introduces Liquid Snake along with

the most relevant high-level dynamics within the game to provide a general understanding of the prototype. In turn, section 4 details the technical and architectural features that we argue make the project suitable for the uses described above, ending in section 6 with conclusions and future work.

2. Related work

In view of the problem described above, in recent years a number of strategies have been proposed to implement automatic testing methods, typically making use of autonomous control agents capable of interacting with a level over a large number of simulations while collecting metrics that must fall within specific ranges to consider that the user experience has not been altered [5, 6]. This same principle has been used in platforms and engines such as Unity3D to introduce tools to support the design and balancing of video games [7].

To create these control agents, one of the most straightforward alternatives is simply to make use of segments recorded manually by a human performing the specified tasks, replaying them every so often over the original environment to check that the player's trace is still able to complete the set objective [8]. However, when the structure of the environment is modified, or the environment includes random elements that do not remain constant between runs, these strategies are no longer valid, motivating the need to create agents with a certain capacity to adapt to changes in their surroundings.

This is where methods based on AI-based strategies come into play, offering more reactive and adaptive policies to changing environments. Some examples of these techniques for automatic testing in video games adopt the use of machine learning algorithms based on Deep Reinforcement Learning (DRL) [9, 10], Imitation Learning (IL) [11], or even hybrid models of the previous strategies with control structures such as behavior trees (BTs) [5].

Nonetheless, the reality of the matter is that at present the implementation of these methodologies in commercial developments suffers from several integration problems that discourage developers from employing them in their projects. First, the generation of autonomous agents capable of naturally playing a given game is complex and requires non-trivial knowledge in the field of machine learning to be implemented. This is cushioned to some extent by new tools such as ML Agents [12] in Unity3D or MindMaker [13] in Unreal Engine, which allow training policies in a more developer-friendly way, but these are mostly aimed at research or small-scale proof-of-concept creation. Secondly, to our knowledge there is a dearth of accessible benchmarks and environments that serve as showcases and examples for the use of automatic testing techniques in games intended for continuous development. Although there is a wide variety of benchmarks and environments designed to test different types of machine learning algorithms, such as DeepMind Lab [14], MineRL [15], OpenAI Gym [16] or Starcraft II Learning Environment [17], to name a few, all of them stem from a closed game in which the aim is to find a policy capable of fulfilling a certain fixed objective (or maximizing a given performance metric), as opposed to the problem that concerns us: starting from a game in open development and using agents to perform regression tests as it is modified.

In this paper we present a prototype stealth game implemented in Unity3D - Liquid Snake, as a contribution in progress to this field, intended to serve as an open source testing envi-

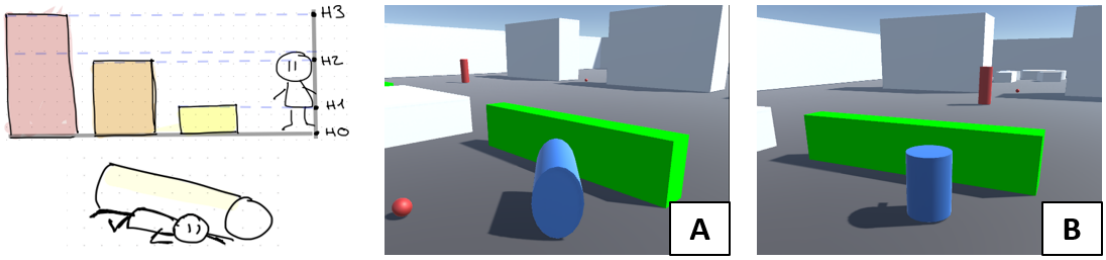


Figure 1: Height system in Liquid Snake. Captures from player in crawling (A) and crouching (B) states.

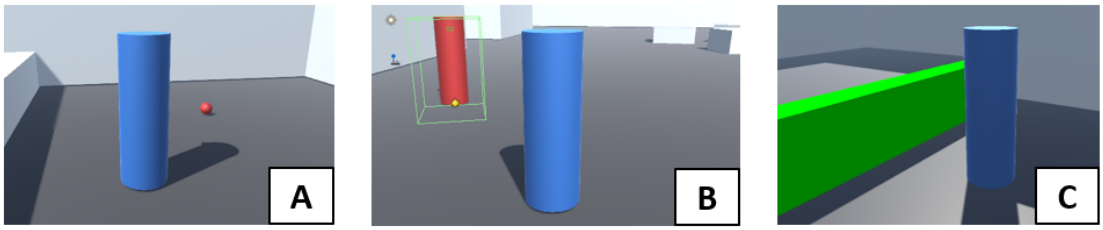


Figure 2: Sample interaction scenarios in Liquid Snake: picking up a collectible item (A), stealthily attacking an enemy from their back (B) and jumping over a log (C).

ronment that comes equipped with the necessary tools to experiment with automatic testing methodologies on a project under development.

3. Liquid Snake

Liquid Snake is a game with mechanics inspired mostly by the stealth genre: in it, the player must navigate through different 3D rooms in an attempt to find the exit, while trying to avoid the enemies that patrol the area in search of intruders and collecting as much loot as possible as they move through the environment. The player has a finite number of health points that are reduced every time they are hit by an enemy projectile, being defeated when these are reduced to 0. Players may choose to walk, run, crouch, or crawl to navigate the room, thus modifying the height at which enemies perceive them, their movement speed, and the noise they produce (which influences the enemies' ability to detect them).

The game considers two different heights for the player, as shown in figure 1. When deciding to walk or run, the player maintains the default height *H2*, the only difference between the two states being the speed of the character's movement and the noise produced (running being the faster but louder action). In the crouching and crawling states, the player adopts a height *H1*, which allows them to conceal themselves behind low obstacles such as the trunks in the figure so as to go unnoticed by enemies operating at a height *H2*. As with the walking-running pair, the only difference between the crouching and crawling states is the speed of the player's movement and the noise generated, with the crawling state being the slower but stealthier of the two.

The game also features interaction actions with objects in the environment and shooting

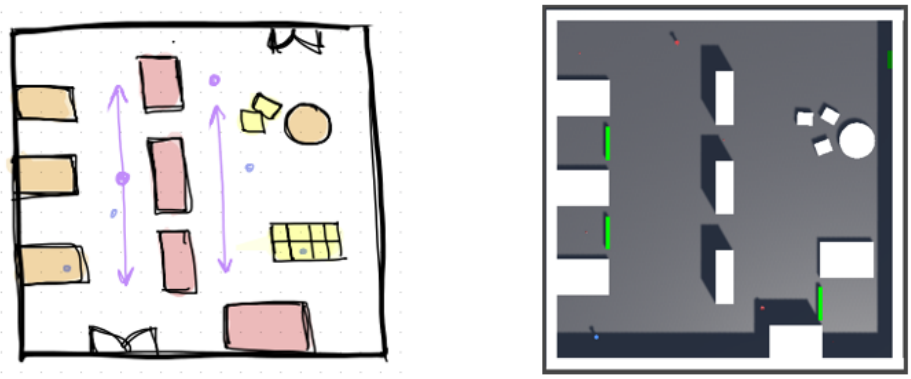


Figure 3: Sample level 1: disjoint enemy patrol paths separated by columns.

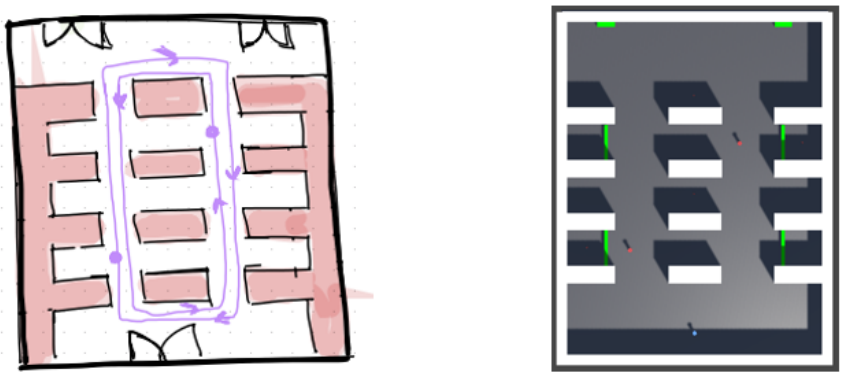


Figure 4: Sample level 2: clockwise overlapping enemy patrol paths.

with limited ammo to deal with certain vulnerable enemies. Some of the available interactions in the sample environments can be found in figure 2. This includes scenarios such as picking up a collectible item (figure 2-A), stealthily eliminating an enemy by interacting with it from its back before being detected (figure 2-B), or jumping over a low obstacle leaving the player positioned on the opposite side of the object (figure 2-C). In all cases, the player must approach the element with which they wish to interact and press an interaction key in order to execute the corresponding action. If there is more than one object with which an interaction can be performed, the one closest to the character at that time is selected.

The current project also includes some preconfigured levels that can be used as a reference for testing. All of them include at least one enemy with a predefined patrolling route and a number of collectibles hidden in various less accessible areas of the level, as well as other interactive elements such as jumpable logs. Some examples can be found in figures 3 and 4.

The decision to take a stealth game as a starting point for this test-bed is based on the fact that these environments are particularly appealing due to the presence of a number of mechanics that are very prevalent in a wide variety of genres and games and the presence of NPCs whose behaviors significantly determine the gameplay of the level. Indeed, in a game of this type it is

common to design enemies in very specific ways to motivate certain strategies on the part of the player, but also to increase the complexity of the AI so as to adapt it to new design needs, which is why we believe them to be an ideal candidate for regression testing.

The source code for the project, along with some usage examples, may be found in [18].

4. Testing environment features

In this section we will summarize the most relevant features of the proposed testing environment: its native integration with the ML Agents library in Unity 3D, the chosen architecture for the input and actions schemes to conveniently switch between control mechanisms, and the use of Behavior Trees to support growing enemy policies.

4.1. Natural integration with ML Agents

Liquid Snake is natively integrated with the ML Agents library for the training of autonomous agents that act as automatic testers and, as will be described later, allows to switch smoothly between manual control of the character and control by means of agents already trained or in training. Additionally, each level of the project is wrapped by a `LevelManager` to orchestrate the initialization and reboot processes of the environment between training episodes. To coordinate the resetting of the components within the environment, an `IResetteable` interface is provided with a single `Reset` method, to be implemented by whichever level constituents require a return to their initial configuration at the beginning of a new training episode, or when respawning the player after dying. The `LevelManager` is responsible for subscribing to all events in the environment that may result in a level reset (player death, time limit exceeded, etc) and maintains references to all elements adhering to the `IResetteable` interface to call their corresponding `Reset` methods each time one of these events is triggered in-game. This way, it is sufficient to implement this interface to ensure that a new level element is restarted when necessary, without the need to establish an explicit dependency between components, thereby facilitating the scalability of the project.

A `ChracterEvents` component is also provided for global notification of the different events of interest involving the character, which is particularly useful when linking occurrences with training rewards (for instance, granting a positive reward after the event of reaching the room exit) or with logistical operations on the scene (such as restarting the level from the `LevelManager` after the event of character death). In practice, all the events triggered by the various components that define the character's behavior (health, interactors, movement managers, etc.) end up being intercepted by the `CharacterEvents` component, which lifts and unifies them to provide a single access point from the outside to the catalog of events exposed by the character. As a result, it is only necessary to subscribe to the events of this component rather than having to access each component of interest individually.

Once a trained model is available, it is possible to perform simulations on the level of interest while collecting different customizable metrics related to the performance of the agents. At the moment, the project provides a set of Unity 3D play mode tests that can be used as a guideline to load scenes automatically, instantiate a standalone controller, associate it to the character object and then run the corresponding scene for a fixed number of times collecting event-driven

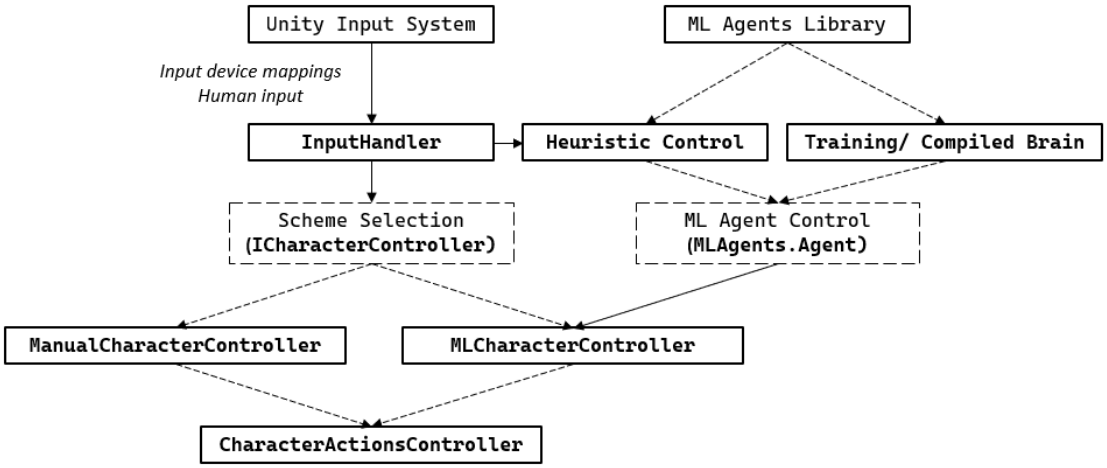


Figure 5: Control scheme in Liquid Snake.

metrics. This allows to gather simulation metrics from a fair number of levels in a convenient way, avoiding manual switches and executions. One limitation at the moment, however, is that no historical record is kept of the metrics collected in the tests, which makes it necessary to keep an external storage in which to place and analyze them retrospectively.

4.2. Decoupled input-actions scheme

One of the main bottlenecks when integrating a machine learning model with a test environment is the difficulty to conveniently switch between control schemes. The most classic examples of this situation are found in the need to alternate between manipulating the character manually, using a policy trained by reinforcement learning, executing traces prerecorded by human demonstrators, or applying hand-scripted routines to specify desired behaviors.

With this in mind, we consider it essential to start from a model in which the command modes mentioned above are kept strictly decoupled from the actual control logic of the character, by means of mechanisms natural to a programmer familiar with the engine. The proposed scheme is summarized in figure 5, for the case in which it is intended to alternate between manual control and control through an ML Agents library agent (either via heuristics, through a trained model or a model undergoing training).

In the previous scheme, there exists an abstraction layer that is always present within the character and acts as an interaction API with its set of supported actions, given by a component `CharacterActionsController` attached to the object to be handled. On top of this layer it is possible to place specific controllers that make direct use of this component to manipulate the character (`ManualCharacterController` and `MLCharacterController` in figure 5).

For those cases where one wishes to implement a controller that requires human input, such as in conventional controls or in the heuristic mode of ML Agents, we introduce an `InputHandler` object in the scene, responsible for transforming human input registered in Unity's input system into methods specific to any controller that implements the `ICharacterController` interface

(methods such as “Move”, “Shoot” or “Interact”, which are more manageable and understandable than raw input). It is worth mentioning here that the heuristic mode differs from traditional control in that in the former case the agent must receive the commands represented following the encoding used for the RL model, in an array of actions. This enables the collection of human demonstrations that can be later used to train learning-by-demonstration models or to reproduce historical traces. In this way, we are able to run a single input system, but operate it in different ways depending on our needs (for instance, one would not expect to use heuristic control as the main control in a commercial game).

To implement and deploy a character controller, we opt for a separation between the character actor object itself (a game object in the scene that exposes a `CharacterActionsController`) and the controller itself, which is nothing more than an empty object in the Unity scene with a component that implements the `ICharacterController` interface and is provided with a reference to the object that represents the character. In this way the character object acts as a “pawn”, and the controller can be modified as desired without further restructuring references from other objects in the scene to the player. This is convenient because in a game of these characteristics it is very common to have a multitude of elements that reference the player in a fairly direct way, being the enemies perhaps the most typical case, as they need to keep a reference to the player in order to know what to chase and attack, and to gather information from the player at run-time. If the controller were present in the character object itself, this would imply that to change the scheme it would be necessary either to replace the object in use with another one with different control components, or to allow the coexistence of numerous controllers in the same object and implement some kind of manager dedicated to enable and disable them as required, with the corresponding clutter and logistical complication that this may cause.

Additionally, when using a controller derived from the `Agent` class of ML Agents, it is common to introduce different sensor components associated to the agent to enable more sophisticated forms of perception such as visual observations from a camera, or spatial observations by means of sensors based on raycasts around the object or grids centered on the agent to detect the relative position of nearby elements in the scene. These sensors must be attached as components to the object that hosts the controller and will always dispatch their observations without the ability to disable them. Since it is usually desirable to experiment with various configurations of sensors and parameters, it seems natural to keep each controller as an independent object that can be easily replaced whenever one wishes to enforce a new perception system. One thing to note here is that if one intends to use a sensor centered on the player’s object, then it becomes necessary to ensure that the position of the controller matches that of the character, which is typically rather straightforward, but important to note nonetheless.

4.3. Behavior trees for rich NPCs

As we mentioned before, in the development of commercial games with enemies and other NPCs, it is common to start with a relatively simple AI that meets the design needs of the initial levels and then evolve and expand it to adapt to new requirements as the project progresses. These AIs are usually coded using behavior trees (BTs) or state machines, which is why in Liquid Snake we include the Behavior Bricks library [19] for specifying the behavior of enemies in the form of BTs. From this library it is possible to design arbitrarily complex flows that can

be expanded during development.

The current project features a suite of pre-designed behavior trees to define the flows of several enemies present in the example levels that may be used as a reference for changes or expansions. Included here are nodes encoding common conditions and checks such as "is target in sight?" or actions such as "advance to the next patrol point" or "shoot at target".

5. Testing process

Following the descriptions given in the previous sections, the process of creating a battery of tests on one of the project's environments would be as follows:

1. Set up a level on which one wishes to perform a regression test as a scene within the game. Currently the levels in figures 3 and 4 are provided as testing samples (for which pre-trained controllers and Play Mode tests that make use of them are provided).
2. Configure a controller that is able to automatically perform the desired behavior at the created level. This can be done in a number of ways, either with a manually programmed AI, with traces of human inputs or, as in the case of the examples in the project, by training an agent through the ML-Agents library to take control of the player. As for the behavior to be generated, this need not be limited to successfully completing the level, but can specify more precise tasks, such as defeating a particular enemy or reaching a sequence of points in order. In the example cases, the goal of the level is always to reach an escape point before dying or running out of time, with a reward function that penalizes enemy deaths and damage, and awards positive stimuli for approaching the goal, reaching an exit point, or retrieving a collectible item. The implementation details for these controllers (observations, reward functions, configuration of the underlying neural network, etc) can be consulted in the project repository, where a quick start guide explaining a simple training process is also included.
3. Write a Play Mode test in Unity's Test Runner that instantiates the implemented controller on the scene to be tested and binds it to the player object in the environment. In general, a test runs the controller simulation over the scene for a set number of times (which should be high enough to be in proportion to the variability of the level in order to collect as many occurrences as possible) and collects execution metrics configured by the test designer, such as number of objects retrieved or number of times that the player is detected by the enemy. These metrics can make use of the player's `CharacterEvents` component to be constructed as certain standardized events are received. In the example cases, damage to the player and goal reached events are used to compute the metrics per episode of health remaining at the end of the level and time taken to escape from the room. The test executor is also responsible for compiling the execution metrics and dumping them into a log file, thus enabling subsequent analysis.
4. After applying a structural change on the scene, re-run the previous test to obtain a new set of simulation metrics with the same agent. At this point it is possible to perform a comparative analysis of the distributions of the metrics (applying, for example, a non-parametric test such as Mann-Whitney to contrast the equality of pre- and post-change

distributions) or, if preferred, simply check if they are within acceptable limits given by the design team.

The use of a Test Runner such as the one integrated in Unity allows to configure and launch large numbers of tests on an arbitrary number of scenes in an automated way, so that it is possible to repeat the capture of metrics at the end of a development day as a first verification mechanism to check whether the metrics are maintained at appropriate values.

6. Conclusions and future Work

In this paper we present a test environment for automatic regression testing in Unity 3D as a contribution to the field of quality control in commercial video games, arguing that it can be used both as a reference for the integration of automatic testing methodologies in new projects, and to evaluate new testing algorithms in a controlled and prepared environment before deciding to proceed to incorporate them into a proprietary project. In particular, we conclude that the architecture proposed in this paper offers a number of features that make it particularly suitable for automated testing, such as its native integration with machine learning libraries such as ML-Agents, a decoupled scheme of character controllers that allows seamless switching between input and control mechanisms, and reference scenarios that exemplify its use in Unity's Play Mode Tests for the automated execution of multiple batteries of simulations.

The test-bed described in this article is under active development, and we aim to incorporate new features that make it as easy as possible to perform tests on it, as well as to develop thoroughly documented application examples of automatic testing techniques such as those described in [5] as a showcase of the environment. In the short term, the highest priority is to improve the tools to conveniently configure and perform the automatic execution of a large number of tests with autonomous agents, as well as to collect the execution results of such tests in a structured and manageable way. This will enable a good degree of control over the evolution of the set of game sections avoiding the tedium of launching each regression test manually.

7. Acknowledgments

This work was supported by the Ministry of Science and Innovation (PID2021-123368OB-I00).

References

[1] M. Ostrowski, S. Aroudj, Automated Regression Testing within Video Game Development, GSTF Journal on Computing (JoC) 3 (2013) 10. URL: http://www.globalsciencejournals.com/article/10.7603/s40601-013-0010-4. doi:10.7603/s40601-013-0010-4.

[2] J. Pfau, J. D. Smeddinck, R. Malaka, Automated Game Testing with ICARUS: Intelligent Completion of Adventure Riddles via Unsupervised Solving, in: Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play, ACM,

Amsterdam The Netherlands, 2017, pp. 153–164. URL: https://dl.acm.org/doi/10.1145/3130859.3131439. doi:10.1145/3130859.3131439.

[3] S. Ariyurek, A. Betin-Can, E. Surer, Automated Video Game Testing Using Synthetic and Humanlike Agents, IEEE Transactions on Games 13 (2021) 50–67. doi:10.1109/TG.2019.2947597.

[4] J. Bergdahl, C. Gordillo, K. Tollmar, L. Gisslén, Augmenting Automated Game Testing with Deep Reinforcement Learning, in: 2020 IEEE Conference on Games (CoG), 2020, pp. 600–603. doi:10.1109/CoG47356.2020.9231552, iSSN: 2325-4289.

[5] P. Gutiérrez-Sánchez, M. A. Gómez-Martín, P. A. González-Calero, P. P. Gómez-Martín, Reinforcement Learning Methods to Evaluate the Impact of AI Changes in Game Design, Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment 17 (2021) 10–17. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/18885.

[6] P. Gutiérrez-Sánchez, M. A. Gómez-Martın, P. A. González-Calero, P. P. Gómez-Martın, A proposal for combining reinforcement learning and behavior trees for regression testing over gameplay metrics, VII Congreso de la Sociedad Española para las Ciencias del Videojuego 2021 (2021). URL: http://ceur-ws.org/Vol-3082/paper13.pdf.

[7] Optimize your game balance with Unity Game Simulation, 2020. URL: https://blog.unity.com/technology/optimize-your-game-balance-with-unity-game-simulation.

[8] M. Ostrowski, S. Aroudj, Automated Regression Testing within Video Game Development, GSTF Journal on Computing (JoC) 3 (2013) 10. URL: https://doi.org/10.7603/s40601-013-0010-4. doi:10.7603/s40601-013-0010-4.

[9] J. Bergdahl, C. Gordillo, K. Tollmar, L. Gisslen, Augmenting Automated Game Testing with Deep Reinforcement Learning, in: 2020 IEEE Conference on Games (CoG), IEEE, Osaka, Japan, 2020, pp. 600–603. URL: https://ieeexplore.ieee.org/document/9231552/. doi:10.1109/CoG47356.2020.9231552.

[10] J. Pfau, J. D. Smeddinck, R. Malaka, Automated Game Testing with ICARUS: Intelligent Completion of Adventure Riddles via Unsupervised Solving, in: Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play, ACM, Amsterdam The Netherlands, 2017, pp. 153–164. URL: https://dl.acm.org/doi/10.1145/3130859.3131439. doi:10.1145/3130859.3131439.

[11] S. Ariyurek, A. Betin-Can, E. Surer, Automated Video Game Testing Using Synthetic and Humanlike Agents, IEEE Transactions on Games 13 (2021) 50–67. URL: https://ieeexplore.ieee.org/document/8869824/. doi:10.1109/TG.2019.2947597.

[12] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, D. Lange, Unity: A General Platform for Intelligent Agents, arXiv:1809.02627 [cs, stat] (2020). URL: http://arxiv.org/abs/1809.02627, arXiv: 1809.02627.

[13] A. Krumins, Mind maker, 2020. URL: https://github.com/krumiaa/MindMaker.

[14] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, S. Petersen, DeepMind Lab, 2016. URL: http://arxiv.org/abs/1612.03801. doi:10.48550/arXiv.1612.03801, arXiv:1612.03801 [cs].

[15] A. Kanervisto, S. Milani, K. Ramanauskas, N. Topin, Z. Lin, J. Li, J. Shi, D. Ye, Q. Fu, W. Yang, W. Hong, Z. Huang, H. Chen, G. Zeng, Y. Lin, V. Micheli, E. Alonso, F. Fleuret, A. Nikulin,

Y. Belousov, O. Svidchenko, A. Shpilman, MineRL Diamond 2021 Competition: Overview, Results, and Lessons Learned, 2022. URL: http://arxiv.org/abs/2202.10583. doi:10.48550/arXiv.2202.10583, arXiv:2202.10583 [cs].

[16] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, 2016. arXiv:arXiv:1606.01540.

[17] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, R. Tsing, StarCraft II: A New Challenge for Reinforcement Learning, 2017. URL: http://arxiv.org/abs/1708.04782. doi:10.48550/arXiv.1708.04782, arXiv:1708.04782 [cs].

[18] P. Gutiérrez-Sánchez, M. A. Gómez-Martın, P. A. González-Calero, P. P. Gómez-Martın, Liquid Snake, 2022. URL: https://github.com/UCM-GAIA/Liquid_Snake.

[19] PadaOne Games, BehaviorBricks, 2021. URL: http://bb.padaonegames.com/.

Chapter 7

Liquid Gym: An Open Stealth Game Test-bed for AI Techniques Controlling Complex Characters

Actas del II Congreso Español de Videojuegos (CEV'24), A Coruña, España, 17–19 de junio de 2024

Liquid Gym: an open stealth game test-bed for AI techniques controlling complex characters*

Ismael Sagredo-Olivenza[1*†], Pablo Gutiérrez-Sánchez[1*†], Marco A. Gómez-Martín[1*] and Pedro A. González-Calero[1*]

[1*]Software Engineering and Artificial Intelligence Department, Complutense University of Madrid, Facultad de Informática, Madrid, Span.

*Corresponding author(s). E-mail(s): isagredo@ucm.es; pabgut02@ucm.es; marcoa@fdi.ucm.es; pagoncal@ucm.es;
†These authors contributed equally to this work.

Abstract

This paper introduces Liquid Gym, a training and testing environment for various machine learning models in Unity. Liquid Gym extends the functionalities of ML-Agents and OpenAI Gym, facilitating the introduction of new algorithms, diverse testing environments, and comparisons with state-of-the-art classical control systems from the video game industry, such as Behavior Trees, Hierarchical Task Networks, or Finite State Machines. Currently, we have implemented both imitation learning and reinforcement learning approaches within a stealth game testing environment named Liquid Snake. Our intention is to broaden the range of supported models and testing environments in the future.

Keywords: Reinforcement Learning, Machine Learning, Video Games, Environments

1 Introduction

After DeepMind began to enjoy great success in the use of Deep Q-Learning (DQN) in different environments, originally in Atari games [1] and later in much more complex cases such as AlphaGo [2] or the Starcraft video-game [3], numerous applications of

*This work was has been partially supported by the Spanish Ministry of Science and Innovation under grant PID2021-123368OB-I00.

this algorithm have been developed successfully. The results achieved by DeepMind did not go unnoticed by academia, as can be seen at [4–6].

Apart from DeepMind, another of the main contributors to the diffusion of DQN has been OpenAI which, in addition to multiple works related to Reinforcement Learning (RL) such as OpenAI Five [7], has developed a testing environment for RL-driven agents, initially called Gym [8] and nowadays re-branded as Gymnasium [1]. This environment enables AI practitioners to test different machine learning algorithms with a large collection of sample environments, ranging from the aforementioned Atari games, to many other classic problems such as Cart Pole, Pendulum, Mountain Car, etc.

This platform and its API have become a benchmark for the evaluation and comparison of different RL algorithms, as in addition to the built-in environments, there are many third-party ones based on Gymnasium, such as Flappy Bird [2], Slime Volleyball Gym Environment [3], or stable-retro [4], with different implementations of classic games such as Super Mario Bros 2 Japan (Lost Levels in the West) from Nintendo Entertainment System (NES), Punch Out (NES), Tetris (Game Boy), Virtua Fighter (32X) or Mortal Kombat (Genesis), among others.

Most of these implementations are either relatively small games (such as Slim Volley or Flappy Bird) or rely on an emulator due to them being titles developed for older consoles, as in the case of stable-retro. This framework is therefore suitable for training and evaluating agents engaging with a selection of simple games, but falls short for assessing the ability of RL-based strategies and other machine learning and AI techniques to be valuable in other major areas of the video-game industry, such as controlling the behaviour of non-playable characters (NPCs) or performing automated testing tasks, on the one hand due to the lack of complexity or closeness to modern games in terms of mechanics and scale, and on the other because of the inability to access and modify the source code of these environments.

One of the most successful implementations compatible with the Gymnasium API, and which partially addresses the limitations of the other test environments included in Gymnasium, is ML-Agents [9]. This tool allows developers to create games built on the popular Unity3D engine [10] and use them as test environments for RL and Imitation Learning (IL) algorithms. The ML-Agents Toolkit has been used in several research works [11, 12], inspiring the creation of challenges and agent creation competitions in complex environments such as Obstacle Tower [13], as well as in commercial games such as Source of Madness [5], where ML-Agents was used to generate the behaviours of a number of enemies. Today, ML-Agents offers implementations of some state-of-the-art algorithms in reinforcement learning (such as PPO [14] or SAC [15]) and imitation learning (such as Behavioural Cloning [16] or GAIL [17]), as well as additional extensions to include reward generation modules based on intrinsic motivation or curiosity through Random Network Distillation [18], curriculum learning, and even neural architectures based on Hyper Networks [19] to dynamically adapt agents'

[1] https://gymnasium.farama.org/
[2] https://github.com/markub3327/flappy-bird-gymnasium
[3] https://github.com/hardmaru/slimevolleygym
[4] https://github.com/Farama-Foundation/stable-retro
[5] https://sourceofmadness.com/

behaviours based on target signals or modules supported by transformers and attention systems [20] to receive variable-length observations. Despite these virtues, this toolkit does come with certain limitations. Just like Gymnasium, it is closely aligned to the use of RL, as well as to specific Python versions and frameworks (Keras or Pytorch) and, despite the recent incorporation of Trainer Plugins to support custom implementations of the training interfaces provided by ML-Agents, these are still very limited, making it difficult to test other algorithms beyond those included in the tool.

There are also other development frameworks that simplify the creation of agents in games such as Serpent.AI [6], a project that stopped in 2018 but seems to have timidly resurfaced. This framework helps the creation of ML agents in already created games.

The purpose of this work is to define a test environment which we call Liquid Gym in a similar spirit to Gymnasium, but which allows the comparison between different techniques and use cases beyond RL and gameplay in a transparent and unified way within a single tool. This includes, for example, imitation learning methods, evolutionary algorithms and other more traditional agent control systems such as Beahvior Trees (BTs) [21], Finite State Machines (FSMs) or Hierarchical Task Network Planning HTN, and we emphasise the use case of automatic testing in video-games as a major practical application of these techniques outside of gameplay. To this end, we iterate on our testing environment called Liquid Snake, around which we establish the foundations of this system. Currently we have performed preliminary experiments with reinforcement learning (PPO) and imitation learning (Behavioural Cloning + GAIL) in this environment [22], and are planning to extend this work with additional settings and algorithms, which we will describe in more detail in this article.

The remainder of this paper is structured as follows. Section 2 reviews related work. Then, Section 3 presents our main testing environment. Section 4 describes the general architecture of the system. Lastly, Section 5 presents the preliminary work conducted on this framework so far along with some conclusions and future lines of work.

2 Related work

In recent years, advances in the field of machine learning have been progressively integrated into the field of video game development, both in academia and industry, offering innovative solutions to persistent problems that are difficult to solve using classical methods. Perhaps the best known and most mediatised area of this incursion of machine learning into video games is the creation of agents capable of interacting at human or even superhuman levels in classic games. Some emblematic examples are those already mentioned in Section 1, such as DeepMind's AlphaGo player, as well as the application of deep neural networks in Atari games. The academic articles associated with these advances boast, at the time of writing, 17600 and 13700 citations respectively, evidencing their impact on research in less than 8 years.

Another context where AI agents are experiencing a remarkable growth is in the automated testing and quality control of video games. The quality control of software applications in general is a very costly process in which it is necessary to make use of a large amount of both technical and human resources to detect problems in the

[6] https://github.com/SerpentAI/SerpentAI

product before and during its release to the public; video games in particular exhibit an extraordinary complexity as far as software is concerned, and their quality control processes are often accompanied by an additional type of analysis known as playtesting, whereby a set of testers play the game in search of bugs and/or problems with the game design, usually requiring a substantial investment of time and money.

In this sense, having automated agents that are capable of autonomously and regularly playing the game under development in search of errors is a highly attractive idea for the industry, and has been the focus of multiple research projects in recent years. [23] for example uses an automated testing methodology using synthetic, near-human agents focused on finding defects in environments. The use of inverse reinforcement learning techniques here improves both the agents' error-finding ability and their perceived humanness when interacting with the game. [24] uses deep reinforcement learning to explore or exploit game mechanics based on a user-defined reinforcement signal in a first-person shooter (FPS) game. In turn, [25] makes use of generative player modelling methods from "procedural people" over Monte Carlo techniques to generate different play styles through different utility functions and evolutionary algorithms.

Meanwhile, some of the most prominent development companies in the video game industry have conducted and published research along these lines, looking for ways to automate the various quality control processes they encounter in their own projects. Electronic Arts (EA), for instance, proposes a set of deep reinforcement learning methods in which the agent is motivated to explore in the proximity of known trajectories derived from expert demonstrations, in a complex 3D environment using a strategy called Curiosity-Conditioned Proximal Trajectories, in order to generate agents that are able to replicate and augment checks and behaviours taught by traces of human players in differents games [26] even in commercial games such as Battlefield V [27]. Along the same lines of the "humanisation" of agents and the need for them to be able to replicate the playing styles of human players, an imitation learning algorithm (Multimodal GAIL) capable of creating autonomous agents that exhibit different playing styles without the need to go through manual reward engineering is described [28].

Continuing with these examples, in [29], a reinforcement learning algorithm is introduced for the generation of agents capable of performing coverage testing of 3D environments by trying to generate a connectivity graph of the different regions of the level during the simulation. Finally, the developer in charge of Candy Crush Saga, discusses a series of strategies based on convolutional neural networks (CNNs) to predict the most "human-like" movement on a game board in order to estimate the difficulty of a new level as soon as it is designed, without the need for playtesting [30].

The key challenge in most of these works, however, appears to be the lack of open source test environments developed on popular game engines that make it possible to reproduce and compare each methodology and domain application on use cases close to those that would be expected in the industry. This is where this work comes into play, proposing an open environment so that different AI practitioners and trainees can find a common ground to benchmark and explore old and new approaches to the various problems of everyday game development.

3 Environment description

Liquid Snake, originally introduced in a preliminary version in [31], is a prototype 3D third-person stealth game built in Unity3D and developed to conduct experiments to test AI techniques in an environment with features close to those of a prototypical commercial video game. In this game, the player's objective is to guide the main character through a series of rooms and locations in search of the level's exit, while avoiding detection by enemies patrolling the area and solving various puzzles. Figure 1 provides a screenshot of the game's appearance in a sample level.

Fig. 1 Top-down capture of a level section from Liquid Snake.

Enemies in this game follow typical stealth game behaviour, sequentially traversing a series of patrol points specified by the level designer, and stopping at each of them for a short period of time to scout the surroundings. When the enemy detects the character within its cone of vision (reflected in Figure 1 using a pink robotic enemy), it chases and shoots at them concurrently until their health points are reduced to 0 or they are out of sight. In the latter case, the enemy proceeds to survey the last place in the environment where it sensed the player and, if it still cannot find the player in the area, returns to its designated patrol route.

In turn, the player has a number of movement options and actions to interact with the environment. Apart from basic movement on the XZ axis by walking or running, the character can jump to overcome low obstacles or reach high platforms that would otherwise be inaccessible, as well as assume a crouched position to hide from enemies behind mid-height obstacles or maneuver under low-ceiling sections. Additionally, the character can interact with certain objects in the scene by approaching their vicinity (as in the vending machine in the figure surrounded by a circular yellow area), triggering different effects on the environment. One of the most typical examples of these

interactions is the presence of triggers that enable or disable laser barriers such as those on the right side of the figure, which serve to block areas.

The game is designed to allow straightforward switching between control systems according to the needs of each use case. To this end, the character object exhibits a series of methods that enable the different actions described above to be invoked, which are then called from different controllers depending on the situation. If the intention is for a human to play in a traditional way, the input is provided by Unity's input system, mapping each keyboard or controller entry with one of the available actions; on the other hand, when the intention is for an AI controller to manipulate the character, a series of specific actuators are provided for each controller type to translate the intentionally of the agent and model used into calls to the corresponding actions. Thus, the decision of how an agent's output is converted into a character's behaviour is entirely in the hands of the implementer. This facilitates working with a wide variety of models such as neural network-based systems, rule-based approaches, behaviour trees or planners, to name a few, each of which can choose to define actions with different degrees of granularity (discrete, continuous, high- or low-level).

Currently, Liquid Snake comes integrated with the Unity ML-Agents library, described earlier, which provides the necessary support to start testing reinforcement and imitation learning methodologies on our environment. In addition, ML-Agents provides its own wrapper on top of the typical Gym environment interfaces, allowing interaction with environments that make use of the tool through an API well known to developers familiar with Gym. Additionally, we include the Behavior Bricks tool for programming and designing the behaviour of both NPCs and gameplay bots through Behavior Trees, as well as Unity's native support for state machine programming through Visual Scripting. Also available as an optional module for this approach is our behaviour generation tool via reinforcement learning and formal task specifications, AI Behaviour Graphs, which has been largely tested on Liquid Snake environments. These features make this an attractive testing and experimentation environment not only from the viewpoint of game AI practitioners, but also from a pedagogical perspective, allowing students of AI or game programming disciplines to apply the skills they have learned in scenarios close to real games.

4 Architecture

In this section we describe the architecture we consider most suitable for the centralisation of AI experiments on our environment, including explanations of features already included in our framework as well as desirable properties we are currently working on.

Our current system is designed to execute ML models within the game's own code (either through CPU or GPU using Computer Shaders) by means of the Unity's Sentis library, while training is carried out in Python in an engine-independent way. Communication between both parts of the system is managed via a network module enabling the exchange of information via TCP/IP connection. A high-level schematic of the architecture is shown in Figure 2.

As depicted in the diagram, depending on the algorithm's requirements we can have multiple instances of the game connected to the training module. This is particularly

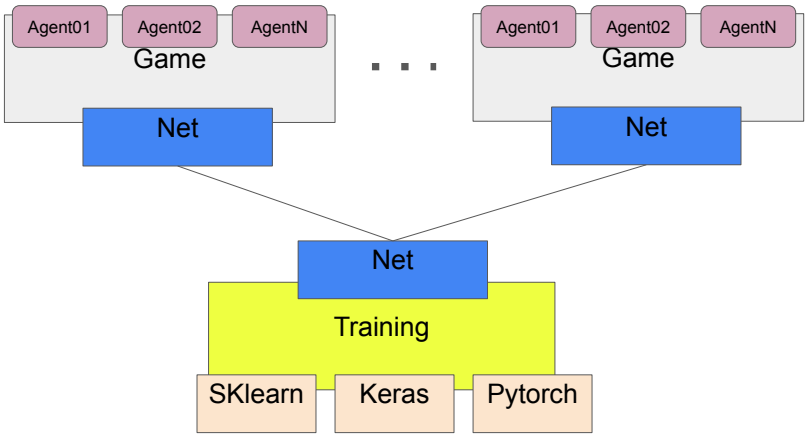


Fig. 2 High-level diagram of the proposed architecture

attractive when one wishes to use a training algorithm that is well conditioned to be parallelised, as may be the case for evolutionary algorithms (where the major cost of execution corresponds to the evaluation the fitness of each individual in the population) or even for RL models where it is necessary to gather significant experience on the environment before making updates to the existing policy. Therefore, the tool enables the creation of different game instances and links them to the trainer module if the model requires it, with the training module assuming the role of server. This separation of training and execution further allows us to integrate the models into the final game and move beyond utilizing the tool solely as a testing and research bench.

It is also important to mention here that the game instances connected to the training server do not always need to only include a single agent. On the one hand, from a conceptual standpoint we may be interested in training several policies or agents simultaneously, such as a bot to represent the player, an enemy or NPC behaviour, a group of collaborative or competitive AIs interacting with each other, or even more abstract AIs learning to control or modify elements of the level or game flow. On the other hand, even in cases where we simply wish to focus on a single behaviour, it is often beneficial from a learning optimisation point of view to include numerous agents in each environment to increase, for instance, the rate of experience collection in RL, or the number of individuals in the population in the case of evolutionary algorithms.

Regardless of the type of model to be used, we will be interested in centralising the training phases in Python in view of the vast collection of resources that this language has at its disposal in the field of ML. Generally, the Python side is only needed during the policy generation phase of our agents, delegating the often complex to implement ML algorithms to widely adopted external libraries, after which it is possible to compile a serializable model of the learned behaviour (e.g., an ONNX file if we are training neural networks). Once a compiled model is available, it is then often straightforward to transfer it to an appropriate game engine-side executor, either programmed manually or using a third-party one, such as Unity's Sentis, continuing the example of the

ONNX files. In this way, the model becomes independent of the Python module and can be used in production. During training, however, it is the Python service that is in charge of receiving environmental information from the game (observations of the agents in the case of RL, or fitness values in the case of evolutionary algorithms) and sending back instructions on how the agents should behave (actions on each tick of the game in RL, or complete neural networks in the evolutionary case). A summary of the most typical use cases in this context would be as follows:

- RL. The service is provided at each game step with the environment observations collected by the agent (e.g. its position, orientation or distance to objects of interest) and the reward signal granted by the game at that time and forwards a representation to the game of the actions that the agent should perform based on the current model on that state. Depending on the type of RL algorithm used, after collecting a certain number of experiences in this way with the current policy, the service will generate new policies that progressively improve the accumulated reward.
- Evolutionary algorithms. The service generates a collection of agents capable of interacting with the game and progressively sends them to the connected game instances for evaluation. After each training episode and for each agent considered as an individual in the current population, the game computes a numerical value representing the individual's quality or fitness, or how well it has managed to complete the assigned task. After receiving the fitness measures of all the individuals in the population, the algorithm produces a new population of (ideally) improved individuals and sends them back to the game for evaluation.
- Imitation learning. The system allows us to record traces of actions performed by the human player, which can then be utilised in Python to train the models via a demonstrations file sent to the service. In simpler versions of this paradigm, such as Behavioural Cloning, it may not be necessary to establish any kind of communication with the game to generate a policy, as the focus is only on producing a network that best fits the input data. If we use an adversarial model such as GAIL, the service acts like the RL case with the difference that the network reward is not given by the game, but by the agent's ability to outsmart an adversarial network trained to distinguish between human and bot behaviours. In the future, we plan to include implementations of interactive algorithms such as DAGGER that allow for a training flow in which the designer can take control of the player to provide real-time demonstrations (e.g., to correct faulty behaviour or explain how to solve new cases), from which the training service dynamically adjusts the policy.

The system, although initially designed to execute neural networks, could easily be extended to other classical ML algorithms such as Random Forest, K-NN, SVM, etc., allowing for comparisons between different approaches depending on the problem we want to solve and the data available. This feature is particularly interesting for use in an academic environment where students can try out different options to understand the functioning and advantages/disadvantages of different models.

Lastly, for classical control systems such as BT or HTN, we have chosen to incorporate them directly into the game engine. Currently, we have Behavior Bricks [32] integrated into Liquid Snake, with plans to include HTN in the future. The aim of

including these classical control is twofold: firstly, to build hybrid models with ML [33], as most current ML implementations cover only certain parts of the behaviour; secondly, to allow for performance comparison between ML models and these classical control systems, which are the most widely used in the video-games today.

5 Preliminary and future work. Conclusions

The first experiments carried out on Liquid Snake were focused on implementing different reinforcement and imitation learning techniques within the field of automated video-game testing [22]. In particular, our main goal was to analyse the feasibility of various methodologies to generate agents capable of dealing with regression testing and level design validation tasks, understood here as those tests that receive a specification usually expressed as a sequence of steps and checks to be executed on the environment and generate a report indicating whether it is still feasible in the current state of the game. Preliminary results on 3 Liquid Snake use cases and levels enabled comparing the suitability of strategies based on PPO guided by dense reward functions with classical IL algorithms such as Behavioral Cloning and GAIL to generate agents capable of carrying out this testing modality, suggesting an advantage of the former over the latter. Training was performed in parallel on different instances of Liquid Snake connected to a single Python service, and includes tasks such as traversing a level with complex navigation, sequentially visiting switches in environments with patrolling enemies, or completing areas with dynamic branching conditions.

Apart from methodologies based on the imitation and reinforcement algorithms provided by ML-Agents, we are currently integrating in our environment neural network training methods based on evolutionary strategies and genetic algorithms in charge of evolving the weights and parameters of the multiple dense layers of the network. With this, we seek to include in our platform another example of a training strategy that naturally benefits from a high degree of parallelisation, and to evaluate the degree of complexity of this approach to more demanding projects and tasks. In this case, the intention is to propose a collection of interfaces through which the user can on the one hand enter their fitness functions of the individuals of the population in Unity, and on the other hand specify the crossover and mutation procedures available in the new population generation phases. We are also considering the possibility of integrating executors and training mechanisms for other classical models not based on neural networks (Random Forest, K-NN, SVM), as mentioned in the previous section.

This paper presents a highly valuable system, suitable for both research and educational environments, where various methods of controlling agents in video games can be created and compared. Drawing on both state-of-the-art classical control systems in the video game industry and increasingly demanded ML-based solutions, this all-in-one tool allows us to hybridize ML behaviors into other models that are more controllable by designers, facilitating comparisons between different approaches to solving a task within the same environment.

Here, we have described some of the applications and test cases we have been working on or are currently integrating into the environment. From this point, we intend to continue expanding the platform with new control paradigms both for classical AI

and machine learning, conducting experiments and developing environments to verify the tool's potential to compare different techniques in games developed on commercial engines whether it be for traditional applications such as NPC control or for emerging strategies in quality control operations and automated testing.

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)

[2] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016) https://doi.org/10.1038/nature16961 . Accessed 2024-01-05

[3] Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., *et al.*: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)

[4] Nichol, A., Pfau, V., Hesse, C., Klimov, O., Schulman, J.: Gotta learn fast: A new benchmark for generalization in rl. arXiv preprint arXiv:1804.03720 (2018)

[5] Perez-Liebana, D., Liu, J., Khalifa, A., Gaina, R.D., Togelius, J., Lucas, S.M.: General video game ai: A multitrack framework for evaluating agents, games, and content generation algorithms. IEEE Transactions on Games **11**(3), 195–214 (2019)

[6] Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., Coulom, R., Sumner, A.: Torcs, the open racing car simulator. Software available at http://torcs. sourceforge. net **4**(6), 2 (2000)

[7] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al.: Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680 (2019)

[8] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI gym. arXiv prepr int arXiv:1606.01540 (2016)

[9] Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D.: Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627 (2020)

[10] Unity-Technologies.: Unity Real-Time Development Platform | 3D, 2D VR & AR

Engine (2022). https://unity.com/ Accessed 2022-06-03

[11] Almeida, P., Carvalho, V., Simões, A.: Reinforcement learning as an approach to train multiplayer first-person shooter game agents. Technologies **12**(3), 34 (2024)

[12] Majumder, A., Majumder, A.: Case studies in ml agents. Deep Reinforcement Learning in Unity: With Unity ML Toolkit, 513–552 (2021)

[13] Juliani, A., Khalifa, A., Berges, V.-P., Harper, J., Teng, E., Henry, H., Crespi, A., Togelius, J., Lange, D.: Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning (2019)

[14] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (2017)

[15] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor (2018)

[16] Bain, M., Sammut, C.: A framework for behavioural cloning. In: Machine Intelligence 15, pp. 103–129 (1995)

[17] Ho, J., Ermon, S.: Generative Adversarial Imitation Learning. arXiv. arXiv:1606.03476 [cs] (2016). https://doi.org/10.48550/arXiv.1606.03476 . http://arxiv.org/abs/1606.03476 Accessed 2024-02-15

[18] Burda, Y., Edwards, H., Storkey, A., Klimov, O.: Exploration by Random Network Distillation (2018)

[19] Chauhan, V.K., Zhou, J., Lu, P., Molaei, S., Clifton, D.A.: A Brief Review of Hypernetworks in Deep Learning (2023)

[20] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need (2023)

[21] Isla, D.: Handling Complexity in the Halo 2 AI. https://www.gamedeveloper.com/programming/gdc-2005-proceeding-handling-complexity-in-the-i-halo-2-i-ai. GDC 2005 Proceeding, Accessed: 2024-03-18 (2005)

[22] Gutiérrez-Sánchez, P., Gómez-Martín, M.A., González-Calero, P.A., Gómez-Martín, P.P.: Reinforcement learning with temporal logic specifications for regression testing npcs in video games. In: 2023 IEEE Conference on Games (CoG). In Print (2023)

[23] Ariyurek, S., Betin-Can, A., Surer, E.: Automated Video Game Testing Using Synthetic and Humanlike Agents. IEEE Transactions on Games **13**(1), 50–67 (2021) https://doi.org/10.1109/TG.2019.2947597 . Accessed 2024-01-05

[24] Bergdahl, J., Gordillo, C., Tollmar, K., Gisslén, L.: Augmenting Automated Game Testing with Deep Reinforcement Learning. In: 2020 IEEE Conference on Games, pp. 600–603 (2020). https://doi.org/10.1109/CoG47356.2020.9231552 . ISSN: 2325-4289. https://ieeexplore.ieee.org/document/9231552 Accessed 2024-01-05

[25] Holmgård, C., Green, M.C., Liapis, A., Togelius, J.: Automated Playtesting With Procedural Personas Through MCTS With Evolved Heuristics. IEEE Transactions on Games **11**(4), 352–362 (2019) https://doi.org/10.1109/TG.2018.2808198 . Accessed 2024-01-05

[26] Sestini, A., Gisslén, L., Bergdahl, J., Tollmar, K., Bagdanov, A.D.: Automated Gameplay Testing and Validation with Curiosity-Conditioned Proximal Trajectories. IEEE Transactions on Games, 1–14 (2022) https://doi.org/10.1109/TG.2022.3226910 . Accessed 2024-01-05

[27] Gillberg, J.: Ai for testing: The development of bots that play battlefield v. In: Game Developers Conference (2019)

[28] Ahlberg, W., Sestini, A., Tollmar, K., Gisslén, L.: Generating Personas for Games with Multimodal Adversarial Imitation Learning. arXiv. arXiv:2308.07598 [cs] (2023). https://doi.org/10.48550/arXiv.2308.07598 . http://arxiv.org/abs/2308.07598 Accessed 2024-01-05

[29] Gordillo, C., Bergdahl, J., Tollmar, K., Gisslén, L.: Improving Playtesting Coverage via Curiosity Driven Reinforcement Learning Agents. arXiv. arXiv:2103.13798 [cs] (2021). https://doi.org/10.48550/arXiv.2103.13798 . http://arxiv.org/abs/2103.13798 Accessed 2024-01-05

[30] Gudmundsson, S.F., Eisen, P., Poromaa, E., Nodet, A., Purmonen, S., Kozakowski: Human-Like Playtesting with Deep Learning. In: 2018 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1–8 (2018). https://doi.org/10.1109/CIG.2018.8490442 . ISSN: 2325-4289. https://ieeexplore.ieee.org/document/8490442 Accessed 2024-01-05

[31] Gutiérrez-Sánchez, P., Gómez-Martín, M.A., González-Calero, P.A., Gómez-Martín, P.P., Lara-Cabrera, R., Leiva, A.: Liquid snake: a test environment for video game testing agents. In: CEV (2022)

[32] Sagredo-Olivenza, I., Gómez-Martín, M.A., González-Calero, P.A.: Supporting the collaboration between programmers and designers building game ai. In: Entertainment Computing-ICEC 2015: 14th International Conference, ICEC 2015, Trondheim, Norway, September 29-Ocotober 2, 2015, Proceedings 14, pp. 496–501 (2015). Springer

[33] Sagredo-Olivenza, I., Gómez-Martín, P.P., Gómez-Martín, M.A., González-Calero, P.A.: Trained behavior trees: Programming by demonstration to support ai game designers. IEEE Transactions on Games **11**(1), 5–14 (2017)

Chapter 8

Reinforcement Learning with Temporal Logic Specifications for Regression Testing NPCs in Video Games

Reinforcement Learning with Temporal Logic Specifications for Regression Testing NPCs in Video Games

1st Pablo Gutiérrez-Sánchez
Software Engineering and Artificial Intelligence Department
Complutense University of Madrid
Madrid, Spain
https://orcid.org/0000-0002-6702-5726

2nd Marco A. Gómez-Martín
Software Engineering and Artificial Intelligence Department
Complutense University of Madrid
Madrid, Spain
https://orcid.org/0000-0002-5186-1164

3rd Pedro A. González-Calero
Software Engineering and Artificial Intelligence Department
Complutense University of Madrid
Madrid, Spain
https://orcid.org/0000-0002-9151-5573

4th Pedro P. Gómez-Martín
Software Engineering and Artificial Intelligence Department
Complutense University of Madrid
Madrid, Spain
https://orcid.org/0000-0002-3855-7344

2023 IEEE Conference on Games (CoG) | 979-8-3503-2277-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/CoG57401.2023.10333208

***Abstract*—Reinforcement learning (RL) is a promising strategy for the development of autonomous agents in various control and optimization contexts, including the generation of autonomous players in video games. However, designing these agents, and in particular their reward functions to perform sequential decision-making, can be challenging for most users and often require tedious trial-and-error processes until a satisfactory result is obtained. Consequently, these strategies are generally beyond reach for designers and quality control teams, who could potentially make use of them to generate automatic testing agents. This paper presents the application of reinforcement learning and behavioral descriptions given through a formal temporal logic task specification language (TLTL) for the design of NPCs that can be employed as surrogates for the player in such contexts. We argue that these techniques enable designers to naturally specify the way in which they would expect the final player to interact with a level and then generate a test that automatically verifies whether this strategy continues to be feasible throughout the development of the game. We include a series of experiments conducted on a custom 3D test environment developed in Unity3D that show that the proposed methodology provides a simple mechanism for training NPCs in settings that are commonly encountered in modern video games.**

***Index Terms*—automated game testing, game-playing AI, reinforcement learning, temporal logics, regression testing**

I. Introduction

This paper will focus mainly on the problem of automatic generation in video games of non-player characters (NPCs) that are able to follow a sequential specification of tasks. This is especially relevant within the scope of the testing and QA process, where typically a group of human testers have to replay the level walkthroughs to find bugs that they report to the programmers for correction. These walkthroughs usually follow a logic that tries to foresee possible ways for the player to solve a level, which in our approach we represent as a set of steps ("Move to a platform, pick up an object, from there advance to the exit, all while avoiding being killed by an enemy attack. Check that it is possible to perform this sequence of tasks while losing less than half of the player's health points"). Our goal is to develop NPCs that are able to learn to follow these steps and test the game's levels, even when those levels or their logic undergo changes, as usually happens during the development of a video game.

When the game content is static or deterministic, it is possible to use pre-recorded traces of such action sequences being performed by a human tester and reproduce them periodically to check that the results continue to be as expected. However, these execution traces are generally very sensitive to changes in the environment and are not appropriate for verifying properties on spaces with random elements or on which modifications of any kind are applied (for example, slightly altering the position of a platform could lead to an error being reported by an execution trace even though it may not significantly influence level playability). These changes are not restricted to the environment but they may be also related to the NPCs themselves. As an example, every NPC of a given breed will have the same properties (speed, strength...) and behavior. During development it is common to fine-tune those behaviors in order to adjust the gameplay of a particular level, hardening or making others easier inadvertently. This is why it seems essential to design robust and reactive mechanisms that are able to adapt to more or less subtle changes in the environment when running regression tests.

To this end, several recent research papers have proposed

This work was supported by the Ministry of Science and Innovation (PID2021-123368OB-I00).

the use of different strategies to generate automatic players capable of performing various quality control tests, often based on artificial intelligence methods such as reinforcement or imitation learning. These mechanisms, while often suffering from overfitting problems to their training environments, seem promising for automatic testing due to their ability to accommodate dynamic situations and constantly changing environments.

In this work we focus on presenting a new approach to the development of agents that act as substitutes for human players in testing, using reinforcement learning based on test descriptions given in a formal task specification language (TLTL). To this end, and in view of being able to develop tools in the future that facilitate the proposed workflow to independent development teams, we developed a 3D experimentation environment in Unity3D with common elements from stealth games, making use of open source Machine Learning libraries that can be easily integrated with the engine, such as ML-Agents. It is important to note here, however, that while the long-term goal is to develop an automated testing framework, in this paper we will only focus on the generation of test agents from design specifications.

The rest of the paper runs as follows. Section II describes the proposed method to enable designers to specify possible solutions to levels using a list of operators; it also provides a description of our test-bed video game and the definition of different example specifications using the operators. Section III presents a high-level overview of the techniques involved in this work; we then proceed to present the experimental results and discuss our conclusions and future work.

II. Temporal Logic Specifications for Game Level Regression Testing

In this section we present the designer's view, what information and in what syntax the designer should provide us so that we can train an agent to impersonate the player and solve the level in the ways the designer has foreseen. To do so, we will start by describing Liquid Snake, the simplified infiltration game that we have developed as a test-bed, and then we will describe the specification language and concrete examples of its use.

A. Test-bed: Liquid Snake

Liquid Snake [1] is a first-person 3D stealth game prototype developed as a test-bed for this paper's experiments in the Unity3D engine [2]. In it, the player must control the main character through a battery of chambers in order to take them to the exit located in each of the levels, while avoiding being detected and shot down by enemies, and collecting as many collectible artifacts as possible along the way.

Enemies in this game follow classic stealth game behavior, whereby they patrol along their associated path in search for the player and, in the event of finding them, chase and shoot them until they are either defeated or out of sight. In the latter scenario, the enemies proceed to the last point where they spotted the player, search that area briefly, and return to their designated patrol path if they fail to find them. Meanwhile, the player enjoys several types of movement that allow them to navigate the game's various scenarios in strategic ways. Since the game features obstacles of varying heights and enemies that may detect the player based on the noise they make around them, the main character is equipped with the ability to run (high noise), walk (medium noise), crouch and lay down (low height, low noise). The player is also provided with a limited amount of ammunition that can be used to attack enemies from a distance, as well as the ability to perform a "stealth kill" when sneaking behind a target without being detected. An example map of the environments in question with designer annotations may be found in Fig. 1.

In both this and subsequent sections we shall make use of this environment as a means to illustrate the issues to be addressed in this work, as well as to test the techniques presented throughout the paper.

B. Context

When designers build a video game level, they usually do so with an idea in mind about how they would expect the player to solve it. As an example, for the level in Fig. 1 an expected resolution flow could be as follows:

1) The player starts in a room with two switches that must be activated in the correct order to unlock the associated door and allow access to the next section.
2) After solving this initial puzzle, the player unlocks a second section with two new doors, where they will need to visit an additional switch that will randomly decide which of the two doors should be opened.
3) If door A is opened, the player will be faced with a simple navigation problem where they must alternate between crouching and jumping segments in order to overcome areas of low ceilings and obstacles on the ground. At the end of this area, the player will be able to activate a final switch which unlocks the door that separates them from the goal.
4) If door B is opened, the player will enter a closed room in which an enemy guards the switch that unlocks the exit. The expected behavior in this case is to manage to flip the switch without being defeated by the enemy and then flee the room. At the end of the area, the player must visit the same final activator mentioned earlier to reach the goal.

In certain cases, the designer may come up with different ways to complete the level depending on the different target player profiles: continuing with the previous example, a collectionist type of player might be interested in returning to the originally locked door in the room at the beginning of the stage after flipping the switch at the end of the level to obtain the collectible object hidden behind it. It is important to note here that these specifications are expressed through a fairly high-level and nonspecific vocabulary, and could in principle be satisfied in a variety of ways; for instance, in the above specification it is irrelevant whether or not the player

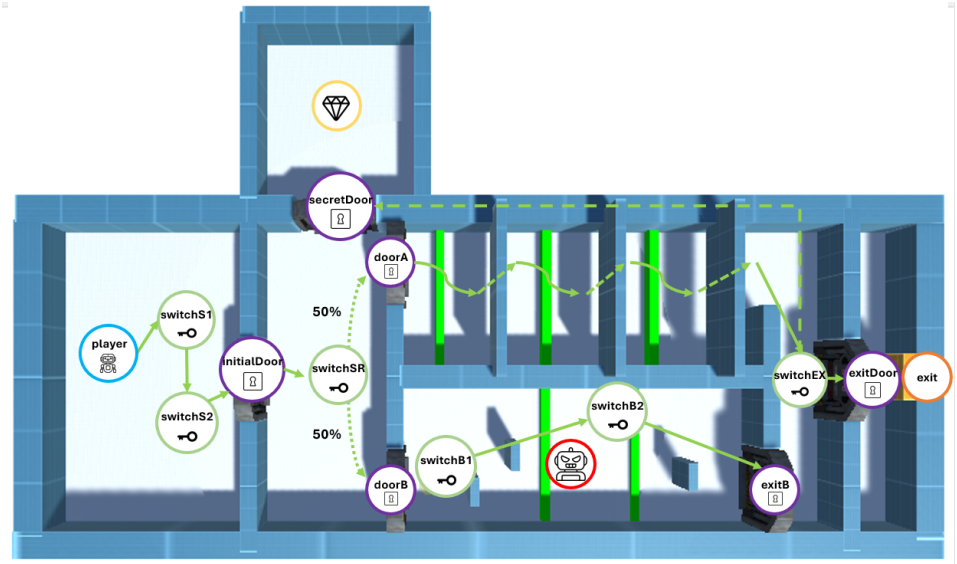


Fig. 1. Example level from *Liquid Snake* with designer annotations.

succeeds in defeating the level's enemy in region B, as long as they manage to activate the switch.

Once a plan as to how the level should be resolved is available, it can be used to establish a test within a quality assurance (QA) test suite whereby a tester can follow the steps described above to verify whether it is possible to complete the level in such a fashion. This test falls into the category of checks which should ideally be repeated fairly frequently to ensure that changes made during the development process do not "break" this specification and that it remains possible to meet the designer's requirements, commonly referred to as *regression testing*. However, these routine checks, while generally manageable at the beginning of development, grow exponentially in time commitment as new content is added to the project, to a point where it becomes infeasible to maintain complete coverage of all regression tests in the game.

This is where automatic testing methodologies come into play as a tool to streamline the process and promote greater product coverage and quality control. Ideally, from the designer's point of view, one would expect to be able to specify the pre-conceived level solution(s) in a standardized way, and from this specification generate a periodic report to establish whether this remains compliant at any given point in time.

C. Designer Operators

The methodology developed in this paper proposes a first workflow by which the designer is able to specify the resolution process of a level through the use of logical operators in natural language, from which an NPC is automatically generated to fulfill (if possible) the requirements established in the task description. We also consider those cases where the designer may wish to enforce some form of restriction on parts of the process, be it limits on level parameters ("the player's health points should not drop below 50%", "no ammunition should be used" or "player must be able to complete the level in less than 3 minutes") or temporal constraints ("switch B should never be visited before switch A"), thereby enabling a greater degree of control over the expected progression of the player through the environment.

We currently support the following high-level operators for the construction of specifications by a designer:

- `avoid` *condition*. Prevent given condition from being fulfilled.
- Operators `and`, `or`, `then`, `eventually`, `always`, `until` and `if/ do`.
- *actorPosition* `reach` *goalPosition* `withDistance` *d*. Get a given game actor to a (possibly mobile) object's transform as quickly as possible.
- *actorPosition* `sequentialReach` $goalPosition_{1...N}$ `withDistance` *d*. Shortcut to express the need to get a given actor to a series of sequential objectives (N `reach` clauses) with a shared proximity threshold.
  This can be replaced by `sequentialReachStrict` to incorporate the clauses

$$\begin{aligned} \mathtt{r}_i = \mathtt{and}\, \lfloor & \mathtt{avoid} \\ & \lfloor actorPosition\ \mathtt{reach}\ goalPosition_i \\ & \mathtt{withDistance}\ d \rceil\ \mathtt{until} \\ & \lfloor actorPosition\ \mathtt{reach}\ goalPosition_{i-1} \\ & \mathtt{withDistance}\ d \rceil \rceil \end{aligned} \tag{1}$$

  for each $i = 2 \ldots N$, which enforces a strict visit order for the specified targets (do not visit target i until you have visited previous target).

While this subset of operators may appear limiting at first glance, we note that they are sufficient to concisely define the

specification given above for the level in Fig. 1:

$$\begin{aligned}\mathtt{g}_A = &\lfloor p_{player}\ \mathtt{sequentialReachStrict}\\ &p_{switchS1},\ p_{switchS2},\ p_{switchSR}\ \mathtt{withDistance}\ 2\rceil\\ &\mathtt{then}\\ &\lfloor \mathtt{if}\ \textit{doorB}\ \text{opened}\ \mathtt{do}\ \lfloor\\ &\lfloor p_{player}\ \mathtt{reach}\ p_{switchB2}\ \mathtt{withDistance}\ 2\rceil\rceil\rceil\ \mathtt{then}\\ &\lfloor p_{player}\ \mathtt{sequentialReach}\\ &p_{switchEX},\ p_{exit}\ \mathtt{withDistance}\ 2\rceil\ \mathtt{and}\\ &\lfloor \mathtt{avoid}\ health_{player} < 50\%\rceil\ \mathtt{and}\\ &\lfloor \mathtt{avoid}\ ammo_{player} < 50\%\ \mathtt{until}\ \textit{doorB}\ \text{opened}\rceil,\end{aligned} \tag{2}$$

this being enough to train an automatic NPC using the strategies that will be described in subsequent sections. Behind the scenes, this process of automatic NPC generation is supported by methods belonging to the field of deep reinforcement learning, equipping them with a layer of abstraction that eliminates the need to be an expert in statistical learning in order to train a bot capable of satisfying the desired specification. This is relevant not only for designers, but also for practitioners of machine learning in video games, by mitigating the problem of tailoring often complex and obscure reward functions that implicitly define the agent's behavior (known as "reward hacking"), and instead having them automatically generated from understandable expressions based on domain knowledge of the expert. This latter point is important to emphasize, as the focus in this work does not lie in creating agents that learn tasks from scratch, but rather in translating a designer's preconceived plan of action into a training system that learns to follow it with minimal friction between the two ends.

III. Temporal Logics for Reinforcement Learning

Recently, and especially in the field of robotics, there has been a growing interest in the use of specifications given by temporal logics (TL) to define reward functions in RL algorithms [3]. Among these logics, the most studied in this context are those which allow the definition of quantitative semantics, also known as robustness [4], as they provide a numerical measure of the goodness of fit of a state trace to a specification. This robustness metric can be used as a natural reward function, enabling the specification of sequential tasks, sub-objectives, temporal limits and behavioral constraints.

One of the most widely used temporal logic languages is STL (Signal Temporal Logic) [5], which has shown good results in robot control tasks, but has the disadvantage that the reward generation process can only be applied on complete trajectories, leading to very sparse rewards and making it very difficult to learn long and/or complicated tasks. Alternatively, TLTL (Truncated Linear Temporal Logic) [6] is a temporal logic whose formulas can be evaluated over finite trajectories of arbitrary length. While in its early applications on RL, robustness evaluation was limited to a single reward evaluating each complete trace at the end of training episodes, later work introduced the idea of applying Finite State Predicate Automata (FSPAs) to absorb temporal dependencies and grant dense rewards based on robustness variations between transitions [7], thus providing a much more appropriate tool for learning complex tasks.

The work presented here is intended as an adapted application of the strategies developed by [8] (which in turn expands upon [7]) to task specification in video games. Here, we focus on performing a series of preliminary tests on stealth game environments with game mechanics commonly found in commercial games and evaluate the feasibility of using these methods in the context of NPC specification for automated testing.

A temporal logic (TL) is any system of rules and symbolisms for representing and reasoning about propositions qualified in terms of time, providing an interface for the specification of tasks linked to a temporal structure. Among the temporal logics, we will be interested in selecting one that has both qualitative (True or False predicates) and quantitative semantics (continuous measure of formula satisfaction), and in particular one that allows us to evaluate the goodness of fit to a specification of finite state trajectories of arbitrary length over a set of possible states S. TLTL [6] is a formal specification language that meets these conditions, providing a convenient mechanism for incorporating complex intentions, domain knowledge and constraints into a task specification.

In essence, a TLTL formula is defined over predicates of the form $f(s) < c$, where $f : \mathbb{R}^n \to \mathbb{R}$ is a function over the state of the system and c is a constant. A specification in TLTL has the following syntax:

$$\begin{aligned}\phi := \top \mid f(s) < c \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid\\ \Diamond\phi \mid \Box\phi \mid \phi\,\mathcal{U}\,\psi \mid \phi\,\mathcal{T}\,\psi \mid \bigcirc\phi \mid \phi \Rightarrow \psi,\end{aligned} \tag{3}$$

where $f(s) < c$ is a predicate, $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction) and $\Rightarrow$ (implication) are Boolean connectives, and $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), $\mathcal{T}$ (then), and $\bigcirc$ (next) are temporal operators. Using this notation, we can assign a TLTL predicate to the operators introduced in the previous section. Thus, the `reach` operator expresses $\Diamond\, dist(p_{actor}, p_{goal}) < d$, whereas the `avoid` operator acts as a shortcut to $\Box\neg$*condition*. The remaining operators behave as literal translations of the corresponding TLTL connectors.

As discussed above, TLTL comes equipped with both Boolean and quantitative semantics (also called degree of robustness), the latter being of great interest for evaluating the numerical extent to which a sequence of states conforms to a formal specification. In particular, if we call $s_t \in S$ the state at time t, and $s_{t:t+k}$ the sequence or trajectory of states between instants t and $t+k$, the degree of robustness of a trajectory with respect to a specification ϕ is given by a real-valued function $\rho(s_{t:t+k}, \phi)$, which denotes how far $s_{t:t+k}$ is from satisfying or violating ϕ. It holds that $\rho(s_{t:t+k}, \phi) > 0 \Rightarrow s_{t:t+k} \vDash \phi$ and $\rho(s_{t:t+k}, \phi) < 0 \Rightarrow s_{t:t+k} \nvDash \phi$, and thus the sign of the degree of robustness is representative of whether a trace conforms to the proposed specification. Although there are different approaches to define the robustness of a predicate in TLTL [9], [10], here we will opt for the one originally

presented by [6], despite it suffering in certain situations from non-smoothness, non-convexity and locality problems that result in the maximum robustness of the system being capped by the least robust sub-goal, due to its ease of implementation (with certain modifications to address these problems).

The degree of robustness of a trace based on a specification ϕ can be used in conjunction with reinforcement learning techniques, choosing as reward function the one that awards the agent with the robustness of the sequence of states visited during an episode at its completion [6]. This reward function is straightforward to interpret and specify due to its proximity to natural language, and does not require any reward-hacking process for its definition. However, this results in very sparse rewards, granted only at the end of an episode, which makes it difficult to train policies capable of learning complex tasks.

To adapt this strategy to more general situations, [7] and [8] propose the generation of finite state predicate automata (FSPAs) from predicates in TLTL that capture the sequences of motions or actions necessary to successfully satisfy the task specification. Instinctively, an FSPA can be defined as a finite state automaton in which the transitions are equipped with TLTL predicates, such that a transition is triggered if the robustness of its associated predicate is positive and maximal (if there are several edges that meet this criterion). Following the conversion process described by [11], given a formula ϕ over predicates on the set of states S, for any trace $\mathcal{T}_{t=0..t}$ satisfying ϕ it is possible to construct an FSPA that accepts $\mathcal{T}_{t=0..t}$. The transitions between states of these automata depend on the robustness of their edge predicates rather than on binary truth values, and consequently each transition is associated with a robustness gain that can be used as a step (rather than episodic) reward. Trap states are added to these FSPAs as a means to represent having failed at the task. Although there are different possibilities for choosing a reward allocation algorithm from the transitions of an FSPA, here we will rely on an adaptation of the one presented by [8].

IV. Experiments

In this section we present the process followed to evaluate the feasibility of the techniques presented above in the context of the generation of agents capable of fulfilling specifications given by TLTL predicates on a given level.

A. Early Tests

Initially, we performed a preliminary implementation of the dense reward assignment algorithm described in [8], which starts from a TLTL predicate and then generates an FSMA that abstracts its temporal dependencies in the form of automaton transitions. We use [12] to perform the conversion from predicates to automata, after transforming the expressions given by abstract operators to TLTL. This FSMA is given the task of granting a reward at each simulation step based on the maximum robustness of the predicates associated with outgoing transitions. A first modification to the original algorithm is that here we only consider the robustness of transitions leading to non-trap states when granting rewards, as otherwise one could

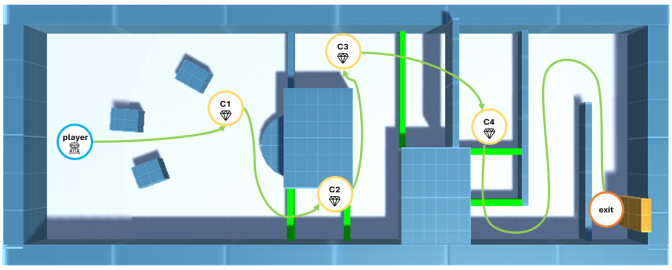


Fig. 2. Basic navigation level with expected path.

reach situations in which the agent ends up favoring trap states by receiving a reward for approaching their edge conditions.

It could be argued, however, that transitions to a trap state may occur indirectly, i.e., if found in an intermediate state A from which a branch of consecutive states $A \implies B \implies C$ starts, where C is a trap state, one could grant rewards for the transition $A \implies B$, and be unable to grant rewards in state B due to the absence of any outgoing edge to a good state. This situation can be solved by combining the last two states into a single trap state BC, and obviating the original predicate between B and C: indeed, in the original configuration, if we have entered state B we already know that the result cannot lead to a success, so in practice there is no value in it and we can immediately consider it as a negative state.

The implementation used in this work involves the use of a single neural network that receives as its base input a set of observations from the environment. These are given by the relative position of the agent in the level and its current velocity (x, z), its angular orientation, spatial information from raycast sensors, and the proportion of health and ammo remaining at the present time. An additional vector representing the current state of the FSMA is added to this in one-hot-encoding, so that the agent may access the data on where it is in the flow at any given time. Agents were trained using the ML-Agents implementation of the PPO algorithm.

Once this preliminary implementation was completed, we went on to test it with a first simple and deterministic environment in which the agent would only be forced to navigate the level by sequentially gathering a series of collectible objects. A representative image of the level can be found in Fig. 2. The specification to be tested in this case is given by the following expression (where c_i denotes the i-th collectible in the level):

$$\begin{aligned} \mathtt{g}_B : \lfloor p_{player}\ \mathtt{sequentialReach} \\ p_{c_1},\, p_{c_2},\, p_{c_3},\, p_{c_4},\, p_{exit}\ \mathtt{withDistance}\ 2 \rceil \end{aligned} \tag{4}$$

leading to a FSMA with 6 states, one for each `reach` clause and a goal state.

After a first training test under these conditions, however, a number of practical problems arise. First, there is the issue of motivating the agent to transition from one state to the next in the sequence. More specifically, given any state A and a transition $A \implies B$, with a reward function r, as the agent learns to optimize $r(A)$ the motivation to switch to the next state decreases, as it moves from a state where it has learned

TABLE I
TIME TAKEN TO FIRST START EACH MAJOR SUB-TASK IN NAVIGATION LEVEL (FROM START OF TRAINING).

Agent	Reach c_2	Reach c_3	Reach c_4	Reach $exit$	*Goal State*
Single Network	52 mins	54 mins	1 h 24 mins	1 h 47 mins	7 h 48 mins
Multiple Networks	7 mins	18 mins	31 mins	57 mins	1 h 31 mins

to secure a safe and constant reward to a state where this is no longer the case ($r(B) \ll r(A)$). In this particular environment, this is manifested by the agent eventually learning to stay at the edge of the transition threshold where the penalty is minimal rather than picking up the object. [8] experimentally proposes to apply a steeper gradient on the robustness in the target region of a predicate, but in our experiments this strategy had unsatisfactory results. We finally opted to grant terminal rewards upon completion of each sub-target; that is, to grant a reward between transitions equal to the number of steps remaining until the length of the episode is reached. This, despite acting against the interpretability of the cumulative reward, ensures that it is always better to make a transition than to stay in a state while minimizing a penalty.

The second problem concerns the effects of using the conjunction operator $\rho(\phi \wedge \psi) = \min(\rho(\phi), \rho(\phi))$, as proposed in [8]. If we make use of this operator, we are constraining the robustness of a predicate by that of its weakest sub-predicate, so that if there are two tasks, ϕ_A and ϕ_B, $\rho(\phi_A \wedge \phi_B)$ will always be the minimum between the individual robustnesses regardless of whether the other is being successfully satisfied. In practice this translates into an increased difficulty in learning the assignments: if it is not possible to improve on all of them simultaneously, it is never possible to perceive a compensation for the actions corresponding to individual gains. To address this problem we propose the following alternative conjunction operator: $\rho(\phi_1 \wedge \ldots \wedge \phi_N) = \sum_{i=1}^{N} \min(\phi_i, 0)$, if any $\rho(\phi_i) < 0$, and $\sum_{i=1}^{N} \phi_i$ otherwise. This is always negative as long as there is at least one sub-predicate with negative robustness (essential to maintain the coherence of the quantitative semantics), and approaches 0 as the sub-objectives of the predicate become more robust.

After these modifications, we went on to repeat the experiments in the navigation environment with acceptable results, which can be found in Table I. Note that the initial sub-task of the sequence is omitted as it is always visited first. An observation in view of the results obtained is that the time required to complete the last task increases markedly, with no significant increase in difficulty at first glance. This led us to hypothesize that by forcing a single network to learn all the sub-tasks with only the one-hot-encoding of the current state index as a way to differentiate between steps, it may prove more complicated to deal with situations in which the inputs are fairly similar but the task or motion to be performed is fundamentally different. Another concern that arises in the process is how well this type of configuration handles a dynamic task with conditional flows such as the one we used as an example in the problem statement section.

B. Further Experiments

To improve performance in these aspects we chose to implement a variation of this strategy in which instead of a single neural network to encode the entire policy we assign a network and policy for each state of the FSMA. This could facilitate the scalability of the method to arbitrarily complex tasks at the cost of potentially losing inter-state knowledge transfer (each state must be relearned entirely). Another question is whether the use of multiple networks accelerates learning or slows it down, taking into account that by using several networks it is possible to modify both the inputs and the configuration of the network structure as well as the learning algorithm in each sub-task (allowing, for instance, to assign simpler networks with less information to easier tasks).

With this, we re-run the experiment in the navigation environment with the multi-network approach, with results that can be found in Table I and that clearly improve on those obtained using a single network. It is important to note that here we reduce the volume of information provided by raycasts in the simpler sub-tasks (for instance, we omit height sensors in sub-tasks that do not require crouching to pass under obstacles), in order to reduce the complexity of each sub-problem. The specification used at this level is however very simple and it is thus essential to confirm whether these patterns are replicable in more complex environments with dynamic flows. We thus consider applying the two strategies described above (with one and several networks) to the case of the level originally presented in Fig. 1, using the specification given in (2). This specification is translated into a FSMA consisting of 8 states: 6 for the `reach` clauses, plus goal and trap states.

After training both models using the same FSMA, we obtained the results presented in Table II. Looking at the recorded times, it would appear that although both models start solving the first section in a similar time span (which is to be expected as there are no major differences between the networks used in this case), the model using a single neural network gradually takes shorter and shorter times than the one using a multi-agent network, reaching the exit from both branching paths in more favorable times than the multi-agent one. This phenomenon can be explained by describing the patterns shown during training with each approach in a qualitative way. In this setting, the agent with a single network preserves knowledge of previous steps when advancing to the next state, so that it "knows" that it must stand on a switch to operate it, and is able to transfer the ability to jump over obstacles between branches, whereas the agent with multiple networks effectively loses all knowledge between sub-tasks. Additionally, it should be noted that the initial situation in

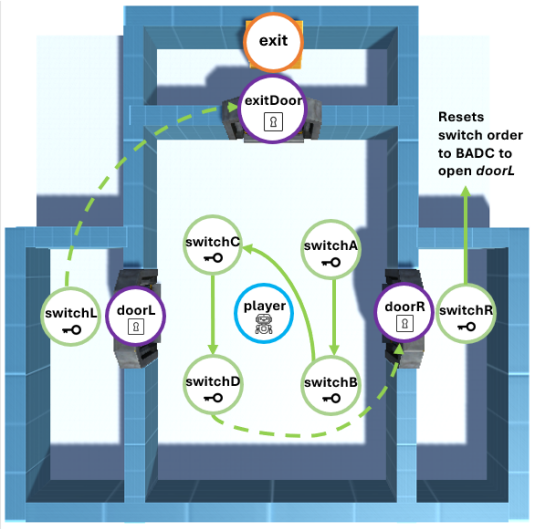


Fig. 3. Locality level with designer notes.

which each problem must be solved differs across schemes: for instance, in the problem of evading the enemy to escape from room B, the multi-network model must learn to simultaneously dodge and navigate from scratch in two different states.

Lastly, to evaluate the performance of both models in specifications in which the agent is forced to undertake different tasks in the same environment, we designed a third level, which is shown in Fig. 3. The expected behavior is to visit the switches in the strict order $ABCD$, unlocking the door on the right, after which the switch locked behind it must be activated to reset the sequence in the central square. This is repeated with the left door, visiting the switches in the order $BADC$. Activating the switch behind the left door unlocks the exit. This forces the agent to make use of the state information in order to determine the correct visit order (as any other order resets the sequence). The associated specification is given by:

$$\begin{aligned} \mathtt{g}_C = & \lfloor p_{player}\ \mathtt{sequentialReachStrict} \\ & p_{switchA},\ p_{switchB},\ p_{switchC},\ p_{switchD},\ p_{switchR} \\ & \mathtt{withDistance}\ 2 \rceil\ \mathtt{then} \\ & \lfloor p_{player}\ \mathtt{sequentialReachStrict} \\ & p_{switchB},\ p_{switchA},\ p_{switchD},\ p_{switchC},\ p_{switchL}, \\ & p_{exit}\ \mathtt{withDistance}\ 2 \rceil, \end{aligned} \tag{5}$$

generating a FSMA with 11 states, plus goal and trap states. The training results for each model can be found in Table III. As can be appreciated, our hypothesis does not seem to hold true at least in this situation: in this case, both models end up exhibiting a fairly similar training process, with the multi-network model taking slightly longer in the last sub-task, albeit not significantly different from that of the unitary network.

V. Related Work and Conclusions

Within the video game industry, there exists a set of tasks usually framed within the scope of quality assurance that demand great efforts to verify the design and functioning of the features included in the final product. As the size of the game grows, so does the number of tests to be repeated, and with it the human time required to carry out these checks. These tests are mainly presented as sequentially structured tasks in which a number of steps to be performed on a given section of the game and an expected result after completion are defined.

In order to automate these tests, a number of strategies have been proposed based largely on the generation of agents that act as surrogates for a human player and are capable of acting like them. One of the most straightforward alternatives is simply to make use of segments recorded manually by a human performing the specified tasks, replaying them every so often over the original environment to check that the player's trace is still able to complete the set objective [13]. However, when the structure of the environment is modified, or the environment includes random elements that do not remain constant between runs, these strategies are no longer valid, motivating the need to create agents with a certain capacity to adapt to changes in their surroundings.

This is where methods based on AI-based strategies come into play, offering more reactive and adaptive policies to changing environments. Some examples of these techniques for automatic testing in video games adopt the use of machine learning algorithms based on Deep Reinforcement Learning (DRL) [14], [15], Imitation Learning (IL) [16], or even hybrid models of the previous strategies with control structures such as behavior trees (BTs) [17]. Once an adequate and reactive model is available, it is possible to collect statistics concerning how the agent perceives its passage through the environment (time taken to complete the task, remaining health, score, etc) and perform statistical analysis between sets of metrics collected across simulations run on different days to determine if significant changes have occurred in the test [17]. A common situation in these papers is that these methods, while promising, often require a great amount of technical knowledge to implement and sometimes a tedious process of trial and error in defining the reward functions in the case of DRL (often referred to as "reward hacking").

Additionally, a good part of these works focus on solving tasks in a "global" way, but do not contemplate the case that concerns us in this work, by which we intend to enforce the fulfillment of a series of ordered sub-objectives within the task specification, which is closer to the structure of a QA test.

To ensure a sufficiently thorough and regular coverage of a video game from the point of view of regression testing in QA, it appears essential to devise automatic test generation strategies that are sufficiently accessible and easy to use for industry practitioners. In this regard, [18] proposes the use of procedural personas derived from utility functions defined around different play styles to perform automatic game testing, but still requires a manually specified reward function and is only applied to discrete 2D environments. [19] on the other hand proposes a DRL algorithm supported by curiosity and imitation based techniques to detect unexpected ways to reach the level objectives, while requiring a complex setup and a

TABLE II
TIME TAKEN TO FIRST START EACH MAJOR SUB-TASK IN BRANCHING LEVEL (FROM START OF TRAINING).

Agent	Flip $switchSR$	Flip $switchEX$ from A	Flip $SB2$	Flip $switchEX$ from B	*Goal State* from A	*Goal State* from B
Single Network	2 h 37 mins	2 h 53 mins	2 h 51 mins	3 h 5 mins	4 h 29 mins	5 h 47 mins
Multiple Networks	2 h 7 mins	3 h 33 mins	3 h 34 mins	9 h 27 mins	6 h 33 mins	11 h 58 mins

TABLE III
TIME TAKEN TO FIRST START EACH SUB-TASK IN LOCALITY LEVEL (FROM START OF TRAINING).

Agent	Reach $switchR$	Unlock $doorL$	Reach $switchL$	Reach $exit$	*Goal State*
Single Network	1 h 27 mins	1 h 58 mins	3 h 26 mins	3 h 44 mins	5 h 1 mins
Multiple Networks	1 h 37 mins	2 h 7 mins	3 h 8 mins	3 h 53 mins	6 h 5 mins

fair number of human demonstrations.

Here we present an implementation and application of emerging techniques in the field of control systems and robotics to the domain of behavior generation in video games to translate quasi-natural language specifications to FSPAs. These in turn abstract the temporal dependencies between sub-tasks within complex flows and grant dense training rewards based on robustness variations within the current sub-task. With this methodology a designer with no prior knowledge in machine learning can focus on scripting the way in which they would expect a user to complete a given corresponding level, and produce an NPC that aims to satisfy this specification.

To address some of the limitations of the model employing a single neural network to encode the agent's policy, we also present an alternative in which a new network is assigned for each of the FSPA states generated from the specification. After evaluating the performance of both models in three levels designed for this purpose in a prototype 3D stealth game, we can verify that both exhibit favorable properties, although future work remains to analyze which factors affect each of the strategies to a greater or lesser extent in order to determine in which situations it is more appropriate to use one or the other.

REFERENCES

[1] P. Gutiérrez-Sánchez, M. A. Gómez-Martín, P. A. González-Calero, and P. P. Gómez-Martín, "Liquid snake: a test environment for video game testing agents," in *Actas del I Congreso Español de Videojuegos, Madrid, Spain, December 1-2, 2022*, ser. CEUR Workshop Proceedings, R. Lara-Cabrera and A. J. F. Leiva, Eds., vol. 3305. CEUR-WS.org, 2022. [Online]. Available: https://ceur-ws.org/Vol-3305/paper7.pdf

[2] Unity-Technologies., "Unity Real-Time Development Platform | 3D, 2D VR & AR Engine," 2022. [Online]. Available: https://unity.com/

[3] H.-C. Liao, "A Survey of Reinforcement Learning with Temporal Logic Rewards," 2020.

[4] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model Predictive Control for Signal Temporal Logic Specification," *arXiv:1703.09563 [cs]*, Mar. 2017, arXiv: 1703.09563. [Online]. Available: http://arxiv.org/abs/1703.09563

[5] O. Maler and D. Nickovic, "Monitoring Temporal Properties of Continuous Signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds. Berlin, Heidelberg: Springer, 2004, pp. 152–166.

[6] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC: IEEE, Sep. 2017, pp. 3834–3839. [Online]. Available: http://ieeexplore.ieee.org/document/8206234/

[7] X. Li, Z. Serlin, G. Yang, and C. Belta, "A formal methods approach to interpretable reinforcement learning for robotic planning," *Science Robotics*, vol. 4, no. 37, p. eaay6276, Dec. 2019. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.aay6276

[8] X. Zhao and M. Campos, "Reinforcement Learning Agent Training with Goals for Real World Tasks," *arXiv:2107.10390 [cs]*, Jul. 2021, arXiv: 2107.10390. [Online]. Available: http://arxiv.org/abs/2107.10390

[9] Y. Gilpin, V. Kurtz, and H. Lin, "A Smooth Robustness Measure of Signal Temporal Logic for Symbolic Control," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 241–246, Jan. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9114883/

[10] N. Mehdipour, C.-I. Vasile, and C. Belta, "Specifying User Preferences Using Weighted Signal Temporal Logic," *IEEE Control Systems Letters*, vol. 5, no. 6, pp. 2006–2011, Dec. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9309020/

[11] K. Y. Rozier, "Explicit or Symbolic Translation of Linear Temporal Logic to Automata," Thesis, Rice University, Jul. 2013. [Online]. Available: https://scholarship.rice.edu/handle/1911/71687

[12] "Spot / Spot · GitLab." [Online]. Available: https://gitlab.lre.epita.fr/spot/spot

[13] M. Ostrowski and S. Aroudj, "Automated Regression Testing within Video Game Development," *GSTF Journal on Computing (JoC)*, vol. 3, no. 2, p. 10, Jul. 2013. [Online]. Available: https://doi.org/10.7603/s40601-013-0010-4

[14] J. Bergdahl, C. Gordillo, K. Tollmar, and L. Gisslen, "Augmenting Automated Game Testing with Deep Reinforcement Learning," in *2020 IEEE Conference on Games (CoG)*. Osaka, Japan: IEEE, Aug. 2020, pp. 600–603. [Online]. Available: https://ieeexplore.ieee.org/document/9231552/

[15] J. Pfau, J. D. Smeddinck, and R. Malaka, "Automated Game Testing with ICARUS: Intelligent Completion of Adventure Riddles via Unsupervised Solving," in *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play*. Amsterdam The Netherlands: ACM, Oct. 2017, pp. 153–164. [Online]. Available: https://dl.acm.org/doi/10.1145/3130859.3131439

[16] S. Ariyurek, A. Betin-Can, and E. Surer, "Automated Video Game Testing Using Synthetic and Humanlike Agents," *IEEE Transactions on Games*, vol. 13, no. 1, pp. 50–67, Mar. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8869824/

[17] P. Gutiérrez-Sánchez, M. A. Gómez-Martín, P. A. González-Calero, and P. P. Gómez-Martín, "Reinforcement Learning Methods to Evaluate the Impact of AI Changes in Game Design," *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 17, no. 1, pp. 10–17, Oct. 2021. [Online]. Available: https://ojs.aaai.org/index.php/AIIDE/article/view/18885

[18] C. Holmgard, M. C. Green, A. Liapis, and J. Togelius, "Automated Playtesting With Procedural Personas Through MCTS With Evolved Heuristics," *IEEE Transactions on Games*, vol. 11, no. 4, pp. 352–362, Dec. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8295256/

[19] A. Sestini, L. Gisslen, J. Bergdahl, K. Tollmar, and A. D. Bagdanov, "Automated Gameplay Testing and Validation with Curiosity-Conditioned Proximal Trajectories," *IEEE Transactions on Games*, pp. 1–14, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9970382/

Chapter 9

A Progress-Based Algorithm for Interpretable Reinforcement Learning in Regression Testing

A Progress-Based Algorithm for Interpretable Reinforcement Learning in Regression Testing

Pablo Gutiérrez-Sánchez, Marco A. Gómez-Martín, Pedro A. González-Calero, and Pedro P. Gómez-Martín

***Abstract*—In video games, the validation of design specifications throughout the development process poses a major challenge as the project grows in complexity and scale and purely manual testing becomes very costly. This article proposes a new approach to design validation regression testing based on a reinforcement learning technique guided by tasks expressed in a formal logic specification language (truncated linear temporal logic) and the progress made in completing these tasks. This requires no prior knowledge of machine learning to train testing bots, is naturally interpretable and debuggable, and produces dense reward functions without the need for reward shaping. We investigate the validity of our strategy by comparing it to an imitation baseline in experiments organized around three use cases of typical scenarios in commercial video games on a 3-D stealth testing environment created in unity. For each scenario, we analyze the agents' reactivity to modifications in common assets to accommodate design needs in other sections of the game, and their ability to report unexpected gameplay variations. Our experiments demonstrate the practicality of our approach for training bots to conduct automated regression testing in complex video game settings.**

***Index Terms*—Automated game testing, game-playing AI, regression testing, reinforcement learning (RL), temporal logics (TLs).**

I. Introduction

VIDEO games are increasingly complex and ambitious software products in terms of extension and the number of elements or assets involved in their creation. In this context, there are a number of quality control tasks related to gameplay validation, i.e., ensuring that the way players interact with the game remains consistent with the designer's preconceived idea of how each challenge should be tackled. Play-testing is a common method of assessing game quality, usually requiring a human tester to answer questions about the degree of difficulty and perceived enjoyment during play, the presence of bugs and glitches, or the feasibility of successfully completing various tasks enforced by the design. The problem with this approach to quality testing is that as levels, shared assets between game scenes, nonplayable character artificial intelligence (AI)s, or functionalities are created or modified throughout development, the answers to these questions may change unexpectedly in an increasing number of scenarios. For instance, modifying the configuration of a shared enemy AI could alter the difficulty of a level, prevent it from being solved using the originally designed strategy, or even enable unexpected new solutions.

In order to maintain a sufficiently thorough coverage of the range of scenarios and checks to be performed within the game, design validations should be performed as frequently as possible, which is not logistically feasible if these tasks are reserved for human testers since they require an unsustainable effort in terms of time and money. By design validations, we will narrow down here to the type of test that receives a specification, typically given by a series of steps or instructions to be performed sequentially by the tester, and reports whether these are feasible in the environment in which they are proposed, together with comments on what makes them invalid in the case where they are deemed nonviable. Having an automated specification validation methodology in the development process does not replace the human tester but allows redirecting their efforts to verify only those sections of the project where it is truly suspected that unexpected changes have occurred, thus increasing efficiency and coverage in QA.

In recent years, different automated playtesting methodologies have been proposed to try to alleviate these incremental costs. In the industry, this is usually done primarily through hand-programmed bots that perform playtesting automatically [1], [2], [3], [4], which is typically cumbersome and experiences issues in adapting to changes in the original environment, often requiring a code rework by a human programmer. Another strategy with increasing traction is the generation of testing agents by means of machine learning methods, such as reinforcement learning (RL) [5], [6], [7], [8], which do not require explicit scripting to learn to play and can be retrained upon level adjustments, thereby alleviating the generalization problem. RL, however, is often sample inefficient and its success depends heavily on the choice of reward function, which often calls for extensive technical expertise, and its integration into a commercial development environment is not straightforward.

In this article, we iterate on our previous work in the field of automated testing in video games, proposing a formal specification-based approach to create agents that are able to perform design validations on a scheduled basis throughout the game development process, as shown in Fig. 1. Our method eliminates the need to specify a manual reward function in the RL loop by automatically constructing a dense function from a staged representation of what is expected to be validated in the

Manuscript received 1 November 2023; revised 3 April 2024 and 20 May 2024; accepted 4 July 2024. Date of publication 11 July 2024; date of current version 17 December 2024. This work was supported by the Spanish Ministry of Science and Innovation under Grant PID2021-123368OB-I00. Recommended by Associate Editor L. Gisslen. *(Corresponding author: Pablo Gutiérrez-Sánchez.)*

The authors are with the Software Engineering, Artificial Intelligence Department, Complutense University of Madrid, 28040 Madrid, Spain.

Digital Object Identifier 10.1109/TG.2024.3426601

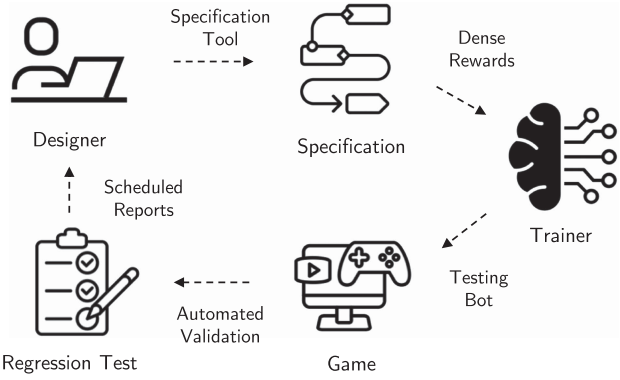


Fig. 1. Our approach allows the designer to provide a plan on how to interact with the level, from which a dense reward function is generated to train a bot to periodically validate the feasibility of the solution throughout development.

level using an improved version of the techniques we previously presented in [9]. More specifically, the algorithm described here builds on the ideas from the domain of robotics in [10] to translate a specification given by a system of temporal logics (TLs) into an automaton that abstracts the temporal structure of the predicate into states with localized tasks. In addition, the generated automaton and reward function are interpretable by design, making it possible to gain a clear idea of the agent's progress in satisfying the specification at all times by examining the reward accumulated during training and the automaton states visited. These agents can be used periodically to perform regression testing on their corresponding in-game validations, allowing the human tester to focus only on reviewing more complex features or sections where the bot alerts they might have undergone significant changes in gameplay. To analyze the feasibility of our approach as an automated testing methodology, we show its applicability in a number of different environments and validation tasks developed in Unity3D [11], training agents for each use case and studying their response to the question of whether it is still possible to satisfy the original design plan when modifications are made in those environments.

The rest of this article is organized as follows. Section II introduces related work in the field of automated game testing. Section III develops the concepts on TLs and their application to RL needed to understand the techniques involved in our contribution, while Section IV exemplifies these notions on a concrete scenario within our testing environment. In Section V, we elaborate on the methodology proposed in this article and the setup used to train our testing agents, while Section VI details the experiments carried out to evaluate this strategy on typical video-game use cases. Finally, Section VII concludes this article.

II. Automated Video Game Testing

Automated video game testing has been gaining traction in both academia and industry in recent years as a consequence of the growing need to maintain adequate coverage of games that are increasingly large in domain and scope [12]. In this section, we will review some of the most relevant strategies proposed in the literature in relation to our contribution.

One of the simplest automated playtesting approaches involves manually recording traces, in which a human player performs the designated tasks, replaying them from time to time within the game environment to confirm that these actions can continue to meet the original objectives [13]. This methodology, as expected, is only functional in static and/or deterministic contexts, motivating the need to produce intelligent agents that are able to adapt to dynamic environments. Currently in the industry, most of the testing techniques used are based on classical AI, such as manually programmed agents via behavior trees or state machines, model-based approaches [2], [3], [4], or randomized exploration methods to complement manual testing tasks.[1,2] While many of these strategies are applicable to simple environments, most do not scale to more complex, 3-D, or high-dimensional state space games.

Alternatively, several other works have proposed the use of machine learning techniques for quality control in video games. Concretely, strategies that make use of RL are particularly prevalent. Gordillo et al. [5] proposed a solution based on intrinsic rewards to explore complex 3-D environments in order to find in-game issues that are difficult to detect manually. Bergdahl et al. [6] compared traditional strategies based on navigation meshes with reinforcement agents trained to reach arbitrary positions in a complex 3-D setting, showcasing the latter's ability to spot navigation bugs and exploits. RL has also been used to try to model the interactions of different types of players with game environments in order to provide support in the design process. Agarwal et al. [7] generated agents with different behaviors to play levels from a 2-D platform game, *Sonic the Hedgehog 2*, using a genetic deep RL strategy and offer an intuitive methodology for visualizing the trajectories of the bots. Meanwhile, Ariyurek et al. [8] used agents with different personas (explorers, pacifists, etc.) to discover different ways of approaching levels, a strategy conceptually similar to that in [14], where a variation of Monte Carlo tree search is used to demonstrate how generative player models can be leveraged to replicate archetypal play styles across different levels.

While not as prominent as RL in the game testing literature, imitation learning (IL) is starting to gain some popularity in recent years. By IL we refer to the generation of agents capable of mimicking the behavior of an expert demonstrator that provides a set of state-action pair traces as a reference for the bot to be trained. Some of the most widespread strategies are behavioral cloning [15], which makes use of supervised learning techniques from the dataset of demonstrations to predict the most likely actions for a given state, and generative adversarial imitation learning (GAIL) [16], based on an adversarial approach, in which the rewards of the actuator agent are proportional to how much it manages to deceive a second discriminator agent trained in parallel to learn to distinguish between human and synthetic behaviors.

In game testing, Sestini et al. [17] introduced the curiosity-conditioned proximal trajectories algorithm for testing complex

[1][Online]. Available: https://www.gdcvault.com/play/1027049/-Final-Fantasy-VII-Remake

[2][Online]. Available: https://www.gdcvault.com/play/1026366/Automated-Testing-of-Gameplay-Features

environments, alternating between policies similar to and distant from those of a human via a weighting system of signals based on IL, RL and curiosity. Sestini et al. [18] proposed a methodology for trace validation based on the DAGGER algorithm [19], in which the designer interactively demonstrates their desired behavior in real time. A similar approach is used in the Google Research project *Falken*.[3] Tucker et al. [20] used inverse RL to generate rewards from human traces for agents learning to play *Atari* games.

III. TLs for RL

One of the key disadvantages of RL is that the user must manually translate the requirements of the task to be performed by the agent into a numerical reward function, which often requires significant mathematical background, considerable expertise in machine learning and domain knowledge of the target environment, making these strategies inaccessible to the general user. For complex problems, and especially for those involving temporal dependencies (goals that must be fulfilled in a certain order, for example), these functions can be very challenging to design; furthermore, the user is typically confronted with the need to manually shape the rewards to guide the agent during learning. Reward shaping can become very error-prone and is frequently accompanied by tedious trial-and-error iterations to produce reward functions that induce the expected behavior. It is in this context that TLs have attracted increasing interest in the last decade as a means of supporting the generation of reward functions in RL frameworks from requirements formally modeled in these languages, particularly in the field of robotics [21].

By TLs we refer to any system of rules and symbols that allows reasoning about propositions in terms of their evolution over time. TLs play an important role in the formal verification of requirements in both hardware and software [22], with truncated linear temporal logic (TLTL) [23] being one of the most widely adopted examples of these logics. TLTL formulas are defined based on a series of core logical and temporal operators and, more importantly, predicates of the form $f(s) > 0$, where $f : \mathcal{S} \to \mathbb{R}$ represents a function applied to the system's state $s \in \mathcal{S}$. The latter are the ones that are truly responsible for incorporating a certain knowledge base into our specifications, as well as conditions related to the world and context in which we operate, such as "being close to a point" or "perceiving a high temperature." These conditions are highly domain-dependent and consequently, for each game we contemplate, we will inevitably encounter the need to define a specific set of predicates that will determine the expressiveness of our tasks. Having said this, a TLTL specification $\phi \in \Phi$ adheres to the following syntax:

$$\phi := \top \mid f(s) > 0 \mid \neg\phi \mid \phi_A \wedge \phi_B \mid \phi_A \vee \phi_B \mid \phi_A \Rightarrow \phi_B \mid \Diamond\phi \mid \Box\phi \mid \phi_A \,\mathcal{U}\, \phi_B \mid \bigcirc \phi. \tag{1}$$

Here, $\top$ is the True boolean constant, $f(s) > 0$ is a check on a domain function on the system state, $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction) and $\Rightarrow$ (implication) are Boolean connectives, and $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), and $\bigcirc$ (next) are temporal operators.

From a design perspective, $\Diamond$ indicates a requirement that must be met at some point during the game, and is usually referred to as a liveliness condition, whereas $\bigcirc$ mandates that the associated condition be met in the very next simulation time step. For instance, if our predicate is $\bigcirc\phi$, we anticipate ϕ to be true in the second time step of the run. Typically, designers would need to use a combination of $\Diamond$ and $\bigcirc$ operators to ensure that the specified condition is met at a later time. This allows for sequences of conditions that must be satisfied one after the other by adding blocks of the form $\phi_A \wedge \bigcirc(\Diamond \phi_B)$. For global constraints that must be continuously in effect (which are often called safety conditions), the $\Box$ operator can be employed, followed by the condition we want to persist. This is especially useful for modeling level constraints, such as maintaining a specific item or preventing health points from falling below a certain threshold.

Among TLs, the focus has been mostly on those that allow the formulation of quantitative semantics, commonly known as robustness metrics. As opposed to qualitative semantics, which simply indicate whether or not a system trace satisfies a logical predicate, these robustness measures make it possible to provide a numerical assessment of how well a state trace $\mathcal{T} = (s_1, \ldots, s_n) \in \mathcal{S}^n$ adheres to the given specification through a robustness function $\rho : \mathcal{S}^n \times \Phi \to \mathbb{R}$, which assigns a real-valued number to each predicate ϕ from the predicate space Φ and system trace $\mathcal{T}$.

Once a quantitative semantics is available (and this need not be unique), the obvious strategy to guide learning in a RL setting on the basis of TL specifications is to assign a reward to the agent at the end of each training episode given according to the robustness of the trace generated during the episode, $\rho(\mathcal{T}, \phi)$ [24]. This strategy suffers however from a problem of reward sparsity during training, as reinforcement can only be provided for a whole trajectory, hampering the learning of complex or long-lasting tasks in a significant way.

In response to this problem, Li et al. [25] introduced the concept of Finite State Predicate Automata (FSPA) to abstract away the temporal dependencies of the original predicate and provide dense rewards based on robustness variations between automaton transitions. Essentially, an FSPA can be thought of as a finite state automaton where transitions are associated with TL predicates, such that a transition occurs when the robustness of its linked predicate is both positive and maximal (if multiple edges meet this criterion). This allows the global problem to be converted into small substeps localized in states of the automaton where the progress conditions are well defined. As any given transition in FSPAs comes with an associated robustness variation, these can be used as a step-based reward instead of the episodic one described earlier. In addition, trap states are introduced in these FSPAs to represent task failure.

An additional advantage of this construction is that it eliminates the need to both define and compute the robustness of temporal operators in the predicates of our language, which is often a complex and costly process. In the case of the grammar from (1), here we define the robustness function $\rho_s(\phi) := \rho(s, \phi)$ that

[3][Online]. Available: https://github.com/google-research/falken

outputs $\rho_s(\top) = 1$, $\rho_s(f(s) > 0) = f(s)$, $\rho_s(\neg\phi) = -\rho_s(\phi)$, $\rho_s(\phi_A \wedge \phi_B) = \min(\rho_s(\phi_A), \rho_s(\phi_B))$, and $\rho_s(\phi_A \vee \phi_B) = \max(\rho_s(\phi_A), \rho_s(\phi_B))$, aiming to keep the range of all the domain functions we define in the same interval $f(s) \in [-1, 1]$ so that they are comparable to each other and no conditions outweigh others in composite predicates. Note how here we drop any mention of traces in favor of single states (now $\rho : \mathcal{S} \times \Phi \to \mathbb{R}$), which are the only evaluations that we will need to perform when relying on a FSPA structure.

In the literature, these automata are usually implemented by means of deterministic Rabin automata (DRA) or limit-deterministic Büchi automata (LDBA). External libraries like Rabinizer [26] offer convenient and optimized solutions for automatically carrying out these conversions in the case of DRAs, making the process easily embeddable with game engines and highly practical, which is why in this work we opt for DRAs instead of LDBAs. Whenever we refer to the automata used in our approach, we will be talking about DRAs. Further conversion options are described in [27].

IV. A Game Environment Example: Liquid Snake

To illustrate the concepts from Section III, we will now provide an example in the test environment used in this work of how TLTL can be used to formulate a design specification for one of its levels, together with a possible FSPA generated from the requirements to be used later in a RL framework.

Liquid Snake, described in [28], is a prototype of a 3-D stealth game created for the purpose of conducting AI-driven testing experiments within the Unity3D engine [11]. In this game, the player's objective is to guide the main character through a series of chambers and lead them to the level exit while evading enemy detection. Enemies display typical stealth game behavior by patrolling their assigned routes in search of their target. When they spot the player, they give chase and continue to pursue them until their target is either defeated or manages to evade their view. In the latter scenario, the enemies retrace their steps to the last known location of the player, conduct a brief search, and then return to their original patrol path if they are unable to find them. The main character can jump to overcome obstacles and reach otherwise inaccessible distant platforms, as well as crouch or crawl to take cover behind low walls, allowing them to evade enemy sight or navigate sections with low ceilings.

For the examples presented here, we define $\eta : \mathcal{S} \to [0, 1]$ as the proportion of the player's remaining health, with $\pi := \eta(s) > 0$ representing the predicate "the character is still alive." For the notion of proximity of the character to a target *tar*, we define the function $\gamma_{\text{tar}} : \mathcal{S} \to [-1, 1]$

$$\gamma_{\text{tar}}(s) = \begin{cases} 1 - \frac{\delta_{\text{tar}}(s)}{b}, & 0 \leq \delta_{\text{tar}}(s) < b \\ \frac{\delta_{\text{tar}}(s)^2 - 2B\delta_{\text{tar}}(s) + 2bB - b^2}{(b-B)^2}, & b \leq \delta_{\text{tar}}(s) \leq B \\ -1, & B < \delta_{\text{tar}}(s) \end{cases} \quad (2)$$

where $\delta_{\text{tar}}(s)$ is the Euclidean distance between the character and the target in s, b is the lower bound below, which we consider proximity to be achieved, and B is the upper bound beyond, which we assume the distance to be too large to be considered close. This is nothing more than the second-degree polynomial that evaluates to -1 when the distance between objects is at its maximum ($\delta_{\text{tar}}(s) = B$) and 0 when it equals the proximity threshold ($\delta_{\text{tar}}(s) = b$), which turns into a linear function approaching 1 as the distance gets closer to 0. Empirically, we choose B as the size of the level's bounding box, $b = 1$, and write $\gamma_{\text{tar}} := \gamma_{(\text{tar})}(s) > 0$ to denote the player's proximity predicate to a target in the scene. For instance, if we set $B = 10$ and are currently 5 units away from the target, then $\gamma_{\text{tar}}(s) \approx -0.69$. This function has a steeper gradient as $\delta_{\text{tar}}(s)$ approaches b and is only positive when proximity has been reached.

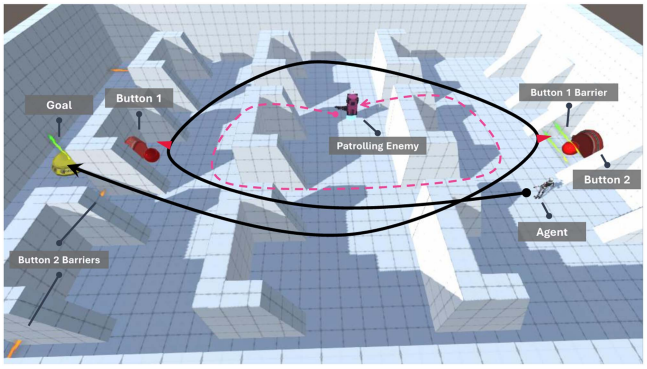


Fig. 2. Screenshot of the "Cover and Hide" environment.

One of the levels included in our experiments, depicted in Fig. 2, introduces an enemy that patrols an environment filled with multiple grid-based obstacles, obstructing visibility and movement for both the enemy and the player. The objective here is to visit and activate the button to the left, s_1 ("Button 1" in the figure), to unlock the barrier guarding the button on the right side of the environment, s_2 ("Button 2"). After that, the player must visit this second button to unlock the barriers protecting the area in the leftmost section of the level, and then reach the goal located there, e. All of this must be done while remaining alive. In terms of TLTL the specification for this level can be expressed as follows:

$$\Diamond(\gamma_{s_1} \wedge \bigcirc(\Diamond(\gamma_{s_2} \wedge \bigcirc\Diamond\gamma_e))) \wedge \Box\,\pi. \quad (3)$$

From the designer's point of view, this translates into listing a series of conditions that must be met sequentially through blocks of "and next eventually" operators as mentioned in Section III, in this case visiting three level points in order, along with a clause with task restrictions ("without dying").

This predicate can then be translated into a DRA using the procedures listed in Section III, outputting an FSPA like the one in Fig. 3. In this figure, the connections between states are labeled with the predicates that must be satisfied to trigger the transition. These automata are constructed so that there is always an edge equipped with a true formula, meaning that there will always be a transition to take at any given step. Furthermore, each of the states is labeled with a predicate representing what part of the specification remains to be fulfilled at that node, with the initial state holding the complete original formula, and subsequent states containing increasingly smaller formulas. We will discuss

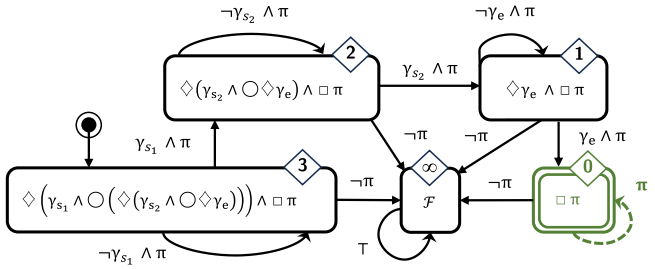


Fig. 3. FSPA translation of the predicate from formula 3.

later how this property can be of help in supplementing the information provided by regression tests. Also note here the trap state that can be reached from any point of the automaton by violating the safety condition (when the player loses all health points), labeled with a false predicate indicating that it is no longer possible to satisfy the specification from it. The final state, on the other hand, is still accompanied by the original safety condition and outlined in green to denote that the automaton will accept any trace that ends in this state. Note that although this example produces a fairly simple automaton for the sake of readability, the FSPAs generated can be arbitrarily complex, especially when conditional clauses are introduced and multiple nested temporal operators are combined.

V. Proposed Method

We propose an approach based on RL with reward functions automatically generated from design specifications in TLTL to conduct automated regression tests throughout the development of the game. These specifications provide a framework to express a course of action whose validity the designer wishes to verify over a given level on a regular basis. In this section, we first detail our setup and training algorithm, and then describe why we decide to adopt this methodology.

A. Reward Generation Algorithm

Our approach to the reward allocation strategy, which is formally stated in Algorithm 1, is based on the concept of tracking progress within the task specification. We heavily rely on the structure of the reward automaton to measure this progress. In simple terms, at each step of the simulation, we calculate a value between 0 and 1, representing the proportion of the specification that has been satisfied up to that point. The agent is then rewarded based on how much it has improved compared to the best value of this metric during the episode.

We start with a design specification provided via a TLTL predicate and translate it into an equivalent DRA-based FSPA. For each state as in the automaton, we then determine the minimum number of transitions required to reach a goal state. This metric is denoted by $d_g(as)$. This also allows us to identify "trap" states from which it is impossible to reach a goal (i.e., $d_g(as) = \infty$). This is valuable for determining when we can end the episode due to a specification violation. In the example from Fig. 3, $d_g(as)$ is indicated by a diamond in the upper right corner of each state. Note how this is ∞ for the trap state and 0 for the single goal state outlined in green. With these distances, we can define SpecSize, as $\max_{as}(d_g(as))$ (line 4). This helps us reason about our progress based on an upper limit. In our example, SpecSize $= 3$.

With this information, during each agent step t (not necessarily linked to the game's update step), we apply the action chosen by the agent's current policy to the game state $gs^t \xrightarrow{\text{Action}} gs^{t+1}$ and update the automaton's state $as^t \xrightarrow{gs^{t+1}} as^{t+1}$. To do that, we evaluate the robustness of each outgoing edge ϕ_e from as^t, $\rho_{gs}(\phi_e)$, and take the transition with the highest value. We then estimate the base progress in the specification from as^{t+1} as $\frac{\text{SpecSize} - d_g(as^{t+1})}{\text{SpecSize}}$. This is stated in lines 7–11 and tells us how many steps we have taken toward a goal state relative to the maximum number of steps required to reach it. Continuing with our example, the state as with label $\diamondsuit\gamma_e \wedge \square\,\pi$ has $d_g(as) = 1$, and so we can estimate that we have completed at least $2/3$ of the task and initialize the progress at that value. This way of estimating progress assumes that all subtasks are equal in size, which might be an oversimplification depending on the context.

Now, if the transition would keep us in the current state, the reward is adjusted based on the best robustness that would improve the distance (lines 12–14). This indicates how much progress has been made in the current task. For instance, if we are at the same as from before, the only edge that would bring us closer to the goal would be the one with predicate $\phi_e = \gamma_e \wedge \pi$. If at the current time-step we are at full health and $d_e = \frac{t+T}{2}$ (half of the maximum distance to the level target), then we would have that $\rho_{gs}(\gamma_e) = -0.75$ [see (2)], $\rho_{gs}(\pi) = 1$ and $\rho_{gs}(\phi_e) = \min(-0.75, 1) = -0.75$. As the base progress was 2/3 this means that our actual progress at the current step is $\frac{2}{3} + \frac{1-0.75}{\text{SpecSize}}$, which is approximately 0.75. If this value is higher than the last maximum recorded up to that point, the difference is added to the step reward (lines 25–28).

However, it could be the case that no transition would allow us to improve the distance to the goal. This means we are either in a trap state (line 19) from which we can never reach a goal or are located in a goal state from which we cannot improve further because it is already as good as possible (line 16). Typically, when we reach a trap state, it is common to end the episode directly. In an acceptance state, it might sometimes be interesting to let the episode continue beyond the original goal for the agent to learn how to stay in it and improve the specification's quality. Nevertheless, for now, we are only interested in tasks that end after reaching a goal state, so we will not delve into the optimization details in that case.

Considering this structure, it is easy to see that the total accumulated reward throughout the episode always falls between 0 and 1. This implies that there is no difference between completing the specification sooner or later, as long as it eventually gets fulfilled. Because of this, it is often useful to introduce a penalty in each simulation step to encourage the agent to optimize the time required to complete the tasks (line 29). This also provides an initial estimate of how long it might take to fulfill the design intent. To maintain the interpretability of the results, we choose the value of the existential penalty in

Algorithm 1: Algorithm for Reward Generation-Rabin FSMA.

Initialize game state as gs^0
Initialize automaton state as as^0
Initialize episode progress as 0
SpecSize $\leftarrow \max_s(d_g(as))$
while $t < t_{\max}$ **do**
  Initialize step reward as 0
  Action $\leftarrow$ policy(gs^t, as^t)
  Step Action in game state $gs^t \xrightarrow{Action} gs^{t+1}$
  Use gs^{t+1} and as^t to tick the Automaton, computing $\rho_{gs^{t+1}}(\phi_e)$ for all predicates ϕ_e in edges attached to as^t
  Select edge from as^t with the highest $\rho_{gs^{t+1}}(\phi_e)$ and transit Automaton using that edge $e : as^t \rightarrow as^{t+1}$
  step progress $\leftarrow \frac{\text{SpecSize} - d_g(as^{t+1})}{\text{SpecSize}}$
  if $(as^t = as^{t+1})$ **then**
    if other e' would improve distance to goal **then**
      step progress += $\frac{1+\max_{e' \in impEdges}(\rho_{gs^{t+1}}(\phi_{e'}))}{\text{SpecSize}}$
    else
      if e leads to a goal state **then**
        step progress = 1
        terminate the episode (success)
      else
        step reward -= terminal penalty
        terminate the episode (failure)
      end if
    end if
  end if
  if step progress > episode progress **then**
    step reward += step progress - episode progress
    episode progress = step progress
  end if
  step reward += existential penalty
  grant step reward to agent
end while

such a way that if the agent fails to complete the specification within the provided time, the total penalty amounts to -1. This changes the total reward range to [-1, 1]. With this decision, a negative value indicates a failure to meet the specification, with values closer to 0 representing progress towards task completion, while a positive value corresponds to successful specification completion, with higher values indicating "faster" policies. If existential penalties are applied, it is often advisable to apply a terminal penalty upon reaching a trap state, so as to prevent the agent from attempting to break the specification on purpose (for instance, by letting themselves die) in order to avoid further negative rewards.

B. Training Setup

Once the reward generation method has been established, we can opt for any RL algorithm that suits the characteristics of our problem to train the testing agents. In our case, we decided to use the proximal policy optimization (PPO) algorithm [29] due to its relative generality and lack of over-reliance on hyperparameter tuning.

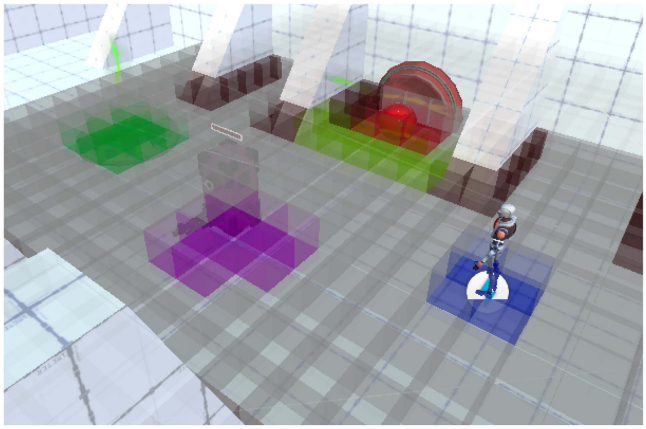


Fig. 4. Semantic map in our environment with a single vertical dimension.

On a different note, in an effort to generalize the approach to as many game environments as possible, we define the state space based on a typical structure of 3-D spaces and actions, which tend to be prevalent in a good part of modern titles.

To setup the training of a new testing agent, developers define the specification on which the test is to be performed by means of a predicate in TLTL. Depending on what is to be described, this process may require the declaration of new atomic predicates containing domain knowledge for the game (e.g., that a switch is activated, that the player does not run out of ammunition, etc). From this specification, it is possible to compile a DRA-based FSPA as described in Section V-A, which will be used to generate the rewards for the RL algorithm. The user can also specify which entities and variables from the scene should be taken into account by the agent for sensing the environment creating a vector v_{user}. In our case, we include information, such as end or intermediate targets, enemies, switch activation states, character health points, whether the player can jump or crouch, or any other piece of information deemed useful to satisfy the specification. We currently allow the user to enter references to integer or float values (normalized by a range of expected values), boolean variables, or relative positions of the agent to entities in the scene. More specifically, for each of the objects in the scene whose position is regarded as relevant, the vector between it and the agent is calculated and its length is normalized to the game area, given by a bounding box. Finally, and following the approaches in [17] and [18], we use a solution based on a semantic map given by a discrete categorization of the space and the elements around the agent to give it a certain level of spatial perception. This is defined by a 3-D grid of cube-shaped regions centered on the agent, with each of these regions describing the type of the object included in it by means of an integer value. The size of the regions, as well as the number of regions per dimension (s_x, s_y, s_z) and the entities detectable by the map can be configured by the developer. An example in our environment for a semantic map with a single vertical dimension can be seen in Fig. 4. Each cube-shaped region represents a

discrete categorization of the entity contained in it (e.g., enemies, switches, goals, etc).

As for the architecture of the neural networks used, we deploy structures with two fully connected layers, each with 512 hidden units. The first segment of the network's input is given by a one-hot-encoded vector $v_{\text{Automata}} \in \mathbb{R}^n$ with a length equal to the number of states n in the automaton, with all values set to 0 except for the one corresponding to the currently active state. Second, the entities and variables declared as relevant by the user v_{user} are appended to the input vector. The last part of the input of these layers is given by the state of the semantic map $M \in \mathbb{R}^{s_x \times s_y \times s_z}$, previously fed in parallel to a convolutional layer with output $x_M \in \mathbb{R}^d$.

VI. Experiments

We applied the proposed approach to multiple levels in the test environment described in Section IV, on each of which a design specification was formulated in TLTL. To evaluate the performance of our approach, we define these specifications to encompass several typical settings that designers may be interested in validating in their work, with use cases addressing a variety of challenges that can arise when training testing agents. In this environment, agents have a set of five discrete actions: move in the *X*-axis (none, left, and right), move in the *Y*-axis (none, backward, and forward), jump, crouch, and sprint.

A. Experimental Setup

For the experiments conducted in this work, we made use of the ML Agents library [30] with the PPO algorithm [29] to train agents and the reward function generated from the specification as detailed in Section 1 (learning rate $= 0.0003$, $\beta = 0.01$, $\epsilon = 0.2$, $\lambda = 0.95$, and $\gamma = 0.99$). The only parameters that were changed between levels of the experiment were the number of steps per episode, which is inherently dependent on the estimated length of each task, and the maximum number of steps per training session, typically linked to the complexity of the specification. It is worth noting that the number of steps per episode has a direct impact on the scale of the accumulated episodic reward due to the nature of the existential punishment in the algorithm. Setting a step number too close to the optimal value (which is unknown to the practitioner except for maybe some empirically recorded estimate) often results in reward values very close to 0, whereas increasing this margin allows for results closer to 1. Therefore, our primary concern is to achieve consistent positive values for the reward, and the magnitude of the reward beyond this threshold is not as crucial.

We compare our approach to an imitation baseline in which a human tester provides a set of demonstrations (at least $10 - 20$ complete episodes per level) satisfying the specification for each of the levels, which are then used to train an agent to learn to mimic the expert's behavior based on this dataset of traces. More specifically, we bootstrap a neural network with an architecture identical to the one of the TL model using behavioral cloning (first 5% of training steps), and then run training using a GAIL-generated reward signal (learning rate $= 0.0003$, encoding size $= 128, \gamma = 0.99$) and adding a terminal reward at the end of each episode of +1 if the specification was successfully completed and -1 otherwise. For each of the methods, we are interested in

1) the agent's success rate in satisfying the specification in the environment, calculated as the proportion of simulations, in which it achieves the level objective over 25 runs,
2) the agent's average progress in the specification's predicate, i.e, at which point in the specification it tends to be when the time allocated to the task runs out or when a safety condition is violated, and
3) the average reward assigned by the FSPA during the simulated episodes, which can be seen as a metric that aggregates the progress value with the "quality" of the generated traces in terms of speed and compliance with the safety conditions.

Note that these metrics are always computable even if they are not used to guide the training of an agent by RL. In fact, for each of the levels, we keep a record of the values of these metrics during demonstrations by human experts in order to provide a reference of the values that would be expected in a real player. In both approaches, training stops when it is detected that the model starts to satisfy the specification consistently (i.e., pass rate $= 1$ for the duration of a fixed number of episodes) or when a training time of 20 h is exceeded without this condition being met, after which the best model based on the pass rate metric is saved.

B. Regression Test Use-Case Evaluation

To evaluate the viability of our approach, we define a series of use cases for testing agents that address some of the typical challenges that can be encountered in modern video game scenarios when training policies that satisfy design specifications. On the one hand, we wish to verify that our methodology is able to generate policies in a reasonable time to validate the specification on the original environment. On the other hand, since the main purpose of these agents is to be used for regression testing, we focus on analyzing how each strategy reacts to common development scenarios that induce unexpected changes in previous levels. Thus, for each of the environments presented, we consider cases in which a designer modifies a shared asset in different scenes of the game to accommodate the design of later levels, and analyze the ability of each testing agent to adapt to the potential unexpected effects of these changes in their corresponding levels.

Where these edits do not completely invalidate the original specification, we expect agents to continue to be able to satisfy their predicates, possibly informing of meaningful variations in their performance. That is, if the agent is still able to validate the specification, but its reward metrics are noticeably altered, we can issue a warning that there is a chance that the gameplay experience of the level has been compromised. Predictably, there will be situations where the agent is no longer able to complete the task after the change, in which case we would like the test to be as informative as possible as to what may have caused this failure to meet the objective. Our intention will therefore be to ensure exhaustive coverage of the game design specifications, running periodic automated checks and reporting suspicious

TABLE I
QUANTITATIVE RESULTS OF OUR EXPERIMENTS

	Case 1									
	Pass Rate			Specification Progress			Automaton Reward			Time
	v_0	v_1	v_2	v_0	v_1	v_2	v_0	v_1	v_2	
Human	1.00	1.00	0.00	1.00 ± 0.00	1.00 ± 0.00	0.50 ± 0.01	0.63 ± 0.01	0.65 ± 0.02	-0.50 ± 0.01	-
Imitation	0.92	0.32	0.00	0.98 ± 0.05	0.79 ± 0.22	0.47 ± 0.07	0.50 ± 0.19	-0.09 ± 0.37	-0.53 ± 0.08	20h
Ours	1.00	0.90	0.00	1.00 ± 0.00	0.93 ± 0.21	0.49 ± 0.05	0.64 ± 0.01	0.39 ± 0.37	-0.51 ± 0.05	3.5h
	Case 2									
	Pass Rate			Specification Progress			Automaton Reward			Time
	v_0	v_1	v_2	v_0	v_1	v_2	v_0	v_1	v_2	
Human	1.00	1.00	0.80	1.00 ± 0.00	1.00 ± 0.00	0.93 ± 0.14	0.73 ± 0.01	0.74 ± 0.01	0.41 ± 0.40	-
Imitation	0.05	0.00	0.00	0.74 ± 0.27	0.49 ± 0.26	0.41 ± 0.22	-0.25 ± 0.27	-0.50 ± 0.27	-0.58 ± 0.22	20h
Ours	1.00	0.50	0.20	1.00 ± 0.00	0.85 ± 0.20	0.70 ± 0.25	0.74 ± 0.03	0.21 ± 0.55	-0.14 ± 0.51	2.7h
	Case 3									
	Pass Rate			Specification Progress			Automaton Reward			Time
	v_0	v_1	v_2	v_0	v_1	v_2	v_0	v_1	v_2	
Human	1.00	1.00	1.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.76 ± 0.01	0.73 ± 0.02	0.74 ± 0.08	-
Imitation	0.40	0.24	0.10	0.82 ± 0.37	0.80 ± 0.36	0.71 ± 0.24	0.01 ± 0.51	-0.11 ± 0.45	-0.24 ± 0.33	20h
Ours	1.00	0.93	0.47	1.00 ± 0.00	0.96 ± 0.17	0.89 ± 0.23	0.66 ± 0.01	0.64 ± 0.36	0.27 ± 0.46	6.8h

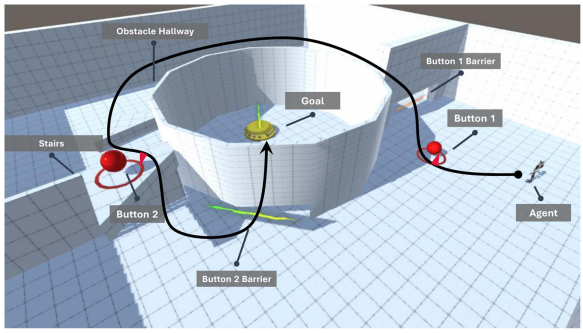


Fig. 5. Screenshot of the "Navigation" environment.

scenarios, these being the only ones that need to be manually reviewed by a human tester. Note that in the testing process the same agent is always used without retraining until a significant change is found.

Case 1—Validating navigation: One of the most common testing use cases is to check that the critical path of a level remains feasible from a navigability point of view. The classic way to manage navigation in modern games involves prebaking a navigation mesh and applying classical pathfinding algorithms on it, but this strategy is usually best suited for enemy movement in constrained environments, requires recalculation every time the level structure is modified, and is often too costly to adapt to environments with a high degree of freedom of movement. For this use case, we consider a first level that serves as a tutorial in our test environment, shown in Fig. 5, introducing players to the basic mechanics of the game. Here, the character must activate the switch on the right side of the image s_1 ("Button 1") to open the barrier on the right wall, which then allows passage through the gap by entering a crouched position. After going through an obstacle section and up a set of stairs, players must perform a careful jump to reach the switch on the left side of the image s_2 ("Button 2"), which in turn unlocks the exit e accessible by purposely falling down from the upper platform. In TL, this can be expressed via the following predicate:

$$\Diamond(\gamma_{s_1} \wedge \bigcirc(\Diamond(\gamma_{s_2} \wedge \bigcirc\Diamond\gamma_e))) \tag{4}$$

following the notation from (3).

Table I summarizes the results for this use case and the ones described hereafter. For each of the metrics mentioned earlier, we report the values produced with each training model in 25 simulations on the level in its original state (v_0) and after applying a subtle and a significant modification on assets shared with other levels (v_1 and v_2, respectively), together with the time needed to train a successful agent. Values are given as averages with their corresponding standard deviation.

In this environment, the first modification involves a slight alteration of the shape of the assets corresponding to the obstacles in the central corridor. This does not affect gameplay significantly, but it does require a minor adjustment to the route taken. The second variation represents an extensive tuning of the obstacle, stretching it so that it starts obstructing the path entirely, thus preventing the level from being completed. As shown in the table, after the first change the imitation model perceives a significant alteration in the level, with a sharply reduced performance after applying the change, while our model, despite showing a small drop in performance, remains at relatively stable values, with the test yielding a consistently positive result. This is a desirable property in testing: given subtle modifications that do not alter the gameplay of the level, a good testing agent should not warn against potential alterations (i.e., it should not throw false positives). On the other hand, both the human and the two models are unable to complete the level after the second change, with both models exhibiting similar results and reporting true positives. However, in the latter case, our model provides us with an additional tool to yield a more informative test result: since the FSPA logs are available during the executions, we are able to identify which state the agent was in before becoming locked in the environment, and how far away it was from moving on to the next one. As mentioned before, each state of an FSPA is labeled with the predicate representing what part of the specification remains to be fulfilled, so we can easily gain a preliminary but intuitive idea of what might have gone wrong and direct our attention to that point in the level during human review. Hence, another desirable property is for the agent's progress to be as close as possible to that of the human tester within the same

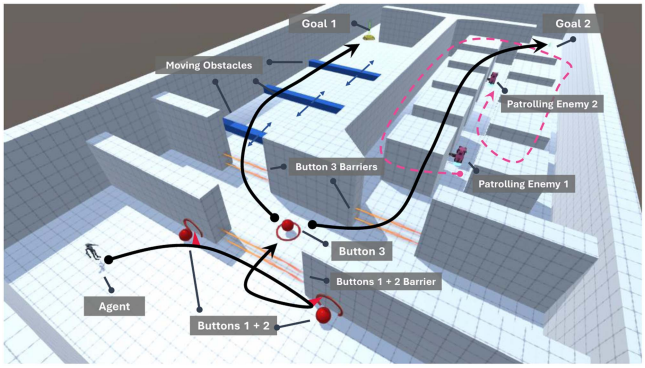


Fig. 6. Screenshot of the "*Branching Tasks*" environment.

situation, so that the result provides information that is as close as possible to a real test.

Case 2—Validating multistage environments: Another typical situation can be seen in environments, such as the one used as an example in Fig. 2, where the design specification requires a series of steps to be followed sequentially, potentially leading to stages where it may be necessary to take very different actions depending on which part of the test the player is in. Returning to the prior example with specification given in (3), the player is required to traverse the same spaces moving in opposite directions based on which switches they have already activated, while looking for ways to avoid detection by the patrolling enemy. Such scenarios can be problematic in machine learning, where it is often difficult for the agent to efficiently learn to discriminate between the stimuli needed to distinguish phases of the sequence where its behavior is likely to be significantly varied.

Here, both modifications are applied to the behavior tree that defines the AI of the enemy. The first situation involves only parameter adjustments to improve the responsiveness of the enemy at later levels, while the second adds an additional behavioral branch in an attempt to make the sections involving tight spaces more compelling. From the results in the table we can see that, although our approach suffers a drop in performance, it keeps being able to regularly complete the level after the first change, indicating that the specification is still valid, whilst requiring human review to verify that the user experience is intact. The second, seemingly innocuous change to the level that motivates the asset modification ends up making it noticeably harder: the human no longer has a perfect success rate, and our agent suffers an even more substantial drop in performance, but continues to indicate that the specification is satisfactory, albeit in need of manual review. Moreover, despite the low success rate, progress remains high and close to that of the human tester. The agent trained by imitation does not manage to complete the specification in any of these cases, making it unfit for testing.

Case 3—Validating branching environments: The third and final level of the experiments, depicted in Fig. 6, provides an example of a complex environment where the agent is encouraged to follow very different behavior flows depending on random conditions. In this scenario, the design specification requires the player to start by sequentially activating the switches in the left room of the level (s_1 and s_2) to disable the first laser barrier. Subsequently, the switch placed behind this barrier s_3 must be activated, which randomly unlocks one of the barriers on its sides. Depending on which section is unlocked at this point, the character must address a different problem: if access to the left section is granted (σ_{left}), the player must navigate a corridor with moving obstacles, timing their jumps correctly to make progress; conversely, if the right section is unlocked (σ_{right}), the player must face an area with grid-based obstacles patrolled by two enemies thoroughly exploring the intersection points. In both cases, the goal is to reach the exits placed at the end of their respective corridors ($\gamma_{e_{\text{left}}}$ and $\gamma_{e_{\text{right}}}$). In terms of TL, this is modeled by the following specification:

$$\Diamond(\gamma_{s_1} \wedge \bigcirc(\Diamond(\gamma_{s_2} \wedge \bigcirc\Diamond(\gamma_{s_3} \wedge \bigcirc\Diamond((\sigma_{\text{right}} \Rightarrow \gamma_{e_{\text{right}}})) \wedge (\sigma_{\text{left}} \Rightarrow \gamma_{e_{\text{left}}})))))) \wedge \Box\pi. \quad (5)$$

The first modification here involves slightly tweaking the path of the moving obstacles to include some vertical elevation, as well as the same changes to the enemy AI that were applied in the first variation in case 2; this does not induce substantial variations in gameplay from a human perspective, but it does force subtle adjustments to the trajectories and jumps to be taken. The second modification applies the same addition to the behavior tree described for the previous case to the enemy AI; this ends up requiring a slightly more cautious strategy to traverse the right section of the level, as enemies are more exhaustive in exploring tight spaces. Looking at the results in the table, we can see that our model only perceives a subtle change in v_1, as expected, while the second change results in a substantial drop in agent performance. This is actually expected and positive, as although the difficulty of the level has not been significantly increased, the way in which it must be dealt with does change, requiring human review to ensure that the design continues to be adequate. Meanwhile, the imitation model has very limited success rates from the outset and progressively lower success rates with each asset update, rendering it once again unreliable for testing.

Overall, these experiments highlight the preliminary potential of our strategy to generate agents with sound regression test properties, with manageable training times especially considering that the ultimate goal is to maintain continuous validation throughout development rather than interactive in-situ validation, and more informative than nonspecification-based agents. The fact that our agents do not always maintain the original levels of performance in the face of subtle changes can be viewed as a virtue as evidenced in case 3, where potentially unexpected variations in the way the level was approached could have gone unnoticed by different human testers playing it with no awareness of how it was handled in its original version. Lastly, an advantage of RL-based systems is that once the presence of a substantial change is established, it is possible to automatically launch a new training session to fine-tune the behavior to the change, so that when several parts of the game are affected, it is no longer necessary for human testers to rerecord traces for all conflicting scenarios.

VII. Conclusions and Future Work

In this article, we introduce a RL approach to perform automated regression tests in the context of the validation of design specifications throughout the development of a game. We propose a new algorithm for generating dense rewards from descriptions of the tasks to be performed expressed through a TL language (TLTL), addressing the issue of reward shaping and only requiring the designer to provide a specification of what they would like to validate within an environment in order to produce a reward function that guides the agent to a behavior that satisfies it. This algorithm is based on the idea of specification progress and is naturally interpretable by a human practitioner. To investigate the performance of our strategy, we design a series of experiments aimed to produce agents capable of satisfying different specifications in three design validation use cases in a 3-D stealth game environment. For each of these cases, we introduce modifications on common project assets that could potentially affect gameplay of the original levels and analyze the ability of the produced agents to both cope with and report possible unexpected consequences on player experience and design soundness. Our experiments offer preliminary results on how this type of approach can be more effective than other techniques based on IL when running periodic regression tests during development.

Our future work focuses on developing regression test analysis methods to improve the information provided by agents, aiding quality control. Using formal language and specification-based models, we aim to generate meaningful descriptions of gameplay variations detected by testing agents. In addition, we plan to study the usability of our system to ensure smooth integration into team workflows. This includes evaluating how easily developers can implement these methods (considering here conceptual complexity and intrusiveness of our framework's ideas, and specially the potentially challenging adoption of TLTL as a design specification language), as well as new ways to deal with computational overhead and deployment logistics, which are often problematic side-effects of using RL-based approaches in real environments.

References

[1] S. Stahlke, A. Nova, and P. Mirza-Babaei, "Artificial players in the design process: Developing an automated testing tool for game level and world design," in *Proc. Annu. Symp. Comput.-Hum. Interact. Play*, 2020, pp. 267–280, doi: 10.1145/3410404.3414249.

[2] G. Lovreto, A. T. Endo, P. Nardi, and V. H. S. Durelli, "Automated tests for mobile games: An experience report," in *Proc. 17th Braz. Symp. Comput. Games Digit. Entertainment*, 2018, pp. 48–488.

[3] S. Iftikhar, M. Z. Iqbal, M. U. Khan, and W. Mahmood, "An automated model based testing approach for platform games," in *Proc. ACM/IEEE 18th Int. Conf. Model Driven Eng. Lang. Syst.*, 2015, pp. 426–435.

[4] S. Stahlke, A. Nova, and P. Mirza-Babaei, "Artificial playfulness: A tool for automated agent-based playtesting," in *Proc. Extended Abstr. CHI Conf. Hum. Factors Comput. Syst.*, 2019, pp. 1–6.

[5] C. Gordillo, J. Bergdahl, K. Tollmar, and L. Gisslen, "Improving playtesting coverage via curiosity driven reinforcement learning agents," in *Proc. IEEE Conf. Games*, 2021, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9619048/

[6] J. Bergdahl, C. Gordillo, K. Tollmar, and L. Gisslen, "Augmenting automated game testing with deep reinforcement learning," in *Proc. IEEE Conf. Games*, Aug. 2020, pp. 600–603. [Online]. Available: https://ieeexplore.ieee.org/document/9231552/

[7] S. Agarwal, C. Herrmann, G. Wallner, and F. Beck, "Visualizing AI playtesting data of 2D side-scrolling games," in *Proc. IEEE Conf. Games*, 2020, pp. 572–575. [Online]. Available: https://ieeexplore.ieee.org/document/9231915/

[8] S. Ariyurek, A. Betin-Can, and E. Surer, "Automated video game testing using synthetic and humanlike agents," *IEEE Trans. Games*, vol. 13, no. 1, pp. 50–67, Mar. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8869824/

[9] P. Gutierrez-Sanchez, M. A. Gomez-Martin, P. A. Gonzalez-Calero, and P. P. Gomez-Martin, "Reinforcement learning with temporal logic specifications for regression testing NPCs in video games," in *Proc. IEEE Conf. Computatonal Intell. Games*, 2023, pp. 1–8.

[10] X. Zhao and M. Campos, "Reinforcement learning agent training with goals for real world tasks," 2021, *arXiv: 2107.10390*.

[11] Unity-Technologies, "Unity real-time development platform 3D, 2D VR & AR engine," 2022. [Online]. Available: https://unity.com/

[12] C. Politowski, Y.-G. Guéhéneuc, and F. Petrillo, "Towards automated video game testing: Still a long way to go," in *Proc. 6th Int. ICSE Workshop Games Softw. Eng.: Eng. Fun, Inspiration, Motivation*, May 2022, pp. 37–43, doi: 10.1145/3524494.3527627.

[13] M. Ostrowski and S. Aroudj, "Automated regression testing within video game development," *GSTF J. Comput.*, vol. 3, no. 2, Jul. 2013, Art. no. 10, doi: 10.7603/s40601-013-0010-4.

[14] C. Holmgard, M. C. Green, A. Liapis, and J. Togelius, "Automated playtesting with procedural personas through MCTS with evolved heuristics," *IEEE Trans. Games*, vol. 11, no. 4, pp. 352–362, Dec. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8295256/

[15] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Proc. Mach. Intell.*, 1995, pp. 103–129.

[16] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4572–4580.

[17] A. Sestini, L. Gisslen, J. Bergdahl, K. Tollmar, and A. D. Bagdanov, "Automated gameplay testing and validation with curiosity-conditioned proximal trajectories," *IEEE Trans. Games*, vol. 16, no. 1, pp. 113–126, Mar. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9970382/

[18] A. Sestini, J. Bergdahl, K. Tollmar, A. D. Bagdanov, and L. Gisslén, "Towards informed design and validation assistance in computer games using imitation learning," in *Proc. IEEE Conf. Games*, 2023, pp. 1–8.

[19] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artif. Intell. Stat.*, 2011, pp. 627–635.

[20] A. Tucker, A. Gleave, and S. Russell, "Inverse reinforcement learning for video games," Oct. 2018 https://arxiv.org/abs/1810.10593

[21] H.-C. Liao, "A survey of reinforcement learning with temporal logic rewards," 2020. [Online]. Available: https://mediatum.ub.tum.de/doc/1579215/document.pdf

[22] I. Buzhinsky, "Formalization of natural language requirements into temporal logics: A survey," in *Proc. IEEE 17th Int. Conf. Ind. Informat.*, 2019, pp. 400–406.

[23] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3834–3839. [Online]. Available: https://ieeexplore.ieee.org/document/8206234/

[24] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *Proc. IEEE Conf. Control Technol. Appl.*, 2017, pp. 1235–1240.

[25] X. Li, Z. Serlin, G. Yang, and C. Belta, "A formal methods approach to interpretable reinforcement learning for robotic planning," *Sci. Robot.*, vol. 4, no. 37, Dec. 2019, Art. no. eaay6276, doi: 10.1126/scirobotics.aay6276.

[26] Z. Komárková and J. Křetínský, "Rabinizer 3: Safraless translation of LTL to small deterministic automata," in *Proc. Int. Symp. Automated Technol. Verification Anal.*, 2014, pp. 235–241.

[27] K. Y. Rozier, "Explicit or symbolic translation of linear temporal logic to automata," Ph.D. dissertation, Rice Univ., Houston, TX, USA, Jul. 2013. [Online]. Available: https://scholarship.rice.edu/handle/1911/71687

[28] P. Gutiérrez-Sánchez, M. A. Gómez-Martín, P. A. González-Calero, and P. P. Gómez-Martín, "Liquid snake: A test environment for video game testing agents," in *Proc. Actas del I Congreso Español de Videojuegos*, Madrid, Spain, 2022, pp. 1–12. [Online]. Available: https://ceur-ws.org/Vol-3305/paper7.pdf

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[30] A. Juliani et al., "Unity: A general platform for intelligent agents," 2020, *arXiv: 1809.02627*.

Chapter 10

AI Behavior Graphs: A Visual Toolkit for Defining NPC Specifications for Regression Testing

AI Behavior Graphs: A Visual Toolkit for Defining NPC Specifications for Regression Testing

Pablo Gutiérrez-Sánchez[1,*], Marco A. Gómez-Martín[1,*], Pedro A. González-Calero[1,*] and Pedro P. Gómez-Martín[1,*]

[1]*Complutense University of Madrid, Madrid, Spain*

Abstract

Reinforcement learning (RL) offers a promising approach for developing autonomous agents in various domains, including the creation of in-game characters. However, crafting these agents, and particularly designing reward functions for sequential decision-making, remains a significant challenge, often involving iterative trial-and-error processes until achieving satisfactory results. Consequently, these strategies often elude game designers and quality control teams, who could otherwise use them to automate testing procedures. This paper extends our prior work by introducing "AI Behavior Graphs," a visual toolkit designed to simplify the creation of behavior specifications for NPCs (Non-Player Characters). Our approach provides an intuitive graphical interface that enables designers to express their expectations for player-NPC interactions within a game level. These specifications are automatically translated into both Linear Temporal Logic (LTL) and Rabin automata, which can in turn be leveraged to dynamically generate reward functions during agent training. This not only expedites NPC development but also makes RL-based methodologies more accessible to a broader audience of game designers and quality assurance teams. Furthermore, it underscores a critical aspect of our approach: the ability to utilize these agents for playtesting game levels. This application ensures continuous validation of designer expectations throughout the development cycle, enhancing the overall game design process.

Keywords

visual toolkit, automated game testing, game-playing AI, temporal logics, reinforcement learning

1. Introduction

In our previous work [1], we tackled the challenge of automating the generation of non-player characters (NPCs) capable of executing complex sequential tasks within video games. This effort was motivated by the need to enhance the efficiency of the testing and quality assurance (QA) process in game development, which has traditionally relied on human testers manually replaying level scenarios to identify and report bugs. These manual walkthroughs often follow a logical sequence of actions, representing possible ways for players to solve a level. In our approach, we represented these sequences as sets of steps—for example, in natural language something such as "Move to a platform, pick up an object, advance to the exit, all while avoiding

II Congreso Español de Videojuegos, November 09–10, 2023, Madrid, Spain

*Corresponding author.

pabgut02@ucm.es (P. Gutiérrez-Sánchez); marcoa@fdi.ucm.es (M. A. Gómez-Martín); pagoncal@ucm.es (P. A. González-Calero); pedrop@fdi.ucm.es (P. P. Gómez-Martín)

0000-0002-6702-5726 (P. Gutiérrez-Sánchez); 0000-0002-5186-1164 (M. A. Gómez-Martín); 0000-0002-9151-5573 (P. A. González-Calero); 0000-0002-3855-7344 (P. P. Gómez-Martín)

enemy attacks, and ensure this sequence can be completed with the player losing less than half of their health points", providing a blueprint for player progression within the game environment.

Our objective was to equip NPCs with the capability to execute these logical sequences autonomously, enabling them to test game levels even in the presence of evolving game content or alterations to level logic—a common occurrence in game development.

While static or deterministic game environments allow for the use of pre-recorded human tester traces [2], the sensitivity of such traces to environmental changes or modifications poses clear limitations. These alterations can be subtle, such as minor adjustments to level platform placement, and yet potentially trigger incorrect outcomes in the execution traces, despite negligible impact on overall playability.

On the other hand, these changes are not confined to just the game environment—they can also affect the NPCs themselves. For instance, all enemies of a certain type may share traits like speed and strength, as well as preset behaviors. However, during the development of the game, tweaking these behaviors to fit the gameplay of specific levels can unintentionally make other levels harder or easier. This highlights the need for robust and adaptable mechanisms that can handle subtle shifts in the environment when conducting regression tests.

To address these challenges, prior research has explored the application of artificial intelligence (AI) techniques, including reinforcement [3] and imitation learning [4], or even blends of these strategies incorporating control structures like behavior trees (BTs) [5], to generate autonomous players for quality control testing. Although promising, these approaches often face overfitting issues or lack accessibility to non-technical project members like game designers and QA teams.

In our previous research, we described an approach that leveraged reinforcement learning (RL) guided by formal task specification language descriptions (Linear Temporal Logic, LTL [6]) to develop autonomous agents for testing. We provided a 3D experimentation environment within Unity3D [7], equipped with elements from stealth games, and integrated it with open-source machine learning libraries, notably ML-Agents [8]. However, our focus was solely on generating test agents from design specifications, laying the groundwork for future automation and streamlining of testing and training procedures.

Building upon this foundation, this paper introduces "AI Behavior Graphs," a visual toolkit that simplifies the creation of behavior specifications for NPCs, offering an intuitive graphical interface enabling designers to conveniently articulate their expectations for player-NPC interactions within a game level. These specifications are then automatically translated into both LTL and Rabin automata [9], which is a necessary step for the dynamic generation of reward functions during agent training.

This contribution streamlines NPC development and democratizes RL-based methodologies, making them accessible to a wider audience encompassing game designers and quality assurance teams. Additionally, our approach emphasizes the capability of utilizing these AI agents for playtesting game levels, ensuring ongoing validation of designer expectations throughout the development cycle. In doing so, it enriches the game design process and offers a promising path toward comprehensive automation of testing procedures in the gaming industry.

The remainder of the paper follows this structure: Section 2 details the proposed approach enabling designers to articulate potential solutions for levels, while Section 3 provides a high-level overview of the formal methodologies employed in this study. Section 4 presents a

comprehensive description of the toolkit and its graphical editor, complete with an illustrative example showcasing the workflow of a designer compiling a level specification. Lastly, we conclude with our final remarks and outline future research directions.

2. Temporal Logic Specifications for Regression Testing

In this section, we outline the perspective of the designer, specifying the particular information and format that designers should provide us with to enable us to effectively train an agent to emulate the player and navigate the level according to the designer's intended solutions.

2.1. Context

When game designers create a video game level, they usually have a particular solution in mind for how they expect players to complete it. For instance, in an adventure game level:

1. The player starts in a dark dungeon and must find a torch to light their way and exit this first chamber.
2. After exiting the chamber, they encounter an area being patrolled by a set of enemies. This area contains a hidden switch that needs to be activated in order for the player to unlock the level door.
3. The player must figure out a way to approach the switch without being detected by the enemies and activate it.
4. Lastly, they may access treasures, and exit the level.

It's important to note that these specifications use a high-level and non-specific vocabulary, allowing for multiple ways to satisfy them. For instance, how the player defeats the level's enemies guarding the doors is irrelevant as long as they deal with all of them.

Once a plan for completing the level is established, it can be used to create a quality assurance test within a test suite. Testers can then follow these steps to verify if the level can be completed as intended. These checks fall under the category of "regression testing": sets of tests ideally conducted regularly to ensure that changes made during development do not disrupt the specification and that it remains feasible to meet the designer's requirements.

However, as development progresses and new content is added, maintaining comprehensive coverage of all regression tests becomes impractical. This is where automatic testing methodologies come into play to streamline the process and enhance product coverage and quality control. From a designer's perspective, the goal is to specify planned level solutions in a standardized way and generate periodic reports to check compliance at any given point in time.

2.2. Original Methodology

The tools developed in this paper build upon our previous work's initial workflow. In this workflow, designers can specify how a level's resolution process should unfold using pseudo-natural language combined with logical operators. This specification then serves as the basis for automatically generating an NPC (Non-Player Character) tasked with fulfilling, to the best of its ability, the requirements outlined in the task description.

Behind the scenes, this automated NPC generation process leverages techniques from the realm of deep reinforcement learning, outlined in Section 3. This approach adds a layer of abstraction, effectively eliminating the requirement for expertise in statistical learning when training a bot to meet the specified criteria. This development holds significant relevance not just for designers but also for machine learning practitioners in the field of video games. It addresses the issue of crafting intricate and often opaque reward functions, which implicitly dictate the behavior of the agent—a problem commonly known as "reward engineering." Instead, our approach automatically generates these reward functions using clear expressions based on domain knowledge provided by experts in the field.

It is important to underscore that our primary focus is not on creating agents that learn tasks entirely from scratch. Rather, we concentrate on translating a designer's preconceived plan of action into a trainable system that can learn to execute it with minimal friction between the designer's intent and the learning process. Additionally, when discussing pseudo-natural language here, we are referring to specifications based on formal logic rather than plain text with such content. Nevertheless, we find it helpful to provide a verbal explanation of the specification to assist readers in understanding the design intention behind it.

3. Temporal Logics for Reinforcement Learning

In recent times, particularly in the field of robotics, there has been a growing interest in using temporal logics (TL) to define reward functions in reinforcement learning (RL) algorithms [10]. Among these logics, those enabling quantitative semantics, termed robustness [11], have gained significant attention, as they quantify how well a state trace adheres to a specification. This robustness metric can serve as a natural reward function, facilitating the definition of sequential tasks, sub-objectives, temporal constraints, and behavioral restrictions.

Signal Temporal Logic (STL)[12] is a widely employed temporal logic language, well-suited for robot control tasks, but its reward generation process applies only to complete trajectories, resulting in sparse rewards and complicating the learning of lengthy or intricate tasks. Alternatively, Truncated Linear Temporal Logic (TLTL)[6] allows formula evaluation over finite trajectories of varying lengths. Early applications of TLTL in RL were limited to a single reward evaluation at the end of training episodes. Later work introduced Finite State Predicate Automata (FSPAs) to capture temporal dependencies and offer dense rewards based on robustness variations between transitions [13], making it a more suitable tool for complex task learning.

We aim to select a logic that combines qualitative (True or False predicates) and quantitative semantics (continuous measure of formula satisfaction). TLTL fulfills these criteria, allowing the incorporation of complex intentions, domain knowledge, and constraints into task specifications.

TLTL formulas are defined over predicates of the form $f(s) < c$, where $f : \mathbb{R}^n \to \mathbb{R}$ and c is a constant, and include operators like $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), and $\bigcirc$ (next), as well as boolean operators ($\wedge$, $\vee$, $\Rightarrow$), with the following syntax:

$$\phi := \top \mid f(s) < c \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \Diamond\phi \mid \Box\phi \mid \phi \,\mathcal{U}\, \psi \mid \phi \,\mathcal{T}\, \psi \mid \bigcirc \phi \mid \phi \Rightarrow \psi \tag{1}$$

As an example of a TLTL formula, the middle part of the specification in Section 2.1, requiring the player to eventually approach a switch and next activate it at some point, while ensuring

that no enemy ever gets too close to the character can be expressed in TLTL as:

$$\phi := \diamondsuit\,(\texttt{switchClose} \wedge \bigcirc\,(\diamondsuit\,\texttt{switchOn})) \wedge \square\,\neg\texttt{enemyClose} \tag{2}$$

While the formula representation of this predicate might seem concise, understanding the logic behind the temporal operators may not be immediately clear upon initial exposure to these languages. In this example, the $\wedge$ operator is used to connect the two fundamental parts of the specification that must be satisfied by the NPC, which will be clarified below.

The first sub-predicate, $\diamondsuit\,(\texttt{switchClose} \wedge \bigcirc\,(\diamondsuit\,\texttt{switchOn}))$, corresponds to the notion that the player must find the switch and then activate it. From a design perspective, this is a requirement that must be achieved at some point in the game to fulfill the specification, which is why this predicate begins with a $\diamondsuit$ operator, indicating the need for the associated predicate to be fulfilled at some future point. A slightly more subtle aspect is the use of the $\bigcirc$ operator to represent that after finding the switch, it must be activated as a next step. In practice, the $\bigcirc$ operator requires the associated condition to be met in the following simulation time step (e.g. if our predicate were solely $\bigcirc\,\phi$ we would expect ϕ to be true in the second time step of the run). This is why it is common to connect it with a $\diamondsuit$ operator to ensure that the specified condition is met at some later time. This allows us to define sequences of conditions that must be eventually met one after another by simply adding blocks of the form $\phi_A \wedge \bigcirc\,(\diamondsuit\,\phi_B)$.

The second sub-predicate, $\square\,\neg\texttt{enemyClose}$, corresponds to the idea that no enemy should get too close to the player. From a logical standpoint, this is a global constraint that is always in effect and can be modeled with a $\square$ operator, followed by what we want to always hold true: that no enemy is within a certain safe distance from the player. $\square$ operators are particularly useful for modeling level constraints like always having a certain item equipped or never allowing health points to drop below a certain threshold.

Going back to TLTL's quantitative semantics, or how well a sequence of states adheres to a specification, the degree of robustness of a trajectory with respect to a specification ϕ is represented by $\rho(s_{t:t+k}, \phi)$, where $s_{t:t+k}$ denotes a state sequence. Positive values indicate adherence to ϕ, while negative values signify non-adherence. Despite some limitations, the robustness definition proposed by [6] was used in our previous work due to its ease of implementation, but other feasible alternatives can be found in the literature [14].

The degree of robustness based on a specification ϕ can be employed with reinforcement learning, where it serves as a reward function. This reward function awards robustness at the end of an episode, aligning with natural language and simplifying the specification process. However, this results in sparse rewards, which hinders learning complex tasks.

To address this issue, [13] and [15] propose generating Finite State Predicate Automata (FSPAs) from TLTL predicates to capture sequences of actions needed to satisfy a task. FSPAs consist of transitions equipped with TLTL predicates, triggering when their robustness is positive and maximal among the set of currently positive transitions. This conversion, which is always possible to perform as shown by [16], allows step-based rewards rather than episodic ones based on the local variations in robustness for the current automaton state. While various reward allocation algorithms exist, we adopt an adaptation of the one introduced by [15] in which we opt to build the FSPAs by means of a structure called a Rabin automaton [9]; this is a finite state machine which makes its runs on infinite words and performs reasoning about them.

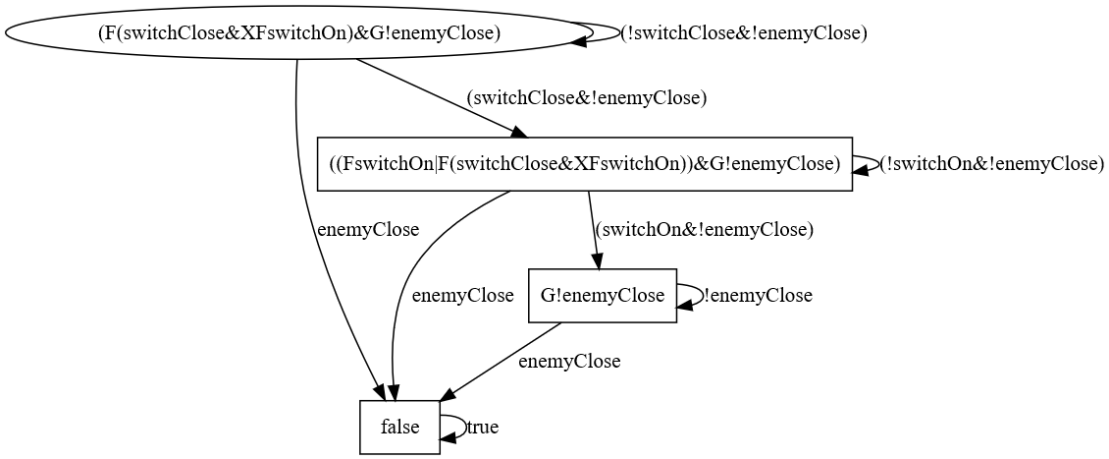


Figure 1: Rabin Automaton-based FSPA for the specification in Formula 2.

Continuing with the previous example, the FSPA shown in Figure 1 allows us to represent the specification from Formula 2 using 4 states and the transitions from the diagram. Please note that in this format, the initial state is represented by an oval and we have $F \equiv \Diamond$ (eventually), $G \equiv \Box$ (always), and $X \equiv \bigcirc$ (next). It is important to underscore that, although the automaton in the figure may be understandable with a little patience, the transitions and labels it generates are generally not trivial and highlight the structural complexity that can unexpectedly arise behind seemingly simple specifications.

4. The AI Behavior Graphs Toolkit

As highlighted in the previous section, while LTL is valuable for automatically generating reward functions tailored to human specifications, manually crafting these specifications can quickly become a cumbersome task. Hence, we introduce The Unity AI Behavior Graphs Toolkit, a dedicated tool engineered to simplify the creation and training of LTL-driven NPC behaviors in Unity 3D. This toolkit empowers designers to outline NPC behaviors through an intuitive visual node editor and subsequently generate reward functions for training reinforcement learning agents, streamlining the process described in Section 3.

4.1. Creating new Graph Specifications

Once the tool has been added to the Unity project, the designer can start interacting with the graphical behavior window from the engine editor through a dedicated inspector menu option. This opens a window like the one shown in Figure 2, with an empty default graph that includes a blackboard with no variables and the predicate return node. In the graph view, two fundamental elements are present:

- **Blackboard**: Used for storing parameters that can be referenced by nodes in the graph. Parameters are categorized based on their type (`GameObject`, `Vector3`, `String`, etc.).

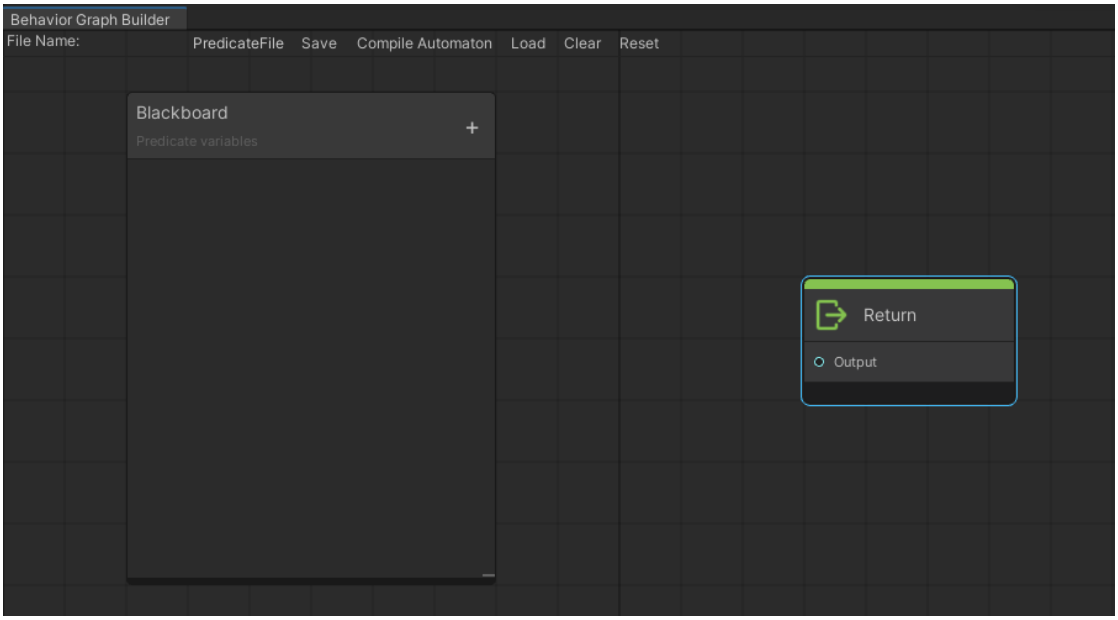


Figure 2: Default Editor Window.

- **Node Editor**: A graphical editor that facilitates the addition, deletion, editing, and management of connections between nodes. To introduce a new node to the graph, the user can right-click in the editor and select the "Create Node" option to access the node exploration panel. This explorer is structured into nested categories, including `Flow` (for core temporal logic operators) and `Conditions` (for basic condition operators). It is also possible to include new categories and nodes, as detailed in Section 4.2.

The input parameters of a node can be other nodes or predicates in the graph, in which case they are represented by input ports to receive a connection through an edge, or blackboard variables. In the latter case, the parameter will be listed in a dedicated section at the bottom of the node along with a drop-down selector and a button with the + symbol. Through the drop-down, the user can select the blackboard variable that will be linked at run-time with the input, while the button allows creating a new variable of the appropriate type on the blackboard.

It is important to note here that while there can be multiple disconnected groupings of nodes in the behavior graph, the only component that will be considered at run-time is the one linked with the special return node (green in color, always one unique node of this type per graph).

As an example of a graph representation that can be created with this tool, Figure 3 depicts a behavior graph along with its blackboard for the specification from Formula 2 that was explained in Section 3. Note that the original predicates have been expressed here by means of more general nodes (e.g. `switchClose` becomes `targetClose`). This sort of tree representation is often more intuitive than the original formula, and easier to understand for non-technical profiles.

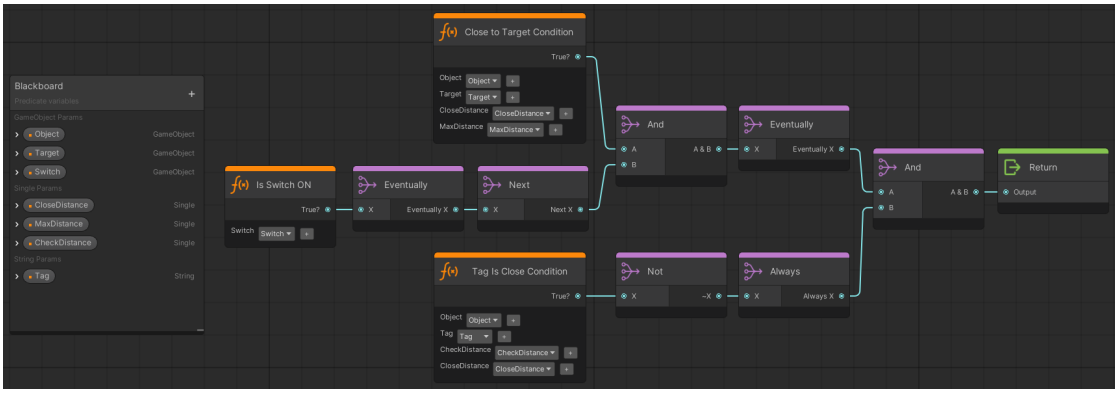


Figure 3: Example editor graph for the specification from Formula 2.

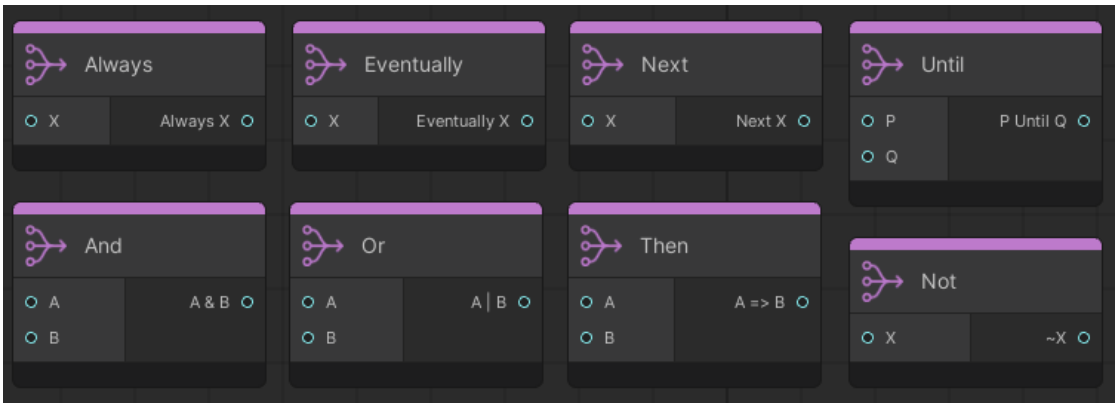


Figure 4: Currently available flow nodes.

4.2. Adding Custom Editor Nodes

The AI Behavior Graphs Toolkit differentiates between two types of nodes based on their role in the specification to be constructed:

- **Flow Nodes** correspond to standard operators in Linear Temporal Logic that take a set of LTL predicates as parameters (eventually, always, until, etc.). Note that this is a closed set of operators and that no other composite nodes are supported to date, meaning that user-created nodes are not allowed to introduce references to other predicates as parameters. A visual overview of the available flow nodes can be found in Figure 4.
- **Condition nodes** correspond to robustness functions in LTL and represent a condition on a set of parameters whose truth value is not given in a boolean format (`true` or `false`) but rather by a numerical value representing its robustness.

Although some default simple conditions are included in the project, it is possible to manually define new condition nodes that can be subsequently included in behavior graphs. To do this, the user must extend the abstract class `TLTLPredicate`, as shown in Figure 5 for an example

```
[Predicate("Conditions/Less Than", "A < B")]
public class LessThanPredicate : TLTLPredicate {
    [InParam("A")]
    public float A { get; set; }
    [InParam("B")]
    public float B { get; set; }
    public override float EvaluateRobustness() {
        return B - A;
    }
}
```

Figure 5: Defining a custom node for $A < B$ predicate.

of a predicate that computes the robustness of $A < B$. The most important function here is `EvaluateRobustness`, which allows returning a numerical value to represent the extent to which the predicate is satisfied at the current moment. While there are no upper or lower bounds for the value returned by the function, it is recommended to keep it within a normalized range of $[-1, 1]$ in order to ensure that interactions with other operators and conditions are consistent and to prevent it from being overwhelmed by the magnitude of nearby predicates.

On the other hand, since a predicate of this type is nothing more than a function that returns a value based on inputs, it is also important to mention these parameters. The input parameters that will be available in the predicate's robustness calculation must be declared in the class that extends `TLTLPredicate` with the annotation `[InParam("paramName")]`, specifying the name by which the user wishes to refer to the parameter from the associated node in the graphical editor. Note that this name does not have to match the variable name in the class.

Finally, for the predicate to be available as a node in the editing tool, it is essential that the implemented class is accompanied by the annotation `[Predicate("path/to/condition", "predicateOutputName")]`. Remembering the structure of the node explorer in the graphical editor, each node can be included in a different category, potentially nested in a chain of subcategories, as in the case of `LessThanPredicate`, which appears under the `Conditions` category (see Figure 6). The first input parameter of the preceding annotation precisely corresponds to this "path" of categories and must be defined manually by the class implementer. On the other hand, the second parameter of the annotation corresponds to the text that will appear in the graphical editor to represent the node's output value, and can be used to further clarify the nature of the condition to the user.

4.3. Compiling Automata

Once a satisfactory graph for an NPC is ready, the next step is to compile it into an automaton for run-time use. In reality, what is generated during the behavior editing process through the visual tool is just a specification in LTL and the metadata for its graph representation, neither of which can be directly used to reward an agent during training. To make it usable in that context, we need to convert the original specification into a structure that allows local reasoning about the predicate, abstracting all dependencies and temporal operators into states where we

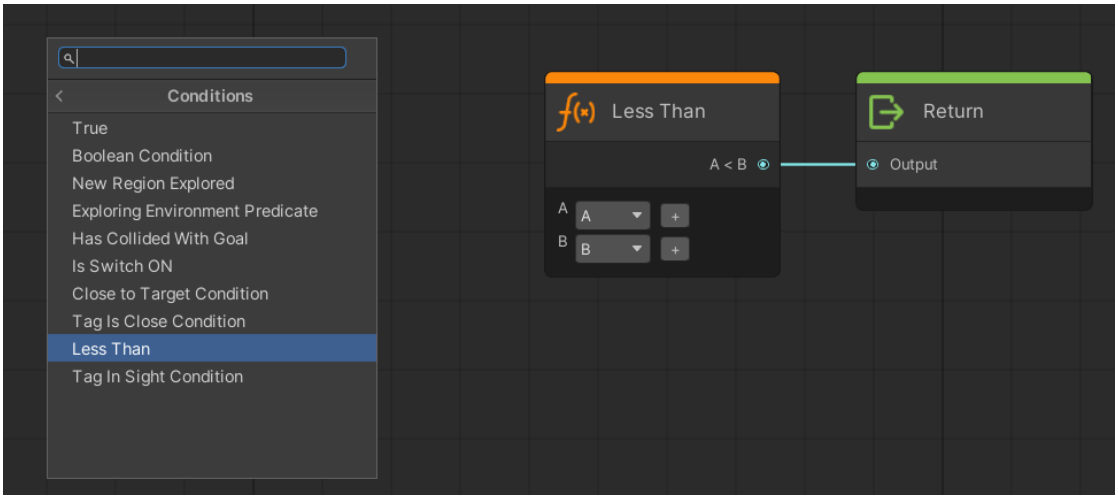


Figure 6: Less Than Node.

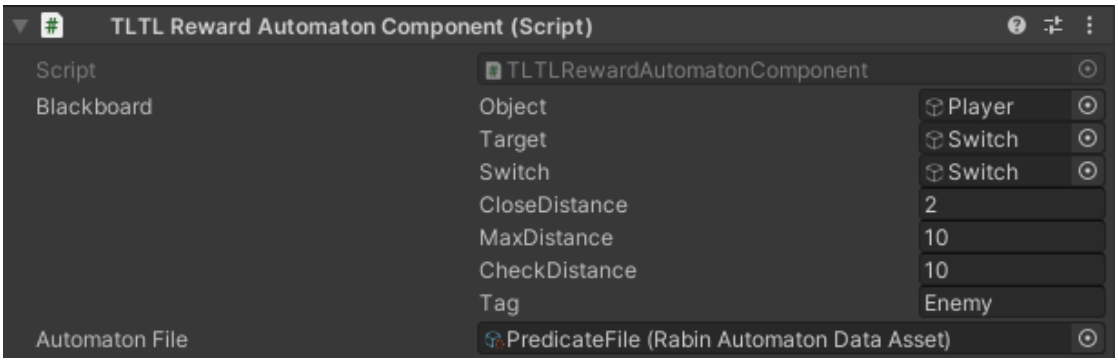


Figure 7: Reward Automaton Component as seen in the Unity Editor.

know our current position in the process. This can be achieved through the use of the FSPAs introduced in Section 3, in our case constructed by means of Rabin Automata.

However, the conversion of a specification written in Linear Temporal Logic into a Rabin automaton poses a complex challenge. To facilitate it, we currently employ an external open-source Java library known as Rabinizer 3 [17], which is executed via a C# Process from the Unity Editor. When selecting the "Compile Automaton" option from the editor menu, the conversion process is triggered. This starts by transforming the current predicate into a format acceptable to the Rabinizer library. Once the output automata is ready, the process proceeds to parse the result into a Rabin Automaton Data Asset, which can be used to perform inference at run-time. Note that the generated asset contains only references to the necessary classes for instantiating different node types in the specification at run-time, as well as the blackboard variables they reference. These variables remain without actual values until assigned to an entity in the scene.

On a different note, the graph in Figure 3 does not accurately reflect the intention we want to convey in the specification. The predicates used are mostly generic (e.g., "object near the target"

instead of "player near the switch") and require specific entities to convey the designer's real intent. To perform this assignment, the toolkit provides the `TLTLRewardAutomatonComponent`, which acts as a bridge between the automaton-format specification asset provided as a parameter, its blackboard variables, and the elements in the scene. This component is typically added to the NPC's game object that should satisfy the specification. An example after assigning the asset compiled from Figure 3 with the necessary input parameters can be seen in Figure 7. Here, some parameters are literals, while others are direct references to game objects.

At run-time, this component proceeds to instantiate the automaton's transitions with specific instances of the predicates to be executed during the game to evaluate conditions and user-introduced references, resulting in what we call a `TLTLRewardAutomaton`. This acts as a data container that does not update itself during execution, delegating this task to other components that can trigger such updates by calling the automaton's `Tick` method. For further details, including possible ways to train an agent to adhere to the specification, please refer to [1].

5. Conclusions and Future Work

This paper introduced "AI Behavior Graphs," a user-friendly toolkit that simplifies NPC development and RL methodologies, making them accessible to game designers and quality assurance teams. While we are confident in the toolkit's capabilities, it is important to note that designers without prior experience in temporal logics may encounter a learning curve. To ensure the practical relevance and broader impact of our toolkit, we are committed to refining AI Behavior Graphs based on real-world feedback from industry professionals. Our goal is to not only make this tool useful for research but also a valuable asset in game development, aligning it with the preferences of game designers. By bridging the gap between research and practical game development, our aim is to create a more accessible and valuable resource for the industry, ultimately advancing the synergy between AI and game development.

Acknowledgments

This work was supported by the Ministry of Science and Innovation (PID2021-123368OB-I00).

References

[1] P. Gutiérrez-Sánchez, M. A. Gómez-Martín, P. A. González-Calero, P. P. Gómez-Martín, Reinforcement learning with temporal logic specifications for regression testing npcs in video games, in: 2023 IEEE Conference on Games (CoG), 2023 (forthcoming).

[2] M. Ostrowski, S. Aroudj, Automated Regression Testing within Video Game Development, GSTF Journal on Computing (JoC) 3 (2013) 10. URL: https://doi.org/10.7603/s40601-013-0010-4. doi:10.7603/s40601-013-0010-4.

[3] J. Bergdahl, C. Gordillo, K. Tollmar, L. Gisslen, Augmenting Automated Game Testing with Deep Reinforcement Learning, in: 2020 IEEE Conference on Games (CoG), IEEE, Osaka, Japan, 2020, pp. 600–603. URL: https://ieeexplore.ieee.org/document/9231552/. doi:10.1109/CoG47356.2020.9231552.

[4] S. Ariyurek, A. Betin-Can, E. Surer, Automated Video Game Testing Using Synthetic and Humanlike Agents, IEEE Transactions on Games 13 (2021) 50–67. URL: https://ieeexplore.ieee.org/document/8869824/. doi:10.1109/TG.2019.2947597.

[5] P. Gutiérrez-Sánchez, M. A. Gómez-Martín, P. A. González-Calero, P. P. Gómez-Martín, Reinforcement Learning Methods to Evaluate the Impact of AI Changes in Game Design, Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment 17 (2021) 10–17. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/18885.

[6] X. Li, C.-I. Vasile, C. Belta, Reinforcement learning with temporal logic rewards, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Vancouver, BC, 2017, pp. 3834–3839. URL: http://ieeexplore.ieee.org/document/8206234/. doi:10.1109/IROS.2017.8206234.

[7] P. Gutiérrez-Sánchez, M. A. Gómez-Martín, P. A. González-Calero, P. P. Gómez-Martín, Liquid snake: a test environment for video game testing agents, in: R. Lara-Cabrera, A. J. F. Leiva (Eds.), Actas del I Congreso Español de Videojuegos, Madrid, Spain, December 1-2, 2022, volume 3305 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3305/paper7.pdf.

[8] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, D. Lange, Unity: A General Platform for Intelligent Agents, arXiv:1809.02627 [cs, stat] (2020). URL: http://arxiv.org/abs/1809.02627, arXiv: 1809.02627.

[9] B. Khoussainov, A. Nerode, Rabin Automata, Birkhäuser Boston, Boston, MA, 2001, pp. 249–328. URL: https://doi.org/10.1007/978-1-4612-0171-7_5. doi:10.1007/978-1-4612-0171-7_5.

[10] H.-C. Liao, A Survey of Reinforcement Learning with Temporal Logic Rewards (2020).

[11] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, S. A. Seshia, Model Predictive Control for Signal Temporal Logic Specification, arXiv:1703.09563 [cs] (2017). URL: http://arxiv.org/abs/1703.09563, arXiv: 1703.09563.

[12] O. Maler, D. Nickovic, Monitoring Temporal Properties of Continuous Signals, in: Y. Lakhnech, S. Yovine (Eds.), Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Springer, Berlin, Heidelberg, 2004, pp. 152–166. doi:10.1007/978-3-540-30206-3_12.

[13] X. Li, Z. Serlin, G. Yang, C. Belta, A formal methods approach to interpretable reinforcement learning for robotic planning, Science Robotics 4 (2019) eaay6276. URL: https://www.science.org/doi/10.1126/scirobotics.aay6276. doi:10.1126/scirobotics.aay6276.

[14] N. Mehdipour, C.-I. Vasile, C. Belta, Specifying User Preferences Using Weighted Signal Temporal Logic, IEEE Control Systems Letters 5 (2021) 2006–2011. URL: https://ieeexplore.ieee.org/document/9309020/. doi:10.1109/LCSYS.2020.3047362.

[15] X. Zhao, M. Campos, Reinforcement Learning Agent Training with Goals for Real World Tasks, arXiv:2107.10390 [cs] (2021). URL: http://arxiv.org/abs/2107.10390.

[16] K. Y. Rozier, Explicit or Symbolic Translation of Linear Temporal Logic to Automata, Thesis, Rice University, 2013. URL: https://scholarship.rice.edu/handle/1911/71687.

[17] Z. Komárková, J. Křetínský, Rabinizer 3: Safraless translation of ltl to small deterministic automata, in: F. Cassez, J.-F. Raskin (Eds.), Automated Technology for Verification and Analysis, Springer International Publishing, Cham, 2014, pp. 235–241.

Chapter 11

Explaining and Clustering Playtraces Using Temporal Logics

Explaining and Clustering Playtraces Using Temporal Logics

Pablo Gutiérrez-Sánchez
Software Engineering and Artificial Intelligence
Complutense University of Madrid
Madrid, Madrid, Spain
pabgut02@ucm.es

Diego Pérez-Liébana*
Queen Mary University of London
London, United Kingdom
diego.perez@qmul.ac.uk

Raluca D Gaina*
Queen Mary University of London
London, United Kingdom
r.d.gaina@qmul.ac.uk

Abstract

This paper addresses the challenge of explaining gameplay behaviours and traces in video games using methods based on linear temporal logics (LTL). Applications for this range from classifying a player's game-style to craft personalised user experiences, to exploring the most significant behaviour patterns within a set of trajectories, particularly in the context of data-driven design and quality control assisted by black-box algorithms. We divide the problem into two complementary tasks. First, to infer a temporal characterisation of a registered play-style by means of a predicate in LTL from a set of representative traces and potential counterexamples. Second, to classify a diverse set of traces into groups in order to identify behavioural patterns within the samples. The first problem focuses on recognising what makes a behaviour unique when compared to others, while the second problem seeks to detect meaningful patterns in groups of players. For the first task, we propose a series of heuristic search methods in the LTL predicate space, such as Monte Carlo Tree Search and Grammatical Evolution. For the second, we introduce a new algorithm that clusters traces based on predicates that split them into cohesive sets, demonstrating how the methods of the first problem can be extrapolated to the latter. Both approaches are evaluated with practical experiments on a 3D third-person stealth game developed in Unity 3D, showcasing how these techniques can be used for analysis. Preliminary results obtained with real player traces provide evidence that these methodologies can support a more comprehensive understanding of observed behaviours.

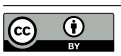
CCS Concepts

• **Theory of computation** → **Linear logic**; **Modal and temporal logics**; • **Computing methodologies** → **Game tree search**; **Discrete space search**; **Randomized search**; **Temporal reasoning**.

Keywords

Temporal Logics, Playtrace Analysis, Game Analytics, Monte Carlo Tree Search, Grammatical Evolution, Gameplay Clustering

*Both authors contributed equally to this research.



FDG '25, Graz, Austria

ACM ISBN 979-8-4007-1856-4/25/04
https://doi.org/10.1145/3723498.3723719

ACM Reference Format:
Pablo Gutiérrez-Sánchez, Diego Pérez-Liébana, and Raluca D Gaina. 2025. Explaining and Clustering Playtraces Using Temporal Logics. In *International Conference on the Foundations of Digital Games (FDG '25), April 15–18, 2025, Graz, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3723498.3723719

1 Introduction

The recent surge of complex black-box systems in artificial intelligence methods has led to a growing interest in devising simple, or at least human-understandable, explanations of the underlying logical processes behind these systems' abstractions. In fields such as healthcare, robot manipulation, or transportation, the design of models that are interpretable and directly tractable by humans has become a focal point for ensuring trust and adoption by real users [1, 24].

This applies to the video game industry as well, where there exists a steadily growing literature on the potential of deep learning for quality assurance (QA), testing, and even the design of bots, Non-Player Characters (NPCs), or artificial players for different areas of interest. It is in QA that these techniques have been most successfully adopted, as the resulting agents are largely proxies for human testers in order to explore environments, look for bugs in the game, or perform systematic regression tests to ensure that certain combinations of actions and interactions lead to the expected results as incremental changes are introduced to other parts of the product. These applications also suffer from a lack of explainability that reduces the trust of practitioners and hinders the clarity of the results gathered in the QA process.

This desire to clarify the rationale behind black box systems is essentially a dual form of another task of interest in the industry, namely the analysis of gameplay and play-styles. Arguably, this problem presents two distinct dimensions depending on the design objective to be pursued. For an already released game, or a game currently under development undergoing beta testing with real players, the first issue of interest is to understand what kinds of interactions are likely to be encountered in practice. This can take several shapes, among which the following can be highlighted:

- Classifying players on the basis of game profiles, which are not known in advance. For example, after recording the behaviour of dozens of users, we could separate them into those who solve a level in an aggressive way by engaging in hand-to-hand combat against their enemies, and those who prefer to follow a stealth strategy to avoid confrontation.
- Conducting a statistical balance analysis of which strategies work best or worst and under what circumstances. This might include observations about a card or character in a

game being used more frequently by top players, or items that are not equipped as much as expected by design.
- Performing a systematic exploration of game map regions via techniques such as heat maps, to understand the physical distribution of events of interest such as deaths, item uses, or scouted areas.

The second dimension considers the problem of inferring what category or style a player belongs to in real time, either based on their closeness to some preconceived archetype (e.g., wary, explorer, fighter) or on their level of skill and proficiency within the game. This is particularly useful for guiding dynamic difficulty adjustment mechanisms or for customising the user experience to the individual's characteristics.

The former is generally an explorative problem where the focus is on detecting patterns in user behaviour, whereas the latter is more in line with a supervised regression or classification paradigm, although what this adjustment is performed on may be a vague notion (e.g., the skill level of a player, which can be complex to quantify objectively).

In this paper, we will focus mainly on this first exploratory question and, more specifically, on the inference of meaningful temporal patterns in game traces. For this purpose, and momentarily departing from the game domain, some of the most typically employed models include finite state machines and temporal logics, which not only possess a substantial set of theoretically desirable properties, but also come with a syntax and structure that are generally simple to understand, alongside human-friendly semantics. The latter turns them into mechanisms that are particularly well conditioned to construct interpretable models in the tasks described above.

Now, this problem, by its inherently exploratory nature, is ill-conditioned in that the notions of what is or is not interpretable, as well as what provides useful information and what does not, have no self-evident or theoretically straightforward numerical definition. As an example, a concise model that classifies a set of game traces correctly could be the trivial model that simply classifies them all as positive (a logical true statement), but this does not provide any useful information that explains what exactly is happening in the game. However, an overly explanatory model that details every step of the interaction could become overwhelming and hard to grasp; hence, the aim here is to strike a compromise between correctness and complexity within our explanations.

With this in mind, we begin the paper by summarising related work on similar problems in Section 2, after which we move on to describing the general concepts and establishing the notions and notations necessary to understand the rest of the article, with concise introductions in Section 3. We then elaborate on our first contribution, which involves formulating formal definitions of two exploratory problems of interest in video games. The first, described in Section 4, aims to characterise what differentiates a set of game traces from others through models that focus on temporal properties that are fulfilled in those examples but not in the provided counterexamples (e.g. one group might be distinguished by always defeating enemies and then opening a chest, while the other might try to avoid foes and go straight for the loot). The second, presented in Section 5, given a set of game traces from different potentially diverse play sets, addresses the problem of dividing them into well-structured groups that share similar behaviours. In turn, we differentiate them from each other, similar to the case of a clustering algorithm, but with a special focus on the temporal part.

For the first problem, we propose a set of predicate space search methods mostly based on heuristic techniques such as Grammatical Evolution or Monte Carlo Tree Search, in what is known as Linear Temporal Logic (LTL), a formal language for specifying properties of a system over time. For the second problem, we introduce a new algorithm that groups traces based on predicates that divide them into cohesive sets separated from each other. These methods are further detailed in the same sections where each problem is defined.

Subsequently, Section 6 reports on experiments conducted on a stealth game testing environment developed in Unity 3D, where the above algorithms are tested and contrasted on real player traces. Section 7 concludes the paper with discussions and future work.

2 Related Work

The problem of learning temporal properties from positive and negative examples has been widely discussed both in the field of robotic control and in more general contexts within various works in the literature. Some of the most relevant contributions in this field come from Grinchtein et al. [29] and Biermann et al. [3] on the synthesis of finite state machines from demonstrations, and again Grinchtein et al. [9] on the automatic learning of time-invariant properties of complex systems. More recently, Roy et al. [26] propose a symbolic approach supported by SAT Solvers for the task of learning predicates in Temporal Logic from positive examples only, which is known as the one class classification (OCC) problem, while Raha et al. [23] propose a set of techniques used to learn segments of predicates in linear temporal logic in a more general fashion.

On the other hand, the problem of automatically recognising play styles in video games enjoys an extensive literature base, both in archetypal style classification and in regression on skill level. Bontchev et al. [4] propose a model for recognising play styles using linear regression techniques on performance metrics in different subtasks in a game with adaptive difficulty. Valls-Vargas et al. [28] detail a new approach where it is suggested that it may be more effective to infer play styles as a dynamic and time-varying property, and suggest a player-modelling framework that exploits this characterisation. Ingram [14] makes use of unsupervised deep clustering strategies such as long short term memory (LSTMs) auto-encoders to convert game traces into points in a latent space of lower dimensionality on which to later apply clustering algorithms.

Similar deep learning techniques have also been applied to various unsupervised clustering problems of game trajectories and general systems, several of which also make use of temporal auto-encoders as a prior dimensionality reduction step [18, 30]. Deep learning has also been used in supervised problems in order to identify which player category a user belongs to by combining latent spaces with datasets of game trajectories labelled with their matching style [27]. While effective for building broad behavioural archetypes, these often lack the granularity to capture specific behavioural flows within categories. Our approach provides a complementary perspective by focusing on the general temporal characterization of groups of play-traces.

Lastly, a similar problem has been studied in the field of inverse reinforcement learning (IRL) to learn reward functions for

RL agents based on positive demonstrations. Kasemberg et al. [15] learn LTL formulas from demonstrations of behaviour trajectories in Markov Decision Processes (MDPs), although they do so on the assumption that the underlying system's internal mechanisms are accessible. In turn, Hasanbeig at al. [13] introduce an algorithm for inferring automata describing high-level goals for RL agents. It is worth mentioning that there are multiple additional works that take specifications expressed in temporal logics as a starting point to train agents by reinforcement, deriving learning mechanisms in which the customary process of reward engineering is avoided, in favour of more accessible and interpretable languages [11, 16, 31].

3 Preliminaries

In this section, we describe the general concepts and establish the notation to be used in the rest of the paper.

Game Traces. To formally represent the executions of a game, we will resort to the concept of traces defined over a non-empty set Σ of possible game states $s \in \Sigma$ to be considered in the analysis. A trace over Σ is a finite sequence $t = \{s_1, \ldots, s_n\}$, where $s_i \in \Sigma$, $1 \leq i \leq n\}$. In the context of this paper, a trace will correspond to an observation of the game states logged over the course of a play session of a given player in their interaction with a game. In what follows, we represent the length of a trace t by $|t|$. Additionally, we denote by Σ^* the set of all possible traces that can be generated from Σ. Here we speak of states in a deliberately ambiguous sense, and shall define what is understood as game state more precisely after introducing predicates and temporal logics.

Temporal Logics. By Temporal Logics (TLs) we refer to any system of rules and symbols that allows reasoning about propositions in terms of their evolution over time. TLs play an important role in the formal verification of requirements in both hardware and software [6], with Linear Temporal Logic (LTL) [17] being one of the most widely adopted examples of these logics. LTL formulas (which we shall also refer to as specifications in what follows) are defined based on a series of core logical and temporal operators and, more importantly, predicates of the form $f(s) > 0$, where $f : \Sigma \to \mathbb{R}$ represents a function applied to the system's state $s \in \Sigma$. The latter are the ones that are truly responsible for incorporating a certain knowledge base into our specifications, as well as conditions related to the world and context in which we operate, such as "being close to a goal" or "an enemy having a high awareness of the player." These conditions are highly domain-dependent and consequently, for each game we contemplate, we will inevitably encounter the need to define a specific set of predicates that will determine the expressiveness of our tasks. In what follows, we shall assume that system states are given directly through arrays of real-valued evaluations of the original game state, i.e. $\Sigma \subseteq \mathbb{R}^n$. An LTL specification ϕ adheres to the following syntax:

$$\begin{aligned} \phi := \top \mid f(s) > 0 \mid \neg\phi' \mid \phi'_A \wedge \phi'_B \mid \phi'_A \vee \phi'_B \mid \\ \phi'_A \Rightarrow \phi'_B \mid \mathcal{F}\phi' \mid \mathcal{G}\phi' \mid \phi'_A \,\mathcal{U}\, \phi'_B \mid \mathcal{X}\phi' \end{aligned} \tag{1}$$

Here, $\top$ is the True boolean constant, $f(s) > 0$ is a check on a domain function on the system state, $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction) and $\Rightarrow$ (implication) are Boolean connectives, and $\mathcal{F}$ (eventually), $\mathcal{G}$ (always), $\mathcal{U}$ (until), and $\mathcal{X}$ (next) are temporal operators. The above grammar is complete in that it is capable of generating any LTL predicate.

Quantitative Semantics and Progress. All of our algorithms rely on the notion of progress of a trace with respect to a specification, which we will briefly introduce in this section. Intuitively, we are interested in defining a function that takes as inputs a trace and a predicate in LTL, and computes a real value indicating how well the trace has complied with the corresponding specification. When the predicate is fulfilled throughout a run $t \in \Sigma^*$ we shall say that the trace models the specification and write $t \models p$ or, alternatively, that t is an example of p. In the opposite case, we will say that t is a counterexample of p, and denote it by $t \not\models p$. Ideally, we would expect our fitness function to be positive if and only if the evaluated trace models the predicate, to be negative for counterexamples, and to report the appropriateness of this satisfaction or rejection by means of its magnitude; that is, the larger its absolute value, the greater the conformance or violation of the trace to the predicate under consideration, for the positive and negative cases, respectively. A function that fulfils these properties is commonly known as a robustness function or qualitative semantics [22].

In the case of the grammar from Equation 1, here we define the robustness function $\rho_s(\phi) := \rho(s, \phi)$ that outputs:

$$\begin{aligned} \rho_s(\top) &= 1 \\ \rho_s(\neg\phi) &= -\rho_s(\phi) \\ \rho_s(f(s) > 0) &= f(s) \\ \rho_s(\phi_A \wedge \phi_B) &= \min(\rho_s(\phi_A), \rho_s(\phi_B)) \\ \rho_s(\phi_A \vee \phi_B) &= \max(\rho_s(\phi_A), \rho_s(\phi_B)). \end{aligned} \tag{2}$$

The above expressions only consider predicates that do not include temporal operators or, in other words, that can only be evaluated over a single time instant and not over complete traces (hence the overloaded notation for state instead of trace). While a number of quantitative semantics exist in the literature that enable the evaluation of goodness-of-fit to predicates including temporal operators, in this paper we shall adopt the automaton construction from [11].

Any predicate in LTL can be transformed into a Finite State Predicate Automaton (FSPA) by means of various different procedures [2]. A FSPA is a finite state automaton that processes a system trace step by step and advances through its different states until it decides whether to accept or reject the input, with the particularity that its edges are equipped with propositional logic predicates and a quantitative semantics. At each step of the execution, the predicates associated with the outgoing edges from the current state of the automaton are evaluated over the next state in the trace, and the most robust positive transition is chosen according to the rules of Equation 2. Whether or not an FSPA constructed from a predicate accepts or rejects a trace (i.e., whether or not it ends up in an acceptance state) is equivalent to deciding if the trace does or does not model the predicate at hand. It is worth noting that an FSPA can be based on different types of automata, as it is simply an abstract construction, with the most common ones being deterministic Rabin and Büchi automata.

An advantage of using an auxiliary automaton to analyse traces is that it makes it possible to calculate the progress of a trace in the specification, measured as the distance advanced in the structure towards the nearest acceptance condition. This is generally simpler

to compute than other quantitative semantics of temporal predicates in LTL that do not rely on such constructs, and also easier to interpret, as it always results in a value between 0 and 1 that intuitively denotes what proportion of the specification has been successfully satisfied.

Let us express this notion formally. Let $t = \{s_0, \ldots, s_n\}$, $s_i \in \Sigma$ be a system trace, ϕ the predicate in LTL containing our specification, s^t_{FSPA} the state of the corresponding automaton reached after processing t sequentially, and D_{FSPA} the maximum distance between two nodes within the automaton. By now terming the minimum distance between the final state of the automaton and an acceptance state as $d_{\text{FSPA}}(s^t_{\text{FSPA}})$, we define the progress attained in the trace $\rho_{\text{FSPA}}(t)$ as:

$$\rho_{\text{FSPA}}(t) = \frac{D_{\text{FSPA}} - d_{\text{FSPA}}(s^t_{\text{FSPA}})}{D_{\text{FSPA}}} + \frac{1 + \max_{e \in impEdg}(\rho_{s_n}(\phi_e))}{D_{\text{FSPA}}}. \tag{3}$$

The second term in the expression represents the distance progressed in the current state to advance to any adjacent state of the automaton that is closer to an acceptance condition (here, $e \in impEdg$ are edges that lead to states improving the progress metric in the specification).

As an example, if we define a predicate $\mathcal{F}(\texttt{enemyDefeated} \wedge \mathcal{X}(\mathcal{F}(\texttt{itemCollected})))$, the resulting automaton would have three states, one for "enemy not yet defeated", one for "enemy defeated but item not yet collected", and a last one for having completed both goals, with $D_{\text{FSPA}} = 2$. If the enemy has just been defeated and the player is still far from the item, we would likely observe that $\rho \approx 0.5$, as the current distance to the next acceptance state is 1, which is halfway through the automaton size.

4 Inferring Temporal Characterisations of Play-Traces

In this section, we present our algorithms for learning temporal predicates to characterise play styles through a set of examples and counterexamples. We begin by formally defining the problem at hand.

Let $\mathcal{P}_g$ be the (possibly infinite) set of LTL predicates that can be generated from an LTL grammar g. We define a trace evaluator σ as a function that takes a predicate $p \in \mathcal{P}_g$ and a real-valued trace of length $n \in \mathbb{N}$, $t = \{s_1, \ldots, s_n\}$, in a given trace space Σ^* and outputs an array of fitness values representing how well the predicate models the trace according to the function's criteria in a series of dimensions. More formally, if $m \in \mathbb{N}$ is the number of fitness dimensions (or alternatively, the number of metrics that are computed for each trace), this is a function with signature $\sigma : \Sigma^* \times \mathcal{P}_g \to \mathbb{R}^m$:

$$\sigma(t, p) = (\sigma_1(t, p), \ldots, \sigma_m(t, p)),$$

where $\sigma_i(t, p) \in \mathbb{R}$, $0 \leq i \leq m$, is the i-th fitness value of the trace t with respect to the predicate p.

This allows us to convert a set of traces into a real vector representation that can be used to compare traces and predicates with each other according to a set of user-defined heuristics. On top of this, we can define additional functions to aggregate the results of each evaluator considered in the analysis into a single goodness value accounting for evaluators' outputs across sample sets (usually a combination of a positive examples set and a negative one). A typical aggregation function has the signature $\gamma_\sigma : (\Sigma^*)^* \times (\Sigma^*)^* \times \mathcal{P}_g \to \mathbb{R}$.

With these notions in mind, we are ready to formally state this section's problem.

Problem 1. Given a set of positive traces or examples P, a second set of negative traces or counterexamples N for a game, a list of trace evaluators σ_i, $i = 1, \ldots, m$, and a metrics aggregator γ_σ, learn an LTL predicate ϕ such that: (1) $t \models \phi, \forall t \in P$; (2) $t \not\models \phi, \forall t \in N$; and (3) $\gamma_\sigma(P, N, p) >= \gamma_\sigma(P, N, p'), \forall p' \in \mathcal{P}_g$.

The first two conditions indicate that we aim to find predicates that fit the considered examples and counterexamples, while the third one states that we are also interested in the chosen predicates being optimal with respect to the established metric aggregator. In practice, we will use approximate strategies to address this problem, so the above conditions will not always be met exactly.

4.1 Brute Force Tree Expansion

The first search method we will consider to approximate Problem 1 can be seen in Algorithm 1, and is a brute-force approach in which all possible solutions within certain constraints are explored in search of the optimal one in terms of fitness.

First of all, it is to be noted that, from this point onwards, whenever we speak of a grammar, we will assume that we are working with one in Backus-Naur form (commonly BNF), a meta-syntax notation for context-free grammars, often used to describe the structure of language specifications. We will also assume when referencing clauses representing functions over the state of the system ($f(s) > 0$ in Equation 1) that the grammar has been extended to represent all variables of interest in the context where the language is used. This implies that the $f(s) > 0$ symbol can be expanded to any of the possible literals recorded in a game as an implicit rule.

That stated, the brute-force algorithm starts from the initial node of the grammar considered, and builds a production tree $\mathcal{T}_g$, in which each node is expanded according to all the possible derivations allowed from it. Since this tree is usually infinitely deep (one need only consider the recursive rules that reference a predicate within another), here we opt to prevent this construction from exceeding a maximum depth specified by the parameter $D \in \mathbb{N}$. Once the pruned tree is available, it is possible to perform any traversal of its leaves, collecting all the predicates that have been completely expanded within the imposed limit. For each of the predicates gathered in this way, it is enough to apply the chosen metrics aggregation function, and keep the individual with the best value according to this heuristic.

Algorithm 1 Brute Force Tree Expansion with Fixed Depth D.

Generate a production tree $\mathcal{T}_g$ from input grammar g, allowing a maximum depth of D.
For each leaf node in the tree that results in a fully expanded LTL predicate p, compute $\gamma_\sigma(P, N, p)$.
return $p \mid \gamma_\sigma(P, N, p)$ is maximal among terminal nodes.

4.2 Grammatical Evolution Search

Grammatical Evolution (GE) [21] is a type of genetic programming (GP; [19]) that evolves solutions by generating computer programs

or expressions, guided by a predefined grammar. A common tool used in GE is precisely the Backus-Naur Form grammar, which provides a formal way to specify the syntactical structure of the programs or expressions to be evolved. GE encodes solutions as sequences of integers, which are then mapped to syntactically correct programs using the BNF grammar.

The adaptation of our problem to a grammatical evolution model is straightforward by setting up the search grammar we are interested in alongside the heuristic function given by the corresponding metric aggregator, and is relatively simple to implement with genetic programming libraries such as the MOEA Framework [12].

4.3 Grammar-Based Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a highly-selective best-first search method that is often used for decision-making. This algorithm balances between exploitation and exploration to balance the search tree growth towards the most promising parts of the search space. This iterative method combines a tree policy (e.g. UCB1 [8]) with Monte Carlo simulations or *rollouts*, which are trajectories sampled at random in the decision space. For more information about this algorithm and its variants, please refer to [5].

In our particular case, we model the search as a sequential decision problem on which production of the reference grammar to apply at each step of a derivation. That is, starting from a first state given by the initial symbol of the grammar, in each run of the MCTS we are interested in first deciding which of the symbols of our current state we wish to expand and, secondly, which of its production rules to use to perform this expansion. As MCTS heuristics, we make use of the metric aggregation function common to all the algorithms in this section, assuming a value of 0 for predicates that have not yet been fully expanded. This means that only rollouts that result in final states can yield useful information.

Algorithm 2 Grammar-Based MCTS.

Initialise base predicate s_p state with the base symbol from the chosen search grammar.
while s_p not fully expanded **do**
  Apply MCTS to s_p with allocated budget to select the seemingly best symbol to expand from the set of unexpanded symbols in s_p and production rule from the given grammar.
  Update s_p after applying the suggested expansion.
end while
return s_p.

5 Tree Clustering of Play-Traces

The second problem to be undertaken in this paper is to propose a classification or clustering approach to distil general behavioural patterns within samples from a potentially diverse set of traces. This is a purely exploratory analysis problem, and as such more complex to define than the task in the previous section. In this case, we shall frame it in terms of looking for predicates that are capable of "separating" traces as much as possible in the sense that the sets of samples that do and do not satisfy the predicate are cohesive and at the same time distant from each other.

More specifically, starting from the previously defined trace evaluators and some form of standard clustering metric, our problem is to construct a predicate that seeks to optimise this metric for the partition given by the sets of traces that it accepts and rejects, respectively.

More formally, given a predicate $p \in \mathcal{P}_g$ and a set of traces T, we define the sets $C^p_{\models}(T) = \{t \in T, t \models p\}$ and $C^p_{\not\models}(T) = \{t \in T, t \not\models p\}$ of traces accepted and rejected by that predicate. By now applying a trace evaluator σ on each member of these clusters we obtain two new sets of individuals $\sigma(C^p_{\models}(T)), \sigma(C^p_{\not\models}(T)) \subset \mathbb{R}^m$, on which it is now possible to apply a clustering metric to heuristically evaluate how good the partition of the samples into these two sets is. With this, we can formally formulate our second problem.

Problem 2. Given a set of unclassified traces or examples T, a list of trace evaluators $\sigma_i, i = 1, \ldots, m$, a clustering quality score function q, and a quality threshold $\alpha > 0$, learn a sequence of predicates $(p_j)_{j=0}^n$ such that:

$$P_j = C^{p_{j-1}}_{\not\models}(P_{j-1}), N_j = N_{j-1} \cup C^{p_{j-1}}_{\models}(P_{j-1}), P_0 = T, N_0 = \emptyset, \quad (4)$$

$$q(\sigma(C^{p_j}_{\models}(P_j)), \sigma(C^{p_j}_{\not\models}(P_j)), \sigma(N_j)) >= \alpha, j = 0, \ldots, n-1, \quad (5)$$

$$q(\sigma(C^{p_n}_{\models}(P_n)), \sigma(C^{p_n}_{\not\models}(P_n)), \sigma(N_n)) < \alpha. \quad (6)$$

Intuitively, we are describing an iterative task. We start from a set of traces T to be partitioned based on predicates p_j that explain a subset of samples with sufficient confidence $\alpha > 0$. In the first iteration, we are simply trying to find a predicate p_0 that splits T into two cohesive sets that are sufficiently separated from one another according to our clustering criterion in the real space induced by our selected metrics. N_1 will then become the set of traces that were modelled by p_0 and that as such do not need further classification, while P_1 becomes populated with the traces discarded in the first round. The next iterations follow this pattern, with the subtlety that we keep expanding N_j with the traces modelled by the sequence of splitting predicates, and we then try to find subsequent predicates in such a way that the groups $\sigma(C^{p_j}_{\models}(P_j)), \sigma(C^{p_j}_{\not\models}(P_j))$ and $\sigma(N_j)$ maintain a good clustering score. Therefore, at each iteration we ask ourselves the question "which trace groups can we find that appear to behave differently to all previous explanations?". In other words, we look for predicates that can explain unique and meaningful behavioural subsets, and progressively more specific ones. These iterations continue until it becomes infeasible to find predicates that induce a reasonable partitioning according to the quality threshold.

Algorithm 3 specifies a procedure by which to heuristically and approximately construct a sequence of predicates with the above conditions. Leveraging the constructs developed in the previous section for Problem 1, the tree clustering algorithm aims to model the trace separation steps as predicate searches that optimise the result of the metric aggregator given by:

$$\gamma_\sigma(P, N, p) = q(\sigma(C^p_{\models}(P)), \sigma(C^p_{\not\models}(P)), \sigma(N)). \quad (7)$$

That is, at each step we look for a predicate that induces the best possible separation between the traces we have already explained through previous predicates, the traces that become explained by the new predicate, and the traces that remain to be modelled.

Algorithm 3 Tree Clustering of Trace Set T with Threshold α.

Choose a list of trace evaluators $\{\sigma_i\}_{i=1}^{m}$.
Choose a clustering quality score q.
Choose a search algorithm S_{alg} from Problem 1, selecting $\gamma_\sigma(P,N,p) = q(\sigma(C^p_{\models}(P)), \sigma(C^p_{\not\models}(P)), \sigma(N))$ as metrics aggregator.
$P_0 = T$.
$N_0 = \emptyset$.
$j = 0$.
$p_0 \leftarrow S_{\text{alg}}(P_0, N_0)$.
while $\gamma_\sigma(P_j, N_j, p_j) >= \alpha$ **do**
  $j = j + 1$.
  $P_j = C^{p_{j-1}}_{\not\models}(P_{j-1})$.
  $N_j = N_{j-1} \cup C^{p_{j-1}}_{\models}(P_{j-1})$.
  $p_j \leftarrow S_{\text{alg}}(P_j, N_j)$.
end while
return $\{p_j\}$, sequence of splitting predicates.

6 Experiments

We start by defining the remaining trace evaluators and the aggregation function that we will use for the first set of experiments, which approach problem 1 by means of different methods.

- **Progress**. We shall consider progress $\rho(t,p)$, already defined in Section 3, as our first metric. Note how progress is 1 if and only if $t \models p$.
- **Exploitation**. Let i be the instant where the FSPA associated with the predicate accepts or rejects the trace t; we define exploitation $e(t,p)$ as $e(t,p) = 1 - \frac{|t|-i}{|t|}$, or the proportion of trace steps that are processed before the underlying FSPA reaches an acceptance decision. Intuitively, we are interested in a predicate showing high values of exploitation in positive examples, where this metric can often be interpreted as implying that the predicate contains relevant information about the reference trace.
- **Reward**. We define the reward $r(t,p)$ as the sum of variations in progress for each of the automaton's execution steps, $r(t,p) = \sum_{i=1}^{|t|-1}(\rho_{\text{FSPA}_p}(t^{i+1}) - \rho_{\text{FSPA}_p}(t^i))$. This value allows us to identify segments in which a player is actively striving to achieve a sub-goal through local variations in the robustness of certain state variables.

Let us look at a concrete example of the above metrics. For instance, if we have that $p = \text{SSEQ}[g_1, g_2, g_3]$, where g_i corresponds to the predicate of reaching a certain location in the level, and in a sample trace the player sequentially visits these locations and then immediately terminates the log (note how this is not necessarily the same as finishing the game; instead, this refers to ending the current play trace), we will observe that $u(t,p) \approx \gamma(t,p) = 1$, since the vast majority of the states in the trace have been relevant to the FSPA, and if the player has additionally managed to steadily approach their goals without straying, we will further notice $r(t,p) \approx 1$. This denotes that the trace successfully models the predicate.

Alternatively, if the player continues to perform other tasks on the trace after having completed the visits to the target points from the predicate, we would perceive a reduction of both the exploitation and the reward of the trace, due to the automaton reaching an early state of acceptance, ignoring all subsequent events, with a progress of 1. This is a symptom that, although the trace models the predicate, the latter is probably not sufficiently informative. In one last scenario, if the player only visits the first goal and then finishes the game, then the exploitation would be 1, since the entire trace is processed by the automaton. However, both the progress and the reward would be fairly low as no FSPA acceptance states would have been encountered. This suggests that the predicate is specifying a more extensive behaviour than what is actually being experienced in the game.

In order to evaluate the quality of a predicate for which the sets of examples and counterexamples in the game have been fixed, we define the following aggregation function:

$$\gamma(P,N,p) = \frac{\sum_{t\in P}(e(t,p) + r(t,p) + \rho(t,p))}{|P|} + \frac{|\{t \in P \mid t \models p\}|}{|P|} + \frac{|\{t \in N \mid t \not\models p\}|}{|N|}. \tag{8}$$

In doing so, we aim to maximise the above metrics for examples, as well as the proportion of examples accepted and counterexamples rejected by the predicate under consideration.

6.1 Game Environment and Recorded States

Liquid Snake [10] is a prototype third-person stealth game developed in the Unity3D engine, in which players control the main character to navigate various levels while collecting valuable items and avoiding detection by robotic guards, security cameras and other surveillance mechanisms. Guards follow pre-set patrol routes, but, if a guard detects the player, they will visit the spot where they last saw them and look around for a short time. As long as the guard keeps noticing the player, or senses them in their vicinity, a suspicion meter will gradually build up. If the suspicion meter reaches a maximum level, the guard will enter an aggressive mode and chase the protagonist until they are captured. The player can use the environment to hide, crouch behind mid-height objects to avoid detection by vision cones, and run to quickly dash through sections of the level before the enemies' suspicion gauges can grow too high. Certain levels also require the player to momentarily allow themselves to be spotted by the robot guards and draw their attention to a secluded spot, forcing them to inspect that location while the protagonist advances through the areas the guards have left unprotected in the process.

In this environment, we register a set of variables as robustness functions on the state of the system every 0.1 seconds of a game in progress. In several of them we make use of the notion of proximity of one object to another within a bounded range, defined as:

$$\delta(o_1,o_2,b,B) = \begin{cases} 1 - \frac{||\vec{o_1o_2}||}{b}, & 0 \le ||\vec{o_1o_2}|| < b \\ \frac{||\vec{o_1o_2}||}{b-B}, & b \le ||\vec{o_1o_2}|| \le B \\ -1, & B < ||\vec{o_1o_2}|| \end{cases} \tag{9}$$

where $||\vec{o_1o_2}||$ represents the Euclidean distance between the points of interest o_1 and o_2. In practice, we take $b = 1, B =$ level size. The variables considered are:

- `objectCollected`$_i$: with a value of 1 if the i-th object has been picked up by the player and $\delta(\text{player}, \text{object}_i, b, B)$ otherwise.
- `goalReached`: same as for `objectCollected`, but only taking on a value of 1 when the player reaches the level exit.
- `safeFromEnemy`$_i$: with a value of -1 if the player is visible within the detection cone of the i-th enemy or camera and 1 otherwise.
- `enemySuspectingPlayer`$_i$: with a value of -1 + awareness, with awareness $\in [0, 1]$ being the proportion of suspicion accumulated by the enemy, and 1 when awareness = 1.
- `playerCrouching`: with a value of 1 if the player is crouching and -1 otherwise.

Within the game, we performed trace recordings of several groups of players interacting with a reference level. This environment, depicted in Figure 1, requires the player to collect all yellow items in order to open the door (blocking the exit to the left of the image) and escape. The lower area of the environment is guarded by a security camera that slowly rotates to cover the room in which it is placed, while the upper area houses a guard who patrols around its central structure with several watch-points that he inspects with greater caution. In this level, we recorded 5 different behaviour patterns defined as follows, with 3 traces per play-style, by having a human tester playing the game according to the instructions given bellow:

- **Style A.** Traces in which the player collects the first two objects, does not crouch to hide, and is then detected by a camera.
- **Style B.** Traces in which the player collects the first two objects, attempts to crouch to hide, but is still detected by a camera.
- **Style C.** Traces in which the player collects the first object, crouches to hide, and then proceeds to collect all other objects in order, while avoiding detection by following the patrolling robot from behind.
- **Style D.** Traces in which the player collects the first object, crouches to hide, and then proceeds to collect all other objects. Follows an order inverse to that of the patrolling robot and is forced to hide behind walls. Sometimes detected, but gets away safely.
- **Style E.** Traces in which the player collects the first object, crouches to hide, and then proceeds to collect all other objects. Follows inverse robot order and is eventually spotted and captured before reaching the exit.

6.2 Temporal Characterisations

With these traces and the variables described above, we apply the algorithms from Section 5 to Problem 1 to try to infer predicates that characterise each of the play styles relative to the rest of the styles. That is, if the styles are given by $S = [A, B, C, D, E]$, then we are considering the problems that take $P = S_i$, $N = \bigcup_{j \neq i} S_j$, $i = 1, \ldots, 5$. In all of them we use the metrics aggregator from Equation 8 and the grammar from Equation 1.

The results obtained can be found in Tables 1, 2 and 3. Each of these tables includes, for every style used as a positive example set: the best predicate found according to the metrics, its fitness

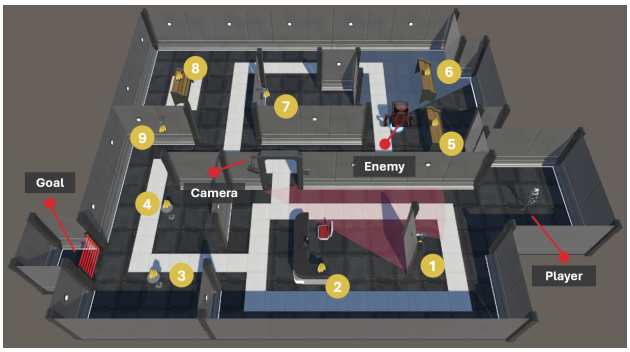


Figure 1: Top View of Reference Environment.

measure normalised to the interval $[0, 1]$ by dividing the value of the metric aggregator by 5, the time taken to reach the solution, and the partial results of each of the terms that make up the search heuristic. Here, $\rho(P)$ refers to progress reached in the predicate by traces in the positive set, $r(P)$ and $e(P)$ are shortened notation for reward and exploitation, respectively, and acc_P, rej_N denote the proportion of traces in the positive and negative examples sets that are accepted or rejected. All of these experiments were run on a PC with 16GB of RAM and an Intel Core i7-9750H, 2.60GHz processor.

In the case of the brute-force algorithm, we choose a maximum expansion depth $D = 6$ to try to strike a balance between expressiveness and computational time. With the given grammar, this leads to process a total of 428,750 terminal nodes in each of the searches; the best one with respect to the chosen fitness metric is reported. As can be noted in the results from Table 1, the time spent is very similar in all cases, at around 6 minutes of computing time. Since this approach systematically explores all possible expansions within the established limit, and these do not vary from one case to another, this result is to be expected and makes it the most stable algorithm in terms of time, although it is important to mention that this would grow exponentially according to the branching factor of the grammar used. The predicates proposed as solutions are fairly modest in this approach given the constraint imposed on the number of derivations of the grammar, and should therefore be considered as broad features of the modelled traces rather than detailed behavioural explanations. Thus, for example, the best individual for the first style is `G(!playerCrouching)`, which corresponds to the property that registered players did not crouch at any time to hide, but does not address the level progress made. Moreover, this very inability to produce complex or highly sequentialised predicates means that the reward values tend to be low: assertions of the type 'globally' result in automata in which it is impossible to increase progress once the state of acceptance has been reached, and eventuality assertions limited to a goal only generate reward in the segment prior to its attainment. In short, this approach produces general explanations that, while optimal in terms of depth, do not yield very fine-grained explanations of the observed behaviour.

In the case of the grammatical evolution algorithm (see Table 2), we choose an initial population size of 100 individuals with individual size of 15, and a total of 100 generations to evolve from this first sample. The time required in this case is markedly shorter than that

Table 1: Brute-force Algorithm Results

Style	Best Individual	Fitness	Time	$\rho(P)$	$r(P)$	$e(P)$	acc_P	rej_N
A	G(!(playerCrouching))	0.81	5m 52s	1	0.03	1	1	1
B	G(!(objectCollected03))	0.76	5m 58s	1	0.03	1	1	0.77
C	F((goalReached)&(playerSafeFromCamera))	0.85	6m 0s	1	0.32	0.99	1	0.92
D	F((objectCollected07)&(goalReached))	0.84	5m 58s	1	0.50	0.99	1	0.69
E	G(F(objectCollected08))	0.76	5m 49s	1	0.34	1	1	0.46

Table 2: Grammatical Evolution Results

Style	Best Individual	Fitness	Time	$\rho(P)$	$r(P)$	$e(P)$	acc_P	rej_N
A	G(!((objectCollected04)|(playerCrouching)))	0.81	13s	1	0.03	1	1	1
B	!(F(objectCollected05))	0.76	1m 18s	1	0.03	1	1	0.77
C	F(objectCollected06)	0.67	3m 45s	1	0.24	0.63	1	0.50
D	F(goalReached)	0.80	31s	1	0.33	0.99	1	0.69
E	F(objectCollected06)	0.68	22s	1	0.24	0.71	1	0.46

of the brute-force algorithm, largely due to the fact that a stagnation state is typically reached quite rapidly when the individuals start to look very similar to each other. Except for time, however, the results obtained with this procedure are generally strictly worse or equal to those of the other algorithms, without this resulting in more interesting predicates from a qualitative point of view.

The MCTS algorithm positions itself as the most successful overall (albeit modestly so), yielding the individuals with the highest fitness in all styles, and with the best metrics in most cases (see Table 3). Here, we take exploration constant $K = \sqrt{2}$, rollout length = 15 and a budget of 300 iterations per expansion. The search capabilities of the algorithm to explore the derivation tree in depth result in predicates that are generally more elaborate than in the other two cases, but at the same time more informative. For instance, in the case of style E, the proposed explanation starts by specifying that the level is not completed, followed by a characteristic notion of order: object 3 is collected first, and then at some point object 8 is collected. This corresponds to the observation that this type of player traverses the top of the level in reverse order to the order in which the guard traverses. In the case of style B, on the other hand, it is indicated that there always comes a point in these traces where the player becomes detected by the camera until the end of the game, which again is a representative explanation of the reason for the defeat of this group. Note how this algorithm generally features significantly higher values for the reward metric, which we intuitively link to the ability of the predicate to capture consistently positive variations in progress across traces, and is a desirable property in individuals exhibiting a good degree of detail.

Something noteworthy here is the time required to run style C, which amounts to about 28 minutes of computation time. Since the budget in this case is specified for each node expansion decision in the derivation tree, and the depth is not initially bounded, this means that it is sometimes necessary to explore a large number of predicates of potentially large size if the rollout leads to a state deep in the structure. Furthermore, it is known that the cost of translating LTL to deterministic automata can grow significantly quickly in practice in both time and size as predicate complexity increases [7], which means that solutions computed at deep nodes may take an unforeseen amount of time. This can be solved by imposing more restrictive conditions on the search, such as limiting the maximum depth of the structure; for the moment, we leave this as future work.

6.3 Tree Clustering

With these same traces, the next step is to apply Algorithm 3 to try to establish a tree structure that explains the most prominent behavioural patterns of the recorded players. In this case, instead of using the grammar from Equation 1, we will introduce a new grammar useful for representing time-ordered sequences of events to demonstrate that our system allows searching over different subsets of predicates driven by more specific questions by simply modifying the reference grammar:

$$\begin{aligned} \phi &:= \texttt{SSEQ}(\texttt{lit-seq}) \\ \texttt{lit-seq} &:= \texttt{lit} \mid \texttt{lit}, \texttt{lit-seq} \\ \texttt{lit} &:= f(s) > 0 \mid \neg(f(s) > 0), \\ \texttt{SSEQ}(l) &= \mathcal{F}(l), \\ \texttt{SSEQ}(l_1, \ldots, l_n) &= \mathcal{F}(l_1 \wedge \mathcal{X}(\texttt{SSEQ}(l_2, \ldots, l_n))) \wedge !l_2 \mathcal{U} l_1. \end{aligned} \tag{10}$$

As can be seen, this new grammar makes use of the SSEQ operator to represent sequences of literals that occur in a strict order. Based on this grammar, we can then use the clustering algorithm to explore what types of behavioural sequences define the traces of T. The result of the analysis can be found in Figure 2. In this case, we have chosen the Silhouette Score [25] as the clustering quality metric, the MCTS-based search algorithm, and $\alpha = 0.51$.

Let us qualitatively analyse the information provided by this trace clustering diagram. From the 15 starting traces, originally introduced without any kind of labelling in the algorithm, the process

Table 3: MCTS Results

Style	Best Individual	Fitness	Time	Steps	$\rho(P)$	$r(P)$	$e(P)$	acc_P	rej_N
A	G(!objectCollected09) & F(objectCollected02 & X(playerSafeFromCamera))	0.89	1m 1s	34	1	0.60	1	1	0.85
B	F(G(!playerSafeFromCamera)) & G(!objectCollected06)	0.88	0m 42s	23	1	0.62	1	1	0.77
C	F(objectCollected09 & objectCollected02 & objectCollected07 & objectCollected04 & X(playerSafeFromCamera) & goalReached & objectCollected06)	0.93	28m 34s	50	1	0.73	1	1	0.92
D	F(objectCollected05 & objectCollected09 & goalReached & objectCollected06 & objectCollected08)	0.87	1m 8s	26	1	0.67	0.99	1	0.69
E	G(!goalReached) & F(objectCollected03 & X(F(objectCollected08)))	0.87	3m 55s	31	1	0.37	1	1	1

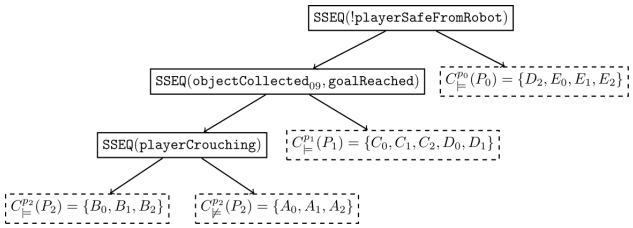


Figure 2: Clustering Diagram for Player Traces.

ends up generating 4 distinct groups based on a total of 3 separation predicates with the newly described grammar. Interestingly, at a preliminary level, the groups established in this way, represented as dashed boxes in the figure, bear a close resemblance to the traces belonging to each of the original play styles, with the only exception being style D, whose traces are distributed between the groups with the E and C styles. Intuitively, this a desirable result, as it suggests that the algorithm is identifying significant patterns that agree with what we consider to be relevant. The new clusters can then be interpreted as:

- Traces in which the player was detected by the robot.
- Traces in which the player collected object 9 and then reached the goal, while not being detected by the robot.
- Traces in which the player crouched at some point, but without collecting object 9 and then reaching the goal, nor being detected by the robot.
- Traces in which player never crouched, nor was detected by the robot, nor collected object 9, and then reached the goal.

As can be seen, each group is defined on the basis of a negation of all previous properties (this follows by construction: had a trace been modelled by a predicate, it would have been discarded in subsequent iterations from that point onwards), plus an additional constraint that refines its characterisation when contrasted with the remaining traces.

7 Discussion and Future Work

The algorithms and experiments described in this paper are not intended as anything beyond initial pilots and proofs of concept to explore what kind of feedback these techniques can generate in quality control and user analysis tasks, as well as what the practical challenges are for introducing them into the design and development loop of a game.

Technically, there remain a number of unresolved issues to be considered for future iterations of the study:

- All the algorithms for Problem 1 are markedly redundant in the searches, for a number of reasons. The first of these is the semantic symmetry of the grammars used, so that, for example, one has that $a \wedge b \equiv b \wedge a$, and the same applies to the disjunction operator. The second, linked to the first, is that at no point are we talking about minimal or simplified predicates according to some system of LTL reduction rules, and so we wind up treating predicates like $\mathcal{F}(\mathcal{F}(p)) \equiv \mathcal{F}(p)$ or $a \vee \neg a \equiv \top$ separately. The same is true for the absence of equivalence rules that would aid in not having to handle pairs of predicates such as $\mathcal{G}(\neg p) \equiv \mathcal{F}(p)$ separately.
- This redundancy significantly affects the time and space cost of the algorithms described here: since the process of translating a predicate in LTL into an automaton can potentially be expensive in both of these ways, calculating automata for equivalent predicates is something that should be avoided as much as possible in the future.
- Moreover, the above algorithms may attempt to evaluate the same predicate more than once due to randomness or construction. The clearest example is in grammatical evolution, where it is possible to generate identical individuals that are then evaluated separately. Similarly, MCTS rollouts can lead to the same terminal nodes. Solution caching is possible to some extent and can alleviate some of the time cost, but in return a potentially significant memory cost may be incurred when the number of cached elements becomes high.

On the other hand, the grammars used for the searches play an integral role in conditioning both the search space, the efficiency of the algorithms and the form of the solutions produced.

- First, a grammar acts as a query in a certain sense, in that by selecting a grammar that generates only predicates with a very specific structure, we are effectively restricting the problem to a more focused and therefore manageable one.

We provide an example of this with Grammar 1 in order to search for predicates that refer to sequences of events across time; while this grammar is not complete in that it does not output every predicate in LTL, it can become even more valuable from a design point of view, by yielding predicates that are simple to interpret and that meet a well-delimited practical question. The same could be done with grammars conditioned on global properties (i.e. events that are true at all times), on logical consequences (events that happen whenever others happen before them), and so on.

- Second, the shape of the grammar can have a significant effect on the quality of the algorithms that employ it, usually by inducing some kind of bias on the probability distribution of the different types of solutions. While beyond the scope of this paper, some early studies on the effect of these properties in the case of grammatical evolution can be found in [20].

Another important point to consider is the metrics used and their effect on the types of solutions obtained. Since we are dealing with unsupervised problems of an exploratory nature, it is complex to define quantitative quality measures that guarantee that the predicates found are truly helpful and insightful; the exploitation, progress and reward metrics are first approximations to understanding which properties might be of interest, but other factors, such as the length or complexity of the predicate, or interactions between different metrics, might be worthwhile depending on how this ill-conditioned notion of fitness is interpreted within this problem.

Acknowledgments

This work was supported by the Spanish Ministry of Science and Innovationnder Grant No.: PID2021-123368OB-I00.

References

[1] [n. d.]. Explainable AI | Royal Society. https://royalsociety.org/news-resources/projects/explainable-ai/

[2] [n. d.]. *Explicit or Symbolic Translation of Linear Temporal Logic to Automata.* Thesis. https://scholarship.rice.edu/handle/1911/71687

[3] A. W. Biermann and J. A. Feldman. 1972. On the Synthesis of Finite-State Machines from Samples of Their Behavior. *IEEE Trans. Comput.* C-21, 6 (June 1972), 592–597. https://doi.org/10.1109/TC.1972.5009015

[4] Boyan Bontchev and Olga Georgieva. 2018. Playing style recognition through an adaptive video game. *Computers in Human Behavior* 82 (May 2018), 136–147. https://doi.org/10.1016/j.chb.2017.12.040

[5] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (2012), 1–43. https://doi.org/10.1109/TCIAIG.2012.2186810

[6] Igor Buzhinsky. 2019. Formalization of natural language requirements into temporal logics: a survey. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, Vol. 1. 400–406. https://doi.org/10.1109/INDIN41052.2019.8972130

[7] Javier Esparza, Jan Křetínský, Jean-François Raskin, and Salomon Sickert. 2022. From linear temporal logic and limit-deterministic Büchi automata to deterministic parity automata. *International Journal on Software Tools for Technology Transfer* 24, 4 (Aug. 2022), 635–659. https://doi.org/10.1007/s10009-022-00663-1

[8] Sylvain Gelly and Yizao Wang. 2006. Exploration exploitation in go: UCT for Monte-Carlo go. In *NIPS: Neural Information Processing Systems Conference Online trading of Exploration and Exploitation Workshop.*

[9] Olga Grinchtein, Martin Leucker, and Nir Piterman. 2006. Inferring Network Invariants Automatically. In *Automated Reasoning*, Ulrich Furbach and Natarajan Shankar (Eds.). Springer, Berlin, Heidelberg, 483–497. https://doi.org/10.1007/11814771_40

[10] Pablo Gutiérrez-Sánchez, Marco Antonio Gómez-Martín, Pedro A. González-Calero, and Pedro Pablo Gómez-Martín. 2022. Liquid Snake: a test environment for video game testing agents. In *Actas del I Congreso Español de Videojuegos, Madrid, Spain, December 1-2, 2022 (CEUR Workshop Proceedings, Vol. 3305)*, Raúl Lara-Cabrera and Antonio José Fernández Leiva (Eds.). CEUR-WS.org. https://ceur-ws.org/Vol-3305/paper7.pdf

[11] Pablo Gutiérrez-Sánchez, Marco Gómez-Martín, Pedro Gonzalez-Calero, and Pedro Gómez-Martín. 2024. A Progress-Based Algorithm for Interpretable Reinforcement Learning in Regression Testing. *IEEE Transactions on Games* PP (01 2024), 1–10. https://doi.org/10.1109/TG.2024.3426601

[12] D. Hadka. [n. d.]. MOEA Framework: A Free and Open Source Java Framework for Multiobjective Optimization (Version 4.5) [Computer Software]. https://github.com/MOEAFramework/MOEAFramework. [Accessed 24-10-2024].

[13] Mohammadhosein Hasanbeig, Natasha Yogananda Jeppu, Alessandro Abate, Tom Melham, and Daniel Kroening. 2021. DeepSynth: Automata Synthesis for Automatic Task Segmentation in Deep Reinforcement Learning. https://doi.org/10.48550/arXiv.1911.10244 arXiv:1911.10244 [cs, stat].

[14] Branden Ingram, Clint van Alten, Richard Klein, and Benjamin Rosman. 2023. Generating Interpretable Play-Style Descriptions Through Deep Unsupervised Clustering of Trajectories. *IEEE Transactions on Games* 15, 4 (Dec. 2023), 507–516. https://doi.org/10.1109/TG.2023.3299074

[15] Daniel Kasenberg and Matthias Scheutz. 2017. Interpretable Apprenticeship Learning with Temporal Logic Specifications. https://doi.org/10.48550/arXiv.1710.10532 arXiv:1710.10532 [cs].

[16] Xiao Li, Zachary Serlin, Guang Yang, and Calin Belta. 2019. A formal methods approach to interpretable reinforcement learning for robotic planning. *Science Robotics* 4, 37 (Dec. 2019), eaay6276. https://doi.org/10.1126/scirobotics.aay6276

[17] Xiao Li, Cristian-Ioan Vasile, and Calin Belta. 2017. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Vancouver, BC, 3834–3839. https://doi.org/10.1109/IROS.2017.8206234

[18] Naveen Sai Madiraju, Seid M. Sadat, Dimitry Fisher, and Homa Karimabadi. 2018. Deep Temporal Clustering : Fully Unsupervised Learning of Time-Domain Features. https://doi.org/10.48550/arXiv.1802.01059 arXiv:1802.01059 [cs, stat].

[19] Robert I McKay, Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O'neill. 2010. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines* 11 (2010), 365–396.

[20] Miguel Nicolau and Alexandros Agapitos. 2018. *Understanding grammatical evolution: Grammar design.* 23–53. https://doi.org/10.1007/978-3-319-78717-6_2

[21] Michael O'Neill and Conor Ryan. 2003. Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language, volume 4 of Genetic programming.

[22] Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. 2017. Smooth operator: Control using the smooth robustness of temporal logic. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 1235–1240.

[23] Ritam Raha, Rajarshi Roy, Nathanaël Fijalkow, and Daniel Neider. 2022. Scalable Anytime Algorithms for Learning Fragments of Linear Temporal Logic. In *Tools and Algorithms for the Construction and Analysis of Systems*, Dana Fisman and Grigore Rosu (Eds.). Springer International Publishing, Cham, 263–280. https://doi.org/10.1007/978-3-030-99524-9_14

[24] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Model-Agnostic Interpretability of Machine Learning. https://doi.org/10.48550/arXiv.1606.05386 arXiv:1606.05386 [cs, stat].

[25] Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20 (1987), 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

[26] Rajarshi Roy, Jean-Raphaël Gaglione, Nasim Baharisangari, Daniel Neider, Zhe Xu, and Ufuk Topcu. 2023. Learning Interpretable Temporal Properties from Positive Examples Only. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 5 (June 2023), 6507–6515. https://doi.org/10.1609/aaai.v37i5.25800

[27] Rukma Talwadker, Surajit Chakrabarty, Aditya Pareek, Tridib Mukherjee, and Deepak Saini. 2022. CognitionNet: A Collaborative Neural Network for Play Style Discovery in Online Skill Gaming Platform. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*. Association for Computing Machinery, New York, NY, USA, 3961–3969. https://doi.org/10.1145/3534678.3539179

[28] Josep Valls-Vargas, Santiago Ontañón, and Jichen Zhu. 2015. Exploring Player Trace Segmentation for Dynamic Play Style Prediction. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 11, 1 (2015), 93–99. https://doi.org/10.1609/aiide.v11i1.12782

[29] Marcell Vazquez-Chanlatte, Susmit Jha, Ashish Tiwari, Mark K. Ho, and Sanjit A. Seshia. 2018. Learning Task Specifications from Demonstrations. https://doi.org/10.48550/arXiv.1710.03875 arXiv:1710.03875 [cs].

[30] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised Deep Embedding for Clustering Analysis. https://doi.org/10.48550/arXiv.1511.06335 arXiv:1511.06335 [cs].

[31] Xuan Zhao and Marcos Campos. 2021. Reinforcement Learning Agent Training with Goals for Real World Tasks. *arXiv:2107.10390 [cs]* (July 2021). http://arxiv.org/abs/2107.10390 arXiv: 2107.10390.

Chapter 12

On the Importance of Contextualizing an Educational Escape Room Activity

On the Importance of Contextualizing an Educational Escape Room Activity

Pedro Antonio González-Calero[1], Irene Camps-Ortueta[2], Pablo Gutiérrez-Sánchez[1] and Pedro Pablo Gómez-Martín[1]

[1]Universidad Complutense de Madrid, Spain

[2]Universidad Francisco de Vitoria, Pozuelo de Alarcón, Madrid, Spain

pedro@fdi.ucm.es
irene.camps@ufv.es
pabgut02@ucm.es
pedrop@fdi.ucm.es

https://doi.org/10.34190/ejel.22.4.3199

Abstract: This paper describes the design and evaluation of "Enigma Bio", an educational escape room activity that aims to convey the abstract concept of biodiversity to children between 11 and 13 years of age, making them aware of the importance of climate change and its impact on biodiversity. The design of Enigma Bio is closely linked to the Biodiversity exhibition at the National Museum of Natural Sciences in Madrid, designed for a visit in groups of between 20 and 30 children, with an approximate duration of one hour, running on mobile devices and including augmented reality technology. The purpose of this research is to determine whether, in the case of educational escape room activities in museums with a limited time duration, it is more effective to have a pre-session introducing the topic. Our hypothesis is that without the context of the pre-explanation, the playful component of the game may be too powerful and may cause children not to pay enough attention to the message that the game intends to communicate, and even more so when dealing with a complex message such as the effect of climate change on biodiversity. To answer this research question, we follow an A/B testing experimental design involving two groups of children, one of which received an introductory talk on biodiversity and climate change before going to the museum and the other did not. The experimental design is completed with a pre-post evaluation of the children's environmental awareness by means of a previously validated questionnaire. The results of the experiment provide valuable insights into the effectiveness of the pre-session introduction in enhancing the learning outcomes of short educational escape room activities. Significant differences were observed between pre- and post-activity tests, indicating a moderate overall increase in awareness scores within both individual groups (A and B) as well as across the combined results. The findings suggest that the pre-session introduction indeed plays a role in enhancing students' awareness of the targeted message. These results represent a breakthrough in the e-learning practice that will be of value to other designers of educational escape rooms with a limited time duration.

Keywords: Educational escape room, Serious games, Games at museums, Augmented reality game

1. Introduction

Education is a major museum function, carried out by a dedicated staff and of concern to curators, exhibition designers, and other museum professionals. In large museums, the education staff, including part-time workers, docents, and occasional teachers, may represent up to 50 percent of all employees (Hein 2006). Museum educators engage in a broad range of activities, including tour programs, informal gallery learning programs, and family programs. Typically, educational activities involve some type of interaction with or around objects in the museum (Witcomb, 2006), which on many occasions may include some form of game (Beale, 2011). In the last few years, there have been a growing number of educational activities in museums using mobile devices to support interactive activities (Koutsabasis, 2017) and games (Paliokas and Sylaiou, 2016; Malegiannaki and Daradoumis, 2017).

The work presented here is motivated by several iterations in the design of the Enigma saga of games for museums developed by PadaOne Games as part of a growing number of initiatives incorporating augmented reality (AR) in exhibit-based informal science education settings (Goff, et al., 2018). The Enigma Bio game is conceived as a tool for educators guiding school groups on an approximately one-hour visit to the biodiversity exhibition at the National Museum of Natural Sciences in Madrid (MNCN, by its acronym in Spanish). The exhibition's content follows a thematic thread. It begins by introducing various forms of biodiversity, then links biodiversity to natural selection and adaptation to the environment, and finally addresses the threats to biodiversity, highlighting how human actions are impacting the climate and causing biodiversity loss.

Cite this article: González-Calero, P.A. et al. 2024. "On the Importance of Contextualizing an Educational Escape Room Activity", *The Electronic Journal of e-Learning*, 22(4) pp 43-56, https://doi.org/10.34190/ejel.22.4.3199

Climate change is a problem that the scientific community and the United Nations have been warning about through the yearly Climate Change Conferences held in the framework of the UNFCCC (United Nations Framework Convention on Climate Change) since 1992. Commercial video games such as “Alba: A Wildlife Adventure” by Ustwo Games (2020), "Beyond Blue" by E-Line Media (2021), and "Gibbon: Beyond the Trees" by Broken Rules (2024) are being used to raise awareness of the climate change problem among young people. The challenge for Enigma Bio is to achieve, in just one hour of play, a positive impact on the awareness of the participating children, taking advantage of the group experience in the physical space of the museum, the content of a specially designed exhibition, and the presence of an educator as a mediator of the experience.

Enigma games are treasure hunts designed specifically for museum settings, using image recognition and augmented reality (AR) on mobile devices to establish a link with the museum’s content. Image recognition guides players through the museum's exhibits, while AR offers clues and supplementary content about these items. However, based on our previous experience in designing this type of treasure hunt game (Camps-Ortueta, et al., 2019), it is very difficult to combine the fun of the game with the control of the interaction of the group of children with the educator. The solution we have found for Enigma Bio, as described in this paper, is to turn the treasure hunt into a form of escape room where the educator takes a central role in the game by controlling the gates.

Enigma Bio arose in response to a need detected at the museum by educators. While new technologies increase children's engagement with proposed tasks, the available applications often do not align with the museum's specific needs. The idea of using the escape room format arose from the need for educators to have a video game that stops the action at certain points so that educators can give their explanations.

Research has shown the effectiveness of using games to raise awareness about the consequences of climate change and the actions we can take to mitigate its effects. (Flood, et al., 2018) discuss two types of games: short ones that serve as motivators, and longer games that allow players to delve deeper into the complex relationships among the factors involved in the problem. Enigma Bio falls within the short game category, with the goal of improving our comprehension of the implications of climate change on species extinction and biodiversity loss.

The main purpose of this research is to determine whether, in the case of educational escape room activities in museums with a limited time duration, it is more effective to have a pre-session introducing the topic. Our hypothesis is that without the context of the pre-explanation, the playful component of the game may be too powerful and cause children not to pay enough attention to the message that the game aims to convey, especially when dealing with a complex message such as the effect of climate change on biodiversity. To answer this research question, we follow an A/B testing experimental design involving two groups of children: one of whom received an introductory talk on biodiversity and climate change before going to the museum, and one who did not. We complete the experimental design with a pre-post evaluation of the children's environmental awareness using a previously validated questionnaire. Our findings may be useful for other designers of educational escape room activities.

The rest of the paper runs as follows. In Section 2 we introduce educational escape rooms and describe how Enigma Bio fits within that framework. Section 3 describes the design and the main game mechanics of Enigma Bio. Section 4 details the experiment we conducted to evaluate the effectiveness of the game and the impact of having a pre-session introducing the topic. Section 5 presents and discusses the findings from the experiment. Finally, Section 6 addresses conclusions and future work.

2. Literature Review

Escape rooms are live-action team-based games where players discover clues, solve puzzles, and accomplish tasks in one or more rooms in order to achieve a specific goal, usually escaping from the room, within a limited amount of time (Nicholson 2015; Wiemker, Elumir and Clare, 2015). From the original mission of escaping from a locked room, many variants have emerged that are nowadays also considered escape room games, including: solving mysterious murders, opening locked boxes, or unraveling mysteries in order to avoid the end of the world (Veldkamp, et al., 2020). The genre has grown and proliferated in all major cities in recent times. Numerous escape rooms have opened, offering visitors a wide variety of experiences. Additionally, over time, alternative games have appeared, some with digital support and others with physical support that also echo the great success.

Escape room games have also begun to gain traction in academia (Fotaris and Mastoras, 2019). Many escape rooms designed for the classroom have been simplified to a group tabletop activity involving a series of locked

boxes (Schaffhauser, 2017), as it is not feasible or even legal to lock a subset of a class in a room and wait until they puzzle their way out. When properly designed, these types of games provide a motivating and immersive experience for students, although they may lose the feeling of complete immersion that escape room experiences provide (Clarke, et al., 2017).

Previous versions of the Enigma saga games for museums developed by PadaOne Games took the form of treasure hunts (Ihamäki, 2014). As such, they consisted of a sequence of searches, clues and quizzes that ran through a museum exhibit and were played at the participants' own pace. Although they worked as games and managed to attract the children's attention to the objects in the museum, they left the educator in the background as he had no role in the game and typically the children did not interact with him except at the beginning and end of the game.

According to Veldkamp, et al. (2020) Enigma saga games already included the main activity of educational escape rooms, namely, cognitive puzzles that make use of the players' thinking skills and logic. Nevertheless, by incorporating additional escape room game design ideas into Enigma Bio we have managed to meaningfully incorporate the educator into the game, without compromising the fun. These are the main design elements added:

- Divide the museum exhibition space into zones, each of which acts as a room, so that you need to solve all the puzzles in one zone in order to leave it and move on to the next.
- Introduce randomness in the order of the puzzles in a room, given that there is a group of people trying to solve the game independently (typically the game is played in pairs, where each pair shares a mobile device). When puzzles are always tackled sequentially in the same order, some players end up leading others, which can be addressed by introducing an element of randomness.
- Position the educator at the end of the zone, serving as the guardian of the room. This arrangement ensures that the objective in each room is to discover a question to pose to the educator. The winner from each zone will have the privilege of being the first to present the correct question to the educator. This setup ensures that all players will reach the educator as they progress through the zone, where they will convene. The educator will then offer an explanation related to their recent discoveries within the museum.

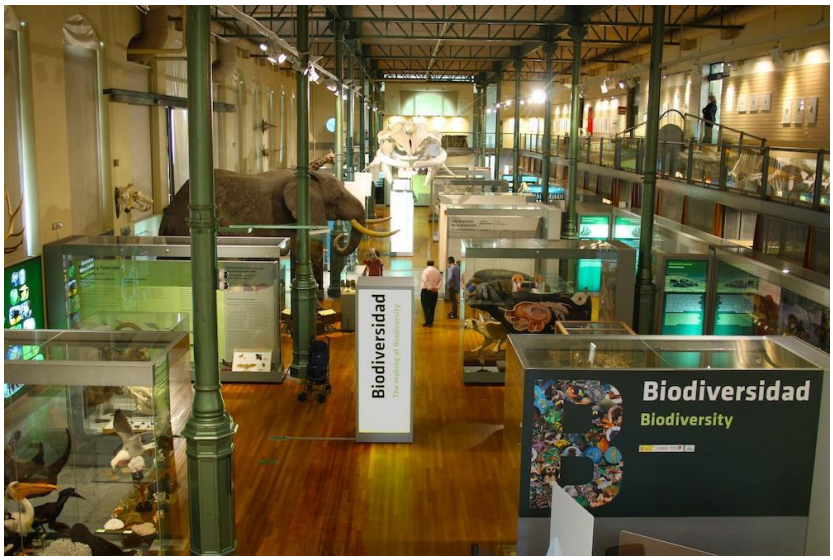
Next Section describes in detail the design of Enigma Bio.

3. Enigma Bio

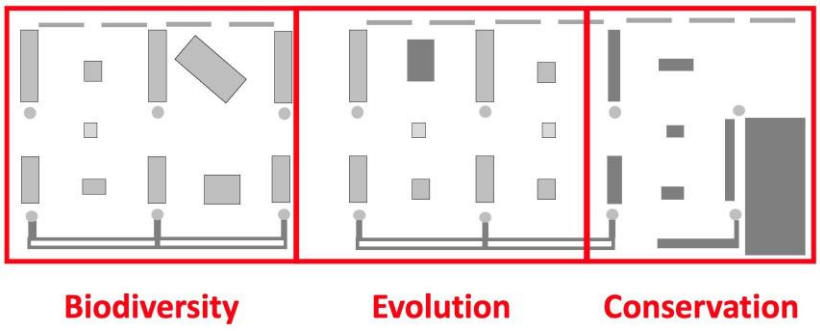
Enigma Bio is built around the biodiversity exhibition at MNCN. The concept of Biodiversity revolutionized the way we approach the study and conservation of nature by bringing together in a single concept the variety of species, their genetic variability, and their interrelationships with each other and with their environment. The exhibition, depicted in Figure 1a, attempts to answer these questions: What is Biodiversity? How has evolution shaped today's Biodiversity? How should Biodiversity be conserved? The exhibition is divided into three areas, as shown in Figure 1b: Biodiversity, Evolution and Conservation.

The first area of the exhibition explains what biodiversity is, how it is distributed in the different biomes of the world and how it manifests itself in the shapes, colors and relationships between the different organisms that make up ecosystems. Large collections of insects and mollusk shells then serve to explain the different levels at which we appreciate Biodiversity, from the gene to the ecosystem level. There is also a space to show how scientists try to order Biodiversity by classifying and naming living things.

The origin and the tree of life, whose branches link all living organisms evolutionarily, pave the way for the second area: "Biodiversity, the Fruit of Evolution." Here, the theory of evolution through natural and sexual selection is expounded upon, and its genetic basis is explained. Extinctions in the distant past, and more recent ones related to human activity, begin the area of "Conservation". This space not only reflects the direct causes of extinction and its victims, exhibiting extinct species such as the marsupial wolf, but also answers the questions of what, where and how to conserve.



(a) General view.



(b) Floor map.

Figure 1: Biodiversity exhibition at MNCN

As shown in Figure 1b, while the areas in the exhibitions are not physically separated, the display cases define spaces that can be perceived as connected, with "doors" enabling passage between zones. In Enigma Bio, each of these three "rooms" adhere to an open structure, allowing players to solve various puzzles simultaneously and without a specific order.

Enigma Bio's core game mechanic centers on using a mobile device's camera to search for artifacts, using image recognition technology. Players advance through a narrative adventure by following clues, answering questions, and engaging in different stages. Throughout these stages, participants encounter narrative elements, dialogues, artifact hunts, and blocks that regulate player advancement.

An artifact hunt prompts participants to use their device's camera to locate an object indicated by a game clue. Successfully identifying the artifact completes the search. Usually, an augmented reality image or text will overlay the object, offering hints or additional information for the game. To encourage students to read the content of the explanatory panels in the exhibition, many of the object searches in Enigma Bio refer to details on those panels, as shown in Figure 2 where several children try to find a detail on the panel "the shapes of biodiversity".

Figure 2: Interaction with the exhibition content

During a room stage, the participant has the freedom to interact with all the sub-sequences of artifact hunts and puzzles positioned within the room in any order they choose. To progress to the next stage from a room, the user must solve a final puzzle, which will only be possible to solve successfully after having completed the rest of the quests hidden in the room.

At the end of the room, the player will find a control stage in the game designed to prevent participants from progressing until they are given a keyword known only to the museum educator. This will allow to run the escape room in a synchronized manner for the group of participants, by allowing to wait until all participants have completed one room before granting passage to the next. And, more important, this will bring full attention of the children to the explanations of the museum educator, as shown in Figure 3.

Next, we describe the interface and more details about the main elements of gameplay in Enigma Bio. The complete game design, including the tutorial and three rooms associated with each of the three areas of the biodiversity exhibition, are described in (SPICE H2020 project, 2023).

Figure 3: Explanations of the educator

3.1 Enigma Bio Game Mechanics

The Narrative Screen (Figure 4) is a common resource in Enigma Bio. It consists of a character icon, character name, and a text box to communicate information and instructions to the player. Multiple narrative screens can be linked together, allowing for character changes, and creating interactive conversations.

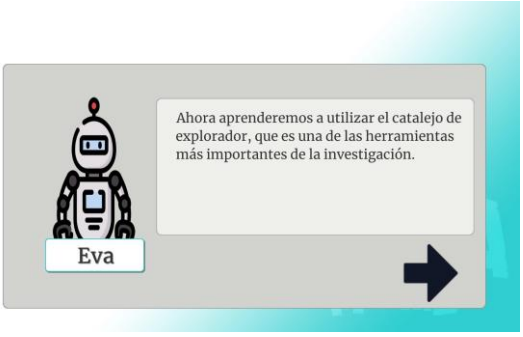


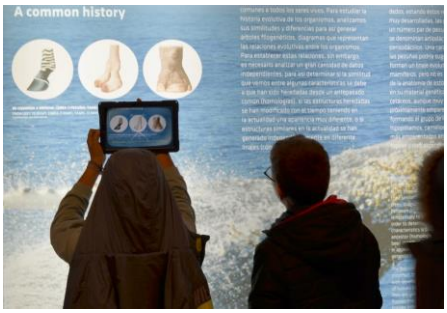
Figure 4: Character and Text screen

The Scan screen used in the game for artifact hunts can be seen in Figures 5a and 5b. By using the device's camera, it can track any image. Once the player finds the correct image, the overlay will change from "Searching" to "found" and the compass shown in the interface will change color to show that´s the correct image. Typically, in Enigma Bio a text or an image will be shown on top of a tracked image, as shown in Figure 5c where the image tracked is the puffer fish and the overlay indicates to search for the "shapes of biodiversity panel".

Since players will usually not be familiar with this mechanic, the main goal of the tutorial phase in Enigma Bio is help the player to learn how to use this tool, so that later she can use it in the escape room.

The Escape Room Screen, shown in Figure 6a serves as the central hub for every room in the game, featuring a total of 5 puzzle blocks. While four of these blocks are accessible at any time and in any order, the central block can only be accessed upon completing the other four.

(a) Scan Screen.



(b) Scan Screen in action.

(c) AR Overlay.

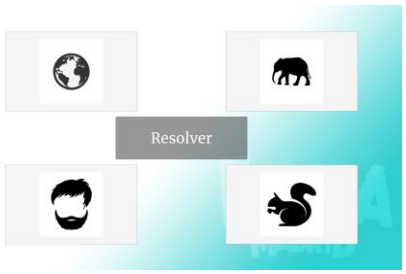
Figure 5: Artifact search

Once a block in the room is successfully completed, the player will receive a piece, as the one shown in Figure 6b, needed to solve the final puzzle in the room, and it will no longer be possible to re-enter that block. Completed items will undergo a color change to visually indicate their status as finished, accompanied by a new image display. As illustrated in Figure 6c, completed blocks exhibit puzzle pieces instead of the icons featured in Figure 6a. Notably, the central button is highlighted, signaling to the player that access to the final block has been unlocked.

The final block in every room is a puzzle to be assembled with the pieces obtained in the room, and possibly, some others, as shown in 7. The solved puzzle will provide a clue for the next artifact to find, as, for example, in Figure 7, where there is a picture of Carl Linnaeus, which tells the player to find that same image for answering the next questions.

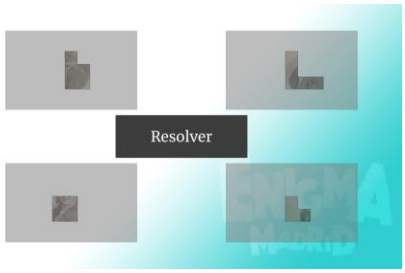
The game is interspersed with different types of questions that usually refer to the content of the exhibition panels. The idea is to alternate search phases with question phases, so that first we make sure that the player is in front of a certain panel and then we ask them a question whose answer is on that panel.

The basic type of question in Enigma Bio is a multiple-choice question as the one depicted in Figure 8a, that may have one or more correct answers. Upon providing a correct answer, a green checkmark will be displayed, or, in the case of an incorrect answer, the screen will exhibit a red cross (as shown in Figure 8b) and will also reveal the correct answer for reference. These are non-blocking questions since the player can proceed the game even with a wrong answer, but the number of right answers will be displayed at the end of the game and players will compete to obtain the highest score.



(a) Escape Room Screen.

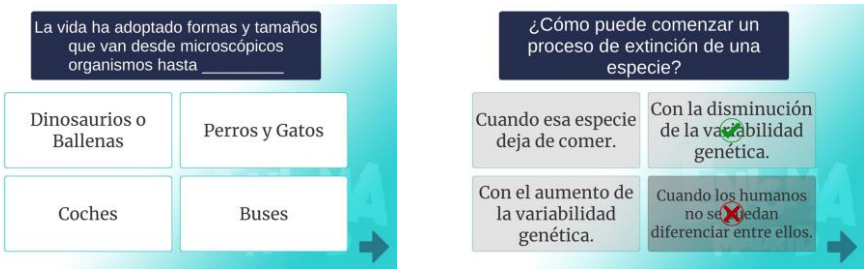
(b) Puzzle piece obtained.



(c) Escape Room Completed.

Figure 6: Escape Room Screen

Figure 7: Puzzle Screen



(a) Question.

(b) Incorrect answer.

Figure 8: Multiple choice questions

There is also a question type that involves specifying a specific four-digit number. This format is particularly suitable for inquiries that require a numerical year as the answer. For instance, in Figure 9a, the question presented is: “In what year was the term ’Biodiversity’ first used?” with the correct response being: 1988. These are blocking questions because the game cannot proceed unless the correct answer is provided, thus forcing the player to find it.

Finally, there is also an open question type that entails a multiple-choice format without a designated correct answer. This type of question can serve various purposes, such as gathering insights about the player or functioning as a questionnaire at the conclusion of each level.

(a) Four-digit question. (b) Open question.

Figure 9: Questions

Every room in Enigma Bio finishes with an "Introduce Room Code" stage as the one shown in Figure 10a. This screen prompts players to input a code for advancement. The code will be provided by the museum educator, once the correct question is posed to him. The last piece of information in every room is a question that must be posed to the educator. In order to promote competition, the educator will distinguish the first players to find the question, making them stand by him while providing the answer to the group, as shown in Figure 10b.

(a) Introduce room code. (b) Museum Educator's explanation.

Figure 10: Room exit

4. Research Design and Method

In this experiment, first, we would like to assess whether playing Enigma Bio raises children's awareness of the threats posed by climate change. In addition, we would also like to answer the question of whether, in the case of short games in museums, it is more effective to have a pre-session introducing the topic. Our hypothesis is that without the context of the pre-explanation, the playful component of the game is too powerful, and the child does not pay enough attention to the message that the game intends to communicate. In addition, as this activity is aimed at primary school children, aged 11-13, our hypothesis was that the concept of biodiversity and its relation to climate change may be too complex in general for these children to grasp just from the museum activity.

In this experiment we had the participation of 57 children in 5th grade of primary school (ages between 11 and 13) who visited the National Museum of Natural Sciences on November 22, 2022. The children were randomly assigned to one of two groups, experimental group (29 children) and control group (28 children).

With a pre/post measures design, we attempt to address the hypothesis at hand: a pre-talk is necessary to develop greater awareness of threats to biodiversity related to climate change. The evaluation is carried out by means of a validated questionnaire on environmental awareness (van Valkengoed, Steg and Perlaviciute, 2021). The questionnaire, included in Appendix A consists of 5 factors that reflect the 5 most common perceptions regarding climate change: people's perceptions of the reality and causes of climate change, and the perceived valence, spatial distance, and temporal distance of the consequences of climate change.

The validated questionnaire provided by van Valkengoed, Steg and Perlaviciute (2021) consists of 7-point Likert scale responses to the questions posed where 1 is "strongly disagree" and 7 is "strongly agree". This type of response can be a problem for children of early age (Mellor and Moore, 2013). Therefore, we chose to simplify the questionnaire and offer students only 2 possible answers: "I agree" or "I disagree". With this modification the scores we will obtain are altered since the items only offer dichotomous scores (0 and 1) and each of the 5

factors offers a maximum score of 5 points if the subject is fully aware of the environmental impact of the factor in question.

The experimental setup involves two groups, both of which were given the validated questionnaire on environmental awareness at the beginning (pre). However, one group was also given a 30-minute lecture on Biodiversity and climate change before the museum activity, while the other was not. Both the initial questionnaire and the lecture took place at their school one week before the visit to the museum.

Both groups visited the museum and carried out the activity in the Biodiversity exhibition which consists of 50 minutes playing Enigma Bio with the active participation of an educator. Finally, one week after the visit to the museum, back at school both groups took again the environmental awareness questionnaire (post).

5. Findings

The results obtained demonstrate encouraging outcomes. Significant differences were observed between pre- and post-activity tests, with overall scores showing a moderate increase after performing the activity, both within each individual group (A/B) and across the combined results of the groups. This tendency can be seen visually in Figure 11, where the clustered distributions of the student scores before and after the activity are represented via a box plot.

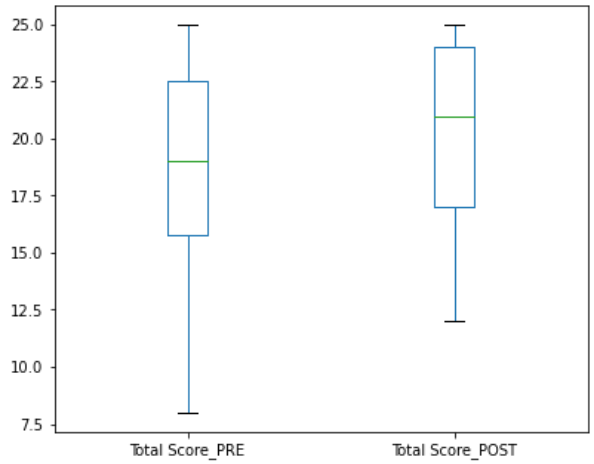


Figure 11: Box plot of the total score distribution (clustered samples)

After conducting the Wilcoxon signed-rank test for paired samples ($\alpha = 0.05$), we observed a clear and significant trend of score improvement among the students (refer to Table 1). The trend is evident and statistically significant in 3 out of the 5 factors assessed in the test, as well as in the overall total score. This test, however, only allows us to establish that there are significant differences at the individual level between test administrations, but by itself does not provide evidence of the direction of the changes.

Table 1: Clustered results of the validated environmental awareness test van Valkengoed, Steg and Perlaviciute (2021) (mean scores and significance of the Wilcoxon test for paired samples, N = 57)

	Reality	Causes	Conse-quences	Spatial Distance	Temporal Distance	Total Score
Pre	4.47	3.18	3.29	3.93	3.58	18.45
Post	4.62	3.83	3.94	3.83	3.63	19.85
p-value	0.0052	0.0020	0.0004	0.0028	0.0579	0.0013
Sig.	**	**	***	**	_	**

To address this concern, we can refer to the data on the proportion of students who improve their total test score, which amounts to 55% of the sample. When looking at individual factors, the proportion of improving students ranges between 16 and 33%, with a good number of participants whose scores in that factor remain unchanged between tests. In fact, if we analyze the number of factors in which students report better scores, we obtain an average of 1.16 factors ($\sigma = 1.02$, not accounting for the total). Consequently, we can confidently state that the activity significantly enhances the students' general awareness, regardless of whether they

received the lecture or not, although this improvement is generally focused on a small subset of awareness factors per student.

The hypothesis aimed at validating the need to include a talk prior to the museum visit to reinforce awareness of the causes threatening biodiversity, particularly the role of climate change as a threat, was moderately supported by this experiment.

After the initial administration of the questionnaire, a Mann Whitney U test was conducted to assess whether the factor-to-factor and total scores' distributions of control and experimental groups before the activity were similar. The results indicated no significant evidence to reject the null hypothesis of similarity of populations in all comparisons (for the total score, p-value = 0.44). In other words, both randomly chosen groups exhibited similar behavior before the activity, a crucial factor in avoiding bias in the experiment.

As shown in Tables 2 and 3, the students assigned to the experimental group exhibit marked and positively trending significant differences in their individual behavior concerning 2 out of 5 considered factors, as well as in the overall score. On the other hand, the control group does not perceive statistically significant differences in the total score of the questionnaire, although it does show moderately significant positive changes in 2 out of 5 factors. Both groups seem to display a subtle negative trend in the spatial distance factor, although the difference in the mean is not particularly high in either case.

Table 2: Control Group (no presentation before)

	Reality	Causes	Conse-quences	Spatial Distance	Temporal Distance	Total Score
Pre	4.50	3.36	3.46	3.86	3.86	19.04
Post	4.74	3.59	4.11	3.78	3.89	20.11
p-value	0.0103	0.1797	0.0177	0.0378	0.1638	0.0845
Sig.	*	_	*	*	_	_

Table 3: Experimental Group (with presentation before)

	Reality	Causes	Conse-quences	Spatial Distance	Temporal Distance	Total Score
Pre	4.44	3.00	3.11	4.00	3.30	17.85
Post	4.48	4.08	3.76	3.88	3.36	19.56
p-value	0.1342	0.0050	0.0086	0.0369	0.0964	0.0059
Sig.	_	**	**	*	_	**

In summary of the results collected from the group analysis, the following points of interest can be established:

- The experimental group shows a statistically significant and positive individual trend in their overall awareness level, while in the control group, there is not enough evidence to confidently establish a similar result. However, in the latter group, the p-value falls below 0.1, and the pre- and post-experiment means are separated by about one point, indicating that the differences in trends between groups are not as pronounced as initially perceived.
- The experimental group appears to have a markedly positive effect on the students' awareness concerning the factors of causes and consequences. The improvement in consequences also extends to the control group, which shows significant improvement in this aspect as well.
- The control group exhibits a statistically significant and positive difference in the reality factor, although the group's mean remains close to the original.
- Both groups display a moderate negative trend in the spatial distance factor, although the differences in the means are not particularly pronounced.

5.1 Discussion

The main aim of an educational game is to use the playful experience as a vehicle for introducing significant learning (Djaouti, et al., 2011). If the design of games is already an activity that requires large doses of creativity

and knowledge (Fullerton, 2008), the design of educational games also requires finding a balance between learning and fun that makes it especially demanding (Shute, et al., 2020), and even more so when dealing with a complex message such as the effect of climate change on biodiversity. Nonetheless, others have demonstrated the effectiveness of using games as a means to raise awareness and educate on biodiversity-related issues such as climate change (Reckien and Eisenack, 2013; Flood, et al., 2018) or sustainability (Fabricatore and Lopez, 2012; Nordby, et al., 2016; Mercer, et al., 2017). Enigma Bio faces additional challenges targeting primary school children aged 11-13, who may find it difficult to understand certain abstract concepts, and even more so if the experience is restricted to the short duration of a school visit to a museum (Camps-Ortueta, et al., 2023). The solution we propose to add an introductory session prior to the game in the museum is in line with Markmlund and Taylor (2016), who states that is not only the game but also the context that matters when we talk about educational games.

The results of the experiment provide valuable insights into the effectiveness of the pre-session introduction in enhancing the learning outcomes of short educational games. Our hypothesis, which posited that without the context of a pre-explanation, the playful element of the game might overshadow the intended message, was moderately supported by the findings. Significant differences were observed between pre- and post-activity tests, indicating a moderate overall increase in awareness scores within both individual groups (A and B) as well as across the combined results. This trend can be visually observed in Figure 11, depicting the distributions of student scores before and after the activity.

Nonetheless, there are differences in the factors that van Valkengoed, Steg and Perlaviciute (2021) refers to as "climate change perceptions": upon closer examination, it is evident that the activity contributed to a statistically significant and positive individual trend in the overall awareness level of the experimental group. However, the control group did not exhibit strong enough evidence to confidently establish a similar result. Although the control group displayed a lower p-value and a noticeable separation between pre- and post-experiment means, differences in trends between the two groups were not as pronounced as initially anticipated.

The findings suggest that the pre-session introduction indeed played a role in enhancing the students' awareness of the targeted message, though further experiments would be required to reinforce this claim on a more general basis. The experimental group displayed marked improvements in understanding the factors related to causes and consequences of biodiversity loss, with the control group experiencing subtler enhancements in these aspects. This implies that the introductory context helped students better comprehend the nuanced relationships presented in the game that were related to these factors.

On another note, the results regarding spatial and temporal distance perceptions yielded unexpected findings. Contrary to our anticipation, students either slightly regressed in these aspects or exhibited insignificant changes post experiment. This surprising outcome prompts a closer look at the complexity of these concepts for young learners.

The intricate nature of spatial and temporal dimensions, especially when connected with the abstract concept of climate change, might pose challenges for primary school children's cognitive understanding. These complex interrelationships may demand a more advanced cognitive development level to be fully grasped.

Considering that spatial and temporal concepts are often challenging even for individuals of various ages, the difficulty could be magnified for children in the primary school age group. The gradual and subtle nature of climate change effects over time and space could require tailored teaching approaches that bridge these abstract notions with the students' real-world experiences.

6. Conclusions and Future Work

In this study, we have introduced Enigma Bio as an enhanced version of the enigma saga's treasure hunt games, infused with escape room elements to enrich the puzzle-solving experience. By incorporating educators into the gameplay mechanics, we have transformed the game into a valuable tool for engaging students.

Our initial findings shed light on the game's effectiveness, even within its brief duration, a result that could be attributed to the immersive museum setting. Interestingly, initial outcomes also suggest that prior exposure to relevant concepts, such as biodiversity, could amplify the game's impact. Notably, the experiment highlighted the positive influence of a pre-session introduction on the efficacy of short educational games in helping primary school children understand complex concepts. While both groups showed increased awareness, the experimental group, armed with contextual introductions, demonstrated more substantial improvements, particularly in comprehending the intricate relationship between biodiversity and climate change, especially

regarding climate change's effects and consequences on biodiversity. These findings underscore the crucial role of well-designed instructional strategies in facilitating young learners' grasp of intricate subjects and fostering a stronger connection to real-world challenges.

Looking ahead, our research will expand into more comprehensive experiments to provide conclusive insights into the effectiveness of our approach. Further exploration into optimizing learning outcomes for young students will involve refining the timing, content, and delivery of pre-session introductions. Additionally, considering the unexpected trends in spatial and temporal distance perceptions, we propose investigating the cognitive development trajectory of young learners in understanding complex environmental concepts. Innovative pedagogical strategies and gamifications aimed at making these abstract notions more relatable could prove vital in nurturing a more accurate and comprehensive understanding of climate change and its significance in the lives of primary school children.

Acknowledgements

We are very grateful to the personnel at the National Museum of Natural Sciences in Madrid for their support and assistance in designing and evaluating Enigma Bio, in particular to Luis Barrera and Pilar López. We also thank the anonymous reviewers for their insightful comments and suggestions

This work has been partially supported by the Spanish Ministry of Science and Innovation under grant PID2021-123368OB-I00.

References

Beale, K., 2011. Museums at Play: Games, Interaction and Learning. London: MuseumsEtc Ltd.

Broken Rules, 2024. Gibbon: beyond the trees. [Apple Vision Pro Game]. Austria: Broken Rules

Camps-Ortueta, I., González-Calero, P.A., Quiroga, M.A., and Gómez-Martín, P.P., 2019. Measuring Preferences in Game Mechanics: Towards Personalized Chocolate-Covered Broccoli. In: E. D. V. der Spek, S. Göbel, E.Y. Do, E. Clua, and J.B. Hauge, eds. 2019. Entertainment Computing and Serious Games - First IFIP TC 14 Joint International Conference, ICEC-JCSG 2019, Arequipa, Peru, November 11-15, 2019, Proceedings, vol. 11863, of Lecture Notes in Computer Science, pp.15–27.

Camps-Ortueta, I., Deltell, L., and Gutiérrez-Manjón, S., 2023. Ludic application of augmented reality (AR) at the National Museum of Natural Sciences in Madrid, Spain. Revista Electrónica Educare, 27(2), pp.187-203.

Clarke, S., Peel, D., Arnab, S., Morini, L., Keegan, H., and Wood, O., 2017. Escaped: A framework for creating educational escape rooms and interactive games to for higher/further education. International Journal of Serious Games, 4(3).

Djaouti, D., Alvarez, J., Jessel, J., and Rampnoux, O., 2011. Origins of Serious Games. In: M. Ma, A. Oikonomou, and L.C. Jain, eds. 2011. Serious Games and Edutainment Applications. Springer, pp.25-43

E-Line Media, 2021. Beyond Blue. [Nintendo Switch Game]. USA: E-Line Media.

Fabricatore, C., and López, X., 2012. Sustainability Learning through Gaming: An Exploratory Study. Electronic Journal of e-Learning, 10(2), pp.209-222.

Flood, S., Cradock-Henry, N.A., Blackett, P., and Edwards, P., 2018. Adaptive and interactive climate futures: systematic review of 'serious games' for engagement and decision-making. Environmental Research Letters, 13(6).

Fotaris, P., and Mastoras, T., 2019. Escape rooms for learning: A systematic review. European Conference on Games Based Learning.

Fullerton, T., 2008. Game Design Workshop. A playcentric approach to creating innovative games. Morgan Kaufmann.

Goff, E. E., Mulvey, K. L., Irvin, M. J., and Hartstone-Rose, A., 2018. Applications of augmented reality in informal science learning sites: a review. Journal of Science Education and Technology, 27(5), pp.433-447.

Hein, G.E., 2006. Museum education. In: S. Macdonald, ed. 2006. A Companion to Museum Studies. Blackwell Publishing. Ch. 20.

Ihamäki, P., 2014. The potential of treasure hunt games to generate positive emotions in learners: Experiencing local geography and history using GPS devices. International Journal of Technology Enhanced Learning (6), pp.5-20.

Koutsabasis, P., 2017. Empirical Evaluations of Interactive Systems in Cultural Heritage: A Review. International Journal on Computational Methods in Heritage Science, 1(1), pp.100-122.

Malegiannaki, I., and Daradoumis, T., 2017. Analyzing the educational design, use and effect of spatial games for cultural heritage: A literature review. Computers and Education (108), pp.1-10.

Marklund, B.B., and Taylor, A.A., 2016. Educational games in practice: The challenges involved in conducting a game-based curriculum. Electronic Journal of e-Learning, 14(2), pp.122-135.

Mellor, D., and Moore, K.A., 2013. The Use of Likert Scales With Children. Journal of Pediatric Psychology, 39(3), pp.369-379.

Mercer, T.G., Kythreotis, A.P., Robinson, Z.P., Stolte, T., George, S.M., and Haywood, S.K., 2017. The use of educational game design and play in higher education to influence sustainable behaviour. International Journal of Sustainability in Higher Education, 18(3), pp.359–384.

Nicholson, S., 2015. Peeking behind the locked door: A survey of escape room facilities. [online] Available at: <https://scottnicholson.com/pubs/erfacwhite.pdf> [Accessed 24 April 2024]
Nordby, A., Øygardslia, K., Sverdrup, U., and Sverdrup, H., 2016. The art of Gamification; Teaching Sustainability and System Thinking by Pervasive Game Development. Electronic Journal of e-Learning, 14(3), pp.152-168.
Paliokas, I., and Sylaiou, S., 2016. The use of serious games in museum visits and exhibitions: A systematic mapping study. In: 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), Barcelona, Spain, 2016, pp.1-8.
Reckien, D., and Eisenack, K., 2013. Climate change gaming on board and screen: A review. Simulation and Gaming, 44(2-3), pp.253–271.
Schaffhauser, D., 2017. Breakout! gaming to learn. T.H.E. Journal Technological Horizons in Education, 44, p.6.
Shute, V., Rahimi, S., Smith, G., Ke, F., Almond, R., Dai, C., Kuba, R., Liu, Z., Yang, X., and Sun, C., 2020. Maximizing learning without sacrificing the fun: Stealth assessment, adaptivity and learning supports in educational games. Journal of Computer Assisted Learning, 37(1), pp.127-141.
SPICE H2020 project, 2023. Deliverable 5.3: Integrated interfaces for citizen curation. [online] Available at: <https://spice-h2020.eu/document/deliverable/D5.3.pdf> [Accessed 24 April 2024]
Ustwo Games, 2020. Alba a wildlife adventure. [PC Game] United Kingdom: Ustwo Games.
van Valkengoed, A.M., Steg, L., and Perlaviciute, G., 2021. Development and validation of a climate change perceptions scale. Journal of Environmental Psychology, 76, p.101652.
Veldkamp, A., van de Grint, L., Knippels, M.P.J., and van Joolingen, W.R., 2020. Escape education: A systematic review on escape rooms in education. Educational Research Review, 31, p.100364.
Wiemker, M., Elumir, E., and Clare, A., 2015. Escape Room Games: "Can you transform an unpleasant situation into a pleasant one?". In: J. Haag, J. Weißenböc, M.W. Gruber, M. Christian, and F. Freisleben-Teutscher, eds. 2015. Game based learning. St. Pölten, Austria: Fachhochschule st Pölten GmbH, pp.55-68.
Witcomb, A., 2006. Interactivity: Thinking beyond. In: S. Macdonald, ed. 2006. A Companion to Museum Studies. Blackwell Publishing. Ch. 21.

Appendix: The Questionnaire

Reality

1.	I believe that climate change is real.
2.	Climate change is NOT occurring.
3.	The world's climate is changing.
4.	I do NOT believe that climate change is real.
5.	Climate change is happening.

Causes

1.	Human activities are a major cause of climate change.
2.	Climate change is mainly due to natural causes.
3.	Climate change is mostly caused by human activity.
4.	The main causes of climate change are human activities.
5.	Climate change is caused entirely by natural processes.

Valence of consequences

1.	Overall, climate change will bring more negative than positive consequences to the world.
2.	Climate change will mostly have positive consequences.
3.	Climate change will bring about serious negative consequences.
4.	The consequences of climate change will be very serious.
5.	There will be mostly positive consequences of climate change.

Spatial distance

1.	My local area will be influenced by climate change.
2.	Climate change only influences locations far away from me.

3.	The region where I live will experience the consequences of climate change.
4.	The consequences of climate change will only take place in distant locations.
5.	Climate change will also influence the place where I live.

Temporal distance

1.	The consequences of climate change are visible now.
2.	It will be a long time before the consequences of climate change are felt.
3.	Only future generations will experience the consequences of climate change.
4.	The consequences of climate change will only be experienced in the far future.
5.	The effects of climate change will be felt very soon.

Chapter 13

Evaluating Usability for an AR enabled Escape Room Authoring Ecosystem

Entertainment Computing 55 (2025) 100982

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: www.elsevier.com/locate/entcom

Evaluating usability for an AR-enabled escape room authoring ecosystem

Pablo Gutiérrez-Sánchez *, Pedro P. Gómez-Martín, Pedro A. González-Calero, Marco A. Gómez-Martín

Software Engineering and Artificial Intelligence Department, Complutense University of Madrid, Calle del Prof. José García Santesmases, 9, Moncloa - Aravaca, 28040, Madrid, Spain

ARTICLE INFO

Keywords:
Authoring tool
Cultural heritage
Augmented reality
End-user programming
Usability
Game experience

ABSTRACT

The increasing ubiquity of mobile devices has enabled the surge of interactive experiences at cultural heritage sites, enriching visitor immersion. However, adapting interactive resources for different settings and types of creators remains a challenge. This paper introduces EnigMachine Editor, a web-based authoring tool for crafting augmented reality-enabled adventures inspired by treasure hunts and escape rooms. Designed for curators and hobbyists, it simplifies the generation of custom experiences without prior technical expertise, streamlining development compared to using traditional game engines directly.

The main objective of this work is to comprehensively evaluate the usability of EnigMachine Editor for non-technical users, alongside its ability to generate engaging game experiences. A game experience study was first conducted in which participants interacted with games from the platform and subsequently completed the Game Experience Questionnaire (GEQ). Next, two usability studies were conducted in which participants first completed a tutorial with Single Ease Questions (SEQ) after each exercise, and then a series of both short- and long-term tasks on the tool concluded with the System Usability Scale (SUS) questionnaire. The results obtained demonstrate that EnigMachine Editoris able to generate satisfactory experiences for museum visitors and highlight desirable usability properties, especially in the short-term test group.

1. Introduction

The ubiquity of mobile devices has allowed the proliferation of applications with which users, using their own devices, can participate in interactive experiences on a wide variety of topics. One context in which this is particularly true is that of museums and other cultural heritage sites. Owing to this, visitors can now easily receive enriched and personalized information along their tours.

In recent years, we have developed several applications of this type, turning the visitor experience into real *treasure hunts* or *escape rooms* with a strong augmented reality component, which we collectively refer to here as *adventures*. To do this, we have made use of tools commonly used in the video game industry, which facilitates development by avoiding the need to create a technological base that would demand a considerable time investment.

Game engines such as Unity,[1] Unreal,[2] or Godot[3] are in themselves complete authoring tools, providing programmers, designers, and artists with truly integrated development environments with which to develop their games. Even so, they remain generalist tools, and it is often the job of programmers to create specific game elements that designers and artists can then arrange and configure to build the gameplay they are aiming for.

Unity has long been our tool of choice for the creation of all our applications. While the details are not of major importance, in order to separate the technical part, in the hands of programmers, from the creative effort of crafting *adventures*, in the hands of designers, we rely on the use of *scriptable objects*. These are the mechanisms by which Unity allows programmers to define new types of *assets*, beyond maps, textures, models, or sounds, which designers can then manipulate to shape the entire interactive experience.

Despite being a highly valuable tool, the Unity editor is not particularly user-friendly when the number of these *scriptable objects* grows and the relationships between them skyrocket. Additionally, whilst still being conceptually *assets* from an execution point of view, *scriptable objects* are often very close to the way in which programmers view the

* Corresponding author.
E-mail addresses: pabgut02@ucm.es (P. Gutiérrez-Sánchez), pedrop@fdi.ucm.es (P.P. Gómez-Martín), pagoncal@ucm.es (P.A. González-Calero), marcoa@fdi.ucm.es (M.A. Gómez-Martín).

[1] https://unity.com/
[2] https://www.unrealengine.com/
[3] https://godotengine.org/

https://doi.org/10.1016/j.entcom.2025.100982
Received 2 February 2025; Received in revised form 19 May 2025; Accepted 20 June 2025
Available online 3 July 2025

game and can often cause a wealth of superfluous details to become visible. When designers devise an interactive experience such as a *treasure hunt* or *escape room*, their approach lies at a much higher level, and having to specify it through a myriad of small, interrelated, *scriptable objects* in an environment that is not tailored for it makes their task much more daunting. In addition, this forces them to have a Unity installation, as well as to adapt to and become minimally proficient in an environment that provides a vast battery of functionality. All of this generates a non-negligible amount of noise for the much more restricted work they are required to undertake.

Lastly, in our context, this way of using Unity for the creation of these kind of experiences introduced an additional logistic problem. Over time, we have created many versions of our different adventures to adjust them to specific museum initiatives (e.g., science weeks, nights at the museum, school visits) or to temporary changes in the exhibitions due to artefact loans or restorations of pieces. Most of the core foundations of our games (i.e., programming and art) are common to all of them and are collectively known as EnigMachine. With an increasing number of adventures (created using different *scriptable objects* configurations) performing version control of so many slightly different applications quickly became a struggle.

As a way to address all these problems, in the last few years we have worked on the development of an authoring tool external to Unity, which we refer to as EnigMachine Editor, that facilitates the creation of these interactive experiences. This is essentially a web application in which designers can write and specify an entire adventure, providing a clear, distraction-free interface directly within the browser so that it can be used from any device. Each adventure is managed in a distinct project within the application and follows a lifecycle completely independent of the Unity project that will later run the experience. The web application also provides an *export* mechanism through which it creates the *scriptable objects* that EnigMachine expects and thanks to which the final mobile application can be created automatically.

Aside from significantly improving workflow, facilitating version control, and avoiding the need for designers to deal with Unity, the time to create each experience has also been greatly reduced. An informal analysis with designers who were familiar with the nature and inner workings of the experiences created determined that, once an adventure is specified, deploying it to EnigMachine went from requiring a few days to mere hours in most cases. This is especially valuable when taking into account that a substantial portion of the testing and quality control process of these applications must be carried out on-site, outside the development space, with the consequent logistical limitations that this entails.

To confirm that the new authoring tool is also sufficiently accessible to novice users, in this paper we propose a study to test and validate the following preliminary hypotheses:

- **H1**. EnigMachine enables the creation of adventures that offer a rich gameplay experience, in which players perceive high levels of competence, fluency, and satisfaction, and low overall stress.
- **H2**. EnigMachine Editor is generally suitable for non-technical users wishing to build adventures from a previous design concept.
- **H3**. EnigMachine Editor is also generally useable by non-technical users when designing adventures directly from the platform itself over an extended period of time.

The rest of the paper runs as follows. Section 2 introduces related work, including relevant literature both for gaming applications for cultural heritage and authoring tools in general interactive experiences. Section 3 presents a high-level overview of the EnigMachine system and the AR-enabled games that can be created with it. Section 4 details the user experiments conducted with the aim of validating hypothesis H1 concerning perceived game experience in adventures created with our tool, with Section 5 presents an in-depth explanation of the different usability evaluations of EnigMachine Editor with preliminary groups of design students at our institution to test hypotheses H2 and H3 concerning short and long-term use of the system, with Section 6 outlining the main results of these sessions. Section 7 then summarizes the major findings in the study for each of the proposed hypotheses and concludes the paper outlining directions for future work.

2. Related work

2.1. Gaming applications for cultural heritage

Mobile technology has become a critical component of interactive initiatives for cultural sites, integrating effectively with other approaches, such as the physical objects and artefacts found in museums and exhibitions [1]. Additionally, there is a growing emphasis on "gamification" as a method to increase the appeal of cultural material to learners, as showcased in recent literature reviews [2–6].

The widespread adoption of AR and VR technology is noteworthy in this regard. According to Kahn et al. [5], 46.66% of the 45 publications assessing gamification strategies between 2015 and 2020 propose the use of AR or VR components to explore cultural heritage. Despite often being regarded as more captivating, serious games pose an array of difficulties for creators in that they are required to reach a careful balance between enjoyment and education [7]. While several general guidelines and heuristics exist for educators, researchers, instructional designers, and developers [6], crafting games that guarantee meaningful learning and present a genuine and respectful image of culture remains a challenging problem [2].

The notion of design patterns was first presented by Christopher Alexander in the context of architecture [8] and later adopted in the field of computer science by Gamma et al. [9]. Formalized in the game design domain by Björk et al. [10] in the mid-2000s, design patterns have recently found applications in AR-driven experiences for museums and exhibitions [11]. Namely, Linda et al. [11] focus on patterns that contain knowledge and functions for interactions like spatial connections or general overviews. These patterns enable content creators to focus on templates that learners want to adapt for particular exhibits rather than on devising elaborate templates from scratch. This strategy is consistent with our methods at EnigMachine, where we design educational escape rooms using escape patterns and templates.

Based on the treasure hunt concept, escape rooms target players wishing to seek clues, solve puzzles, and complete certain tasks within a particular period of time, usually with the goal of escaping the room [12,13]. In the recent past, this genre of entertainment has gained immense popularity in major cities and has begun to gain traction in academia [14]. Furthermore, the use of escape rooms for educational purposes has become increasingly popular across different levels of education, including higher education institutions [15]. When properly designed, escape rooms can provide an engaging environment that motivates learners to complete the tasks at hand [16] and also can provide a way to facilitate learning through failure [17]. EnigMachine has also adopted this format, progressively moving the focus away from simple treasure hunts [7] to allowing for the design of rich, fully fledged educational escape rooms where learners solve tasks in more holistic contexts.

2.2. Authoring tools for interactive experiences

Despite the potential of these technologies and concepts, authoring remains a significant challenge in the medium to this day [18,19]. The technical skills required to develop games for both cultural heritage and educational institutions can act as a major barrier to entry, preventing creatives or educators from accessing the field, while the production process can prove prohibitively expensive for institutions that do not possess the financial resources for it. The accessibility of the medium can thus be considered a major challenge for this field, especially given the already existing concerns and difficulties in the adoption of new

technologies, both in the cultural heritage [18,20] and educational domains [21].

The collective solution adopted by the research community appears to be focused on the development of authoring tools that enable users such as educators or curators, who are generally non-technical, to design interactive activities and narratives and translate them onto digital platforms. These efforts fall within the research line of what is known as end-user development, i.e., the creation of authoring tools aimed at professionals who may not be technically savvy enough to operate complex systems during the content authoring process [22]. Included here are tools geared towards the creation of augmented reality mobile applications and interactive stories in cultural heritage [23,24], multiplatform guides and experiences for museums [25–27], or narratives with GPS-based interaction mechanisms in urban environments [28].

Recent work about educational escape rooms [29,30] mention the use of commercial authoring tools for creating digital educational escape rooms (DEER), which use digital environments and may be played by participants not present on site, such as Genially [31], Telescape Live [32], Breakout EDU [33], Room Escape Maker [34], and even open-source platforms such as Escapp [35]. While these tools are suitable for remote or hybrid learning environments, they typically lack features tailored to physical gameplay or augmented reality (AR) integration. For example, Genially offers strong support for interactive visual storytelling but lacks spatial interaction. Telescape Live supports real-time collaboration but is primarily designed for digital replicas of physical escape rooms. Escapp, although open source, does not natively support AR features or physical integrations. In fact, at the time of writing, we are not aware of any authoring tool specifically designed for creating physical or AR-supported educational escape rooms such as our EnigMachine ecosystem.

These efforts do, however, demand careful attention to interface design and user experience across individual authoring tools. While numerous works and heuristics exist that focus on deriving design principles and evaluations for the games or interactive experiences themselves [36–41], whether for commercial games, serious games, or other educational or cultural heritage activities, a comparatively small fraction of this work addresses a similar analysis of the actual authoring tools. A number of state-of-the-art reviews on the issue explicitly highlight this limitation [42,43], stressing how the current spotlight lies primarily on the audience space as opposed to that of the authors, resulting in a landscape where few guidelines or studies exist examining what constitutes an effective end-user authoring tool. This scarcity could potentially stem from the complex nature of this kind of analysis, which often requires long experimental user sessions in order to meaningfully probe the characteristics of the system [44].

3. Description of the EnigMachine platform and games

As stated previously, the system presented in this paper, EnigMachine, constitutes a platform aimed at facilitating the construction of mobile games with significant augmented reality components. These games are built on the basis of the concepts of treasure hunts and escape rooms, and in what follows will be collectively referred to as *adventures*. The main objective of these adventures is to enhance visitors' immersion and involvement in museums and other similar spaces while encouraging interaction with the various elements of the exhibitions in which they take place.

This system is comprised of a web editor for content creation, known as EnigMachine Editor, and a mobile application to distribute the experience resulting from the design process within the tool. Games generated with EnigMachine feature a narrative that places the player as the protagonist of the adventure, as they navigate a series of interconnected puzzles, riddles, and quests interwoven into the main story. Structurally, adventures are divided into stages or phases from among the 13 types available at the time of writing. These phases, which can be interpreted as scriptable authoring templates (i.e., customizable content blueprints that follow a set of preconfigured patterns to dynamically generate or modify content), are grouped into distinct categories based on their underlying design intent (e.g., finding an artefact in the exhibit, collecting responses from participants, posing a puzzle or challenge to be solved by inspecting the environment, etc.), and are summarized in Table 1. These are presented to the player sequentially, requiring the completion of a stage to advance to the next, with the only exception being the Room-type phase, in which a set of parallel sub-blocks of sequential phases are displayed to be solved in any desired order. An essential aspect aimed to be attained is to provide the sensation of deep immersion, offering players the experience of being actively involved in the site visit instead of being merely passive observers of the venue. In order to improve technological accessibility, these games can be run on the smartphones and tablets of both the organizing institution and the visitors themselves. Players can then interact with puzzles by means of touchscreen features, GPS systems requiring them to reach specific geographic locations as part of the narrative of the experience, and augmented reality modules using the device's camera to scan objects or codes in their vicinity.

From a game designer's point of view, the complete process of authoring and deploying an adventure can be summarized as follows:

1. Content creators log into the web application and, via their user account, create the adventure leveraging the tools provided by the platform.
2. Once completed, the web tool generates a mobile application that can be downloaded by the user.
3. The application is installed on the mobile devices used by visitors.
4. Visitors engage with it.

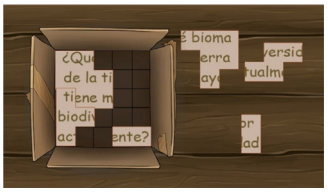
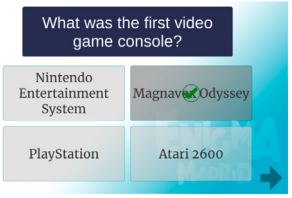
While the model of creating content through the web by non-technical users is not entirely novel and in fact is currently being actively applied by well-known platforms such as *Kahoot!* or *Canva*, EnigMachine Editor distinguishes itself by being specifically designed for the creation of escape rooms and treasure hunts with a strong environmental exploration component through AR, QRs and GPS in educational centres or museums, adapting its components to the needs of this context. Following the same approach of some of the mentioned platforms for content creation, our tool organizes the structure of the visit in sequences of building blocks represented as slides.

The web editor is conceived as a single-page solution with React as the main development framework and a supporting server in NodeJS for managing user data, adventures, and metrics. The phase-based modularity described above for the adventures extends naturally to the authoring tool, where each of the stages supported in the game is translated to a dedicated editing component, as shown for selected examples in Fig. 1.

Adventures created using the web editor are serialized to a structured JSON file with the adventure metadata (such as its title, description, characters, and overall settings), followed by the ordered list of stages, each specified with its own parameters according to the stage type. References to multimedia assets, such as character images or AR targets, are stored separately on the server and linked using unique URLs in the JSON. When a content creator completes their adventure, they can request a mobile app to be built through the platform. This triggers a dedicated builder service, which downloads the JSON file and automatically converts it to Unity3D's native scriptable object format, a structure optimized for runtime performance.

The building process can take some time (between 5 and 10 minutes for a typical adventure), however, as it involves assembling and packaging all the contents of the adventure into an application. To accommodate this efficiently, builds are managed by secondary "workers" running in the background. Once a requested build is complete, the frontend is automatically notified by the system to let the user

Table 1
Types of phases in Enigma games.

Phase Types	Description	Example
AR-Scan *AR-Overlay* *QR-Scan* *Reach GPS Spot*	Ask players to explore the venue and scan a given object using their device's camera in different fashions, or reach a certain GPS location in the vicinity.	
Pack Puzzle *Number Lock* *Password*	Ask players to solve a puzzle to progress, such as a four-digit code, a password stage, or a tile packing level.	
Narrative *Object Obtained* *Diary Page*	Show a narrative phase to players in which characters interact with one another, or the player obtains an item or diary page.	
MC Test *MC Free Answer*	Get players to choose from a set of multiple choice options either in a quiz fashion or as a free answer.	
Room Module	Have players solve blocks of puzzles in an escape room fashion, collecting clues to help them unlock a final puzzle block.	

know that their app is ready to download. This asynchronous process enables users to continue editing or creating new adventures without idly waiting for builds to finish.

The builder then runs the Unity batch editor to create a standalone mobile application that contains the complete adventure and all the assets needed. The app is therefore completely self-contained and requires no Internet connection once installed. This is in contrast to platforms such as *Kahoot!*, which load content on demand at runtime, while EnigMachine pre-assembles everything for robustness and offline operation. In addition, experienced Unity users can also bypass the web editor and directly create or edit adventures within the Unity template project, allowing for deeper customization, such as altering background visual effects or fonts or adding entirely new phase types.

4. Experimentation on gameplay experience

One of the main objectives of the design and development of this work is to enable the creation of augmented reality video games that combine, on the one hand, exploration of the environment and, on the other hand, a satisfying game experience that is motivating and entertaining for users while encouraging discovery and enjoyment of the different elements of the museum. To achieve this, a study was conducted to evaluate game experience using different adventures generated with the tools described in this paper. This evaluation aims to observe and analyse user feedback in order to identify strengths and weaknesses in the balance between enjoyment and difficulty of the generated games.

In the last decade, game research has become increasingly prominent in human–computer interaction studies, leading to the development of several self-assessment instruments aimed at evaluating gameplay experience. Among others, the Game Experience Questionnaire (GEQ) and its variants [45] have been extensively employed by practitioners across a wide spectrum of genres, interactive experiences and user types [46,47]. Due to this popularity in the literature, the in-game GEQ was used here, administered to the players after completing the adventure assigned to them. Key aspects such as level of immersion, fluency, positive and negative affect, challenge, tension, and anxiety were assessed. Additionally, information was collected on users' previous experience with escape rooms along with basic demographic data.

Regarding the design of the experiment, two variations of an adventure game set on the third floor of the García Santesmases Museum, dedicated to the history of computer science and located at the Faculty of Computer Science of the Complutense University of Madrid, Spain, were constructed. These were then administered to different groups of users working in teams of 2 to 4 players. The adventures designed comprised a total of 8 blocks of puzzles, making use of most of the types of phases available in the authoring tool. Each of these puzzle blocks

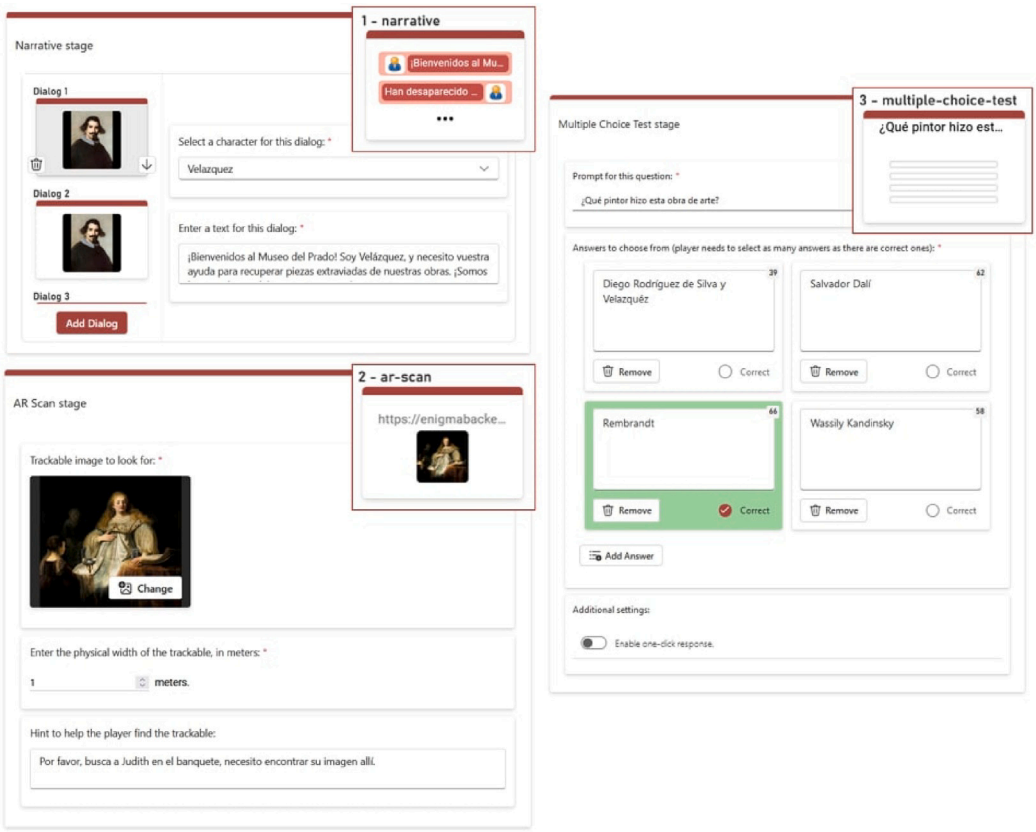


Fig. 1. Sample Editors for *Narrative*, *Multiple Choice*, and *AR-Scan* Stages.

required the careful inspection of at least one artefact or collection of artefacts in the museum and were distributed in two *Room*-type stages, each confined to one of the wings of the area where the activity took place. As mentioned above, the 4 blocks in each room could be completed in any order, being shown to the players in a random arrangement in each game.

The experiment was carried out across multiple sessions with a total of 62 participants, including 17 females, 44 males, and one person who chose not to answer the question. The mean age was 24.58 years (standard deviation = 3.05; minimum age: 19 years, maximum age: 34 years). Among the participants, 61.3% had little or no experience with escape rooms, 29% had moderate experience, and 9.7% considered themselves skilled players. Advertisements targeting our educational institution's design, programming, production, and art students were used to recruit participants, and their involvement was scheduled into one-hour blocks of museum visits.

In each of these sessions, participants were randomly split into teams of 2 to 4 people, each receiving a tablet with the pre-installed game, as well as sheets of paper and writing pens to take notes if they deemed it necessary. Initially, given that the museum can receive several groups of visitors at the same time, the simultaneous participation of multiple teams in the same time slot was allowed in cases where time restrictions required it. However, to avoid overcrowding of players around certain key points in the museum or having groups blindly following others to copy their answers, alternative versions of the game were administered in these sessions, with randomized sections and distinct starting points for each team.

Once the equipment and materials had been prepared, players were free to interact with the game in the corresponding section of the museum, with a one-hour time limit, after which they were instructed to return to their starting point to conclude the activity. These sessions were designed with competition in mind, by announcing that the first players to complete the escape room would receive a small reward in order to encourage immersion and rivalry between teams. After finishing the adventure, users were asked to submit the 14 items of the In-Game GEQ questionnaire [45].

Table 2
GEQ results by dimension (N = 62).

Dimension	Mean	Std	Median
Competence	2.69	0.79	2.75
Sensory and imaginative immersion	2.06	0.72	2.0
Flow	2.12	0.97	2.0
Tension	0.46	0.59	0.5
Challenge	2.36	0.78	2.5
Negative affect	0.29	0.46	0.0
Positive affect	3.28	0.66	3.5

Participants rated each item on a five-point Likert scale, where a value of 0 indicated complete disagreement and a value of 4 indicated complete agreement. The GEQ components of the game are grouped into pairs of items categorized as follows: Competence (items 2 and 9), Sensory and Imaginative Immersion (items 1 and 4), Flow (items 5 and 10), Tension (items 6 and 8), Challenge (items 12 and 13), Negative Affect (items 3 and 7), and Positive Affect (items 11 and 14). In addition, and on a voluntary basis, each participant had the opportunity to provide an open text commentary outlining what they enjoyed most and least about the experience, as well as their personal qualitative evaluation of the session. The only incentive for participants to take part in the study was the opportunity to try out an escape room with augmented reality elements in the museum, with minor prizes for the first ones to complete it.

4.1. Results

Overall results are reported below following data collection and questionnaire analysis. The distribution of GEQ scores by dimension

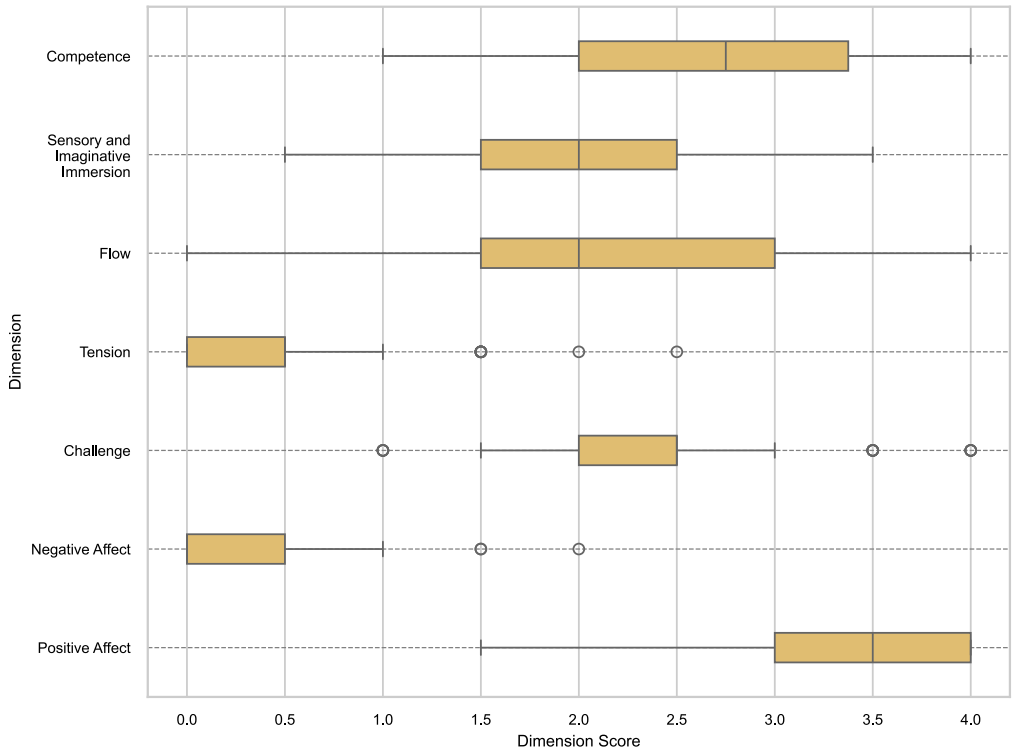


Fig. 2. Distribution of GEQ Scores by Dimension.

is represented visually in Fig. 2, with the aggregates for mean score, median, and standard deviation detailed in Table 2. In general, most players exhibited very low levels of stress and negative affect, with high levels of positive affect with the experience. Perceived levels of challenge were moderate for much of the population, as were levels of competence, flow, and sensory and imaginative immersion.

Of the latter, the results for flow and immersion merit closer inspection, as they report the weakest overall minimum and mean scores (mean for flow = 2.12, mean for immersion= 2.06). If we look at the sub-dimensions that make up these components, we can see that the worst results are found in item 1 ("I was interested in the story of the game", mean = 1.79) and item 12 ("I felt completely absorbed", mean = 1.97). Analysing the open-ended user responses, we can trace the origin of these results back to the following key points:

- A number of users note their preference for shorter textual explanations, with a greater focus on puzzles than on narrative. In other instances, players report favouring other multimedia formats such as video or sound over text in order to engage with the story of an escape room.
- Several players found the sharing of the museum space with other teams participating in the experience to be a source of friction, especially when more than one group was simultaneously tasked with solving puzzles located in close proximity to each other in the exhibit.
- Some participants described a degree of friction over usability issues such as not being able to go back in narrative stages to revisit previous messages. Furthermore, a significant number of the concerns regarding immersion were linked to the visual and auditory aspects of the game, which they found unattractive and consequently less immersive than expected.

Other general trends of a more positive nature include:

- The vast majority of participants reported not feeling at all tired or irritable during the experience. Perceived frustration was also very low, albeit marginally higher than the other two negative values as a consequence of the aforementioned points.
- A large number of users reported feeling very good about the experience, with only 5 users expressing feeling less than moderately successful. No participant submitted the minimum score for the item "I felt skilled".
- No participant answered below "Moderately" for the items "I felt satisfied" or "I felt good".

Additionally, we sought to analyse whether the perceived game experience varied significantly depending on the users' prior experience with escape rooms. Of all participants, 61.3% reported having little or no previous experience with escape rooms, with the other 38.7% considering themselves intermediate or advanced players. Given the ordinal qualitative nature of the variables studied, a non-parametric analysis was used to evaluate whether there were significant differences between the variables measured. Due to the samples' independence, the Mann–Whitney U test [48] was used to compare medians. The only dimension where a statistically significant difference was detected between the groups with and without previous escape room experience ($p < 0.05$) was for the challenge component ($U = 657$, $p = 0.001$, median without experience = 2.5, median with experience = 2).

5. Experimentation on authoring tool usability

Authoring tools are a critical part of the creation and research of interactive experiences, impacting the authorship process and the resulting works. While numerous tools dedicated to different types of multimedia artefacts exist, comparably limited work has been done to evaluate the user experience aspects of these applications relative to the effort invested in the assessment of the actual artefacts generated with these tools. In the field of IDN (interactive digital narrative) authoring tools, for instance, a 2023 literature review [19] revealed that only 7 out of the 37 authoring tools examined had conducted a thorough UX evaluation, with 26 of them having simply introduced the tool along with some examples and demonstrations, but no usability analysis to further validate them with intended end-users.

This same study raises possible hypotheses for the reasons behind this observation, emphasizing the following fundamental points:

- It is challenging to break down a complex creative process for task-based usability tests.
- Content authoring is often a much lengthier process than typical use cases in the systems for which conventional usability analyses are designed, and short-term evaluations are consequently not representative of real tool usage.
- Quantitative metrics are useful but, by themselves, insufficient to truly understand the experience of creatives.

Completing a series of mechanical tasks where a clear goal exists for the user is effectively not directly comparable to dedicating an extended period of time to operating a tool to design multimedia content. This is not to say, however, that traditional usability analyses are of no value, but it does highlight the need for more comprehensive studies that consider the properties of the system both when used to perform mechanical, time-bound, and domain-bound tasks and to develop end-to-end projects.

For these reasons, in this section we discuss the experiments conducted to evaluate the user experience in EnigMachine Editor in both settings, starting with a preliminary usability analysis in short-term sessions with bounded tasks and a modest group of participants in Section 5.1 and continuing with a roughly one-month-long evaluation of a creative task in Section 5.2, with a significantly larger participant group. The results of these studies, both quantitative and qualitative, are then reported in detail in Section 6.

5.1. Evaluating usability in design implementation tasks

Our second hypothesis raised the question of whether a user without a technical background or prior familiarity with the tool's concepts is generally able to use EnigMachine Editor for what we refer to as "design implementation tasks". By this, we denote tasks in which a user "translates" a set of design ideas into a working prototype, without these being necessarily devised by the user, assuming that everything included in these ideas is feasible to implement with the tools provided. Conceptually, this corresponds to the "short-term" system usage discussed at the beginning of this section. To assess this property, a study was conducted to analyse the ability of non-programmer users to convert a design document into a complete adventure using EnigMachine Editor.

This evaluation aims, on the one hand, to observe user feedback to understand the possible frictions and mechanical complexities perceived when attempting to translate a design concept into a functional element on the platform and, on the other hand, to inspect the games created by these users to determine the degree of success achieved in the implementation process. For this purpose, the System Usability Scale (SUS) questionnaire [49], administered to the participants after the end of the testing sessions, was used in tandem with Single Ease Question (SEQ) questionnaires [50] after each task completed by the users in a first tutorial stage at the beginning of the experiment. Both of these evaluation systems have been repeatedly demonstrated to be valid, sensitive and reliable for UX evaluation. Key properties such as ease of use, complexity, consistency, confidence in use, and the learning curve of the system were assessed.

The experiment was designed as a practical session in the context of the Master's Degree in Video Game Design at our educational centre, with the participation of 14 individuals, comprising 5 women and 9 men, with a mean age of 24.88 years (standard deviation = 2.31; minimum age: 22 years, maximum age: 34 years). The participants were university design students, most with little or no programming experience, and with non-technical profiles. All data collected during the session were anonymized prior to analysis.

The session began with a brief introduction aimed at contextualizing the activity and the tool. This was followed by a self-guided tutorial integrated into the web application, consisting of a battery of tasks to introduce the most important sections of EnigMachine Editor. This tutorial was comprised of 13 tasks presented through textual prompts, allowing the user the option to quit any of them if they felt unable to complete it successfully. Once a task was completed, and before proceeding to the next task, the system displayed a single-item SEQ questionnaire with the prompt, "Overall, how difficult or easy was it to perform this task?". Responses to this question were given on a 7-point Likert scale, where a value of 1 indicated great difficulty in completing the task and a value of 7 indicated great ease.

Once the tutorial phase was completed, participants were given a draft design document for the game used in the experiments from Section 4, outlining the structure of the game flow and the different puzzles to be incorporated, referencing the museum artefacts needed to overcome each challenge. In addition, they were provided with a reference adventure, which was blank save for having the necessary assets pre-included to build the game (i.e., images required for AR events or to define characters, for instance). After the material was provided, participants were given 2 h to translate the design document into a working implementation using the authoring tool. Subsequently, and for each participant, the number of essential phases (i.e., ignoring cosmetic or split narrative phases) correctly translated into EnigMachine Editor from the design document in the time allotted was computed. We consider that a minimum implementation of the draft script includes at least 18 stages, including those nested in room-type phases.

Finally, and briefly before concluding the session, participants were asked to fill in the SUS questionnaire designed to assess the overall usability perceived by users during the experiment. This questionnaire consists of 10 items to assess components such as ease of use, complexity, consistency, confidence in use, and the learning curve of the system.

Participants rated each item on a 5-point Likert scale, where a value of 1 indicated complete disagreement and a value of 5 indicated complete agreement. In addition to analysing the results for each of the questions independently, the SUS score of each user was computed following the formula proposed in the original SUS manuscript [49]. Additionally, and on a voluntary basis, each participant could write a free text commentary elaborating in more detail how they had felt about the tool, both positively and negatively.

Section 6 enumerates the question sentences included in the survey and details the result of the evaluation.

5.2. Evaluating usability and user experience in creative authoring

The final research issue of this paper raises the question of whether EnigMachine Editor is a generally useable tool for non-technical audiences to design interactive experiences directly and from scratch from the web platform itself over a longer period of time. The main difference between this use case and the one from the previous section is that we are now considering situations where the user is not a priori handed a set of design ideas for which a translation is guaranteed to exist in the tool but a much more ambiguous and creative task for which they are responsible for crafting a comprehensive design solely employing the features provided by EnigMachine.

Our hypothesis prior to the start of the experiments was that this type of evaluation would yield overall weaker usability results than those observed in the design implementation tasks, as a consequence of the potential inability to translate certain ideas from the creative process into the palette of features provided by the system, and the issues with traditional usability evaluation assumptions outlined at the beginning of this section. Here we considered hypothetical situations such as users wishing to include conditional flows, re-skinning assets from the final game, or making more complex puzzles than those available in EnigMachine.

This experiment was set up as a practical project in the video game design subject of the video game development undergraduate degree at our educational centre, with the participation of 50 individuals. The participants were first-year undergraduate students, most of whom had

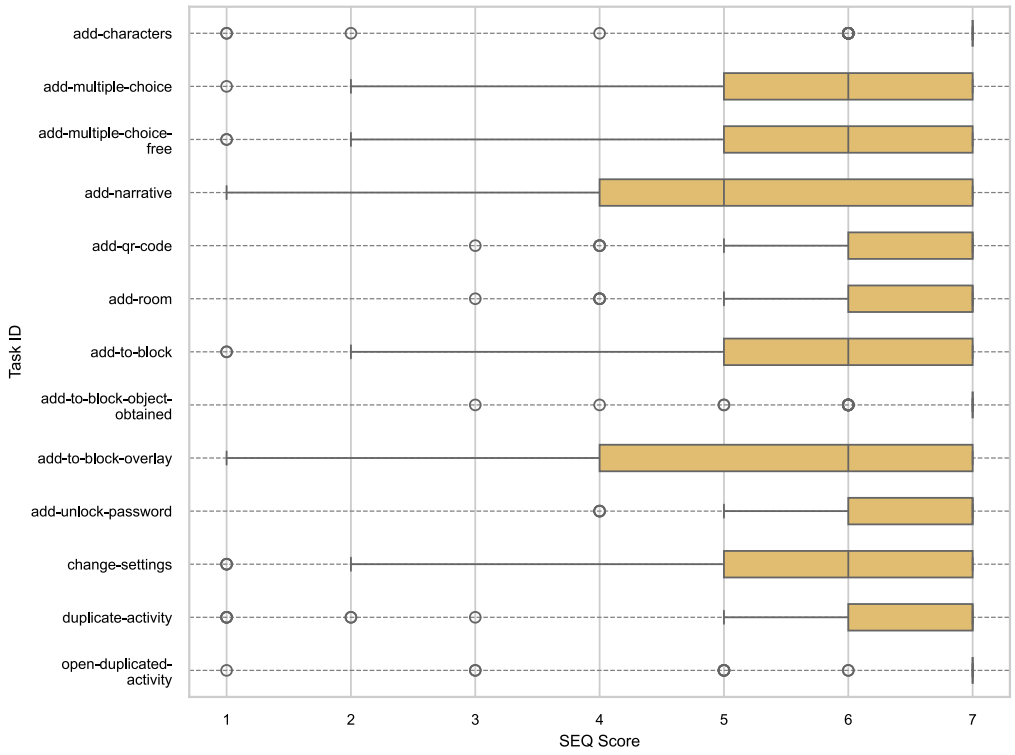


Fig. 3. Distribution of SEQ Scores by Task.

Table 3
Overview of evaluation types conducted with EnigMachine.

Evaluation type	Setting and length	Questionnaires
Game Experience of EnigMachine Games	García Santesmases Museum visitors, one-hour sessions	GEQ, Open-ended questions
Short Term Usability of EnigMachine Editor	Design Students (bachelors and masters), three-hour sessions	SEQ, SUS, Calculated Accuracy
Long Term Usability of EnigMachine Editor	Design Students (bachelors), one-month free design	SUS, Open-ended questions

little or no programming experience at the time the experiment was carried out.

In this context, the participants, in groups of 3 to 4 students, and over a period of time of roughly one month, were tasked with using the web editor to build their own adventures based on specific areas of the García Santesmases Museum, which was the same as the one used for the experiments in Section 4. During this time, participants were granted access to the museum venue to take photographs and gather information about the different artefacts available in the exhibitions, with the aim of constructing rooms, narratives, and puzzles that would motivate visitors to inspect the museum's objects carefully during the game. In the final stages of construction of these adventures, and on the occasion of a series of visits to the museum in the context of open days for secondary schools, a set of immersive tours was organized using the games created by those students interested in having live beta testers for their games. These were organized in 3 sessions of 37, 45, and 23 visitors, respectively, coming from a total of 6 educational centres in our city and mostly between 15 and 18 years old.

The usability questionnaires employed, as well as the tutorial and tool contextualization sessions in this experiment, were identical to those in Section 5.1. Thus, the initial phase of both groups was the same, with both contributing to the tutorial data with individual tasks linked to SEQ surveys, with the main differences being located in the second phase of the experiment. In this latter case, all participants were administered the SUS questionnaire after the end of the period allotted to complete their adventures.

An overview of the different evaluations conducted with our system can be found in Table 3.

Table 4
SEQ results for tutorial tasks (N = 73).

Task ID	Mean	Std	Completion %
add-characters	6.53	1.20	95.89
add-multiple-choice	5.90	1.38	90.41
add-multiple-choice-free	5.70	1.50	82.19
add-narrative	5.22	1.57	94.52
add-qr-code	6.42	0.94	89.04
add-room	6.25	0.98	87.32
add-to-block	5.72	1.56	85.51
add-to-block-object-obtained	6.67	0.75	94.03
add-to-block-overlay	5.33	1.74	86.76
add-unlock-password	6.49	0.79	93.06
change-settings	5.75	1.53	94.52
duplicate-activity	6.15	1.65	100.0
open-duplicated-activity	6.64	1.06	100.0

6. Results

The results obtained in the usability evaluations on design implementation and creative authoring tasks are presented in this section after data collection and questionnaire analysis.

Starting with the data collected from the tutorials, looking at the joint results for both groups in Table 4 (and a visual distribution in Fig. 3), it is possible to gather detailed insights into the specific tasks that lead to the most conceptual problems. Note that responses are given on a scale of 1 to 7 following the SEQ Likert format. It is noteworthy here that the most challenging assignments in terms of both SEQ and success rate are those related to modifying room-type stages by incorporating and editing new nested blocks within them. This is also to be expected, given that dealing with a nested structure is clearly more complex than working with individual stages. Multiple-choice questions also introduce a certain degree of fiction, especially those with free-response answers. Nonetheless, the vast majority of the tasks exhibit satisfactory results, with no more than a small set of users reporting low SEQ values in any of them.

Table 5
Test results for SUS prompts.

Prompt		Median	U	p-value
1. I think that I would like to use this system frequently.	DES IMP	3.0 3.0	154.0	0.12
2. I found the system unnecessarily complex.	DES IMP	2.0 1.0	342.0	0.01
3. I thought the system was easy to use.	DES IMP	4.0 5.0	104.5	0.01
4. I think that I would need the support of a technical person to be able to use this system.	DES IMP	2.0 1.0	287.0	0.16
5. I found the various functions in this system were well integrated.	DES IMP	3.0 4.0	82.5	0.0
6. I thought there was too much inconsistency in this system.	DES IMP	3.0 1.0	382.5	0.0
7. I would imagine that most people would learn to use this system very quickly.	DES IMP	4.0 4.0	186.5	0.36
8. I found the system very cumbersome to use.	DES IMP	2.0 1.0	305.5	0.07
9. I felt very confident using the system.	DES IMP	3.0 4.0	72.5	0.0
10. I needed to learn a lot of things before I could get going with this system.	DES IMP	2.0 2.0	228.0	0.96

From an overall usability point of view, the SUS Score was computed for each of the users following the SUS Score formula in [49] (which outputs a score between 0 and 100), resulting in a mean of 80.83 (standard deviation = 15.31, median = 82.5) for the design implementation task group, and 63 (standard deviation = 14.86, median = 65) for the creative authoring task group. According to a review including data from 241 industrial usability studies [51], the usability of the system in the first task group would approximately place in the 90 to 95 percentile range (grade A in the study), with the second group placing slightly below the industry average of 68 points (grade C in the study) and would benefit from improvements in the usability of the system. This corresponds to the expectations prior to conducting the experiments, taking into account that the design implementation tasks are clearly simpler and narrower in scope than those of the second test, and also more similar to the typical usability analysis setting focusing on specific tasks instead of wholistic, long-term creative evaluations.

Instead of focusing solely on an overall usability index, the participants' responses to each of the SUS questionnaire items are broken down below to help identify where the system's usability succeeds and fails. In what follows, note that item scores are given on a scale of 1 to 5 following the SUS Likert format. Given the ordinal qualitative nature of the variables under enquiry, a non-parametric test was used to determine whether significant differences existed between the variables examined. Due to sample independence, the Mann–Whitney U test was utilized to compare medians. Table 5 suggests statistically significant differences ($p < 0.05$) for items 2, 3, 5, 6, 8, and 9, but no evidence for this in the other questions. Note that in this table, "IMP" stands for the design implementation group and "DES" for the long-term creative design one. We discuss these trends in greater detail below.

- While significant differences are discernible, neither group consistently found the system unnecessarily complex (question "2. The system was unnecessarily complex", with means of 1.44 and 2.16 for implementation and creative authoring, respectively), nor very complicated to use (question "8. I found the system very complicated to use", means of 1.33 and 1.84). In practice, only one participant in both groups rated the latter question with a value above 3.
- While both groups agree on the overall ease of use of the system (question "3. I found the system easy to use", means of 4.56 and 3.7), those assigned with design implementation tasks found it significantly easier than those with creative authoring ones. As discussed above, this is fitting in that the second group was given a significantly less constrained assignment with a higher degree of freedom, where no obvious correspondence between design intent and tool feature existed.
- In line with the previous pattern, the implementation group felt significantly more confident than the creative authoring group (question "9. I felt very confident using the system", means of 4.33 and 3.00), which is explainable in light of the more guided nature of the first experiment.
- Lastly, the creative group perceives a significantly higher level of inconsistency in the system (question "5. I found the various functions of this system to be well integrated", with means of 4.22 and 2.98, and question "6. There was too much inconsistency in this system", with means of 1.44 and 2.76). While this point requires further analysis, we hypothesize that extensive use of the tool enables the user to explore it more thoroughly and engage with it closely enough to spot areas involving tedious sequences of steps or usability patterns that are dissonant with either their mental model or with the rest of the application's elements.

The non-significant items yield generally desirable system properties from the users' perspective. The majority of participants do not anticipate that they would require technical support to work with the tool (question "4. I think I would need the support of a technician to be able to use this system", with means of 1.56 and 1.96), consider the system easy to learn (question "7. I would imagine that most people would learn to use this system very quickly", means of 4.0 and 3.78), and do not perceive a high entry barrier (question "10. I had to learn a lot of things before I started using this system", means of 2.22 and 2.14).

Lastly, for the design implementation experiment, the success rate for each of the users was computed as the number of items correctly translated from the design document to the authoring tool in the time provided, divided by the minimum number of steps required to represent the proposed adventure, resulting in a favourable success rate of 85.8%.

On a qualitative level, the open-ended text comments of the participants of the creative authorship experiment can be summarized in the following key points:

- Designers request the inclusion of additional settings to customize the interface of the resulting game and enhance player immersion. In some instances, these suggestions also include comments aimed

at introducing mechanisms to customize the music and sound effects of the generated adventure, as well as the backgrounds, colours, and other visual elements of each stage.
- Another prevalent request is the inclusion of flow control systems that allow for the addition of a notion of "consequences for good and bad decisions" to the adventure. This includes ramifications on the game's narrative or scoring systems based on player performance.
- Designers working in groups express a demand for tools geared towards collaborative and structured game design. In the current version, they argue that while it is easy to work with EnigMachine Editor to implement a design, the platform does not provide suggestions or design ideas in the way that brands such as Kahoot or PowerPoint do, delegating the full weight of the design to the user.

7. Conclusions and future work

This paper presented an authoring tool designed to facilitate the development of AR-enabled adventures in cultural heritage contexts. The main goal is to empower content creators with tools that allow them to generate rich and immersive experiences, minimizing friction with the system and providing designers with an authoring process that is as useable and accessible as possible.

The three hypotheses underpinning this study concerned the game experience of the adventures generated with the EnigMachine system and the usability of the tool for users with non-technical profiles when creating experiences with and without prior design concepts. The results obtained provide information about the quality of our ecosystem in these three dimensions as perceived by users and players.

Hypothesis H1 was confirmed, as the adventures built with the system were found to exhibit good metrics in positive affect, flow, and competence, with low levels of stress and negative affect and a moderate level of challenge. The most critical points raised in the analysis were the need to introduce improvements aimed at enhancing the sensory and imaginative immersion, either through mechanisms to prevent the problem of excessive crowding of groups in the museum space or via refinements to the multimedia elements, giving more importance to visual and audio immersion and de-emphasizing narrative.

Hypothesis H2 stated that non-technical users would be able to use the tool to transform a design script into a playable adventure. The results confirmed this hypothesis, as these users reported positive usability levels for most of the dimensions evaluated. This was not necessarily the case in the context of hypothesis H3, which considered the situation where designers would have to use the tool to create complex designs over an extended period of time and without a reference script. In the latter, general usability metrics, without being too poor, were found to show room for improvement when compared to commonly accepted industry standards.

One of the most prominent of these points is the general desire of designers to feel explicit control over the aesthetics and the visual and acoustic aspects of the system, in line with the findings of the game experience study. Additionally, users expect the system to go beyond providing a platform for the implementation of ideas by offering mechanisms to support actual design from within the tool itself. This observation is consistent with the expectations outlined at the beginning of Section 5.2, where we argued that the concessions made to the authoring system in order to simplify the creative process could prove limiting for users; while we regard the reduction in expressivity as a necessary compromise to ensure the democratization of these authoring platforms to the general public, these insights allow us to identify those points where relaxing the rigidity of these concessions may be desirable. It is also important to note that games of this complexity are usually constructed on the foundation of a proper design document, external to the development engine itself.

In general, the results of the study support the usability of the evaluated tool and user satisfaction with the games derived from it. However, some areas for improvement were highlighted, such as a greater focus on the introduction of design support mechanisms or more freedom of visual and acoustic configuration for creatives. Study participants showed consistent enthusiasm for both the games and the authoring tool, expressing a desire to continue using them to craft new digital experiences in the future. These findings support the idea that our EnigMachine ecosystem has the potential to be an effective and accessible tool for building immersive activities with low development costs in the domain of cultural heritage.

These findings also have implications for other developers interested in producing similar tools and games, who can use this knowledge to devise more engaging and easily adoptable games and authoring systems for non-technical users, as well as for researchers, who can continue to explore the impact and challenges of these authoring pipelines in related institutions. One consideration for future research, however, is the evaluation of the authoring process by domain experts—e.g., museum curators, tourism officers, and so forth—as these latter are closer representations of the intended users of the platform. While having undergraduate students with a game design background (and some programming skills) provided valuable early insights, involving actual domain experts in subsequent evaluations will be crucial to more effectively establish the tool's suitability and usefulness for its intended users.

In the future, our main goal is to improve the overall satisfaction among users of the tool and the gaming experience of players interacting with the museum applications. All the insights described here will be taken into account in future development iterations, as well as considering the integration of new methodologies such as large language models to further support the creative process, following the trend of other authoring platforms such as Kahoot! or Notion. Meanwhile, and also stemming from the input of the study participants, we contemplate including new phases and interaction systems into the system to enrich the spectrum of options available to the user, especially in terms of increasing variety in the game flow, such as inventory systems, free exploration, branching narratives, etc. By investigating different interaction modalities and editing interfaces, we believe that developers and researchers can continue to innovate and improve the design of interactive content authoring tools.

CRediT authorship contribution statement

Pablo Gutiérrez-Sánchez: Writing – original draft, Validation, Writing – review & editing, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Pedro P. Gómez-Martín:** Writing – review & editing, Writing – original draft, Supervision, Software, Resources, Methodology, Investigation, Data curation, Conceptualization. **Pedro A. González-Calero:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization. **Marco A. Gómez-Martín:** Supervision, Software, Resources, Project administration, Methodology, Funding acquisition, Conceptualization, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Marco A. Gomez-Martn reports financial support was provided by Spanish Ministry of Science and Innovation (PID2021-123368OB-I00). No other conflicts of interest to report, except for those corresponding to the authors own affiliations (Complutense University of Madrid). If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the Spanish Ministry of Science and Innovation (PID2021-123368OB-I00).

Data availability

Data will be made available on request.

References

[1] P. Koutsabasis, Empirical evaluations of interactive systems in cultural heritage: A review, Int. J. Comput. Methods Herit. Sci. 1 (1) (2017) 100–122.
[2] M. Mortara, C.E. Catalano, F. Bellotti, G. Fiucci, M. Houry-Panchetti, P. Petridis, Learning cultural heritage by serious games, J. Cult. Herit. 15 (3) (2014) 318–325, http://dx.doi.org/10.1016/j.culher.2013.04.004, URL: https://www.sciencedirect.com/science/article/pii/S1296207413001349.
[3] I. Paliokas, S. Sylaiou, The use of serious games in museum visits and exhibitions: A systematic mapping study, in: 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games), Barcelona, Spain, 2016, pp. 1–8.
[4] I. Malegiannaki, T. Daradoumis, Analyzing the educational design, use and effect of spatial games for cultural heritage: A literature review, Comput. Educ. 108 (2017) 1–10, http://dx.doi.org/10.1016/j.compedu.2017.01.007.
[5] I. Khan, A. Melro, A.C. Amaro, L. Oliveira, Systematic review on gamification and cultural heritage dissemination, J. Digit. Media Interact. 3 (8) (2020) http://dx.doi.org/10.34624/jdmi.v3i8.21934.
[6] B. DaCosta, C. Kinsell, Serious games in cultural heritage: A review of practices and considerations in the design of location-based games, Educ. Sci. 13 (1) (2023) http://dx.doi.org/10.3390/educsci13010047, URL: https://www.mdpi.com/2227-7102/13/1/47.
[7] I. Camps-Ortueta, P.A. González-Calero, M.A. Quiroga, P.P. Gómez-Martín, Measuring preferences in game mechanics: Towards personalized chocolate-covered broccoli, in: E.D.V. der Spek, S. Göbel, E.Y. Do, E. Clua, J.B. Hauge (Eds.), Entertainment Computing and Serious Games - First IFIP TC 14 Joint International Conference, ICEC-JCSG 2019, Arequipa, Peru, November 11-15, 2019, Proceedings, in: Lecture Notes in Computer Science, vol. 11863, Springer, 2019, pp. 15–27, http://dx.doi.org/10.1007/978-3-030-34644-7_2.
[8] C. Alexander, A Pattern Language: Towns, Buildings, Construction, Oxford University Press, 1977.
[9] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.
[10] S. Björk, J. Holopainen, Patterns in Game Design, Charles River Media, 2005.
[11] L. Rau, J.L. Bitter, Y. Liu, U. Spierling, R. Dörner, Supporting the creation of non-linear everyday AR experiences in exhibitions and museums: An authoring process based on self-contained building blocks, Front. Virtual Real. 3 (2022) http://dx.doi.org/10.3389/frvir.2022.955437, URL: https://www.frontiersin.org/journals/virtual-reality/articles/10.3389/frvir.2022.955437.
[12] S. Nicholson, Peeking Behind the Locked Door: A Survey of Escape Room Facilities, Technical Report, 2015, URL: http://scottnicholson.com/pubs/erfacwhite.pdf. (Visited 28 June 2024).
[13] M. Wiemker, E. Elumir, A. Clare, Escape room games: "Can you transform an unpleasant situation into a pleasant one?" in: J. Haag, J. Weißenböc, M. Gruber, M. Christian, F. Freisleben-Teutscher (Eds.), Game Based Learning, Fachhochschule st Pölten GmbH, Austria, 2015, pp. 55–68.
[14] P. Fotaris, T. Mastoras, Escape rooms for learning: A systematic review, in: European Conference on Games Based Learning, 2019, pp. 235–243, http://dx.doi.org/10.34190/GBL.19.179.
[15] E. González-Muñoz, T. Ouariachi, Unlocking learning: The impact of an educational escape room on energy transition, Electron. J. E-Learn. (EJEL) 22 (4) (2024) URL: https://doi.org/10.34190/ejel.22.4.3186.
[16] S. Clarke, D. Peel, S. Arnab, L. Morini, H. Keegan, O. Wood, EscapED: A framework for creating educational escape rooms and interactive games to for higher/further education, Int. J. Serious Games 4 (3) (2017) http://dx.doi.org/10.17083/ijsg.v4i3.180.
[17] R.E. Rawlinson, N. Whitton, Escape rooms as tools for learning through failure, Electron. J. E-Learn. (EJEL) 22 (4) (2024) URL: https://doi.org/10.34190/ejel.21.7.3182.
[18] D. Green, C. Hargood, F. Charles, Use of tools: UX principles for interactive narrative authoring tools, J. Comput. Cult. Herit. 14 (3) (2021) http://dx.doi.org/10.1145/3458769.
[19] C. Hargood, D. Green, The authoring tool evaluation problem, in: C. Hargood, D.E. Millard, A. Mitchell, U. Spierling (Eds.), The Authoring Problem: Challenges in Supporting Authoring for Interactive Digital Narratives, Springer International Publishing, Cham, 2022, pp. 303–320, http://dx.doi.org/10.1007/978-3-031-05214-9_19.
[20] C.E.K. Jones, S. Theodosis, I. Lykourentzou, The enthusiast, the interested, the sceptic, and the cynic: Understanding user experience and perceived value in location-based cultural heritage games through qualitative and sentiment analysis, J. Comput. Cult. Herit. 12 (1) (2019) 6:1–6:26, http://dx.doi.org/10.1145/3297716, URL: https://dl.acm.org/doi/10.1145/3297716.
[21] E. Garzón-Artacho, T. Sola-Martínez, J.-M. Romero-Rodríguez, G. Gómez-García, Teachers' perceptions of digital competence at the lifelong learning stage, Heliyon 7 (7) (2021) e07513, http://dx.doi.org/10.1016/j.heliyon.2021.e07513, URL: https://www.sciencedirect.com/science/article/pii/S2405844021016169.
[22] H. Lieberman, F. Paternò, V. Wulf, End user development, in: Human-Computer Interaction Series, vol. 9, Springer, Dordrecht, 2006.
[23] C. Fidas, C. Sintoris, N. Yiannoutsou, N. Avouris, A survey on tools for end user authoring of mobile applications for cultural heritage, in: 2015 6th International Conference on Information, Intelligence, Systems and Applications, IISA, 2015, pp. 1–5, http://dx.doi.org/10.1109/IISA.2015.7388029, URL: https://ieeexplore.ieee.org/document/7388029.
[24] C. Sintoris, N. Yiannoutsou, A. Ortega-Arranz, R. López-Romero, M. Masoura, N. Avouris, Y. Dimitriadis, TaggingCreaditor: A tool to create and share content for location-based games for learning, in: 2014 International Conference on Interactive Mobile Communication Technologies and Learning, IMCL2014, 2014, pp. 280–284, http://dx.doi.org/10.1109/IMCTL.2014.7011148, URL: https://ieeexplore.ieee.org/document/7011148.
[25] G. Ghiani, F. Paternò, L.D. Spano, Cicero designer: An environment for end-user development of multi-device museum guides, in: V. Pipek, M.B. Rosson, B. de Ruyter, V. Wulf (Eds.), End-User Development, Springer, Berlin, Heidelberg, 2009, pp. 265–274, http://dx.doi.org/10.1007/978-3-642-00427-8_15.
[26] M.T. Linaza, I. Torre, R. Beusing, A. Tavernise, M. Etz, Authoring Tools for Archaeological Mobile Guides, The Eurographics Association, 2008, URL: https://doi.org/10.2312/VAST/VAST08/047-054.
[27] M. Roussou, L. Pujol, A. Katifori, A. Chrysanthi, S. Perry, M. Vayanou, The museum as digital storyteller: ollaborative participatory creation of interactive digital experiences, 2015, http://dx.doi.org/10.13140/RG.2.1.3917.2889.
[28] F.A. Hansen, K.J. Kortbek, K. Grønbæk, Mobile urban drama: interactive storytelling in real world environments, New Rev. Hypermedia Multimed. 18 (1–2) (2012) 63–89, http://dx.doi.org/10.1080/13614568.2012.617842, URL: http://www.tandfonline.com/doi/abs/10.1080/13614568.2012.617842.
[29] A.M. Sánchez, Using digital educational escape rooms as a motivational review tool for economics, Int. J. Manag. Educ. 21 (3) (2023) 100852, http://dx.doi.org/10.1016/j.ijme.2023.100852.
[30] M. Wolf, M. Montag, H. Söbke, F. Wehking, C. Springer, Low-threshold digital educational escape rooms based on 360VR and web-based forms, Electron. J. E-Learn. (EJEL) 22 (4) (2024) http://dx.doi.org/10.34190/ejel.21.7.3182.
[31] Genially Web S.L., Genially, 2025, URL: https://genially.com/. (Accessed 30 January 2025).
[32] Buzzshot, Telescape live, 2025, URL: https://telescape.com/. (Accessed 30 January 2025).
[33] Breakout Inc., Breakout EDU, 2025, URL: https://breakoutedu.com. (Accessed 30 January 2025).
[34] ROOM ESCAPE MAKER - INPI, Room escape maker, 2025, URL: https://roomescapemaker.com. (Accessed 30 January 2025).
[35] Grupo de Internet de Nueva Generación, Escapp, 2025, URL: https://github.com/ging/escapp. (Accessed 30 January 2025).
[36] H. Desurvire, C. Wiberg, Master of the game: assessing approachability in future game design, in: CHI '08 Extended Abstracts on Human Factors in Computing Systems, ACM, Florence Italy, 2008, pp. 3177–3182, http://dx.doi.org/10.1145/1358628.1358827, URL: https://dl.acm.org/doi/10.1145/1358628.1358827.
[37] H. Desurvire, C. Wiberg, Game usability heuristics (PLAY) for evaluating and designing better games: The next iteration, in: A.A. Ozok, P. Zaphiris (Eds.), Online Communities and Social Computing, Springer, Berlin, Heidelberg, 2009, pp. 557–566, http://dx.doi.org/10.1007/978-3-642-02774-1_60.
[38] H. Korhonen, Evaluating playability of mobile games with the expert review method, 2016, URL: https://www.semanticscholar.org/paper/Evaluating-Playability-of-Mobile-Games-with-the-Korhonen/80e976e4571bed7a20bad52e209e365fd5532557.
[39] R. Andreoli, A. Corolla, A. Faggiano, D. Malandrino, D. Pirozzi, M. Ranaldi, G. Santangelo, V. Scarano, A framework to design, develop, and evaluate immersive and collaborative serious games in cultural heritage, J. Comput. Cult. Herit. 11 (1) (2017) http://dx.doi.org/10.1145/3064644.
[40] G.E. Raptis, C. Fidas, N. Avouris, Do game designers' decisions related to visual activities affect knowledge acquisition in cultural heritage games? An evaluation from a human cognitive processing perspective, J. Comput. Cult. Herit. 12 (1) (2019) 4:1–4:25, http://dx.doi.org/10.1145/3292057, URL: https://dl.acm.org/doi/10.1145/3292057.
[41] L. Pujol-Tost, Did we just travel to the past? Building and evaluating with cultural presence different modes of VR-mediated experiences in virtual archaeology, J. Comput. Cult. Herit. 12 (1) (2019) 2:1–2:20, http://dx.doi.org/10.1145/3230678, URL: https://dl.acm.org/doi/10.1145/3230678.
[42] M.K. Bekele, R. Pierdicca, E. Frontoni, E.S. Malinverni, J. Gain, A survey of augmented, virtual, and mixed reality for cultural heritage, J. Comput. Cult. Herit. 11 (2) (2018) 7:1–7:36, http://dx.doi.org/10.1145/3145534, URL: https://dl.acm.org/doi/10.1145/3145534.

[43] M. Konstantakis, G. Caridakis, Adding culture to UX: UX research methodologies and applications in cultural heritage, J. Comput. Cult. Herit. 13 (1) (2020) 4:1–4:17, http://dx.doi.org/10.1145/3354002, URL: https://dl.acm.org/doi/10.1145/3354002.
[44] M. Kuniavsky, E. Goodman, A. Moed, Observing the User Experience: A Practitioner's Guide to User Research, second ed., Morgan Kaufmann, Amsterdam ; Boston, 2012.
[45] W. IJsselsteijn, Y. de Kort, K. Poels, The Game Experience Questionnaire, Technische Universiteit Eindhoven, 2013.
[46] D. Janßen, C. Tummel, A. Richert, I. Isenhardt, Towards measuring user experience, activation and task performance in immersive virtual learning environments for students, in: C. Allison, L. Morgado, J. Pirker, D. Beck, J. Richter, C. Gütl (Eds.), Immersive Learning Research Network, Springer International Publishing, Cham, 2016, pp. 45–58.
[47] C. Marín-Lora, M. Chover, M.Y. Martín, L. García-Rytman, Creating a treadmill running video game with smartwatch interaction, Multimedia Tools Appl. 83 (19) (2024) 57709–57729, http://dx.doi.org/10.1007/s11042-023-17752-1.
[48] H.B. Mann, D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, Ann. Math. Stat. 18 (1) (1947) 50–60, http://dx.doi.org/10.1214/aoms/1177730491.
[49] J. Brooke, SUS – a quick and dirty usability scale, 1996, pp. 189–194.
[50] J. Sauro, J.S. Dumas, Comparison of three one-question, post-task usability questionnaires, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 1599–1608, http://dx.doi.org/10.1145/1518701.1518946, URL: https://dl.acm.org/doi/10.1145/1518701.1518946.
[51] Admin, Item benchmarks for the system usability scale - JUX, 2018, URL: https://uxpajournal.org/item-benchmarks-system-usability-scale-sus/.

Chapter 14

Initializing Interactive Treasure Hunts in Cultural Heritage Sites: An LLM-Based Approach

Initializing Interactive Treasure Hunts in Cultural Heritage Sites: An LLM-Based Approach

Pablo Gutiérrez-Sánchez[1(✉)], Pedro A. González-Calero[1], Marco A. Gómez-Martín[1], Pedro P. Gómez-Martín[1], and Ruck Thawonmas[2]

[1] Complutense University of Madrid, 28040 Madrid, Spain
pabgut02@ucm.es
[2] Ritsumeikan University, Ibaraki, Osaka 567-8570, Japan

Abstract. Cultural heritage sites are increasingly embracing playful approaches such as treasure hunt games to enhance interactive educational experiences, proving effective in engaging diverse audiences, encouraging exploration, and promoting knowledge retention. However, designing such experiences remains a complex task that requires careful integration of narrative, interactivity, and factual accuracy.

In this article, we propose an approach that leverages large language models (LLMs) to automate the initial drafting of treasure hunt games for cultural heritage sites. Our method allows content curators to specify key parameters—such as the target artefacts to be included in the hunt, intended audience, and narrative styles—after which the system generates a structured sketch of the game. We compare two generation strategies: a basic sequential method and a revised approach that incorporates a pre-planning phase. Our evaluation assesses the resulting drafts in terms of their correctness, consistency, and stylistic coherence.

Results suggest that pre-planning improves the quality of the generated content, producing generally more structured and contextually appropriate outputs. Moreover, we describe some of the remaining challenges, such as the need for interactive and validated co-design mechanisms or the introduction of factual accuracy guarantees in the adventures.

Keywords: Large Language Models · Educational escape room · Games at museums · AI-assisted game design · Automated content generation

1 Introduction

In recent years, cultural heritage sites have increasingly adopted mobile technology to enhance their repertoire of interactive educational experiences, complementing traditional museum tools and practices. Among these digital strategies,

Published by Springer Nature Switzerland AG 2025
M. Sugimoto et al. (Eds.): ICEC 2025, LNCS 16042, pp. 151–165, 2025.
https://doi.org/10.1007/978-3-032-02555-5_11

gamification has emerged as a powerful way to make cultural learning more engaging, as noted in numerous literature reviews [7,15,17–19].

A particularly effective gamified approach is the treasure huntâĂŞstyle game, typically implemented in the form of apps for mobile devices. These experiences engage users through interactive elements such as geolocated navigation stages, clue-following, and puzzle-solving. They may also incorporate object collection, customised quizzes, and AR-based scanning of QR codes or venue artefacts, showcasing new immersive alternatives to promote problem-solving and deeper engagement with local heritage [3,14].

Targeting a wide range of audiences, including children, school groups, university students, and general museum visitors, these gamified interventions have been proven to enhance knowledge retention [28], promote collaborative learning [8], and encourage exploration and contextual learning [5], positioning them as effective tools for cultural interpretation [3]. Despite their potential, the design and implementation of these experiences remain challenging, requiring a careful balance and integration of narrative, interactivity and technology.

To ease the design process, generative artificial intelligence models can provide assistance in brainstorming [30] or generating initial versions of these games. This is in line with what is already present within commercial interactive content creation platforms such as *Kahoot!*[1] or *Quizizz AI*[2], which offer simple mechanisms supported by Large Language Models (LLMs) as a source of inspiration and novel ideas. When developing content for cultural heritage sites, however, it is essential that the material produced remain factually grounded on the venue's knowledge base, often comprised of numerous texts with information regarding the artefacts of its distinct exhibits. Given the tendency of LLMs to hallucinate, ensuring factual accuracy is paramount, particularly in in learning-centred experiences.

In the past, we have developed *Enigmas* [10], treasure hunt type games for museums supported by AR technology, mixing narrative elements with clues, artefact searches, mini-games, and multiple-choice questions to test players on the content of the exhibition. Historically, designing such adventures has been time-consuming, particularly when it comes to formulating initial ideas for the structure of the experience. With this in mind, in this paper we propose an LLM-based approach to automate the production of these initial drafts in a limited subset of *Enigma* games. Our method involves content curators specifying key domain information, such as the artefacts to be included, the target audience, or the desired narrative style, after which the system generates structured data files to bootstrap the game. We compare two approaches: a baseline method using direct and sequential prompting and a revised version incorporating a pre-planning step. We then evaluate their effectiveness in terms of content correctness, stylistic appropriateness, and overall perceived quality.

[1] https://kahoot.com/.
[2] https://quizizz.com/quizizz-ai.

2 Related Work

The rise of LLMs has spurred interest in using these AIs for text generation across domains, including video games. Trained on large text corpora, LLMs generate responses based on user prompts, usually accompanied by a set of rules or conversation history. Without prompt-embedded domain knowledge, their responses rely entirely on input and training data, known as a zero-shot paradigm, whereas adding examples enables few-shot reasoning [29]. These models are not free of reasoning and action errors, however, which has given rise to different strategies to improve the generation quality by means of patterns such as chain-of-thought prompting [25] or ReAct (Reasoning and Acting) [27].

In video games, the use of LLMs has been explored for a variety of tasks including narrative creation, interactive stories, and scene generation [4,16,20]. This includes the production of games in the form of visual novels with complex branching that propagate over time [22], although some work reports that LLM-generated dialogues are not always perceived as positively as those written by humans [2]. They have also been applied in level design (e.g., Mario Bros [21] or Sokoban levels [23]) or for game environments and layouts [9,13].

For cultural heritage sites, LLMs have been used for guided tours and personalized narratives rather than traditional games. Helmy et al. [12] combine a 3D scene generation tool (NeRF) with the LLaMa 3 language model to power a virtual tour guide for Egyptian heritage sites. In turn, Trichopoulos et al. [24] use OpenAI's GPT model to generate personalized narratives and recommendations for museum visitors. These approaches use conversational interfaces to encourage user exploration and engagement.

In contrast to our work, existing studies—including those focused on cultural heritage applications—have not placed particular emphasis on the factual accuracy of the generated content. Moreover, our study differs from the literature in that a treasure hunt consists of several phases of different nature, each of which requires a certain level of internal consistency and factual accuracy. While existing generators create narratives, quests, questions, or environments in isolation, our approach integrates all components into a cohesive, unified narrative.

3 Games Description and Problem Definition

Enigma games are interactive adventures for mobile devices, inspired by escape rooms and treasure hunts and designed to engage players in story-driven experiences at cultural heritage sites such as museums or archaeological sites. Players assume the role of the protagonist, going through a series of interconnected puzzles, riddles, and quests. The primary goal is to motivate visitors to actively explore their surroundings and interact more deeply with the artefacts on display, dynamizing their experience and encouraging reflection on the content.

The gameplay is structured in distinct stages, each presenting a unique challenge, such as locating artefacts, solving puzzles, answering multiple-choice questions, or unlocking digital locks. These phases are sequential, requiring completion of one to proceed to the next. Players interact with the game via touchscreen,

GPS navigation, and augmented reality features, bringing the venue and its institutions to life. The aim is to foster a sense of immersion and active participation, making players feel like an integral part of the adventure. These games have been used at several institutions in Madrid, Spain, such as the National Museum of Natural Sciences, the Lázaro Galdiano Museum, and the García Santesmases Museum of computer science history.

In this work, we focus on three of the thirteen phases currently available in these games, which form the core structure of most Enigma adventures:

- *Narrative phases*: virtual characters interact via dialogues to tell a story or elaborate on concepts and events that take place in the game.
- *Artefact search phases*: players are prompted to explore the area to locate a museum item based on a treasure hunt-style clue.
- *Multiple-choice question phases*: the system presents questions with a correct answer and distractors to assess the player's understanding.

Typically, these three kinds of phases are introduced in this same order, giving rise to thematic blocks focusing on each of the artefacts to be discovered by the player: (1) A narrative stage introduces or continues the story and motivates the search for an artefact; (2) a search phase prompts the player to reason about the given clue and explore the area to locate the artefact; and (3) a question phase encourages further scrutiny of the work in search of the correct answer.

4 Approach

Our approach starts from a set of treasure-hunt configuration details provided by the game designer or content curator and generates a JSON file with the definition of an Enigma game based on the information provided (see Fig. 1).

4.1 Game Parameters

To build an adventure based on a solid knowledge of the museum's objects, the first essential step is to list the artefacts the designer wants to include in the game. If the museum already displays panels with textual descriptions of the works, these can be directly used as input. However, depending on the use case, it may be beneficial to enrich this data with other remarks such as information about nearby objects, metadata regarding the artefact's location, or any relevant facts that could help generate clues in the treasure hunt. An example of an artefact description from our case study is shown bellow in Sample 1 for the artwork "Apollo in the Forge of Vulcan".

"Apollo in the Forge of Vulcan": "Diego Velázquez's painting *Apollo in the Forge of Vulcan* (c. 1630) depicts the Roman god Vulcan at work in his forge, surrounded by assistants hammering metal in the intense heat. Vulcan, engaged in forging, represents both creation and destruction. Velázquez contrasts the warm glow of fire with cooler tones..." (1)

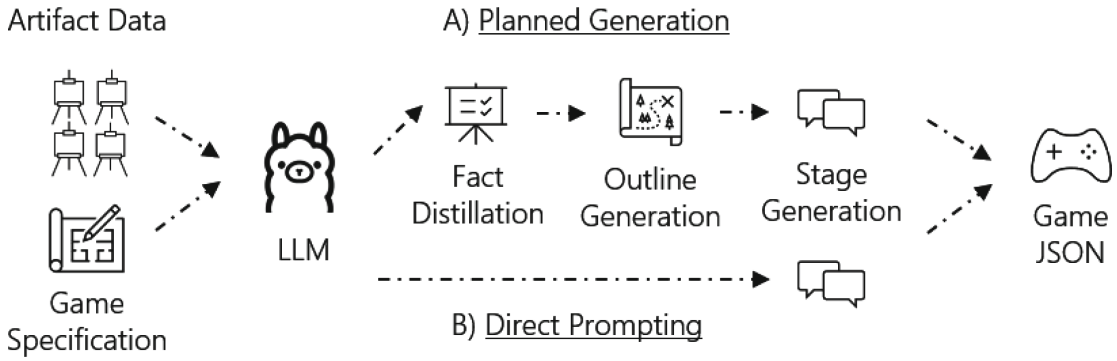


Fig. 1. Treasure Hunt generation flow for both approaches.

Table 1. Sample input for a "Mythology Mystery" specification.

Parameter	Input
`General Description`	A mythology-based adventure where players uncover the legendary stories hidden in paintings and sculptures
`Intention`	Provide a deep dive into mythology through artwork
`Fact Focus`	Myths and legends behind classical artworks
`Narrative Style`	A mysterious quest where players act as seekers of lost mythological knowledge
`Clue Style`	Riddles based on myths, e.g., "Seek the god who..."
`Question Style`	Questions exploring the accuracy of myths in art
`Target Audience`	Mythology and history enthusiasts
`Difficulty`	Hard

Once this information is provided, the rest of the parameters start to refer to stylistic and focus considerations both at the overall domain level and for the different phases of the adventure, as can be seen in the example from Table 1. Notably, parameters are provided to control the target audience (e.g., "Mythology and history enthusiasts"), the design intent, the expected difficulty, and a general description of the goal of the application. Numerical inputs to control other aspects of the game, such as the length of narrative blocks or the number of answers per multiple-choice question are also included.

4.2 Fact Distillation and Plan-and-Execute Strategy

Once the designer has entered this information, the game generation process begins. This starts by taking the texts for each of the artefacts and producing a reduced set of succinct facts with the focus indicated by the designer in the `Fact Focus` parameter. Given that the descriptions provided by these institutions can be quite lengthy, the first hypothesis (**H1**) of this study is that distilling the most

relevant facts from these descriptions improves the consistency and accuracy of the clues and questions generated by the LLM. The number of facts output per artefact is controlled by the `Number of Facts` input parameter. We preliminarily argue that this step may contribute to focusing the attention of the model on the pieces of information that the designer considers most pertinent in the given context. An example of the facts suggested by the model for the artwork "Apollo in the Forge of Vulcan" using the sample inputs above is shown below.

1. Velázquez's painting 'Apollo in the Forge of Vulcan' humanizes the Roman god Vulcan, depicting him as a labourer rather than an idealized deity, making it an allegory of craftsmanship, creativity, and transformation.
2. Velázquez masterfully contrasts the warm glow of fire with cooler tones, creating a dramatic atmosphere that blends mythology with naturalism.
3. Velázquez enhances the realism and depth of the scene through his use of light and shadow, showcasing his technical mastery.

During all stages of generation, we make use of Microsoft's Guidance library [1] to control the output of the LLM in a structured way. This ensures that it is never necessary to perform parsing operations on the data generated by the model and that, in general, there are always mechanisms in place to guarantee the structural integrity of the proposed solutions. For instance, it is possible to restrict the output of the LLM during narrative generations to pick from only valid characters available in the adventure metadata or to impose the generation of an exact number of single-sentence answers in the question stages.

Secondly, given that the generated adventures may become excessively long, this study proposes a second hypothesis (**H2**): that developing an action plan prior to phase generation supports narrative focus and coherence across the game's stages. To this end, once the most relevant facts of each work have been distilled, we prompt the model to prepare a structured draft in which the points to be followed throughout the adventure are briefly defined. This is in line with common design patterns in prompt engineering, such as Plan-and-Execute, where the model first plans a strategy for solving a complex problem and then executes each of the solving steps in a systematic way.

More specifically, such a plan involves designing a narrative-question-clue sequence for each of the artefacts, along the lines of what was discussed at the end of Sect. 3. For each of the works in the itinerary, the LLM must produce a fact on which to base the clue that leads the player to the piece, the point of emphasis on which to focus in the multiple-choice question to be asked after finding the artefact, and a summary of the narrative from the instant in which the previous object was found to the moment in which the player is to find the next one. Continuing with the example above, the general points generated as a sketch of the artefact block would be:

– **Narrative summary:** "as Kaida and Lyra delve into the world of ancient mythology, they discover that the mysterious collector's cryptic letter is linked to Velázquez's enigmatic painting "Apollo in the Forge of Vulcan", which holds the

key to understanding the transformative power of art and the connection between creativity and craftsmanship, setting the stage for their quest to uncover the secrets hidden within the masterpieces".
- **Focus of the clue:** "the transformative power of art and the connection between creativity and craftsmanship."
- **Focus of the question:** "the accuracy of myths in art by asking about the depiction of Vulcan in the painting".

Once this outline is available, and with it the general narrative, the model makes a proposal of the characters to be used, together with a brief description of each of them. In the ongoing example, proposed characters are **Kaida Asteria**, "a brilliant and enigmatic art historian with a passion for uncovering the hidden secrets of ancient mythologies", and **Lyra Flynn**, "a charismatic and resourceful adventurer with a knack for deciphering cryptic clues and unravelling mysteries".

4.3 Stage Generation

Having devised the plan to be followed, our approach then moves on to individually generate the stages of the game according to the general descriptions in the draft. Here, each of the designed stages is created from a sequence of prompts to the LLM in which the style settings, the focus to be followed, and the rules that must be generally respected in any stage of the corresponding type are injected. For instance, in question-type stages, one of the basic restrictions is that answers must not be repeated, while in clue-type stages, a clue must not explicitly reveal the name of the artefact to be searched for.

Structural and design constraints are a key aspect to take into account during the generation process. By the former, we refer to constraints on the shape of the produced content from a formatting point of view:

- A multiple-choice question must have exactly the requested number of possible answers, all of them consisting of a single sentence. The latter is also a restriction in the case of clues.
- A narrative stage can only reference characters previously defined in the game, must not exceed the maximum number of interactions per story block, and each of the interventions must be made up of a single sentence.

By design constraints, we denote rules that cannot generally be verified automatically as they are based on respecting the design intent of the person who specifies the adventure. Some examples for this paper's use case are:

- A narrative stage should never give overly obvious clues to the next artefact to be encountered in the game (e.g., the name of the artefact itself). Dialogues should not be repeated either directly or paraphrased, must respect the specified storytelling style, and must be cohesive and smoothly link the different stages of the game, especially in terms of providing the necessary context so that subsequent stages are sufficiently grounded.

- A clue stage should not explicitly reveal the name of the artefact to which it leads, but neither should it be so ambiguous or broad that the player has insufficient information to locate it. The level of difficulty, focus, and style should respect the design intentions.
- A multiple-choice question stage should not explicitly reveal the answer in the question, nor list the name of the artefact as one of the possible choices. Responses should not be repeated, either verbatim or in paraphrased but essentially equivalent versions. In addition, the correct answer should not be made more emphatic than the distractors (e.g., with an exclamation mark at the end). The level of difficulty, focus, and style should respect the design intentions. Most importantly, questions must always be correct and respect the factual data provided by the user's descriptions. The latter is particularly important in any adventure with a pedagogical focus.

With this, a final stage sequence for the running example would be:

- **Narrative:**
  Kaida: The symbol seems to be a variant of the ancient Greek character for the Muses.
  Lyra: That's a great observation, Kaida.
  Kaida: I think it's more than just a coincidence.
  Lyra: Then we need to find out what Velázquez's paintings have to do with it.
  Kaida: The paintings are said to hold the key to understanding the power of art.
- **Clue:** Seek the divine craftsman where fiery passion forges the very essence of art itself.
- **Question:** What is the primary representation of Vulcan in Velázquez's "Apollo in the Forge of Vulcan", according to the mythological context depicted in the artwork?
  - **Correct Answer:** Vulcan is portrayed as a labourer rather than an idealized deity, humanizing him and making the painting an allegory of craftsmanship, creativity, and transformation.
  - **Distractor 1:** The painting shows Vulcan as a powerful, godlike figure, emphasizing his divine nature and mythological significance.
  - **Distractor 2:** The artwork depicts Vulcan as a mere mortal, struggling to control the fiery forge, highlighting his vulnerability and humanity.
  - **Distractor 3:** The painting presents Vulcan as a symbol of destruction, emphasizing his role in the mythological context and the chaos he brings to the world.

While we write these examples here in a simplified form to improve readability, note that the output of the system is delivered as a JSON file containing the definitions of each of the phases, metadata, and character listings. This structure is repeated as many times as artefacts included in the game, with a final additional narrative block between the point at which the player finds the last artefact and the end of the game. All the prompts and Guidance code used for the generation can be found in the project repository[3].

[3] Repository Link: https://github.com/pgutierrez858/enigma-llms.

5 Experiments and Results

As introduced in Sect. 4, we aim to validate two hypotheses: (**H1**) distilling relevant facts about the artefacts to be encountered reduces the number of factual errors and improves the overall consistency of adventures; and (**H2**) applying a Plan-and-Execute strategy prior to the generation of game stages has an overall positive effect on design error rates. We evaluate this by comparing the planned approach with strategies that do not make use of these preliminary steps (direct) on 10 use cases available in the project repository, similar to the one in Table 1. These test cases simulate realistic scenarios for designing an interactive treasure hunt, varying in theme, focus, difficulty, and descriptions of the target audience and styles to be followed. A fixed sequence of 4 popular artworks from the Prado Museum was set as input for all use cases. All the experiments in this section were performed using a Llama 3.1 8B model, queried through Guidance calls, and on a PC with 16 GB of RAM and an Intel Core i7-9750H, 2.60 GHz processor.

For each of the generated stages, we define three possible creation errors: correctness failure (i.e., the phase includes false information), style failure (i.e., the phase does not follow the style stated in the input, or presents some form of generation error), and finally domain failure (i.e., the phase does not conform to the general design constraints). Note that the failure cases do not necessarily have to be mutually exclusive. Table 2 reports the error counts by type found upon close inspection of each of the generated adventures. The complete issue log together with more detailed notes can be found in the project repository.

Table 2. Error Type Counts by Experiment Type and Stage Group.

Stage Type	Correctness		Domain		Style	
	Planned	Direct	Planned	Direct	Planned	Direct
Narrative	0	0	31*	16	12	20
Question	2	4	1	1	2	4
Clue	0	0	0	0	0	0

The first remark to be made here is that while the table includes rows for errors related to clue-type stages, all of their values are 0, as no errors were encountered for any of the categories described in the collection of adventures generated for both modalities. In order to understand the error counts for each of the categories in the table, it is nonetheless necessary to carry out a more exhaustive analysis of the nature of the errors recorded.

In the planned generation scenario, one of the most conspicuous features is the high number of domain errors observed, almost twice as many than with direct generation. On closer inspection, 24 of the 31 reported errors correspond to the problem "The next artefact is revealed in the narrative ahead of time", meaning that the narrative explicitly mentions the name of the artefact that

the player must find in the next clue-type phase. This is true at some point in a large majority of the adventures produced with this method. The remaining mistakes correspond mostly to generation problems arising from the tendency of the model to repeat phrases or whole blocks of context, effectively invalidating an affected narrative block, or problems related to stopping rules of the regular expressions used to control generation colliding with the intention of the model. As an example, we have the case of sentences terminated prematurely by writing "Mr."; since the regular expression decides that a valid sentence ends in a full stop, this halts generation, leaving the phrase half-finished. Problems with questions are mostly linked to the introduction of distractors that are correct or whose phrasing makes them too close to being correct. Correctness errors for questions are tied to suggestions of invalid correct answers.

Table 3. Aggregated execution time and word count statistics both approaches.

Category	Planned				Direct			
	Time (s)		Words		Time (s)		Words	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Narrative	149.42	55.40	339.87	246.61	140.49	62.46	331.38	294.24
Question	84.84	8.74	99.85	26.50	80.88	7.57	91.13	24.18
Clue	54.68	2.89	24.43	4.80	53.58	2.55	24.45	4.68
Character Generation	44.08	1.48	16.85	3.37	42.40	1.53	16.90	3.63
Fact Extraction	165.03	4.89	–	–	–	–	–	–
Plan Generation	443.45	18.30	–	–	–	–	–	–

In the case of direct generation, a higher number of stylistic errors are observed. These often stem from two issues: the model either fails to follow the designer's input or, more commonly, gets caught in a loop, repeatedly generating the same sentences. This problem can spread across multiple stages of the narrative—something that occurs far less frequently in the planned approach. Besides this issue, domain errors in narratives of this generation type are typically more disruptive than those of the previous method, as they stem largely from inconsistencies in dialogue or story. For instance, in the use case "Interactive Family Tour", the characters, both children, hold conversations while referring to each other as "mum" and introduce this into their interactions in a significantly confusing way, producing a difficult-to-understand narrative. On the other hand, several situations were observed in which the model references artworks unrelated to the user's input as if they were the next artefacts to be found by the player. Something similar happens, for example, in the use case "Time Travel Adventure", where the model implies repeatedly that the adventure was focused on the works of authors outside the visit, such as Leonardo Da Vinci.

Regarding questions, on top of the problems present in the planned model, errors can arise that are linked to an excess of information both in the question

(e.g., revealing data on the correct answer) and in the correct answers, as these may become overly verbose and clearly different from the distractors in expressive tone. This problem of overly long questions also translates in specific cases into sentences truncated for exceeding the token limit during generation.

The average time to generate each phase type for both approaches, as well as the average word count in phase definitions, are listed in Table 3. The only notable numerical difference is that the planned version takes about 10 min longer on average than the direct one, corresponding to the execution time of the fact extraction and pre-planning part of the approach. However, since neither method needs to operate in real time, this might not prove relevant in practice.

6 Discussion

In Sect. 5, we focused on providing objective data to draw an initial quantitative profile of our approaches. From a practical point of view, however, it is essential to analyse the overall quality of the adventures created using each method. In this sense, we are interested in understanding which adventures best meet the objective of generating a first skeleton for an adventure based on cultural heritage works. This refers to suggestions that a designer or curator would be able to use as a starting point to build upon in the creative process.

With this in mind, there is a clear tendency for planned adventures to be more consistent with respect to a clear narrative scheme. That is, this methodology tends to result in schemes with a more directed and centred narrative focus throughout the different dialogue blocks than its counterpart without planning. Also, domain errors found with planning are generally easier to correct than those introduced in the direct approach, in that they are often reduced to simply revealing the name of the artefact of the next clue; thus, in many cases it is sufficient to omit this information to obtain a valid phase.

On the other hand, it is not uncommon to find minor hallucinations in the narrative discourse of the direct method, where the model occasionally makes comments regarding works that were never entered as input and therefore have no reason to be in the collection. For example, in the use case "Museum Treasure Hunt",[4] the characters discuss that the piece "The Ecstasy of St. Teresa" caught their attention, without this being part of the actual collection, and then talk about Caravaggio's "The Taking of Christ" holding the key to the next clue, again without this being substantiated by any input and therefore being incorrect from a design point of view. While the planned model is not without these flaws (see the same use case where "The Wynter Gallery's Dutch art gallery" is mentioned), these references tend to be more anecdotal than in its counterpart.

As previously discussed, clues are always valid during generation and can often be considered compelling suggestions, similar to those a designer might propose. However, it is important to note that the clues are generated without the model being aware of all the possible works in the environment where the target

[4] Full examples may be found in https://github.com/pgutierrez858/enigma-llms/tree/master/experiments/results.

artefact is located, and therefore the generation cannot account for possible distractors the riddle could lead to. Now, writing engaging clues for the target audience with a single solution is a challenge and, based on our experience, is something whose validation often depends on user playtesting.

The questions generated by both approaches are generally valid, adhering to the source text, the designer's intention, and the specified difficulty, making them useful first suggestions in the adventure construction process. However, it is important to note that both methods sometimes generate questions that, while correct, are not directly based on the text but likely on the information used to train the original model. While this is not necessarily a flaw, it can lead to hallucinations, for instance, in situations when multiple artefacts seen in training texts share the same name and the model references the wrong one.

7 Conclusions and Future Work

In this paper we introduce a first approach to the automatic generation of treasure hunt games for cultural heritage sites supported by large language models. Starting from a user input specifying the artefacts to be included in the adventure and various textual design parameters, our approach produces a JSON-structured draft of a preliminary quest that strives to comply with the entered constraints. Following the discussion in Sects. 5 and 6, we find that our approach provides a tentative validation of the hypotheses raised in Sect. 4, proposing that the application of a pre-planning strategy has a positive effect on the correctness, consistency, and style of the generated drafts.

There are, however, limitations and points for future work. The first is the potential bias incurred in the selection of the design use cases, which were gathered manually by the authors. The use of a broader dataset derived from interviews with professional content curators in museums is left for future work. Secondly, while here we focus on a pilot for the generation of consistent initial versions of such games, comprehensive usability evaluations, both of how useful the LLM responses are to the designer and a detailed analysis of problems that may arise in the flow of human interaction with the system, remain to be carried out.

Thirdly, our approach as it stands does not allow for interactivity during creation. Alongside the approaches in this work and in the literature, there is a developing interest in the notion of co-design, or turning the human designer into an active actor in the generation algorithm, requesting their insight and participation at different points throughout the applied method [9,26]. In this sense, one of the clearest lines of future work consists of incorporating this paradigm into the workflow for creating treasure hunts, allowing a scenario in which the designer can make use of these generative tools at any point in the creative process and not only in the bootstrapping phase.

Finally, while more advanced models such as GPT-3.5 or GPT-4 exist that have been found to work well for a multitude of tasks [6], their proprietary nature raises concerns. Llama 3.1 Instruct 8B was used due to it being an open

source model that can be deployed on modest hardware and without the need for a dedicated GPU, while remaining performant for general applications [11]. This is relevant, as not all cultural institutions have the resources to run more demanding models. While this explains the a priori high execution times shown in the results section, it remains for future work to explore more efficient generation strategies, especially if interactivity is to be introduced in the workflow.

References

1. guidance-ai/guidance (2025). https://github.com/guidance-ai/guidance. Original-date: 2022-11-10T18:21:45Z
2. Akoury, N., Yang, Q., Iyyer, M.: A framework for exploring player perceptions of LLM-generated dialogue in commercial video games. In: Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 2295–2311. Association for Computational Linguistics, Singapore (2023). https://doi.org/10.18653/v1/2023.findings-emnlp.151, https://aclanthology.org/2023.findings-emnlp.151
3. Ardito, C., Costabile, M.F., De Angeli, A., Lanzilotti, R.: Enriching archaeological parks with contextual sounds and mobile technology. ACM Trans. Comput.-Hum. Interact. **19**(4), 1–30 (12 2012). https://doi.org/10.1145/2395131.2395136
4. Buongiorno, S., Klinkert, L., Zhuang, Z., Chawla, T., Clark, C.: PANGeA: procedural artificial narrative using generative AI for turn-based, role- playing video games. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 20, no. 1, pp. 156–166 (2024). https://doi.org/10.1609/aiide.v20i1.31876, https://ojs.aaai.org/index.php/AIIDE/article/view/31876
5. Cesário, V., Radeta, M., Matos, S., Nisi, V.: The ocean game. In: Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play, pp. 99–109. ACM (2017). https://doi.org/10.1145/3130859.3131435
6. Chen, X., et al.: How robust is GPT-3.5 to predecessors? A comprehensive study on language understanding tasks (2023). https://arxiv.org/abs/2303.00293
7. DaCosta, B., Kinsell, C.: Serious games in cultural heritage: a review of practices and considerations in the design of location-based games. Educ. Sci. **13**(1), 47 (2022). https://doi.org/10.3390/educsci13010047, https://www.mdpi.com/2227-7102/13/1/47
8. Di Nezza, M., De Santis, A., D'Addezio, G.: Cityquest and 'Caccia al...Tesoro dei Castelli'. La nuova frontiera della divulgazione formato 2.0. Rendiconti Online della Società Geologica Italiana **45**, 17–22 (2018). https://doi.org/10.3301/rol.2018.23
9. Gallotta, R., Liapis, A., Yannakakis, G.: Consistent game content creation via function calling for large language models. In: 2024 IEEE Conference on Games (CoG), pp. 1–4. IEEE, Milan (2024). https://doi.org/10.1109/CoG60054.2024.10645599, https://ieeexplore.ieee.org/document/10645599/
10. Gonzalez-Calero, P., Camps-Ortueta, I., Gutiérrez-Sánchez, P., Gómez-Martín, P.: On the importance of contextualizing an educational escape room activity. Electron. J. e-Learn. **22**, 43–56 (2024). https://doi.org/10.34190/ejel.22.4.3199
11. Grattafiori, A., et al.: The LLaMA 3 herd of models (2024). https://arxiv.org/abs/2407.21783
12. Helmy, M., et al.: Navigating the world with an intelligent tourist guide using generative AI. In: 2024 International Telecommunications Conference (ITC-Egypt), pp. 1–6. IEEE, Cairo (2024). https://doi.org/10.1109/ITC-Egypt61547.2024.10620592, https://ieeexplore.ieee.org/document/10620592/

13. Hu, S., Huang, Z., Hu, C., Liu, J.: 3D building generation in minecraft via large language models (2024). https://arxiv.org/abs/2406.08751
14. Camps-Ortueta, I., Rodríguez-Muñoz, J.M., Gómez-Martín, P.P., González-Calero, P.A.: Combining augmented reality with real maps to promote social interaction in treasure hunts. In: Conference of the Spanish Association for Videogames Sciences (2017)
15. Khan, I., Melro, A., Carla, A., Oliveira, L.: Systematic review on gamification and cultural heritage dissemination. J. Digit. Media Interact. **3**(8), 19–41 (2020). https://doi.org/10.34624/JDMI.V3I8.21934, https://proa.ua.pt/index.php/jdmi/article/view/21934
16. Kumaran, V., Rowe, J., Lester, J.: NarrativeGenie: generating narrative beats and dynamic storytelling with large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 20, no. 1, pp. 76–86 (2024). https://doi.org/10.1609/aiide.v20i1.31868, https://ojs.aaai.org/index.php/AIIDE/article/view/31868
17. Malegiannaki, I., Daradoumis, T.: Analyzing the educational design, use and effect of spatial games for cultural heritage: a literature review. Comput. Educ. 108, 1–10 (2017). https://doi.org/10.1016/j.compedu.2017.01.007, https://linkinghub.elsevier.com/retrieve/pii/S0360131517300076
18. Mortara, M., Catalano, C.E., Bellotti, F., Fiucci, G., Houry-Panchetti, M., Petridis, P.: Learning cultural heritage by serious games. J. Cult. Heritage **15**(3), 318–325 (2014). https://doi.org/10.1016/j.culher.2013.04.004, https://linkinghub.elsevier.com/retrieve/pii/S1296207413001349
19. Paliokas, I., Sylaiou, S.: The use of serious games in museum visits and exhibitions: a systematic mapping study. In: 2016 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), pp. 1–8. IEEE, Barcelona (2016). https://doi.org/10.1109/VS-GAMES.2016.7590371, http://ieeexplore.ieee.org/document/7590371/
20. Short, A.R.: Designing fictional worlds for play through large language models. In: Volume 3B: 50th Design Automation Conference (DAC), p. V03BT03A051. American Society of Mechanical Engineers, Washington, DC (2024). https://doi.org/10.1115/DETC2024-143923, https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings/IDETC-CIE2024/88377/V03BT03A051/1208864
21. Sudhakaran, S., González-Duque, M., Glanois, C., Freiberger, M., Najarro, E., Risi, S.: MarioGPT: open-ended Text2Level generation through large language models (2023). https://doi.org/10.48550/arXiv.2302.05981, arXiv:2302.05981
22. Taveekitworachai, P., Nimpattanavong, C., Gursesli, M.C., Lanata, A., Guazzini, A., Thawonmas, R.: Multiverse of greatness: generating story branches with LLMs (2024). https://doi.org/10.48550/arXiv.2411.14672, arXiv:2411.14672
23. Todd, G., Earle, S., Nasir, M.U., Green, M.C., Togelius, J.: Level generation through large language models. In: Proceedings of the 18th International Conference on the Foundations of Digital Games, pp. 1–8. ACM, Lisbon (2023). https://doi.org/10.1145/3582437.3587211, https://dl.acm.org/doi/10.1145/3582437.3587211
24. Trichopoulos, G.: Large language models for cultural heritage. In: Proceedings of the 2nd International Conference of the ACM Greek SIGCHI Chapter, pp. 1–5. ACM, Athens (2023). https://doi.org/10.1145/3609987.3610018, https://dl.acm.org/doi/10.1145/3609987.3610018
25. Wei, J., et al.: Chain-of-thought prompting elicits reasoning in large language models (2023). https://doi.org/10.48550/arXiv.2201.11903, arXiv:2201.11903

26. Yannakakis, G.N., Liapis, A., Alexopoulos, C.: Mixed-initiative co-creativity (2014). https://www.semanticscholar.org/paper/Mixed-initiative-co-creativity-Yannakakis-Liapis/0a410eecf3b23042f95ebfbe0ffdc08ea697c9d9
27. Yao, S., et al.: ReAct: synergizing reasoning and acting in language models. In: International Conference on Learning Representations (ICLR) (2023). https://par.nsf.gov/biblio/10451467-react-synergizing-reasoning-acting-language-models
28. Yu, J., et al.: Personalized treasure hunt game for proactive museum appreciation by analyzing guide app operation log. In: Goh, D.H., Chen, SJ., Tuarob, S. (eds.) ICADL 2023. LNCS, vol. 14458, pp. 30–45. Springer, Singapore (2023). https://doi.org/10.1007/978-981-99-8088-8_3
29. Yu, S., et al.: Few-shot generative conversational query rewriting. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1933–1936. ACM, Virtual Event China (2020). https://doi.org/10.1145/3397271.3401323, https://dl.acm.org/doi/10.1145/3397271.3401323
30. Yu-Han, C., Chun-Ching, C.: Investigating the impact of generative artificial intelligence on brainstorming: a preliminary study. In: 2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), pp. 193–194 (2023). https://doi.org/10.1109/ICCE-Taiwan58799.2023.10226617

The evening's the best part of the day. You've done your day's work. Now you can put your feet up and enjoy it.

Kazuo Ishiguro, The Remains of the Day

Whether you're sad, you're a mess, or you've hit rock bottom, you still have TO PLAY! That's how people like us survive.

Kaori Miyazono, Your Lie in April

Hi, Mom and Dad, it's me, Christine. It's the name you gave me. It's a good one. Dad, this is more for Mom. Hey, Mom, did you feel emotional the first time that you drove in Sacramento? I did and I wanted to tell you, but we weren't really talking when it happened. All those bends I've known my whole life, and stores, and the whole thing. But I wanted to tell you I love you. Thank you, I'm... thank you..

Greta Gerwig, Lady Bird