SISTEMA DE GESTIÓN DE NARRATIVA EN UN ENTORNO VIRTUAL CONTROLADO

DRAMA MANAGEMENT SYSTEM IN A CONTROLLED VIRTUAL ENVIRONMENT

AUTORES:

Néstor Cabrero Martín, Marcos García García, Pablo Martín García, Aaron Reboredo Vázquez

DIRIGIDO POR:

Prof. Dr. D. Carlos León Aznar

Prof. Dr. D. Alan Tapscott Baltar

Trabajo de Fin de Grado del Grado en Desarrollo de Videojuegos

Facultad de Informática, Universidad Complutense de Madrid

Tabla de Contenido

Resumen

La narrativa es algo que está presente en todas las culturas y épocas. En un contexto en el que los videojuegos se consideran un medio global, existe una creciente necesidad de equilibrar la autoría del diseñador y la libertad del jugador. Los sistemas de gestión narrativa han existido desde hace años, pero pocas veces han podido demostrar sus capacidades en entornos virtuales complejos. En este proyecto se ha diseñado e implementado un *drama manager* capaz de orientar la trayectoria narrativa de un videojuego, mientras se maximiza la sensación de libertad del jugador. Además, se han implementado dos videojuegos, *Emerald Hunt* y *Tell No Tales*, como entornos de pruebas para el sistema. Los experimentos se han llevado a cabo sobre *Tell No Tales,* evaluando a 40 participantes humanos. Si bien los resultados de esta evaluación no son conclusivos, y ponen de manifiesto la gran labor de diseño que supone utilizar adecuadamente estos gestores, existen indicios de que un trabajo futuro más exhaustivo, en coalición con otros sistemas, puede ser un camino para el desarrollo de videojuegos con una narrativa procedural viable.

Palabras clave: *videojuegos, generación de narrativa, narrativa procedural, narrativa interactiva, videojuegos narrativos, drama manager*

Abstract

Narrative is something that is present in all cultures and eras. In a context where videogames are considered a global medium, there is a growing need to balance the designer's authorship and the player's freedom. Drama management systems have been around for years, but have rarely been able to demonstrate their capabilities in complex virtual environments. In this project we have designed and implemented a drama manager capable of guiding the narrative trajectory of a videogame, while maximizing the player's agency. In addition, two video games, *Emerald Hunt* and *Tell No Tales*, have been developed as test environments for the system. Experiments have been conducted on *Tell No Tales*, evaluating 40 human participants. While the results of this evaluation are not conclusive, and highlight the great design work involved in properly utilizing these drama managers, there are indications that more comprehensive future work, in coalition with other systems, may be a path to the development of videogames with viable procedural narrative.

Keywords: *videogames, narrative generation, procedural narrative, interactive narrative, narrative videogames, drama manager*

CAPÍTULO I: INTRODUCCIÓN

La narrativa ha formado parte de los videojuegos desde los primeros días de la industria, aunque esta ha visto una evolución significativa. En títulos clásicos como *Super Mario Bros.* (Nintendo, 1985), la narrativa se presentaba como un complemento del videojuego en sí, contada muchas veces en medios externos como el manual de instrucciones (Iorizzo, 2016).

Con el paso de los años, la narrativa ha ido tomando un papel más importante, llegando a ser el núcleo de muchos videojuegos. La narrativa no solo provee de contexto e historia, sino que en muchas ocasiones logra crear en los jugadores una motivación por seguir avanzando en el videojuego, gracias a la emoción y la curiosidad (Iorizzo, 2016).

Debido a la naturaleza interactiva del videojuego, surge la narrativa interactiva, en la que los jugadores pueden crear o influenciar una historia a través de acciones. El objetivo de la narrativa interactiva es sumergir a los jugadores en un mundo virtual y hacerles sentir que son una parte integral en cómo se desenvuelve la historia y que sus acciones tienen consecuencias (Riedl & Bulitko, 2012).

Un ejemplo de narrativa interactiva es una narrativa ramificada en forma de árbol, en la cual las decisiones de los jugadores llevan a una serie de resultados predefinidos, en contraposición a la forma linear que toman las narrativas no interactivas en las que los eventos ocurren siempre en el mismo orden (Niesz & Holland, 1984; Dunfield, 2017).

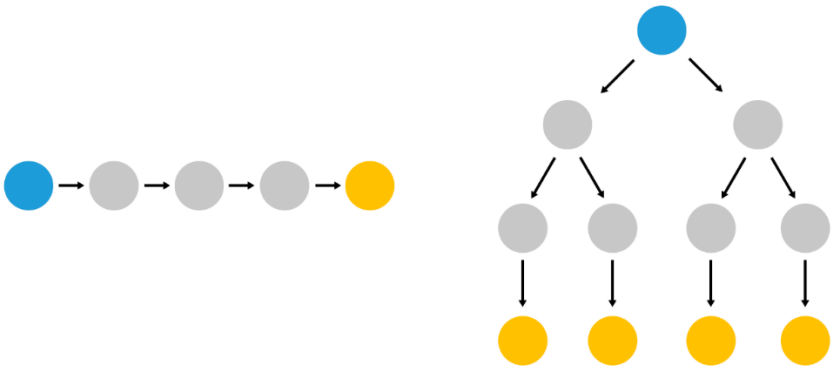

Figura 1: A la izquierda, la forma que toma la narrativa no interactiva lineal. A la derecha, la forma que toma la narrativa interactiva ramificada. En azul, inicio de la historia. En amarillo, final.

Sin embargo, el foco de esta investigación se centra en otro tipo de narrativa interactiva, la narrativa generada de manera procedural. La generación procedural de contenido tiene muchas ventajas a la hora de ser aplicada a los videojuegos, ya que permite customizar el juego para cada jugador, lo que ha demostrado incrementar el disfrute por parte de los usuarios (Yannakakis & Togelius, 2011).

La generación procedural de narrativa comparte estas mismas ventajas, gracias a añadir agencia al jugador (el grado de influencia que el personaje controlado por el jugador puede tener en el estado del mundo del juego) y customización a un videojuego, lo que concluye en mayor inmersión por parte del jugador (Dunfield, 2017).

Para alcanzar mayores niveles de agencia y customización, el rol de determinar cuál debe ser la experiencia del jugador (incluyendo cómo el mundo responde a las acciones de este) puede verse delegado en una Inteligencia Artificial encargada de ello.

Un Sistema de Gestión de la Narrativa (DM, del inglés *Drama Manager*) es un sistema inteligente que pretende controlar el estado del mundo para que una narrativa estructurada se desenvuelva con el tiempo sin reducir la agencia percibida por el jugador. Este sistema usa el principio de la narrativa de observar los posibles futuros de la experiencia del jugador y determinar qué debería ocurrir en el mundo para conseguir una experiencia estructurada y entretenida (Riedl, Thue, & Bulitko, 2011).

La inteligencia artificial en videojuegos hace referencia a las técnicas y algoritmos utilizados para incrementar y mejorar la experiencia jugable (Riedl, 2012).

En videojuegos es típica la concepción de que el objetivo de una inteligencia artificial es el de generar la ilusión de inteligencia (humana o animal) en el comportamiento de personajes no jugables (NPCs del inglés *Non-Playable Characters*), como enemigos o aliados: personajes con los que, en definitiva, se interactúa.

La definición de Inteligencia Artificial parece tener diferentes acepciones en función del campo de estudio y de aplicación, que, si bien se asientan en las mismas bases, tienen matices diferentes (Marr, 2018).

Sin embargo, en esta investigación, cuando se haga una referencia a Inteligencia Artificial se hará pensando en el concepto más generalista y que el padre de la IA, Alan Turing, dio en su momento: la IA es un sistema o programa informático capaz de comprender y gestionar una serie de acciones sobre el sistema y de responder de manera inteligente acorde con ello.

Realmente, el hecho de que exista un personaje que reacciona de manera inteligente a una acción implica que por debajo hay un sistema con las características antes descritas, pero es importante no quedarse simplemente con la idea aislada de IA como personaje que actúa de manera inteligente.

En Inteligencia Artificial, los problemas en el campo de la narrativa interactiva relacionados con la gestión de la acción dramática son normalmente modelados como espacios de estados. Con un DM, estos estados representan una configuración específica del entorno del videojuego.

El sistema se encarga de elegir una secuencia de estados por los que el jugador debería pasar para lograr la experiencia deseada (Hogg, Lee-Urban, Muñoz-Avila, Auslander, & Smith, 2011). Por supuesto, existen más acercamientos (Riedl, 2012) al problema de la generación de narrativa, pero el objeto de estudio será un gestor de la narrativa.

1.1 Motivación

La narrativa es algo que está presente en todas las culturas y épocas. Un fenómeno tan complejo como la narración resulta difícil de modelar utilizando las técnicas convencionales del campo de la Inteligencia Artificial, e igualmente complejo es evaluar los resultados de los experimentos.

Acercamientos previos a los sistemas de gestión de la narrativa en videojuegos han ofrecido resultados limitados, dado que el espacio de posibilidades que ofrecen en sus implementaciones es bastante contenido; no se puede obviar que no existe una solución completa al problema de la narrativa interactiva. Además, muy pocas de estas soluciones se han implementado dentro de las convenciones de un videojuego narrativo moderno.

Vivimos en un contexto en el que los videojuegos son un fenómeno global y la sensación de control y libertad es un punto clave en su disfrute (Muriel & Crawford, 2018). Esta necesidad de libertad a veces contrasta con el deseo del diseñador de contar una historia. La motivación de este proyecto es puramente técnica: se trata de querer equilibrar la libertad de juego con la narrativa, apoyándose no sólo en el conocimiento preexistente sobre Inteligencia Artificial, sino también en técnicas de evaluación de la experiencia de usuario más cercanas a la Psicología.

1.2 Hipótesis

Con la motivación de la sección anterior, se plantea como hipótesis inicial que se puede desarrollar un sistema de gestión de la acción dramática que se adapte a diversos contextos narrativos ofrecidos en un entorno virtual como el de los videojuegos, garantizando el cumplimiento de ciertos puntos clave de la trama especificados por el diseñador, y sin que el jugador perciba durante su estancia en dicho entorno una reducción significativa de su sensación de agencia.

Concretamente, se plantea que el uso de gestores de narrativa en videojuegos puede garantizar que el jugador perciba más elementos narrativos sin detrimento de su sensación de control y libertad en el entorno de juego.

1.3 Objetivos

Este proyecto busca demostrar que existe una mejora de la calidad de la experiencia narrativa al utilizar sistemas de gestión de la narrativa en videojuegos.

Se debe recalcar que no existe una solución universal al problema que se ha propuesto resolver; los objetivos que se plantean son un acercamiento a una posible solución.

La solución se aleja de ser completa, puesto que abarcar todos los módulos que componen los acercamientos clásicos supondría tantear terrenos que van desde lo tangencial, como el Procesamiento del Lenguaje Natural, hasta lo remoto, como la Psicología, necesaria en la generación de perfiles de jugador.

De esta manera, se definen los siguientes objetivos:

- Diseñar e implementar un gestor de la acción dramática, núcleo principal del proyecto, que debe ser capaz de guiar al jugador hasta el final de una historia, utilizando ciertos elementos de narrativa procedural.
- Diseñar una forma de representación lógica de las historias que permita al sistema DM detectar, procesar y reaccionar de manera coherente y correcta a las acciones del jugador.
- Implementar diferentes algoritmos de búsqueda para comparar la eficiencia de varios acercamientos al diseño de los sistemas DM.
- Diseñar un entorno razonablemente sencillo y controlado para evaluar el correcto funcionamiento del DM, integrando ambos módulos.
- Llevar a cabo las pruebas experimentales necesarias para verificar la hipótesis inicial con el prototipo desarrollado.
- Probar la utilidad de elementos narrativos procedurales en este mismo entorno, así como dilucidar cómo afectan a la agencia percibida por el jugador en contraste a la selección y aplicación de guías propuestas por el sistema de gestión de la acción dramática.

1.4 Metodología

En los primeros meses de trabajo se determinará cuál será la herramienta de desarrollo elegida para la implementación del proyecto, tanto de la IA como de los videojuegos necesarios para probarla. Las dos herramientas más atractivas son el motor Unity y el videojuego *Minecraft* (Mojang, 2011). Más adelante, en el **CAPÍTULO II**: TRABAJO PREVIO, se expondrán otras herramientas, mientras que en el **CAPÍTULO IV**: IMPLEMENTACIÓN, se discutirán las ventajas y desventajas de cada una de las ellas, y se expondrá la elección final.

Más tarde, comenzará el desarrollo del Sistema de Gestión de la Acción Dramática, haciendo uso de los conceptos ya aprendidos mediante un proceso de documentación inicial. Para probar el sistema, se desarrollará un primer prototipo, un videojuego sencillo que permita comprobar si todo funciona de la manera esperada y, en caso de que esto no sea así, arreglar posibles errores del gestor.

Una vez se haya determinado el correcto funcionamiento del sistema, el nuevo objetivo será la implementación de un segundo prototipo. Se tratará de un videojuego bastante más completo que el primero, el cual servirá como un ejemplo de uso del Gestor de la Narrativa más convencional, haciendo ver que el uso del sistema es posible en entornos más complejos.

Cuando el prototipo final esté acabado, se realizará un experimento. Este experimento consistirá en hacer que varias personas jueguen a dos versiones distintas, una con el sistema integrado y otra sin él. Más tarde, tendrán que responder una serie de preguntas que se analizarán una vez finalizado el experimento para poder trazar las conclusiones.

1.4.1 Herramientas de Investigación

Además de las herramientas para la implementación, se hará uso de otras para la investigación y la coordinación de esta. Para que tanto los tutores como los investigadores tengan en todo momento acceso al proyecto, se utilizará GitHub. Periódicamente, se actualizará el contenido del repositorio con la última versión

del proyecto. Este repositorio será privado, solo dando acceso a las personas involucradas con la investigación.

Para la escritura de la memoria, al tratarse de un trabajo conjunto, se utilizará Microsoft Office con funciones online. Estas permiten escritura simultánea por parte de diferentes usuarios. También permiten a los tutores del proyecto poder observar los nuevos cambios en la memoria en todo momento, así como añadir comentarios en los posibles errores.

Por último, para la realización del experimento final, se hará uso de Google Forms. Esta herramienta permite crear y analizar los resultados de cuestionarios de forma sencilla. Se explicará con mayor detalle en el **CAPÍTULO V**: EXPERIMENT.

1.4.2 Planificación

Tras exponer la metodología, en este apartado se adjunta una Tabla 1: Planificación mes a mes del desarrollo del proyectoTabla 1que contiene la planificación del proyecto mes a mes. La investigación comenzó en julio de 2020 y terminó en febrero de 2021.

Periodo	Tareas Planificadas
Julio 2020	Proceso de documentación: búsqueda de bibliografía sobre narrativa interactiva y procedural. Comienzo de escritura de la memoria: planteamiento de los capítulos de introducción y trabajo previo.
Agosto 2020	Proceso de documentación: búsqueda de bibliografía sobre gestores de narrativa y videojuegos narrativos. Decisión final de las herramientas de desarrollo que se utilizarán en el proyecto.
Septiembre 2020	Implementación preliminar del gestor de narrativa. Integración por texto. Escritura de la memoria: primer acercamiento a la discusión técnica e implementación.
Octubre 2020	Implementación completa del primer prototipo del gestor de narrativa. Implementación del primer videojuego para probar el prototipo.
Noviembre 2020	Comienzo del desarrollo del gestor de narrativa mejorado. Diseño del segundo videojuego y búsqueda de *assets* para su desarrollo. Escritura de la memoria: ampliación.
Diciembre 2020	Implementación completa del prototipo final del gestor de narrativa. Integración completa de los módulos de IA y juego.
Enero 2021	Diseño y realización del experimento. Análisis de los resultados. Escritura de la memoria: experimento y conclusiones.
Febrero 2021	Finalización de la memoria.

Tabla 1: Planificación mes a mes del desarrollo del proyecto

1.5 Estructura del Documento

Tras un primer capítulo de introducción a la temática del proyecto y prospección, este documento se orienta en su segundo capítulo a realizar un repaso exhaustivo de todos los antecedentes del proyecto que se han considerado de interés para contextualizar el desarrollo.

Así pues, se repasará brevemente la historia de la narrativa interactiva, haciendo hincapié en aquellos hitos que han sido significativos para tomar ciertas decisiones de diseño. Después, se repasarán algunos de los sistemas de gestión narrativa que se han desarrollado en los últimos años, así como algunos de los videojuegos que han intentado traer avances y novedades al terreno de la narrativa. Se terminará el capítulo explicando brevemente las herramientas que serán usadas para desarrollar los prototipos de esta investigación.

En el tercer capítulo se llevará a cabo una discusión técnica sobre el diseño del prototipo de gestor de la narrativa planteado por este proyecto, que se complementará en el capítulo cuarto con una explicación pormenorizada de su implementación. En este capítulo también se dará cobertura a la implementación de los juegos que serán utilizados para desarrollar los experimentos pertinentes.

El quinto capítulo de este documento estará dedicado a cubrir los resultados de los experimentos con el sistema integrado en un videojuego. Tras un sexto capítulo dedicado a la discusión de estos resultados, en el capítulo séptimo se expondrán las conclusiones y posible trabajo futuro.

El documento se cierra con la sección de Bibliografía, después de la cual podrán consultarse tanto la introducción como las conclusiones en inglés. En los Anexos se aportará documentación relacionada, especialmente, con el diseño del videojuego. Por último, se glosarán las aportaciones de cada miembro al proyecto.

CAPÍTULO II: TRABAJO PREVIO

En esta sección se exploran acercamientos previos a soluciones técnicas al problema de mejorar la experiencia narrativa en la ficción interactiva. Estas soluciones técnicas tiendan a configurarse en dos grupos: sistemas centrados en la narrativa emergente y sistemas de gestión de la narrativa. Ha habido distintos acercamientos al problema de mejorar la complejidad narrativa sin afectar a la sensación de agencia, pero la mayoría pasan por la implementación de sistemas que hacen uso de gestores de narrativa. Este proyecto se centra en ellos, ya que son los que permiten, posiblemente, un acercamiento más centrado en el equilibrio entre agencia del jugador e intención del autor. También se hará un breve repaso a la historia de la narrativa interactiva, elaborando algunos ejemplos centrados especialmente en los videojuegos.

2.1 Breve Historia de la Narrativa Interactiva

La narrativa puede definirse como una serie estructurada de eventos (Roberts, Riedl, & Isbell, 2011). La narrativa interactiva involucra al jugador en la historia, permitiéndole cambiarla de manera significativa. La interactividad, sin embargo, trae nuevos problemas a la narración, que requieren soluciones técnicas y de diseño.

Los primeros videojuegos contaban con unos recursos muy limitados para mostrar representaciones del mundo real. Juegos como *Tennis for Two* (Higinbotham, 1958) y *Pong* (Atari, 1972) carecían de memoria operativa, y por tanto estaban muy limitados en lo que a la representación visual de elementos del juego se refiere.

Desde la invención del microprocesador en 1971, los videojuegos y su apartado técnico han ido creciendo en complejidad de forma continua, pudiendo ser cada vez más detallados y expresivos. Por ejemplo, la vibrante y colorida experiencia que supuso *Centipede* (Atari, 1980) sólo fue posible gracias a estos avances técnicos. Con su llegada, se puso sobre la mesa la posibilidad de incluir elementos narrativos.

Adventure (Atari, 1980) es probablemente el primer videojuego del género de aventuras, influido por las historias de ficción interactiva basadas en texto que le precedieron, y por el fenómeno que sacudió a mediados de los años 70 el panorama de los juegos de rol de mesa, *Dungeons and Dragons* (Gygax, 1974). *Donkey Kong* (Nintendo, 1981) es probablemente el primer videojuego con una historia que utiliza sus elementos visuales para transmitir una narrativa: se cuenta cómo Jumpman trata de rescatar a Pauline de las garras de un simio descomunal.

Desde estos primeros pasos de la industria quedó establecido un vínculo tácito entre el apartado técnico de los videojuegos y la complejidad mecánica que podían y debían exhibir. En la actualidad esta necesidad es aún más patente si cabe; un videojuego con un mundo muy detallado, pero con unos agentes cuyo comportamiento no está al mismo nivel de realismo, puede romper por completo la experiencia y el disfrute del jugador. Paralelamente, la complejidad de la narrativa en los videojuegos también se ha ido incrementando para estar a la altura de estos otros elementos.

La Inteligencia Artificial (IA) en los videojuegos ha ido incrementando su complejidad progresivamente con el objetivo de alcanzar el mismo avance que se producía en el apartado técnico de estos. Se trataba de proveer a los jugadores de una experiencia inmersiva donde los personajes no jugables (NPCs) se comporten de forma creíble. Algunas aplicaciones desarrolladas en este sentido incluyen inteligencias artificiales que puedan oponerse y adaptarse al comportamiento del jugador, incrementando el desafío del juego. En el contexto de los videojuegos narrativos, esta adaptabilidad también es deseable, pero con el objetivo de construir historias en las que el jugador tenga una mayor percepción de intervención y de control.

El término que en inglés se conoce como *agency*, y que se traducirá como agencia, indica el grado de influencia que el personaje controlado por el jugador puede tener en el estado del mundo del juego. Así, esta agencia en la narrativa es clave para muchos jugadores, ya que afecta significativamente a la historia y les hace sentir que sus decisiones cuentan (Hernandez, Bulitko, & Hilaire, 2014). La agencia es importante para la narrativa interactiva, dado que le da credibilidad a cualquier situación. Se podría decir que existe una correlación entre la agencia del jugador y la inmersión, un elemento que, sin duda, contribuye al disfrute de los videojuegos.

La narrativa brinda un aspecto fundamental a los videojuegos modernos, aportando un contexto al detallado apartado visual y a las acciones que debe llevar a cabo el jugador, con la agencia jugando un importante papel en el disfrute de dicha experiencia. Existe una creciente comunidad de investigadores centrada en el desarrollo de aplicaciones que implementan sistemas de Inteligencia Artificial orientados a la narración interactiva. Esta área de investigación es muy amplia y cubre muchos campos y aplicaciones, tales como la generación de narrativa, la interpretación del lenguaje natural o la gestión de la narrativa (Sengers, 2000). La narración es una tarea compleja que este campo de investigación trata de modelar.

Los sistemas de gestión narrativa son, como ya se ha dicho, uno de los campos de investigación de esta área. En vez de generar historias de forma procedural, estos sistemas seleccionan el contenido de la historia en base a las acciones del jugador utilizando un Gestor de la Narrativa (DM, del inglés *Drama Manager*) para guiar la historia a través de un arco narrativo cambiante (Riedl, 2012). Suelen utilizarse para seleccionar y ordenar el contenido de una historia previamente creada por un autor humano. Los sistemas DM varían en su metodología, pero comparten muchos de los retos de diseño, tales como el equilibrio entre la intención del diseñador y la agencia del jugador (Riedl, 2012), o el llamado problema del límite, que se produce cuando las acciones del jugador no están previstas por el contenido escrito por el autor humano (Sharma, Ontañón, Mehta, & Ram, 2010); el sistema, al no estar preparado para dichas acciones, puede producir respuestas erróneas o ilógicas.

Para resolver algunos de los problemas que presentan las narrativas en los complejos espacios de contenido escrito por autores humanos, un sistema ideal debería incorporar aspectos positivos de la narrativa ramificada convencional, así como el diseño de un sistema DM. De este modo, la experiencia narrativa quedará ligada a las acciones de cada jugador, incrementando la efectividad de la ilusión de libertad de elección y haciendo, así, mucho más complicado detectar el engaño. En última instancia, esto favorecerá la sensación de agencia en el jugador, sin los inconvenientes tradicionales.

Un sistema más flexible aplicaría restricciones sin reducir el rango de libertad de acción percibido por el jugador, limitando no tanto lo que el jugador puede hacer, sino lo que es probable que piensen hacer a continuación (Sharma, Ontañón, Mehta, & Ram, 2010).

En los videojuegos orientados a procesos los jugadores disponen de un sistema con el que experimentar, en vez de unos objetivos establecidos por el diseñador (Nielsen, 2008). Estos videojuegos permiten al jugador elaborar sus propios objetivos de forma emergente. *The Stanley Parable* (Davey Wreden & William Pugh, 2013), descrito como un *sandbox* narrativo, utiliza la falta de objetivos propuestos por el diseñador como base para elaborar una auténtica narrativa con ramificaciones en la que se permite que las decisiones del jugador sean expresivas y significativas sin tener por ello que contribuir a un objetivo final mayor.

Los creadores del juego opinan que esto convierte al juego en un reflejo de los propios jugadores (Wreden & Pugh, 2016). Este concepto de objetivos establecidos por el propio jugador puede aplicarse utilizando un DM en un videojuego narrativo, lo que aumentaría la agencia en estos espacios narrativos complejos de los que se hablaba anteriormente.

Sin embargo, la idea de utilizar un DM que facilite una mayor libertad de acción al jugador para alcanzar las metas que se autoimponga y, aun así, dirigirlo hacia un determinado arco dramático preestablecido, es un camino que todavía no se ha explorado en videojuegos comerciales.

Como se ha mencionado anteriormente, el interés por una narrativa más compleja en los videojuegos es algo que ha ido en aumento en los últimos años. Pero, poco a poco, esa necesidad por hacer historias de calidad capaces de competir con la literatura o la industria del cine ha derivado en una ambición por conseguir un tipo de narración que va mucho más allá.

Si los videojuegos cuentan con una ventaja, esta es su naturaleza eminentemente interactiva, ya que brinda unas posibilidades que, aunque exploradas en la literatura ya desde finales de los años 70 (*Elige tu propia aventura*, de R.A. Montgomery), en el mundo del cine (*Black Mirror: Bandersnatch*, de Netflix), o en los juegos de mesa (*Risk*, de Hasbro), nunca plantearon una interactividad tan pura como en este caso.

Aun así, esta interactividad es limitada, ya que todas las acciones que pueda realizar el jugador están pensadas, analizadas y diseñadas, excepto en aquellos casos en los que se busca la jugabilidad emergente. Este término hace referencia a, precisamente, cualquier interacción desarrollada por el jugador que no estuviese explícitamente prevista por los creadores (D'argenio, 2018).

Si bien la narrativa emergente no es el eje central de esta investigación, no se puede desviar la atención de títulos que llevan la interacción a un nuevo nivel. En proyectos en los que se busca dotar al jugador de una gran libertad, tienden a generarse situaciones que no han sido diseñadas de antemano. Títulos como *The Legend of Zelda: Breath Of the Wild* (Nintendo, 2017) son hoy uno de los mejores ejemplos en cuanto a la jugabilidad emergente y de cómo los usuarios son capaces de combinar mecanismos para generar nuevas dinámicas. Sin embargo, esta jugabilidad emergente no tiene un efecto visible en el videojuego, y no afecta a la historia.

En algunos títulos la cumplimentación de ciertas misiones secundarias no sólo lleva a modificar la experiencia o la historia jugada por el simple hecho de que se jueguen esas misiones, sino que en algunas ocasiones la historia evoluciona de manera totalmente diferente y orientada hacia un final alternativo. En muchas ocasiones, esta evolución y cambios en la historia se llevan a cabo al jugar misiones clave que

lanzan una nueva rama de la historia, como puede ocurrir en el caso de *Fallout 4 o Skyrim* (Bethesda Game Studios, 2011-2015). Hay otros títulos en los que estos finales alternativos vienen marcados por acciones más explícitas, como pueden ser escoger opciones en pantalla, como en *Far Cry 3* (Ubisoft, 2012).

Existen también otros títulos que han pulido esta fórmula y que incorporan un sistema de karma, en el que cualquier acción que pueda realizar el jugador en el juego tiene su efecto, no sólo en los posibles finales, sino en la manera que el mundo reacciona a nuestra presencia. Uno de los títulos más recientes que cumple a la perfección con este objetivo es *Red Dead Redemption 2* (Rockstar Games, 2019). En este título cada buena acción supone un incremento de la barra de honor. Unos niveles de honor altos llevarán a los jugadores a los finales más benévolos con el personaje que manejan y a que el mundo sea menos hostil con ellos, mientras que un nivel de honor bajo desencadenará todo lo contrario. El modo en que el jugador se relaciona con este sistema de karma es en sí una narrativa emergente.

Otro germen de inspiración que podría estar relacionado, en cierta medida, con este proyecto, son los títulos que tienen como pilares fundamentales de la experiencia la generación procedural. En este punto se debe tener en cuenta que, cuando se hace referencia a este tipo de juegos, se encuentran títulos que generan normalmente entornos, mapeados y criaturas de manera procedural a medida que se va avanzando en la historia o a medida que se van cargando áreas a las que el jugador accede, como ocurre en *No Man's Sky* (Hello Games, 2016). Si bien es cierto que suponen la creación de experiencias diferentes para cada usuario debido a la aleatoriedad y proceduralidad de sus entornos, no contribuyen como tal al desarrollo de la historia ni la narrativa. El jugador se encuentra con criaturas diferentes y planetas diferentes, pero el mundo no genera elementos que mantengan al jugador en el camino correcto para seguir la historia (Murray, 2017).

Existen títulos cuya aproximación a lo procedural explora otros límites más allá de la generación de elementos puramente estéticos, dando pie a subtramas y situaciones imprevistas para los jugadores: es el caso del sistema Némesis de *El Señor de los Anillos: Sombras de Guerra* (Taljonick, 2014).

En este caso, existe un sistema en el que cada orco tiene sus propios rasgos característicos y reacciona a las decisiones del jugador. Si el jugador decide humillarle durante la batalla, descenderá su rango; si, por el contrario, el orco sale victorioso del encuentro, se empoderará, pudiendo ascender de rango y volver con más fuerza a por el jugador. También es posible dominar a las huestes y generales, poniéndolos bajo el mando del jugador. Estos generales pueden aumentar de rango en el ejército del jugador y ser de gran ayuda en el futuro, o, por el contrario, traicionar al jugador. Todo esto se genera a medida que el jugador avanza en la historia, dando lugar a un sistema y un universo que parecen tener vida propia. Sin embargo, no se está generando narrativa a medida que se avanza, ya que la historia sigue siendo la misma en todo momento. Se trata, por tanto, de un sistema que enriquece la experiencia jugable.

Resulta curioso que el paralelismo más directo entre el videojuego que se desarrolla para este proyecto esté, precisamente, en los orígenes de la narrativa interactiva. El modelo que se trata de explorar es bastante similar al de una larga partida de rol de *Dragones y Mazmorras*, en el que la historia en la que se sumergen los jugadores va cambiando en función de la imaginación de un *Game Master*, que tiene pensado desde el principio los puntos trascendentales de la historia que se quiere contar. Algunos antropólogos aseguran que los orígenes de este gusto por las historias interactivas se remontan a las civilizaciones más antiguas (Short, 2019).

La idea de crear una inteligencia artificial capaz de sustituir las funciones del *Game Master* es, cuanto menos, compleja, especialmente en términos computacionales. La representación del conocimiento es otro punto crítico, que implica que el sistema entienda lo que está ocurriendo en el mundo y sepa generar soluciones factibles a los problemas y estados de la narrativa que se vayan sucediendo, y que reaccione de manera coherente y entendible por los usuarios humanos. Es en esta representación del conocimiento donde se encuentra uno de los mayores retos de la investigación y del proyecto y uno de los motivos por los que gran número de los estudios que giran en torno a este problema solo plantean soluciones a nivel teórico o se alejan de implementaciones complejas.

2.2 Sistemas de Gestión de la Narrativa

La generación procedural de narrativa puede definirse como cualquier proceso automático que crea narrativa a lo largo del tiempo, siempre y cuando esa narrativa no sea definida antes del comienzo de dicho proceso (Togelius, Kastbjerg, Schedl, & Yannakakis, 2011).

El principal reto a la hora de diseñar estos sistemas es mantener el equilibrio entre la espontaneidad y la estructura; o, dicho de otra manera: se puede priorizar la libertad de los jugadores dentro del sistema o se puede priorizar la estructura narrativa propuesta por el autor.

Estos dos acercamientos, uno centrado en la simulación y otro centrado en la trama, son los más comunes dentro del campo de estudio de la narrativa procedural (Martens & Cardona-Rivera, 2017). En ambos acercamientos, la generatividad narrativa se introduce tomando como base una modelización lógica o representación formal de los personajes que intervienen en la historia, así como de las situaciones cambiantes que atraviesan durante los eventos de esta.

En el modelo centrado en la simulación, el foco suele estar en el diseño de un conjunto de personajes, incluyendo sus personalidades, aspiraciones y relaciones; además, se describen las reglas que indican cómo cambiarán esas propiedades a través de sus interacciones. El diseñador debe especificar qué reglas de interacción suministra al sistema, así como un estado inicial que representa la situación global del mundo en el que se va a producir la simulación.

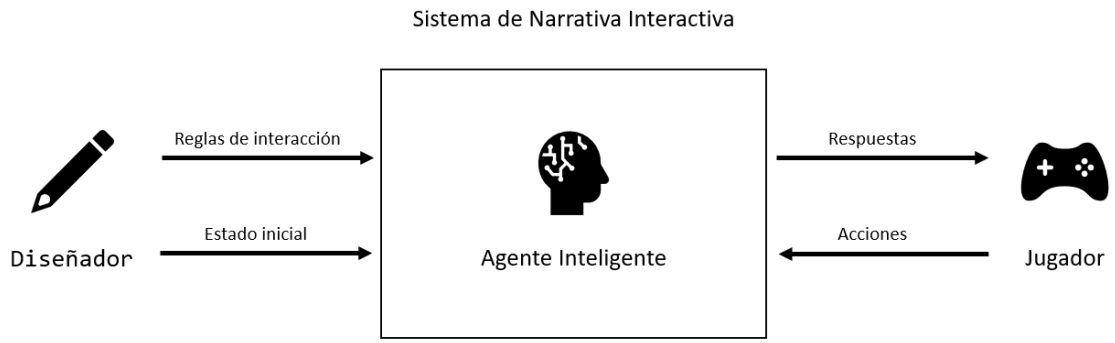

Figura 2: Conceptualización del acercamiento centrado en simulación para el diseño de un sistema de narrativa interactiva

La simulación consiste en explorar el espacio de estados, ya sea de manera completamente aleatoria o permitiendo que sea el propio jugador quien, con sus acciones, determine algunos de los pasos del proceso. Sin embargo, la interacción supone un trabajo añadido para el autor, que debe pensar más en el diseño de la experiencia.

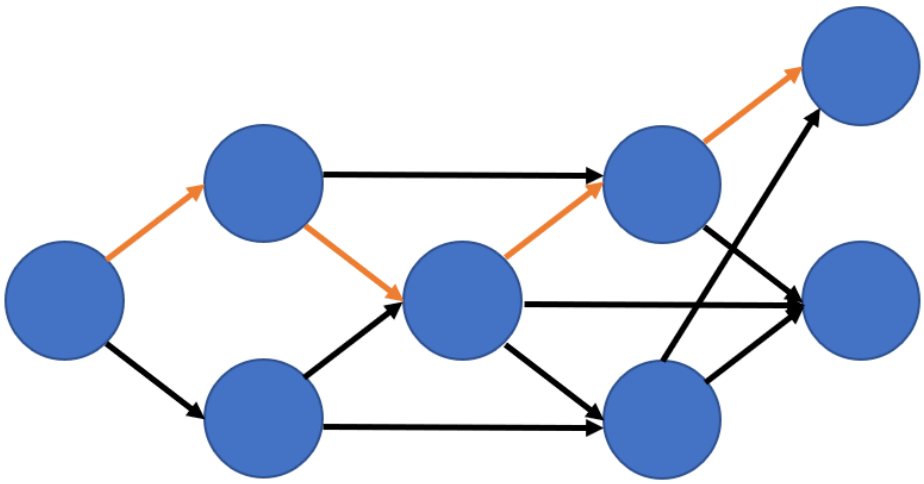

Figura 3: Ejemplo de espacio de estados generado por un estado incial y un conjunto de reglas de interacción. Cada estado se representa con un círculo azul y cada transición con una flecha; las flechas naranjas representan una búsqueda completa o solución.

Cada vez que se explora este espacio de estados, se genera un camino a través del grafo: en este modelo, una historia es una sucesión de estados. Sin embargo, las acciones del jugador pueden afectar negativamente a la simulación, llevando al sistema a estados imprevistos.

Es por esto por lo que la interacción del jugador con el sistema se puede ver limitada. Alguna de las formas de controlar cómo interactúa el jugador con el sistema son permitiéndole manejar a un solo personaje, a todos (*Prom Week*) o, incluso, manipulando el mundo en vez de a los personajes (*Blood and Laurels*).

En cualquier caso, al ser el jugador el que decide qué hacer en cada momento, se hace complicado establecer una progresión narrativa en aquellos sistemas centrados en la simulación. Cuando el diseñador necesita reforzar ciertos puntos del argumento, estableciendo metas claras al sistema, utilizará un acercamiento centrado en la trama.

Los sistemas centrados en la trama se cimentan sobre la arquitectura de los sistemas centrados en la simulación: de nuevo, se necesitará suministrar al sistema un estado inicial y un conjunto de reglas de interacción. Adicionalmente, el sistema necesita una Agenda Narrativa, que representa la trama deseada por el diseñador o autor.

El diseñador busca que se cumpla la Agenda Narrativa en el mundo interactivo, y, para garantizar que esto ocurre, se introduce un segundo agente inteligente en el sistema. Este mediador se encarga de monitorizar las acciones del jugador, e intenta dirigir su trayectoria narrativa hacia los objetivos exigidos por la Agenda Narrativa.

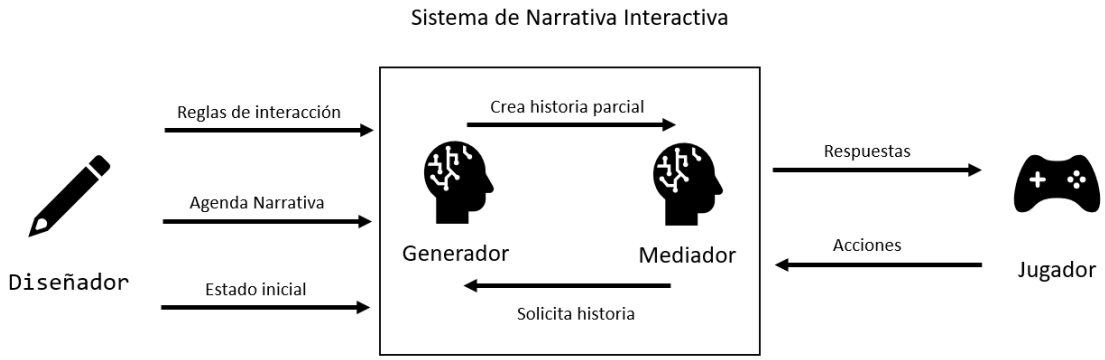

Figura 4: Conceptualización del acercamiento centrado en la trama para el diseño de un sistema de narrativa interactiva

La exploración del espacio de estados, en este caso, se orienta a alcanzar estados deseados, un subconjunto de estados que representan puntos clave de la trama que, en conjunto y secuencialmente, componen la historia que el diseñador quiere transmitir al jugador.

Durante la ejecución, el sistema turna la recepción de interacciones procedentes del jugador con la generación de respuestas, en un intento por facilitar que se alcancen los estados deseados.

Además, el sistema debe contar con herramientas para impedir que el jugador tome decisiones que le alejen de los estados deseados; si estas acciones se permiten, el sistema debe contar con un generador de historia que le permita replanificar la experiencia.

Actualmente, se trata de pasar de un modelo centrado en la trama a uno centrado en el discurso. Tradicionalmente, se ha utilizado el espacio de estados para elaborar un discurso, pero la intención de estos nuevos sistemas es invertir el proceso, y no generar aquellas narrativas que nunca se presentan al jugador (Koenitz, 2015). También es importante destacar que, con un número limitado de reglas de interacción, es complicado generar narrativas complejas; este objetivo requiere de mucho trabajo por parte del autor o diseñador.

A continuación, se desgranarán las características de algunos de los sistemas de narrativa interactiva más conocidos (León, Gervás, Delatorre, & Tapscott, 2020).

2.2.1 MINSTREL

MINSTREL (Turner, 1994) es un sistema que requiere de una librería de historias preexistentes para la creación de nuevas historias mediante su función de generación. En este se encuentran muy presentes las nociones de orden, causalidad y entidades de los personajes. El sistema verifica constantemente la consistencia y añade hechos y precondiciones para mantenerla. En *MINSTREL* las historias siempre comienzan con una escena introductoria que presenta hechos del mundo mediante descripciones breves. Estos hechos son referenciados y/o modificados más tarde en acciones que hagan avanzar dicha historia.

De esta forma, el sistema se asegura de que los eventos ocurren en un orden coherente y consigue generar sensación de causalidad. Además, todos los avances de la historia se verifican respecto a los personajes y sus objetivos, el mundo y la consistencia emocional, junto con otras limitaciones añadidas que se aseguran de mantener la consistencia a la hora de añadir o modificar la historia.

Debido a que el generador de narrativa genera acciones realizadas por los personajes, la estrategia para hacerlas más consistentes consiste en añadir nuevos hechos del mundo a los que las acciones incoherentes puedan referirse. Por ejemplo, si surgiese una acción incoherente del tipo "*Aaron mató al dragón*", se puede introducir un hecho descriptivo del tipo "*Aaron quería impresionar al rey*" para convertir esa acción en coherente y reforzar la sensación de causalidad.

2.2.2 Universe

Universe (Lebowitz, 1984) es un sistema dedicado a la generación de guiones para telenovelas. Hace uso de estructuras de datos complejas que representan datos sobre los personajes y depende de algoritmos e input de usuario. Las estructuras de datos de los personajes se crean antes que la historia, dando gran importancia a la coherencia de los estos y a sus relaciones.

Los datos de cada personaje incluyen características de estos y relaciones interpersonales cuantificadas (Por ejemplo: "*Liz Candler es básicamente una mujer social y rica que está actualmente casada con Tony Dimera, pero le odia*"). Los personajes también constan de objetivos y subobjetivos con precondiciones que están pensados para generar conflictos con otros personajes. El sistema entonces procede eligiendo objetivos que cumplan todas sus precondiciones e implementa desarrollos de la trama para lograr alcanzarlos, cambiando el estado de la historia y posiblemente completando precondiciones de otros objetivos. De esta forma se consiguen desarrollar diversas líneas narrativas que ocurren al mismo tiempo, como en las telenovelas reales.

Como en un principio los personajes solo constan de características y relaciones, pero no han realizado ninguna acción, se está ante una representación descriptiva del mundo. Estos personajes son usados más tarde para implementar segmentos de historia que generen acciones. Estas dos dinámicas describen el proceso creativo de establecer una descripción inicial del mundo (los datos de los personajes) y luego ir añadiendo acciones que dependan de acciones anteriores o de la descripción inicial para generar una historia, lo que produce una sensación de causalidad.

2.2.3 Virtual Storyteller

Virtual Storyteller (Theune, Faas, Nijholt, & Heylen, 2003) posee un agente director que conoce la estructura de la trama. Sus objetivos principales son la creación de historias consistentes y bien estructuradas. Para alcanzar estos objetivos, el director consta de control ambiental (manipula el mundo añadiendo o eliminando elementos), control motivacional (cambia las motivaciones de los personajes) y control prohibitivo (prohíbe algunas acciones).

El director, sin embargo, no puede realizar ninguna acción por sí mismo. Por otro lado, los personajes constan de creencias, deseos y acciones. En este sistema, la distinción entre las descripciones, cuyo objetivo es establecer hechos, y la cadena de acciones, que dependen de dichos hechos, está implícita en el propio proceso creativo.

En *Virtual Storyteller*, el director usa los dispositivos de control para introducir elementos óptimos y motivaciones que desencadenan acciones. Estas no solo representan una historia bien construida, sino que también son coherentes. Existe una separación clara entre el contenido descriptivo manipulado por el director que configura la historia del mundo, y los desarrollos de la trama llevados a cabo por los agentes de los personajes. El proceso resultado contiene descripciones que se introducen antes de las acciones. Las nuevas acciones siguen referenciándose entre sí hasta que una nueva descripción es necesaria para mantener la consistencia desde el conocimiento estructural sobre la trama del director.

2.2.4 Fabulist

De manera similar, *Fabulist* (Riedl & Young, 2010) es un generador de narrativa de varios niveles que depende del algoritmo IPOCL (*Intent-Driven Partial-Order Casual Link*) para producir secuencias que representan narrativas coherentes y personajes creíbles. Con este acercamiento, se percibe un gran interés por la credibilidad y causalidad de los personajes, lo que resulta en una cadena de descripciones y acciones que reflejan esto.

La historia resultante alterna entre establecer hechos de manera descriptiva ("*El rey Antonio no está casado*", "*Jasmine es muy bella*"), imponiendo acciones que mueven la historia ("*Antonio se enamora de Jasmine*") y estableciendo deseos e intenciones de los personajes para mejorar la coherencia y causalidad de la historia ("*Antonio es muy enamoradizo*").

Aunque estos procesos son cortos e iteran muy frecuentemente, existe un claro esfuerzo en separar estas frases y en mantenerlas conectadas de forma causal para formar una frase coherente.

2.2.5 MEXICA

MEXICA (PÉrez & Sharples, 2001) genera historias cortas sobre los habitantes nativos de México mediante dos estados cíclicos: *engagement* y *reflection*. En el estado *engagement*, el generador depende de una serie de acciones de la trama para alimentar la historia.

En el estado *reflection*, por otra parte, el generador añade las postcondiciones necesarias para que la historia sea coherente. Las precondiciones son descriptivas, estableciendo hechos que pueden ser explotados en una cadena de eventos que acabe resultando en una historia casual, dado a que estas permiten seguir una lógica de causa-consecuencia.

2.2.6 BRUTUS

BRUTUS (Bringsjord & Ferrucci, 1999) escribe historias cortas basándose en temas predefinidos. Su proceso de generación primero instancia un marco temático y luego lanza una simulación para que los personajes alcancen objetivos definitivos, para finalmente expandir la gramática de la historia en el output final.

Los marcos temáticos están formados por una serie de eventos que se desarrollan durante el proceso de simulación, en el cual los personajes realizan acciones con una mezcla de comportamientos proactivos y reactivos.

Los comportamientos proactivos hacen uso de precondiciones en lenguaje descriptivo para enfatizar en los eventos necesarios o los hechos establecidos que necesitan ocurrir antes de dichas acciones, lo que sirve para mantener el tema de la historia.

Por otro lado, los comportamientos reactivos dependen de las consecuencias de acciones anteriores para desencadenar nuevas, como acciones arquetípicas o diálogos, lo que aumenta la variabilidad de dicha historia.

Por tanto, se puede observar de nuevo como el contenido descriptivo es establecido antes que las acciones que lo siguen, siempre asegurándose de que las implicaciones causales dispuestas por el marco temático se respetan y las condiciones de las acciones se cumplen en orden.

2.2.7 Author

Otro punto de vista en cuanto a la generación de narrativa se ve reflejado en el sistema *Author* (Dehn, 1981), que pretende modelar la mente del autor mientras escribe una historia. *Author*, por tanto, representa el conocimiento involucrado en el proceso (personajes o momentos memorables) y cómo está organizado.

Mientras que esto no sigue el patrón de bucles de descripción seguida de acción como se ha visto en sistemas similares, aún mantiene una distinción entre establecer hechos (modelados en base a la memoria humana del autor) y usar dichos hechos para hacer avanzar la historia.

La conexión entre un desarrollo de la trama y el conocimiento que este usa es explícita, pero las cadenas de eventos causales no son tan lineales en este caso. En vez de poseer diversos hechos establecidos en una descripción, este modelo propone establecer una representación de conocimiento separada de la historia. Esta representación y la historia comparten una implícita conexión causal, asegurándose de obtener coherencia en el resultado.

2.2.8 Generación Narrativa con Redes Neuronales

Más recientemente, en un nuevo estudio (Fan, Lewis, & Dauphin, 2018) se ha propuesto un generador de narrativa que dependa de una base de datos de gran tamaño que contenga historias ya existentes y otra base de datos que contenga una colección de premisas inspiracionales para nuevas historias.

Este enfoque busca mejorar la coherencia y la estructura de las tramas resultantes usando generación jerárquica a partir de una premisa textual. Mediante la fusión de modelos basados en lenguajes más tradicionales y modelos estadísticos de secuencia a secuencia, primero genera una premisa base que más tarde es usada para generar un pasaje más largo de texto, con lo que se intenta asegurar la consistencia. Con este acercamiento, se ve un claro interés en mantener uniones causales entre diversas representaciones históricas.

El enfoque del modelo de secuencia a secuencia es preservar el orden de tanto la dimensión lingüística (palabras y frases) como la dimensión de la trama (el orden

en que se introducen y ordenan los hechos). Este modelo no separa la descripción de la acción, por lo cual se pone en cuestión si es necesario establecer primero unos hechos de forma descriptiva antes de articular acciones.

A pesar de existir cierta repetición en los resultados, los autores ponen ímpetu en que esto es un problema que surge de usar una distancia dependiente corta. Esto se puede interpretar como un síntoma derivado de la falta de una estructura narrativa de alto nivel, una decisión de diseño que se encuentra muchas veces en acercamientos modernos al problema que dependen de técnicas estadísticas.

2.2.9 Construcción Narrativa Basada en el Conocimiento

Otro estudio (Khalpada & Garg, 2019) introduce un nuevo generador de narrativa computacional en el cual los autores humanos pueden definir puntos de la trama que se usan más tarde para la generación.

Para lograr consistencia y buena estructura de la trama se incorpora el conocimiento de *ConceptNet*, un repositorio para relaciones semánticas basadas en el sentido común. Con este acercamiento, la causalidad se pide prestada de una fuente externa y es incorporada en un grafo direccional que mueve la generación de la historia.

El autor humano interactúa con el sistema definiendo los personajes y sus emociones, además de una serie de eventos iniciales, los cuales se usan más adelante para calcular sus objetivos y su viabilidad. Si el sistema encuentra que los objetivos no son alcanzables según su conocimiento interno, este requiere modificaciones en el input humano. Si el input es aceptable, se establecen, mediante una estrategia de planificación, una serie de eventos con conexiones causales.

Mientras que este acercamiento no es innovador en sus dinámicas generadoras internas, aún requiere de una construcción básica del mundo (los eventos iniciales y los personajes) que luego sirve para desarrollar la trama a través de eventos subsecuentes. De nuevo, se observa la distinción entre la descripción inicial del mundo y la cadena de desarrollos de la historia que siguen y establecen conexiones causales hacia atrás hasta el input original.

2.2.10 Computación de Historia Evolutivas

Un nuevo estudio (Wang, Bui, Petraki, & Abbass, 2018) presenta una metodología de programación evolutiva que transforma una historia escrita en una gramática basada en eventos y jerarquía representada en forma de red. En este trabajo, las representaciones formales de la historia están compuestas de eventos que dependen unos de otros en una configuración jerárquica. La definición aportada de dependencia (los eventos sirven como condiciones habilitantes de eventos subsecuentes) sugiere una relación causal. El foco en los personajes como actores, el orden temporal y un mundo compartido parecen ser claves a la hora de determinar las relaciones de dependencia.

El resultado de esta metodología es un conjunto de entidades formales (una jerarquía de eventos interdependientes con personajes, tiempo, espacio, una descripción, un tema y un objetivo) que pueden ser transformados en un cromosoma (y viceversa) apto para las técnicas de programación evolutiva que desarrollan una historia usando métricas subjetivas (coherencia, novedad, interés y calidad) y objetivas (estructura lógica de los eventos e interacciones entre los participantes) para controlar el resultado.

En general, la representación formal de una historia y las métricas usadas apoyan la hipótesis de que los sistemas de generación narrativa dependen de las nociones de tiempo y secuencia para atribuir causalidad. En cuanto a las acciones y las descripciones, sin embargo, esta metodología está diseñada para extraer las representaciones de la historia para hacerlas evolucionar con varias iteraciones. No presupone que los datos recolectados vayan a describir acciones o descripciones y, por ende, está condicionado por las historias fuente y sus estructuras.

2.3 Herramientas de Desarrollo de Videojuegos

En la actualidad, existe una gigantesca selección de herramientas para el desarrollo de videojuegos. En esta sección se explorarán algunas de estas herramientas, teniendo en cuenta que la decisión sobre su uso o no se discutirá detalladamente en el **CAPÍTULO IV**: IMPLEMENTACIÓN.

2.3.1 SDL

En primer lugar, un posible acercamiento al desarrollo de cualquier videojuego puede ser la implementación de un motor propio. Una herramienta útil para dicho propósito es SDL (*Simple DirectMedia Layer*), desarrollada inicialmente por Sam Lantinga.

SDL es una librería multiplataforma diseñada para proveer acceso a bajo nivel a audio, ratón, teclado, joystick y hardware de gráficos mediante OpenGL y Direct3D. SDL está escrita en C y funciona con C++, existiendo también maneras de usarlo con lenguajes como C# y Python.

Haciendo uso de esta librería, se puede empezar el desarrollo de un videojuego prácticamente de cero, diseñando e implementando por completo un motor propio antes de poder darle uso para la creación del juego. Además, SDL es una librería gratuita y de código abierto, por lo que ha tenido muchas contribuciones.

2.3.2 Phaser

Otra herramienta de desarrollo actual es Phaser 3.0, desarrollada por Photon Storm. Phaser es un framework utilizado para la creación de videojuegos en 2D de navegador, tanto para juegos de ordenador como juegos para dispositivos móviles. Utiliza HTML5 y el único requerimiento para funcionar es que el navegador objetivo disponga de la etiqueta <*canvas*> de HTML. En ordenador, esto incluye Chrome, Firefox, Safari, Opera, etcétera.

Algunos elementos de los que Phaser se hace cargo como motor son las imágenes, los *spritesheets* y los *tweens* (imágenes estáticas o dinámicas y un sistema para animarlas); el control del input; y las físicas del juego, pudiendo elegir entre tres diferentes opciones dependiendo de la exactitud deseada en estas.

2.3.3 RPG Maker

Otra de las herramientas más populares debido a su sencillo uso es RPG Maker, una amplia serie de programas para el desarrollo de videojuegos de rol (RPGs, *Role-Playing Games*) desarrollados por ASCII Corporation y publicada por Kadokawa Games. Con estos programas, solamente se pueden desarrollar videojuegos en 2D,

en los cuales los entornos suelen estar compuestos haciendo uso de *TileSets*, conjuntos de cuadrículas que pueden combinarse para crear escenarios. A pesar de ser una herramienta muy limitada al desarrollo de RPGs, es de muy fácil uso, lo que la convierte en una alternativa popular. Para el desarrollo, la última versión del programa permite no hacer uso de código o programar en Ruby o JavaScript.

2.3.4 Unreal Engine

Uno de los nombres más populares en el campo de las herramientas y motores para el desarrollo de videojuegos es Unreal Engine, desarrollado por Epic Games. Al tratarse de una de las herramientas más poderosas y tener versión gratuita desde 2014, se ha convertido en una de las más populares para los desarrolladores de videojuegos.

Unreal Engine está escrito en C++ y, desde su cuarta versión, usa ese mismo lenguaje para el *scripting*, el cual le otorga altos niveles de rendimiento y una gran capacidad de depuración. Además, también desde su cuarta versión, Unreal incluye los llamados *Blueprints*, un sistema de *scripting* visual para que los no desarrolladores (como los diseñadores y los artistas) también puedan programar comportamientos deseados en el videojuego. Sin embargo, el uso de Unreal Engine no es tan sencillo como las otras alternativas discutidas en este apartado, por lo que este queda más reservado para desarrolladores con experiencia.

2.3.5 *Minecraft*

Otra opción que se puede tener en cuenta para el desarrollo de experiencias narrativas es hacer uso de videojuegos ya existentes y modificarlos con *mods*, es decir, extensiones del software que modifican un videojuego original proporcionando nuevas posibilidades, ambientaciones, personajes, diálogos, objetos, etcétera. Esto podría ser viable en títulos como *Minecraft* (Mojang, 2011), uno de los videojuegos más populares de la historia. *Minecraft* dispone de una comunidad muy extensa de desarrolladores de *mods* y de herramientas potentes para su desarrollo.

Aprendiendo a usar dichas herramientas, se podría desarrollar una experiencia narrativa dentro del videojuego. Esto tiene como ventajas el hecho de no tener que preocuparse por desarrollar el juego en sí, sino poder centrar todo el foco del desarrollo a la experiencia narrativa y, en nuestro caso, por ejemplo, el sistema de DM. Además, *Minecraft* está escrito en Java, lo que permitiría utilizar el código del AIMA para la IA.

2.3.6 Unity

Por último, una de las herramientas más populares para el desarrollo de videojuegos es Unity, un motor un motor de videojuegos multiplataforma creado por Unity Technologies y lanzado en 2005. Unity permite la creación de videojuegos tanto en 3D como en 2D, permitiendo a sus usuarios centrarse plenamente en la programación de la lógica de sus videojuegos, pudiendo dejar apartados más técnicos y complicados en manos del motor, como el renderizado, la simulación física, el audio, etcétera.

Además, Unity consta de una gran serie de componentes predefinidos que facilitan y aceleran la implementación de videojuegos en el motor y las variables de todos los componentes se pueden modificar de forma sencilla en el propio editor de Unity, evitando así tener que editar código cada vez que se quiera cambiar uno de los valores, y facilitando la construcción de sistemas flexibles y escalables.

CAPÍTULO III: DISEÑO DEL GESTOR DE NARRATIVA

Uno de los objetivos de este proyecto es implementar un sistema de narrativa procedural centrado en la trama, con algunos conceptos adicionales presentes en los nuevos modelos basados en el discurso. Suministrando al sistema un estado inicial, un objetivo y un conjunto de reglas de interacción se genera un espacio de estados, bajo un estricto control del autor para determinar el discurso y orientar la trayectoria narrativa del jugador.

Antes de concretar cada uno de los subsistemas que han sido diseñados para el motor de IA, nos acercaremos de forma genérica a la arquitectura completa del sistema. Para esta explicación, se tomará la misma base conceptual e inspiración que se utilizará para el prototipo final: el agente inteligente (Russell S., 2018).

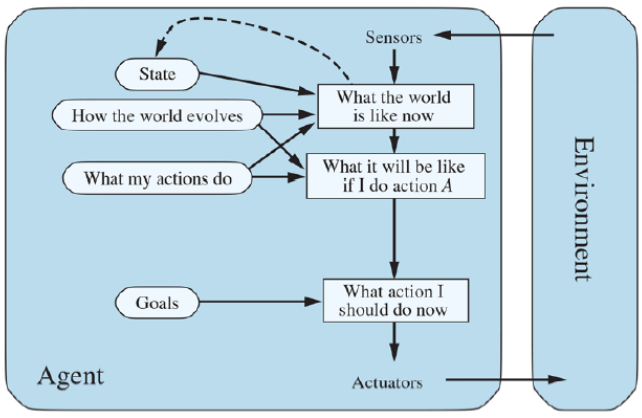

Figura 5: Modelo de agente basado en objetivos según *Artificial Intelligence: A Modern Approach*, 4ª ed. Este agente toma decisiones de acción basándose no sólo en sus percepciones, sino también en información sobre sus objetivos o metas.

El sistema implementado puede percibir su entorno a través de sensores, y modificarlo a través de actuadores: tras recibir una secuencia de percepciones, los subsistemas del motor de IA mapean esas percepciones a acciones concretas. El objetivo era que, además, el sistema pudiese modelar el entorno en estados, siendo capaz de evaluarlos y realizar acciones en consecuencia para cumplir ciertos objetivos. Éste es el modelo que se utiliza para diseñar este sistema de narrativa procedural, y que se bautizará como Drama Manager.

3.1 El Sistema de Gestión de la Narrativa

El Drama Manager (DM) implementado es un módulo que integra un motor de Inteligencia Artificial, conformado por tres subsistemas básicos: el subsistema de sensores (Evaluador), el subsistema de resolución de problemas (Resolutor) apoyado en la unidad de representación del conocimiento (Base de Conocimiento), y el subsistema de actuadores (Generador).

Por su orientación a la trama, el DM recibe durante su inicialización un conjunto de reglas de interacción, así como la Agenda Narrativa, el conjunto de estados objetivo que deben visitarse para completar la trama. El entorno virtual con el que se relaciona el DM es un videojuego que se podría esquematizar en dos elementos: el mundo interactivo (que es la representación audiovisual) y el *Game Manager* (que gestiona internamente los elementos del mundo interactivo).

Durante este capítulo, se hará referencia a diferentes personas que interactúan con el sistema. Por un lado, se tiene al jugador, que interactúa en el entorno virtual o videojuego. Por otro lado, se tiene al usuario del sistema, haciendo referencia a la persona que utiliza nuestro sistema para integrarlo en un videojuego. El diseñador o autor es la persona encargada de escribir tanto la Agenda Narrativa como las reglas de interacción que se suministran al sistema.

En ocasiones, puede ocurrir que las labores del usuario del sistema y el diseñador se solapen, como ha ocurrido en el caso de este proyecto, al haber escrito la historia, diseñado el videojuego e implementado el sistema en el mismo.

Se deben destacar dos ideas clave a la hora de diseñar el DM. Por una parte, se planteó la necesidad de maximizar la independencia del DM frente al videojuego que lo utilizase y, por tanto, se necesitaba implementarlo de la forma más agnóstica posible; se darán más detalles en el **CAPÍTULO IV**: IMPLEMENTACIÓN.

En línea con esto, y en pos de favorecer la libertad creativa de los usuarios de nuestro sistema, las funciones clave del DM, aquellas que utilizan los subsistemas de evaluación, resolución y generación, debían ser ajenas al DM.

Idealmente, serían recibidas como parámetros durante la inicialización del sistema, pero se podrá, además, facilitar su actualización en tiempo de ejecución, si el usuario lo desea. Como herramientas adicionales de configuración de nuestro sistema, el DM ofrecerá parámetros de control para los procesos de evaluación y búsqueda. Estas funcionalidades se explorarán en profundidad en los correspondientes subapartados.

En una iteración normal, el sistema se comporta como se puede apreciar en el diagrama. Las acciones del jugador sobre el entorno virtual afectan al mundo interactivo y se comunican al DM, idealmente, a través del *Game Manager*, en un proceso de representación formal del estado actual del mundo llevado a cabo por el Evaluador. Este subsistema, además, valora si el estado actual cumple con los requisitos del estado objetivo fijado por la Agenda Narrativa.

Si no es así, el DM manda el estado actual y el estado objetivo al Resolutor, que inicia un proceso de búsqueda en el espacio de estados, utilizando la Base de Conocimiento como respaldo. Este subsistema de resolución de problemas no se comunica en ningún momento con el entorno virtual del videojuego, utilizando un lenguaje de representación simbólica que sólo pueden interpretar el Evaluador y el Generador.

La solución de la búsqueda, si existe, se envía al subsistema de generación. El Generador procesa la solución y, a través del *Game Manager*, crea el contenido necesario para orientar la trayectoria narrativa del jugador.

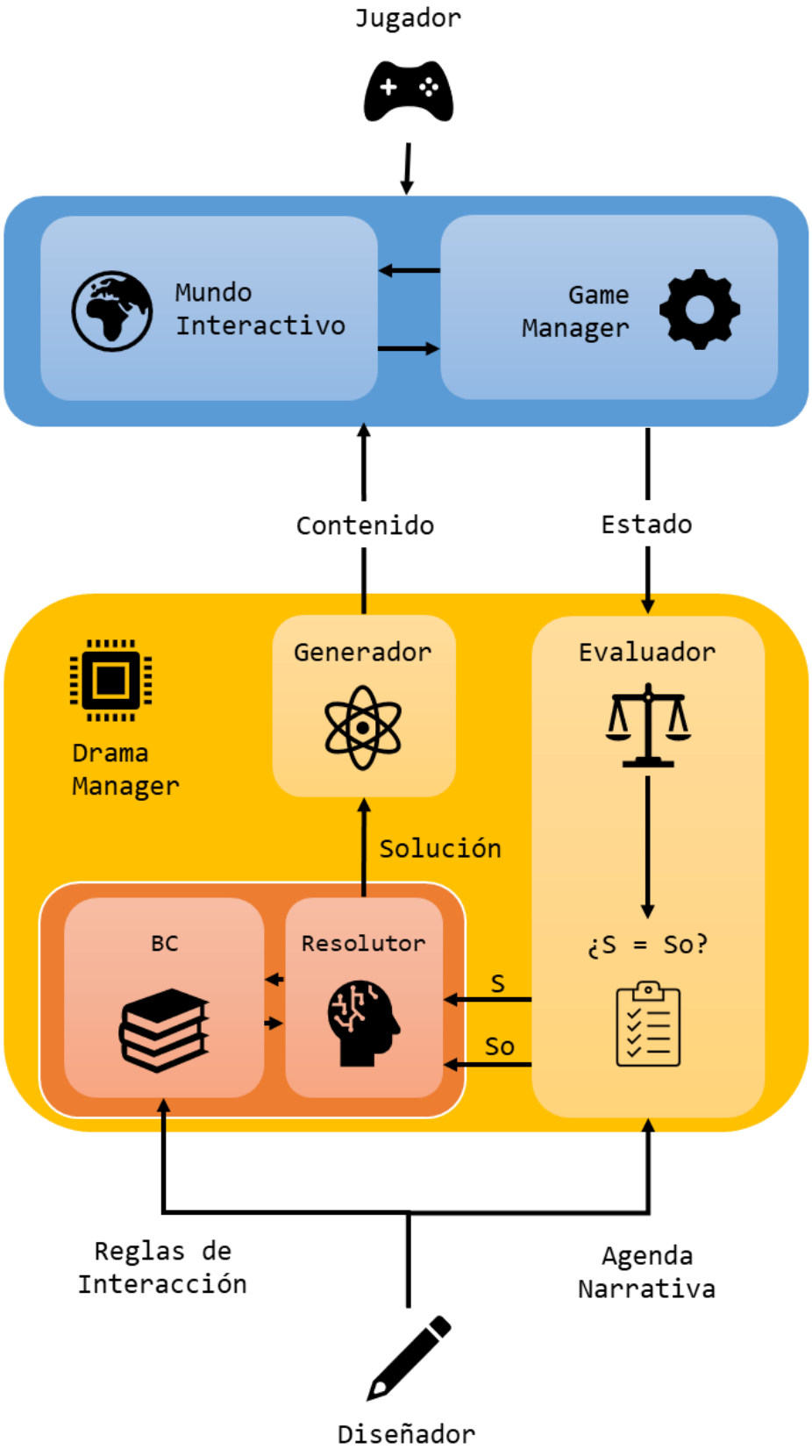

Figura 6: Esquema general del funcionamiento del Drama Manager. El módulo azul representa el entorno del virtual. El módulo amarillo representa el módulo de IA, es decir, el DM y sus subsistemas. El subsistema de resolución, en rojo, se destaca como el único en utilizar un lenguaje únicamente simbólico no entendible por el entorno virtual.

3.2 El Subsistema de Evaluación

El subsistema de evaluación tiene como principal objetivo extraer del mundo interactivo la información necesaria para saber en qué estado se encuentra la historia; es, por tanto, el principal sensor de nuestro módulo de IA. Por tanto, se comunica con el entorno virtual y con el subsistema de resolución de problemas del DM. El Evaluador también es el encargado de gestionar la Agenda Narrativa, que se le suministrará durante la inicialización del DM.

La Agenda Narrativa se entiende como una sucesión de estados que describen los puntos álgidos de la trama, formalizados del mismo modo que se representa el mundo; es decir, el lenguaje común en todo el DM es el de los estados (se profundizará más en la representación del conocimiento en el subapartado 3.3.1).

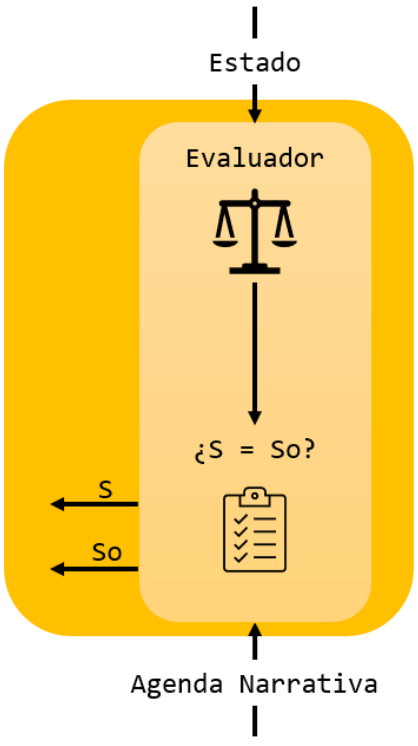

Figura 7: Subsistema de Evaluación

Cada vez que se completa un estado objetivo se pasa al siguiente, y así sucesivamente hasta completar la trama. En ese momento, el DM deja de tener utilidad y deja de actualizarse y generar contenido.

El diseñador podrá añadir nuevos elementos a la Agenda Narrativa en cualquier momento, con vistas a que el sistema pudiese verse ampliado con un generador de historia que resultase compatible con nuestro DM.

Cuando se activa el DM, el Evaluador elabora un estado a partir de la situación que percibe en el mundo interactivo. Inmediatamente después, debe comprobar si el estado actual coincide con el estado objetivo que marca en ese momento la Agenda Narrativa, para pasar al siguiente si fuese preciso.

Esta comparación entre estados puede ser problemática, ya que en muchos casos el propio diseñador no buscará que los puntos importantes de la trama se cumplan al 100%. Este movimiento, además, puede favorecer la variabilidad narrativa de un jugador a otro.

En cualquier caso, nuestro sistema busca la flexibilidad para el usuario, por lo que el Evaluador requerirá en su inicialización de una función que se ha llamado de completitud. La función de completitud se encarga de comparar dos estados, uno actual y uno objetivo, y determinar en qué grado se cumple el estado actual con respecto al objetivo. Este grado, como se indicaba en el subapartado anterior, puede configurarse como un parámetro de inicialización más del DM.

Esta comprobación de completitud podría enfocarse desde varios puntos de vista. Se podrían ver los estados como una lista de hechos, que se han cumplido o no; la función de completitud, en tal caso, sólo tendría qué comprobar cuántos de esos hechos coinciden en el estado actual y el objetivo, y el Evaluador determinaría si se considera que se ha alcanzado adecuadamente el punto de interés de la Agenda Narrativa.

No obstante, debido a la implementación final de los estados (en la que se profundizará en el **CAPÍTULO IV**: IMPLEMENTACIÓN), se ha optado por una comparación directa de igualdad para el videojuego implementado; recuérdese que cada usuario podrá configurar cómo se realiza esta comprobación de completitud.

3.3 El Subsistema de Resolución

Cuando el Evaluador captura el estado actual del entorno virtual, puede ocurrir que se decida que el estado no es lo suficientemente completo, y por tanto aún no se ha alcanzado el estado objetivo de la Agenda Narrativa. En este caso, el Evaluador envía ambos estados al sistema de resolución de problemas. El Resolutor es, como su propio nombre indica, el subsistema encargado de resolver problemas.

Un problema se describe formalmente como (Bulitko, Björnsson, Sturtevant, & Lawrence, 2011):

1. Un espacio de estados que contiene todas las posibles configuraciones de los atributos relevantes del problema.
2. Un estado inicial que describe la situación desde las que el proceso de resolución puede comenzar.
3. Un estado objetivo que sería aceptable como solución del problema.
4. Un conjunto de reglas que describen las acciones disponibles.

Si reúne estas características, el problema puede resolverse aplicando las reglas establecidas, en combinación con una estrategia de control apropiada, para navegar por el espacio de estados hasta encontrar un camino desde el estado inicial hasta el estado objetivo. En Inteligencia Artificial, este proceso se conoce como búsqueda.

Un estado puede entenderse como la representación de los elementos del problema en un determinado momento. En consecuencia, el espacio de estados es el conjunto de todos los estados alcanzables desde el estado inicial.

Un espacio de estados conforma un grafo, en el cual los nodos son los estados y las aristas entre los nodos son las acciones (Bulitko, Björnsson, Sturtevant, & Lawrence, 2011). En el espacio de estados, un camino es una secuencia de estados conectados por una secuencia de acciones. La solución a un problema es parte del grafo generado por el espacio de estados: es el camino que une el estado inicial con el estado objetivo.

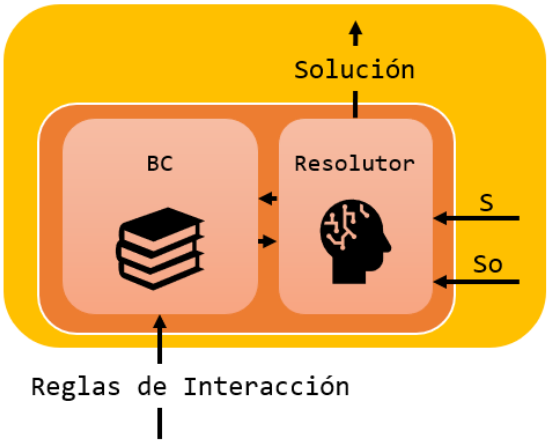

Figura 8: Subsistema de Resolución de Problemas

Parece obvio que nuestro problema se adapta perfectamente a estas características: utilizando el estado inicial y objetivo suministrados por el Evaluador, así como el conjunto de reglas de la Base de Conocimiento, el Resolutor es capaz de utilizar la búsqueda para encontrar un camino que conecte el punto inicial de la trama con el punto final, es decir, una solución.

Un aspecto importante a la hora de diseñar nuestro Resolutor fue, por tanto, escoger un algoritmo de búsqueda adecuado para implementar:

a. ***Depth-First Search* (DFS):** con este algoritmo se explora en profundidad el grafo; si no encuentra una solución, vuelve atrás y considera otro camino. En el peor caso, encontrar una solución llevaría a recorrer todo el espacio de estados, con un coste en tiempo del orden de $O(|V| + |E|)$, siendo V el número de nodos y E el número de aristas; esto puede ser especialmente grave si se tiene en cuenta la explosión combinatoria que supone tener un número moderado de reglas de interacción.
b. ***Breadth-First Search* (BFS):** este algoritmo realiza una búsqueda intensiva, expandiendo todos los nodos de un nivel antes de pasar al siguiente. De nuevo, al tratarse de una estrategia de búsqueda no informada, se puede llegar a recorrer todo el espacio de estados antes de encontrar una solución; al menos, en este caso, la solución será óptima en referencia al número de pasos (acciones) que hay del estado inicial hasta el objetivo.

c. ***A-Star* (A*):** el algoritmo de búsqueda A* permite llevar a cabo una búsqueda informada, en la que la expansión de los nodos del grafo se hace con cierto conocimiento del problema, permitiendo así establecer preferencias para orientar la búsqueda hacia unos nodos u otros. Esta preferencia viene definida por una función de coste, que indica el coste del camino desde el nodo inicial hasta el nodo expandido, y una función heurística, que estima el coste hasta el nodo objetivo.

Aunque en la implementación final de nuestro sistema incluye los tres algoritmos, se decidió utilizar el A* por defecto para los experimentos de este proyecto. Por un lado, la optimalidad no era calve en este sistema; la narrativa no es optimizable, puesto que el arte es subjetivo: no se puede determinar que una historia es buena de forma universal. Por otra parte, la necesidad de que el algoritmo fuese completo y razonablemente óptimo para nuestra heurística hizo que la elección final fuese el A* (Cui & Shi, 2010).

En cualquier caso, queda en manos del usuario decidir qué algoritmo utilizar para realizar las búsquedas en el subsistema del Resolutor, y siempre podrían ampliarse las opciones implementando nuevos algoritmos. El Resolutor recibe durante su inicialización, además, tres funciones claves para la búsqueda que deben ser implementadas por el usuario:

- **Función de Coste:** esta función indica con un valor numérico el coste que tendría pasar de un estado a otro aplicando una determinada acción.
- **Función Heurística:** esta función indica con un valor numérico una estimación del coste de llegar desde el estado actual hasta el objetivo.
- **Función Objetivo:** esta función comprueba si un estado dado es un estado objetivo, finalizando así el proceso de búsqueda.

El diseño de las funciones de coste y heurística determina en gran medida el éxito del sistema en su conjunto. Las particularidades sobre estas funciones y su implementación en el videojuego se explicarán en el **CAPÍTULO IV**: IMPLEMENTACIÓN.

Durante el proceso de búsqueda, el espacio de estados se va generando con cada expansión que se realiza sobre los nodos del grafo. La función de expansión del Resolutor se comunica directamente con la Base de Conocimiento, que se explorará en el siguiente subapartado, suministrando las acciones de las que dispone el sistema para aplicarlas, si procede, al nodo en cuestión.

En un primer acercamiento, se valoró si al expandir los nodos las acciones debían aplicarse una a una o por lotes. La segunda vía habría desembocado en la implementación de una función que ofreciese todas las combinaciones sin repetición de todas las acciones.

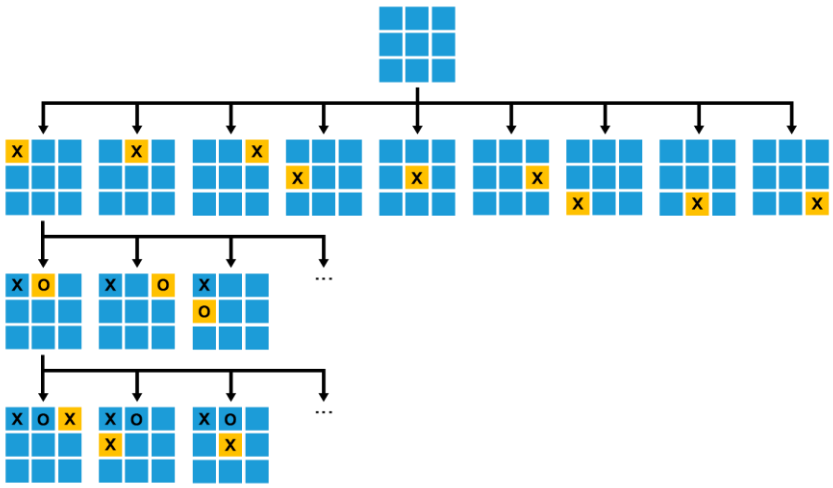

Figura 9: Ejemplo de generación del espacio de estados del juego clásico Tic-Tac-Toe. Con 9 acciones por turno la explosión combinatoria ya se hace patente en las primeras expansiones de cada estado. Estos grafos pueden volverse inmanejables muy rápido.

Esta explosión combinatoria habría supuesto un duro golpe a la eficiencia del Resolutor (Nelson & Mateas, 2008), por lo que finalmente se decidió reforzar el trabajo de autoría en el diseño de las acciones para poder aplicarlas una a una sólo cuando fuese necesario.

En ese sentido, el Resolutor cuenta con un parámetro configurable por el usuario del sistema que indica, con un valor numérico, qué nivel de coherencia deben cumplir las acciones. Esta "coherencia" no debe confundirse con el término utilizado habitualmente en el campo de estudio de la narrativa interactiva; actúa como un indicador de cómo de prometedora es dicha acción.

Otro detalle de configuración del DM presente en el Resolutor es un valor de control de la expansión, que indica el número máximo de expansiones al que puede llegar la búsqueda antes de cortarse sin devolver una solución. Si el diseño del estado y las reglas de interacción es pobre, la búsqueda puede volverse ineficiente (Cui & Shi, 2010) y llegar a bloquear el progreso del juego, por lo que parece un parámetro adecuado paras controlar estas situaciones.

En la misma línea, un diseño pobre por parte del usuario del sistema puede arrojar espacios de estados sin solución. El sistema garantiza que el algoritmo de búsqueda por defecto es completo (aunque se puede regular cuánto se demora en encontrar una solución), pero no puede garantizar que la heurística lo sea. Si el Resolutor no encuentra una solución, lanza una excepción para avisar al usuario de esta eventualidad. Si todo va bien y el Resolutor encuentra una solución, esta se envía al subsistema de generación para poder ser interpretada y utilizada en el mundo interactivo correspondiente.

3.3.1 Representación del Conocimiento

Con el fin de resolver los problemas más complejos que se encuentran en algunos campos de la Inteligencia Artificial, se hace necesaria una base de conocimiento y algún mecanismo para manipular ese conocimiento y crear soluciones (Lin, Zhao, Huang, Liu, & Pu, 2020). En este proyecto era necesario representar de modo formal las historias, y, más concretamente, modelar un entorno virtual y los eventos que en él se producen. Un módulo de IA requiere de un problema bien definido para procesar y proveer soluciones a la altura: debe ser capaz de obtener el conocimiento de su entorno y representarlo de manera formal para que pueda procesarse y computarse una respuesta.

El conocimiento es una descripción del mundo; a más conocimiento, más competente es un sistema inteligente. La representación del conocimiento es la forma en la que se codifica ese conocimiento, y define la eficiencia de un sistema inteligente al realizar una tarea. En cualquier caso, distintos tipos de conocimiento requieren distintos tipos de representación.

Nuestro DM cuenta con un subsistema que cumple la función de representación, el Evaluador, y otro subsistema que cumple la función de interpretación, el Generador. Es por esto por lo que, a la hora de diseñar las funciones de evaluación y de generación, el usuario ha de tener en cuenta que ambos sistemas comparten aspectos formales con el mundo interactivo.

Por decirlo de una manera más sencilla, tanto el Generador como el Evaluador hablan el mismo idioma que el videojuego, pero también hablan otro idioma que sólo ellos y el Resolutor entienden: esa es la representación del conocimiento que utiliza nuestro sistema.

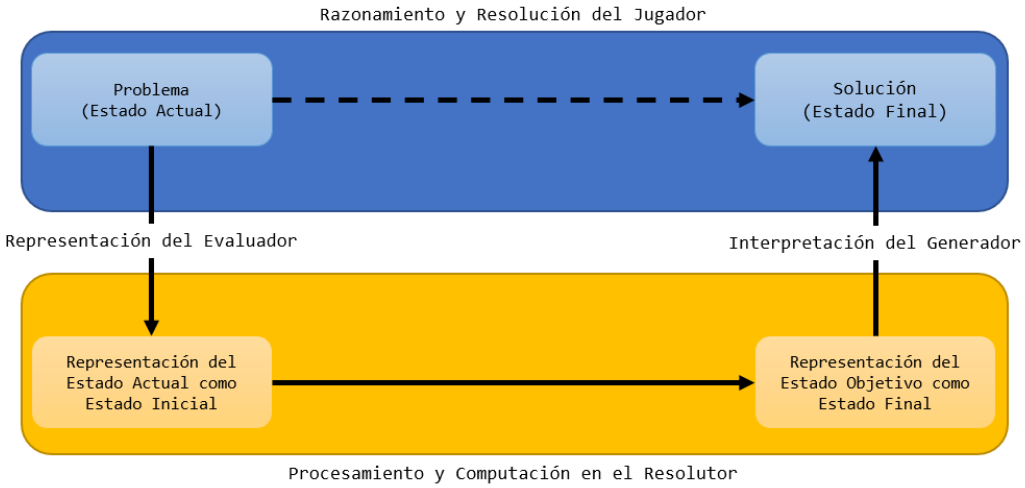

Figura 10: Esquema del funcionamiento del conocimiento en nuestro sistema de IA. La zona azul simboliza una representación informal, mientras que la amarilla simboliza una representación formal.

Hoy en día no existe ninguna alternativa genérica para un procesado completo de entrada por lenguaje natural. A pesar de que hay alternativas (Wolf, y otros, 2019), ninguna ofrece una robustez genérica que reemplace con fiabilidad a una representación explícita de conocimiento. La manera en la que se realizará una representación formal del conocimiento son los estados, y las acciones definen cómo pueden modificarse esos estados.

Las acciones son el conjunto de reglas de interacción básicas con las que se surte la Base de Conocimiento de nuestro sistema en el momento de su inicialización, aunque existe la posibilidad de añadir nuevas acciones en tiempo de ejecución, si el usuario lo cree oportuno.

Todas las acciones representan hechos, verdades irrevocables del mundo interactivo en el que se definen; es responsabilidad del usuario diseñar adecuadamente cada una de las acciones que se suministran a la Base de Conocimiento. Cuando una acción se aplica sobre un estado, el estado cambia, por lo que se modifica el conocimiento almacenado por nuestro agente inteligente.

La Base de Conocimiento, además de mapear las acciones y proporcionárselas al Resolutor durante la búsqueda, se utiliza indirectamente para decidir qué acciones son legales en un determinado estado, y, de serlo, qué acciones son prometedoras en dicho estado. Una explicación más detallada sobre la implementación de las acciones y su integración en la Base de Conocimiento puede consultarse en el **CAPÍTULO IV**: IMPLEMENTACIÓN.

3.4 El Subsistema de Generación

Cuando el Resolutor, con el apoyo de la Base de Conocimiento, encuentra una solución al problema planteado, esta se envía al subsistema de generación. El Generador se encarga de recibir la solución e interpretarla mediante la función generadora.

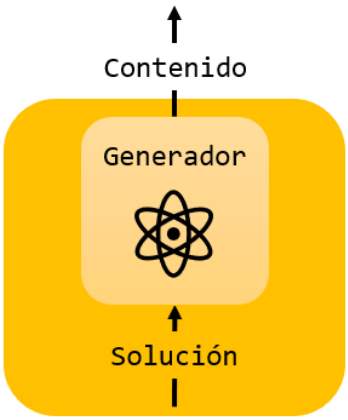

Figura 11: Subsistema de Generación

La función generadora, al igual que las demás, debe implementarla el usuario para adaptarla lo más posible a sus necesidades. Idealmente, la función generadora se comunicará con el entorno interactivo a través de un gestor o *Game Manager*.

A la hora de diseñar esta función, hay que destacar que la solución se compone de dos caminos completos: uno de estados, y otro de acciones que han producido ese camino de estados. De este modo, puede determinarse más fácilmente qué contenido generar para orientar la trayectoria narrativa del jugador.

La generación procedural de contenido es un tema extensivo y complejo que no se cubre en este proyecto. No obstante, esta función de generación puede utilizarse en conjunto con sistemas de generación de terreno, de música, o cualquier otro conjunto de datos digitales. En este proyecto, se limitará a demostrar sus capacidades más básicas, utilizándolo para modificar algunos de los elementos interactivos del videojuego que se utilizará como ejemplo.

CAPÍTULO IV: IMPLEMENTACIÓN DEL GESTOR DE NARRATIVA

Si bien el capítulo anterior se ha visto centrado en un acercamiento puramente teórico a este proyecto, en este capítulo se hará un recorrido por toda la implementación. Aunque se pondrá el foco en la implementación del motor de Inteligencia Artificial, también se hará un análisis de la implementación de los dos prototipos que han sido desarrollados.

Entender cómo están diseñados estos videojuegos es de gran importancia para comprender todas las implicaciones del proyecto, ya que la comunicación entre los dos módulos es uno de los aspectos más complejos de la implementación de todo el sistema.

Además, esto dará la oportunidad de introducir una de las herramientas que más ha resultado de ayuda a la hora de agilizar el desarrollo de los prototipos, *TopDown Engine*. Otras herramientas, sin embargo, fueron descartadas.

En primer lugar, el desarrollo de un motor propio haciendo uso de SDL no ha resultado de interés en el contexto de esta investigación. Debido a que los videojuegos en sí no van a ser mucho más que pruebas para el sistema de DM desarrollado, no es oportuno tener que dedicar tanto tiempo de la investigación al desarrollo del motor y las herramientas para poder crear dicho videojuego.

También se descartó el uso de Phaser 3.0, pues este hubiese obligado a desarrollar los videojuegos para navegador haciendo uso de JavaScript, lo cual podría haberse traducido en problemas de rendimiento en la IA del DM.

RPG Maker es posiblemente una de las herramientas que podrían haber resultado más útiles para la implementación de los videojuegos de esta investigación, debido a su simple uso y a que el género RPG resulta adecuado para los videojuegos narrativos. Sin embargo, también se descartó, especialmente por la falta de flexibilidad que ofrece a la hora de programar módulos fuera de los sistemas que propone.

Por otra parte, una gran falta de familiarización con la herramienta ha hecho que Unreal Engine sea menos interesante para este proyecto. Además, el motor utiliza como lenguaje C++, y sus particularidades en el manejo de la memoria y los tipos de datos habrían ralentizado el desarrollo de los módulos más centrados en la Inteligencia Artificial.

A pesar de que la idea de utilizar *Minecraft* parecía en un principio atractiva, la realidad es que para desarrollar *mods* hace falta dedicar una gran cantidad de tiempo a entender cómo funciona el videojuego y la API de *modding*, y qué limitaciones presentan. Esta situación, de nuevo, haría desperdiciar demasiado tiempo en el propio desarrollo de los videojuegos, los cuales no son el foco de esta investigación.

Se podría extender este razonamiento a casi cualquier videojuego con un fuerte soporte para *modding*, aunque otras opciones no fueron consideradas con tanto peso como *Minecraft*.

Gracias al potente sistema basado en componentes de Unity, la fácil comunicación entre estos y la posibilidad de delegar aspectos técnicos y muy laboriosos del desarrollo de videojuegos en el motor, crear videojuegos resulta mucho más sencillo. Esto convierte a Unity una herramienta muy atractiva a la hora de desarrollar prototipos y experimentos de investigación como los necesarios para este proyecto.

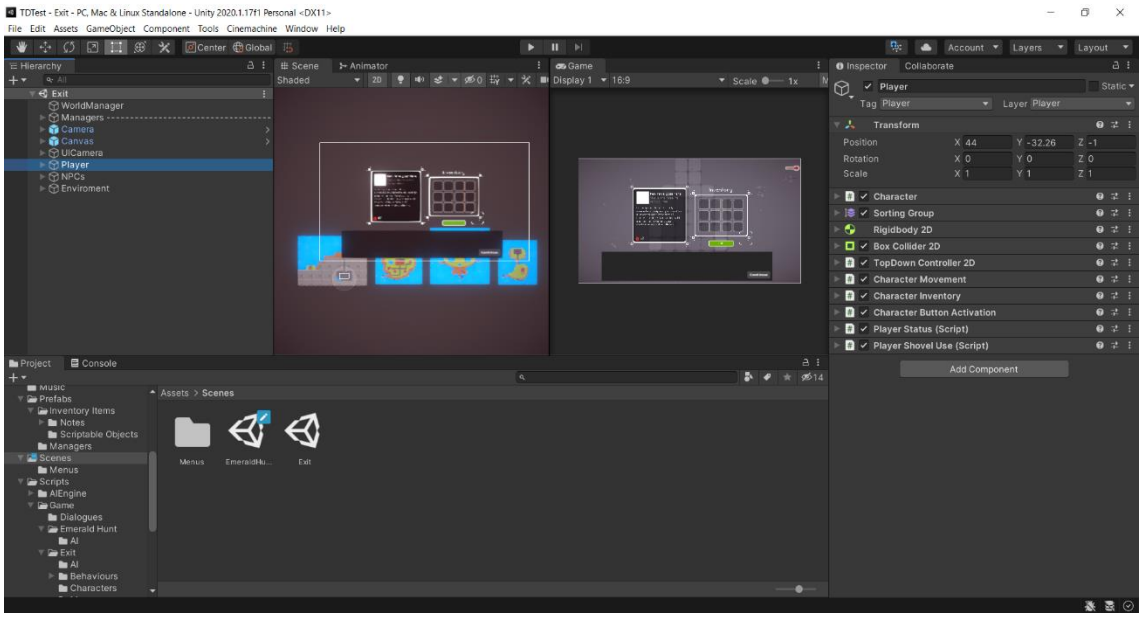

Figura 12: Editor de Unity, en la escena "Exit" para el videojuego *Tell No Tales* desarrollado. A la derecha, en la pestaña "Inspector" se puede ver la lista de componentes del GameObject "Player" seleccionado en la escena.

Además, todos los investigadores estamos familiarizados con el uso del motor. Por estas razones, se ha elegido a Unity como herramienta para el desarrollo del proyecto, tanto de la IA como de los videojuegos necesarios, haciendo uso del lenguaje predeterminado en esta, C#. Hacer uso de Unity permitirá conseguir resultados adecuados para la investigación en menos tiempo que con las alternativas ya discutidas.

Además, Unity consta de un sistema de control de versiones propio, Collab. Este permitirá trabajar de forma coordinada, donde todos los investigadores podrán actualizar sus cambios y descargar nuevas actualizaciones de forma sencilla.

4.1 Estructura del Proyecto

Tal y como se explicó en el apartado correspondiente del **CAPÍTULO II**:

TRABAJO PREVIO, este proyecto se ha desarrollado con el motor de videojuegos Unity. Por una parte, se ha implementado un módulo dedicado al motor de Inteligencia Artificial, núcleo del proyecto, y por otra, un módulo dedicado al videojuego sobre el que se probarán las capacidades del sistema de IA.

Así, el módulo dedicado al motor de IA es el que implementa el Drama Manager y todos los subsistemas que fueron introducidos en el capítulo anterior. El módulo dedicado al videojuego recoge todos los scripts que se han empleado en el desarrollo del entorno de pruebas.

La decisión de utilizar Unity era indisociable de la de utilizar C# como lenguaje de programación, y esto ofreció algunas ventajas en cuanto a organización del proyecto que se explicarán a continuación. Para garantizar la independencia de ambos módulos, se creó una definición de ensamblador (*Assembly Definition*) para cada uno: *AIEngine.asmdef* y *Game.asmdef*.

Además, todo el código fuente del proyecto se alberga en la carpeta *Assets/Scripts* del directorio raíz del proyecto de Unity. Esta organización de directorios se traslada al propio código mediante la implementación de un árbol de espacios de nombres (*namespaces*) que comparte denominación con los directorios correspondientes.

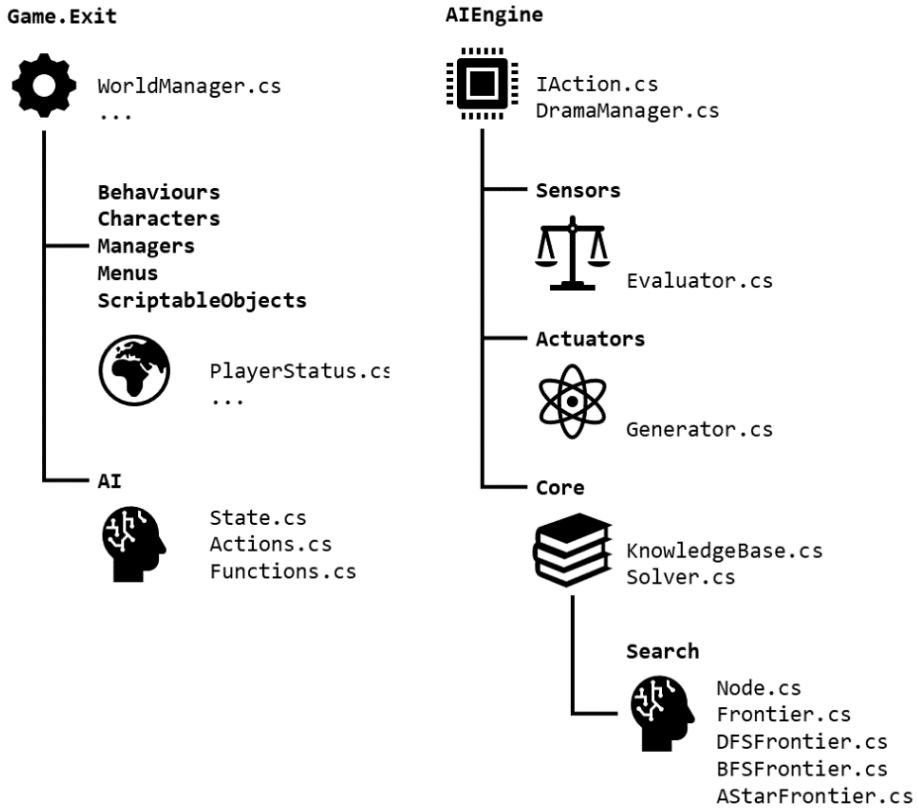

Figura 13: Árbol de directorios y archivos de código fuente del motor de Inteligencia Artificial y del módulo de juego, tal y como se encuentran en el proyecto de Unity. No se incluyen todos los scripts utilizados en el videojuego.

Con esta organización se redujo el nivel de dependencia entre los módulos al mínimo imprescindible: mientras que el *assembly* del módulo de juego hace referencias al *framework* que le da soporte y al motor de IA (aunque sólo en tres scripts), el *assembly* del módulo de IA no tiene ninguna dependencia externa. Esto supone, por otra parte, una facilidad añadida para el usuario de este sistema. En los próximos apartados se explorará la implementación de cada módulo.

4.2 El Módulo de Inteligencia Artificial

El módulo de más peso específico en este proyecto es el dedicado al motor de Inteligencia Artificial, puesto que en este módulo es donde se ha implementado el sistema de gestión de la acción dramática, el Drama Manager, así como todos sus subsistemas, cuyo diseño ya se exploró en el **CAPÍTULO III**:

.

Cada subsistema dentro del DM tiene su propio directorio; dentro del directorio *AIEngine*, se encuentran los directorios *Sensors*, *Actuators* y *Core*, y a su vez, dentro de este, el directorio *Search*. Esta organización se refleja, como se comentó en el apartado anterior, en la estructura de espacios de nombres. Así, el *namespace AIEngine* alberga las clases *IAction* y *DramaManager*, el *namespace AIEngine.Sensors* la clase *Evaluator*, el *namespace AIEngine.Actuators* la clase *Generator*, el *namespace AIEngine.Core* las clases *KnowledgeBase* y *Solver* y el *namespace AIEngine.Core.Search* las clases *Node, Frontier, DFSFrontier, BFSFrontier* y *AStarFrontier*.

Esta estructura trata de encapsular los subsistemas del DM, exponiendo tan sólo aquellos elementos que deben ser visibles desde el módulo del videojuego: la propia clase *DramaManager* y la interfaz para las acciones. El otro elemento clave del sistema, y que se menciona con asiduidad en el **CAPÍTULO III**:

, es el estado, cuya implementación específica se verá en el apartado dedicado al módulo de juego.

4.2.1 Estados como Parámetros de Clases Genéricas

El lenguaje C# permite elaborar clases genéricas, que encapsulan operaciones que no son específicas de un tipo de datos determinado, permitiendo parametrizarlos y hacer más flexible y reutilizable nuestro código.

Y eso es justo lo que se necesita para el DM propuesto: que cada subsistema y cada función sea lo suficientemente flexible como para trabajar con la implementación que haga cada usuario de los estados. Así, todas las clases de este módulo de IA son, en realidad, genéricas, ya que reciben un parámetro que representa el tipo estado.

En este caso, el tipo estado queda representado por la clase *State*, cuyo código fuente, como se indica en el esquema de directorios, se encuentra en el subdirectorio *AI* del módulo de juego (*State.cs*). De nuevo, se da flexibilidad al usuario para nombrar y organizar el tipo estado a su criterio, pero parece bastante razonable tener un directorio reservado para las clases que interactúan con el motor de IA, separándolas del resto del código del juego.

La implementación de *State* varía en función del videojuego, ya que debe reflejar el contenido específico de este. Sin embargo, C# ofrece algunas ventajas que deberían aprovechar todas las implementaciones. En primer lugar, hacer de todos los atributos de *State* propiedades lo convierte de facto en una suerte de *struct*; esto facilita enormemente la modificación de los estados. Combinando esto con la definición de una serie de constructoras adecuadas, trabajar con estados se vuelve mucho más cómodo. La comodidad, en este caso a la hora de depurar posibles errores, es también la principal motivación a la hora de implementar el método *ToString*.

Por otra parte, si se quiere que esta implementación del tipo estado funcione adecuadamente dentro del DM, es imprescindible implementar los métodos *Equals* y *GetHashCode*, que facilitan y aceleran en gran medida el proceso de búsqueda llevado a cabo por el Resolutor. Ambos métodos, además, pueden auto-implementarse con la función de refactorización que ofrece C#.

4.2.2 Las Acciones y la Base de Conocimiento

Las acciones, como se vio en el **CAPÍTULO III**:

, representan el conjunto de reglas de interacción que han sido suministradas al DM. En la práctica, son el conjunto de transformaciones que se pueden aplicar a cada uno de los atributos del estado.

El usuario puede definir todas las acciones que considere oportunas, pero en este caso sí que se proporciona una estructura básica que se debe seguir: la interfaz *IAction*. Como se ha explicado previamente, esta interfaz es genérica, ya que recibe como parámetro el tipo estado. Parece evidente que, si no se aporta una definición de clase *State* cerrada, la interfaz se puede volver mucho más potente. Antes de ver un ejemplo, urge conocer los cuatro métodos que define la interfaz *IAction* y que deben implementarse en cada clase que implemente la interfaz:

- ***Apply***: este método aplica los cambios propios de la acción sobre el estado que recibe como parámetro, y devuelve el nuevo estado resultante.
- ***IsLegal***: este método determina si la acción se permite o no dentro del estado que recibe como parámetro, devolviendo un valor booleano.
- ***Coherence***: este método determina cuánto de prometedora es la acción dentro del estado que recibe como parámetro, devolviendo un valor numérico. Como ya se indicó en el **CAPÍTULO III**:
- , el término coherencia no hace referencia al uso académica de la palabra en el campo de la narrativa interactiva, aunque en cierto modo garantiza que la secuencia de acciones seleccionadas por el DM conserva la coherencia de la trayectoria narrativa propuesta como solución.
- ***GetActionType***: este método indica el tipo de la acción a través de un valor entero que, idealmente, se mapeará con tipo enumerado.

Originalmente, la interfaz incluía también el método *ToString*, pero se decidió delegar su implementación en cada una de las clases de tipo acción, flexibilizando así la implementación de una interfaz de por sí bastante rígida.

Al igual que se hizo con *State*, se decidió incluir todas las clases que implementan *IAction* en el archivo *Actions.cs* del directorio *AI* del módulo de juego.

Sin entrar en detalles que se verán en el subapartado dedicado al módulo de juego, más adelante se presenta un ejemplo de cómo funcionaría una implementación muy básica de la interfaz *IAction*. Suponiendo una clase *State* con dos atributos *x* e *y* que indiquen una posición, y un enumerado *ActionType* que indique los tipos de acción:

```
public class ActionMoveUp : IAction<State>
{
    public State Apply(State state)
    {
        State child = new State(state.x, state.y - 1);
        return child;
    }

    public bool IsLegal(State state)
    {
        return state.y - 1 >= 0;
    }

    public double Coherence(State state)
    {
        if (state.y == 0) return 0.0;
        else return 1.0;
    }

    public int GetActionType()
    {
        return (int)ActionType.MoveUp;
    }

    public override string ToString()
    {
        return "MOVE UP";
    }
}
```

Al implementar *IAction<State>*, y no *IAction<S>*, con un tipo estado genérico, se está forzando a la acción *ActionMoveUp* a utilizar nuestra implementación de *State*. Como su propio nombre indica, esta acción permite desplazarnos hacia arriba una unidad; esto es lo que sucede en el método *Apply*.

La acción sólo es legal si no se pasa a coordenadas negativas, y por tanto sólo es coherente si el valor de la *y* de *State* es distinto de 0; en otros contextos, la

diferenciación entre legalidad y coherencia puede ser mucho menos sutil, y se hablará de ello en el apartado dedicado específicamente al módulo de juego.

Antes de profundizar en el sistema central del DM, sería oportuno detenerse brevemente en la implementación de la Base de Conocimiento, perteneciente al subsistema de resolución de problemas, ya que su diseño se sustenta fuertemente en las decisiones que se tomaron a la hora de implementar las acciones.

KnowledgeBase no es más que una clase estática que alberga una colección de acciones. Esta clase contiene métodos para consultar la colección y para añadir nuevos elementos a la misma.

Se decidió utilizar una lista por comodidad, pero gracias a la implementación que hace C# de sus estructuras de datos, técnicamente se podría usar cualquier otro contenedor genérico.

Por ejemplo, podría haberse utilizado un diccionario, en el que las claves fuesen el propio enumerado que señala el tipo de la acción, o, incluso, una cadena de texto que expresase en un lenguaje informal la naturaleza de la acción. Sin embargo, dado que en esta implementación las acciones sólo son visibles al programador, no se consideró oportuno ofrecer ninguna de estas funcionalidades.

Otros descartes del diseño de las acciones fueron los parámetros. Generalizar las acciones habría llevado a necesitar parámetros para saber a qué personajes u objetos hacían referencia. Por ejemplo, las acciones *MoveUp*, *MoveDown*, *MoveLeft*, y *MoveRight* podrían haberse resumido en una acción *Move* que admitiese parámetros *x, y* para indicar la dirección del movimiento. Igualmente, una acción *Use* podría haber recibido dos parámetros { *actor, object* } que indicasen qué personaje iba a utilizar qué objeto.

La representación del conocimiento es todo un campo de estudio aparte, y ya es lo suficientemente compleja. El diseño de este tipo de acciones y su respectiva base de conocimiento no era el objetivo central del proyecto. En cualquier caso, apostar por esta falsa duplicidad de acciones llevó a un esfuerzo mayor de diseño, como se verá más adelante.

4.2.3 Ciclo de Vida del Sistema

Ahora que se han explicado los fundamentos de la implementación del sistema, se puede pasar a ver cómo interactúan todos los sistemas entre sí. La clase *DramaManager* es la que gestiona la comunicación entre todos los subsistemas, y la que expone la API del sistema al usuario.

Por tanto, cuenta con atributos que referencian a cada uno de los subsistemas: *Evaluator, Solver* y *Generator*; la clase *KnowledgeBase*, al ser estática, no cuenta con una referencia en *DramaManager*.

Como ya se explicó en el **CAPÍTULO III**:

, la clase *DramaManager* cuenta con diversos valores numéricos configurables para regular distintos atributos de los subsistemas: *CoherenceLevel*, *CompletenessLevel*, *DepthControl* y *Algorithm*. Una vez más, se decidió hacer propiedades de todos estos atributos, para facilitar su acceso y modificación. Todos cuentan con los valores por defecto que se han utilizado en este experimento. Cuando se crea una instancia de la clase *DramaManager*, además de indicar el tipo estado a utilizar se debe:

1. Suministrar a la Base de Conocimiento el listado inicial de acciones.
2. Suministrar todos los parámetros de configuración de los subsistemas.
3. Suministrar todas las funciones que utilizarán los subsistemas.

DramaManager tiene referencias a todas estas funciones: evaluación, completitud, coste, heurística, objetivo y generación. Estos atributos, que también funcionan como propiedades, son posibles gracias a los objetos delegados de C#, una suerte de punteros a funciones. En este proyecto, se han utilizado los prototipos *Func* y *Action* a conveniencia:

a. *Func<T, U>*: encapsula una referencia a una función que devuelve un objeto de tipo T y recibe como parámetro un objeto de tipo U (la lista de parámetros de entrada puede extenderse hasta 16 tipos distintos). Este prototipo es el que utilizan todas las funciones excepto la de generación.

b. *Action<T, U>*: encapsula una referencia a una función que no devuelve nada (void) y recibe como parámetro un objeto de tipo U (la lista de parámetros

de entrada puede extenderse hasta 16 tipos distintos). Este es el prototipo que utiliza la función de generación.

Pero, como ya se ha dicho, estos atributos no son más que referencias a funciones. La definición e implementación de las funciones es responsabilidad del usuario.

Al igual que pasaba con la implementación del tipo estado y las acciones, en este proyecto se decidió aislar esta tarea en el archivo *Functions.cs* del subdirectorio *AI* del módulo de juego. Se decidió también mantener la nomenclatura de las funciones (por ejemplo, la función de coste se llama *CostFunction*) y hacerlas estáticas para evitar contratiempos. Así, sería una inicialización típica del DM:

```
DramaManager = new DramaManager<State>(
      Functions.EvaluationFunction,
      Functions.CompletenessFunction,
      Functions.GoalFunction,
      Functions.CostFunction,
      Functions.HeuristicFunction,
      Functions.GenerationFunction
   );
```

Esta construcción se realiza desde la clase *WorldManager* del módulo de juego. Como se puede observar, omite todos los parámetros de configuración de los subsistemas, ya que utiliza la configuración por defecto: *CoherenceLevel = 0.5*, *CompletenessLevel = 0.9*, *DepthControl = 5000* y *Algorithm = A**.

Con el DM configurado, se puede realizar la llamada al método *Init*, que espera recibir una colección de estados para incluirla en la Agenda Narrativa del Evaluador. Internamente, la clase *Evaluator* gestiona esta colección de estados como una cola. Se pueden añadir uno o más estados a la cola, reemplazarla por completo, comprobar si está vacía y actualizar los estados actual y objetivo.

La clase *DramaManager* cuenta con un método *Update* encargado de desencadenar toda la cascada de llamadas a los subsistemas. El usuario tiene la responsabilidad de llamar al método *Update* cuando y como crea oportuno. Además, la clase *WorldManager*, con el soporte de Unity, se encarga de crear una corrutina en la que se ejecuta este método *Update*, liberando al hilo principal de esta carga y no bloqueando el progreso del jugador. La primera comprobación que se hace en el

método *Update* del DM es si la trama ha concluido: de ser así, no tendría sentido actualizar el DM, por lo que el método acabaría aquí. El método *IsStoryOver* de la clase *Evaluator* permite realizar esta comprobación: si la cola que representa la Agenda Narrativa está vacía, es que la historia ha acabado.

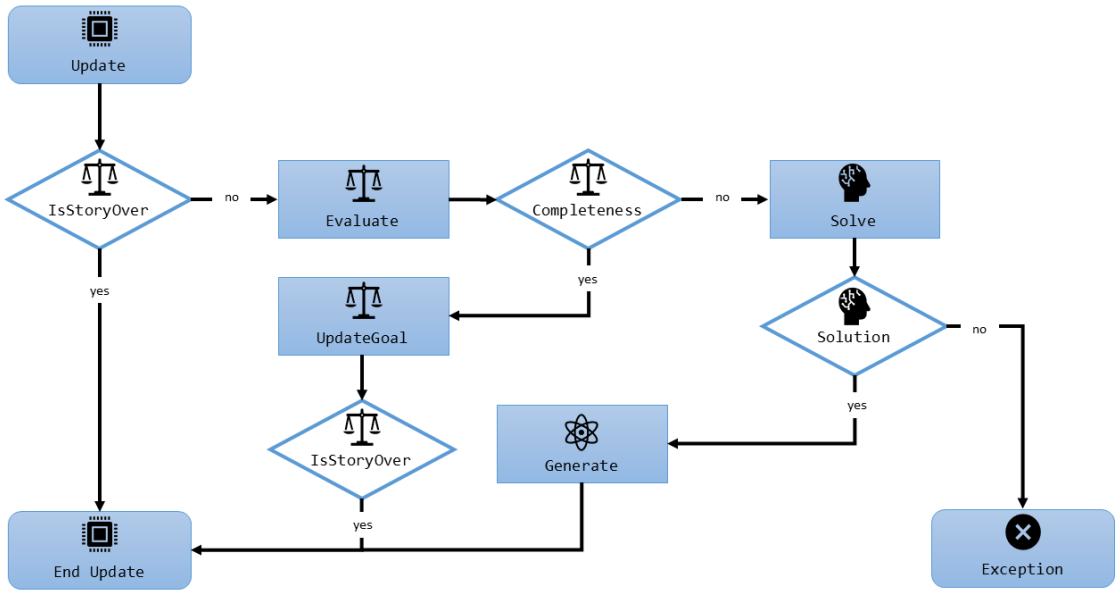

Figura 14: Diagrama de flujo del funcionamiento del método Update de la clase DramaManager.

Si la Agenda Narrativa no está vacía y, por tanto, se dispone de un estado objetivo, el Evaluador llama a su función de evaluación para plasmar el estado actual del juego en su propiedad *CurrentState*. Acto seguido, se comprueba si dicho estado es equiparable al estado objetivo con la función de completitud. De ser así, el Evaluador debe actualizar su estado objetivo: se desencola un elemento de la Agenda Narrativa; si la cola queda vacía, se habrá terminado la historia.

En cualquier otro caso, se dispondrá de un nuevo estado objetivo y de un estado actual, que hará las veces de estado inicial para el Resolutor y su método *Solve*. Si el Resolutor no encuentra una solución, bien porque la heurística no esté completa o porque se exceda el límite de expansiones permitido, se lanzará una excepción. Es responsabilidad del usuario capturarla y tratarla como crea oportuno, aunque lo más probable es que se produzca por un error en el diseño del estado o las acciones, llevando a una heurística incompleta que hace fracasar la búsqueda.

En cambio, si el Resolutor encuentra una solución, esta se envía como parámetro al método *Generate* del Generador, que llamará a la función de generación correspondiente. Si bien las clases *Evaluator* y *Generator* no cuentan como mucha más complejidad, sí que merece la pena detenerse en la clase *Solver* y el subsistema de resolución de problemas para ver con más detalle el proceso de búsqueda.

4.2.4 Estructuras de Datos y Algoritmos de Búsqueda

Tal y como se ha mencionado ya en varias ocasiones, el sistema de resolución de problemas es el corazón de nuestro agente inteligente. A nivel técnico, es el subsistema encargado de realizar el proceso de búsqueda en el espacio de estados. La clase *Solver* es la que implementa esta funcionalidad, en colaboración con *KnowledgeBase*, y ambas se encuentran en el directorio *Core*.

Dentro del directorio *Core*, se encuentra el subdirectorio *Search*. Aquí se ha decidido incluir todos los archivos de código fuente que son utilizados específicamente por el proceso de búsqueda. Como ya se ha explicado, el espacio de estados se puede entender como un grafo en el que los vértices son estados y las aristas acciones; pero es necesaria una estructura más agnóstica: los nodos.

Los nodos son la estructura de datos que utiliza el Resolutor durante el proceso de búsqueda para representar los vértices del grafo generado por el espacio de estados. Así, una instancia de la clase *Node* cuenta con estos atributos:

- ***State***: los nodos encapsulan estados, y por tanto deben tener una referencia al estado que representan. A través del estado y sus métodos *Equals* y *GetHashCode* se pueden comparar nodos y almacenarlos y buscarlos en contendores estándar como listas o tablas hash. Nótese que, en la implementación de este proyecto, los estados y, por tanto, los nodos son iguales cuando todos sus atributos tienen el mismo valor.
- ***G***: esta propiedad es un valor numérico que indica el coste del camino desde el nodo inicial hasta el nodo actual.
- ***H***: esta propiedad es un valor numérico que indica el coste estimado para llegar desde el nodo actual hasta un nodo objetivo.

- ***F***: esta propiedad es un valor numérico que indica el coste total del nodo, es decir, la suma de los valores de *G* y *H*.
- ***Parent***: todos los nodos guardan una referencia a su nodo padre, es decir, al nodo que les dio origen cuando fue expandido; el nodo raíz no tiene nodo padre, por lo que la referencia es *null*. Esta referencia es necesaria para trazar el camino solución desde el nodo objetivo hasta el nodo inicial.

- ***Action***: todos los nodos guardan una referencia a la acción que les dio origen, es decir, la acción que fue aplicada a su nodo padre durante la expansión; el nodo raíz no tiene acción generadora, por lo que la referencia es *null*. Esta referencia es necesaria para trazar un camino de acciones como parte complementaria de la solución.

La clase *Node* cuenta con dos constructoras: una pensada específicamente para el nodo raíz, que sólo toma como parámetro un estado, y otra pensada para el resto de los nodos, que además del estado toman como parámetros el nodo padre y la acción generadora.

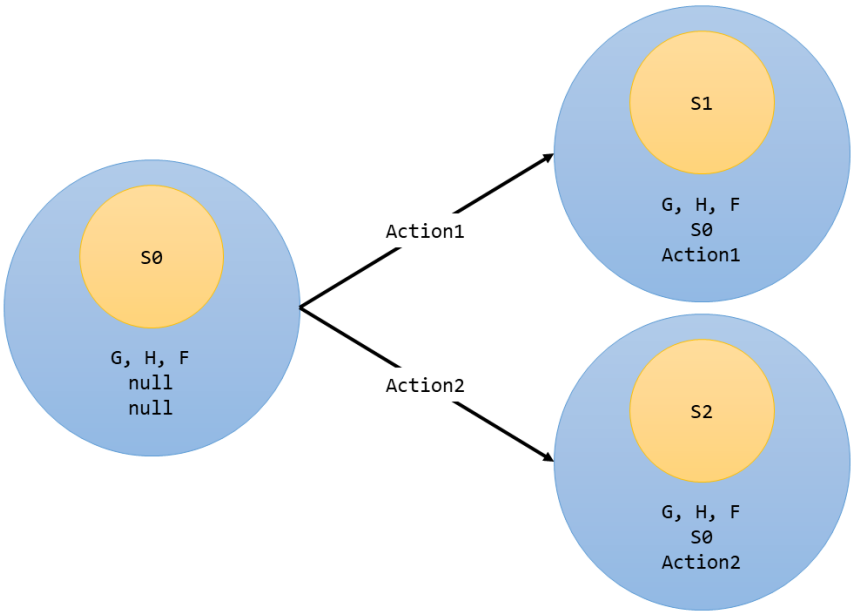

Figura 15: Ejemplo de expansión del nodo S0 mediante las acciones Action1 y Action2, dando como hijos el nodo S1 y el nodo S2.

La forma en la que los nodos se exploran y se expanden es lo que marca la diferencia a la hora de emplear un algoritmo u otro en la búsqueda. En nuestra implementación, se ha definido una estructura de datos denominada *Frontier*.

Como ya se dijo en el **CAPÍTULO III**:

, se decidió implementar tres algoritmos clásicos de búsqueda: DFS, BFS y A*. Aunque el proceso de búsqueda es el mismo para los tres, la forma en la que se gestiona la frontera difiera notablemente de unos a otros. Así pues, la clase *Frontier* se implementa como una clase abstracta, de la que heredan las clases *DFSFrontier, BFSFrontier* y *AStarFrontier*.

Estas clases implementan sus propias operaciones para añadir y eliminar elementos de la frontera, así como para consultar el número de estos; la principal diferencia radica en la estructura de lista abierta que maneja cada subclase:

- ***DFSFrontier***, al utilizar el algoritmo DFS, utiliza una pila.
- ***BFSFrontier***, al utilizar el algoritmo BFS, utiliza una cola.
- ***AStarFrontier***, al utilizar el algoritmo A*, utiliza una cola de prioridad.

Todas las fronteras utilizan, además, una tabla hash como lista cerrada. El tipo de frontera que utiliza el Resolutor durante el proceso de búsqueda va ligado, lógicamente, al tipo enumerado *SearchAlgorithm* que recibe como parámetro en la llamada a su función principal, *Solve*.

Esta, además, recibe como parámetros el estado final y el estado objetivo, así como las funciones de coste, heurística y objetivo. El proceso de búsqueda, que se desarrolla de forma privada en el método *Search*, se inicia tras crear el nodo raíz con el estado inicial e introducirlo en la frontera escogida:

1. Mientras la frontera no esté vacía, se saca un nodo con la función *Next*.
2. Si el nodo cumple la función objetivo, se devuelve como solución.
3. Si no, el nodo se expande mediante una llamada al método *Expand*.
4. Se elabora una lista de acciones legales y coherentes para el nodo.
5. Se aplica la lista de acciones elegidas al nodo, generando nodos hijo.

6. Se actualizan los costes con las funciones de coste y heurística.
7. Se añaden todos los nuevos nodos a la frontera.
8. Si se ha alcanzado el máximo de expansiones, se devuelve *null*.

Cada vez que se concluye una búsqueda con éxito, el método *Solve* rellena la pila de estados y la pila de acciones con los datos obtenidos del nodo solución. Estas dos pilas conforman la solución que se envía posteriormente al Generador.

Y con esto, concluye el repaso a la implementación del motor de Inteligencia Artificial. En el próximo apartado se explorará la implementación del módulo de juego, y se comprobará de primera mano cómo se comunica con el DM.

4.3 El Módulo de Juego

Para comprobar que el motor de Inteligencia Artificial implementado producía los resultados esperados, se desarrollaron dos videojuegos: *Emerald Hunt* y *Tell No Tales*. Ambos sirvieron como prototipo para comprobar el funcionamiento del DM, aunque *Tell No Tales* es mucho más complejo y, por tanto, fue el videojuego que se utilizó para el experimento; no obstante, resulta de interés diseccionar *Emerald Hunt* para explicar el funcionamiento básico de este motor.

4.3.1 Implementación de *Emerald Hunt*

Emerald Hunt es el primer juego que se implementó para poder probar la funcionalidad del DM. Hace uso del sistema de una forma sencilla que permite comprobar que funciona correctamente de manera controlada.

4.3.1.1 Descripción e implementación del videojuego

El juego consta de una escena con una cuadricula 3x3 y un personaje controlado por el jugador que puede moverse en cuatro direcciones (arriba, abajo, izquierda y derecha) dentro de los límites de la cuadrícula. En la posición (2, 2) de dicha cuadrícula se encuentra una esmeralda, que puede ser recogida por el jugador si este la toca. Cuando esto ocurre, la esmeralda se destruye, evitando que se pueda recoger más de una vez. Para moverse, el usuario hace uso de las teclas 'W', 'A',

'S' y 'D'. Con estos elementos, ya se puede definir una historia dentro de este juego:

- **Estado Inicial:** El jugador se encuentra en la posición (0, 0) y no posee ninguna esmeralda.
- **Estado Final:** El jugador se encuentra en la posición (1, 1) y posee una esmeralda.

4.3.1.2 Integración del Drama Manager

Para hacer uso del DM, se deben definir también *State*, *Actions* y *Functions* como se ha explicado anteriormente. El Estado (*State*) del juego se representa con la posición (x, y) del jugador y el numero 'e' de esmeraldas recogidas. En cuanto a las acciones (*Actions*), se dispone de las siguientes:

a. ***MoveUp*:** Al aplicarse sobre un estado, se generará un nuevo estado en el que la posición 'y' del jugador se vea disminuida por una unidad, moviéndole así a la casilla superior. En caso de que en dicha casilla haya una esmeralda, se aumentará el número 'e' en una unidad. Esta acción solo es legal si la posición del nuevo estado a generar no se sale de los límites del tablero, y es coherente cuando la posición 'y' del jugador es mayor que 0.
b. ***MoveDown*:** Al aplicarse sobre un estado, se generará un nuevo estado en el que la posición 'y' del jugador se vea aumentada por una unidad, moviéndole así a la casilla inferior. En caso de que en dicha casilla haya una esmeralda, se aumentará el número 'e' en una unidad. Esta acción solo es legal si la posición del nuevo estado a generar no se sale de los límites del tablero, y es coherente cuando la posición 'y' del jugador es menor que el número de filas del tablero menos una unidad (Rows – 1).
c. ***MoveLeft*:** Al aplicarse sobre un estado, se generará un nuevo estado en el que la posición 'x' del jugador se vea disminuida por una unidad, moviéndole así a la casilla a su izquierda. En caso de que en dicha casilla haya una esmeralda, se aumentará el número 'e' en una unidad. Esta acción solo es legal si la posición del nuevo estado a generar no se sale de los límites del tablero, y es coherente cuando la posición 'x' del jugador es mayor que 0.

d. ***MoveRight*:** Al aplicarse sobre un estado, se generará un nuevo estado en el que la posición 'x' del jugador se vea aumentada por una unidad, moviéndole así a la casilla a su derecha. En caso de que en dicha casilla haya una esmeralda, se aumentará el número 'e' en una unidad. Esta acción solo es legal si la posición del nuevo estado a generar no se sale de los límites del tablero, y es coherente cuando la posición 'x' del jugador es menor que el número de columnas del tablero menos una unidad (Cols – 1).

A modo de ejemplo, si el jugador se encontrase en un estado `{x = 1, y = 2, e = 0}` y se aplicase la acción *ActionMoveRight* anteriormente descrita, el nuevo estado sería `{x = 2, y = 2, e = 1}`.

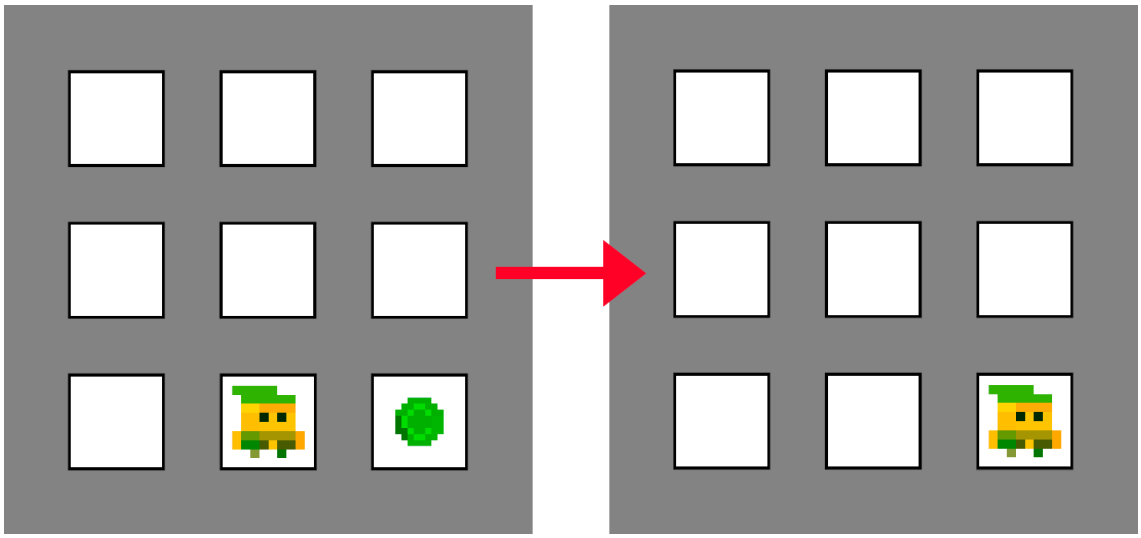

Figura 16: Al aplicar la acción *ActionMoveRight* sobre el estado anteriormente descrito, la variable 'e' se vería aumentada porque la esmeralda se encuentra en la posición (2, 2).

Por último, hace falta definir las funciones (*Functions*), necesarias para el funcionamiento del DM. Estas son la función de evaluación (*EvaluationFunction*), la función de completitud (*CompletenessFunction*), la función objetivo (*GoalFunction*), la función de coste (*CostFunction*), la función heurística (*HeuristicFunction*) y la función de generación (*GenerationFunction*):

- ***EvaluationFunction*:** Devuelve el estado actual, que se genera comprobando la posición del jugador y el número de esmeraldas que tiene, información disponible en la lógica del juego.

- ***CompletenessFunction*:** Devuelve 1 si los dos estados proporcionados son equivalentes, es decir, si todos sus campos tienen el mismo valor.
- ***GoalFunction*:** Devuelve *true* si el estado actual es igual al objetivo.
- ***CostFunction*:** Devuelve 1. No se ha tenido en cuenta para este videojuego. Implica que pasar de una casilla a otra tiene coste 1 en cualquier caso.
- ***HeuristicFunction*:** Se han considerado dos situaciones: la primera, en la que el jugador no dispone aún de ninguna esmeralda; y la segunda, en la que ya se dispone de al menos una esmeralda. Una vez hecha esa diferenciación, se calcula la distancia Manhattan al siguiente punto de interés: (1, 1) en el primer caso, y (2, 2) en el segundo. Así, se pueden estimar cuántos pasos quedan para alcanzar un estado final válido.
- ***GenerationFunction*:** Imprime por consola lo que el jugador debe hacer a continuación en el formato "Try doing this: *[Acción traducida a string]*".

Una vez está todo definido, se llama al DM cada vez que el jugador pulsa una de las teclas de movimiento ('W', 'A', 'S' o 'D') para que genere la historia correspondiente.

Por poner un ejemplo, si el estado actual es `{x = 0, y = 0, e = 0}`, se podría trazar el camino más corto que el jugador debe seguir para llegar al estado final, según la solución proporcionada por el DM, de la siguiente forma:

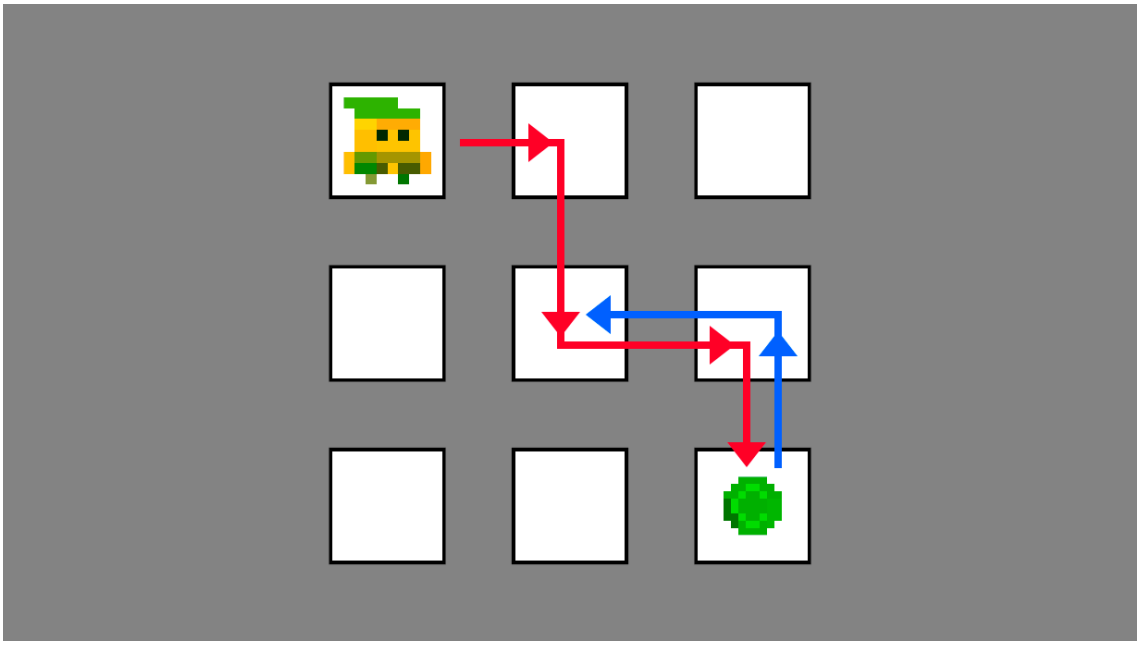

Figura 17: Camino a recorrer por el jugador en la posición (0, 0) para llegar al final. En rojo, camino hasta que se recoge la esmeralda; en azul, camino desde que se recoge la esmeralda.

Por lo tanto, mediante la función de generación, se escribiría el mensaje "Try doing this: MOVE RIGHT", dado a que la siguiente acción que el jugador debe aplicar es moverse a la casilla de su derecha. Sin embargo, el jugador puede ignorar la pista y, por ejemplo, moverse hacia la casilla inferior. Por tanto, se realizaría de nuevo la búsqueda y se trazaría un nuevo camino.

Nótese que la búsqueda no encuentra el camino más corto en términos espaciales, necesariamente. Esta particularidad es más apreciable si se aumenta el número de esmeraldas a recoger antes de llegar al centro.

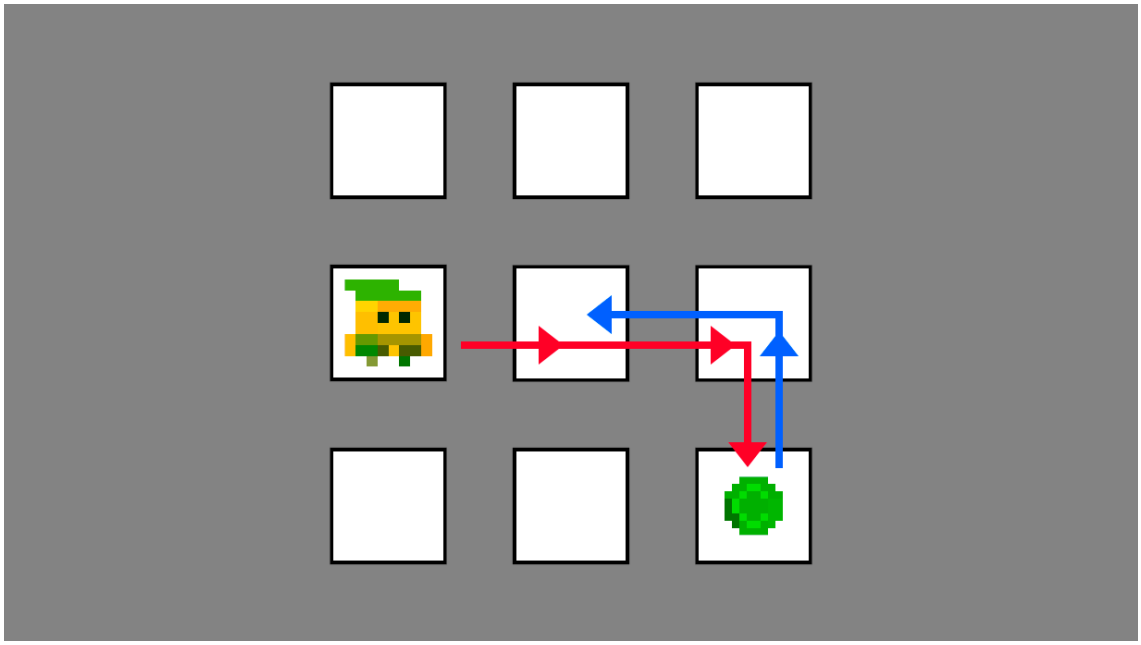

Figura 18: Camino a recorrer por el jugador en la posición (0, 1) para llegar al final. En rojo, camino hasta que se recoge la esmeralda; en azul, camino desde que se recoge la esmeralda.

Una vez terminado de implementar este juego y haber obtenido los resultados esperados, se puede concluir el correcto funcionamiento del DM y proceder a implementar un juego más complejo, *Tell No Tales*. Si bien *Emerald Hunt* fue implementado sin el apoyo de herramientas externas, para agilizar el desarrollo de *Tell No Tales* se hizo uso de un *framework* llamado TopDown Engine.

4.3.2 El *Framework*: TopDown Engine

TopDown Engine es un *framework* que contiene todas las herramientas necesarias para desarrollar un videojuego en perspectiva *Top-Down*, también conocida como vista de pájaro o vista elevada. Desarrollado por *More Mountains* y lanzado en 2018, este *framework* se encuentra disponible en la Asset Store[1] de Unity.

TopDown Engine consta de innumerables herramientas útiles, de las cuales solo se ha hecho uso de unas pocas. En primer lugar, se ha hecho uso del movimiento del personaje y el sistema de input del *framework*, el cual ha permitido tener a un personaje completo y funcional en poco tiempo.

[1] https://assetstore.unity.com/packages/templates/systems/topdown-engine-89636

Para construir el entorno del juego se han utilizado los *sprites* ofrecidos por TopDown Engine, tanto el *tileset* (el conjunto de casillas que se usan para dibujar el entorno: suelo, pared, agua, etcétera), como el personaje principal.

También se han tenido que diseñar *sprites* nuevos para diferentes elementos del juego que no estaban representados en TopDown Engine, como el resto de los personajes (modificando el principal), elementos del escenario y los objetos del inventario.

Figura 19: Escena de demostración incluida con el paquete básico de TopDown Engine.

Además, también se ha hecho uso de su amplio sistema de interfaz de usuario. Dentro de este apartado, se puede destacar el sistema de diálogos integrados, que ha permitido la inclusión y modificación fácil de estos, pudiendo centrar el foco de trabajo así en la elaboración del guion más que en la implementación.

Posiblemente, el elemento más importante que ha ofrecido TopDown Engine para acelerar el desarrollo del videojuego ha sido el sistema de inventario. Este sistema es muy maleable, por lo que se ha podido implementar un inventario funcional con las cualidades deseadas. El sistema de inventarios de TopDown Engine se aprovecha de las funcionalidades de los *ScriptableObjects* de Unity, por lo que su integración en nuestro juego ha sido bastante sencilla.

Por último, también se ha hecho uso de efectos y *shaders* que nos brinda el sistema, y que confieren al videojuego un apartado visual más interesante y reactivo a las acciones del jugador.

4.3.3 Implementación de *Tell No Tales*

Tras terminar la implementación de *Emerald Hunt*, el objetivo era la implementación de un juego más complejo para demostrar las capacidades del DM, *Tell No Tales*.

4.3.3.1 Descripción del videojuego

En *Tell No Tales* el jugador toma el papel de un pirata, que ha quedado encerrado en una mazmorra. Tras escapar, el jugador realiza una sucesión de tareas para eventualmente conseguir llegar al final de la historia. El jugador puede moverse en ocho direcciones, haciendo uso de las teclas de movimiento ('W', 'A', 'S' y 'D') y puede interactuar con el entorno pulsando la tecla 'E' o la barra espaciadora. Además, el jugador posee un inventario, el cual puede abrirse pulsando la tecla 'I'. Por último, cuando el jugador posea la pala en su inventario, este puede hacer uso de la tecla 'Q' para cavar, evitando así tener que abrir el inventario cada vez que se quiera usar este objeto.

En resumen, la Agenda Narrativa suministrada al DM se podría resumir con este breve argumento: el jugador está encerrado en una mazmorra en la isla inicial, cansado. En la mazmorra hay un amigo, que le otorga una nota que solo puede ser leída con luz (haciendo uso de una cerilla). El jugador consigue escapar y, una vez descansado, coge el barco para llegar a la primera isla, donde hay un posadero que le otorga una nota. Una vez se tiene dicha nota, el jugador debe ir a la segunda isla, donde un pirata le otorga una nueva nota.

El jugador debe dirigirse después a la isla inicial y agitar una palmera para recibir una nota más. Como las notas indican la existencia de un tesoro en la segunda isla, el jugador necesita una pala, la cual se consigue hablando con un personaje de la isla inicial situado cerca de la salida de la mazmorra. Con la pala, el jugador debe dirigirse ahora a la segunda isla, al norte, y cavar siete veces en la zona establecida para encontrar el tesoro.

Dicho tesoro es una última nota y una llave, con la cual el jugador puede ir a la tercera y última isla y entrar en una casa que está cerrada. Una vez dentro, la puerta vuelve a cerrarse, dejando solo la posibilidad de que el jugador duerma en la cama central de la casa, terminando ahí el videojuego.

Las notas mencionadas sirven para explicar la trama al jugador y deben conseguirse en el orden establecido para completar la historia. Por ejemplo, si el jugador hablase primero con el pirata de la segunda isla en vez de con el posadero de la primera, este no le otorgaría la segunda nota, pues no es el objetivo actual.

4.3.3.2 Implementación del videojuego

Antes de profundizar en el uso del DM, es necesario explicar los diferentes sistemas que permiten sustentan el videojuego. En primer lugar, al tratarse de un videojuego conversacional, hay que destacar los dos sistemas de diálogos.

Figura 20: Sistema de diálogos para la narración y voz del protagonista.

El primero es un sistema que se usa como narrador y voz del protagonista, que hace aparecer un cuadro de texto en la parte inferior de la pantalla. Cada vez que el jugador interactúa con el mundo del videojuego se hace uso de este sistema. Este sistema de diálogos ha sido implementado de cero para este proyecto, y por tanto cuenta con su propia definición de ensamblador y espacio de nombres: *Dialogues*.

Figura 21: Sistema de diálogos para las conversaciones con personajes del juego.

El segundo sistema de diálogos usado es el propio de *TopDown Engine*; se usa para las conversaciones que el jugador puede tener con diferentes personajes. Cada personaje interactuable cuenta con un componente *DialogueZone* que contiene información sobre el diálogo. Este se construye como una lista de *Dialogue Elements* o Elementos de Diálogo, teniendo cada elemento información sobre una cadena para ser una de las líneas del diálogo.

Los diálogos se pueden cambiar tanto desde código como desde el propio editor de Unity, simplemente definiendo el número de líneas que se desean y escribiéndolas. Una vez dentro de la Zona de Diálogos, el jugador debe pulsar la tecla definida para interactuar y se activará el diálogo del personaje. Este se muestra línea a línea, por lo que el jugador debe seguir presionando la tecla de interactuar para seguir avanzando hasta terminar el diálogo.

Por otra parte, para gestionar los diferentes objetos que se obtienen durante el juego, se hace uso del sistema de inventario de *TopDown Engine*, que consta de un inventario muy maleable. Se decidió que el jugador puede llevar hasta 12 objetos en total, los cuales solo puede "Usar". Algunos objetos, al ser usados, se eliminan del inventario.

Figura 22: Sistema de inventario de TopDown Engine, con tres objetos recogidos.

Los objetos como tal están implementados como *Scriptable Objects* de Unity. Estos son una clase que permite al usuario almacenar grandes cantidades de datos compartidos independientes de instancias de script. *TopDown Engine* contiene la implementación de un objeto *BaseItem*, del cual se debe heredar en la implementación de cada objeto, para así poder usarlos correctamente con el sistema de inventario del *framework*. Estos objetos permiten la implementación de diversas funciones, pero en este caso solo se sobrescribe lo que ocurre al usarlos en la función *Use*. Por tanto, los objetos que han sido creados y sus funcionalidades son los siguientes:

- **Notas:** Todas las notas del videojuego son un *Scriptable Object* diferente, pero funcionan de la misma forma. Al ser usadas, las notas hacen uso del sistema de diálogos de narrador para mostrar por pantalla el texto deseado.
- **Nota del amigo:** La primera nota obtenible, otorgada por el amigo de la mazmorra, tiene un comportamiento especial. La mazmorra está demasiado oscura para leer la nota, así que el jugador debe tener luz para poder ver lo que dice. Si no se tiene luz, el sistema de diálogos informará de que está demasiado oscuro para leer la nota. Si el jugador hace uso de una cerilla, consigue luz y puede leerla. El jugador pierde la nota al salir de la mazmorra.

- **Cerilla:** El uso de la cerilla depende del contexto en el que se use. Si se usa con la nota del amigo en el inventario, el jugador consigue tener luz y puede leer la nota. Si no se tiene dicha nota y se usa cerca del barril situado a la salida de la mazmorra, el barril prenderá fuego y acabará explotando, permitiendo escapar de la mazmorra. Por último, si se usa sin cumplir ninguna de las dos condiciones anteriores, la cerilla queda desperdiciada. Tras su uso, la cerilla se elimina del inventario.
- **Ron:** Al usar la botella de ron, el jugador cambia su estado a descansado. Se pueden llevar varias botellas de ron. Al hacer uso de una, se elimina del inventario.
- **Pala:** La pala permite excavar justo en el punto en el que el jugador se encuentra. Al usarse, se aumenta el contador adecuado en el componente *PlayerStatus* dentro del struct *digCont*, que tiene información sobre cuántas veces se ha cavado en cada zona del juego. El struct *digCont* contiene el número total de excavaciones, el número de veces que se han hecho en la zona correcta y el número de veces que se han hecho en zonas incorrectas en cada isla. Cuando el contador de veces que se ha hecho en zonas correctas excede siete, se otorga al jugador de la última nota y la llave para la casa final. Además, si se está haciendo uso del DM y su función de generación, la pala puede activar la aparición de un personaje que indica al jugador dónde debe cavar si este ha fallado muchas veces, pero esto se explicará más adelante. El jugador puede hacer uso de la pala desde el inventario, como el resto de los objetos, o presionando la letra 'Q' (una vez se haya obtenido, lógicamente).
- **Llave:** Usarla no tiene ningún efecto, pero una vez se tiene en el inventario el jugador podrá desbloquear la puerta de la casa final.

Debido a que no todo el resto de los comportamientos que permiten el correcto funcionamiento del videojuego son de interés para el estudio, solo se destacarán los siguientes:

- ***BedBehaviour*:** Permite dormir al jugador, cambiando su estado de cansado a descansado.
- ***RumGiverBehavior*:** Se trata de unos personajes especiales que otorgan ron al jugador cuando este está cansado. Se les reconoce por su color verde y por llevar una botella de ron encima.
- ***MapSelectorBehaviour*:** Permite viajar entre islas cuando se está descansado. Al acercarse a un barco e interactuar, se abre un panel en el que el jugador deberá pulsar el icono de la isla deseada para viajar a ella. Entonces, el jugador será transportado al muelle de la nueva isla y se cansará. La variable *CurrentIsland* se actualiza acorde a la isla destino elegida. Si el jugador interactúa con el barco estando cansado se usa el sistema de diálogos de narrador para avisar de que no es posible viajar cansado. Además, cada isla en el selector posee una descripción que puede cambiarse entre "segura" o "peligrosa", como se verá más adelante en la función de generación. Por ejemplo, para la primera isla, la descripción segura lee "El mar anda calmado en esa ruta y las gentes son maravillosas. Una gran parada para tomarse un descanso y un buen ron en la posada" y la peligrosa "Las mareas no son favorables y el viaje podría ser demasiado duro para los tripulantes de mi barco, no creo que sea una buena idea viajar hacia ella ahora mismo".
- ***NPCDialoguesManager*:** Sistema encargado de poseer información los diálogos de los personajes durante el juego y cambiarlos cuando se desee. Desde el editor, se definen distintos *Dialogue Packs* o Paquetes de Diálogos, que están constituidos por una cadena que sirve como ID y una lista de diálogos, cada uno correspondiendo a un personaje. Se debe definir un *Dialogue Pack* por defecto y una lista de *State Dialogue Packs*, distintos diálogos usados en la función de generación explicada más adelante. Tiene una función *SetDialoguePack* que toma como parámetro la ID del paquete deseado y se cambian los diálogos de los personajes a los disponibles en dicho paquete. En caso de que no exista un paquete con la ID indicada, se avisa por consola al usuario y se cambia al paquete por defecto.

- ***WorldManager*:** Se trata de una entidad utilizada para almacenar la instancia del Drama Manager. Se encarga de inicializarlo y actualizarlo cuando se desee para hacer la búsqueda. Posee funciones útiles para los sistemas de evaluación y generación, que se llamarán desde el DM para construir los estados y generar contenido acorde a la solución de la búsqueda del sistema resolutor. Se explicarán estas funciones más adelante. El *WorldManager* toma el papel de *Game Manager* visto en el **CAPÍTULO III**:
- . Su nombre se debe a la existencia conflictiva de otro *Game Manager* en la escena, usado por *TopDown Engine*.

Además, también se ha hecho uso del sistema de cámaras *CineMachine* de Unity, una serie de herramientas para cámaras dinámicas, inteligentes y sin código. El uso de este tipo de cámara provee al juego de un aspecto más interesante. Se ha hecho uso también del sistema de interfaz de usuario de Unity, con el cual se puede añadir una entidad con el componente *Canvas* a la escena, la cual permite crear un UI de forma sencilla, lo que acelera mucho el proceso en implementaciones no interesantes para este estudio, como los menús.

4.3.3.3 Integración del Drama Manager

Con todo esto ya implementado, se debe definir *State*, *Actions* y *Functions* al igual que en *Emerald Hunt*, para integrar y hacer uso del DM. Un estado (*State*) de este videojuego viene representado por las siguientes variables:

- ***NumOfNotes*:** El número de notas conseguidas por el jugador.
- ***NumOfDrinks*:** El número de botellas de ron que tiene el jugador.
- ***NumOfDigs*:** El número de veces que se ha cavado en el lugar adecuado.
- ***PlayerHasShovel*:** Booleano que indica si se ha obtenido la pala o no.
- ***PlayerHasHomeKey*:** Booleano que indica si se ha obtenido la llave o no.
- ***PlayerIsTired*:** Booleano que indica si el jugador está cansado o no.
- ***CurrentIsland*:** Un enumerado que contiene información sobre la isla en la que el jugador se encuentra: inicial (*ISLAND_0*), primera (*ISLAND_1*), segunda (*ISLAND_2*) o tercera (*ISLAND_3*).

- ***CurrentPlace*:** Un enumerado que contiene información sobre el lugar de importancia en el que el jugador se encuentra: ninguno (*OTHER*), la posada (*INN*), la guardia pirata (*HIDEOUT*), el lugar del tesoro (*TREASURE_SPOT*), la casa final (*HOME*) o la mazmorra (*DUNGEON*).

Figura 23: El jugador con CurrentIsland = ISLAND_2 y CurrentPlace = TREASURE_SPOT

Conociendo la estructura de los estados, se puede pasar a definir los *plot points* que han sido utilizados como Agenda Narrativa en este videojuego:

1. El jugador obtiene la primera nota en la posada de la primera isla.
   {NumOfNotes = 1, NumOfDrinks = 0, NumOfDigs = 0, PlayerHasShovel = false, PlayerHasHomeKey = false, PlayerIsTired = false, CurrentIsland = ISLAND_1, CurrentPlace = INN}

2. El jugador obtiene la segunda nota en la guarida pirata de la segunda isla.
   {NumOfNotes = 2, NumOfDrinks = 0, NumOfDigs = 0, PlayerHasShovel = false, PlayerHasHomeKey = false, PlayerIsTired = false, CurrentIsland = ISLAND_2, CurrentPlace = HIDEOUT}

3. El jugador obtiene la tercera nota en la palmera de la isla inicial (0).
   {NumOfNotes = 3, NumOfDrinks = 0, NumOfDigs = 0, PlayerHasShovel = false, PlayerHasHomeKey = false, PlayerIsTired = false, CurrentIsland = ISLAND_0, CurrentPlace = OTHER}

4. El jugador obtiene la pala del hombre de la isla inicial (0).
   {NumOfNotes = 3, NumOfDrinks = 0, NumOfDigs = 0, PlayerHasShovel = true, PlayerHasHomeKey = false, PlayerIsTired = false, CurrentIsland = ISLAND_0, CurrentPlace = OTHER}

5. El jugador obtiene la cuarta nota y la llave al cavar siete veces en el lugar del tesoro de la segunda isla.
   {NumOfNotes = 4, NumOfDrinks = 0, NumOfDigs = 0, PlayerHasShovel = true, PlayerHasHomeKey = true, PlayerIsTired = false, CurrentIsland = ISLAND_2, CurrentPlace = TREASURE_SPOT}

6. El jugador entra en la casa final.
   {NumOfNotes = 4, NumOfDrinks = 0, NumOfDigs = 0, PlayerHasShovel = true, PlayerHasHomeKey = true, PlayerIsTired = false, CurrentIsland = ISLAND_3, CurrentPlace = HOME}

Adicionalmente, además de la agenda narrativa, se suministrará al DM el estado inicial descrito anteriormente, en el que el jugador se encuentra en la mazmorra de la isla inicial, cansado:

{NumOfNotes = 0, NumOfDrinks = 0, NumOfDigs = 0, PlayerHasShovel = false, PlayerHasHomeKey = false, PlayerIsTired = true, CurrentIsland = ISLAND_0, CurrentPlace = DUNGEON}

Por último, se debe suministrar a la base de conocimiento del DM las acciones (*Actions*) aplicables a los estados. Para *Tell No Tales* se han diseñado 14 acciones:

- ***Rest*:** Al aplicarse en un estado, devuelve uno nuevo en el que el booleano *PlayerIsTired* es falso. Esta acción solo es coherente cuando el jugador está cansado y solo es legal siempre y cuando se posean una o más botellas de ron o se encuentre en una isla donde haya camas para descansar.
- ***GetDrink*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *NumOfBottles* se ve incrementada en una unidad. La coherencia de esta acción aumenta si el jugador no tiene ninguna botella y si el jugador se encuentra en las islas 2 o 3, debido a que en estas están los personajes que otorgan ron al estar cansado. Por otro lado, solo es legal si se está cansado.
- ***GetNote*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *NumOfNotes* se ve incrementada en una unidad. La coherencia de esta acción aumenta si el jugador se encuentra en una de las tres islas donde se pueden obtener notas (la inicial, la primera o la segunda) y si se está en la isla correcta para recibir una nota, por ejemplo, si *NumOfNotes* es 0, eso significa que el jugador tiene que conseguir la primera nota, en la primera isla. La acción solo es legal cuando el jugador consigue dicha nota en el lugar adecuado también, por ejemplo, la primera nota solo puede conseguirse si se está en la primera isla y dentro de la posada, la segunda solo se puede conseguir en la segunda isla y dentro de la guardia pirata, etcétera.
- ***GetKey*:** Al aplicarse en un estado, devuelve uno nuevo en el que el booleano *PlayerHasKey* es cierto. La coherencia de esta acción aumenta, por orden de importancia, si el jugador tiene la pala, si el jugador está en el lugar del tesoro y si el jugador está en la segunda isla. La acción solo es legal si el jugador tiene la pala.

- ***GetShovel*:** Al aplicarse en un estado, devuelve uno nuevo en el que el booleano *PlayerHasShovel* es cierto. La coherencia de esta acción aumenta si el jugador ya ha obtenido tres notas, y si el jugador se encuentra en la isla inicial. La acción solamente es legal si el jugador ya ha obtenido tres notas.
- ***Dig*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *NumOfDigs* se ve incrementada en una unidad. La coherencia de esta acción aumenta si el jugador se encuentra en el lugar del tesoro y ha cavado menos de siete veces. La acción solo es legal si el jugador tiene la pala.
- ***ArriveAtIsland0*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *CurrentIsland* es *ISLAND_0* y *CurrentPlace* es *OTHER* (el muelle no es ningún lugar de importancia). Esta acción es coherente si el jugador tiene dos notas (la tercera nota se consigue en la isla inicial) o tiene tres notas, pero no tiene la pala (la pala se consigue en la isla inicial). La acción es legal si el jugador no está cansado y no está ya en la isla inicial (0).
- ***ArriveAtIsland1*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *CurrentIsland* es *ISLAND_1* y *CurrentPlace* es *OTHER* (el muelle no es ningún lugar de importancia). Esta acción es coherente si el jugador no tiene ninguna nota (la primera nota se consigue en la primera isla). La acción es legal si el jugador no está cansado y no está ya en la primera isla (1).
- ***ArriveAtIsland2*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *CurrentIsland* es *ISLAND_2* y *CurrentPlace* es *OTHER* (el muelle no es ningún lugar de importancia). Esta acción es coherente si el jugador tiene una nota (la segunda nota se consigue en la segunda isla) o tiene tres notas y la pala (el lugar del tesoro está en la segunda isla). La acción es legal si el jugador no está cansado y no está ya en la segunda isla (2).
- ***ArriveAtIsland3*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *CurrentIsland* es *ISLAND_3* y *CurrentPlace* es *OTHER* (el muelle no es ningún lugar de importancia). Esta acción es coherente si el jugador tiene la llave de la casa final (esta está situada en la tercera isla. La acción es legal si el jugador no está cansado y no está ya en la segunda isla (3).

- ***ArriveAtInn*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *CurrentIsland* es *ISLAND_1* y *CurrentPlace* es *INN*. La coherencia de esta acción aumenta si el jugador se encuentra en la primera isla y si no tiene ninguna nota (la primera nota se consigue en la posada o *Inn*). La acción solo es legal si el jugador está en la primera isla, ya que no es posible llegar a la posada desde las demás.
- ***ArriveAtHideout*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *CurrentIsland* es *ISLAND_2* y *CurrentPlace* es *HIDEOUT*. La coherencia de esta acción aumenta si el jugador se encuentra en la segunda isla y si tiene una nota (la segunda nota se consigue en la guardia pirata o *Hideout*). La acción solo es legal si el jugador está en la segunda isla, ya que no es posible llegar a la guardia desde las demás.
- ***ArriveAtTreasure*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *CurrentIsland* es *ISLAND_2* y *CurrentPlace* es *TREASURE_SPOT*. La coherencia de esta acción aumenta, por orden de importancia, si el jugador tiene la pala, y el jugador está en la segunda isla (debido a que entonces debe llegar al lugar del tesoro para cavar). La acción solo es legal si el jugador está en la segunda isla.
- ***ArriveAtHome*:** Al aplicarse en un estado, devuelve uno nuevo en el que la variable *CurrentIsland* es *ISLAND_3* y *CurrentPlace* es *HOME*. La coherencia de esta acción aumenta si el jugador se encuentra en la tercera isla y si el jugador tiene la llave de la casa final (la casa final es el lugar de importancia *HOME*). La acción solo es legal si el jugador está en la tercera isla.

Quedan por definir las funciones (*Functions*) usadas por el DM. Al igual que en *Emerald Hunt* (y todos los videojuegos que hagan uso del sistema), estas son la función de evaluación (*EvaluationFunction*), la función de completitud (*CompletenessFunction*), la función objetivo (*GoalFunction*), la función de coste (*CostFunction*), la función heurística (*HeuristicFunction*) y la función de generación (*GenerationFunction*):

- ***EvaluationFunction*:** Hace uso de funciones del *WorldManager* para construir el estado actual. Como el *WorldManager* tiene acceso al *PlayerStatus* y al inventario, pueden solicitársele las variables necesarias para representar un estado a partir de funciones de consulta.
- ***CompletenessFunction*:** Devuelve 1 si los dos estados proporcionados son equivalentes, es decir, si todos sus campos tienen el mismo valor.
- ***GoalFunction*:** Devuelve *true* si el estado actual es igual al objetivo.
- ***CostFunction*:** En el caso concreto de este juego, la función de coste cumple parte de la funcionalidad que se asociaría clásicamente a la heurística. Este diseño agresivo, que otorga valores muy elevados a acciones que podrían percibirse como indeseadas por el autor, contribuye notablemente a orientar la búsqueda llevada a cabo internamente por el DM. Así, por ejemplo, acciones como ir a una isla u otra varían su coste en función del número de notas que aporte el estado padre. Con esto se consigue además descartar rápidamente acciones que no serían prometedoras para alcanzar el estado objetivo indicado al DM.
- ***HeuristicFunction*:** Realizar una estimación numérica global con estados tan fácilmente mutables y con una amplia variabilidad en las acciones que permiten desarrollar el espacio de estados resulta casi imposible. Es por ello que se ha optado por diseñar una función heurística lo más simple posible, que únicamente tenga en cuenta la cercanía del jugador al final deseado del juego; es decir, a mayor número de notas y cercanía con la isla final (3), menor es el valor de retorno.
- ***GenerationFunction*:** Si existe una solución, la función de generación comprueba la pila de acciones y estados para discernir el contenido que debe instanciar en el juego. A continuación, se expondrán algunos ejemplos de cómo funciona en este videojuego la función de generación:
    a. ***Rest*:** Si el jugador debe descansar, se activa la aparición de una botella de ron en la isla en la que está. Si el jugador no ha conseguido descansar en 60 segundos, una botella llegará a una de las playas de

la isla. Si el jugador descansa antes de esos 60 segundos, se cancela la aparición de la botella. Además, se comprueba la siguiente acción que el jugador debe hacer, y se realiza la generación acorde a esa también.

b. ***GetShovel*:** Se llama a la función adecuada del *WorldManager* para que, haciendo uso del *NPCDialoguesManager* explicado anteriormente, se cambie el paquete de diálogos de los personajes al de ID "Estado3.1". Estos diálogos indican al jugador cómo proceder para conseguir la pala.

c. ***Dig*:** Se llama a *HintToTreasure* del *WorldManager*. En esta función se hacen varias distinciones para determinar cómo guiar al jugador de forma adecuada. Si el jugador ha cavado pocas veces, se cambiará el paquete de diálogos de los personajes al de ID "Estado4", en el cual se indica cómo proceder de forma sutil, simplemente mencionando que el jugador debe ir a la segunda isla. Sin embargo, si el jugador cava un número elevado de veces sin conseguir el tesoro, se interpreta como que este no entiende el objetivo aún, por lo que se cambiará el paquete de diálogos de los personajes al de ID "Estado4.1", en el que los diálogos son bastante más concretos a la hora de indicar al jugador lo que debe hacer, mencionando la existencia de un tesoro e incluso hablando del norte de la segunda isla, donde se encuentra este tesoro. Por último, si el jugador cava en un lugar equivocado demasiadas veces en una misma isla, se pondrá a cierto *canSpawnAngryNPC*. Cuando esta variable esté a cierto, la próxima vez que se cave en esa isla, aparecerá un personaje enfadado de debajo de la tierra, el cual indica al jugador exactamente a dónde debe ir, el norte de la segunda isla. Este personaje aparece solamente una vez por isla, a no ser que el jugador viaje a otra isla y vuelva, en cuyo caso aparecerá de nuevo si se cava en el lugar equivocado.

d. ***ArriveAtIsland0*:** Como esta acción puede darse en dos casos diferentes, se hace una distinción importante. Si el jugador no ha obtenido aún la nota de la palmera en esta isla, se cambia el paquete

de diálogos al de ID "Estado3", en el cual los diálogos guían principalmente a esta palmera. Sin embargo, si el jugador ya tiene la nota, significa que debe conseguir la pala, por lo que el paquete de diálogos se cambiará al de ID "Estado3.1", en el cual se habla al jugador sobre un hombre en esta isla, que tiene una pala. Además, se llama a una función del *WorldManager* que accede al *MapSelectorBehaviour* para cambiar las descripciones de todas las islas en este, haciendo que la isla inicial tenga la descripción "segura" y el resto las "peligrosas", indicando que la isla más recomendable para viajar es la isla inicial. Para el resto de las islas la generación sería similar, pero adaptada a las circunstancias de cada isla.

e. ***ArriveAtInn*:** Se cambia el paquete de diálogos al de ID "Estado1", en el cual se insta al jugador a llegar a la posada. También se cambian las descripciones de las islas para que la primera isla sea la "segura" y el resto "peligrosas". Para el resto de las localizaciones (como la guarida pirata, por ejemplo) la generación sería similar, pero adaptada a las circunstancias de cada localización.

Con todas estas funciones definidas, el DM ya puede funcionar. Cada vez que el jugador presione la tecla definida para interactuar ('E' o la barra espaciadora), o cada vez que cava (letra 'Q' una vez se ha obtenido la pala o usando la pala desde el inventario), el sistema se actualiza, recibiendo el estado actual mediante la función de evaluación y haciendo una búsqueda para determinar qué acciones debe aplicar el jugador para llegar al siguiente *plot point* de manera más rápida.

Con *Tell No Tales*, se ha conseguido implementar un videojuego mucho más complejo que *Emerald Hunt* que haga uso del Gestor de la Acción Dramática, demostrando muchas de las cualidades de este sistema.

CAPÍTULO V: EXPERIMENTACIÓN CON USUARIOS DEL SISTEMA

Con el experimento se desea saber hasta qué punto la hipótesis del proyecto se cumple, observando si los participantes sienten que su agencia o libertad a la hora de jugar disminuye, todo ello de acuerdo a la implementación del gestor de narrativa (Drama Manager) realizada en este proyecto. De esta manera se puede saber si merece la pena implementar sistemas similares en videojuegos o entornos más complejos.

De esta manera se puede saber si, por ejemplo, merece la pena poner esfuerzos en programación, implementación, recursos consumidos por la máquina que ejecute el programa en vez de en otros aspectos que podrían poner en peligro el cuidado y pulido de otros elementos de la experiencia.

5.1 Descripción del Experimento

El experimento consistirá principalmente en pedir a los participantes jugar al videojuego implementado y rellenar un cuestionario formado por una serie de preguntas relacionadas con la experiencia. Para realizar el experimento, se cuenta con dos versiones del juego, uno con Drama Manager implementado y otra sin él.

El cuestionario a rellenar es exactamente igual para ambas versiones. A cada participante se le asignará de forma aleatoria una de las dos versiones. El principal objetivo es observar hasta qué punto el Drama Manager enriquece la experiencia del videojuego y analizar cómo de interesante es hacer este tipo de implementaciones en proyectos de pequeña envergadura que, llevados a gran escala, pueden hacer comprender si son factibles o rentables para videojuegos más profesionales.

El cuestionario se compone de dos secciones: una dedicada a la demografía y otra con preguntas sobre la experiencia de juego. Con la sección de demografía se quiere sacar conclusiones para los diferentes públicos objetivos que enriquecerán el análisis del experimento y permitirán detectar ciertos patrones. La segunda sección permitirá sacar conclusiones acerca del Drama Manager. Se compararán las respuestas obtenidas para ambas versiones (con DM y sin DM) y se analizará si merece la pena dedicar recursos a implementar un sistema tan sofisticado como es el Drama Manager o, si, por el contrario, su aportación a la experiencia no justifica el trabajo de desarrollo que conlleva.

En la versión con el DM desactivado se seguirá una línea argumental clara y bien definida, en la que el jugador no tendrá más opción que seguir los pasos que se le marcan. Pase lo que pase, independientemente de las vueltas, viajes o errores que el jugador pueda cometer, el estado actual en cada momento del mundo no se verá alterado lo más mínimo. Los personajes seguirán teniendo las mismas líneas de texto y los diálogos solo cambiarán para los personajes principales una vez que el jugador vaya avanzando en la historia.

El jugador se puede mover libremente por islas y lugares, independientemente del objetivo que tenga a continuación sin recibir *feedback* del juego más allá del que ya ha recibido jugando. Para la versión con DM la cosa cambia sustancialmente. El juego recogerá las acciones fundamentales del jugador y reaccionará en consecuencia. El objetivo del DM es guiar al jugador de forma eficiente por el mundo del juego y los estados, de tal manera que cada vez se acerque más al estado final del juego.

Se analizarán las acciones que quiere hacer el jugador y las ya realizadas, y el motor responderá en consecuencia: invitándole siempre a realizar acciones que completen la historia (como viajar a la isla correcta o hablar con la persona adecuada) o se generarán objetos que permitan al jugador seguir avanzando (como botellas de ron para que el jugador descanse y así poder viajar entre islas). Así como personajes que ayudarán a avanzar en la historia o nuevos diálogos para los NPC's del juego que darán pistas más elaboradas de los pasos que el jugador tiene que dar a continuación. De esta manera, pase lo que pase, el juego siempre evoluciona a favor del jugador de una forma fluida y eficiente, enriqueciendo la historia e impidiendo que el jugador pierda el interés.

5.2 Participantes del Experimento

Para llevar a cabo el experimento en ambas versiones, se eligieron participantes con un amplio rango de edad (12 a 53 años), aunque la gran mayoría sean jóvenes de entre 20 y 30 años. Todos ellos aceptaron de forma voluntaria, sin ningún tipo de remuneración económica y pudiendo abandonar en cualquier momento. También fueron informados de que los resultados obtenidos serían públicos con este trabajo.

5.3 Pautas de Experimentación

Los experimentos han sido realizados mayoritariamente de forma virtual, a través de plataformas como Google Meet o Skype. Aunque en los casos en que ha sido posible (debido a las restricciones de la COVID-19) se han hecho de manera presencial.

En el caso de que la prueba se realizase de forma virtual, los participantes debían descargar el ejecutable del videojuego e instalarlo en su ordenador. Al instalar el juego, los participantes eran informados de la finalidad del mismo, pudiendo decidir libremente si aceptar la prueba o rechazarla.

En el caso de prueba presencial, el ejecutable estaba ya descargado en un ordenador de alguno de los integrantes del grupo de investigación, de tal forma que el participante sólo tenía que instalarlo y acceder voluntariamente al experimento.

Antes de comenzar el juego, cada participante recibió una serie de instrucciones relacionadas con el proceso de instalación y el juego, para que todos los participantes pudiesen completar la prueba satisfactoriamente. Las instrucciones eran las siguientes:

Instrucciones de instalación:

- Cuando recibas el archivo *tfg.zip* y procedas a descargarlo es posible que tu navegador lo reconozca como un virus**.** Si esto ocurre, simplemente haz click en la flecha (o cualquier botón para mostrar más opciones) y selecciona *Guardar* en el menú desplegable.
- Para descomprimir el archivo puedes utilizar cualquier programa habitual como WinRAR**,** 7Zip o la herramienta de serie de Windows 10.
- Una vez descomprimido el archivo *tfg.zip* encontrarás dos documentos: el instalador (.exe) y una copia de estas instrucciones en PDF.
- Es posible que el antivirus detecte el instalador como un programa dañino. Haz click en la opción *Más Información* y aparecerá un segundo botón con la etiqueta *"Ejecutar de todas formas"*; haz click en dicha opción y para comenzar el proceso de instalación.
- Durante la instalación sólo tendrás que aceptar el acuerdo de licencia pertinente y hacer *click* en el botón *"Siguiente"* o *"Instalar"* cuando corresponda.

Instrucciones de juego:

- Puedes salir del juego pulsando la X roja del menú principal situada en la esquina superior derecha.
- Para jugar a nuestro videojuego sólo necesitas un teclado y un ratón o *trackpad*. Puedes consultar los controles desde el menú principal.
- Puedes interactuar con la mayoría de los objetos que hay en el juego.

- Recuerda prestar atención a la historia del videojuego, ya que en el cuestionario hay preguntas relacionadas con la trama.
- Ten en cuenta que *Tell No Tales* es un videojuego diseñado para probar la efectividad de un sistema muy particular en desarrollo, por lo que, por un lado, pueden producirse errores y, por otro, puede que te haya tocado una versión de control en la que dicho sistema no esté implementado.
- Si tienes algún problema durante la partida, contacta con nosotros.
- Cuando completes tu partida, podrás volver al menú de créditos y hacer *click* sobre la opción *"Cuestionario"*.
- El cuestionario es la parte más importante de tu contribución, así que lee atentamente las preguntas y contesta con sinceridad.

Tras completar el juego, los participantes debían rellenar el cuestionario correspondiente a la versión que habían jugado. Para realizar el experimento no se necesitaba ser jugador habitual de videojuegos.

5.4 Cuestionario

Para recoger las respuestas de los participantes se ha optado por usar Google Forms, debido a las facilidades que ofrece a la hora de difundirlo y analizar los datos. Se han hecho dos cuestionarios iguales para cada versión con el fin de facilitar el análisis de datos. El acceso a cada cuestionario se encuentra en el menú principal (mediante un botón interaccionable) de cada uno de los ejecutables. El experimento se ha llevado a cabo entre los días 16 y 24 de enero, teniendo un alcance de 20 personas (40 en total) por cada una de las versiones. El cuestionario se divide principalmente en dos secciones: una de demografía y otra de investigación.

5.4.1 Cuestiones Demográficas

Con esta sección se busca comprender si existe algún tipo de relación entre diferentes variables como son la edad, profesión, género, asiduidad a la hora de jugar, etc, con los datos posteriores que se extraen en la investigación. Las preguntas propuestas fueron las siguientes:

1. *¿Qué edad tienes?*
2. *¿Cuál es tu género?*
3. *¿Cuál es tu profesión?*
4. *¿Cuánto juegas a videojuegos?*
5. *¿Cómo de acostumbrado estás a jugar a videojuegos en los que destaca la narrativa*? Puntúa del 1 (poco) al 5 (mucho).*
6. *¿Cuál ha sido tu nivel de satisfacción general al jugar? Puntúa del 1 (bajo) al 5 (alto).*
7. *Si te pidiese un breve resumen de la historia que has jugado ¿Cuál sería?*

*La pregunta hace referencia a aquellos videojuegos en los que la historia es una parte muy importante, siendo muchas veces el motivo principal para querer jugarlo. Algunos ejemplos serían *Red Dead Redemption II, The Last of Us, Batman Arkham City, Detroit Become Human* o *Tomb Raider*.

5.4.2 Cuestiones sobre Experiencia y Agencia

En esta segunda sección se encuentran el resto de las preguntas del cuestionario, todas relacionadas con el juego, la sensación de libertad y la experiencia de los jugadores. Estas preguntas son las más importantes, ya que permitirán sacar las conclusiones de la investigación.

Se analizarán los valores relacionados con la fluidez con la que se desarrolla el juego, la sensación de libertad y la sensación de estar guiado o perdido que ha tenido cada participante.

Una vez obtenidos los resultados, se compararán con el grado de asiduidad a jugar a videojuegos narrativos de los encuestados.

8. *¿En qué medida crees que el juego ha ido transcurriendo de manera fluida? Puntuar de 1 (nada fluida) a 5 (muy fluida).*
9. *¿Cuál es la historia detrás del protagonista?*
10. *¿Cuál ha sido el personaje o personajes más relevantes para el desarrollo de la historia?*
11. *¿Quién es el personaje al que controlamos? Elegir una:*

- *Un amigo del capitán que se menciona en las notas*
- *Un marinero*
- *El capitán que se menciona en las notas*
- *Un pirata*
- *El hijo del capitán al que se menciona en las notas*

12. *¿Te has sentido en algún momento perdido y sin saber qué hacer a continuación? Puntúa de 1 (poco perdido) a 5 (muy perdido).*

13. *¿Cuántos son los puntos en los que has tenido que tomar decisiones realmente relevantes para el transcurso de la historia? Elegir una:*

- *Uno*
- *Dos*
- *Más de dos*
- *Ninguno*

14. *Si la pregunta anterior es distinta a "Ninguno", ¿Cuáles han sido esas decisiones relevantes?*

15. *¿En qué medida crees que el juego te ha guiado de manera constante a lo largo de la historia? Puntúa del 1 (poco) al 5 (mucho).*

16. *¿Crees que si hubieses jugado de otra manera tu experiencia hubiese cambiado? Elegir una:*

- *Si*
- *No*

17. *¿Cuáles son los objetos o coleccionables que te han resultado claves para entender la historia?*

18. *¿Hasta qué punto te han parecido creíbles las conversaciones con los personajes de las islas? Puntúa de 1 (poco creíbles) a 5 (muy creíbles).*

19. *¿Qué ocurre al final del juego?*

20. *¿Qué grado de libertad has tenido? Puntúa del 1 (escasa libertad) al 5 (libertad absoluta).*

21. *¿Cuánto te ha gustado la historia detrás del juego? Puntúa del 1 (nada) al 5 (mucho).*

22. *¿Te gustaría comentarnos algo más?*

*Los resultados del cuestionario serán accesibles cuando se publique el trabajo (finales de febrero de 2021)

5.5 Resultados del Experimento

A continuación, se recopilarán resultados obtenidos en ambos cuestionarios, poniendo énfasis en los datos que se consideren más importantes referentes a la demografía y la experiencia del juego.

5.5.1 Resultados del Cuestionario A (con DM)

Los 20 participantes aceptaron formar parte del experimento y lo completaron con total libertad.

Con relación a los resultados demográficos, un 90% de los encuestados se encuentra en la veintena de edad, exceptuando a dos personas, una que se encuentra en la treintena y otra en la cincuentena. Un 40% de los participantes declararon jugar a videojuegos entre 5 y 10 horas semanales, un 15% más de 10 horas semanales, un 5% entre 2 y 5 horas semanales, un 15% entre 1 y 2 horas semanales y un 25% nunca o de manera casual. Así mismo, se les preguntó en una escala del 1 al 5 cómo de acostumbrados están a jugar videojuegos narrativos, respondiendo el 35% un valor de 5 (el más alto), un 20% un valor de 3 (valor medio) y un 25% el valor más bajo. Otro 10% respondió con un valor bajo (no el más bajo) y el otro 10% restante con un valor alto (por encima de la media, pero sin ser el valor más alto).

Los participantes también fueron preguntados por el nivel de fluidez que habían notado mientras jugaban, resultando ser un 90% un nivel alto y un 10% un nivel bajo (sin ser el más bajo posible). De la misma forma, el 60% declaró haberse sentido perdido en algún momento de la historia, mientras que el 40% no tuvo problemas. Así mismo, en una escala del 1 al 5, el 55% de los participantes afirmó haberse sentido guiado por el juego (sin ser el valor más alto), un 25% puntuó con el valor máximo y el 20% restante otorgó un valor medio. Por último, se les pidió

que puntuaran del 1 al 5 el grado de libertad que habían sentido jugando, respondiendo algo más de la mitad (55%) con un valor por encima de la media, un 25% con el valor medio y el 20% con un valor bajo.

5.5.2 Resultados del Cuestionario B (sin DM)

Los 20 participantes aceptaron formar parte del experimento y lo completaron con total libertad. Con relación a los resultados demográficos, un 50% de los encuestados se encuentran en la veintena de edad, un 10% en la treintena, un 15% en la cincuentena y el 25% restante entre 18 y 19 años. Un 30% de los participantes declaró jugar a videojuegos más de 10 horas semanales, un 5% entre 5 y 10 horas semanales, un 20% entre 2 y 5 horas y el 45% restante nunca o de manera ocasional. De igual manera, se les preguntó en una escala del 1 al 5 cómo de acostumbrados están a jugar videojuegos narrativos, respondiendo el 40% con el valor más bajo, un 20% con el valor medio, un 5% con el valor más alto, un 25% con un valor bajo (no el más bajo) y el 10% con un valor alto (por encima de la media, pero sin ser el más alto).

Los participantes también fueron preguntados por el nivel de fluidez que habían notado en el juego, siendo un 40% un nivel alto, un 30% un nivel medio y un 30% un nivel bajo. De la misma forma, un 75% de los encuestados declararon haberse sentido perdidos en algunos momentos de la historia. Sin embargo, el 55% afirmó haberse sentido guiado por el juego durante la experiencia. Por último, se les preguntó en una escala del 1 al 5 sobre el grado de libertad que habían tenido en el juego, experimentando casi la mitad de los resultados (45%) una máxima libertad, un 20% un valor alto y el 35% restante un valor medio.

5.6 Análisis de los Resultados del Experimento

A continuación, se van a analizar los resultados obtenidos en ambas versiones. A la hora de agrupar las respuestas a preguntas en las que era necesario puntuar en una escala del 1 al 5, se ha optado por tomar las respuestas como afirmativas a partir del valor 3. Por ejemplo, a la pregunta: ¿Te has sentido perdido? Si el encuestado

respondía con un 3, se tomará como un sí (en menor medida que alguien que puntuó con un 5, pero como un sí).

5.6.1 Análisis de Resultados del Cuestionario A (con DM)

Tras analizar los resultados obtenidos, se pueden exponer las diferencias encontradas entre los jugadores más acostumbrados a jugar juegos que destacan por su narrativa y los que no.

Se observa que el 92% de los jugadores más experimentados consideran que juego ha transcurrido de manera fluida. Sin embargo, un porcentaje menor, el 86%, de los no experimentados opinan que la narrativa lo ha sido.

El 92% de los jugadores experimentados frente al 71% de los no experimentados considera que las interacciones con los personajes del juego son creíbles.

Se observa que tanto el 100% de los jugadores experimentados como el 100% de los no experimentados han tenido, en mayor o menor medida, la sensación de haber sido guiados durante todo el juego.

Figura 24: Gráfica enfrentando la experiencia en juegos narrativos contra la sensación de sentirse guiado constantemente. Se encuentra una tendencia en la que la pendiente es plana. La sensación de sentirse guiado de media se mantiene independientemente del grado de experiencia con juegos narrativos.

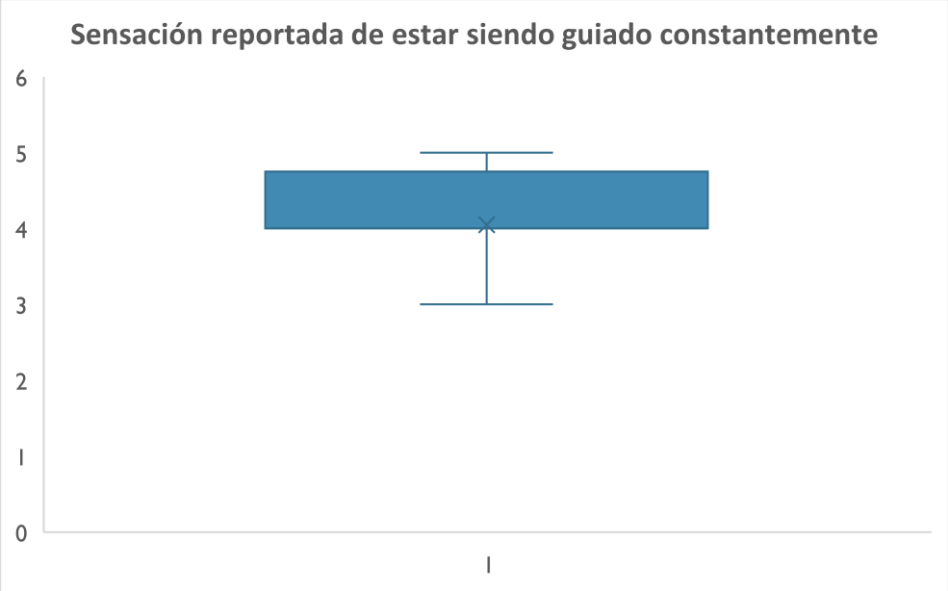

Figura 25 Box-Plot Sensación de estar siendo guiado constantemente Versión con Dama Manager. Muestra como los valores oscilan entre el 5 y el tres con una mayor cocnentración entre el 4 y el 5.

Por otro lado, el 62% de los jugadores experimentados ha tenido la sensación en algún momento de que no sabía qué hacer. El 57% de los no experimentados tienen esa misma sensación.

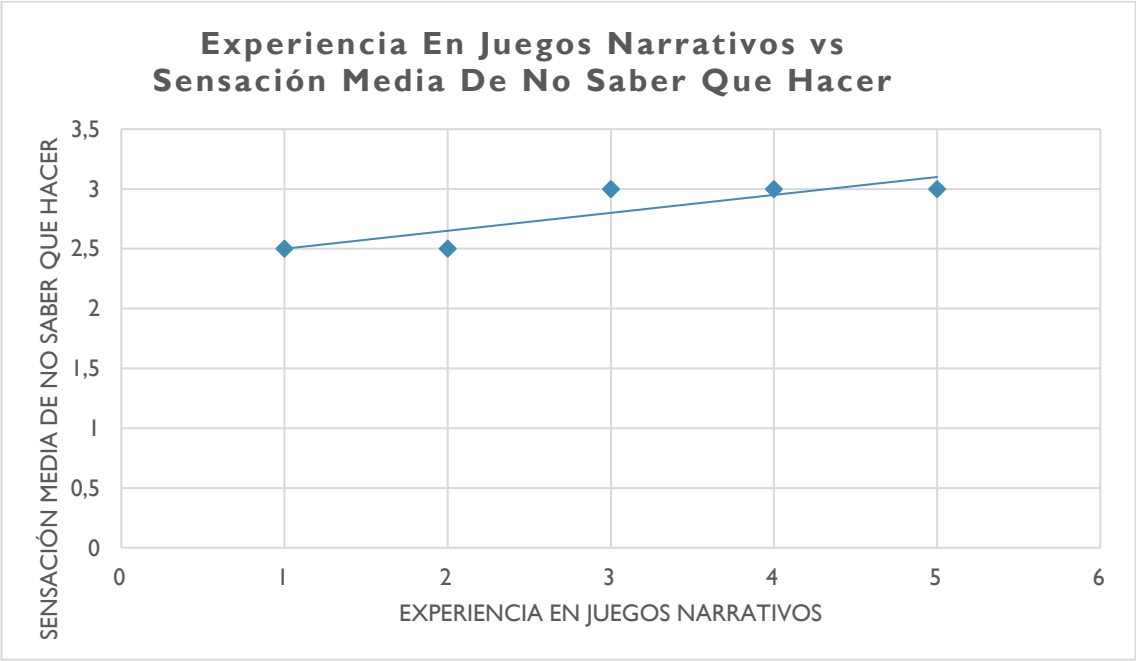

Figura 26: Gráfica enfrentando la experiencia jugando a juegos narrativos y la sensación de no saber qué hacer. Se comprueba cómo la sensación de estar perdido, de media, se hace más patente a medida que aumenta lo acostumbrado que está el usuario a juegos narrativos.

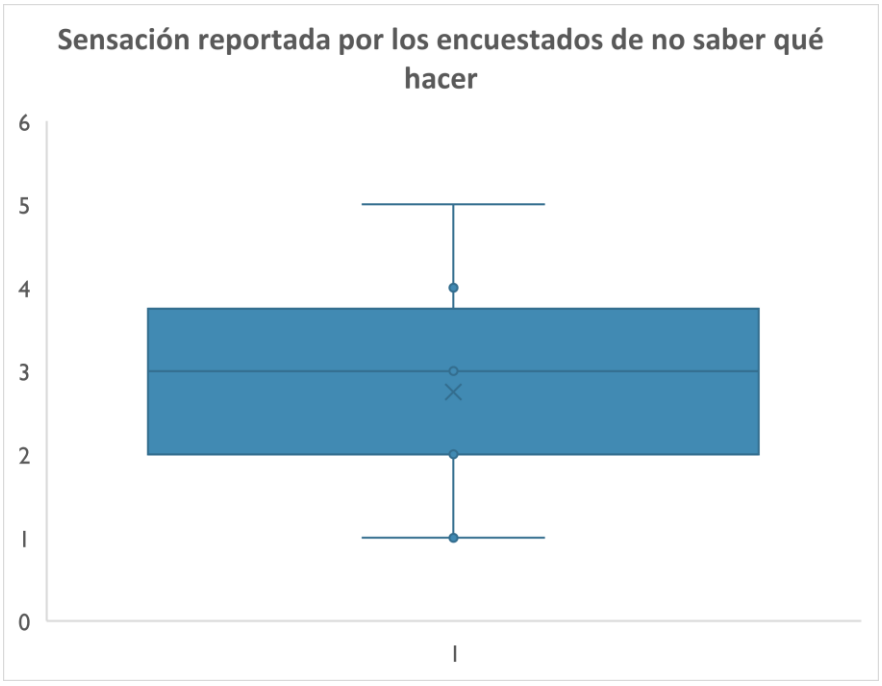

Figura 27 Box-Plot Sensación de no saber qué hacer Versión con Dama Manager. Muestra que los valores oscilan en todo el registro de valores pero que se concentran entre los valores 2 y 3.

Por último, el 92% de los jugadores experimentados han sentido altas cotas de libertad, mientras que solo el 57% de los menos experimentados han tenido esa sensación.

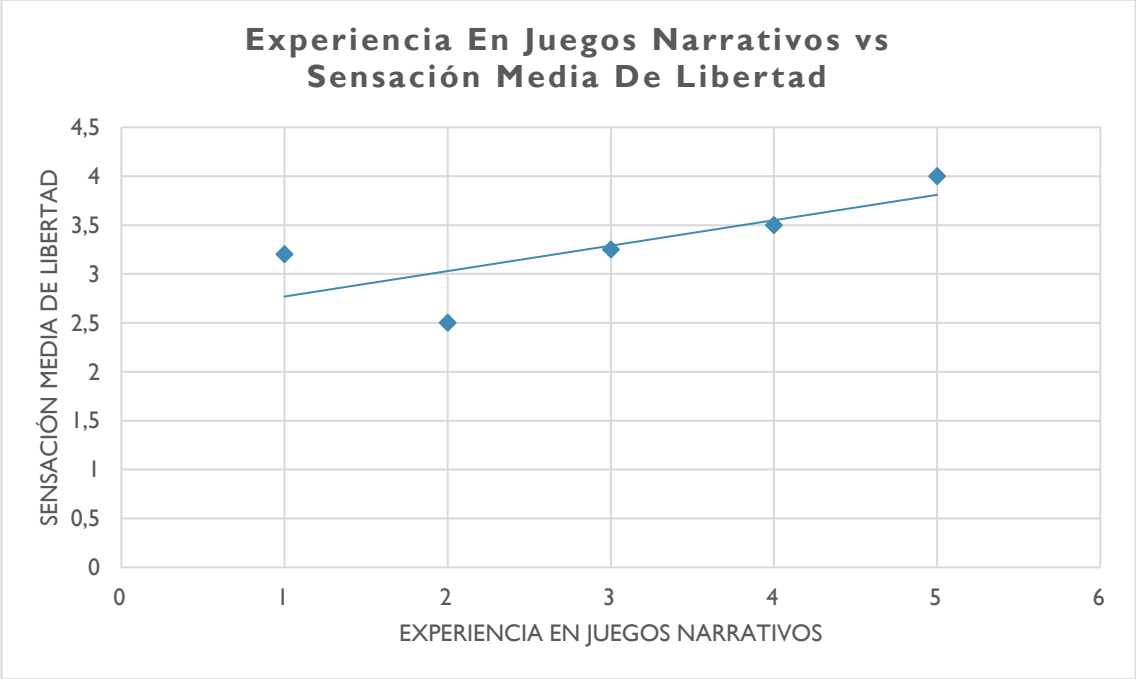

Figura 28. Gráfica enfrentando la sensación de libertad con la experiencia jugando juegos narrativos. Se trata de la pendiente más pronunciada. Aquellos que han dedicado más tiempo a juegos de narrativa son los que han tenido, de media, mayor sensación de libertad durante el experimento.

Figura 29 Box-Plot sensación de libertad Versión con Dama Manager. Los valores se concentran entre los valores 3 y 4 con elementos que caen hasta valores próximos al 2 y valores que llegan hasta el 5.

5.6.2 Análisis de Resultados del Cuestionario B (sin DM)

Tras analizar los resultados obtenidos, se puede extraer una serie de diferencias entre aquellos jugadores acostumbrados a jugar videojuegos narrativos y los que no. Se puede observar que una amplia mayoría (86%) de los jugadores experimentados consideran que el juego ha transcurrido de forma fluida, mientras que esta sensación sólo ha estado presente en algo más de la mitad de los no experimentados (62%).

En cuanto a la credibilidad de las interacciones personajes, los resultados son muy similares para los jugadores experimentados y los menos experimentados, un 86% y un 85% respectivamente.

Refiriéndose a la sensación de estar guiado por el juego, los jugadores experimentados afirman en un 71% tener ese sentimiento frente a un más pobre 46% de los no experimentados, que afirman tener dicho sentimiento en algún momento.

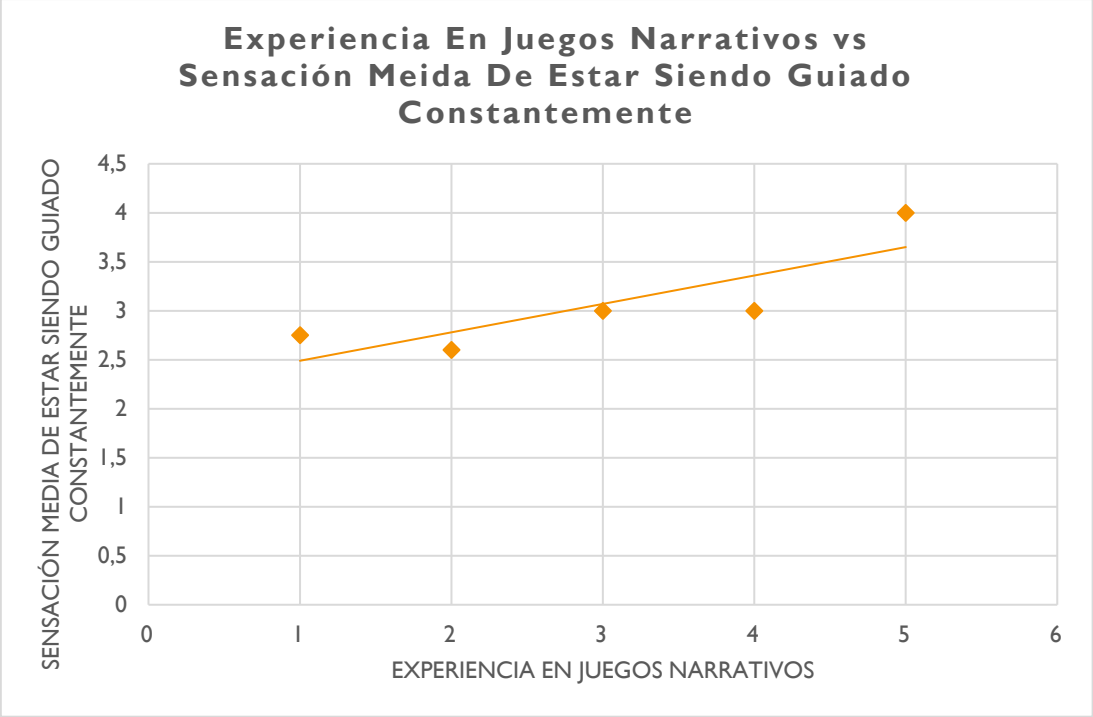

Figura 30: Gráfica enfrentando la experiencia en juegos narrativos contra la sensación de estar siendo guiado constantemente. Se puede observar que los jugadores con más experiencia, de media, se sienten más guiados, ya que les resulta más fácil encontrar patrones que hayan observado en otros juegos.

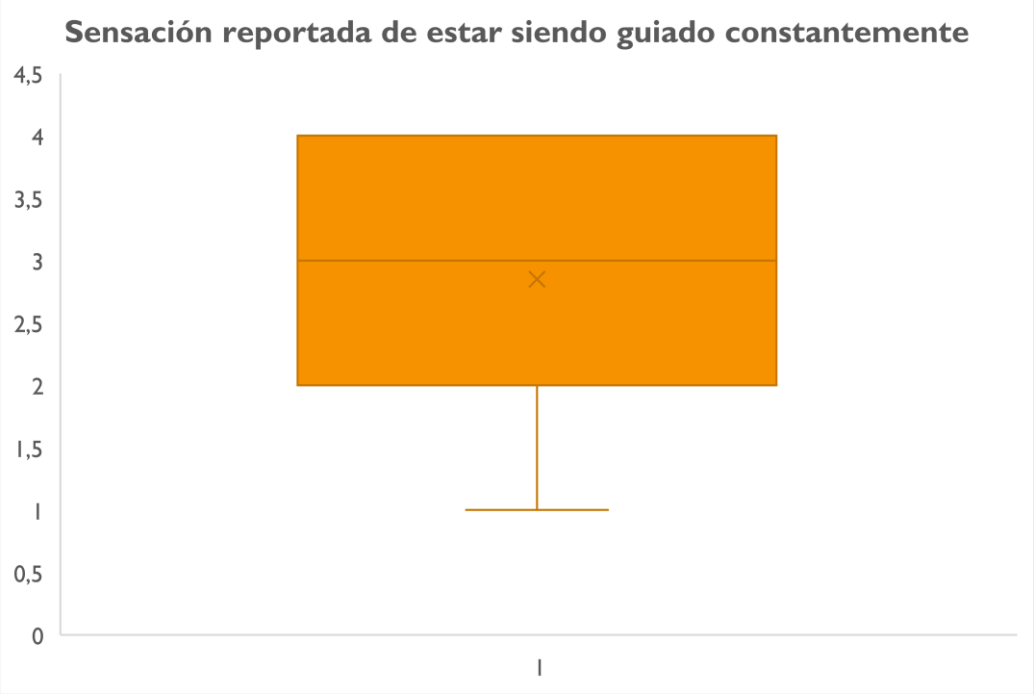

Figura 31 Box-Plot Sensación de estar siendo guiado constantemente Versión Sin Dama Manager. Muestra una mayor concentración entre los niveles 2 y 4 con un valor mínimo de hasta 1.

En cuanto a la sensación de sentirse perdidos en algún momento del juego, un 71% de los jugadores experimentados afirma haberlo sufrido, mientras que un mayor porcentaje de los no experimentados (92%) ha experimentado esa sensación de no saber que hacer en algún momento.

Figura 32: Gráfica enfrentando la experiencia en juegos narrativos contra la sensación de no saber qué hacer. Los jugadores con más experiencia experimentan, de media, menos sensación de no saber qué hacer.

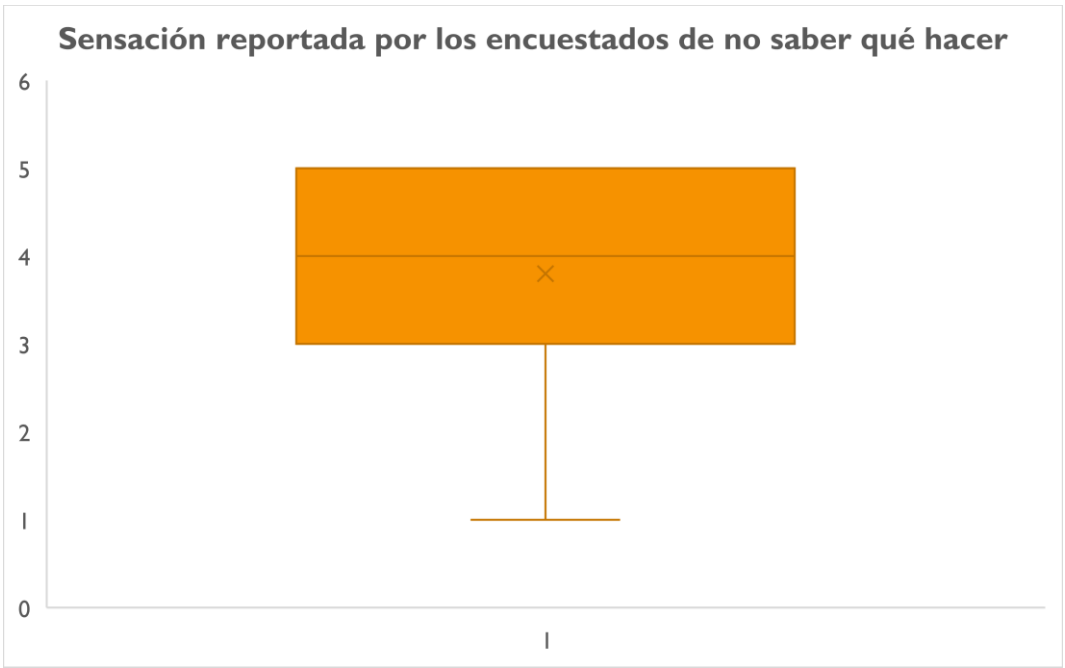

Figura 33 Box-Plot Sensación de no saber qué hacer Versión Sin Dama Manager. Demuestra una concentración de valores entorno a los 3-5 puntos en los niveles de no saber qué hacer con unos mínimos que bajan hasta el 1.

Con respecto a la sensación de libertad, tanto jugadores experimentados como no experimentados coinciden en que se han sentido libres durante la partida en mediana o mayor medida con un 100% en ambos casos.

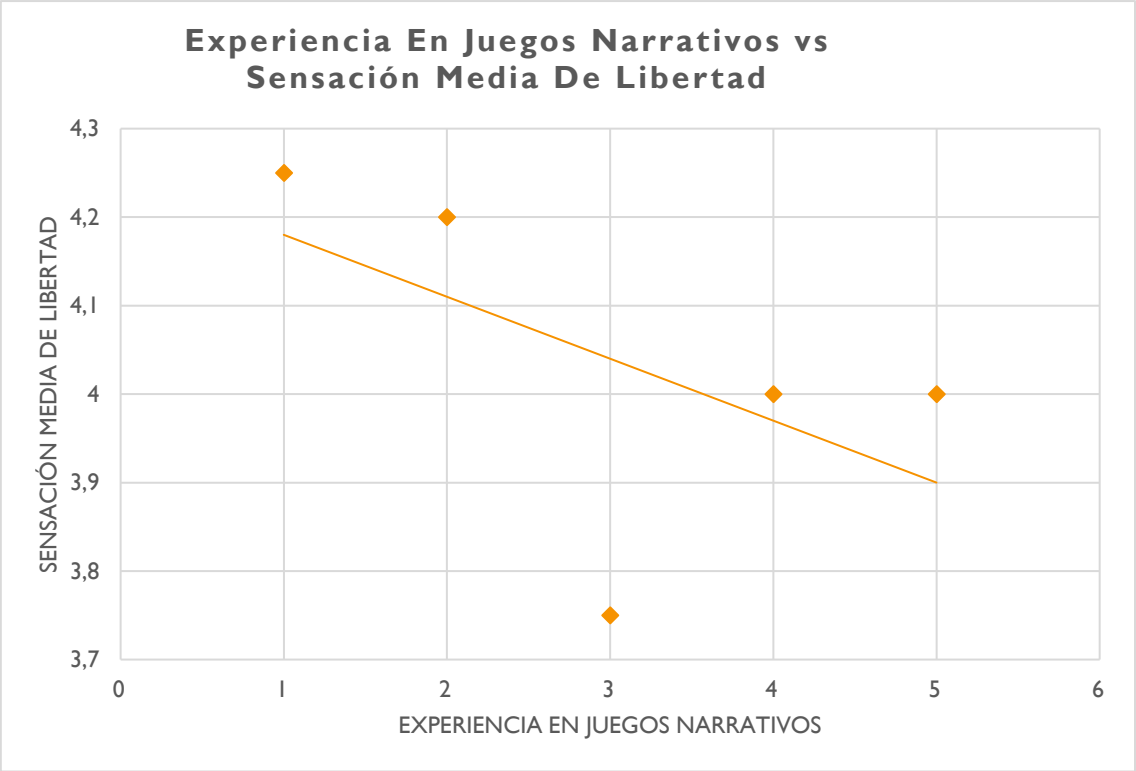

Figura 34: Gráfica enfrentando la experiencia en juegos narrativos contra la sensación de libertad. Esta sensación de media decrece cuanto mayor es la experiencia del participante jugando a juegos de carácter narrativo.

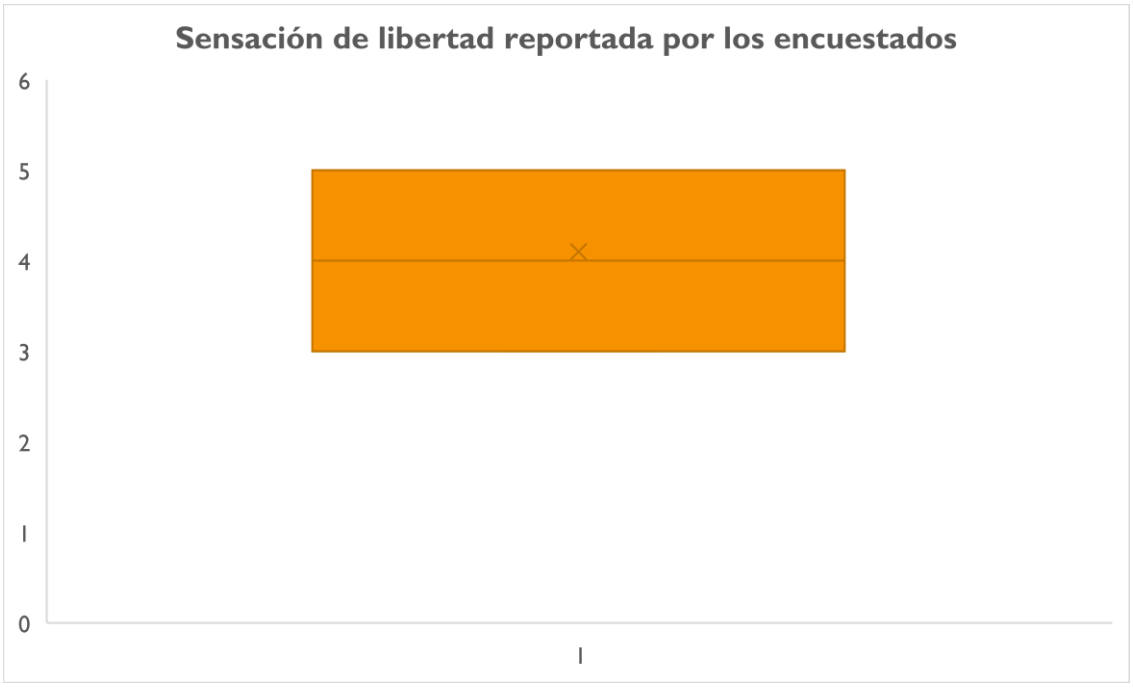

Figura 35 Box-Plot sensación de libertad Versión Sin Drama Manager. Muestra una dispersión muy pequeña de los datos y una concentración entre los valores de 3 y 5.

5.6.3 Análisis de Resultados Globales

Comparando los resultados obtenidos tras el análisis de sendos cuestionarios, se pueden destacar las siguientes diferencias entre los diferentes tipos de jugadores. El 40% de los participantes de la versión con DM creen que la experiencia es medianamente fluida, mientras que otro 50% la consideran muy fluida. Respecto a los participantes de la versión sin DM, un 30% creen que el juego es medianamente fluido frente a un 40%, a los que les ha parecido muy fluida.

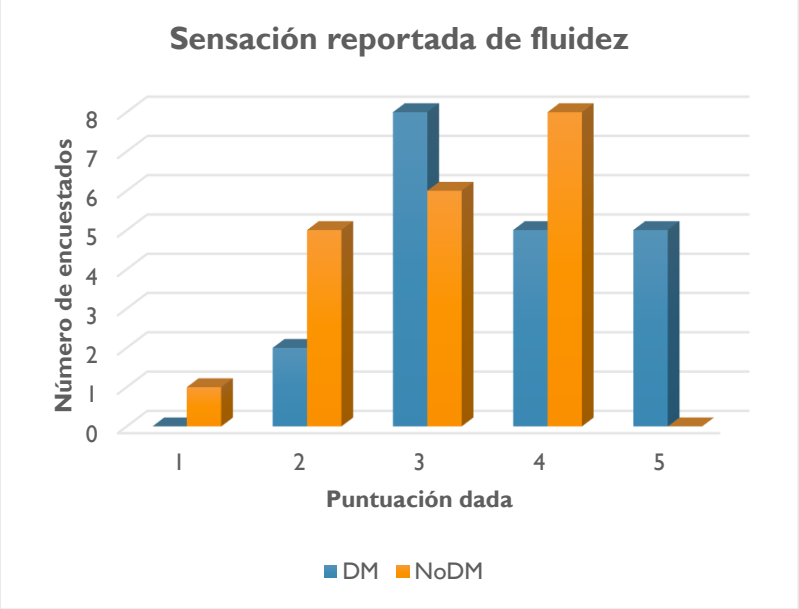

Figura 36: Gráfica comparando la sensación de fluidez de las versiones con DM y sin DM.

Respecto a la credibilidad de las interacciones con personajes, un 30% de los jugadores de la versión con DM creen que son creíbles, frente a un 55% que las consideran muy creíbles. Respecto a la versión sin DM, un 75% de los encuestados opinan que las conversaciones son muy creíbles frente a otro 10% que cree que lo son medianamente.

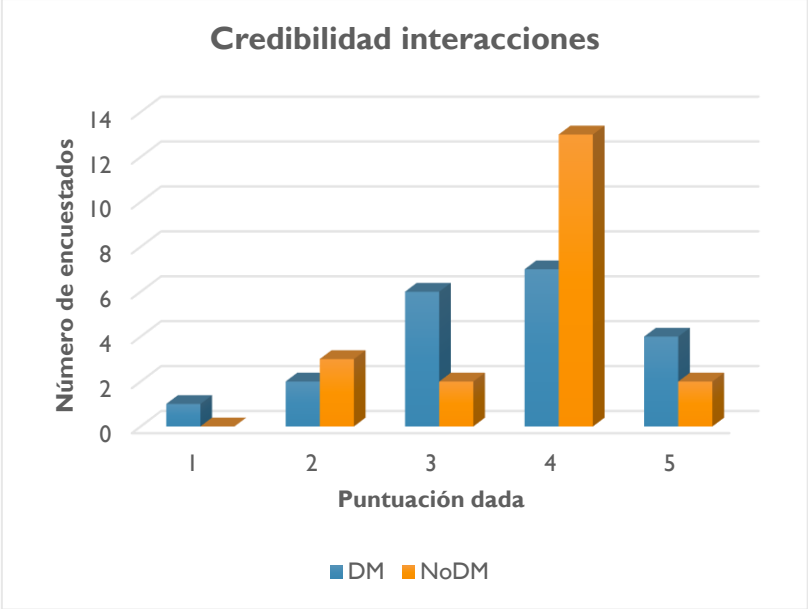

Figura 37: Gráfica comparando la credibilidad de las iteraciones de las versiones con DM y sin DM.

En relación a la sensación de sentirse guiado por el juego, el 20% de los jugadores de la versión con DM afirman que la experiencia está guiada con respecto a una gran mayoría (80%) que opinan que está muy guiada. En la versión sin DM, se encuentra que un 15% de los participantes que se han sentido medianamente guiados por el juego frente a un 40% que afirman haberse sentido bastante guiados.

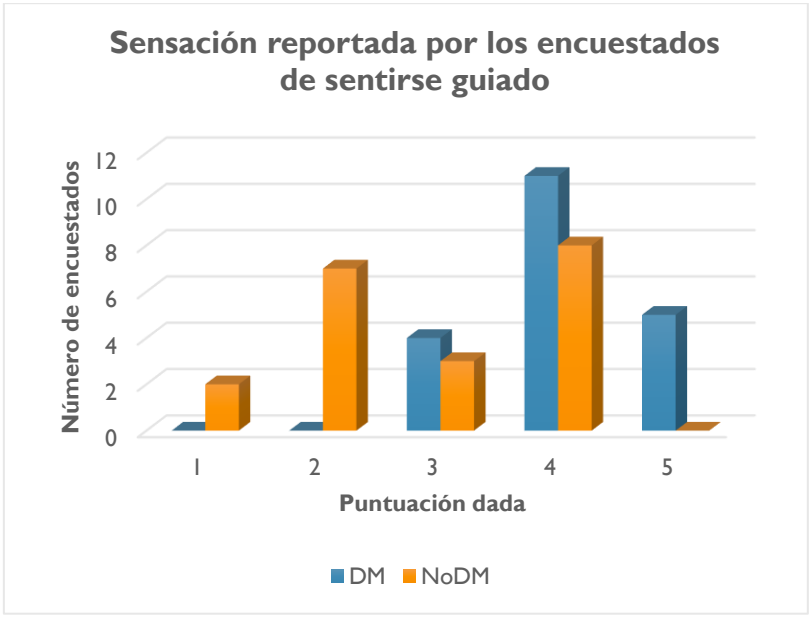

Figura 38: Gráfica comparando la sensación de sentirse guiado de las versiones con DM y sin DM.

Un 35% de los participantes de la versión con DM se han sentido algo perdidos con respecto a otro 25% que se han sentido muy perdidos. En relación a la versión sin DM, un 15% afirman haberse encontrado perdidos frente a un 70% que se han visto muy perdidos durante la experiencia.

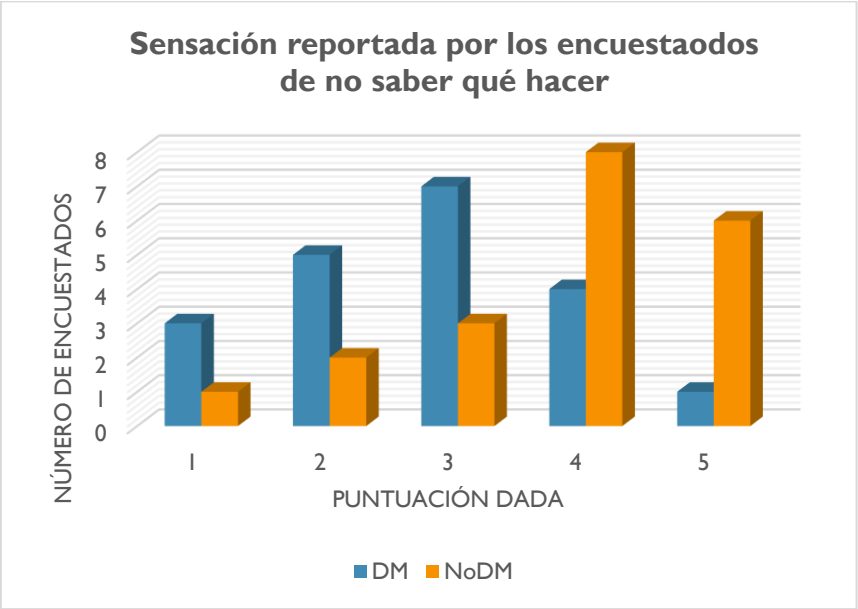

Figura 39: Gráfica comparando la sensación de no saber qué hacer de las versiones con DM y sin DM.

Con relación a la sensación de libertad experimentada por los jugadores en la versión con DM, se tiene que un 25% ha experimentado cierto grado de libertad (valor medio), frente a un 55% en los que este grado ha sido más alto. En la versión sin DM, se tiene que un 35% se han sentido medianamente libres durante la experiencia con respecto a un 60% que afirma haber experimentado total libertad.

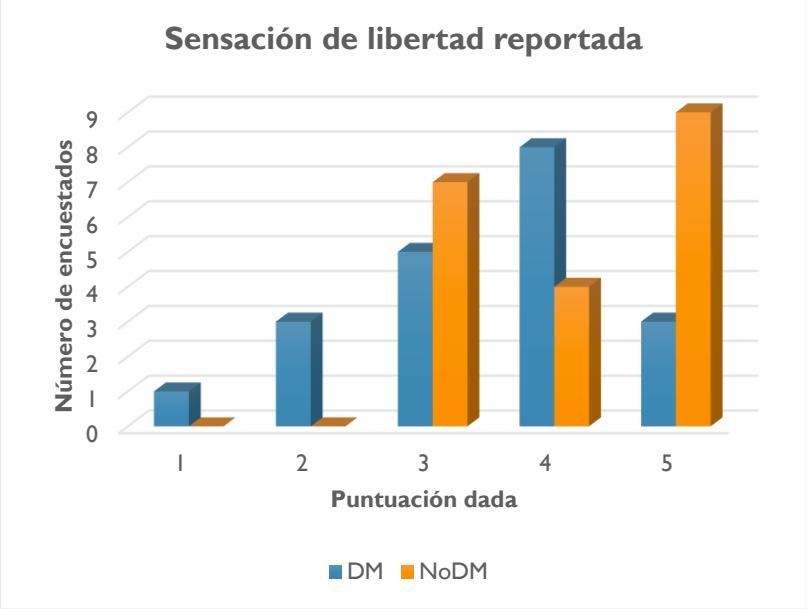

Figura 40: Gráfica comparando la sensación de libertad de las versiones con DM y sin DM.

CAPÍTULO VI: DISCUSIÓN

Una vez analizados los resultados obtenidos en los experimentos, en este capítulo se procede a su discusión. Se tendrán en cuenta los datos y las gráficas reflejados en los apartados **5.5 Resultados** y **5.6 Análisis** respectivamente.

Aunque a la hora de realizar la discusión los resultados puedan parecer no conclusivos, tras analizar los datos, se puede afirmar que las tendencias serían las mismas para muestras mucho mayores. Partiendo de este punto es desde donde se desarrolla este ejercicio. Asimismo, en el segundo apartado del capítulo se hará una discusión sobre el proyecto en su conjunto.

6.1 Discusión de los resultados del experimento

Cuando se comparen los porcentajes se debe recordar que se han obtenido basándose en una escala del 1 al 5 y que el punto de corte para concretar si un encuestado pertenece a un grupo o a otro se ha puesto en el 3, inclusive, a favor de las puntuaciones más altas.

A grandes rasgos y echando un primer vistazo a los resultados, se podría pensar que algunos se alejan un poco de lo que se esperaba al comienzo de la investigación. Con el *Drama Manager* se suponían unas cotas de libertad altas a todos los niveles y, sobre todo, superiores a las de la versión del juego sin él.

Debe quedar patente que las respuestas recopiladas se basan en la experiencia obtenida tras jugar a un videojuego en el que se encuentran más parámetros que

pueden alterarlas de alguna manera, como por ejemplo las mecánicas, el diseño del juego, la historia, etc, y que, dependiendo de su calidad y complejidad, pueden dar lugar a sensaciones totalmente diferentes.

El videojuego implementado es una prueba de concepto en la que el mayor detalle se ha puesto en la implementación y estructuración del Drama Manager y todos los sistemas que se comunican con él (incluido el propio videojuego). El esfuerzo en el diseño del juego ha residido en crear una experiencia sencilla, entretenida y adaptada a todos los públicos, y quizás no ha sido el enfoque más adecuado. Un diseño más elaborado, detallado o un experimento más específico o controlado (para las versiones con y sin Drama Manager) se hubiese acercado más a que el jugador no experimentase una reducción de la libertad en la versión con Drama Manager.

En primera instancia, si se comparan los porcentajes referentes a la fluidez del juego para ambas versiones (con DM y sin DM), parece ser que los valores obtenidos siguen la tónica que se esperaba**,** siendo mayores para la versión con Drama Manager. Si bien es cierto que, esto podría estar muy relacionado con el diseño del juego, los resultados más favorables en la versión con DM hacen ver la implementación del sistema como algo exitoso.

Además, los altos valores obtenidos en la credibilidad de las interacciones con los personajes, en especial los jugadores experimentados (92%), refuerzan esa idea de que el Drama Manager funciona de manera correcta, modificando y seleccionando los diálogos pertinentes a medida que avanza el juego.

Es cierto que los jugadores no experimentados, en este caso, declaran en menor porcentaje (71%) esa sensación de credibilidad. Sin embargo, esto puede ser debido a que por su poca costumbre jugando videojuegos esperasen la calidad vista en películas, libros o videojuegos mucho más ambiciosos y con una profundidad y complejidad mayor en los diálogos. Aun así, el porcentaje sigue siendo alto.

Para la versión sin Drama Manager, los valores de credibilidad en las interacciones se mantienen muy similares para ambos tipos de jugadores, un 86% en el caso de los experimentados y un 85% en el caso de los no experimentados.

Por otro lado, y como se ve en la **Figura 37,** comparando los resultados para ambas versiones, se encuentra que en términos generales los resultados en credibilidad son también positivos para la versión con Drama Manager.

En lo que se refiere a las sensaciones de sentirse guiado por el juego y sentirse perdido en algún punto de la experiencia, el 100% de los participantes de la versión con Drama Manager (jugadores experimentados y no experimentados) se han sentido muy guiados. Esto se puede ver en la **Figura 25** en la que los valores se concentran, con poca dispersión entre los valores de 4 y 5, indicando altos grados de esa sensación. Figura 29

Así mismo, los porcentajes relacionados con la sensación de estar perdidos, también en la versión con Drama Manager, son mucho menores, del 62% para los jugadores experimentados y de 57% para el resto.

Vemos como en la **Figura 27** los valores para la sensación de no saber que hacer en la versión con Drama Manager se concentran entre el 2 y el 4, reforzando esa idea de que en términos generales los jugadores no se sienten del todo perdidos.

Nuevamente, esto da una idea de que el Drama Manager está generando respuestas idóneas a las acciones del jugador para que en todo momento este sepa cómo continuar la aventura. El hecho de que los porcentajes en la sensación de sentirse perdidos, vistos los altos valores en la sensación de estar siendo guiados, no son más pequeños, se relacionan nuevamente con el menor detalle y profundidad en el diseño del juego. Sin embargo, los resultados son prometedores y favorables.

Para la versión sin Drama Manager los resultados obtenidos a la sensación de no saber qué hacer y la de estar siendo guiados pueden parecer poco coherentes para el caso de los jugadores experimentados. Pues son porcentajes relativamente altos y similares (71% para ambas variables)**.** Esto es, porcentajes no despreciables de

jugadores experimentados creen que el juego le ha guiado y sin embargo que también se ha sentido perdidos en algún momento.

Esta incoherencia se achaca nuevamente a problemas en el diseño que se han podido detectar en respuesta a preguntas relacionadas con la experiencia en si misma dentro del cuestionario que se realizó.

En el caso de los no experimentados, se observa que una gran cantidad de los encuestados se han sentido perdidos (92%), y que un porcentaje muy inferior se ha sentido guiado a lo largo del juego (46%), lo que tiene sentido. Al hablar de gente no habituada a jugar videojuegos se puede entender que la creencia entre los participantes es que ambas cosas sean excluyentes, dando los porcentajes obtenidos como válidos.

A grandes rasgos encontramos valores con una alta concentración entre el 3 y el 5, como se puede observar en la **Figura 33,** para la sensación de no saber qué hacer y entre el 2 y el 4 para la sensación de ser guiados constantemente como se observa en la **Figura 31.**

Comparando los datos para ambas versiones (con y sin Drama Manager) obtenidos para ambas versiones reflejados en la **Figura 38** y **Figura 39** se puede observar, a grandes rasgos, como existe una mayor adhesión a la narrativa en la versión con el Drama Manager en funcionamiento.

En términos de libertad experimentada por los participantes en el caso de la versión con Drama Manager se ve como los jugadores no experimentados han sentido cotas de libertad mucho inferiores (57%), y es que por su poca experiencia jugando juegos narrativos, pueden llegar a entender, que el hecho de recibir por parte del juego una retroalimentación que les orienta en la aventura, en el caso de nuestra implementación tratándose de diálogos que se actualizan y orientan al jugador acerca de cuál es el próximo paso que tiene que dar, es de alguna manera una obligación, perdiéndose esa sensación de libertad. Esto se ve muy claramente en el caso de los jugadores experimentados, que en porcentaje (92%), han sentido libertad a la hora de jugar a pesar de haberse sentido guiados en la historia. Es

posible que esto se deba a que su mayor experiencia en juegos de este tipo les hace entender que el hecho de recibir información sobre lo que pueden hacer para avanzar, no le resta libertades a la hora de actuar, jugar y avanzar en la historia.

Como podemos observar en la **Figura 29** tenemos una gran concentración de valores entre el 3 y el 4, lo que apunta en términos generales a la existencia en buena medida de una sensación de libertad.

En la versión sin el Drama Manager, el 100% coinciden en que se han sentido libres a la hora de jugar. Sin embargo, los resultados obtenidos en la sensación de no saber qué hacer y en la de estar siendo guiados, dan a entender que esa libertad se ha conseguido a expensas de unos jugadores que no han encontrado unas instrucciones más específicas y claras, y que al no verse obligados a actuar de según que formas, han experimentado esa libertad, en detrimento de la experiencia jugable.

En la **Figura 35** vemos como los valores para la sensación de libertad en la versión sin Drama Manager se concentran en gran parte entre el 3 y el 5 con una dispersión mínima. Esto señala que en términos generales se experimenta unas altas tasas de libertad para el total de los encuestados.

La **Figura 40,** en la que se compara el grado de libertad experimentado por los participantes para ambas versiones (con y sin Drama Manager), es reflejo de lo que se menciona en líneas anteriores, y se puede ver como a grandes rasgos, y a excepción de los jugadores menos experimentados, las tasas de libertad son altas.

Analizando los datos en su totalidad se observa que la versión con el Drama Manager no solo ha sido una experiencia que para una mayoría de los jugadores experimentados ha estado impregnada de libertad, sino que de alguna manera ha sido más completa y fácil de terminar.

Tras ver como a lo largo de los experimentos la implementación ha funcionado y que el juego ha ido evolucionando sin problemas técnicos, se puede asegurar que ha sido exitosa. Sin embargo, los resultados ayudan a entender que, efectivamente, para sacar mayor partido al Drama Manager, queda del lado del desarrollador que

quiera hacer uso de él, un trabajo de diseño acorde a las cotas de calidad del videojuego que desee crear.

6.2 Discusión General

Con el marco teórico estudiado en los capítulos previos y con los resultados del experimento analizados, se puede realizar una discusión más global del proyecto. Pese a que los resultados no son conclusivos, podemos extraer algunas reflexiones valiosas. Aunque la hipótesis de la que se partía ha sido demostrada parcialmente, queda mucho trabajo que realizar para implementar un gestor de narrativa completo que cumpla las expectativas planteadas.

Si bien, como veremos en el capítulo siguiente, los objetivos planteados han sido cumplidos en su totalidad, aunque la investigación no ha arrojado los resultados esperados, sí que se ha cumplido la hipótesis planteada al comienzo. Cabe plantearse la posibilidad de un diseño muy simple del experimento, o tal vez una falta de profundidad en las preguntas y temas tratados en el cuestionario.

Muchos de los sesgos del experimento y de los participantes en el mismo ya se han discutido en el apartado anterior, por lo que no se profundizará mucho más en ellos, pero si es importante subrayarlos. , ya que, aunque los resultados obtenidos no hayan sido los esperados, si que han sido útiles y válidos.

La implementación de un gestor narrativo completo aplicado al juego sobre el que queríamos llevar a cabo el experimento habría requerido de un trabajo mucho más exhaustivo y prolongado, y con este proyecto se buscó desde un principio la eficiencia en términos de tiempo de desarrollo y funcionalidad implementada.

Muchos de los gestores de narrativa que se estudiaron en el **CAPÍTULO II**:

TRABAJO PREVIO no centraban su diseño en la interactividad tanto como el que se desarrolla en este proyecto. Uno de los puntos clave del sistema debía ser su utilidad como elemento integrador en un videojuego, que ya de por sí, es una de las aplicaciones informáticas más interactivas que existen.

La narrativa en los videojuegos está evolucionando, pero las herramientas para equilibrar la autoría y la agencia parecen estancadas en proyectos alejados de la industria más comercial, ya que requieren de una gran cantidad de recursos y optimización a nivel de programación y diseño que muchas veces no merece la pena para los resultados que se van a obtener. Aun así, los cimientos que sustentan este proyecto parten de estos sistemas previos y pueden ser de utilidad para el diseño de futuras funcionalidades a incluir.

Por ejemplo, MINSTREL va un paso por delante en lo que a representación del conocimiento y generación de contenido se refiere. Pese a que nuestro sistema se centra más en la gestión narrativa, sí es cierto que una representación más precisa del conocimiento sólo aportaría potencia creativa al sistema.

Autores como Gordon (Andrew S. Gordon, 2017) utilizan el término psicología del sentido común para referirse a las teorías que todos utilizamos implícitamente para descifrar el comportamiento humano, en términos de creencias, objetivos, planes y emociones. Si bien se plantean acercamientos bastante exhaustivos que formalizan todo este conocimiento, resulta complicado imaginar un gestor de narrativa que gestione adecuadamente toda esta información.

Cuanto más se quiere generalizar la implementación del sistema, mayor esfuerzo de diseño requiere. Este es uno de los retos que se planteó en el paso del primer prototipo al prototipo final, y que se resolvió poniendo el foco en una representación del conocimiento más simplificada y un entorno más controlado y limitado. Estas limitaciones vienen impuestas, en parte, por el medio escogido para la integración del gestor: el videojuego. Sistemas como *Universe*, basados en generación de texto, pueden permitirse un mayor esfuerzo de generación.

De otros sistemas como *Virtual Storyteller* o *Fabulist* se tomó la idea del sistema como director de la acción, una suerte de narrador omnisciente que trata de reforzar una historia estructurada. Sin embargo, y, de nuevo, por optar por un medio interactivo, no se consideró oportuno importar el modelo que proponen basado casi íntegramente en la simulación.

La narrativa emergente sería el resultado más probable de este tipo de implementación en un videojuego, perjudicando profundamente la autoría del diseñador pero aportando diferentes historias a cada jugador. No obstante, la presencia de un director en el sistema fue la inspiración clave para la implementación del Evaluador.

Este subsistema del DM es uno de los candidatos con más potencialidad para ser mejorados. Como ya se adelantaba en el capítulo 3, la completitud de los *plot points* es una cuestión que en muchos casos depende del propio diseñador. A través de nuestra investigación parece que aportar mayor flexibilidad a estos condicionantes sólo beneficiaría a la sensación de libertad percibida por los jugadores, en detrimento menor de la autoría.

Sistemas de gestión narrativa como *MEXICA* plantean la coherencia en términos postcondicionales, pero en este proyecto se ha optado por un acercamiento más similar al de *Author*, en ese sentido. Construir la coherencia narrativa es una labor, por una parte, previa al inicio de la simulación y/o evento interactivo, y como mucho puede construirse de forma simultánea al devenir de los acontecimientos. La decisión de aportar una Agenda Narrativa blinda en gran parte esta coherencia, siempre y cuando la heurística aportada por el diseñador sea completa.

Si bien esto garantiza el cumplimiento de los arcos narrativos propuestos por el autor, es una contrapartida a la hora de facilitar la generación de nuevo contenido no previsto. Una vez más, una decisión orientada en otro sentido habría orientado este proyecto hacia la emergencia, y no tanto hacia la proceduralidad.

La filosofía de diseño del gestor de narrativa podría haberse visto ampliamente mejorada en una tercera iteración del modelo, que sin embargo quedó fuera de la implementación por razones estrictamente temporales.

La incorporación de algoritmos evolutivos podría haber supuesto una dinamización interesante en la generación del espacio de estados. Igualmente, un acercamiento a la generación narrativa basada en el conocimiento habría permitido la consulta de

repositorios, idealmente online, para la adición de nuevo contenido narrativo una vez acabado el espacio de posibilidades ofrecido por el sistema planteado.

En esta línea de mejora de lo ya establecido, se podrían haber incluido algoritmos de búsqueda distintos a los planteados. En relación con el proceso de búsqueda, no se han detectado grandes ralentizaciones, en cualquier caso. A pesar de que el sistema ya cuenta con un algoritmo A* optimizado y restringido en varios ámbitos, existen acercamientos que podrían haber guiado el proyecto a la inclusión de nuevos módulos. Por ejemplo, la inclusión de una búsqueda estocástica habría acercado el proyecto al modelo evolutivo que se mencionaba antes.

En vista de los resultados obtenidos en los experimentos, y atendiendo a los comentarios de los participantes en los mismos, uno de los módulos que parece más interesante para integrar sería el de modelización del jugador. Obtener un perfil del jugador en base a sus acciones y hacer que el sistema de gestión narrativa oriente la búsqueda de estados deseables a aquellos estados que, idealmente, gustarán más al jugador, podría ser el siguiente paso en el desarrollo de este sistema. Esto podría generar respuestas aún más positivas por parte de los jugadores, haciendo sientan más esa sensación de libertad y agencia mencionadas anteriormente, ya que cada uno jugaría una historia adaptada a él mismo.

En el siguiente capítulo se explorarán adecuadamente estas opciones de futuro, y se expondrán las conclusiones definitivas de todo el proceso de investigación y desarrollo.

CAPÍTULO VII: CONCLUSIONES Y TRABAJO FUTURO

Se ha desarrollado un sistema de gestión la acción dramática con un acercamiento en el diseño centrado en la trama, que puede utilizase en cualquier entorno narrativo propuesto por el usuario, siempre y cuando el trabajo de diseño se adapte a las necesidades del autor. Esto, no obstante, puede resultar un trabajo arduo para el usuario del sistema, que no siempre valorará positivamente esta adición a su proyecto.

Con el Drama Manager, se pone al alcance de cualquier desarrollador de videojuegos una herramienta potente de fácil implementación en cualquier proyecto que utilice C# como lenguaje de programación básico, aunque se haya diseñado con el motor Unity en mente.

Implementar un sistema de estas características en una de las herramientas de desarrollo de videojuegos más populares en la actualidad como es Unity, abre las puertas a futuros avances en el campo de la narrativa interactiva y procedural.

El sistema, sin embargo, no está completo, ni incluye todos los avances en diseño e implementación que se han realizado a lo largo de los años, pero es una sólida piedra angular para trabajo futuros, que se cubrirá en el último subapartado de este capítulo. Antes, algunas reflexiones y conclusiones sobre el proyecto.

7.1 Conclusiones

Con el proyecto se busca aportar evidencia de que existe una mejora de la calidad de la experiencia narrativa al utilizar sistemas de gestión de la acción dramática en videojuegos. Atendiendo a los resultados de los experimentos, se pueden apreciar indicios de que, efectivamente, estos sistemas pueden mejorar la experiencia de usuario sin afectar notablemente a su sensación de libertad.

No obstante, el esfuerzo que supone integrar el DM en un juego, a nivel de diseño y trabajo de autor, puede convertir la utilización del sistema en un desafío para aquellos desarrolladores no muy experimentados en los videojuegos narrativos. Por otra parte, una vez dominado, el sistema es lo bastante potente como para permitir un amplio abanico de posibilidades al usuario.

El sistema de gestión de la acción dramática que se ha empleado es capaz de guiar al jugador a lo largo de un arco dramático sin grandes dificultades; una vez más, queda en manos del diseñador facilitar en mayor o menor medida este proceso, pero el sistema cumple este cometido. No se puede perder de vista el hecho de que el jugador, ante todo, se aferrará a su libertad, y en historias complejas esta tarea de orientación puede ser mucho más complicada.

La versatilidad es, tal y como se ha propuesto, una de las principales virtudes del sistema. Es por ello por lo que integra diferentes algoritmos de búsqueda. Se ha conseguido que el Drama Manager sea un sistema con una heurística completa, aunque si se busca una mayor optimalidad aún hay recorrido. El usuario puede definir qué algoritmos usar, e incluso cambiarlos en tiempo de ejecución.

El Drama Manager, además, utiliza un lenguaje de representación del conocimiento basado en acciones y estados que diseña el propio usuario, por lo que puede volverse todo lo complejo o sencillo que requiera.

Esto ha sido posible, en gran medida, gracias al diseño que se ha hecho del propio entorno virtual del videojuego, así como de sus mecánicas. Desarrollar el juego con las necesidades del DM en mente ha sido clave para facilitar la comunicación entre ambos módulos. Evidentemente, no se puede esperar que cualquier juego sea tan

fácilmente adaptable, pero una vez más todo quedaría en manos del usuario. No obstante, parece lógico que, si alguien hiciera uso del sistema implementado y se plantease integrarlo en un videojuego con un fuerte componente narrativo, cabría esperar que diseñase con estas consideraciones en mente.

El entorno diseñado y las herramientas seleccionadas permitieron desarrollar con soltura ambos prototipos, por lo que cabe destacarlos como elementos clave a la hora de desarrollar rápidamente videojuegos narrativos. Del mismo modo, gracias a Unity fue posible montar con soltura una *build* funcional en multitud de equipos, lo que permitió distribuir el juego y el cuestionario con eficacia y sin grandes problemas.

Aún con todo, quedan muchos interrogantes sobre la integración de este tipo de sistemas en los videojuegos que no sean puramente narrativos. Los jugadores parecen satisfechos, pero habría que ir un paso más allá para conseguir una verdadera proceduralidad narrativa. Conseguir la implementación integral de un gestor de narrativa en un videojuego depende de un esfuerzo considerable del diseñador, así como de la incorporación de otros módulos que expandan sus capacidades.

7.2 Trabajo Futuro

En la actualidad se pueden encontrar propuestas de diseño de *drama managers* con módulos más allá de los que se ha implementado. Por ejemplo, existen sistemas orientados a modelar la personalidad del jugador, cuyo objetivo es ofrecer arcos narrativos que tienen mayor probabilidad de gustar a esa persona.

En el módulo de resolución de problemas podrían añadirse nuevos algoritmos, así como sofisticaciones al mecanismo de búsqueda en el espacio de estados. Idealmente, en los casos en los que la búsqueda no arrojase resultados, sería interesante incluir un generador de historia. Así, si el jugador llegase a un punto en el que el DM fuese incapaz de encontrar un camino que lo orientase hacia el punto

de la trama previsto, podría generarse nuevo contenido que abriese la posibilidad de completar la historia.

Del mismo modo, la representación del conocimiento podría llegar a ser mucho más compleja y dar cabida a una generalización mayor de las acciones. Esto favorecería esa generación de historia procedural, aunque plantearía otras dudas. Paralelamente a esa generación procedural de historia, el subsistema de generación debería ser capaz de producir texturas, terrenos, personajes, música y cualquier tipo de datos digital necesario para representar el entorno virtual de un videojuego.

Si se fuese un paso más allá, esta generación procedural podría llevar incluso a plantearse un cambio de mecánicas y dinámicas en el propio videojuego. No obstante, está por ver que un videojuego cuyo contenido sea enteramente procedural pueda ser viable comercialmente, a pesar de que su valor académico sería incuestionable.

Bibliografía

Andrew S. Gordon, J. R. (9 de 2017). *A Formal Theory of Commonsense Psychology: How People Think People Think.* CAMBRIDGE. Obtenido de https://www.ebook.de/de/product/29801046/andrew_s_gordon_jerry_r_hobbs_a_formal_theory_of_commonsense_psychology_how_people_think_people_think.html

Bringsjord, S., & Ferrucci, D. (1999). Artificial Intelligence and Literary Creativity: Inside the Mind of Brutus, A Storytelling Machine.

Bulitko, V., Björnsson, Y., Sturtevant, N., & Lawrence, R. (1 de 2011). *Real-time Heuristic Search for Pathfinding in Video Games.*

Chauvin, S., Levieux, G., Donnart, J.-Y., & Natkin, S. (8 de 2015). Making sense of emergent narratives: An architecture supporting player-triggered narrative processes. *2015 IEEE Conference on Computational Intelligence and Games (CIG).* IEEE. doi:10.1109/cig.2015.7317936

Cui, X., & Shi, H. (11 de 2010). A*-based Pathfinding in Modern Computer Games. *11*.

D'argenio, A. (2 de 2018). Gaming Literacy: What is emergent gameplay? Gamecrate. *Gaming Literacy: What is emergent gameplay? Gamecrate*.

Dehn, N. (1981). Story Generation After TALE-SPIN. *IJCAI.*

Dunfield, B. (2017). *Exploration of Narrative Structure in Games for Story Creation.* Carleton University. doi:10.22215/etd/2018-12727

Fan, A., Lewis, M., & Dauphin, Y. (5 de 2018). Hierarchical Neural Story Generation.

Hernandez, S. P., Bulitko, V., & Hilaire, E. S. (1 de 2014). Emotion-based interactive storytelling with artificial intelligence. *Proceedings of the 10th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2014*, 146–152.

Hogg, C., Lee-Urban, S., Muñoz-Avila, H., Auslander, B., & Smith, M. (2011). Game AI for Domination Games. En *Artificial Intelligence for Computer Games* (págs. 83–101). Springer New York. doi:10.1007/978-1-4419-8188-2_4

Iorizzo, M. B. (2016). The Mystery Machine: An Answer Set Programming Approach to Generating Plots for Games.

Khalpada, P., & Garg, S. (10 de 2019). Balancing Consistency and Plot Structure in Computational Storytelling. *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).* IEEE. doi:10.1109/iemcon.2019.8936138

Koenitz, H. (2015). *Interactive digital narrative : history, theory and practice.* New York: Routledge, Taylor & Francis Group.

Lebowitz, M. (6 de 1984). Creating characters in a story-telling universe. *Poetics, 13*, 171–194. doi:10.1016/0304-422x(84)90001-9

León, C., Gervás, P., Delatorre, P., & Tapscott, A. (2020). Quantitative Characteristics of Human-Written Short Stories as a Metric for Automated Storytelling. *New Generation Computing, 38*, 635–671. doi:10.1007/s00354-020-00111-1

Lin, J., Zhao, Y., Huang, W., Liu, C., & Pu, H. (6 de 2020). Domain knowledge graph-based research progress of knowledge representation. *Neural Computing and Applications, 33*, 681–690. doi:10.1007/s00521-020-05057-5

Lindley, C. A. (2005). Story and Narrative Structures in Computer Games. (B. Bushoff, Ed.) *Developing interactive narrative content sagas-sagasnet-reader*.

Marr, B. (2 de 2018). The Key Definitions Of Artificial Intelligence (AI) That Explain Its Importance. *The Key Definitions Of Artificial Intelligence (AI) That Explain Its Importance*.

Martens, C., & Cardona-Rivera, R. (2017). Procedural narrative generation. *Procedural narrative generation.* Obtenido de https://www.gdcvault.com/play/1024143/Procedural-Narrative

Muriel, D., & Crawford, G. (1 de 2018). Video Games and Agency in Contemporary Society. *Games and Culture, 15*, 138–157. doi:10.1177/1555412017750448

Murray, S. (2017). Building Worlds in No Man's Sky Using Math(s). *Building Worlds in No Man's Sky Using Math(s)*.

Nelson, M. J., & Mateas, M. (2008). Another Look at Search-Based Drama Management. *AAAI.*

Nielsen, S. (2008). *Understanding video games : the essential introduction.* New York: Routledge.

Niesz, A. J., & Holland, N. N. (1984). Interactive Fiction. *Critical Inquiry, 11*, 110–129. Obtenido de http://www.jstor.org/stable/1343292

PÉrez, R. P., & Sharples, M. (4 de 2001). MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence, 13*, 119–139. doi:10.1080/09528130010029820

Riedl. (1 de 2012). Interactive Narrative: A Novel Application of Artificial Intelligence for Computer Games Artificial Intelligence in Computer Games 1. *Proceedings of the National Conference on Artificial Intelligence, 3*.

Riedl, M. O., & Bulitko, V. (12 de 2012). Interactive Narrative: An Intelligent Systems Approach. *AI Magazine, 34*, 67. doi:10.1609/aimag.v34i1.2449

Riedl, M. O., & Stern, A. (2006). Believable Agents and Intelligent Story Adaptation for Interactive Storytelling. En *Technologies for Interactive Digital Storytelling and Entertainment* (págs. 1–12). Springer Berlin Heidelberg. doi:10.1007/11944577_1

Riedl, M. O., & Young, R. M. (9 de 2010). Narrative Planning: Balancing Plot and Character. *Journal of Artificial Intelligence Research, 39*, 217–268. doi:10.1613/jair.2989

Riedl, M., Stern, A., Dini, D., & Alderman, J. (1 de 2008). Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications - ITSSA, 4*.

Riedl, M., Thue, D., & Bulitko, V. (2011). Game AI as Storytelling. En *Artificial Intelligence for Computer Games* (págs. 125–150). Springer New York. doi:10.1007/978-1-4419-8188-2_6

Roberts, D., Riedl, M., & Isbell, C. (5 de 2011). Beyond Adversarial: The Case for Game AI as Storytelling.

Russell S., N. P. (11 de 2018). *Artificial Intelligence: A Modern Approach, Global Edition.* Addison Wesley. Obtenido de https://www.ebook.de/de/product/25939961/stuart_russell_peter_norvig_artificial_intelligence_a_modern_approach_global_edition.html

Sengers, P. (2000). 1. Narrative Intelligence. En *Human Cognition and Social Agent Technology* (pág. 1). John Benjamins Publishing Company. doi:10.1075/aicr.19.04sen

Sharma, M., Ontañón, S., Mehta, M., & Ram, A. (5 de 2010). DRAMA MANAGEMENT AND PLAYER MODELING FOR INTERACTIVE FICTION GAMES. *Computational Intelligence, 26*, 183–211. doi:10.1111/j.1467-8640.2010.00355.x

Short, T. (2019). *Procedural storytelling in game design.* Boca, Raton, FL: CRC Press.

Taljonick, R. (8 de 2014). Shadow of Mordor's Nemesis system is amazing–here's how it works. *Shadow of Mordor's Nemesis system is amazing–here's how it works*.

Theune, M., Faas, E., Nijholt, A., & Heylen, D. (3 de 2003). The Virtual Storyteller: Story Creation by Intelligent Agents.

Togelius, J., Kastbjerg, E., Schedl, D., & Yannakakis, G. N. (2011). What is procedural content generation? *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games - PCGames \textquotesingle11*. ACM Press. doi:10.1145/2000919.2000922

Turner, S. (1994). *The creative process : a computer model of storytelling and creativity.* Hillsdale, N.J: L. Erlbaum.

Wang, K., Bui, V., Petraki, E., & Abbass, H. A. (2018). Human-Guided Evolutionary Story Narration. *IEEE Access, 6*, 13783–13802. doi:10.1109/access.2018.2797879

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Rush, A. M. (10 de 2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing.

Wreden, D., & Pugh, W. (2016). The Stanley Parable A Negotiation Expressive Choice Reality Time The Stanley Parable. *The Stanley Parable A Negotiation Expressive Choice Reality Time The Stanley Parable*.

Yannakakis, G. N., & Togelius, J. (7 de 2011). Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing, 2*, 147–161. doi:10.1109/t-affc.2011.6

ANEXO A: INTRODUCTION (TRANSLATED TO ENGLISH)

Narrative has been a part of videogames since the early days of the industry, although it has seen a significant evolution. In classic titles such as *Super Mario Bros.* (Nintendo, 1985), the narrative was presented as a complement to the videogame itself, often told in external media such as the instruction manual (Iorizzo, 2016).

Over the years, narrative has taken on a more important role, becoming the core of many videogames. The narrative not only provides context and story, but in many occasions it manages to create motivation in the players to continue advancing in the videogame, through emotion and curiosity (Iorizzo, 2016).

Due to the interactive nature of videogames, interactive narrative arises, in which players can create or influence a story through actions. The goal of interactive narrative is to immerse players in a virtual world and make them feel like they are an integral part on how the story unfolds and that their actions have consequences (Riedl & Bulitko, 2012).

An example of interactive narrative is a branching tree-like narrative in which players' decisions lead to a series of predefined outcomes, as opposed to the linear form taken by non-interactive narratives in which events always occur in the same order (Niesz & Holland, 1984; Dunfield, 2017).

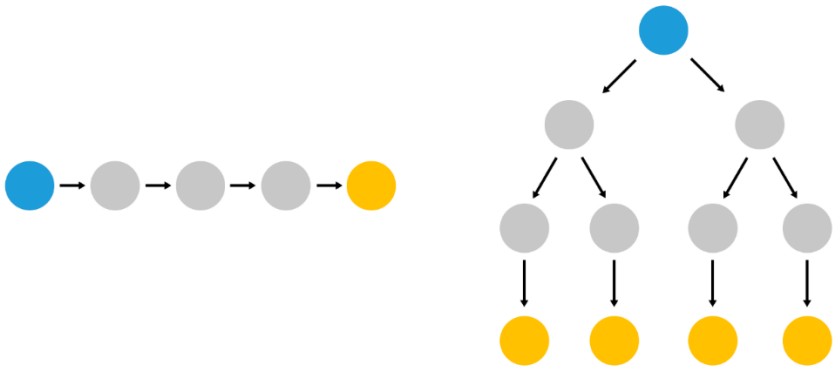

Figure 1: On the left, the form taken by the linear non-interactive narrative. On the right, the form taken by the branching interactive narrative. In blue, the beginning of the story. In yellow, the end.

However, the focus of this research is on another type of interactive narrative: procedurally generated narrative. Procedural content generation has many advantages when applied to video games, as it allows customization of the game for each player, which has been shown to increase user enjoyment (Yannakakis & Togelius, 2011).

Procedural narrative generation shares these same advantages, thanks to adding player agency and customization to a videogame, which concludes in greater immersion for the player (Dunfield, 2017). To achieve higher levels of agency and customization, the role of determining what the player's experience should be (including how the world responds to the player's actions) can be delegated to an Artificial Intelligence in charge of this.

A Drama Manager (DM) is an intelligent system that aims to control the state of the world so that a structured narrative unfolds over time without reducing the perceived agency of the player.

This system uses the narrative principle of looking at the possible futures of the player's experience and determining what should happen in the world to achieve a structured and entertaining experience (Riedl, Thue, & Bulitko, 2011).

Artificial intelligence in video games refers to the techniques and algorithms used to increase and improve the playable experience (Riedl, 2012).

In videogames it is usually conceived that the objective of an artificial intelligence is to generate the illusion of intelligence (human or animal) in the behavior of non-playable characters (NPC), such as enemies or allies: characters with whom, ultimately, one interacts.

The definition of Artificial Intelligence seems to have different meanings depending on the field of study and application, which, although based on the same foundations, have different nuances (Marr, 2018).

However, in this research, when a reference is made to Artificial Intelligence, it will be made thinking of the more general concept that the father of AI, Alan Turing, gave at the time: AI is a computer system or program capable of understanding and managing a series of actions on the system and responding intelligently accordingly.

Actually, the fact that there is a character that reacts intelligently to an action implies that there is a system underneath with the characteristics described above, but it is important not to remain simply with the isolated idea of AI as a character that acts intelligently.

In Artificial Intelligence, problems in the field of interactive narrative related to drama management are usually modeled as state spaces. With a DM, these states represent a specific configuration of the videogame environment. The system is responsible for choosing a sequence of states through which the player should pass to achieve the desired experience (Hogg, Lee-Urban, Muñoz-Avila, Auslander, & Smith, 2011). Of course, there are more approaches (Riedl, 2012) to the problem of narrative generation, but the object of this study will be a drama manager.

A.1 Motivation

Narrative is present in all cultures and eras. A phenomenon as complex as narrative is difficult to model using conventional techniques from the field of Artificial Intelligence, and it is equally complex to evaluate the results of the experiments.

Previous approaches to drama management systems in videogames have offered limited results, given that the space of possibilities offered in their implementations is quite limited; it cannot be ignored that there is no complete solution to the problem of interactive narrative. Besides, very few of these solutions have been implemented within the conventions of a modern narrative video game.

We live in a context where video games are a global phenomenon and the feeling of control and freedom is a key point in their enjoyment (Muriel & Crawford, 2018). This need for freedom sometimes contrasts with the designer's desire to tell a story. The motivation of this project is purely technical: it is wanting to balance freedom of play with narrative, relying not only on pre-existing knowledge about Artificial Intelligence, but also on user experience evaluation techniques closer to Psychology.

A.2 Hypothesis

With the motivation of the previous section, it is proposed as an initial hypothesis that it is possible to develop a drama management system that adapts to various narrative contexts offered in a virtual environment such as those seen in videogames, ensuring the fulfillment of certain key points of the plot specified by the designer, and without the player perceiving during his stay in that environment a significant reduction in his sense of agency.

Specifically, it is proposed that the use of drama managers in video games can ensure that the player perceives more narrative elements without detriment to his sense of control and freedom in the game environment.

A.3 Objectives

This project seeks to demonstrate that there is an improvement in the quality of the narrative experience when using narrative management systems in video games. It has to be emphasized that there is no universal solution to the problem that has been proposed to solve; the proposed objectives are an approach to a possible solution.

The solution is far from being complete, since covering all the modules that make up the classic approaches would mean working in areas ranging from the tangential, such as Natural Language Processing, to the remote, such as Psychology, which is necessary in the generation of player profiles.

Thus, the following objectives are defined:

- Designing and implementing a drama manager, the main core of the project, which must be able to guide the player to the end of a story, using certain elements of procedural narrative.
- Designing a form of logical representation of stories that allows the DM system to detect, process and react consistently and correctly to the player's actions.
- Implementing different search algorithms to compare the efficiency of various approaches to DM system design.
- Designing a reasonably simple and controlled environment to evaluate the correct performance of the DM, integrating both modules.
- Carrying out the necessary experimental tests to verify the initial hypothesis with the developed prototype.
- Testing the usefulness of procedural narrative elements in this same environment, as well as elucidate how they affect the player's perceived agency in contrast to the selection and application of guidelines proposed by the drama management system.

A.4 Methodology

The first months of work will be employed on determining which development tool will be chosen for the implementation of the project, both for the AI and the videogames needed to test it. The two most attractive tools for this research are the Unity engine and the videogame *Minecraft* (Mojang, 2011). Later, in **CAPÍTULO II**: TRABAJO PREVIO, other tools will be presented, while in **CAPÍTULO IV**: IMPLEMENTACIÓN, the advantages and disadvantages of each of them will be discussed, and the final choice will be presented.

Later, the development of the Drama Management system will begin, making use of the concepts already learned through an initial documentation process. To test the system, a first prototype will be developed. It will consist of a simple videogame that will allow to check if everything works as expected and, in case this is not the case, to fix possible errors of the manager.

Once the correct functioning of the system has been determined, the next objective will be the implementation of a second prototype. It will be a much more complete videogame than the first one, which will serve as an example of the use of the Drama Manager on a more conventional context, showing that the use of the system is possible in more complex environments.

When the final prototype is finished, an experiment will be carried out. This experiment will consist of having several people play two different versions, one with the system integrated and one without. Later, they will have to answer a series of questions that will be analyzed once the experiment is finished in order to draw conclusions.

A.4.1 Research Tools

In addition to the tools used for the implementation, other tools will be used for the research and its coordination. GitHub will be used so that both the tutors and the researchers of this project have always access to the project. Periodically, the content of the repository will be updated with the latest version of the project.

This repository will be private, giving access only to the people involved in the research.

For the writing of the report, as it is a joint work, Microsoft Office with online functions will be used. These functions allow simultaneous writing by different users. They also allow the project tutors to be able to observe the new changes in the report at all times, as well as adding comments on possible errors.

Finally, for the final experiment, Google Forms will be used. This tool allows to create and analyze the results of questionnaires in a simple way. This will be explained in more detail in **CAPÍTULO V**: EXPERIMENT.

A.4.2 Planning

After setting out the methodology, this section contains a table showing the month-by-month planning of the project. The research began in July 2020 and ended in February 2021.

Period	Planned Tasks
July 2020	Documentation process: bibliography search on interactive and procedural narrative. Beginning of the writing process of the report: introduction chapters and previous work.
August 2020	Documentation process: bibliography search on narrative managers and narrative videogames. Final decision on the development tools to be used in the project.
September 2020	Preliminary implementation of the drama manager. Integration by text. Writing of the report: first approach to technical discussion and implementation.
October 2020	Complete implementation of the first prototype of the drama manager. Implementation of the first videogame to test the prototype.
November 2020	Beginning of the development of the improved drama manager. Design of the second videogame and search of assets for its development. Writing of the report: expansion.
December 2020	Complete implementation of the final prototype of the drama manager. Complete integration of the AI and game modules.
January 2021	Design and conduct of the experiment. Analysis of the results. Writing of the report: experiment and conclusions.
February 2021	Completion of the report.

Table 2: Month-by-month planning of the project development

A.5 Document Structure

After a first chapter of introduction to the subject of the project and prospecting, this document is oriented in its second chapter to make an exhaustive review of all the background of the project that have been considered of interest to contextualize the development.

Thus, we will briefly review the history of interactive narrative, emphasizing those milestones that have been significant in making certain design decisions. Then, we will review some of the drama management systems that have been developed in recent years, as well as some of the videogames that have tried to bring advances and novelties to the field of narrative. The chapter will end by briefly explaining the tools that will be used to develop the prototypes of this research.

The third chapter will provide a technical discussion on the design of the drama manager prototype proposed by this project, which will be complemented in the fourth chapter with a detailed explanation of its implementation. This chapter will also cover the implementation of the games that will be used to develop the relevant experiments.

The fifth chapter of this document will be devoted to cover the results of the experiments with the system integrated in a videogame. After a sixth chapter dedicated to the discussion of these results, the seventh chapter will present the conclusions and possible future work.

The document closes with the Bibliography section, after which both the introduction and the conclusions can be consulted in English. The Annexes provide documentation related to the design of the video game. Finally, the contributions of each member to the project will be glossed.

ANEXO B: CONCLUSIONS AND FUTURE WORK (TRANSLATED TO ENGLISH)

A drama management system has been developed with a plot-driven design approach, which can be used in any narrative environment proposed by the user, as long as the design work is adapted to the author's needs. This, however, can be hard work for the user of the system, who may not always appreciate this addition to his project.

With the Drama Manager, a powerful tool is made available to any game developer. This tool can be easily implemented in any project using C# as the basic programming language, even if it was designed with the Unity engine in mind.

Implementing such a system in one of today's most popular game development tools like Unity opens the door to future advances in the field of interactive and procedural storytelling.

The system, however, is not complete, nor does it include all the advances in design and implementation that have been made over the years, but it is a solid cornerstone for future work, which will be covered in the last subsection of this chapter. First, some reflections and conclusions about the project.

B.1 Conclusions

The project seeks to provide evidence that there is an improvement in the quality of the narrative experience when using drama management systems in videogames. Based on the results of the experiments, there are indications that, indeed, these systems can improve the user experience without significantly affecting the user's sense of freedom.

However, the effort involved in integrating the DM into a game, at the level of design and authoring, can make the use of the system a challenge for developers who are not very experienced in narrative video games. On the other hand, once mastered, the system is powerful enough to allow a wide range of possibilities to the user.

The drama management system that has been employed is capable of guiding the player through a dramatic arc without great difficulty; again, it is up to the designer to facilitate this process to a greater or lesser extent, but the system accomplishes this task. One cannot lose sight of the fact that the player, first and foremost, will hold on to his freedom, and in complex stories this task of guidance can be much more complicated.

Versatility is, as proposed, one of the main virtues of the system. That is why it integrates different search algorithms. The Drama Manager has been achieved as a system with complete heuristics, although there is still a long way to go if greater optimality is sought. The user can define which algorithms to use, and even change them at runtime.

The Drama Manager also uses a knowledge representation language based on actions and states designed by the user, so it can become as complex or as simple as required.

This has been possible, to a great extent, thanks to the design of the virtual environment of the videogame itself, as well as its mechanics. Developing the game

with the DM's needs in mind has been key to facilitating communication between the two modules. Obviously, not every game can be expected to be so easily adaptable, but once again it would all be up to the user. Nevertheless, it seems logical that, if someone were to make use of the implemented system and consider integrating it into a videogame with a strong narrative component, they would be expected to design with these considerations in mind.

The designed environment and the selected tools made it possible to develop both prototypes with ease, so it is worth highlighting them as key elements in the rapid development of narrative video games. Likewise, thanks to Unity, it was possible to easily assemble a functional build on a multitude of computers, which allowed the game and the questionnaire to be distributed efficiently and without major problems.

Even so, many questions remain about the integration of this type of system in video games that are not purely narrative. Players seem satisfied, but it would be necessary to go a step further to achieve true narrative procedurality. Achieving a comprehensive implementation of a drama manager in a video game depends on considerable effort by the designer, as well as the incorporation of other modules that expand its capabilities.

B.2 Future Work

Currently, drama manager design proposals can be found with modules beyond those that have been implemented. For example, there are systems aimed at modeling the player's personality, whose objective is to offer narrative arcs that are more likely to be liked by that person.

New algorithms could be added in the problem-solving module, as well as sophistications to the state-space search mechanism. Ideally, in cases where the search does not yield results, it would be interesting to include a story generator. Thus, if the player reached a point where the DM was unable to find a path to the

intended plot point, new content could be generated to open the possibility of completing the story.

Similarly, the knowledge representation could become much more complex and allow for a greater generalization of actions. This would favor such procedural history generation, although it would raise other doubts. Parallel to this procedural story generation, the generation subsystem should be able to produce textures, terrains, characters, music, and any kind of digital data needed to represent the virtual environment of a videogame.

Going one step further, this procedural generation could even lead to a change of mechanics and dynamics in the game itself. However, it remains to be seen whether a video game whose content is entirely procedural can be commercially viable, although its academic value would be unquestionable.

ANEXO C: APORTACIONES

En el siguiente apartado, se exponen detalladamente las aportaciones de cada miembro del equipo de investigación a este proyecto.

Néstor Cabrero Martín

Durante los primeros días de la investigación mi principal ocupación fue la de buscar bibliografía. Por una parte, me centré en buscar artículos académicos que sirviesen para aportar una base teórica fuerte a la memoria. Por otra, traté de encontrar material que nos pudiese servir de guía a la hora de implementar el Drama Manager.

Tras consultar múltiples fuentes, llegué a la conclusión, junto a mis compañeros, de que el sistema se sustentaba en la búsqueda en el espacio de estados. Así fue como se me ocurrió plantear un primer acercamiento al proyecto puramente experimental: escribí una pequeña historia, formalizada en personajes, objetos, acciones y lugares. La idea era que mis compañeros escribiesen una historia a partir de un estado base que yo proporcionaba: era una forma de comprobar cómo se generaba el espacio de estados, y pronto nos dimos cuenta de la potencia que podía alcanzar el sistema.

La primera idea que tuvimos al percatarnos del reto al que nos enfrentábamos fue utilizar el código de AIMA, pero había un problema: sólo estaba escrito en Java. Así fue como *Minecraft* se puso sobre la mesa, y durante el siguiente mes yo y mis compañeros pasamos horas buceando en tutoriales de *modding* para, finalmente, llegar a la conclusión de que avanzaríamos mucho más rápido utilizando Unity.

Siguiendo con mi asignación original de encontrar una librería adecuada que nos facilitase el trabajo de implementación, busqué diferentes opciones para el nuevo lenguaje de programación escogido, C#. Como ninguna de las opciones se ajustaba plenamente a nuestras necesidades, decidí desarrollar un módulo propio inspirado en varias librerías de algoritmos y en el propio modelo de AIMA, lo que con el tiempo evolucionaría hasta convertirse en nuestro Resolutor.

Tras esta implementación en Unity del primer prototipo del Drama Manager, comencé a escribir una discusión técnica que acabaría convirtiéndose en el grueso del capítulo 3 y el subapartado dedicado al módulo de Inteligencia Artificial del capítulo 4.

Poco después nos enfrentamos al reto de ampliar las capacidades del Drama Manager. Por un lado, mi principal ocupación en aquellos días fue buscar una herramienta que nos permitiese desarrollar rápidamente el prototipo de un videojuego en Unity, y acabé dando con TopDown Engine.

Mientras mis compañeros aprendían a manejarse con el *framework*, yo seguí ampliando las funcionalidades del Drama Manager. El resultado de esas semanas de trabajo fue el módulo de IA descrito en este documento. La necesidad de flexibilidad vino, en gran medida, por trabajar en este módulo de forma independiente al desarrollo del videojuego *Tell No Tales*.

Con todo el sistema funcionando, trabajé junto a Marcos, primero, en la integración de los módulos de IA y de juego. Después, junto a Aaron y Pablo, coordiné la implementación de algunas de las funciones de comunicación entre el juego y el Drama Manager.

En los últimos compases del proyecto, he participado en los experimentos aportando sujetos de pruebas para ambos cuestionarios. También he trabajado en algunas reescrituras de este documento (especialmente en los capítulos de Introducción, Trabajo Previo y Conclusiones), así como en su maquetación y edición final.

Marcos García García

Como se ha mencionado anteriormente en la metodología, al comienzo del proyecto nuestro objetivo era determinar la herramienta de desarrollo más adecuada para la investigación. Por ello, mi trabajo en esos primeros días consistió de aprender la API de *modding* de *Minecraft*. Haciendo uso de información encontrada en internet desarrollé un *mod* muy básico antes de que se tomase la decisión de usar Unity como herramienta definitiva, consiguiendo introducir un bloque y un objeto nuevos al videojuego. Mis demás compañeros también consiguieron realizar un *mod* similar, y esto nos ayudó a acercarnos a la conclusión de que nos resultaría más eficiente usar Unity en vez de *Minecraft*.

Una vez en Unity, mi trabajo consistió en la implementación del videojuego *Tell No Tales*. Partiendo de una escena base realizada por mi compañero Pablo con un jugador ya funcional y un personaje con el que se podía hablar, construí el entorno entero del nuevo videojuego e implementé los comportamientos requeridos para su correcto funcionamiento.

Haciendo uso de la herramienta de *Tile Palette* de Unity y de los *sprites* de *TopDown Engine*, dibujé las cuatro islas que aparecen en *Tell No Tales* en el editor de Unity. Tras ello, implementé el comportamiento del barco del muelle, que permite al jugador transportarse entre las islas (el componente *MapSelector*). En un principio, cada isla estaba representada en una escena diferente de Unity, por lo que tuve que hacer uso de la función *DontDestroyOnLoad* del motor para el viaje entre las distintas islas. Finalmente se decidió que todas las islas estarían en una sola escena para una más cómoda implementación del videojuego. Además, para la construcción del entorno, tuve que dibujar una serie de *sprites* nuevos, como el del barco, los carteles, la cama, etcétera.

Una vez terminado el entorno, implementé el comportamiento de los objetos de inventario necesarios (excepto el de la llave y la puerta que abre, que ya estaban implementados en la escena base), así como de los personajes que se los otorgan al jugador, siguiendo el diseño sugerido por mis compañeros. También implementé otros comportamientos de diversos elementos en el entorno, como el barril que

bloquea la salida de la mazmorra, la palmera que suelta una nota al ser agitada, el personaje borracho que te da ron si estás cansado, etcétera. Además, dibujé los *sprites* de todos los objetos de inventario y los personajes (modificando al personaje amarillo del protagonista, ofrecido por *TopDown Engine*).

Tras varios ajustes sugeridos, terminé de implementar el juego, funcional de principio a fin sin hacer uso de un gestor de narrativa. Tras esto, mi objetivo era la implementación de las funciones del *WorldManager* que se usarían en la función de evaluación y función de generación del gestor. Tras terminar las funciones para la evaluación, implementé el componente llamado *NPCDialogueManager*, con el cual pueden añadirse paquetes de diálogos (un paquete contiene todos los diálogos de todos los personajes de la escena) y cambiar fácilmente entre ellos. Más tarde, también implementé el comportamiento de un personaje enfadado que aparece si el jugador cava en la zona equivocada muchas veces. Además, implementé que pudiesen aparecer botellas de ron en la costa y que las descripciones de las islas en el selector del muelle pudiesen cambiar. Con todo esto, desde la función de generación del gestor se puede ver cuál es la acción que el jugador debería realizar a continuación y llamar a la función correspondiente del *WorldManager* para generar el contenido adecuado que ayude al jugador a progresar.

Finalmente, realicé el arte de los menús para *Tell No Tales*, y me aseguré de que se podía acceder a los cuestionarios correspondientes desde el menú a través de la pulsación de un botón.

Una vez habiéndome corregido errores y habiendo terminado *Tell No Tales*, realicé el instalador de las dos versiones del videojuego (con y sin DM) para poder repartirlo fácilmente para la realización del experimento. Como el resto de mis compañeros, estuve presente en los experimentos asegurándome de que todo iba como es debido y aporté en la escritura de esta memoria durante todo el desarrollo de la investigación. En concreto, por ejemplo, escribí toda la sección de implementación de los videojuegos desarrollados para este proyecto en el **CAPÍTULO IV**: IMPLEMENTACIÓN, debido a mi amplio trabajo en esta parte de la investigación.

Pablo Martín García

En un inicio se consideró usar Minecraft como motor para implementar el proyecto, así que investigué y aprendí las cosas básicas del API para elaborar mods. Finalmente, y entre otras cosas, por nuestro grado general de experiencia y familiaridad con Unity y C#, decidimos que éste sería el motor elegido para el proyecto.

A continuación, investigué acerca de cómo usar las funcionalidades básicas del asset *More Mountains*, que anteriormente había incluido mi compañero Néstor en el proyecto de Unity, creando así los *prefabs* del *Player* y de su inventario, ambos simplificados y adaptados a las necesidades de nuestro juego. Seguidamente creé la escena inicial de la mazmorra con dichos *prefabs* y la mecánica de abrir una puerta con una llave determinada. Esta escena es la que serviría de base para probar las primeras mecánicas del juego y las primeras versiones de la implementación del Drama Manager, así como todo el desarrollo artístico del juego.

Más adelante, trabajé junto a Aaron para desarrollar la historia que finalmente se contaría en el juego, la idea es que no fuese excesivamente compleja, ya que no era el núcleo de la investigación ni del experimento que llevaríamos a cabo. En este proceso también creamos el guión del juego definiendo cuáles serían los puntos clave (*plot points*) de la historia, los cuales estaban relacionados con elementos interactivos como la obtención de notas, palas o llaves. El siguiente paso fue crear los diferentes diálogos que tendrían todos los personajes del juego cuando el jugador interactuase con ellos. Este proceso llevó varias iteraciones, ya que los diálogos de cada personaje debían cambiar en función del estado del juego en cada momento, cubriendo todos los *plot points* de la historia y siendo coherentes.

Una vez terminado el diseño del juego y la historia empecé a diseñar junto con Aaron las preguntas del cuestionario. Dichas preguntas tenían como principal finalidad extraer datos que nos permitiesen responder a la hipótesis descrita en el apartado 1.2 Hipótesis. Tras varias iteraciones y aplicando las correcciones de nuestros tutores, finalizamos el cuestionario y lo llevamos a Google Forms.

También trabajé en algunos detalles relacionados con la implementación del juego. Implementé los diferentes menús (menú principal, opciones y créditos), a los que luego mi compañero Marc añadió un arte coherente con el resto del juego, y la logística necesaria para poder navegar entre diferentes escenas. De la misma manera desarrollé e implementé un sistema de audio que permitía reproducir música y sonidos durante el juego. También fui el encargado de buscar dicha música y sonidos, todos sin licencia de copyright. Además, junto con mi compañero Aaron, establecimos los diferentes costes de cada acción que podía realizar el jugador en función del estado en el que el juego se encontrase.

Una vez listos los ejecutables del juego y los cuestionarios empezamos a buscar participantes para realizar las pruebas. Como el resto de mis compañeros, estuve de forma virtual y presencial (en los casos en los que fue posible) durante la realización del experimento para resolver cualquier error o duda que los participantes pudiesen tener.

Una vez terminado el periodo de experimentación y tras un breve proceso de investigación para saber cómo organizar y analizar toda la información relativa al experimento, opté por mostrar todos los datos y el trabajo realizado como actualmente se encuentra en el apartado CAPÍTULO V: EXPERIMENT. Con esta guía ya definida y tras un primer y rápido análisis de los datos obtenidos que hice junto con Aaron, establecimos cuáles serían los resultados en los que nos centraríamos y que nos servirían para extraer las conclusiones del experimento.

Acto seguido, empecé a extraer todos los resultados, tanto demográficos como los de la experiencia y sensación de libertad a la par que mi compañero Aaron se encargaba de realizar las gráficas para su posterior análisis. Dichos datos están presentes en esta memoria.

Una vez hecho esto, llevé a cabo junto con Aaron el análisis de los resultados obtenidos, tanto gráficos como numéricos, y que han quedado escritos en esta memoria. De igual manera, como principales encargados de la parte de

experimentación del proyecto, elaboré junto con Aaron las conclusiones finales que estos resultados nos permitieron extraer.

Junto con el resto de mis compañeros he trabajado en esta memoria durante todo el proceso de investigación.

Aaron Reboredo Vázquez

Si he de organizar de manera cronológica las aportaciones que he hecho al proyecto debería empezar por el principio: la investigación. En un primer momento y cuando todavía apenas habíamos definido el futuro de nuestra implementación y nuestro experimento, realicé, en conjunto con mis compañeros, una labor de investigación bastante amplia. Esta abarcó temas que van desde la inteligencia artificial hasta la interactividad en los videojuegos, pasando por una revisita a ciertos juegos que hemos referenciado como parte de la historia de la narrativa interactiva y que aparecen en a lo largo de esta memoria, más concretamente en los apartados de 2.1 Breve Historia de la Narrativa Interactiva.

De la misma manera y como el resto de mis compañeros estudié y di mis primeros pasos en el Modding de Minecraft con Java, que se planteaba como un fuerte candidato para la implementación de nuestro Drama Mánager, cuando estábamos en la búsqueda de una herramienta sobre la que desarrollar nuestro videojuego. Esto, como ha quedado reflejado en líneas de esta memoria, fue trabajo que nos quedamos como aprendizaje para el presente y píldora de conocimiento para el futuro. Sin embargo, cesamos nuestro desarrollo en Minecraft en aras de la viabilidad para nuestro proyecto para el que finalmente utilizamos Unity.

En los primeros compases del proyecto desarrollé, a grandes rasgos, la historia que posteriormente serviría como contexto para el experimento y el juego sobre el que implementaríamos nuestro sistema. Sin embargo, y tras hacer un balance y un trabajo de producción, nos dimos cuenta de que no cumplía con lo que buscábamos para desarrollar el experimento con el que nos planteábamos analizar la implementación de nuestro Drama Mánager y junto con Pablo comenzamos a

desarrollar la historia definitiva que, ahora sí, vestiría el diseño del videojuego que daría contexto a nuestro experimento.

Una de nuestras necesidades a la hora de desarrollar el videojuego que haría gala de nuestro Drama Mánager fue la de desarrollar un sistema de diálogos. Yo mismo me encargué de implementarlo, a raíz de lo cual, nos permitió mostrar en pantalla y durante el juego los textos como resultado de interactuar con los objetos o hablar con los diferentes personajes. Este sistema fue esencial para que el jugador entendiese y siguiese la historia. Así mismo, el sistema debía convivir con el resto de los desarrollados por mis compañeros y funcionar para con las funciones generadoras, entre otras. Por ello, lo tuve que desarrollar con la vista puesta en el futuro del sistema completo.

De la misma manera y referente al Drama Manager, junto con Pablo, establecimos los diferentes costes de cada acción que podía realizar el jugador en función del estado en el que el juego se encontrase y que es parte de la implementación del mismo.

A posteriori desarrollé junto con mi compañero Pablo todo lo referente al guion y diseño del juego. Diseñamos todas las mecánicas, número de personajes y de islas, diálogos de los diferentes personajes para cada uno de los posibles estados del juego, número y variedad de objetos, notas, consumibles y personajes clave para la historia, etc. También todas las acciones y sucesión de circunstancias que dan forma al ciclo del juego.

De la misma manera, Pablo y yo, como encargados del diseño del juego que haría gala del Drama Manager que estábamos implementando, nos encargamos de todo el diseño, decisiones y logística del trabajo experimental. Diseñamos las preguntas y desarrollamos el formulario y modo de actuación para la realización del experimento. Buscábamos un cuestionario que nos permitiese recaudar información suficiente, valiosa y referente a los objetivos del proyecto. Tras varias interacciones y modificaciones del cuestionario desarrollamos el cuestionario final

que es el que los jugadores que probaron nuestra demostración tuvieron que rellenar al final.

Este experimento es el que nos ha permitido recoger toda la información y los datos necesarios sobre los que se ha desarrollado el análisis y discusión de resultados de la memoria.

Así mismo, me he encargado de desarrollar todos los recursos gráficos que dan soporte al análisis de los datos y la discusión de los resultados. Desde los Box-Plots hasta las gráficas de tendencias, pasando por las gráficas comparativas. De igual manera me encargué del análisis de datos de manera numérica para poder sacar los porcentajes sobre jugadores experimentados y no experimentados que también dan soporte a los datos y los apartados de discusión.

Final y nuevamente con mi compañero Pablo y como principales responsables del desarrollo del experimento, fuimos los encargados de llevar a cabo el análisis al completo y en profundidad de los resultados, gráficos y porcentajes que ya he mencionado anteriormente y que ha quedado por escrito en esta memoria.

Junto con el resto de mis compañeros, he dado forma durante toda la investigación a estas líneas que conforman la memoria y que ahora, ya forman parte de mi vida.