Juegos Serios para Promover el Pensamiento Computacional y la Programación

Serious Games to Promote Computational Thinking and Coding

Trabajo de Fin de Grado
Curso 2020-2021

Autores

Dany Faouaz Santillana
Arturo García Cárdenas
Álvaro Poyatos Morate

Director

Prof. Dr. Baltasar Fernández Manjón

Codirector

Dr. Antonio Calvo Morata

Grado en Desarrollo de Videojuegos
Facultad de Informática
Universidad Complutense de Madrid

Agradecimientos

A todas las personas que, de alguna manera, han apoyado y han facilitado la realización de este trabajo. A todos los docentes que forman parte del grado, por hacer que consigamos las competencias necesarias para enfrentarnos al futuro.

En especial al director del TFG, a Balta, por apoyarnos desde el primer momento, confiar en nosotros e implicarse para realizar con éxito el proyecto.

A todo el equipo de investigación de e-UCM, en especial a Toni, codirector del TFG, por darnos feedback constante y ayudarnos a pulir al detalle la memoria.

A Victor Manuel Pérez Colado, por su apoyo técnico en el apartado de las analíticas y el soporte del motor uAdventure.

A nuestros amigos, en especial al equipo *SpicySeed*, por darnos tantos buenos momentos en los momentos más difíciles. A todos nuestros compañeros de clase, por apoyarnos y compartir la experiencia universitaria a nuestro lado.

A la asociación sin ánimo de lucro Programamos.es y a su cofundador Jesús Moreno León, por brindarnos apoyo y motivarnos. Especial agradecimiento a los expertos Henar Lanchas López, Gaspar Ferrer Soria, Marcos Román González y Ángel Serrano Laguna; por interesarse y ayudarnos a mejorar los resultados. Por aguantar a Balta y sus correos constantes.

Gracias a los más de 45 profesores e investigadores que han jugado a nuestros dos juegos y nos han proporcionado un feedback detallado que nos ha permitido mejorar tanto los juegos como su aplicabilidad en la docencia.

Muchas gracias a Ivan Turegano, director y a Ana García Pulio, coordinadora del Colegio La Inmaculada Escolapias Puerta de Hierro por confiar en nosotros y darnos el privilegio de que realicemos pruebas con alumnos en un entorno real. Sabemos que aguantar la presión de las peticiones de Balta es complejo.

A la cátedra Telefónica-Complutense en educación digital y juegos serios, por apoyar el proyecto, sobre todo en los aspectos experimentales.

A nuestras familias, en especial a nuestros padres y hermanos, por apoyarnos y confiar en nosotros.

Resumen

Cada vez es más importante el uso de la tecnología y la ciencia en cualquier ámbito, es por ello por lo que existe un creciente interés en el sector de la educación por enseñar conceptos relacionados con las Ciencias de la Computación desde temprana edad. Dos aspectos que destacan son el aprendizaje de la programación o de lenguajes informáticos, y el Pensamiento Computacional (PC), una forma de afrontar la resolución de problemas utilizando habilidades propias de la computación. Personalidades como Jeannette Wing (2008), una investigadora muy reconocida en el campo del PC, creen que el pensamiento computacional será una habilidad fundamental para todos a mediados del siglo XXI. Aunque comúnmente se suele relacionar el PC con campos más tecnológicos, el PC también puede ser muy útil en otros campos como la medicina o el periodismo.

Por otro lado, existe un creciente interés en introducir los videojuegos en distintos ámbitos ya que se han demostrado los beneficios y la eficacia que tienen como herramienta educativa (e.g., entrenamiento de habilidades, concienciación, etc). Por ello, se han empezado a desarrollar los denominados juegos serios en diversos campos como la salud, medioambiente o la educación. Los juegos serios son un mercado en crecimiento. La cantidad de juegos serios ha incrementado en los últimos años y gracias a la creciente demanda de nuevas herramientas interactivas muchas empresas comienzan a tener interés por desarrollar juegos serios.

El objetivo principal de este proyecto es proponer y utilizar juegos serios para promover el aprendizaje de conceptos de pensamiento computacional y de programación de manera guiada y motivadora. Para llevar a cabo este objetivo se ha seguido el siguiente proceso: primero se ha realizado un análisis de la situación actual del PC en la educación y del uso en el aula de juegos serios que abordan este concepto. Posteriormente se han analizado un conjunto de juegos y/o aplicaciones representativas para identificar sus principales características. Con las conclusiones del análisis se ha desarrollado un primer juego orientado al PC como prueba de concepto y se han realizado pruebas de validación con profesores (incluyendo entrevistas). Estas pruebas han servido para comprobar la aplicabilidad del juego. Aprovechando la experiencia del primer juego desarrollado se ha creado un segundo juego orientado a promover la programación, que es el principal resultado de este trabajo. Se han realizado pruebas con expertos, con profesores y con alumnos a partir de las cuales se ha mejorado y validado los juegos.

Palabras Clave

Juegos Serios, Pensamiento Computacional, Programación, Ciencias de la Computación.

Repositorios de código abierto de los juegos

La Mansión Paranormal - https://github.com/WeArePawns/ParanormalMansion

Articoding - https://github.com/WeArePawns/Articoding

Abstract

The use of technology and science in any field is becoming increasingly important, which is why there is a growing interest in the education sector in teaching concepts related to Computer Science from an early age. Two aspects that stand out are the learning of programming or computer languages, and Computational Thinking (CT), a way of approaching problem solving using computing skills. Personalities such as Jeannette Wing (2008), a well-known researcher in the field of CT, believe that computational thinking will be a fundamental skill for everyone by the middle of the 21st century. Although the CT is commonly associated with more technological fields, the CT can also be very useful in other fields such as medicine or journalism.

On the other hand, there is a growing interest in introducing video games in different fields as they have demonstrated their benefits and effectiveness as an educational tool (e.g., skills training, awareness, etc.). Therefore, so-called serious games have started to be developed in various fields such as health, environment or education. Serious games are a growing market. The number of serious games has increased in the last years and thanks to the growing demand for new interactive tools many companies are starting to be interested in developing serious games.

The main objective of this project is to propose and use serious games to promote the learning of computational thinking and programming concepts in a guided and motivating way. In order to achieve this objective, the following process has been followed: first, an analysis of the current situation of CT in education and the use of serious games that address this concept in the classroom has been carried out. Subsequently, a set of representative games and/or applications have been analyzed to identify their main characteristics. With the conclusions of the analysis, a first CT-oriented game has been developed as a proof of concept and validation tests have been carried out with teachers (including interviews). These tests have served to check the applicability of the game. Taking advantage of the experience of the first game developed, a second game oriented to promote programming has been created, which is the main result of this work. Tests have been carried out with experts, teachers and students from which the final game has been improved and the game has been validated.

Keywords

Serious Games, Computational Thinking, Programming, Computational Science.

Open source game repositories

La Mansión Paranormal - https://github.com/WeArePawns/ParanormalMansion

Articoding - https://github.com/WeArePawns/Articoding

Índice

Índice de figuras

Capítulo 1

Introducción

"La educación científica de los jóvenes es al menos tan importante, quizá incluso más, que la propia investigación." - Glenn Theodore Seaborg

1.1. Motivación

Nuestra sociedad ha vivido un gran avance tecnológico en los últimos años. Este avance también se refleja en el ámbito educativo, donde cada vez es mayor el interés por usar los videojuegos y la gamificación como herramientas de aprendizaje. Además, cada vez más, en los colegios se están introduciendo asignaturas de programación.

La situación vivida por la pandemia del COVID-19 ha puesto de manifiesto la necesidad y utilidad de la tecnología en ámbitos muy diferentes, que ha permitido realizar tareas del día a día como trabajar, dar clases, solicitar citas y hacer gestiones, de manera telemática. En este aspecto, la programación y el pensamiento computacional son la base de habilidades relacionadas con la tecnología, lo que pone de manifiesto la necesidad de crear herramientas para facilitar su aprendizaje.

En estas circunstancias el objetivo de este trabajo es promover la programación y el pensamiento computacional mediante el uso de videojuegos.

1.1.1. ¿Qué es el Pensamiento Computacional (PC)?

El Pensamiento Computacional es una manera de afrontar la resolución de problemas (Wing, 2008). En la literatura que hay respecto al PC se habla de los distintos elementos que lo componen. Analizando diferentes artículos referentes al PC se puede observar una amplia variedad de definiciones y, sobre todo, de características principales.

Jeannette Wing, considera que el PC se compone de cuatro aspectos fundamentales: reformulación de problemas, recursión, descomposición, abstracción y testeo (Wing, 2006).

Investigadores como los pertenecientes a la Computational Thinking Task Force creada por la CSTA (Computer Science Teacher Association) destaca nueve habilidades como la base del PC: recolección de datos, análisis de datos, representación de datos, descomposición, abstracción, algoritmos y automatización, paralelización y simulación (Barr & Stephenson, 2011).

> *"[...] decomposing problems, abstracting essential information, generalizing the solution for different problems, creating algorithms, and automating procedures."* (Shute et al., 2017, p. 6)

A pesar de esta amplia variedad de habilidades, varios estudios señalan que el PC es un proceso cognitivo que involucra las habilidades de descomposición, búsqueda de patrones o generalización, abstracción, diseño de algoritmos y depuración o evaluación (Angeli et al., 2016) (Selby, 2013). Además, la investigación de (Araujo et al., 2019) remarca que estas son las habilidades más citadas en la mayoría de los estudios. Este trabajo parte de estas 5 características como marco de referencia para definir las habilidades a desarrollar a lo largo del proceso de aprendizaje:

- **Descomposición**: Capacidad de convertir problemas complejos en problemas más sencillos. La descomposición permite descubrir los pasos necesarios para resolver un problema.
- **Identificación de patrones**: Analizar problemas de manera eficiente e identificar casos recurrentes.
- **Abstracción**: Extraer cualidades comunes de un conjunto de objetos.
- **Diseño de Algoritmos**: Desarrollar la capacidad de crear algoritmos, es decir conjunto de operaciones detalladas que permite la resolución de problemas.
- **Depuración**: Capacidad para analizar una solución y encontrar posibles errores en el diseño del algoritmo.

1.1.2. ¿Qué son los juegos serios?

Como mencionó Smeddinck (2016) citando a Abt, se puede afirmar que los juegos serios son aquellos que tienen un propósito educativo explícito y que no están pensados para ser jugados sólo por diversión.

Esto significa que, además de entretener, tienen un objetivo normalmente relacionado con la educación y la comprensión de conceptos. Aunque también pueden utilizarse como medio de comunicación, con fines comerciales o de denuncia social[1].

[1] Fuente: http://www.exelweiss.com/blog/356/serious-games-juegos-serios/

Los juegos serios de tipo educativo suelen estar orientados a un público joven, esto se debe a que prefieren un producto motivante con características propias de los juegos: progresión, aumento de la dificultad, desafío, libertad, etc.

En un estudio realizado en 2014 se reveló que el 62% de los profesores han usado juegos en sus aulas para estimular y complementar el aprendizaje (Takeuchi & Vaala, 2014).

Se pueden encontrar numerosos juegos serios gratuitos en páginas web colaborativas como GamesForChange[2] o PurposeGames[3]. En el Anexo A se dispone de una tabla con la taxonomía de los juegos serios.

1.2. Objetivos

El objetivo principal de este proyecto es proponer y utilizar juegos serios para promover el desarrollo de aptitudes del pensamiento computacional y del aprendizaje de conceptos de programación.

Para conseguir este objetivo, se ha dividido en objetivos más pequeños y alcanzables a corto plazo. Se han propuesto los siguientes objetivos:

- Analizar la situación actual del PC viendo la forma en la que se imparte, cómo puede ser beneficioso para el alumnado y de qué manera los juegos serios pueden ayudar en su desarrollo.
- Desarrollo de un juego serio narrativo para desarrollar el PC mediante la resolución de puzles (La Mansión Paranormal[4]).
- Creación de un juego para el desarrollo del PC y la enseñanza de la programación (Articoding[5]).
- Realizar pruebas de usuario con profesores y expertos interesados con la finalidad de mejorar los juegos y, en general, la experiencia de usuario (UX). Además de comprobar la aplicabilidad de los juegos en el ámbito educativo con alumnos reales y su utilidad como herramientas de aprendizaje .

1.3. Plan de trabajo

Para poder llevar a cabo el desarrollo del trabajo, se ha hecho uso de la metodología ágil de producción conocida como SCRUM. SCRUM es una metodología ágil de producción definida en los años 80 por Nonaka y Takeuchi

[2] http://www.gamesforchange.org/
[3] https://www.purposegames.com/es/
[4] https://github.com/WeArePawns/ParanormalMansion
[5] https://github.com/WeArePawns/Articoding

(Hidalgo, 2019). La principal base de SCRUM es el desarrollo iterativo del producto en pequeños o medianos periodos de producción conocidos como sprints.

> *"The scrum methodology facilitates the coordinated activity of programmers who break their work into small tasks that can be completed within fixed dura- tion cycles or "sprints", tracking progress and re-planning in regular meetings in order to develop products incrementally"* (Hidalgo, 2019, p. 5)

En SCRUM existen tres roles principales, el Scrum master, el Product owner y el equipo de desarrolladores. Al inicio de cada sprint el equipo acuerda una serie de tareas a realizar durante el sprint, durante el sprint se realizan reuniones diarias para comprobar el progreso del sprint. Al final del sprint el equipo se reúne para comentar las tareas realizadas durante el sprint y posibles mejoras (Marcal et al., 2007, p. 3). Durante este proceso, el Scrum master es el encargado de resolver conflictos y asegurarse de que el sprint transcurra adecuadamente (Hidalgo, 2019, p. 6).

Durante el desarrollo del proyecto se ha aplicado el modelo SCRUM al equipo de desarrollo. De tal manera que el director y codirector del proyecto actuaban como product owners y los miembros del equipo actuaban como desarrolladores y uno de ellos como Scrum Master.

Para alcanzar los objetivos propuestos anteriormente y organizar mejor el trabajo, se ha propuesto una planificación dividida en tres periodos de tiempo bien diferenciados. Estos periodos están descritos en las siguientes tres secciones.

1.3.1. Investigación del tema (septiembre - noviembre)

Con el objetivo de obtener el conocimiento necesario para llevar a cabo este trabajo y adquirir así una buena base teórica, el primer paso ha sido realizar una investigación de los conceptos relacionados con el PC y de las diferentes herramientas que se utilizan en los centros educativos. El conocimiento adquirido y las herramientas analizadas son descritas en la sección 2 de esta memoria. Los hitos completados a lo largo de este primer periodo son los siguientes:

- Realizar un análisis profundo de cuál es la situación actual del PC en la enseñanza, ver de qué manera se imparte y cómo puede ser beneficioso para el alumnado.
- Analizar cuál es el impacto de los juegos serios en la educación y cómo pueden ayudar para desarrollar el PC.

1.3.2. Desarrollo y pruebas de la Mansión Paranormal. Prototipado de Articoding (noviembre - marzo)

Esta etapa ha consistido en realizar el diseño y llevar a cabo el desarrollo de un primer juego sobre el PC, desde su primera versión hasta su versión final, realizando pruebas y mejoras durante todo su desarrollo. Los detalles de este juego se describen en el capítulo 3 de esta memoria. Además, también abarca el primer prototipo de Articoding. Durante esta fase, se ha extendido la revisión del estado del arte a través del análisis de otros juegos enfocados al PC y al aprendizaje de los conceptos básicos de programación. Los hitos a cumpliar en esta segunda etapa han sido:

- Realizar un análisis de un amplio catálogo de juegos relacionados con el PC y/o con la enseñanza.
- Diseñar un juego (La Mansión Paranormal) siendo la primera toma de contacto con el PC.
- Desarrollar un primer prototipo del juego mencionado como prueba de concepto e iterar sobre este para pulir el diseño inicial.
- Ampliar el contenido del prototipo para la realización de pruebas con profesores y alumnos.
- Reservar un periodo iterativo para probar el juego, encontrar errores y arreglarlos.
- Utilizar la retroalimentación obtenida para cerrar la versión final del juego.

1.3.3. Desarrollo de Articoding (marzo - junio)

Esta última etapa está dedicada al desarrollo del segundo juego, que tiene el objetivo de enseñar conceptos de programación. La etapa abarca desde el diseño del juego hasta su desarrollo y validación con usuarios. En las fases finales de desarrollo se realizan iteraciones rápidas de 1 a 3 días puliendo los detalles restantes para crear un producto y poder realizar pruebas con usuarios. Estas iteraciones rápidas realizadas consisten en:

- **Reunión con el tutor y cotutor.** Dónde se muestra el estado actual del juego y se debaten las posibles mejoras y correcciones. Se generan las tareas en este paso.
- **Implementación de las mejoras y corrección de errores.** Se valoran qué tareas pueden implementarse y cuáles son las más prioritarias. Se implementan dichas tareas y se descartan aquellas que se han considerado no tan importantes.

- **Envío de builds y recogida de feedback.** Se generan las distintas builds y se envían al tutor y al cotutor para que comprueben que todo está implementado correctamente. Se recibe la retroalimentación y se fija una fecha para la próxima reunión.

El diseño y desarrollo de Articoding está descrito en la Capítulo 4 de esta memoria. Además, los hitos completados en esta etapa son:

- Diseñar el juego principal (Articoding) y realizar iteraciones rápidas de prototipado para pulir dicho diseño.
- Desarrollar e implementar las ideas de diseño, teniendo pequeños periodos iterativos para cambiar decisiones de diseño.
- Reservar un periodo iterativo para probar el juego principal, encontrar errores y arreglarlos.
- Internacionalizar y localizar el juego al inglés y al español.
- Diseñar un modelo de evaluación con usuarios para la realización de pruebas de usuarios.
- Analizar los resultados obtenidos y cerrar el juego con la retroalimentación obtenida.

1.4. Estructura de la memoria

El resto de esta memoria, que describe el trabajo realizado, está estructurada de la siguiente manera:

- Capítulo 2. Expone la situación actual del PC, relacionándolo con distintos aspectos como la programación o asignaturas escolares. Además, hace un estudio de los tipos de juegos que desarrollan el PC y resume un exhaustivo análisis de algunos de los juegos que mejor promueven el PC.
- Capítulo 3. Presenta el primer juego que se desarrolló para realizar este trabajo: La Mansión Paranormal. Hace una reflexión de los resultados obtenidos y de las pruebas realizadas con usuarios.
- Capítulo 4. Plantea la evolución de la idea del segundo juego, Articoding, desde su concepción hasta su versión final. Profundiza de forma detallada en el diseño del juego, en su desarrollo e implementación. También expone algunos problemas y dificultades que se han encontrado durante el desarrollo. Para terminar, presenta las conclusiones y resultados, además de hablar sobre las pruebas con usuarios realizadas.
- Capítulo 5. Expone las conclusiones sobre el trabajo realizado, comprobando si los objetivos se han conseguido. Además, recoge ideas que no llegaron a implementarse, así como posibles mejoras que podrían desarrollarse en el futuro.

- Capítulo 6. Enumera las aportaciones realizadas por cada integrante del grupo, donde se pueden ver las contribuciones clasificadas en tres categorías: implementación, memoria, y otros.

Además se incluyen un conjunto de anexos con información complementaria y con algunos de los documentos que se han desarrollado a lo largo del presente trabajo:

- Anexo A. Muestra una tabla detallada sobre la taxonomía de los juegos serios donde se hace una clasificación del uso de los juegos serios y los campos de conocimiento.
- Anexo B. Expone una tabla donde se muestran los beneficios y aplicaciones de los juegos serios en la educación y los distintos sectores educacionales.
- Anexo C. Ofrece una tabla-resumen del análisis exhaustivo realizado en el Capítulo 2 sobre juegos serios que promueven el PC.
- Anexo D. Proporciona la guía docente del juego La Mansión Paranormal usada durante la evaluación formativa con los expertos.
- Anexo E. Proporciona la guía docente de Articoding, que ha sido usada durante las pruebas con profesores y en el colegio.
- Anexo F. Presenta de forma más detallada el trabajo realizado sobre internacionalización y localización de Articoding.

Chapter 1

Introduction

"Science education of young people is at least as important, perhaps even more important, than research itself." - Glenn Theodore Seaborg

1.1. Motivation

Our society has experienced great technological progress in recent years. This progress is also reflected in the educational field, where there is an increasing interest in using video games and gamification as learning tools. In addition, more and more programming subjects are being introduced in schools.

The situation experienced by the COVID-19 pandemic has highlighted the need and usefulness of technology in many different areas, which has made it possible to carry out day-to-day tasks such as working, teaching classes, requesting appointments and carrying out formalities telematically. In this aspect, programming and computational thinking are the basis of technology-related skills, which highlights the need to create tools to facilitate their learning.

In these circumstances, the objective of this work is to promote programming and computational thinking through the use of video games.

1.1.1. What is Computational Thinking (CT)?

Computational Thinking is a way of dealing with problem solving (Wing, 2008). In the literature on CT, the different elements that compose it are discussed. Analyzing different articles on CT, a wide variety of definitions and, above all, main characteristics can be observed.

Jeannette Wing, considers that CT is composed of four fundamental aspects: problem reformulation, recursion, decomposition, abstraction and testing (Wing, 2006).

Researchers such as those belonging to the Computational Thinking Task Force created by the CSTA (Computer Science Teacher Association) highlight nine

skills as the basis of CT: data collection, data analysis, data representation, decomposition, abstraction, algorithms and automation, parallelization and simulation (Barr & Stephenson, 2011).

> *"[...] decomposing problems, abstracting essential information, generalizing the solution for different problems, creating algorithms, and automating procedures."* (Shute et al., 2017, p. 6)

Despite this wide variety of skills, several studies point out that CT is a cognitive process that involves the skills of decomposition, pattern search or generalization, abstraction, algorithm design, and debugging or evaluation (Angeli et al., 2016) (Selby, 2013). In addition, the research by (Araujo et al., 2019) remarks that these are the most cited skills in most studies. This work takes these 5 characteristics as a frame of reference to define the skills to be developed throughout the learning process:

- **Decomposition**: Ability to convert complex problems into simpler problems. Decomposition allows discovering the steps needed to solve a problem.
- **Pattern identification**: Analyze problems efficiently and identify recurring cases.
- **Abstraction**: Extracting common qualities from a set of objects.
- **Algorithm Design**: Develop the ability to create algorithms, i.e. set of detailed operations that allow problem solving.
- **Debugging**: Ability to analyze a solution and find possible errors in the algorithm design.

1.1.2. What are Serious Games?

As mentioned by Smeddinck (2016) quoting Abt, it can be stated that serious games are those that have an explicit educational purpose and are not intended to be played just for fun.

This means that, in addition to entertaining, they have an objective usually related to education and the understanding of concepts. However, they can also be used as a means of communication, for commercial purposes or for social denunciation.

Serious games of an educational type are usually aimed at a young audience, this is because they prefer a motivating product with game-like characteristics: progression, increased difficulty, challenge, freedom, etc.

A 2014 study revealed that 62% of teachers have used games in their classrooms to stimulate and supplement learning (Takeuchi & Vaala, 2014).

Numerous free serious games can be found on collaborative websites such as GamesForChange or PurposeGames. A table with the taxonomy of serious games is available in Appendix A.

1.2. Objectives

The main objective of this project is to propose and use serious games to promote the development of computational thinking skills and the learning of programming concepts.

To achieve this goal, it has been divided into smaller and achievable short-term objectives. The following objectives have been proposed:

- Analyze the current situation of CT by researching the way it is taught, how it can be beneficial to the student body and how serious games can help in its development.
- Development of a serious narrative game to develop CT by solving puzzles (La Mansión Paranormal).
- Creating a game for CT development and teaching programming (Articoding).
- Conducting user tests with teachers and interested experts with the aim of improving the games and, in general, the user experience (UX). In addition to testing the applicability of the games in the educational environment with real students and the usefulness of the games as learning tools.

1.3. Work plan

In order to carry out the development of the work, the agile production methodology known as SCRUM has been used. SCRUM is an agile production methodology defined in the 1980s by Nonaka and Takeuchi (Hidalgo, 2019). The main basis of SCRUM is iterative product development in small to medium production periods known as sprints.

> *"The scrum methodology facilitates the coordinated activity of programmers who break their work into small tasks that can be completed within fixed dura- tion cycles or "sprints", tracking progress and re-planning in regular meetings in order to develop products incrementally"* (Hidalgo, 2019, p. 5).

In SCRUM there are three main roles, the Scrum master, the Product owner and the team of developers. At the beginning of each sprint the team agrees on a set of tasks to be performed during the sprint, during the sprint daily meetings are held

to check the progress of the sprint. At the end of the sprint the team meets to discuss the tasks performed during the sprint and possible improvements (Marcal et al., 2007, p. 3). During this process, the Scrum master is in charge of resolving conflicts and making sure that the sprint runs properly (Hidalgo, 2019, p. 6).

During the development of the project, the SCRUM model has been applied to the development team. In such a way that the project director and co-director acted as product owners and the team members acted as developers and one of them as Scrum Master.

In order to achieve the objectives proposed above and to better organize the work, a planning divided into three distinct time periods was proposed. These periods are described in the following three sections.

1.3.1. Research on the subject (September - November)

In order to obtain the necessary knowledge to carry out this work and thus acquire a good theoretical basis, the first step was to carry out an investigation of the concepts related to the CT and the different tools used in educational centers. The knowledge acquired and the tools analyzed are described in section 2 of this document. The milestones completed during this first period are the following:

- To carry out an in-depth analysis of what is the current situation of CT in education, to see how it is taught and how it can be beneficial for students.
- To analyze what is the impact of serious games in education and how they can help to develop CT.

1.3.2. Development and testing of La Mansión Paranormal. Prototyping of Articoding (November - March)

This stage consisted of designing and carrying out the development of a first game to teach CT concepts, from its first version to its final version, carrying out tests and improvements throughout its development. The details of this game are described in chapter 3 of this report. In addition, it also covers the first Articoding prototype. During this phase, the review of the state of the art has been extended through the analysis of other games focused on CT and the learning of basic programming concepts. The milestones to be met in this second stage have been:

- To carry out an analysis of a wide catalog of games related to the CT and/or to teaching.
- To design a game (La Mansión Paranormal) being the first contact with the CT.

- Develop a first prototype of the mentioned game as a proof of concept and iterate on it to polish the initial design.
- Expand the content of the prototype for testing with teachers and students.
- Set aside an iterative period to test the game, find bugs and fix them.
- Use the feedback obtained to close the final version of the game.

1.3.3. Development of Articoding (March- June)

This last stage is dedicated to the development of the second game, which has the objective of teaching programming concepts. The stage covers from the game design to its development and validation with users. In the final stages of development, quick iterations of 1 to 3 days are performed, polishing the remaining details to create a product and to be able to test it with users. These rapid iterations consist of:

- **Meeting with the tutor and co-tutor**. Where the current state of the game is shown and possible improvements and corrections are discussed. Tasks are generated in this step.
- **Implementation of improvements and bug fixes**. It is evaluated which tasks can be implemented and which are the highest priority. These tasks are implemented and those that have been considered not so important are discarded.
- **Shipment of builds and collection of feedback**. The different builds are generated and sent to the tutor and cotutor to check that everything is implemented correctly. Feedback is received and a date is set for the next meeting.

The design and development of Articoding is described in Chapter 4 of this report. In addition, the milestones completed at this stage are:

- Design the core game (Articoding) and perform rapid prototyping iterations to polish that design.
- Develop and implement the design ideas, having small iterative periods to change design decisions.
- Set aside an iterative period to test the main game, find bugs and fix them.
- Internationalize and localize the game into English and Spanish.
- Design an evaluation model with users for user testing.
- Analyze the results obtained and close the game with the feedback obtained.

1.4. Document structure

The rest of this document, which describes the work carried out, is structured as follows:

- Chapter 2. It exposes the current situation of the CT, relating it to different aspects such as programming or school subjects. In addition, it makes a study of the types of games that develop the CT and summarizes an exhaustive analysis of some of the games that best promote the CT.
- Chapter 3. It presents the first game that was developed for this work: La Mansión Paranormal. Reflects on the results obtained and the tests carried out with users.
- Chapter 4. It presents the evolution of the idea of the second game, Articoding, from its conception to its final version. It goes into detail about the design of the game, its development and implementation. It also exposes some problems and difficulties encountered during the development. Finally, it presents the conclusions and results, and talks about the user tests carried out.
- Chapter 5. It exposes the conclusions on the work done, checking if the objectives have been achieved. It also includes ideas that were not implemented, as well as possible improvements that could be developed in the future.
- Chapter 6. Lists the contributions made by each member of the group, where the contributions can be classified into three categories: implementation, memory, and others.

It also includes a set of annexes with complementary information and some of the documents that have been developed throughout this work:

- Appendix A. It shows a detailed table on the taxonomy of serious games where a classification of the use of serious games and the fields of knowledge is made.
- Appendix B. Exhibits a table showing the benefits and applications of serious games in education and the different educational sectors.
- Appendix C. It offers a table-summary of the exhaustive analysis carried out in Chapter 2 on serious games that promote CT.
- Appendix D. Provides the teaching guide for the game La Mansión Paranormal used during the formative evaluation with the experts.
- Appendix E. Provides the Articoding teaching guide used during testing with teachers and at school.

- Appendix F. Presents in more detail the work done on internationalization and localization of Articoding.

Capítulo 2

Contextualización y estado del arte

"El conocimiento no es una vasija que se llena, sino un fuego que se enciende" - Plutarco

En este capítulo se contextualiza el trabajo realizado abordando los dos aspectos principales que son, por un lado, la generalización de la enseñanza del pensamiento computacional en general y la programación en particular en la escuela y, por otro lado, como se pueden usar los juegos para promover el interés de los alumnos en estos temas y mejorar su enseñanza.

2.1. Pensamiento Computacional en la educación

El Pensamiento Computacional ofrece grandes beneficios en cualquier ámbito incluyendo, por ejemplo, el campo de los negocios, mercados financieros, energía, viajes y turismo, o servicios públicos.

> *"Computational thinking can be applied to any function of a commercial business or public service. Planning and forecasting are based on patterns of generalisation or abstraction."* (Snalune, 2015, p. 2)

En el caso del ámbito educativo el interés por desarrollar el PC ha aumentado en gran medida, apareciendo asignaturas y herramientas enfocadas a su desarrollo, ya que ha demostrado ser útil tanto en campos tecnológicos, como no tecnológicos. Este capítulo presenta el estado actual del PC en el ámbito educativo y algunas de las herramientas que existen para su desarrollo.

2.1.1. PC en STEM

La capacidad de desarrollo tecnológico no ha dejado de avanzar. Las herramientas digitales que proporciona la tecnología forman, cada vez más, parte de la vida diaria de las personas. Y es que tecnologías como las redes sociales, las aplicaciones de mensajería instantánea, los sistemas de gestión de contenido como

Wikipedia, discos virtuales o streaming como Google Drive, están presentes en nuestro día a día.

> *"The digital world has a growing need for creators to solve problems in all walks of life. They have to be able to adapt existing disciplinary practices with digital technologies."* (Kong, 2016, p. 378).

Uno de los ámbitos en los que la tecnología se está introduciendo es el de la educación, este hecho ha sido sobre todo importante durante el curso 2020/2021 con la situación vivida globalmente durante la pandemia de COVID19. Cada vez más centros educativos optan por el uso de dispositivos electrónicos como ordenadores, tabletas y pizarras digitales, etc., e introducen asignaturas y conceptos relacionados con la programación y el PC en las aulas.

La generalización del uso de tecnologías ha incrementado el interés de muchos países por introducir el PC como un conjunto de habilidades de resolución de problemas a adquirir por la nueva generación de estudiantes (Román-González et al., 2017). Este interés es mayor en las disciplinas de STEM (acrónimo de las palabras en inglés Ciencia, Tecnología, Ingeniería, Matemáticas) ya que se está empezando a considerar el PC como el núcleo de dichas disciplinas (Román-González et al., 2017).

> *"Research has shown that CT may bring about a variety of cognitive and non-cognitive benefits, especially for learning in STEM subjects"* (Zhao & Shute, 2019, p. 1).

Además, las investigaciones han mostrado que los niños desde los cuatro años pueden comprender conceptos básicos de programación y construir programas simples en proyectos de robótica (Flannery et al., 2013).

> *"The recent focus on computational thinking as a key 21st century skill for all students has led to a number of curriculum initiatives to embed it in K-12 classrooms"* (Yadav et al., 2016, p. 565).

Como menciona S. Kong (2016), si se quiere apostar en unas futuras generaciones integradas con las tecnologías digitales y que posean capacidades creativas y de resolución de problemas, hay una necesidad creciente de desarrollar el pensamiento computacional en los más jóvenes.

2.1.2. PC en otras asignaturas

Además, las habilidades que puede aportar el PC pueden ser utilizadas para otras disciplinas que, comúnmente, se consideran muy distanciadas de las disciplinas STEM.

> *"[...], these techniques have proven to be effective in a broader range of disciplines, with applications as diverse as engineering, medicine, law, and journalism"* (Gouws et al., 2013, p .10)

Es decir, no sólo refuerza capacidades en áreas más científicas-tecnológicas sino también en otras como las áreas sociales, haciendo que el PC sea beneficioso para cualquier persona independientemente de su especialización o de su tipo de trabajo. Es por este motivo que el PC lleva años aplicándose en campos ajenos a las disciplinas clásicas de STEM.

> *"[...] areas of active study include algorithmic medicine, computational archaeology, computational economics, computational finance, computation and journalism, computational law, computational social science, and digital humanities."* (Wing, 2010, p. 2)

Las habilidades que componen el PC pueden incluso ser aplicadas en la vida diaria de todas las personas, habilidades como la secuenciación, la ordenación o la descomposición de un problema complejo en otros más simples, son algunos de los ejemplos que propone Wing (2010) para mostrar que pensar computacionalmente es una habilidad que ofrece beneficios en cualquier campo.

En el Anexo B de este trabajo se ofrece una tabla con los beneficios y aplicaciones que tiene el pensamiento computacional en distintas asignaturas escolares.

2.2. Relación entre PC y la programación

La programación y el PC se suelen relacionar en la literatura, pero es importante destacar que no son lo mismo. La programación es una herramienta clave para apoyar las funciones cognitivas (e.g., razonamiento, cálculo, comprensión) involucradas en el PC (Grover & Pea, 2013). Es por esto que normalmente se suele utilizar la programación como el vehículo principal para desarrollar el PC.

Es por esta relación entre programación y CT, que existen una serie de conceptos propios del PC que son esenciales a la hora de aprender programación.

> *"There is a set of fundamental CT knowledge that are essential for all learners of computer programming at each level of the curriculum: events, sequences, repetition, conditional, parallelism, variables, operators, manipulation of data, elementary data structures, and simple algorithms like sorting and searching"* (Kong, 2016, p. 382)

Sin embargo, el simple hecho de enseñar a programar no implica que se aprendan los conceptos que conforman el PC.

> *"For example, while using these tools, students might develop good programming practice or spontaneously acquire a CT strategy, but there is little feedback available to alert them to this. The corollary to this might be when students create a working linear scenario without considering reusable patterns and other good programming practices"* (Kazimoglu et al., 2012, p. 524).

Es por esto por lo que también es necesario dar soporte a los estudiantes, recompensando las buenas conductas y ofreciendo la información suficiente para entender los errores. Nosotros consideramos que esto se puede conseguir de forma eficiente mediante el uso de los juegos serios. De esta manera, los estudiantes se mantendrían más motivados a la vez que lograrían comprender mejor las abstracciones necesarias y aprender buenas prácticas de programación.

2.3. Actividades para desarrollar el PC

Existe una gran variedad de actividades que sirven para desarrollar el PC. Estas se pueden clasificar en dos categorías: *unplugged* y *plugged*, según si se emplean dispositivos digitales o no. Además, existen distintos enfoques para el desarrollo de dichas actividades, como puede ser el uso de la programación, o de robots.

2.3.1. Un-plugged

Las actividades *unplugged* son actividades que se pueden realizar en un contexto sin dispositivos digitales. Es por eso que el desarrollo en escuelas del Pensamiento Computacional a base de actividades *unplugged* es de gran utilidad, sobre todo en lugares donde no se dispone de una infraestructura tecnológica, como electricidad, internet, ordenadores o cualquier otro dispositivo electrónico (Brackmann et al., 2017).

> *"Such activities involve logic games, cards, strings or physical movements that are used to represent and understand computer science concepts such as algorithms or data transmission"* (Brackmann et al., 2017, p. 65).

Este tipo de actividades son útiles para empezar a desarrollar las aptitudes del PC en estudiantes de escuela intermedia, aunque de momento parece que no hay evidencia suficiente para determinar que sólo con actividades *unplugged* se pueda desarrollar al máximo las aptitudes de PC.

> *"[...], even if the unplugged activities can be a good resource for introducing students into CT, it is apparent that this approach has limitations and, therefore, further research is necessary to identify the point at which the unplugged approach loses its effectiveness and the use of computing devices is required to keep on developing CT skills."* (Brackmann et al., 2017, p. 71).

2.3.2. Plugged

Al contrario que las actividades unplugged, las actividades *plugged* son aquellas actividades cuya realización requiere de algún tipo de dispositivo digital. Este tipo de actividades son las que más se suelen utilizar para desarrollar el PC, no solo por la comodidad y flexibilidad que ofrecen los dispositivos electrónicos, sino también por que habitualmente tienen una mayor cercanía con las Ciencias de la Computación y con la programación.

De hecho en las escuelas se utilizan cada vez más recursos tecnológicos y esto ha provocado, por ejemplo, que el aprendizaje de un lenguaje de programación como Python ha llegado a suscitar un mayor interés que el aprendizaje de la lengua extranjera más popular en los colegios como es el francés en Reino Unido[6].

2.3.3. PC sin programación

Aunque generalmente se suele relacionar la enseñanza de PC con la programación, es posible enseñar conceptos propios del PC sin necesidad de utilizar un lenguaje de programación concreto.

Normalmente estas aplicaciones plantean problemas lógicos que pretenden simular situaciones de la realidad y donde cada uno de estos problemas suele estar relacionado con al menos una habilidad del PC. Un claro ejemplo de este tipo de

[6]Fuente: https://www.information-age.com/python-overtakes-french-most-popular-language-taught-primary-schools-123460073/

problemas son las Bebras task. Bebras[7] es una comunidad internacional que cada año organiza un concurso de PC conocido como Bebras Contest.

> *"The task is designed to be solvable without previous knowledge from Computer Science or programming. Nevertheless, each task is related to one computational concept and one or more CT skills"* (Araujo et al., 2019, p. 3).

Este tipo de problemas pueden considerarse un buen punto de entrada a los conceptos básicos del PC ya que no es necesario ningún conocimiento previo. (Araujo et al., 2019)

2.3.4. Lenguajes de programación usados

Otro enfoque consiste en lograr mejorar la enseñanza de la programación haciendo que los entornos para programar sean más sencillos y fáciles de comprender (e.g., programación visual) o que los lenguajes de programación utilizados sean más sencillos de utilizar y más legibles.

2.3.4.1. Programación visual

En este tipo de programación se utilizan elementos visuales que, combinándolos con otros, terminan formando un programa que se puede ejecutar. Unos de las formas de programación visual más usada es la programación por bloques.

Esta forma de programación consiste en, como si de piezas de puzle se tratara, encajar unos bloques con otros. Normalmente, la funcionalidad que realizan los distintos bloques viene descrita con texto dentro del propio bloque. De esta manera, si un bloque se encarga de mover un objeto, dentro del bloque vendrá escrito "mover".

Uno de los sistemas de bloques más conocidos es el de *blockly*[8], un proyecto hecho por desarrolladores de Google.

[7] https://www.bebras.org/about.html
[8] https://developers.google.com/blockly

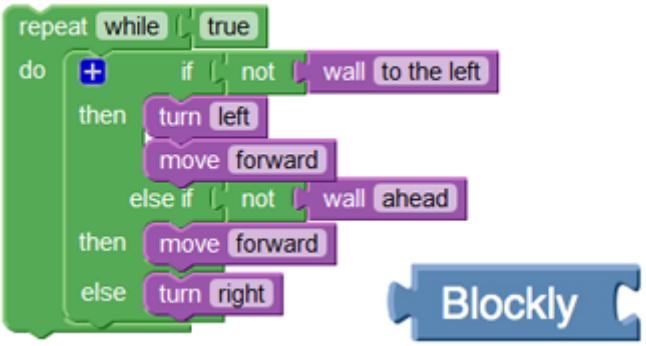


Figura 1: Imagen de bloques de *blockly*

Existe una gran variedad de proyectos profesionales que usan blockly ya que sus bloques pueden traducirse directamente a lenguajes de programación como JavaScript, PHP, Dart o Python[9].

Scratch es otro de los más populares y que más se usan en la escuela en España. Scratch es un entorno de desarrollo dirigido a programadores de entre 8 y 16 años.

Figura 2: Imagen de bloques de Scratch

Contiene una serie de bloques con los que puedes controlar una serie de personajes. Scratch está bastante acotado en ese sentido ya que tiene una expresividad predeterminada. Por eso se considera que *blockly* es más completo, ya que se pueden hacer bloques personalizados y crear proyectos más complejos, mientras que con Scratch no se puede.

[9] Fuente: https://www.robotlab.com/blog/blockly-vs-scratch-whats-best-for-me

Otra forma de programación visual es la de programación por símbolos. Este tipo de programación es muy similar a la programación por bloques. La diferencia más destacable es que, mientras que en la programación por bloques utiliza texto para describir la funcionalidad que realiza el bloque, en la programación por símbolos utiliza simbología descriptiva.

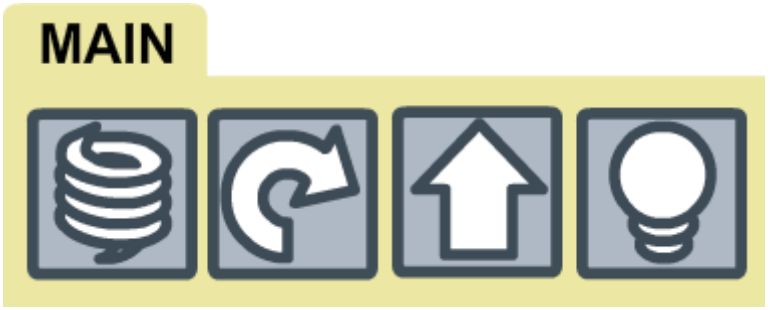


Figura 3: Símbolos (LightBot, 2021)

De esta manera, si la funcionalidad de un bloque[10] es descrita por el texto "mover", en el caso de la programación por símbolos es descrita por el símbolo de una flecha, como se puede observar en la Figura 3.

Esto disminuye mucho la escalabilidad de este tipo de programación, por eso se suele encontrar en entornos muy bien definidos con funcionalidades muy limitadas como se muestra en la Figura 3.

Además, dependiendo del contexto, estos símbolos se pueden mostrar como piezas de puzle o simplemente como rectángulos que se tienen que colocar uno al lado del otro.

2.3.4.2. Programación en texto

Este tipo de programación es el más cercano a la programación real. Gracias a ello, es el que menos limitaciones tiene independientemente del entorno. Los lenguajes de programación más utilizados son los de JavaScript y Python, aunque normalmente se usa más la sintaxis de un lenguaje que el propio lenguaje en sí.

La ventaja de estos lenguajes es que no requieren de entornos de ejecución complejos y, en cierta manera, son más próximos al lenguaje natural que otros lenguajes tradicionales (e.g., C, C++ o Java). Esto ha hecho que Python sea el lenguaje de programación elegido por muchas escuelas para introducir la programación en cursos más avanzados (y normalmente como continuación de una introducción en cursos inferiores mediante Scratch).

[10] Referencia al bloque de un sistema de bloques como *blockly* o Scratch.

2.3.5. Otros enfoques

2.3.5.1. Programación tangible

Este tipo de programación se trata de programar a base de utilizar objetos que se pueden tocar, mover, combinar, etc. Esta aproximación llama la atención en áreas de educación porque se acerca al área conocida como *tangible learning*, un área en el que se enseña a base de utilizar objetos manipulables.

> *"Design-Based Research (DBR) is concerned with the pragmatic application of learning theories, aiming to narrow the relationship between theory and practice, through tangible learning examples and artefacts, which can be used and reused in the real world"* (Vahldick et al., 2020, p. 3).

Uno de los proyectos más conocido en relación con la programación tangible es el de Project Bloks[11] de Google. Un programa de investigación cuyo objetivo era "construir la nueva generación de programación tangible para niños". Se trata de un sistema de módulos hecho con electrónica. Conectando de distintas maneras dichos módulos se consigue programar distintos comportamientos.

2.3.5.2. Robots

> *"At larger scales, we see a growing use of mobile phones, radio frequency identification tags, sensors, actuators and robots."* (Wing, 2008, p. 3723).

Con el avance tecnológico, la robótica también se ha hecho un hueco en el campo del pensamiento computacional. La robótica es un área fructífera muy cercana a la programación que ha demostrado ser de utilidad para el desarrollo de las aptitudes del PC en las personas independientemente de su edad o género (Shute et al., 2017).

> *"[...] research has shown that children as young as four years old can understand basic computer programming concepts and can build and program simple robotics projects"* (Flannery et al., 2013, p. 1).

Si junto a los robots se *gamifica*[12] el contexto en el que se usan, puede ofrecer un objetivo lúdico y atractivo que puede ayudar al desarrollo del pensamiento computacional en un entorno físico y táctil.

> *"[...] game design and gameplay entail various goals for players to solve. These can be smaller goals represented within levels of the game, and larger*

[11] https://projectbloks.withgoogle.com/
[12] Trasladar mecánicas y/o dinámicas de un juego a un ámbito real, como el de la educación.

goals represented by boss levels and/or part of the narrative. To succeed, players need to derive solution plans, and if a plan fails, then a modified plan is developed" (Shute et al., 2017, p. 8).

"Qualitative data generated from interviews and think-aloud protocols confirmed the effectiveness of robotics to develop CT concepts and solve problems effectively." (Shute et al., 2017, p. 7).

2.4. Análisis de juegos relacionados con el PC o la programación

Como el principal objetivo del proyecto es desarrollar juegos para promover el interés de los alumnos sobre el PC y la programación y para mejorar su enseñanza se ha realizado una extensa búsqueda y análisis de juegos relacionados con estos temas. El objetivo es entender que se ha hecho en este campo, estudiando decisiones de diseño e ideas de modelos de aprendizaje, e incluso, cuando ha sido posible, probando los juegos o aplicaciones. Estos juegos son los que se han considerado más relevantes y en los que, gracias al análisis realizado, se han podido identificar cuáles eran sus enfoques y decisiones de diseño, algunos de los cuales se han reutilizado en nuestras soluciones.

No obstante, no sólo se han considerado juegos propiamente dichos si no también entornos o aplicaciones que de alguna manera tratan de motivar o simplificar el aprendizaje del PC o de la programación (e.g. KIBO, ScratchJr). También se han analizado otras aplicaciones en otros campos como, por ejemplo, Duolingo debido al éxito que ha tenido en la promoción de un tema tan complejo como es el aprendizaje de idiomas.

2.4.1. Programming Hero

Programming Hero[13] es una aplicación móvil para el aprendizaje de la programación creado por Codinism[14]. Su principal idea es que tras aprender un concepto nuevo, el usuario añade nuevas mecánicas a un juego que se va programando de manera progresiva. La aplicación fue seleccionada por CODE[15] como aplicación número uno promotora de programación.

[13] https://www.programming-hero.com/
[14] http://www.codinism.com/
[15] https://code.org/

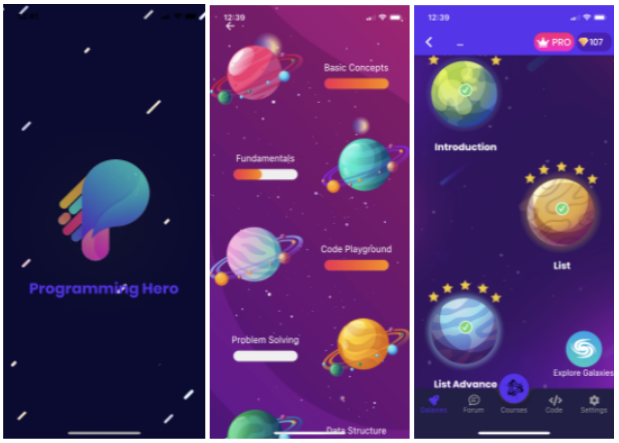


Figura 4: Imágenes del temario (Programming Hero, 2020)

La aplicación tiene todo el temario dividido en galaxias y cada galaxia tiene una serie de planetas que poseen el grueso de la teoría. Cada nivel o planeta se compone de módulos que pueden ser texto explicativo teórico, ejemplos teóricos o ejercicios a completar.

Al completar todos los módulos, desbloqueas el siguiente planeta. Y al completar todos los planetas, desbloqueas la siguiente galaxia. Esto le añade un componente de progresión muy interesante a la aplicación.

2.4.2. Grasshopper

Grasshopper[16] es una aplicación móvil desarrollada por Google cuyo objetivo es enseñar conceptos básicos de programación de una manera rápida y cómoda. Para esto, se utilizan varios elementos que *gamifican* el aprendizaje (perfil de usuario, avatar de jugador, medallas, puntuaciones, etc.). El juego utiliza un sistema de tarjetas para completar el código. Algunas tarjetas se pueden rellenar con texto.

[16] https://grasshopper.app/es_419/

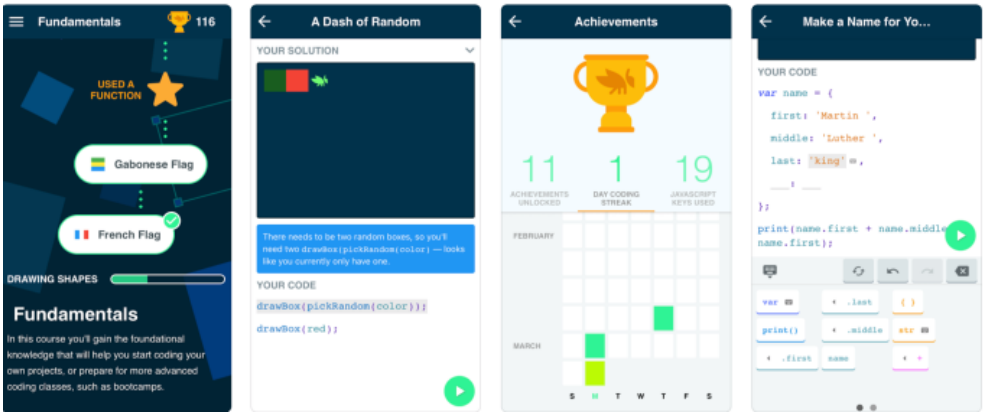


Figura 5: Imágenes del juego (Grasshopper, 2020)

La aplicación cuenta con niveles teóricos que explican con mayor profundidad conceptos ya utilizados o sirven como introducción a nuevas instrucciones de programación. También hay niveles prácticos donde el usuario tiene que resolver el problema propuesto. Este problema se describe de manera textual donde puede haber una pequeña introducción teórica y además, se muestra visualmente el objetivo que el usuario debe alcanzar. En estos niveles el usuario tiene un apartado donde escribir el código y una ventana que muestra la ejecución del programa. Por último, existen desafíos que ponen a prueba los conocimientos del usuario a través de preguntas de opción múltiple.

Además de jugar los niveles principales, la aplicación cuenta con un apartado de práctica para repasar conceptos ya vistos mediante nuevos problemas. Asimismo, incluye el apartado Code playground donde el usuario puede crear su propio código.

2.4.3. Code Hero

Code Hero[17] es un juego educacional creado por Primer Labs. Su objetivo principal es enseñar a los jugadores a programar y a familiarizarse con motores de videojuegos como Unity. Todo ello lo hace desde dentro de un entorno 3D, simulando que se está en una "universidad virtual". También enseña conceptos de publicación y anima a la publicación de un juego una vez se haya aprendido lo suficiente.

[17] https://codeherogame.wordpress.com/

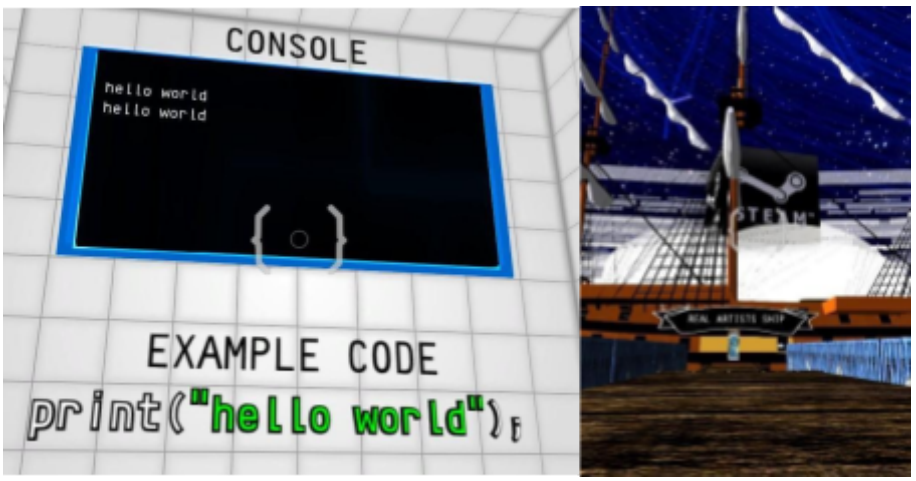


Figura 6: Imágenes in-game (Code Hero, 2020)

El juego cuenta con una sala principal desde donde se puede acceder a las distintas áreas en las que se divide la "Gamebridge Unityversity": Biblioteca, Niveles, Museo, Auditorio, Sala de Exposiciones, Puerto de Steam, etc.

Además, el jugador tiene total libertad para aprender como quiera. De esta manera, los que ya cuenten con conocimientos de programación y/o Unity pueden ir directamente a las fases más avanzadas y complejas del juego.

2.4.4. CodeCombat

CodeCombat[18] es un juego serio web enmascarado de RPG[19] cuyo objetivo es enseñar a estudiantes a programar. Posee una gran comunidad de más de 5.000.000 de jugadores y ofrece herramientas al profesorado para hacer el seguimiento de sus alumnos.

Cada nivel que posee el juego tiene un número de objetivos que el jugador tiene que cumplir para que el nivel sea completado. El juego ofrece a los profesores estadísticas generales de su clase, progreso de sus alumnos y tiempos de compleción. Para avanzar en el juego, puedes escribir y ejecutar código con el que controlas el movimiento de un personaje por el escenario para completar los objetivos del nivel.

[18] https://codecombat.com/
[19] Role-playing game (Videojuego de rol).

Figura 7: Imagen del juego (CodeCombat, 2020)

Al completar los distintos niveles, se van desbloqueando items (i.e. armadura, armas, etc.) que, al equipar, el jugador obtiene la posibilidad de usar ciertos métodos. Por ejemplo, las primeras botas que se desbloquean permiten usar los métodos que hacen que tu personaje se mueva por el escenario.

2.4.5. Light Bot

Light Bot[20] es un juego que pretende enseñar conceptos básicos de programación (i.e. bucles, recursividad, funciones, etc.) sin mostrar explícitamente código. En Light Bot el objetivo es dar las instrucciones correctas al robot para que se mueva por el escenario encendiendo todas las casillas azules. Según un estudio, Light Bot es una aplicación muy adecuada para enseñar conceptos de Pensamiento Computacional, sobre todo en el apartado de modelos y abstracción (Gouws et al., 2013).

[20] https://lightbot.com/

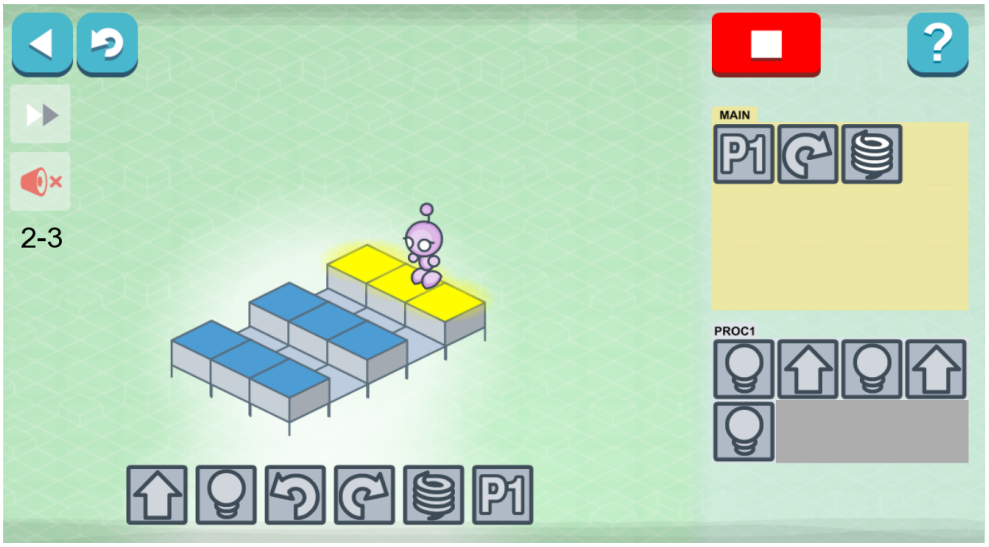


Figura 8: Imagen in-game del juego (Light Bot, 2020)

El jugador tiene hasta tres espacios donde colocar los bloques de acción: el bloque principal main, el bloque de función 1 y el bloque de función 2. La mayoría de los niveles están diseñados para que no se pueda completar el nivel si no se hace un uso correcto de las funciones, incluyendo niveles en los que se limita el espacio del bloque principal para forzar el uso de las funciones.

Una función puede incluirse a sí misma como parte de sus instrucciones, pudiendo así crear recursiones infinitas. Esto obliga al jugador a buscar patrones que se repiten en el escenario para completar el nivel. Como dice D. Yaroslavaski en su análisis de la aplicación, todos los niveles de Light Bot podrían reescribirse como código de un lenguaje tipado de una manera sencilla, lo que permite una entrada más fácil a lenguajes de programación más comunes (Yaroslavaski, 2014).

2.4.6. KIBO

KIBO[21] es un kit de robots físicos sin ordenador que tiene como objetivo enseñar conceptos de robótica y programación a los más pequeños de la casa, fomentando así la enseñanza en STEAM (Science, Technology, Engineering, Arts y Mathematics).

Consiste en experimentar con distintos bloques y componentes, para dar vida (programar) a un pequeño robot de forma divertida, rápida y sencilla.

[21] https://www.shop.kinderlabrobotics.com/main.sc

Figura 9: Imagen de KIBO

El usuario dispone de multitud de bloques distintos. Se pueden usar dichos bloques para formar una secuencia de acciones (código) para conseguir la funcionalidad que desea: mover el robot, girar el robot si choca con algo, etc. De esta forma, el niño irá probando y cambiando cosas hasta que consiga que el robot haga lo que él quiera. Así aprenderá rápidamente de sus errores y obtendrá una recompensa inmediata cuando lo haga bien, lo que le motivará a seguir programando funcionalidades nuevas.

KIBO no enseña un lenguaje de programación en sí, pero se asemeja mucho a lenguajes visuales como Scratch. Aunque KIBO aplica el concepto de *Tangible Programming*, que consiste en "programar con las manos". De esta forma, los niños pueden descubrir nuevas texturas, formas, colores, etc.

2.4.7. BeBlockly

BeBlocky[22] es un juego móvil destinado al aprendizaje de conceptos de programación para niños pequeños. En este juego, los niños tienen que superar los niveles moviendo a un personaje por un escenario por casillas hasta llegar a cumplir el objetivo del nivel.

[22] https://beblocky.com/

Figura 10: Imagen del juego (BeBlockly, 2020)

Cuando se entra a un nivel, se explica para qué se usan los bloques nuevos de los que se dispone. Se muestran ciertas anotaciones dependiendo del nivel y se entra en el escenario. Al pulsar el botón de programar, sale una ventana donde se tiene los bloques disponibles y se pueden combinar, similar a Scratch. Para ejecutar los bloques colocados, se le da al botón de *Play* que ejecuta bloque a bloque y se refleja de manera visual en el escenario.

El juego asume que el dispositivo móvil que se usa es de un padre o una madre y son estos los que deben de controlar el proceso de aprendizaje.

2.4.8. Scratch Jr.

Scratch Jr.[23] es una aplicación móvil dedicada a enseñar programación a niños de entre 5 y 7 años. Es una versión simplificada de Scratch donde los bloques no tienen texto, sino que se utilizan imágenes representativas de la funcionalidad de cada bloque.

> *"The ScratchJr project rests on the premise that children in kindergarten to second grade can in- deed learn and apply concepts of programming and problem- solving to create interactive animations and stories"* (Flannery et al., 2013).

[23] https://www.scratchjr.org/

Figura 11: Imagen de la aplicación (Scratch Jr., 2020)

Scratch Jr. no cuenta con niveles o una progresión determinada. En su lugar, da la posibilidad de crear proyectos desde 0. Estos proyectos no tienen objetivo o guía, simplemente se le da libertad al usuario para crear lo que quiera.

La aplicación cuenta con varias herramientas para aprender y mejorar el uso de Scratch Jr., tanto guías de las mecánicas básicas de la aplicación como proyectos de ejemplo que pueden dar ideas al usuario. Además, Scratch Jr. también ofrece en su web un apartado "Enseña" dedicado a educadores donde se presentan varias actividades, formas de evaluación y un currículo a seguir para mejorar y guiar el proceso de aprendizaje.

Scratch Jr. no cuenta con ningún método de evaluación. En ese aspecto, no asegura que los usuarios no sigan o aprendan malas prácticas a la hora de programar, aunque al tratarse de una abstracción tan grande de los conceptos básicos de programación, es difícil determinar cuándo se producen. Aunque Scratch Jr. no ofrece evaluación dentro de la aplicación, en su página web hay disponibles varios tests para comprobar el progreso de los usuarios.

2.4.9. SpriteBox: Code Hour

SpriteBox: Code Hour[24] es un juego serio para el desarrollo del pensamiento computacional en los niños de K-8 (niños de hasta 14 años). El juego ofrece al

[24] https://spritebox.com/hour.html

jugador ciertos niveles o puzles que tiene que resolver con programación. Además, el jugador puede moverse por el nivel saltando, andando o usando escaleras.

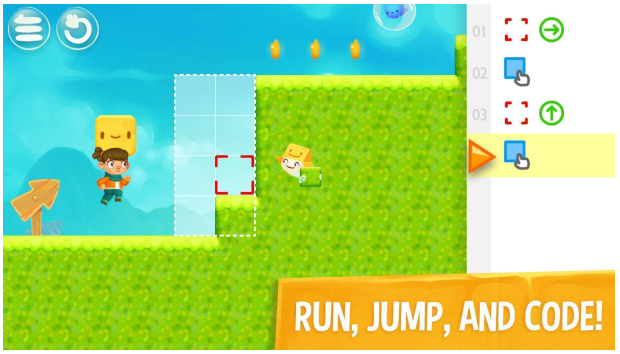


Figura 12: Imagen promocional del juego (SpriteBox: Code Hour, 2020)

El juego busca ofrecer una experiencia de juego relajada, donde el jugador puede aprender los conceptos a su ritmo y sin presión. El juego cuenta con una pequeña narrativa mediante la que se explican las mecánicas principales y los objetivos del jugador, el jugador tiene que completar varios escenarios mientras que rescata los espíritus encerrados mediante la resolución de puzles de programación.

Las dinámicas principales del juego son: la interacción con el nivel mediante las diferentes mecánicas de juego de plataformas y por otra parte la capacidad de analizar y resolver los puzles de programación que el juego plantea. El juego tiene dos formas de puntuar al jugador. Existen ciertos objetos coleccionables (estrellas y espíritus encerrados en botellas) que se tiene que recoger para completar al 100% el nivel.

2.4.10. CodeSpark Academy

Code Spark Academy[25] es un juego serio para el desarrollo del pensamiento computacional en los niños de K-5 (escuela primaria). El juego ofrece al jugador ciertos niveles o puzles que tiene que resolver con programación. El jugador debe indicar la secuencia de instrucciones que el avatar realizará para completar el nivel.

La dinámica principal del juego viene directamente dada por la mecánica principal. Al tener el puzle del nivel programado e iniciar la prueba, se ve de manera visual cómo el personaje realiza las acciones programadas.

[25] https://codespark.com/

Al introducir la mecánica de los bucles, el jugador debe buscar patrones que se repiten en el nivel para así minimizar el número de instrucciones usadas y obtener mayor puntuación.

Este juego ofrece un gran soporte para su despliegue en las aulas, cuenta con varias herramientas para los profesores entre las que se incluyen varias guías y actividades guiadas, herramientas de monitorización del progreso de cada alumno, soluciones a varios problemas o actividades "desconectadas".

Además, en su página web afirman que CodeSpark Academy ha sido probado y se ha mostrado eficaz, donde un 22% de los niños incrementó su precisión en tareas de secuenciación y un 55% mostró un aumento en su confianza a la hora de resolver problemas.

Figura 13: Imagen in-game del juego (CodeSpark Academy, 2020)

El juego en su versión Code Hour ofrece niveles que tratan conceptos básicos de programación, como la secuenciación de instrucciones, el uso de bucles y el uso de funciones (dadas por el propio juego), estos conceptos ayudan a reforzar elementos propios del PC como la abstracción (uso de funciones sin conocer su implementación) o el reconocimiento de patrones (para poder utilizar bucles).

2.4.11. May's Journey

May's Journey[26] es un juego serio para el desarrollo computacional. El juego cuenta con una serie de salas. En estas salas encontrarás pequeños puzles que se tienen que resolver usando código en texto plano.

Además de poder programar para resolver problemas (arreglar caminos, construir escaleras, etc.), puedes recolectar gemas que son coleccionables. También se puede investigar y buscar llaves que son necesarias para abrir algunas puertas.

Por el camino te encuentras pergaminos con los que desbloqueas nuevas habilidades (en este caso, conocimiento en instrucciones nuevas).

Figura 14: Imagen in-game del juego (May's Journey, 2021)

El entorno del juego es en 3D con vista cenital. Esto añade gran atractivo y hace que se sienta más un juego al que jugar y pasarlo bien a un juego con el que sólo aprender. Además, se muestra como un mundo misterioso en el que usan luces, estatuas y demás elementos que llaman la atención incrementando la impresión de que es realmente un juego y no tanto una aplicación de aprendizaje.

2.4.12. Blockly Games

Blockly Games[27] es una aplicación web creada por google como parte de la iniciativa *Code with google*[28]. La aplicación pretende enseñar conceptos de

[26] https://maysjourney.com/
[27] https://blockly.games
[28] https://edu.google.com/code-with-google/

programación, planteando diferentes juegos y desafíos que el jugador debe solucionar utilizando el lenguaje de programación visual blockly.

El juego cuenta con varias categorías, cada categoría plantea un tipo de juego diferente. Además las categorías tienen varios niveles con una dificultad incremental. De esta forma, se introducen todos los conceptos disponibles en blockly de una manera ordenada y gradual. Se enseñan tanto conceptos básicos de programación (bucles condicionales) como conceptos más avanzados(bucles anidados, funciones)

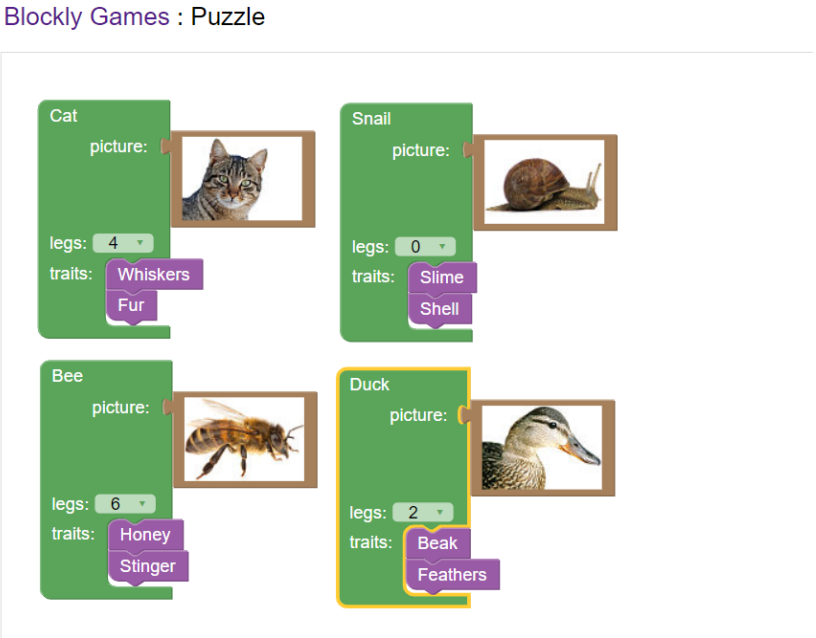


Figura 15: Imagen in-game del juego (Blockly Games, 2021)

La aplicación también ofrece la posibilidad de resolver algunos niveles utilizando javascript en lugar de blockly. De esta manera, también se puede utilizar la aplicación con usuarios con conocimientos básicos de programación. También hay que destacar que la aplicación ofrece muy pocas ayudas o descripciones de los bloques que hay que utilizar.

2.4.13. Dragon Architect

Dragon architect[29] es un juego serio educativo desarrollado por el Center for Game Science[30] que pretende enseñar conceptos básicos de programación. El juego consiste en la construcción de edificios mediante el uso de un lenguaje de programación visual (blockly).

[29] http://dragonarchitect.net
[30] http://centerforgamescience.org

Dragon architect cuenta con una pequeña narrativa en la que el jugador debe ayudar a un dragón a construir diferentes elementos. Para ello, el jugador primero debe aprender el lenguaje del dragón que consiste en los bloques de código.

El juego plantea dos modos de juego, el primero consiste en una serie de tutoriales y niveles prediseñados donde se enseñan los conceptos básicos del juego (mover, construir bloques, bucles y funciones). Una vez se han completado estos niveles, los conceptos aprendidos se desbloquean en el modo sandbox. En este modo, el jugador puede utilizar los bloques de código como quiera para realizar sus propias construcciones. Este modo de juego pretende dar la oportunidad de explorar al jugador para mejorar la experiencia de aprendizaje.

> *"The educational literature also highlights the need to combine open ended exploration with sufficient structured guidance"* (Bauer et al., 2017, p. 6).

Este juego propone un enfoque diferente al modelo tradicional de este tipo de aplicaciones, ya que centra su atención en el modo sandbox y en la libertad de creación del jugador, mientras que mantiene el modelo basado en movimientos en un tablero de otras aplicaciones.

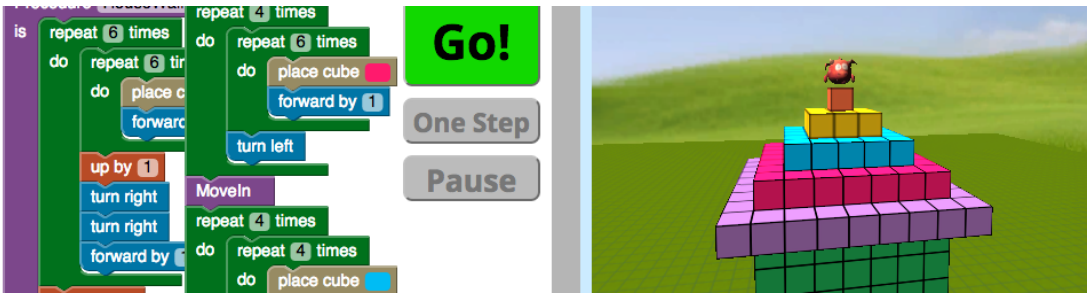


Figura 16: Imagen in-game del juego (Dragon Architect, 2021)

2.4.14. Duolingo

Duolingo[31] es una aplicación móvil y web creada en Estados Unidos con el propósito de aprender distintos idiomas. Cuenta con más de 30 idiomas y un examen para obtener la certificación del nivel de inglés. Si bien Duolingo no promueve o enseña conceptos del PC y la programación, utiliza varias técnicas de enseñanza que son muy efectivas, entre ellas la división del contenido por lecciones y un sistema de recompensas. Además es una aplicación muy utilizada y extendida. Por estos motivos se ha utilizado como referencia para el trabajo llevado a cabo.

[31] https://es.duolingo.com/courses/all

Se dispone de una versión de Duolingo donde se pueden realizar pruebas concretas para obtener certificados o un porcentaje de fluidez para cada idioma. También existe una versión Duolingo for Schools, que incluye analíticas y donde los profesores pueden ver el progreso de sus alumnos.

Figura 17: Menú de niveles del idioma inglés en Android (Duolingo, 2020)

En el juego, se van superando lecciones compuestas por rondas. Las lecciones están a su vez, agrupadas en niveles. Cuando se completa un nivel se consigue 1 corona. Se puede seguir jugando al mismo nivel para llegar a las 5 coronas.

Además, el jugador puede ganar experiencia de varias formas como completando un nivel, jugando varios días seguidos o completando logros. Todo tu progreso se puede comparar con el progreso de los amigos a los que tengas agregados.

2.5. Conclusiones

Como conclusión del análisis de las principales herramientas utilizadas para enseñar programación, se pueden distinguir tres grupos de aplicaciones según sus características similares:

- Por un lado, las aplicaciones dirigidas a los niños más pequeños, las cuales utilizan lenguajes muy simplificados, basados en símbolos e iconos simples,

para enseñar conceptos muy básicos de programación, como bucles, funciones y variables. En este grupo se pueden encontrar aplicaciones como Scratch Jr. y Light Bot o herramientas como KIBO (aunque en este caso usa programación mediante bloques físicos).

- Otro de los grupos sería aplicaciones dirigidas a un público de mayor edad, niños entre 10 y 12 años. En este grupo se comienza a usar un lenguaje más parecido a los lenguajes de programación habituales, pero sigue existiendo una abstracción para facilitar el aprendizaje. Esta abstracción se ve en el uso de bloques o de funciones dadas por la propia aplicación. En este grupo se pueden observar aplicaciones como CodeCombat, BeBlockly y herramientas de este estilo como Scratch.
- El último de los grupos se compone de aplicaciones enfocadas en enseñar un lenguaje de programación en concreto, se suele utilizar una sintaxis concreta de este lenguaje y se le da mayor libertad al usuario a la hora de crear sus programas. En estas aplicaciones la abstracción de conceptos de programación es reducida y limitada a ciertos casos (e.g. funciones de renderizado, física, etc.). También se suele dar un entorno más abierto al usuario donde realizar sus propios programas sin un objetivo predeterminado. Se pueden encontrar aplicaciones o juegos como Programming Hero, Grasshopper o Code Hero.

En el Anexo C del documento, se muestran las tablas de análisis realizadas en conjunto con el análisis de aplicaciones descrito en esta sección.

Capítulo 3

PC con puzles: La Mansión Paranormal

"En algún lugar, algo increíble está esperando ser conocido." - Carl Sagan

3.1. La mansión paranormal

La mansión paranormal[32] es un juego serio narrativo con puzles para el desarrollo del Pensamiento Computacional (PC). El juego está desarrollado en Unity junto con el paquete de uAdventure[33], que proporciona las herramientas para generar escenas, transiciones, conversaciones, etc. de manera rápida y sencilla. El juego utiliza una idea similar al de las e*scape rooms*, donde el jugador se ve encerrado en un recinto y tiene que lograr encontrar una manera de escapar resolviendo puzles.

Los puzles incluidos en el juego están diseñados para cubrir 4 de los aspectos identificados como muy relevantes en el pensamiento computacional: descomposición, patrones, abstracción y algoritmos. En el Anexo D se puede encontrar una descripción del juego más orientada a profesores y a su uso docente.

3.1.1. Contexto y narrativa

El juego se ha desarrollado inicialmente como parte de las prácticas y del proyecto final de la asignatura optativa de Juegos Serios (JS) del grado de Desarrollo de Videojuegos aunque posteriormente se ha mejorado de forma significativa y se ha evaluado para que forme parte de los resultados de este trabajo. El juego está pensado como una herramienta de ayuda al profesor y está disponible gratuitamente para ordenador (Windows y Linux).

El objetivo de este proyecto es desarrollar las habilidades del PC en los usuarios que lo jueguen, experimentando con un modelo de juego narrativo con puzles y finalmente, observar y analizar su efectividad.

---

[32] Repositorio de GitHub: https://github.com/WeArePawns/ParanormalMansion
Vídeo-presentación (YouTube): Mansión Paranormal - Proyecto Final

[33] Plataforma desarrollada en Unity para crear juegos narrativos por el equipo de la universidad e-UCM: uAdventure – eUCM

La narrativa del juego es que eres un detective junior que debe resolver el misterio de la mansión Fernández Manjón: desde que el dueño desapareció han ocurrido cosas extrañas y se escuchan sonidos que provienen del despacho de la mansión. Al entrar, todas las puertas se cierran. Debes resolver los puzles para poder escapar. Para resolver el caso contarás con la ayuda de los detectives hermanos John, Jim y Jack, que te darán pistas y consejos que te serán de utilidad para resolver los intrincados puzles. Además, se ha convocado al mayordomo de la mansión y a la señora Ada Mantine para ser interrogados.

El jugador puede explorar libremente la mansión moviéndose entre distintas escenas, donde puede interactuar con los elementos que se presentan en ellas como: personajes con los que hablar, objetos que inspeccionar, etc. Así podrá conocer a los personajes y los secretos que esconden, investigar los escenarios en busca de pistas y resolver el misterio que esconden las cuatro paredes de la mansión. Para avanzar el jugador tendrá que resolver distintos puzles.

Este juego es el primer juego serio desarrollado por los autores de este trabajo, siendo la primera toma de contacto tanto en desarrollo como en abordar el CT con juegos.

3.1.2. Puzles

En el juego hay 5 puzles únicos, donde cada uno tiene 3 niveles progresivos de dificultad. Cada puzle cuenta también con la posibilidad de solicitar varias pistas que ayuden al usuario a resolverlo. A continuación se describen los puzles mencionados y cómo se relacionan con cada uno de los aspectos anteriores.

3.1.2.1. Anillas

Se muestran 2 o más anillos (dependiendo de la dificultad), cada uno de ellos marcado con un color diferente. El jugador tiene a disposición una serie de palancas que giran los anillos una cantidad de grados. El objetivo es alinear las marcas de color de las anillas. El jugador debe identificar el patrón de giro, pensar un algoritmo y descomponer el problema en partes más pequeñas. Los aspectos del PC que se cubren en este puzle son los siguientes:

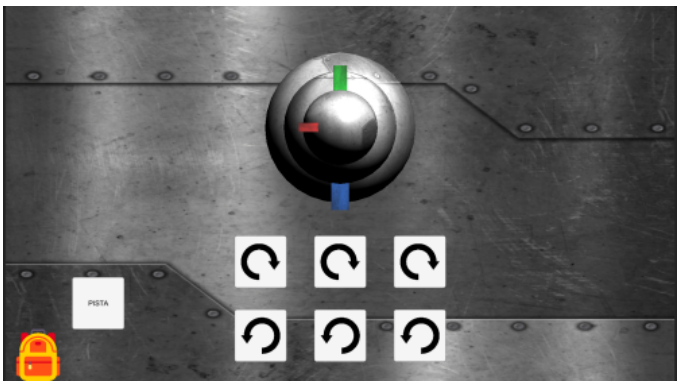


Figura 18: Puzle “Anillas” (Mansión Paranormal)

- **Identificación de patrones**: identificar cuántos grados gira cada anilla con el uso de las distintas palancas. Las palancas provocan el giro de las anillas en sentido dextrógiro (en el sentido de las agujas del reloj) o en sentido levógiro (sentido contrario a las agujas del reloj).
- **Descomposición**: a medida que se aumenta la dificultad, aumenta el número de anillas y la necesidad de fragmentar el problema en problemas más pequeños para hallar la solución.
- **Diseño de algoritmos**: usar los patrones identificados y la descomposición de problemas para conseguir llegar a la solución.

3.1.2.2. Hacking ético

El jugador tiene a su disposición una serie de tarjetas con unas líneas pintadas de azul o rojo. En la parte superior puede observarse el progreso del jugador y el resultado esperado. En la parte inferior se muestran una serie de tarjetas. El usuario debe seleccionar 3 tarjetas para que, combinando las líneas de colores, se obtenga la tarjeta objetivo. El jugador debe averiguar cuál es el patrón de combinación de las tarjetas y puede ir depurando su combinación actual con la tarjeta marcada como “Resultado”. Los aspectos del PC que se cubren en este puzle son los siguientes:

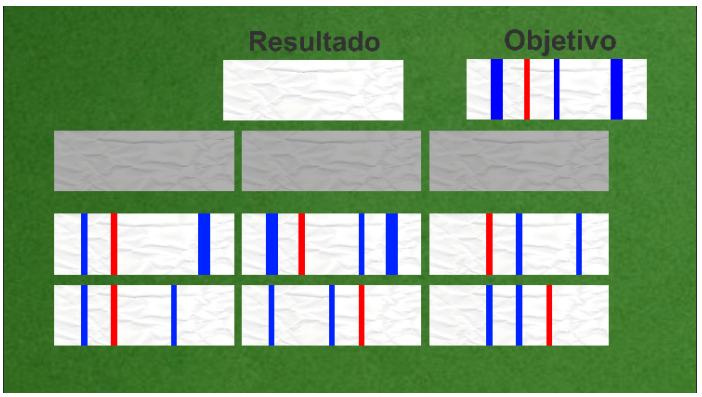


Figura 19: Puzle “Hacking ético” (Mansión Paranormal)

- **Identificación de patrones**: identificar cómo se combinan las distintas tarjetas entre sí.
- **Descomposición**: descomponer en grupos de tarjetas o incluso en segmentos concretos que faciliten la obtención de la solución correcta.
- **Abstracción**: ignorar las bandas de colores que pueda identificar como innecesarias.
- **Diseño de algoritmos**: usar los patrones identificados y la descomposición de problemas para conseguir llegar a la solución.

3.1.2.3. Formas y colores

En la pantalla aparecen varios botones con formas geométricas y una secuencia numérica. El jugador debe encontrar una relación entre las formas y los números e introducir, mediante la pulsación de los botones, la secuencia correcta de polígonos. El jugador debe abstraerse de las formas geométricas y hacer una relación entre dichas formas y los números (relación: número de lados del polígono - número). Los aspectos del PC que se cubren en este puzle son los siguientes:

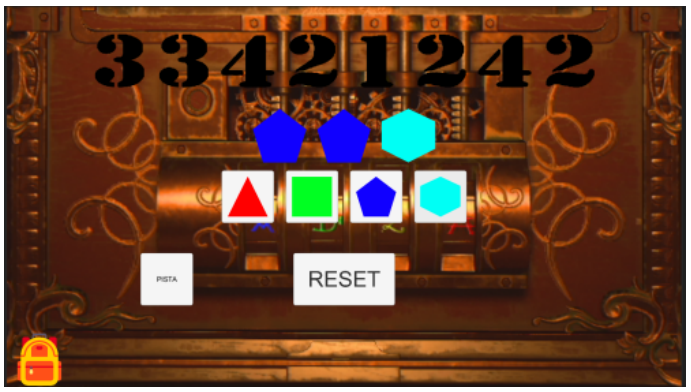


Figura 20: Puzle "Formas y colores" (Mansión Paranormal)

- **Abstracción**: ignorar la información irrelevante tras averiguar la relación existente entre los números y las formas geométricas (e.g., el orden de posición del botón o el color de la forma geométrica)

3.1.2.4. Laberinto de números

El objetivo de este puzle es averiguar e introducir un código de números, de longitud dependiente a la dificultad. Para ello, se le proporciona una nota con las instrucciones del algoritmo que debe seguir y una matriz de casillas con números. El algoritmo empieza en la casilla azul y termina en la casilla roja. El jugador debe hacer uso de la secuenciación para seguir el algoritmo y abstraerse de los números que no le interesen. Los aspectos del PC que se cubren en este puzle son los siguientes:

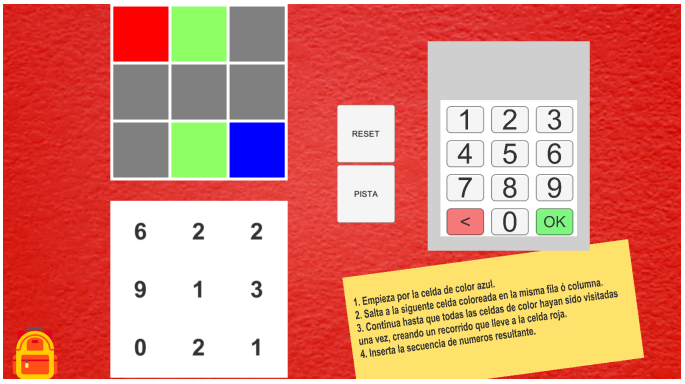


Figura 21: Puzle "Laberinto de números" (Mansión Paranormal)

- **Abstracción**: ignorar los números cuyas casillas correspondan a casillas grises, teniendo en cuenta exclusivamente los números de las casillas rojas, verdes y azules.
- **Diseño de algoritmos:** en este caso, el algoritmo se muestra en forma de nota. Se debe ser capaz de entender el algoritmo y seguirlo para resolver el puzle.

3.1.2.5. Electricista

Se dispone un tablero que comienza con ciertas casillas encendidas de manera aleatoria. El objetivo del jugador es lograr que todas las casillas se encuentren apagadas. Cuando el jugador pulsa sobre una casilla, ésta y las cuatro casillas adyacentes invierten su estado. Es decir, si la casilla estaba apagada se enciende, y viceversa. Se pretende que el jugador busque la secuencia correcta de casillas que debe pulsar siguiendo un patrón válido. Los aspectos del PC que se cubren en este puzle son los siguientes:

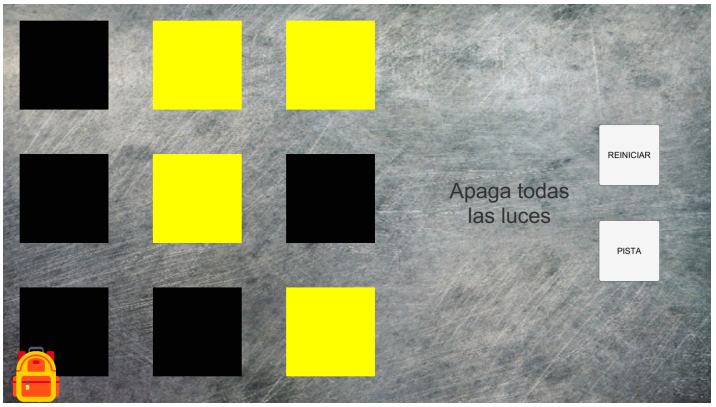


Figura 22: Puzle “Electricista” (Mansión Paranormal)

- **Identificación de patrones**: identificar que al interactuar/activar una casilla, las casillas adyacentes también cambian de estado (si está apagado se enciende y viceversa).
- **Abstracción**: junto con la identificación de patrones, lograr una mayor abstracción al entender la lógica de estados.

3.1.3. Pruebas con usuarios y resultados

Tras concluir la asignatura de Juegos Serios, se ha prolongado el desarrollo de La Mansión Paranormal para incorporar analíticas a los puzles, realizar pruebas con usuarios reales y mejorar el juego mediante los comentarios y los datos recibidos de los usuarios.

Durante este periodo de validación y mejora, se ha adoptado un proceso de desarrollo de iteraciones cortas en el que se ha pretendido pulir al máximo el juego y el proceso de aprendizaje. Inicialmente se han realizado estas iteraciones con integrantes del grupo de investigación de e-UCM como probadores. Después de un par de iteraciones, se ha enviado el juego a Jesús Moreno León[34], profesor doctor en informática y experto en el campo del PC. Moreno León también es cofundador de Dr. Scratch y autor de la web Programamos[35]. Jesús ha aportado feedback adicional sobre el juego y además lo ha apoyado escribiendo una entrada en su web, para que los profesores interesados puedan probar el videojuego a través de un formulario de registro de participación. El formulario se ha redactado en colaboración con investigadores del grupo e-UCM, que ha sido publicado en la web Programamos[36] junto a un panfleto informativo sobre el videojuego La Mansión Paranormal.

Como parte del proceso de validación del juego se propuso un formulario de registro que se ha publicado en la página web de Programamos. En el formulario se han registrado un total de 21 profesores de colegios e institutos procedentes de distintas provincias de España: Valencia, Vizcaya, Madrid, Cataluña, Murcia, Sevilla y Granada. Así como de otros países: Colombia, Argentina y Uruguay.

En este formulario de registro, el 95.2% de los profesores registrados (20 de ellos) han afirmado que estarían dispuestos a probar el juego en sus clases con sus alumnos. Estos mismos profesores también han contestado que estarían dispuestos a hacer una entrevista tras probar el juego. Además un 76,2 % (16 de los registrados) han indicado que el sistema preferido para ejecutar el juego era Windows mientras que el 23,8% restante (5 registrados) preferían usar Linux.

Desde ese punto se comenzó a trabajar en la *build* (versión estable del videojuego) que se ha enviado a los profesores, así como en una guía docente sobre el videojuego en colaboración con los investigadores de e-UCM. La guía docente pretende dar contexto sobre el videojuego, además de explicar brevemente el

[34] https://programamos.es/author/j-morenol/
[35] https://programamos.es/el-proyecto/
[36] https://programamos.es/participa-con-tus-estudiantes-en-la-validacion-de-un-videojuego-sobre-pensamiento-computacional/

funcionamiento de los puzles, los conceptos que enseñan y estrategias para su resolución. Esta guía docente se encuentra en el Anexo E.

La build enviada a los profesores cuenta con una encuesta pre-juego y una encuesta post-juego. Además, la versión cuenta con el sistema de telemetría de uAdventure, y todas las trazas recopiladas durante la ejecución han sido mandadas a un servidor a través de Simva. De esta forma se han podido observar tanto el progreso como las acciones de los jugadores casi en tiempo real.

Una vez se enviaron las versiones del juego a los profesores registrados, se recogieron trazas de sus sesiones de juegos. Además, los profesores también tuvieron que contestar un cuestionario anterior a la sesión de juego y otro cuestionario tras haber completado el juego. 9 de los 21 profesores registrados contestaron la encuesta pre-test. Aunque se obtuvieron trazas de los 9 profesores, solo 8 de ellos contestaron la encuesta post-test.

El análisis de los cuestionarios previos a la prueba muestra que todos los profesores registrados juegan a videojuegos (ordenador, móvil o consola) entre una y cuatro veces a la semana y la mayoría de los registrados se dedicaban a la labor docente en un campo relacionado con la tecnología o la informática. Además 6 de los 9 profesores habían utilizado con anterioridad videojuegos en el aula. También hay que resaltar que todos los profesores estaban familiarizados con el PC ya que todos ellos dieron descripciones válidas sobre el PC.

En lo que respecta a los datos obtenidos mediante las trazas recolectadas, hay que destacar que según la información trazas, solo 6 de los 9 usuarios completaron el juego (se perdieron trazas de algunos usuarios por problemas de sincronización del Tracker). Se puede observar que el tiempo medio de juego es de 45 minutos. La sesión más larga de los 6 profesores que completaron el juego, duró 1 hora y 20 minutos, mientras que la más corta duró cerca de media hora, la desviación estándar de la duración de las partidas es de 20 minutos.

En cuanto a la duración media de los puzles, se puede observar que el puzzle al que se dedicaba más tiempo es el de electricista con un tiempo medio de 11 minutos. El puzle que más rápido se completaba fue el de formas y colores con una media de 2 minutos y medio por compleción.

	Tiempo medio	Tiempo mínimo	Tiempo máximo	Desviación típica
Formas y Colores	2m 26s	1m 29s	3m 41s	52s
Hacking Ético	7m 8s	4m 6s	12m 37s	3m 13s
Anillas	4m 18s	2m 49s	6m 21s	1m 25s
Laberinto de números	3m 46s	2m 8s	6m 10s	1m 49s
Electricista	10m 56s	2m 57s	29m 27s	11m 45s
Juego Completo	45m 45s	30m 17s	1h 19m 45s	20m 47s

Es importante remarcar que tres de los tiempos máximos (electricista, hacking ético y formas y colores) pertenecen a la misma sesión de juego. Además observando los tiempos de compleción medios de todos los puzles, se puede observar un salto de los 2 a 4 minutos en puzles como anillas, formas y colores o laberinto de números a tiempos más elevados en los puzles de hacking ético y sobre todo en electricista, estos puzles también tienen una mayor desviación típica que los otros 3. Este salto puede deberse a un repunte en la dificultad de estos puzles frente a los demás.

Por último, el análisis de las respuestas al cuestionario post-test da una visión de la opinión de los profesores sobre el juego. A las preguntas *¿El juego es útil para trabajar el pensamiento computacional con los alumnos?*, *¿Crees que sería fácil de aplicar el juego en el aula?*, *¿Aplicarías el juego en alguna de tus clases?*, *¿Te ha resultado útil la guía de descripción proporcionada con el juego para entender el objetivo y la utilidad del juego?* todos los profesores contestaron de manera positiva con medias de 4, 3.6, 4.25 y 3.75 sobre 5 en una escala likert-5 donde 1 es muy poco y 5 mucho . Es por esto que se puede decir que consideran el juego una manera viable de introducir los conceptos base del PC y como una herramienta apta para ser utilizada en la enseñanza. Las preguntas

En cuanto a la duración y a la dificultad del juego, 2 de los profesores consideraron el juego demasiado largo para una clase mientras que el resto considero su duración adecuada. A la pregunta *¿Qué te parece el nivel de dificultad para tus alumnos?* 4 de los 7 profesores consideraron el juego difícil para los alumnos, confirmando las conclusiones obtenidas mediante el análisis de las trazas. Esto también se ve reflejado en los principales comentarios sobre el juego, en ellos, los profesores indican la diferencia de dificultad entre los diferentes puzles y en concreto

en el puzle electricista. Estos comentarios indican una mayor dificultad de estos niveles, ya que es de esperar que en una validación con estudiantes los alumnos tuviesen más dificultades, se considera necesario reducir la dificultad de estos niveles. El resto de comentarios, destacaban algunas incoherencias en los diálogos o el tipo de información que les gustaría ver sobre las partidas de los alumnos (principalmente tiempo en cada nivel, pistas usadas y el número de intentos).

3.1.4. Retroalimentación de la evaluación y versión final

Tras realizar la evaluación formativa con expertos se ha analizado la retroalimentación obtenida. Como se ha comentado en el apartado anterior, los comentarios más frecuentes en las encuestas de los profesores han sido acerca de la dificultad de los diferentes puzles y en concreto sobre el puzle 5, electricista.

También se ha recibido retroalimentación indicando que las conversaciones eran poco orgánicas y que no se entendía bien la manera en la que se presentaban algunos personajes. Con estos comentarios se ha realizado una versión final que incorpora el siguiente historial de cambios:

- Se han revisado las conversaciones y se han modificado para tener más coherencia con la situación que se presenta.
- Se ha modificado el puzle del electricista variando los parámetros de dificultad. Actualmente, la curva de aprendizaje y de dificultad se han aplanado respecto a las curvas de la versión anterior.

Por último, se ha llevado a cabo una entrevista a través de Google Meet con Henar Lanchas López (una de las profesoras que se registró para hacer la prueba) donde comentó que al juego le podría venir bien una hoja de ejercicios complementaria. Esta idea resulta interesante por lo que forma parte del trabajo futuro del videojuego.

3.1.5. Conclusiones

Una vez completada esta periodo de desarrollo y pruebas, las conclusiones sacadas del trabajo realizado hasta este punto son las siguientes:

- Es importante hacer una progresión adecuada de dificultad para el correcto aprendizaje de los conceptos.
- Es fundamental el uso de algún elemento como recompensa para mantener la motivación de los usuarios.
- Es mejor hacer iteraciones cortas de desarrollo combinándolo con prueba y recogida de feedback para hacer un pulido más eficaz del producto final.

Capítulo 4

Articoding: Desarrollo e implementación.

"Lo que vemos cambia lo que sabemos. Lo que conocemos cambia lo que vemos." - J. Piaget

4.1. ¿Qué es Articoding?

Articoding es un juego serio cuyo objetivo principal es la promoción del Pensamiento Computacional y el aprendizaje de conceptos básicos de programación. El jugador debe superar niveles resolviendo problemas que se plantean en un escenario en forma de tablero.

El objetivo del jugador es conseguir guiar el haz de luz de un láser hacia un objetivo utilizando la programación visual por bloques (similar a Scratch). Cuando se alcance el objetivo, el jugador puede avanzar al siguiente nivel.

El público al que se dirige el juego es a estudiantes de entre 12 y 16 años que hayan tenido poco contacto con el mundo de la programación y puedan obtener claros beneficios del desarrollo de su PC, no sólo en el campo de la computación sino en otros campos de STEM.

Aunque el juego está pensado para uso escolar, también puede ser utilizado en un ámbito personal, en ambos casos el usuario juega de forma individual resolviendo diversos problemas mediante los diferentes conceptos de programación que se le enseñan.

El uso en un ámbito escolar puede suponer una forma interesante de enseñar competencias básicas de programación frente a medios más tradicionales o aplicaciones más formales. La componente jugable que ofrece el juego puede aportar una mayor motivación al jugador y la pequeña narrativa subyacente al juego puede mejorar la inmersión.

Además, se ofrecen dos versiones del juego. Una en el que el jugador tiene que ir desbloqueando los niveles de manera progresiva y otra en la que todos los

niveles están desbloqueados desde el principio. Esta última versión resulta interesante para un uso guiado por un profesor en las aulas de los colegios.

Este juego forma parte de la etapa final del desarrollo de este trabajo, en el que se ha utilizado y tenido en cuenta todo lo aprendido en las etapas anteriores (e.g., investigación, PC en la educación, análisis de juegos, resultados de La Mansión Paranormal, etc.)

4.1.1. Objetivos pedagógicos

Los objetivos pedagógicos del juego se pueden resumir en introducir al usuario en la programación para promover el desarrollo de las habilidades clave del PC. En este juego se tienen en cuenta la descomposición, identificación de patrones, abstracción, diseño de algoritmo y depuración.

La enseñanza de conceptos de programación asegura, de manera intrínseca, el desarrollo de los principales elementos que componen el pensamiento computacional. Estudios relacionados mencionan cómo los estudiantes están expuestos al PC durante la programación ya que involucra el uso de conceptos de ciencias de la computación como los de abstracción, depuración o iteración para resolver problemas (Lye & Koh, 2014). Además, aseguran que esta forma de pensamiento puede ser considerada como fundamental para los niños de temprana edad ya que requiere pensar a múltiples niveles de abstracción (Wing, 2006). Algunos autores creen que el PC está al mismo nivel que muchas de las competencias que se consideran claves para el siglo XXI como son la creatividad, el pensamiento crítico y la resolución de problemas (Lye & Koh, 2014).

Por ello, se han diseñado una serie de objetivos pedagógicos que pretenden reforzar los objetivos propuestos anteriormente:

- Distinguir las distintas magnitudes de dificultad de los problemas.
- Identificar los componentes que constituyen el problema.
- Relacionar patrones vistos anteriormente con patrones de problemas para su resolución.
- Identificar y comparar la eficiencia y utilidad de los patrones para la resolución de un problema específico.
- Relacionar datos con un nombre que lo identifique.
- Tratamiento de datos a través del pensamiento lógico.
- Discriminación de malas prácticas en el ámbito de la programación.
- Composición lógica y matemática de operadores condicionales.
- Organizar el orden de ejecución (su jerarquía) de manera clara y lógica.

- Aprender a comparar soluciones válidas para la resolución de un mismo problema.
- Distinguir las ventajas y desventajas de elegir una solución de entre varias posibles soluciones.
- Definir pautas claras y coherentes para diseñar un algoritmo.
- Desarrollar la capacidad de analizar el comportamiento de un programa para saber qué fallos puede causar.
- Incentivar el interés por la programación y el desarrollo del pensamiento lógico.

4.2. Desarrollo de la idea y su evolución

Articoding es un juego cuyo diseño ha sido replanteado múltiples veces a lo largo del curso. Con cada etapa del desarrollo de este trabajo, se han planteado nuevas opciones y se han puesto en cuestión las decisiones ya tomadas para el diseño del juego. Siendo este un diseño que ha cambiado repetidas veces hasta obtener una idea de juego atractiva tanto para los autores como para los profesores y expertos.

A continuación, se presenta la evolución del diseño del juego, desde la idea inicial, junto con los cambios que ha sufrido durante el desarrollo, hasta la versión final y los motivos por los que se decidieron estos cambios.

4.2.1. Primera idea

Al inicio del proyecto, se ideó el juego como una app para dispositivos móviles (Android). Esta decisión de diseño se basaba principalmente en las siguientes ideas:

- Aprovechar la comodidad del formato móvil para motivar a los usuarios a usar la aplicación en cualquier momento y a cualquier hora.
- Utilizar funcionalidades de la plataforma móvil para fomentar el uso reiterado de la app e impulsar la constancia en el proceso de aprendizaje como son: enviar notificaciones cada cierto tiempo para recordar al usuario que debe avanzar en el juego.

De esta forma, se diseñó el juego con una interfaz adaptada a la orientación *portrait*[37] que tienen los dispositivos móviles.

En cuanto al contenido de la aplicación, originalmente se planteó como una aplicación que enseñase un lenguaje de programación concreto, en este caso se eligió

[37] Orientación vertical que tienen por defecto la mayoría de los smartphones y se caracteriza porque permite la sujeción del dispositivo con una sola mano.

enseñar C#. En el planteamiento inicial, el jugador controlaba a un programador junior que tenía como objetivo convertirse en un desarrollador de videojuegos. Los niveles consistían en problemas únicos que el programador tenía que resolver en su día a día utilizando la programación. Además, la aplicación contaba con un modo extra donde el jugador iba construyendo su propio juego a medida que avanzaba en la aplicación.

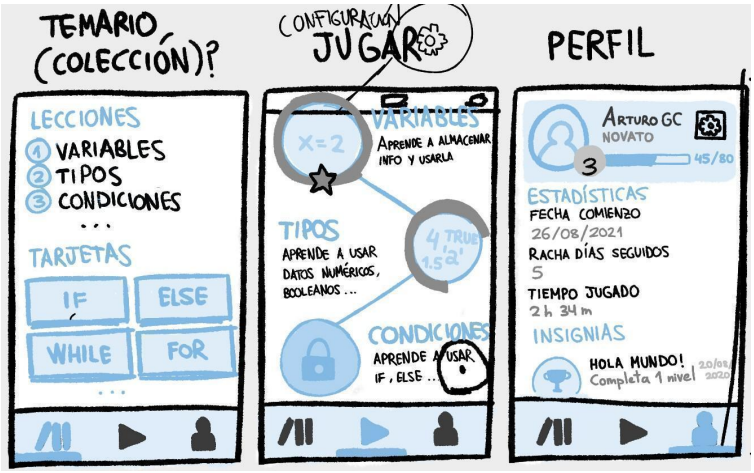


Figura 23: Primeros conceptos del diseño para móvil.

4.2.2. Idea intermedia

Durante nuestra primera entrevista con Jesús Moreno León, surgieron varios puntos a mejorar en el diseño de la aplicación. El primer punto fue la plataforma de la aplicación. Como mencionó Jesús, el despliegue de una aplicación móvil en un aula puede llegar a suponer grandes complicaciones ya que no todos los alumnos tienen dispositivos y en muchos colegios ni siquiera se permiten. El segundo punto que se comentó con Jesús fue el lenguaje de programación utilizado. Jesús comentó que, en los últimos años, el lenguaje de programación más utilizado con alumnos de entre 10 y 14 años en España era Scratch.

Teniendo en cuenta los comentarios de Jesús y lo que se había leído en la literatura sobre PC, se llevó a cabo un rediseño de la aplicación. Durante este rediseño, se decidió que en la aplicación se usaría un lenguaje de programación visual lo más similar posible a Scratch, para facilitar el uso en colegios y la adaptación de usuarios familiarizados con Scratch. Por otro lado, la plataforma principal de la aplicación pasó a ser el ordenador (Windows y Linux), ya que esta plataforma permitía un rápido despliegue en las aulas y reducía mucho los problemas potenciales. También se tomó la decisión de no incluir música ni efectos de sonido para evitar que los alumnos activasen el sonido y así facilitar su uso en el aula.

En este punto, el contenido de la aplicación seguía siendo el mismo al planteado originalmente, con la diferencia del lenguaje de programación utilizado. Los niveles eran pequeños problemas para resolver mediante la programación visual y existía un modo extra donde el jugador debía programar un videojuego "desde cero". El juego que se eligió recrear en este modo fue una simplificación del conocido juego para móvil *Flappy Bird* y el jugador podía ir construyendo el juego a medida que avanzaba en los niveles del modo principal. Cuando este modo se completa, es decir, cuando el juego está completamente programado, el jugador podría jugar al juego cuando quisiera como recompensa. Este último modo se decidió incluirlo tras analizar el juego de *Programming Hero*, que ofrece un modo de juego similar y resultó atractivo a los autores del trabajo ya que era un buen incentivo para mantener motivado al usuario.

4.2.3. Idea final

Finalmente, tras los resultados obtenidos en las pruebas de la mansión paranormal y un análisis de las aplicaciones similares a la planteada (con el objetivo de enseñar programación mediante lenguajes similares a Scratch). Se llegó a la conclusión de que tener un mismo modelo de problema para todos los niveles mejoraría el proceso de aprendizaje de los usuarios. Por este motivo, se decidió adoptar un modelo de problemas basado en un tablero, tal y como se observaba en aplicaciones de la competencia y en herramientas de validación de conceptos de PC como es el CTT[38].

Al ser el principal modelo en las aplicaciones más destacadas, se intentó buscar un modelo de problema que no se limitase a mover a un único personaje, pero manteniendo la representación mediante un tablero. Es por esto que se llegó a la decisión de utilizar láseres y espejos como elementos principales del juego. Además de diferenciar la solución de problemas, permitía el movimiento de más de un elemento del tablero, lo que aportaba una mayor expresividad a la hora de diseñar niveles. Esto además fuerza al jugador a que tenga el concepto de estado de juego donde tenga que considerar el estado de varios elementos.

Una vez tomada esta decisión, la narrativa original y el modo de juego extra sobre crear un videojuego desde cero carecían de sentido en el nuevo diseño, por este motivo se planteó una nueva narrativa que diese coherencia al videojuego. Además, se diseñó un nuevo modo de juego en el que el jugador fuese capaz de crear sus propios niveles, sustituyendo al modo extra planteado en la idea intermedia. Este

[38]Computational Thinking Test: https://docs.google.com/forms/d/e/1FAIpQLSdPdSj_ZVUhIhG4S3bCH6zXSHZoHHbv6OsmCF9drmbDpfBy_Q/viewform?gxids=7628

nuevo modo mantiene la coherencia de los niveles en forma de tablero y resulta mucho más interesante desde el punto de vista educativo, puesto que el alumno debe hacer una reflexión inversa y plantear un nuevo problema en vez de dar una solución a un problema ya existente. Además, permite a los profesores crear problemas concretos para que sus alumnos puedan resolverlos.

Tras estas reflexiones, se llegó al diseño final de la aplicación. Una aplicación para ordenador (Windows y Linux) cuyo objetivo es enseñar conceptos básicos de programación y fomentar el PC. Los niveles consisten en un tablero solucionable con elementos móviles. En estos niveles, el jugador utiliza un lenguaje de programación visual similar a Scratch para resolver el tablero propuesto. Además, se ofrece un modo de creación de niveles donde el jugador puede crear sus propios tableros para consolidar los conceptos aprendidos.

4.3. Diseño de Articoding (GDD)

4.3.1. Gameplay/Jugabilidad

Al iniciar el juego, el usuario puede seleccionar el nivel que quiere jugar dentro de los que tiene desbloqueados. Cuando se juega un nivel por primera vez, se le muestran al jugador los tutoriales necesarios para explicar los elementos y tarjetas que se introducen en ese nivel. Una vez explicados, estos conceptos se añaden al temario y el jugador los puede consultar cuando quiera.

Una vez que el usuario ha leído la teoría, se le presenta el tablero del problema que tiene que resolver. Después de esta pequeña introducción al problema, el jugador tiene libertad para añadir tarjetas y comprobar el resultado de la ejecución. Si el jugador se atasca en el nivel se le ofrecerán pequeñas ayudas para que no se quede atascado. Una vez el jugador obtenga la solución correcta, el nivel se dará por terminado, se puntuará cómo de bien se ha completado el nivel y el jugador podrá decidir si continuar con el siguiente nivel. Esta estructura es similar en todos los niveles.

Se ofrece un modo Creación al jugador con el que podrá crear sus propios niveles. Para que un nivel creado se considere válido, el usuario deberá ser capaz de solucionar el nivel que ha creado.

4.3.2. Narrativa

Aunque la narrativa no se transmite de manera explícita al jugador, es de gran importancia la inclusión de un personaje protagonista y un contexto narrativo que diese más sensación de juego.

El protagonista es Albert[39], un pingüino científico que ha sido encerrado en un laboratorio lleno de ordenadores, láseres, espejos, etc. Albert debe escapar del laboratorio haciendo uso de sus dotes como programador resolviendo cada nivel para avanzar al siguiente. Además de ser el protagonista, Albert está relacionado con el nombre del juego (Articoding = Artic + coding) y con el icono del producto.

Figura 24: Imagen de Albert, el protagonista de Articoding

La figura de este personaje se muestra de forma que el jugador se vea reflejado en él, ya que tanto él como Albert están programando mientras se está completando un nivel. Para ello, se ha colocado una pequeña ventana en la parte inferior derecha para mostrar al pingüino programando en la escena del nivel. Además, el personaje reacciona a diferentes situaciones:

- Mientras se está en el nivel, el pingüino realiza una animación en bucle programando.
- Si se completa el nivel, se abre una puerta y el pingüino se pondrá a caminar hacia ella para avanzar al siguiente nivel.
- Si se falla el nivel, su ordenador explota y el pingüino se cae hacia atrás.

[39] En honor al gran Albert Einstein.

- Si se reintenta el nivel, el pingüino se incorpora y se pone a programar de nuevo.

4.3.3. Estilo visual

El estilo artístico de la interfaz sigue unas pautas minimalistas, usando una paleta de colores planos blancos, grises y azules. Se ha elegido este estilo para dar una mayor comprensión y accesibilidad a los usuarios.

Para la visualización del nivel se optó por crear modelos 3D para los elementos del tablero en vez de usar sprites 2D. Esto permite rotar y acercar/alejar la cámara del nivel (inicialmente en vista cenital[40]) para ver el puzle desde cualquier ángulo. De esta forma, el jugador puede pensar la solución al problema con otro punto de vista.

Esta representación gráfica resulta más atractiva para el público más joven y puede motivarlos a seguir jugando. Además, permite destacar Articoding de otros juegos de enseñanza con un apartado gráfico más sencillo.

4.3.4. Mecánicas y dinámicas

Las mecánicas con las que el jugador puede interactuar con el juego son simples. Arrastrar bloques de código y encajarlos entre sí para generar un algoritmo que permita solucionar el problema que toque. El jugador tiene disponible una serie de bloques de código distintos en cada nivel y con un número de usos limitados, de esta manera se limita que el jugador se salga de la curva de aprendizaje deseada. También puede consultar pistas para resolver el problema, consultar los tutoriales previamente enseñados si lo necesita y, por supuesto, poder volver a jugar tantas veces se desee un mismo nivel.

Además de la mecánica principal, el jugador cuenta con otra serie de mecánicas únicas para el modo de Creación de niveles. En este el jugador puede cambiar el tamaño del tablero que se quiera crear, modificar el tipo de celda, colocar los distintos elementos disponibles en el tablero y guardarlo para poder jugarlo más adelante. En este modo también se harán uso de estas mecánicas del modo principal para la validación de los niveles creados.

[40] Perspectiva de juego en la que la cámara se encuentra situada perpendicular al suelo y por tanto se ofrece una visión orientada de arriba a abajo. (https://www.gamerdic.es/termino/vista-cenital/)

4.3.5. Sistema de pistas

Para ayudar a los jugadores que tienen más dificultades a la hora de resolver los niveles y evitar que se frustren se ha implementado un sistema de pistas.

En la interfaz del nivel, arriba a la derecha, se encuentra un botón con un icono de una bombilla que proporciona una pista cuando se pulsa. Las pistas consisten en flechas, tanto de dirección como de giro, que se dibujan en el tablero para indicar los movimientos y rotaciones de elementos que deben hacerse.

En cada nivel se pueden solicitar hasta un máximo de 3 pistas, proporcionando ⅓ de las flechas del nivel cada vez. Estas flechas son añadidas "a mano" en la creación de cada nivel.

4.3.6. Sistema de progresión y recompensas

Cuando se juega por primera vez, sólo se puede acceder al primer nivel de la primera categoría de niveles (Variables - 1), estando todos los demás niveles bloqueados. Para desbloquear los niveles se deben completar los niveles anteriores.

Esto permite ir introduciendo nuevas tarjetas de código y nuevos elementos del tablero poco a poco al usuario, enseñando a su vez conceptos de programación en un orden lógico y creciente. Esta progresión crea una sensación de juego imprescindible para que el usuario aprenda correctamente y no pierda el interés.

También se ha incorporado un sistema de recompensas para motivar al usuario a seguir completando los niveles y premiarlos por dar soluciones óptimas. Este sistema de recompensas consiste en otorgar estrellas al jugador dependiendo de su desempeño en la resolución de los niveles.

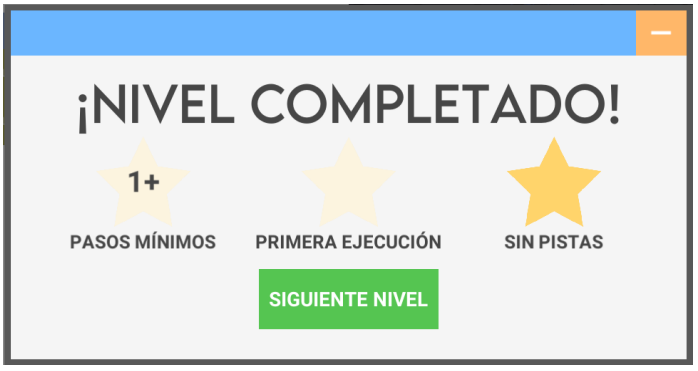


Figura 25: Panel de estrellas de Articoding

Para cada nivel es posible obtener hasta un máximo de 3 estrellas cuando éste es completado, estas estrellas se consiguen de la siguiente forma:

- Completar el nivel sin haber usado pistas.
- Completar el nivel en la primera ejecución.
- Completar el nivel con el mínimo número de pasos[41].

Las estrellas que ha conseguido el usuario se muestran al final del nivel, en el panel de "Nivel completado", y en el menú de selección de niveles. Estas estrellas también sirven para medir el nivel de habilidad de cada jugador.

4.3.7. Sistema de ventanas emergentes

Este sistema está pensado para remarcar elementos concretos que aparecen en pantalla y acompañarlos con una ventana emergente que contenga información sobre el elemento marcado. También se puede utilizar sin remarcar elementos, mostrando simplemente un mensaje que se quiere que el usuario lea.

Para ello, se hace uso de un shader encargado de remarcar una región de pantalla que recibirá por parámetros. Para mantener las cosas sencillas, basta con oscurecer el área no marcada y dejar intacta la zona a remarcar. De esta manera se consigue focalizar la atención en un punto concreto.

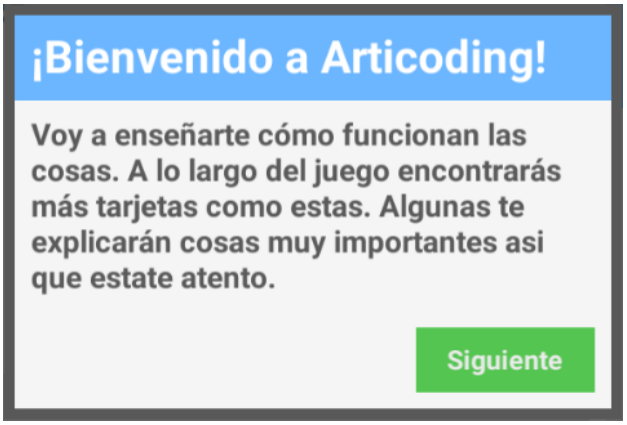


Figura 26: Ejemplo de pop-up en Articoding

La ventana emergente tiene la capacidad de mostrar un título y una descripción concretas. La ventana se redimensiona de manera dinámica por lo que, si se elige no poner un título, la ventana lo ignora y no deja un espacio vacío.

[41] Se considera un paso a cualquier acción que modifique el estado del tablero: mover un elemento, girar un elemento, cambiar el estado de una puerta, etc.

Para describir la información que se va a mostrar en la ventana emergente, se hace uso de una clase que guarda dicha información además de la posibilidad de encolar mensajes para el mismo elemento.

4.3.8. Sistema de tutoriales

El juego requiere de un sistema que vaya enseñando al usuario todos aquellos conceptos nuevos que se quieran introducir. Para ello, se usa un sistema de tutoriales que, junto con el sistema de *pop-ups* (ventanas emergentes), van señalando y explicando cada elemento del juego.

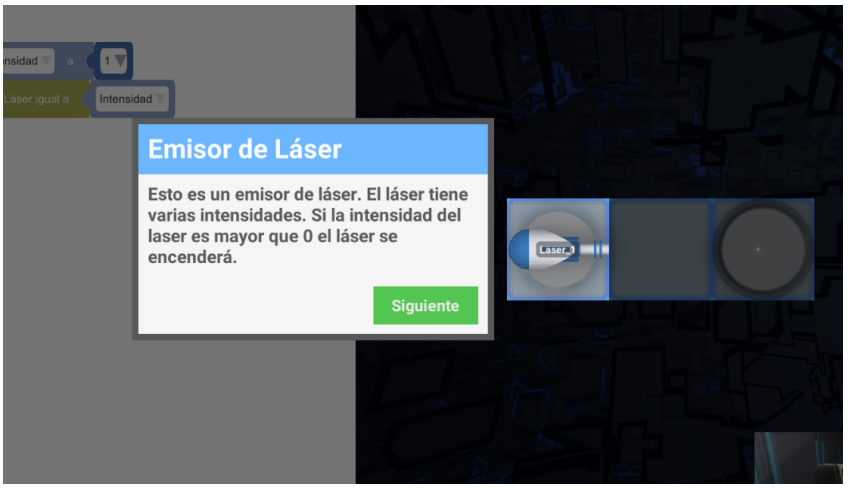


Figura 27: Ejemplo de tutorial usando el sistema de pop-ups en Articoding

Los tutoriales abarcan desde el uso de la interfaz, como la utilidad de las distintas mecánicas.

4.3.9. Sistema de temario

Cada vez que se desbloquea un concepto nuevo en cada nivel se añade al "temario". Se puede acceder a él siempre que se quiera desde el menú principal y desde la pantalla de juego para poder consultar cada concepto. Es el equivalente a un manual de referencia.

Figura 28: Sección de TUTORIALES (temario) en Articoding

En el contenido del temario, se encuentra la utilidad de todas las tarjetas desbloqueadas y los elementos del tablero desbloqueados.

Para simplificar la implementación, se hace uso del sistema de tutoriales para que todo aquello que se muestre sea lo que aparece en el contenido del temario.

4.3.10. Sistema de creación de niveles

Una parte importante del proceso de aprendizaje es emplear los conceptos aprendidos en una situación real, este concepto es conocido como construccionismo. Como destaca (Lye & Koh, 2014) citando un concepto enunciado por Seymur Papert sobre el construccionismo en el proceso de aprendizaje. El construccionismo da una mayor importancia al rol de las construcciones en el mundo como soporte a las construcciones mentales.

Mientras que los niveles básicos cuentan con un acercamiento más clásico a la enseñanza, donde se enseña un concepto y posteriormente se aplica para buscar una solución en una situación diseñada para desarrollar ese concepto. El juego ofrece otro modo donde el jugador es capaz de crear sus propios tableros y resolverlos utilizando los elementos vistos en los niveles básicos. Este modo de juego ofrece al jugador una forma de afianzar los conceptos aprendidos plasmándolos en sus propios niveles. De esta manera se utiliza uno de los pilares del construccionismo según Kafai & Resnick, que describe como en el construccionismo, los estudiantes están constantemente concentrados en el desarrollo del conocimiento mediante la construcción de productos (Lye & Koh, 2014).

Figura 29: Modo Creación en Articoding

Este modo de creación de niveles es accesible desde el menú principal. Inicialmente aparece un tablero vacío y debajo una serie de elementos que se pueden colocar en el tablero. Estos elementos pueden ser arrastrados y añadidos al tablero, una vez añadidos al tablero pueden girarse y moverse, además si se colocan fuera del tablero se eliminarán. También es posible cambiar el tipo de cada una de las celdas del tablero.

Una vez el jugador haya creado su nivel debe probar que es un nivel válido, para esto, existe un modo de prueba de nivel que consiste en disponer del espacio de código de los niveles normales, con todas las tarjetas desbloqueadas. Si el usuario es capaz de completar el nivel que ha creado, se considera un nivel válido y se añade a la lista de niveles creados por el usuario. Todos los niveles creados por el usuario que hayan sido validados se añaden como niveles jugables al panel del creador en el menú principal.

4.3.11. Sistema de guardado de progreso

El progreso del jugador (niveles completados y estrellas obtenidas) se guarda en la memoria del sistema, de forma que el usuario pueda cerrar el juego y retomarlo más tarde.

El sistema de guardado utiliza el patrón singleton, además implementa métodos de guardado a un archivo y cargado desde un archivo. Los datos que el sistema de guardado se encarga de gestionar consisten, principalmente en datos sobre el progreso del jugador en el juego y datos sobre los tutoriales que ya se han visto. Es por esto que existen dos subsistemas que se encargan de gestionar estos datos, así, el

sistema de guardado se comunica con estos dos sistemas para cargar y guardar los datos necesarios. Estos subsistemas son: el sistema de progreso y el sistema de tutoriales.

El progreso del jugador consiste en los siguientes elementos: las categorías desbloqueadas, los niveles completados, las estrellas conseguidas en cada nivel, los niveles creados por el usuario. El sistema de progreso se encarga de gestionar estos datos y suministrarlos al sistema de guardado cuando es necesario.

El sistema de progreso utiliza principalmente dos estructuras de datos (ED) para almacenar la información.

- Una de las ED se encarga de almacenar los datos de cada categoría, donde se incluyen las estrellas conseguidas en cada nivel, el último nivel desbloqueado en esta categoría y las estrellas totales conseguidas.
- La otra ED utilizada almacena los niveles creados por el usuario. Como estos niveles se crean durante la ejecución, se almacenan los nombres de los archivos creados para poder cargarlos como niveles posteriormente.

Otro de los sistemas que se encarga de gestionar datos necesarios para el sistema de guardado es el sistema de tutoriales. Los tutoriales que ya se han visto, se almacenan en el sistema de tutoriales mediante una cadena *hash* codificando su contenido, así se confirma que el mismo tutorial no se guarde varias veces. El sistema de tutoriales solo actualiza los tutoriales que han sido vistos cuando se muestra un tutorial que esté marcado como "*save checkpoint*", de esta forma se puede hacer que el usuario vuelva a ver tutoriales ya vistos si no ha llegado a cierto punto.

El sistema de guardado utiliza los datos de los dos subsistemas explicados anteriormente para guardar el progreso de la partida. Una vez se dispone de estos datos, realiza una cadena hash y la almacena junto con los datos en el archivo de guardado. De esta forma, al cargar una partida si la cadena hash no coincide porque el usuario ha intentado modificar los datos guardados, se borra el progreso del jugador.

4.3.12. Toma de métricas y diseño de evaluación

Debido a que el PC es un campo aún en expansión, no existe una forma clara y acordada de evaluar las habilidades que componen el PC. Sin embargo, existen estudios que intentan definir estas habilidades y una forma de evaluar su avance de manera empírica (Román-González et al., 2019).

Articoding integra un sistema de analíticas (basado en el uso de un Tracker con soporte xAPI y en el entorno de analíticas Simva del grupo e-UCM) que permite capturar las interacciones de los jugadores para analizarlas ya sea en tiempo real o una vez completada la sesión de juego.

El Tracker[42] forma parte del paquete de analíticas del ecosistema RAGE (Realising an Applied Gaming Eco-system) del grupo e-UCM. Permite recolectar datos sobre las acciones del usuario en el juego y se encarga de enviarlos a un servidor web (también tiene un modo dual que además permite que dichas trazas se almacenen en modo local). La información se envía al servidor en tiempo real siguiendo el formato xAPI.

> "*Real-time analytics - It is highly desirable to access key analytics in real-time (or with minimal delay).*" (Freire et al., 2016, p. 21)

Por otro lado, Simva es una herramienta web que simplifica la realización de experimentos con juegos serios, y entre otras funcionalidades, permite gestionar actividades de presentación de cuestionarios a los jugadores y su correlación con los datos de analíticas recolectados. Esto facilita la validación de componentes educativos[43]. Se ha utilizado Simva para facilitar la visualización de los datos recogidos por el Tracker. Así como para la secuenciación de las encuestas pre- post- realizadas durante las pruebas con usuarios.

La integración de un sistema de analíticas permite, entre otras cosas: (1) depurar errores y mejorar las mecánicas de juego e interfaz según las interacciones y dificultades encontradas por jugadores; (2) evaluar la dificultad del videojuego,a través de las soluciones proporcionadas en los diferentes niveles, el coste en tiempo, bloques utilizados, etc. Incluso podría servir para tener una idea aproximada del conocimiento de los jugadores (la evaluación formal de dicho conocimiento es muy difícil y demostrar la validez de dicha evaluación es mucho más complejo que la creación del propio juego).

> "*A basic implementation of a Game Learning Analytics system would need to inspect how each player interacts with the game, storing detailed information about the interactions and the changes in the internal game state for further analysis.*" (Freire et al., 2016, p. 20)

Los datos que se recogen a lo largo del juego permiten recolectar datos que ilustran el uso que hacen los jugadores y si su interacción con el juego es la esperada. Se puede observar si el jugador medio tiene los comportamientos que se esperan y

[42] https://github.com/e-ucm/unity-tracker
[43] https://www.e-ucm.es/es/portfolio-item/simva/

toma las decisiones anticipadas por los desarrolladores. Si los jugadores usan ambos modos y cuánto tiempo pasan en cada uno, niveles que sean demasiado fáciles o que pocos jugadores superen, si las pistas realmente ayudan a los jugadores a resolver el nivel, el tiempo que pasan los usuarios jugando, etc.

La recogida de las interacciones del usuario y el uso de analíticas también permiten evaluar al jugador, las dificultades que tiene y si mejora a lo largo del juego. Mediante las trazas tomadas se puede obtener información sobre el tiempo que tarda el jugador en completar un nivel, los intentos que ha realizado hasta completarlo, la cantidad de pistas que utiliza para resolver un nivel, la cantidad de niveles que completa, la cantidad de tarjetas que borra o que genera, y cómo mueve las tarjetas para crear el código.

Además del análisis del comportamiento del jugador, el conjunto de trazas de interacción recogido permite rehacer paso por paso las acciones del jugador.

En todo caso las trazas deberían permitir saber qué ha pasado en el juego, o al menos sus situaciones y decisiones más relevantes, sin tener que observar al jugador o grabar sus interacciones.

4.3.13. Información xAPI y ejemplos de trazas

El tracker envía la información en formato xAPI. Este formato se basa en *Activity Stream* que es el estándar utilizado para el seguimiento de usuarios en redes sociales (Calvo-Morata, 2020). xAPI representa los datos de interacción como frases compuestas por sujeto, verbo y predicado. La información (Serrano-Laguna et al., 2017) se compone de un **actor** que realiza la acción, un **verbo** que representa las acciones realizadas, el **objeto** sobre el que se realiza la acción y **extensiones** como complementos de información que recaen en la acción (e.g., contexto en el que se realiza la acción).

> "*[...] xAPI defines extensions properties to establish further semantics whenever verbs and activity types are not enough*" (Serrano-Laguna et al., 2017)

Además, todas las trazas xAPI contienen un *timestamp*, el instante de tiempo en el que se ha generado la traza.

```
{
    "actor": {
        "name": "fybu",
        "account": {
            "homePage": "https://simva-api.simva.e-ucm.es",
            "username": "fybu"
        }
    },
    "result": {
        "score": {
            "raw": 3
        },
        "extensions": {
            "minimum_steps": true,
            "no_hints": true,
            "first_execution": true,
            "steps": 5
        },
        "success": true
    },
    "verb": {
        "id": "http://adlnet.gov/expapi/verbs/completed"
    },
    "_id": "60bf4b3e1f7061006ecb50b9",
    "object": {
        "definition": {
            "type": "https://w3id.org/xapi/seriousgames/activity-types/level"
        },
        "id": "https://simva-api.simva.e-ucm.es/activities/60bf4b3e1f7061006ecb50b9/types_6"
    },
    "timestamp": "2021-06-09T19:32:57.369Z"
}
```


Figura 30: Ejemplo de trazas recogidas en Articoding en formato xAPI

La traza de la Figura 30 presenta la siguiente información: El usuario o actor está identificado con el código “fybu” (*actor.name*); La acción que ha realizado es completar (*verb.id*) un nivel (*object.definition*) que además es el sexto de la categoría de tipos (*object.id*); Además como parte de las extensiones, encontramos que el jugador ha conseguido las 3 estrellas (*result.score.raw*) que se pueden conseguir por nivel y que corresponden a usar el mínimo de pasos (*result.extensions.minimum_steps*), a no usar pistas (*result.extensions.no_hints*) y a no equivocarse ninguna vez al obtener la solución ya que es la primera ejecución (*result.extensions.first_execution*).

```
{
    "actor": {
        "name": "fybu",
        "account": {
            "homePage": "https://simva-api.simva.e-ucm.es",
            "username": "fybu"
        }
    },
    "result": {
        "extensions": {
            "block_type": "variables_get",
            "code": "\r\n<block type=\"variables_get\" id=\"variables_get_brub\" x=\"30\" \
                    y=\"-280\">\r\n <field name=\"VAR\" id=\"tT8T=N7x38oJ/Ep)@%fP\" \
                    variableType=\"\">pasos</field>\r\n</block>",
            "level": "types_6",
            "action": "create"
        }
    },
    "verb": {
        "id": "http://adlnet.gov/expapi/verbs/interacted"
    },
    "_id": "60bf4b3e1f7061006ecb50b9",
    "object": {
        "definition": {
            "type": "https://w3id.org/xapi/seriousgames/activity-types/game-object"
        },
        "id": "https://simva-api.simva.e-ucm.es/activities/60bf4b3e1f7061006ecb50b9/variables_get_brub"
    },
    "timestamp": "2021-06-09T19:32:45.707Z"
}
```


Figura 31: Ejemplo de trazas recogidas de Articoding en formato xApi

La traza de la Figura 31 muestra la siguiente información: El actor está identificado con el código “fybu” (*actor.name*); La acción que se ha realizado es interactuar (*verb.id*) con un gameobject (object.definition) que es una tarjeta de variable(*object.id*). Además las extensiones indican el tipo de bloque (*result.extensions.block_type*) el nivel actual, que es el nivel 6 de la categoría tipos (*result.extensions.level)*, la acción realizada (*result.extensions.action)* que es crear una tarjeta y por último el estado del codigo (*result.extensions.code)*.

El estándar xApi permite definir diferentes perfiles de aplicación (*application profiles*) enfocados a usos específicos en un cierto dominio, como puede ser la visualización de vídeos o la compleción de un temario en un MOOC. En este caso, el Tracker implementa el perfil de aplicación de juegos serios, que cuenta con un amplio catálogo de verbos y objetos relacionados con la captura de la interacción de un jugador en un videojuego. En este juego se utilizan los siguientes:

- ***Initialized*, *Progressed* y *Completed***: este tipo de traza se utiliza para indicar cuando un elemento de juego que implique progresión ha empezado, ha progresado o ha terminado. Como completables por defecto se ofrecen:
    - ***Game***: el principio y la finalización del propio juego.
    - ***Session***: similar al anterior, una sesión o ciclo de juego completo.
    - ***Level***: un nivel de juego, que depende del juego.

    - ***Completable***: tipo genérico, usado para los completables no recogidos en los tipos anteriores (e.g. categorías de niveles en Articoding) .
- ***Interacted***: esta traza es utilizada para todo tipo de interacción con los GameObjects de Unity. Los eventos que se ofrecen son:
    - ***NPC***[44]: cuando se interactúa o cambia el estado de un NPC.
    - ***Enemy***: cuando se interactúa o cambia el estado de un enemigo.
    - ***Item***: útil para cuando se ha interactuado con un artículo específico.
    - ***GameObject***: tipo genérico para referirse a objetos que no pertenecen a las categorías anteriores (e.g. objetos del tablero en Articoding).
- ***Accessed***: esta traza se usa cuando se accede a alguna zona del juego. Estas zonas pueden ser:
    - ***Screen***: pantalla del juego. Diferente en cada juego.
    - ***Área***: área del juego. Diferente en cada juego.
    - ***Zone***: zona de juego. Similar al área.
    - ***Cutscene***: cinemática del juego.
    - ***Inventory***: acceso a un inventario como una mochila.
    - ***Accesible***: tipo genérico, que engloba las zonas diferentes a las indicadas anteriormente (e.g. categorías de niveles en el menú de Articoding).

Para usar el Tracker sólo hay que añadir el paquete al proyecto de Unity e invocarlo mediante su API que está estructurada de manera que sea fácil de usar. En cada sesión de juego, es necesaria una credencial para poder identificar al usuario, en este caso la credencial consiste en un token alfanumérico de 4 caracteres. En las trazas, este token se ve reflejado en la propiedad "name" del actor. Para invocar al Tracker, simplemente hace falta hacer una llamada como la siguiente:

```
TrackerAsset.Instance.GameObject.Used("my_gameobject_id");
```

En esta traza se enviará la información de que el usuario ha usado el objeto del juego identificado por `my_gameobject_id.`

Para generar una traza como la presentada en la imagen anterior, es necesario indicar las extensiones de la traza antes de hacer la llamada al Tracker de la siguiente forma:

```
TrackerAsset.Instance.setVar("steps", GetCurrentSteps());
TrackerAsset.Instance.setVar("first_execution", IsFirstRunStarActive());
TrackerAsset.Instance.setVar("minimum_steps", IsMinimumStepsStarActive());
TrackerAsset.Instance.setVar("no_hints", IsNoHintsStarActive());
TrackerAsset.Instance.Completable.Completed(levelName, Completable.Level, true, stars);
```

[44] NPC (Non Playable Character), un personaje del juego que no se controla por el jugador.

Esta traza indica que el jugador ha completado un nivel, además, en este ejemplo, se muestra la forma en la que se añaden extensiones a una traza. Haciendo llamadas antes de lanzar una traza a:

```
TrackerAsset.Instance.setVar("var_name",value);
```

se consigue incorporar a la traza toda la información extra que sea necesaria, consiguiendo mayor riqueza en la calidad de la información que se traza.

Haciendo uso de la funcionalidad que ofrece el Tracker descrita anteriormente, a continuación se muestra una tabla detallada con las trazas que se recogen en Articoding.

Información recogida	**Verbo xAPI usado**	**Tipo objeto xAPI**	**Extensiones**
Inicio del juego	Initialized	Game	Resolución Pantalla completa Idioma
Finalización de un nivel de juego	Progressed	Game	
Finalización del último nivel del juego	Completed	Game	
Acceso a la sección de Jugar (Variables, Bucles...)	Accessed	Screen	Categoría elegida (si la hay)
Acceso a la sección de Tutoriales (en el menú)	Accessed	Screen	Desde menú Categoría elegida (si la hay)
Acceso a la sección de Modo Creación	Accessed	Screen	
Inicio de un nivel de juego (con estado inicial del código)	Initialized	Level	Código inicial
Pulsado botón de ejecución del código	Interacted	GameObject	
Ejecución del código (con estado del código)	Progressed	Level	Estado actual del código
Finalización de un nivel de juego (victoria con info de las 3 estrellas o derrota)	Completed	Level	Pasos realizados Estrellas
Inicio del primer nivel de una categoría	Initialized	Completable	

Finalización de un nivel de una categoría (no el último)	Progressed	Completable	
Finalización del último nivel de una categoría	Completed	Completable	
Pulsado botón de reinicio de nivel	Interacted	GameObject	
Reintento de un nivel de juego	Initialized	Level	
Salida al menú principal (desde un nivel de juego)	Completed (fail)	Level	No. de pasos Nivel Estado actual del código
Creación de una variable	Interacted	GameObject	Nombre de la variable Tipo de bloque Action Nivel
Borrado de una variable	Interacted	GameObject	Nombre de la variable Tipo de bloque Action Nivel
Generación de una tarjeta	Interacted	GameObject	Tipo de bloque Action ID del bloque
Eliminación de una tarjeta	Interacted	GameObject	Tipo de bloque Action ID del bloque
Encaje de una tarjeta con otra	Interacted	GameObject	Tipo de bloque Action ID del bloque Info de conexión
Desencaje de una tarjeta con otra	Interacted	GameObject	Tipo de bloque Action ID del bloque Info de conexión
Movimiento de una tarjeta	Interacted	GameObject	Tipo de bloque Action ID del bloque Posición
Cambio del valor de una variable (textos, números, etc.)	Interacted	GameObject	Tipo de bloque Action ID del bloque Valor antiguo

			Valor nuevo
Duplicación de una tarjeta	Interacted	GameObject	Tipo de bloque Action ID de la nueva tarjeta
Uso de una pista	Interacted	GameObject	No. de pistas restantes Nivel
Acceso/Cierre del panel de Tutoriales (desde un nivel del juego)	Accessed	Screen	
Categoría de Tutorial elegida (en un nivel)	Accessed	Accesible	Categoría Desde: nivel
Inicio del modo creación	Initialized	Completable	
Pulsado botón de reinicio de nivel de creación	Interacted	GameObject	
Reinicio del modo creación	Initialized	Completable	
Salida al menú principal (desde el modo creación)	Completed (fail)	Completable	
Validación completada	Completed	Completable	No. de pasos
Generación del tablero (con el tamaño elegido)	Interacted	GameObject	No. de filas No. de columnas
Elemento de tablero cogido desde el tablero	Interacted	GameObject	ID del elemento Acción Posición antigua Posición nueva
Elemento de tablero cogido desde el creador	Interacted	GameObject	ID del elemento Acción Posición
Elemento de tablero colocado	Interacted	GameObject	ID del elemento Acción Posición Rotación
Elemento de tablero eliminado	Interacted	GameObject	ID del elemento Acción Posición Rotación
Elemento de tablero girado	Interacted	GameObject	ID del elemento Acción

			Posición Rotación
Celda cambiada	Interacted	GameObject	Posición Nuevo estado
Estado del objeto cambiado	Interacted	GameObject	ID del elemento Nombre del valor cambiado Valor antiguo Valor nuevo
Pulsado botón de cambio de modo (en el creador)	Interacted	GameObject	
Cambio de modo (con el estado del tablero si se cambia al modo de validación)	Accessed	Screen	Modo Estado del tablero
Botón de reseteo de vista (en nivel o creación)	Interacted	GameObject	
Minimización de panel de Victoria	Interacted	GameObject	
Minimización de panel de Derrota	Interacted	GameObject	
Aparición de un pop-up de tutorial	Initialized	Dialog fragment	Contenido
Cierre de un pop-up de tutorial	Completed	Dialog fragment	Contenido
Menú de configuración interactuado	Interacted	GameObject	Abierto o cerrado
Cambio de lenguaje	Interacted	GameObject	Nuevo lenguaje
Cambio de resolución	Interacted	GameObject	Nueva resolución
Cambio a modo ventana/pantalla completa	Interacted	GameObject	Modo
Botón de créditos pulsado	Interacted	GameObject	
Botón de salir pulsado	Interacted	GameObject	

4.3.14. Evaluación del usuario

Con los datos recogidos por el Tracker durante la ejecución del videojuego es posible evaluar el progreso del jugador. Algunos de los datos de interacción más importantes para entender mejor al jugador y las dificultades que encuentra son: El tiempo que se tarda en completar un nivel puede compararse con el tiempo medio del resto de estudiantes para comprobar si el nivel está siendo más difícil de lo normal; la cantidad de tarjetas creadas y eliminadas durante la resolución del nivel. Muchas

eliminaciones y/o creaciones pueden suponer que el objetivo del nivel no ha quedado claro o que el estudiante está recurriendo a la prueba - error para solucionar el nivel; otra de las señales de que el estudiante intenta forzar la solución es el número de ejecuciones en cortos periodos de tiempo y con pocas modificaciones entre estas. Por último, el tiempo entre modificaciones de código puede indicar si el usuario tiene claro las instrucciones a seguir para resolver el nivel o si por el contrario tiene dudas al respecto.

4.4. Desarrollo y herramientas de trabajo

Para facilitar el desarrollo del juego, se ha hecho uso de un conjunto de herramientas y proyectos de repositorios externos que ha permitido agilizar el proceso. Estas herramientas no sólo han sido de gran utilidad en el desarrollo de Articoding sino que algunas de ellas también han formado parte del desarrollo de la Mansión Paranormal. En esta sección se exponen las herramientas de trabajo utilizadas y de qué manera se han aplicado durante el periodo de desarrollo de Articoding.

4.4.1. Entorno de desarrollo (Unity)

Para el desarrollo del juego se ha utilizado Unity, un motor de videojuegos que ofrece una interfaz visual para desarrollar videojuegos de una manera más sencilla. Ha sido la herramienta más fundamental para el desarrollo. Gracias a ello y a su gran comunidad de desarrolladores, se ha podido agilizar bastante el ritmo de desarrollo.

4.4.2. Pivotal Tracker

Se ha decidido utilizar Pivotal Tracker haciendo uso de los conocimientos obtenidos en la asignatura de *Metodologías Ágiles de Producción* del grado. Pivotal Tracker es un sistema de planificación de proyectos que ofrece varias configuraciones de planificación: pudiendo configurar las duraciones de los sprints, elegir el posicionamiento de los distintos hitos, clasificar de tareas en categorías y asignar tareas a una o más personas, entre otras tantas funcionalidades.

Esta herramienta ha permitido planificar los hitos de desarrollo de muchos de los proyectos que se han realizado en la carrera. Es por ello, y por la experiencia previa, que se decidió optar por esta herramienta.

4.4.3. Git

Se ha utilizado Git para gestionar las distintas versiones del proyecto. Git es un sistema de control de versiones que ayuda a gestionar los proyectos. Git guarda un registro de cambios de los archivos coordinando las modificaciones propias con las modificaciones que haya podido realizar otra persona.

En el caso de este proyecto, se ha utilizado Git a través de GitHub que aloja proyectos que utilizan el sistema de control de versiones Git. De esta manera, todos los cambios que se realizaban en el proyecto de Unity se coordinaban con los cambios realizados por el resto de los integrantes. Gracias a esta herramienta, se ha podido dividir y distribuir el trabajo de manera que se pudiera realizar de forma paralela y sin conflictos a la hora de juntarlo.

4.4.4. Repositorios externos

Para facilitar el desarrollo del proyecto, se ha optado por utilizar proyectos de terceros de código abierto. A continuación, se listan los repositorios externos utilizados en el proyecto.

4.4.4.1. Blockly en Unity (*ublockly*)

Uno de los grandes problemas que se presentaron fue la necesidad de crear un lenguaje de programación visual por bloques familiar para los usuarios a los que se dirige el juego.

Entrevistando a Jesús Moreno León, se llegó a la conclusión de introducir un sistema similar, sino igual, al de Scratch. Ya que, como comentó en la entrevista, es un programa muy utilizado en las asignaturas de las TIC[45] en los centros educativos españoles.

Investigando se encontró un repositorio público de GitHub de imagicbell[46] llamado *ublockly*[47]. Este repositorio es lo más parecido a lo que se buscaba y por ello se decidió contactar con la autora para colaborar y que diera soporte. Además, se tuvo la suerte de que unos estudiantes portugueses se interesaron también por el proyecto de *ublockly.* Gracias a ello, se colaboró con la autora que resolvió varios *bugs* que ambos equipos encontrábamos y, en consecuencia, mejoramos la integridad de dicho repositorio y del juego que se apoya en dicho código.

[45] Tecnologías de la Información y la Comunicación.
[46] https://github.com/imagicbell
[47] https://github.com/imagicbell/ublockly

Generalmente, la comunicación con la autora del repositorio se desarrollaba mediante github. Cuando el equipo de desarrollo encontraba un *bug* en *ublockly* o era necesaria una funcionalidad que no estaba implementada, se creaba una *issue* en el repositorio de *ublockly* explicando la situación. En alguna ocasión el grupo de estudiantes portugueses planteaba *issues* cuya solución sería de ayuda posteriormente (e.g., *Issue* con los archivos de localización[48]).

Aunque la mayoría de *issues* planteadas fueron resueltas, en algunos casos la implementación de estas funcionalidades era más complicada. Sin embargo, la autora aportó las indicaciones necesarias para llevar a cabo la implementación (e.g., búsqueda de errores antes de ejecutar[49] o la implementación de un modo "debug"[50]). Algunas de estas funcionalidades no se consideraron necesarias por cambios en el diseño del juego. Aún así, siguen siendo posibles formas de mejorar el repositorio.

Por último, también fue necesario implementar funcionalidades más específicas a nuestro proyecto como el contador de bloques o permitir una cierta cantidad de bloques por nivel. En estos casos, fue necesario modificar o añadir funcionalidades a la librería. Esto implicaba conocer el funcionamiento interno y la estructura general de una librería externa de considerable extensión y complejidad. Esta fue una de las tareas más importantes y costosas que conllevó el hecho de elegir esta librería. Aunque gracias a ello, juzgar la viabilidad de ciertas mejoras y modificaciones resultó más sencillo, y se agilizó el proceso de desarrollo.

4.4.4.2. Tracker (e-UCM)

Puesto que se había trabajado previamente con un tracker en el juego de *La Mansión Paranormal* se decidió utilizar el mismo tracker. Se ha utilizado para tomar datos de juego, mejorar la experiencia del usuario y encontrar posibles deficiencias en el diseño de los distintos niveles.

El tracker que se decide utilizar es *unity-tracker*[51], un tracker cuyo código pertenece al proyecto RAGE[52] (Realising an Applied Gaming Ecosystem) y desarrollado por el equipo de investigación dirigido por Baltasar Fernández Manjón, con los que ya se había trabajado y colaborado anteriormente.

Como ya se había utilizado el tracker, fue sencillo incorporarlo al juego y configurarlo para que empezara a guardar los datos que se requerían.

---

[48] https://github.com/imagicbell/ublockly/issues/8
[49] https://github.com/imagicbell/ublockly/issues/14
[50] https://github.com/imagicbell/ublockly/issues/11
[51] https://github.com/e-ucm/unity-tracker
[52] http://rageproject.eu/

4.4.5. Internacionalización y localización

Uno de los objetivos principales de este proyecto era internacionalizar y localizar el juego para que más personas pudieran acceder a él. Debido a que las pruebas con usuarios se iban a realizar con usuarios hispanohablantes, se decidió que el lenguaje original del juego fuese el español. Sin embargo, hoy en día es muy importante que los proyectos que se encuentran en la red estén en inglés. Por esto, para dar una mayor visibilidad y aumentar el alcance del juego, se consideró necesaria la opción de adaptarlo al inglés. Esto permite una accesibilidad mayor en el ámbito educativo.

Para ello se han usado los conocimientos obtenidos en la asignatura de *Usabilidad y Análisis de Juegos (UAJ)* y, además, se ha presentado esta parte del desarrollo como proyecto final de dicha asignatura.

Para realizar esta tarea se ha hecho uso del paquete de Unity *Localization 0.10.0-preview*[53], donde se proporcionan herramientas para facilitar la localización de textos, la localización de texturas y la generación de tablas de localización. Concretamente, se han realizado las siguientes tareas:

- Localización de textos fijos
- Localización de texturas (imágenes de tutoriales)
- Internacionalización y localización de textos dinámicos
- Sincronización de la localización de *ublockly*
- Sincronización de estados iniciales de cada nivel

Este paquete ofrece una gran escalabilidad. Esto permite añadir un nuevo idioma con bastante facilidad. En el Anexo G se puede encontrar una memoria más detallada de esta parte del desarrollo.

4.4.6. Canales de comunicación

Durante todo el periodo de trabajo ha habido una constante comunicación entre los distintos miembros con distintos canales de comunicación. Se ha utilizado Discord[54] como el programa de comunicación principal. Ya que se hacían reuniones casi semanales para la puesta en común de los avances, tener un canal único y privado para el desarrollo del proyecto ha sido fundamental.

En Discord, además de realizar reuniones semanales, se apuntaban todas las decisiones que se tomaban para transcribirlas a documentos más adelante, se

[53] https://docs.unity3d.com/Packages/com.unity.localization@0.10/manual/index.html
[54] https://discord.com/

comunicaban los errores encontrados en el juego, se avisaba de cuando se realizaba un cambio en el repositorio del proyecto, etc. Discord ha funcionado también como una herramienta para el trabajo colaborativo con los autores de uAdventure (durante el desarrollo de La Mansión Paranormal) y con los autores del tracker a la hora de integrar analíticas en ambos juegos. Ha sido un punto de unión para la resolución de dudas sobre el uso de estas herramientas y para reportar problemas encontrados.

Con respecto a la comunicación con el director y codirector del proyecto, los canales de comunicación utilizados han sido Google Meet[55], una plataforma de reuniones telemáticas, y el correo electrónico.

Google Meet se ha utilizado como medio para las reuniones semanales con los directores del TFG. En estas reuniones se mostraban los avances del trabajo, se planificaban los siguientes pasos y se recibía retroalimentación constante. También se ha utilizado para llevar a cabo las entrevistas a profesores como Hénar Lanchas López, Gaspar Ferrer Soria y Jesús Moreno León.

El correo electrónico también ha sido una herramienta fundamental para la comunicación con el director del TFG. Se ha utilizado para establecer fechas y horas de reunión, preguntas del estado del desarrollo de los juegos, retroalimentación, entre otras cosas.

4.5. Limitaciones y dificultades

A lo largo del desarrollo se han tenido que tomar ciertas decisiones en el diseño y el desarrollo debido a la ambición de un proyecto tan grande y cierta limitaciones que han ido surgiendo en el proceso. La primera limitación y la más importante es la cantidad de tiempo disponible para realizar el trabajo. Aunque se han cumplido, con creces, los objetivos gracias a la planificación realizada, un mayor tiempo de desarrollo habría permitido ampliar el contenido de niveles, pulir la experiencia de usuario y realizar pruebas de usuario más exhaustivas junto al análisis y visualización de los datos de interacción recogidos en dichas pruebas.

Por otro lado, al principio se planteó desarrollar el sistema de lenguaje por bloques sin utilizar contenido de terceros ya que nada de lo que se había visto se ajustaba a las necesidades iniciales. Sin embargo, la limitación de tiempo hacia esta opción inviable, por lo que finalmente se encontró y optó por usar el repositorio externo de *ublockly* mencionado anteriormente.

[55] https://meet.google.com/

El uso de este repositorio externo ha traído consigo otras limitaciones adicionales. El repositorio estaba incompleto y los últimos cambios habían sido en febrero de 2018. Mucha de la funcionalidad que se buscaba estaba planteada pero no implementada. Esto hizo que se descartada del diseño inicial la posibilidad de poder depurar la ejecución bloque a bloque de código dentro del juego . Además, la autora del repositorio había dejado disponible documentación del repositorio en el idioma chino, lo que ha dificultado el uso de su proyecto. A partir del trabajo y el uso del repositorio se ha contactado con la autora del proyecto de ublocky para resolver algunos de los problemas que han ido surgiendo. Gracias a esta colaboración y al interés que se ha mostrado por ublocky, el repositorio ya dispone de documentación en inglés y se sigue actualizando.

Por último, otra de las dificultades afrontadas ha sido encontrar una idea original que tuviera cierta escalabilidad y que gustara, no sólo a los autores de este trabajo, si no también a los profesionales y expertos en la educación, para conseguir así videojuego que sirva como herramienta educativa en las aulas. Y es que, como se ha mencionado en la sección 4.1, la idea inicial del videojuego era distinta.

4.6. Pruebas con expertos, profesores y alumnos. Resultados preliminares

Una vez concluido el desarrollo principal de Articoding, se planteó un periodo de pruebas de usuario, similar al planteado en el desarrollo de la Mansión Paranormal. En este periodo, inicialmente se realizaron iteraciones cortas donde se enviaba la build al director y codirector. Estos aportaban feedback sobre elementos a pulir y rápidamente se generaba una nueva versión para repetir el proceso.

Una vez se consideró que los principales problemas habían sido pulidos, se inició un proceso de pruebas formativas con expertos externos. En una primera fase se envió el juego a cuatro expertos (Henar Lanchas López, Jesús Moreno León, Marcos Román González y Gaspar Ferrer) sin recogida de analíticas en Simva ni cuestionarios. Una vez se recibió el feedback de estos profesores, se realizó una nueva versión para enviar a los profesores que se habían registrado a través del formulario de un nuevo artículo promocionando la participación en la prueba del juego de la página web de Programamos[56]. En esta versión del proyecto se realizó un estudio de Simva para poder visualizar las analíticas y se incluyeron cuestionarios pre-post para obtener la opinión de los profesores. En el formulario de registro se

56 https://programamos.es/oportunidad-para-probar-y-validar-un-nuevo-juego-serio-narrativo-sobre-pensamiento-computacional/

registraron 27 profesores para probar el juego. Estos profesores son de distintas comunidades de España como Castilla la Mancha, País Vasco, Cataluña, Murcia y La Comunidad de Madrid. También se han registrado profesores de otros países hispanohablantes como Perú, Argentina y Chile, e incluso países no hispanohablantes como Dubai o Reino Unido.

Además cabe destacar que para facilitar el uso del juego, durante este periodo de pruebas, se realizó una guía docente en la que se explica el concepto básico del juego, los elementos principales y las principales mecánicas del juego. Esta guía docente está disponible en el anexo E. Con este mismo motivo, también se realizó un solucionario con las soluciones óptimas de todos los niveles del juego.

Debido al tiempo que supone completar todos los niveles de Articoding, que puede superar las 2 horas según la experiencia del jugador con lenguajes de bloques, el diseño de las pruebas de evaluación debía ser distinto al usado en la Mansión Paranormal. En este caso, el cuestionario posterior a la sesión de juego no podía ser lanzado al acabar el juego (porque no todos los profesores lo terminarían), por esto se incluyó un botón en el juego para acceder al cuestionario final. Este botón permite que los profesores puedan usar el juego en varias sesiones diferentes, dedicando el tiempo que puedan. Para estas sesiones experimentales, junto a la versión de Articoding se adjuntó la guía docente y un documento con instrucciones a seguir durante las sesiones de juego. Estas instrucciones muestran la forma de ejecutar y acceder al juego (introduciendo el código asignado) y cómo realizar los cuestionarios del juego. Además se pide a los profesores que, al menos, jueguen a tres niveles de cada categoría (para que generen una cantidad de datos significativa) antes de acceder al cuestionario final y en un máximo de 10 días para tener sus comentarios lo antes posible. Sin embargo, se les recomienda que jueguen todos los niveles de cada categoría si tienen tiempo.

A fecha de la entrega de esta memoria y antes de acabar este periodo de 10 días, 8 de los profesores han jugado el juego todos ellos con al menos 2 de las 5 categorías jugadas. De estos 8 profesores, se ha recibido realimentación de 2 de ellos tras jugar al juego.

Aunque no ha finalizado el periodo dado a los profesores para probar el juego, se ha hecho un análisis parcial de los resultados obtenidos. La gran mayoría de profesores poseen un perfil similar entre ellos. Una gran parte juega a videojuegos al menos una hora a la semana. Además, todos ellos imparten clases en campos relacionados con la informática con al menos 10 años de experiencia docente. Únicamente tres dicen no haber aplicado nunca videojuegos en sus clases como herramienta de aprendizaje. Cabe destacar que todos los profesores afirman que saben programar.

Analizando la realimentación de los profesores que han completado el juego, se pueden destacar dos observaciones importantes. En primer lugar, los profesores consideran que los niveles propuestos son difíciles pero que pueden ser interesantes para alumnos más mayores de cursos a partir de 4º de la ESO. En segundo lugar, creen que la introducción del juego sobrecarga al usuario con tutoriales y pueden abrumar a los jugadores. Aunque estos datos son parciales, pueden ser útiles de cara a las pruebas con alumnos.

Como parte importante del proceso de validación, se van a realizar pruebas con estudiantes de 1º, 2º de la ESO y 1º de BACHILLERATO del Colegio La Inmaculada Escolapias Puerta de Hierro[57] los días 15, 16 y 17 de junio. Para estas pruebas, se ha realizado un nuevo estudio adaptado a los estudiantes, con diferentes encuestas pre-post y una nueva versión de juego con la progresión normal del juego.

Figura 32: Colegio La Inmaculada Escolapias Puerta de Hierro

A fecha de la entrega de esta memoria, se ha realizado el primer día de pruebas dividida en 5 sesiones de una hora con cinco grupos diferentes (3 grupos de 2º de la ESO, un grupo de 1º de la ESO y un grupo de 1º de BACHILLERATO). En el caso del grupo de 1º de la ESO la mayoría de alumnos consiguió completar la primera categoría. Por otro lado, la mayoría de alumnos de todos los grupos de 2º de la ESO consiguieron completar varios niveles de la segunda categoría. Por último, los alumnos de 1º de BACHILLERATO, en general, llegaron a completar más de la mitad de la segunda categoría. Todos estos alumnos han completado un cuestionario inicial sobre su edad, sexo y conocimiento inicial de programación. Los alumnos de

[57] http://cliph.es/

1º de BACHILLERATO, como sólo jugaban una hora en un único turno, también han completado el cuestionario final dando la opinión sobre el juego.

Estas sesiones han aportado mucha realimentación, tanto cuantitativa mediante las trazas recogidas como cualitativa con la observación del comportamiento de los usuarios. Examinando el comportamiento de los usuarios se han podido observar una serie de problemas y dificultades comunes a la mayoría de alumnos. Por un lado, el concepto de variables resultaba muy complejo de entender solo con los tutoriales propios del juego. En ocasiones los nombres de las variables resultaban confusos, algunos de los estudiantes pensaban que cambiar el valor de la variable llamada *Desplazamiento* era suficiente para desplazar un elemento del tablero y no recordaban la funcionalidad de las tarjetas de acciones. Sin embargo, con el apoyo de un profesor o una explicación externa, los alumnos entienden la base del concepto y son capaces de aplicarlo en los siguientes niveles. Por estos motivos, consideramos que el juego es útil para introducir estos conceptos en el aula.

Otra de las principales dificultades que los alumnos han encontrado en el juego ha sido la interacción con algunos elementos de la interfaz. Una gran cantidad de alumnos de todos los grupos no sabían cuándo se había seleccionado una categoría en el menú principal por lo que puede ser necesario dar más realimentación al jugador en este aspecto. También han encontrado dificultades para rellenar los huecos de las tarjetas, en algunos casos no entendían que dejar un hueco vacío hacía que la ejecución fuese errónea. Además en algunos casos intentan rellenar estos huecos haciendo click en el hueco pensando que era un desplegable.

Cabe destacar que pocos alumnos han hecho uso del sistema de pistas, o bien porque no han visto el botón para usar una pista, o porque no les han sido de utilidad. Esto se debe a que las pistas sugieren movimientos y giros de los elementos en el tablero pero no ayudan a completar el código. Esta observación fue mencionada también por un profesor y se tiene en cuenta en el trabajo futuro.

Uno de los principales problemas detectados, y que puede influir en los puntos anteriores, es la cantidad de tutoriales que aparecen en la sección inicial del juego. Además, este fue uno de los principales comentarios durante las pruebas con profesores. El nivel de atención y/o comprensión de los tutoriales es bajo, muchos no comprenden lo que el tutorial quiere explicar o incluso los saltaban. Sin embargo en ciertos casos, estos tutoriales eran eficaces y los alumnos entendían la funcionalidad que se explicaba, estos casos se daban generalmente en niveles con menos carga de tutoriales o en elementos donde los tutoriales eran más resumidos. En lo que respecta a lo mencionado anteriormente, también es importante destacar que el sistema de revisión de tutoriales ha sido usado por pocos de los usuarios, aunque en algunos casos esto se debía a que no sabían que podían usar este sistema, algunos usuarios no consultaron el temario incluso cuando se les informó de su existencia.

El comportamiento general que han tenido los alumnos a la hora de afrontar los problemas ha sido de prueba y error. Aunque esto supusiera la pérdida de una de las estrellas, la gran mayoría de alumnos parecen preferir resolver el problema por pasos (comprobando el funcionamiento de cada paso) antes que intentar resolverlo en la primera ejecución. Esto puede haberse visto influenciado por el conocimiento previo con Scratch o a otros factores externos.

En cuanto a la recepción de los jugadores, ha sido buena. En los niveles iniciales, la motivación no era muy alta. Sin embargo, una vez que los niveles comienzan a complicarse, la motivación aumenta. Además, la competitividad entre compañeros ha creado un mayor interés. También se ha podido observar cierta cooperación entre los estudiantes aunque en principio es algo positivo, es necesario controlar estos comportamientos para evitar que compartan soluciones con errores

conceptuales. Los alumnos de cursos más avanzados o con mayor conocimiento matemático parecen entender de forma más rápida el concepto de variables y, en general, construyen más fácilmente las abstracciones necesarias para completar los niveles. Por último es necesario destacar que el rendimiento de los equipos en los que se han realizado las pruebas no ha sido óptimo (e.g., equipos bastante antiguos y de poca potencia de cálculo) y ha provocado que algunos de los alumnos hayan tenido una peor experiencia de juego.

Debido a que se ha planteado un experimento secuenciado del juego, pasando por todos los niveles, no se ha evaluado el modo de creación con los alumnos. Tampoco se disponía de alumnos con un amplio conocimiento de Scratch como para saltarse niveles. Esta parte queda como trabajo futuro para posteriores evaluaciones.

Figura 34: Imagen de las pruebas en el colegio 2

4.7. Conclusiones

Del desarrollo de Articoding y las pruebas realizadas con expertos se han obtenido las siguientes conclusiones:

- Es importante presentar un mismo sistema de representación para los niveles de un juego con propósito educativo. Ya que aporta coherencia y claridad al proceso de aprendizaje.
- Es necesario estructurar el contenido que se enseña de una forma coherente y que maximice los resultados del aprendizaje. Estos conceptos se pueden ver aplicados en la ordenación de categorías enseñadas en el juego.

- Al igual que en el desarrollo de la Mansión Paranormal, es importante realizar ciclos cortos de desarrollo de manera iterativa y con feedback externo al equipo de desarrollo. Para asegurar un producto pulido y que cumpla con los objetivos propuestos.

En general, la realimentación obtenida en la evaluación formativa con expertos y profesores, aunque todavía limitada, ha sido muy positiva. Esto ha permitido mejorar la versión final. Además se ha realizado una evaluación sumativa con alumnos reales en un entorno de clase en un colegio que ha sido muy gratificante y con resultados también parciales pero prometedores.

Capítulo 5

Conclusiones, contribuciones y futuro trabajo

"Ese hombre va a luchar contra nosotros cada día y cada hora
hasta el final de la guerra." - James Longstreet

Consideramos que el proyecto ha conseguido los objetivos planteados de forma muy satisfactoria. Se han creado dos juegos serios para promover tanto el pensamiento computacional como la programación: La Mansión Paranormal y Articoding. Además dichos juegos se han evaluado formativamente con expertos y con profesores e incluso Articoding se ha probado con alumnos en un entorno real como es un colegio.

Es un proyecto que ha exigido un gran esfuerzo ya que ha tenido un componente de investigación importante en aspectos como el pensamiento computacional, el uso de juegos serios para su promoción en entornos escolares o las analíticas de aprendizaje aplicadas a juegos (e incluso estándares como xAPI). Se han tenido que leer muchos artículos de investigación además de buscar y revisar diversos juegos existentes en este campo.

El desarrollo ha sido complejo y ha exigido no sólo esfuerzo sino también aprender nuevos conceptos y herramientas. Se han usado dos herramientas, por un lado uAdventure pero combinado con puzles y minijuegos desarrollados en Unity para el desarrollo de La Mansión Paranormal y por otro lado la inclusión de un intérprete de ublockly, disponible como código abierto, que ha sido clave para el desarrollo de Articoding. Para facilitar el uso de estas dos herramientas, ha sido necesario entender su funcionamiento y familiarizarse con su estructura. Esto ha permitido detectar diferentes debilidades o fallos en estos proyectos y mejorar tanto el juego como las propias herramientas.

Hasta donde ha sido posible, se ha tratado de evaluar el juego con expertos, con profesores e incluso con alumnos para probar que son aplicables en un entorno de clase real. Este es un aspecto no muy habitual en un proyecto académico como es un trabajo de fin de grado, debido a la complejidad inherente de hacer todo el diseño experimental, pero sobre todo a la dificultad de encontrar a los sujetos para realizar la experimentación y a la escasez de tiempo desde que se tiene el producto final hasta

que se tiene que entregar. No obstante, en este caso con el apoyo de nuestros directores, hemos logrado hacerlo y estamos muy satisfechos del resultado. Esto hace que los juegos resultantes, que están disponibles como código abierto en *github,* sean herramientas útiles para los profesores a la hora de motivar a sus alumnos en el aprendizaje del pensamiento computacional y de la programación.

También hay que destacar que hemos tenido que hacer un esfuerzo especial al tener que contrastar ideas con expertos educativos y que eso nos ha llevado a tener que desarrollar productos que no habíamos anticipado. Por ejemplo, aunque sí teníamos claro que había que hacer juegos que se pudieran jugar sin tener que leerse un manual previo, no habíamos considerado que habría que hacer guías docentes para los profesores sobre cómo usar los juegos en la clase. Si queríamos que nuestros juegos fueran utilizados por un amplio rango de profesores, no podíamos depender de su conocimiento de los conceptos de juegos educativos y para eso hemos tenido que hacer unas guías detalladas y más adaptadas a sus conceptos y vocabulario.

Por último, como parte de la validación del juego Articoding se han planteado pruebas sumativas con alumnos del Colegio La Inmaculada Escolapias Puerta de Hierro. Estas pruebas se realizarán los días 15, 16 y 17 de junio de 2021 con 6 clases de estudiantes de 1º, 2º de la ESO y 1º de BACHILLERATO (con un total de 13 horas de práctica de los alumnos con el juego). Debido a la fecha de entrega, este aspecto está descrito parcialmente en el capítulo anterior, pero no está incluido de forma completa en la memoria.

5.1. Contribuciones

En este apartado se analiza con más detalle algunas de las contribuciones y cómo este trabajo ha contribuido al cumplimiento de los objetivos propuestos al comienzo del desarrollo del trabajo:

- **Analizar la situación actual del PC**. En el capítulo 2 se hace un análisis profundo sobre el PC en la actualidad, con especial hincapié en la educación. Se ha comprobado como el PC cada vez es más importante debido a la generalización de las tecnologías y su uso en cada vez más facetas de la vida diaria. Y por supuesto, en la escuela el PC se usa en las distintas disciplinas de STEM, pero sobre todo, en la programación.

  Es por ello que se decidió realizar un amplio estudio y análisis de diversos juegos, entre ellos algunos que utilizan el PC para enseñar conceptos de programación. De este estudio, se puede destacar el uso de bloques de código (programación visual) y el movimiento por casillas para el planteamiento de problemas en la mayoría de juegos analizados. Esto ayudó bastante a

formalizar la idea de cómo iban a ser los niveles de Articoding, pero se intentó dar otra aproximación para destacar entre los demás juegos. Por ejemplo, es bastante novedosa la presentación temprana de las variables y tratar de que los jugadores tengan una idea más clara del estado del juego proporcionado no sólo por el elemento móvil sino también por el estado de otros objetos (e.g., espejos). Es por esto que la idea final consiste en combinar el movimiento de objetos por casillas con el uso de espejos para reflejar láseres o que haya casillas con posibles estados diferentes.

- **Desarrollo de un juego serio narrativo para el desarrollo del PC mediante puzles (La Mansión Paranormal)**. En el Capítulo 3 se describe en qué consiste La Mansión Paranormal, el primer juego que se desarrolló para llevar a cabo este trabajo. En él, se experimentó con distintas formas para desarrollar el PC, terminando en un juego narrativo con puzles, que en cierta manera es una "escape room". De los puzles que se ofrecen, se puede destacar la incorporación de una progresión de 3 dificultades en todos ellos y un sistema de recompensas para motivar al jugador.

  De este proyecto se puede observar la importancia de un hilo conductor narrativo que ocasione una sensación de progreso al usuario. A esta progresión se le suma la inclusión de una retribución en forma de estrellas si el usuario lo hace de forma óptima. Estas características funcionaron muy bien y es por ello que se decidió mantener en Articoding. Además, se consiguió una dinámica de trabajo y una organización a base de iteraciones cortas que se conservará durante el resto del desarrollo del trabajo.

- **Creación de un juego para el desarrollo del PC y la enseñanza de la programación (Articoding)**. En el Capítulo 4 se habla de forma extensa sobre el desarrollo y los detalles técnicos de la implementación del segundo y principal juego, Articoding. La idea inicial, al ser contrastada con expertos educativos, pasó de una app móvil a un juego de escritorio, pero se mantuvo el objetivo de enseñar conceptos de programación mediante el PC.

  De este punto, se puede destacar la clasificación y secuenciación novedosa frente a otros juegos de los conceptos de programación, de forma que se introduzcan tarjetas de código en un orden adecuado para su correcto aprendizaje. Además de los aspectos ya mencionados de variables y estado del juego, se planteó una discusión sobre qué conceptos presentar antes y tras un largo debate se tomó la decisión de poner la categoría "Condicionales" después de "Bucles". Esto se debe a que los condicionales tienen más utilidad dentro de bucles que solos, además de que muchos de los juegos analizados previamente también siguen este orden. Cabe destacar también que se insistió

en la representación visual del tablero, de forma que se entendiese perfectamente el problema y el objetivo del jugador. Es por ello que se optó por unos gráficos 3D con posibilidad de rotar la cámara. Este aspecto visual ayuda también a tener más clara la idea del estado completo del juego y que su percepción como juego y no sólo como aplicación educativa mejore.

- **Realizar pruebas de usuario con profesores y expertos interesados**. En los Capítulos 3 y 4 se habla sobre el proceso de realización de las pruebas con usuarios. Este objetivo es uno de los más importantes del trabajo, pues es un punto clave para que el juego pueda considerarse apto para su uso en el ámbito escolar. Los resultados de estas pruebas muestran que el juego podría aplicarse como una herramienta útil para el profesor con el propósito de mejorar la motivación y la enseñanza de la programación y el desarrollo de las habilidades del PC.

  De este objetivo puede concluirse que las pruebas han sido una parte importante del desarrollo, ya que se han integrado en los ciclos de trabajo. Esto ha dado lugar a un producto más maduro que ha ido mejorando continuamente. En general, las pruebas se han realizado sin grandes dificultades y la mayoría de los profesores que participaron han afirmado que usarían el juego en sus clases. Además en estas pruebas no sólo se ha tenido en cuenta la opinión de los profesores y expertos si no que se ha contrastado con la información de las analíticas de modo que sabemos realmente lo que han jugado y cómo han jugado lo que proporciona mayor validez a sus opiniones.

No obstante, aunque se han conseguido dos productos completos y se han cumplido todos los objetivos propuestos con éxito, como resultado del amplio análisis de juegos y de las discusiones con expertos, hay distintas ideas que se contemplaron pero que no se han implementado debido a la falta de tiempo o a su complejidad. Estas ideas se presentan en la siguiente sección de trabajo futuro centrándose en el juego Articoding.

Sin embargo, la estructura del proyecto está preparada para integrar estas ideas y otras posibles ampliaciones en el proyecto, ya que cuenta con un código robusto, escalable y que cuenta con un sistema de analíticas que simplifica su evaluación. Las analíticas facilitan la realización continuada de pruebas con usuarios y la mejora del producto con correcciones de errores.

5.2. Trabajo futuro

A continuación se analiza cada una de estas ideas que pueden implementarse en el proyecto con algo más de tiempo y que dan aún más valor al videojuego de Articoding como herramienta educativa. Algunas de estas ideas son más complejas de implementar ya que exigiría disponer de al menos una cierta estructura de servidor de juego.

5.2.1. Tienda con moneda virtual

La tienda se planteó como un elemento clave en el diseño inicial de la progresión del jugador, pudiendo intercambiar una moneda virtual por artículos estéticos (personalización del pingüino, apariencias de elementos del tablero, etc.). Para una futura ampliación de contenido del juego se plantea introducir tanto la moneda virtual como la tienda.

Se pueden obtener unidades de esta moneda virtual al obtener estrellas (5 monedas por cada estrella, más un bonus de 5 monedas por obtener las 3 estrellas del nivel) o al completar misiones. La incorporación de una moneda virtual puede servir de incentivo para seguir jugando y así conseguir todos los cosméticos y artículos que se puedan conseguir en la tienda. Cabe destacar que en ningún momento se ha planteado utilizar un sistema de pago real para conseguir dicha moneda virtual. Es decir, no se puede pagar para conseguir la moneda virtual del juego si no que sólo y exclusivamente se consigue a base de jugar y practicar.

5.2.2. Perfil, gestión de usuarios y experiencia de uso

En el perfil se podrá consultar tus estadísticas de usuario (tiempo jugado, días seguidos, etc.) y tus estadísticas de juego (nivel de experiencia, título). Esto implica un sistema de niveles de experiencia donde el usuario obtiene experiencia al completar cada nivel y finalizar misiones.

Además, se podrán agregar a una lista de amistades todos aquellos usuarios que se quiera, pudiendo ver las estadísticas de los usuarios agregados. Este es otro incentivo para que el usuario quiera seguir jugando, la competitividad. De esta manera, puede comparar el nivel con los amigos y competir para conseguir mejores estadísticas que el resto.

5.2.3. Misiones, insignias y títulos

Para ofrecer objetivos secundarios al jugador, se propone la implementación de un sistema de misiones, insignias y títulos.

Con las misiones se establecen ciertos objetivos que se exponen de manera clara al jugador. Con esto se consigue que el jugador sienta que tiene todavía tiene contenido que completar y se incentiva la rejugabilidad. Además, se pueden incluir objetivos diarios y/o semanales a cambio de una recompensa como un cosmético, una cantidad de monedas virtuales, experiencia, etc.

Por otro lado, las insignias y los títulos son recompensas que pueden equiparse en el perfil y se consiguen cuando cumples grandes objetivos. Una insignia es una medalla con un símbolo distintivo que representa una proeza obtenida por el jugador. Cada insignia va acompañada de un título, que consiste en un texto que da nombre al logro completado por el jugador. Por ejemplo, el jugador recibe el título e insignia de “Atrapado en un bucle infinito” si llega a utilizar 500 veces los bucles en su código. Los títulos y las insignias se podrán mostrar en el perfil del usuario, así todos los jugadores podrán verlos.

5.2.4. Niveles creados por la comunidad

Actualmente en el juego sólo existe la posibilidad de crear niveles y jugarlos de manera local. Esto puede suponer un problema para que un profesor pueda distribuir los los niveles creados a los ordenadores de sus alumnos. Por eso, resulta interesante la incorporación de un sistema para compartir los niveles a nivel comunitario.

Así, los profesores podrán compartir con sus alumnos niveles diseñados por ellos mismos de manera más cómoda y también podrán compartirlo con otros profesores, creando una comunidad de profesores en el juego. Además, se abre un caso de uso nuevo en el que el profesor puede pedir una actividad de creación a sus alumnos y posteriormente estos entregarla utilizando este sistema que se propone.

Puesto que el contenido original del juego está muy acotado, esta funcionalidad da la posibilidad de seguir obteniendo experiencia, completar misiones y obtener títulos jugando los niveles que ha creado la comunidad de jugadores.

5.2.5. Validación automática de niveles creados por los usuarios

En la versión final del videojuego, no se valora la complejidad de los niveles creados por el usuario ni de la forma de solucionar estos tableros. Para asegurar que el estudiante aplique correctamente los conceptos aprendidos, sería necesario un sistema más complejo de validación de niveles.

Este sistema debería ser capaz de evaluar la complejidad del tablero creado por el usuario en función de diferentes factores como por ejemplo la variedad o la

cantidad de elementos usados. Por otro lado, sería necesario un sistema que mediante heurísticas obtuviese la mejor solución a cualquier tablero, de esta forma se podría analizar si la solución propuesta por el usuario es la solución óptima o si existen fallos en la resolución. Por último, la implementación de un sistema de evaluación de código, (similar al que propone Dr. Scratch) permitiría conocer el estado en el proceso de aprendizaje en el que se encuentra el jugador y en función de esto ajustar su experiencia.

5.2.6. Pistas sobre el código

Actualmente, las pistas que se ofrecen son ayudas visuales sobre el tablero, que indican los movimientos que hay que realizar. Sin embargo, estas pistas no ofrecen ninguna ayuda sobre las tarjetas de código que hay que utilizar para resolver el nivel. Como trabajo futuro, se plantea un sistema de pistas que, cuando sea necesario, ofrezca pistas que indiquen al jugador algunas de las tarjetas de código necesarias para superar el nivel. Este sistema podría resaltar estas tarjetas de manera que el usuario tenga que colocarlas manualmente. O, por otro lado, se podrían colocar automáticamente las tarjetas en el área de código (respetando las tarjetas que ya hubiese colocadas).

5.2.7. Categoría: funciones

Siguiendo el acercamiento gradual a la enseñanza de conceptos de programación, el siguiente concepto a introducir en la aplicación, es el de funciones. La librería *ublockly* ofrece una implementación de este tipo de bloques, pero actualmente se encuentra muy limitada.

Sin embargo, las funciones pueden suponer un gran avance en la adquisición de habilidades propias del PC, ya que tienen como base conceptos como la abstracción o el reconocimiento de patrones. Además, el modelo de problema adoptado por la aplicación (basado en movimientos en el tablero) permitirá introducir fácilmente la enseñanza de estos conceptos.

5.2.8. Evaluación con usuarios

Haciendo un análisis detallado de los datos de interacción recogidos de las pruebas con usuarios en entornos reales podemos encontrar cuales son los problemas que tiene un jugador en la interacción con el juego, los cuales pueden ser reflejo de errores de concepto sobre los elementos de programación o de un mal diseño de juego. Estas pruebas con usuarios y los análisis permitirán la mejora del videojuego, tanto como elemento de entretenimiento como herramienta educativa.

Además, en un futuro, el análisis de estas trazas en tiempo real para determinar si el jugador tiene problemas a la hora de resolver un nivel o si le resulta demasiado fácil puede permitir dos cosas: (1) la evaluación del conocimiento del usuario por parte del profesor y conocer los problemas generales de la clase. (2) crear un juego adaptativo en función de las necesidades y las carencias del jugador.

Chapter 5

Conclusions, contributions and future work

"That man will fight us every day and every hour till the end of the war." - James Longstreet

We consider that the project has achieved its objectives in a very satisfactory way. Two serious games have been created to promote both computational thinking and programming: La Mansión Paranormal and Articoding. In addition, these games have been formatively evaluated with experts and teachers and even Articoding has been tested with students in a real environment such as a school.

It is a project that has required a great effort since it has had an important research component in aspects such as computational thinking, the use of serious games for their promotion in school environments or learning analytics applied to games (and even standards such as xAPI). It has been necessary to read many research articles as well as to search and review several existing games in this field.

The development has been complex and has required not only effort but also learning new concepts and tools. Two tools have been used, on the one hand uAdventure but combined with puzzles and minigames developed in Unity for the development of La Mansión Paranormal and on the other hand the inclusion of an ublockly interpreter, available as open source, which has been key for the development of Articoding. To facilitate the use of these two tools, it has been necessary to understand how they work and become familiar with their structure. This has allowed us to detect different weaknesses or bugs in these projects and to improve both the game and the tools themselves.

As far as possible, we have tried to evaluate the game with experts, with teachers and even with students to prove that they are applicable in a real classroom environment. This is not a very common aspect in an academic project such as a final degree project, due to the inherent complexity of doing the whole experimental design, but above all to the difficulty of finding the subjects to carry out the experimentation and the shortage of time from the time the final product is available until it has to be delivered. However, in this case, with the support of our directors, we have managed to do it and we are very satisfied with the result. This makes the

resulting games, which are available as open source on github, useful tools for teachers to motivate their students in learning computational thinking and programming.

It should also be noted that we have had to make a special effort to contrast ideas with educational experts and that this has led us to have to develop products that we had not anticipated. For example, although it was clear to us that we had to make games that could be played without having to read a manual beforehand, we had not considered that we would have to make teaching guides for teachers on how to use the games in the classroom. If we wanted our games to be used by a wide range of teachers, we could not depend on their knowledge of the concepts of educational games and for that we had to make detailed guides more adapted to their concepts and vocabulary.

Finally, as part of the validation of the Articoding game, summative tests have been proposed with students of the Colegio La Inmaculada Escolapias Puerta de Hierro. These tests will be carried out on June 15, 16 and 17, 2021 with 6 classes of students of 1st and 2nd of Secondary School. Due to the proximity with the delivery date this aspect is mentioned here but it will not be included completely in the report.

5.1. Contributions

This section analyzes in more detail some of the contributions and how this work has contributed to the fulfillment of the objectives proposed at the beginning of this work:

- **Analyze the current situation of CT**. Chapter 2 provides an in-depth analysis of the current situation of the CT, with special emphasis on education. It has been shown that the CT is becoming more and more important due to the generalization of technologies and their use in more and more facets of daily life. And of course, at school the CT is used in the different STEM disciplines, but above all, in programming.

  That is why it was decided to conduct an extensive study and analysis of various games, including some that use the CT to teach programming concepts. From this study, we can highlight the use of code blocks (visual programming) and the movement by boxes for problem posing in most of the games analyzed. This helped a lot to formalize the idea of how the Articoding levels were going to be, but another approach was tried to stand out among the other games. For example, it is quite novel to present the variables early and try to give the players a clearer idea of the game state provided not only by the moving element but also by the state of other objects (e.g., mirrors).

This is why the final idea is to combine the movement of objects by squares with the use of mirrors to reflect lasers or squares with different possible states.

- **Development of a serious narrative game for the development of the CT through puzzles (La Mansión Paranormal)**. Chapter 3 describes what La Mansión Paranormal, the first game that was developed to carry out this work, consists of. In it, we experimented with different ways to develop the CT, ending up in a narrative game with puzzles, which in a way is an "escape room". Of the puzzles offered, we can highlight the incorporation of a progression of 3 difficulties in all of them and a reward system to motivate the player.

  From this project we can see the importance of a narrative thread that causes a sense of progress to the user. Added to this progression is the inclusion of a reward in the form of stars if the user performs optimally. These features worked very well and that is why it was decided to keep them in Articoding. In addition, a work dynamic and an organization based on short iterations was achieved that will be maintained for the rest of the development of the work.

- **Creation of a game for CT development and teaching programming (Articoding)**. Chapter 4 discusses extensively the development and technical details of the implementation of the second and main game, Articoding. The initial idea, when contrasted with educational experts, changed from a mobile app to a desktop game, but the goal of teaching programming concepts using the CT was maintained.

  From this point, we can highlight the new classification and sequencing of programming concepts, so that code cards are introduced in an appropriate order for proper learning. In addition to the aforementioned aspects of variables and game state, a discussion was raised about which concepts to present before and after a long debate the decision was made to put the category "Conditionals" after "Loops". This is because conditionals have more utility within loops than alone, in addition to the fact that many of the games previously analyzed also follow this order. It should also be noted that we insisted on the visual representation of the board, so that the problem and the player's objective would be perfectly understood. That is why we opted for 3D graphics with the possibility of rotating the camera. This visual aspect also helps to have a clearer idea of the complete state of the game and to improve its perception as a game and not only as an educational application.

- **Conduct user tests with teachers and interested experts**. Chapters 3 and 4 discuss the process of conducting user testing. This objective is one of the most important of the work, as it is a key point for the game to be considered suitable for use in the school environment. The results of these tests show that the game could be applied as a useful tool for the teacher with the purpose of improving the motivation and teaching of programming and the development of CT skills.

  From this objective it can be concluded that testing has been an important part of the development as it has been integrated into the work cycles. This has resulted in a more mature product that has been continuously improved. In general, the tests have been carried out without major difficulties and most of the teachers who participated have stated that they would use the game in their classes. In addition, these tests have not only taken into account the opinion of teachers and experts but have been contrasted with the information from the analytics so that we really know what they have played and how they have played, which provides greater validity to their opinions.

However, although two complete products have been achieved and all the proposed objectives have been successfully met, as a result of the extensive game analysis and discussions with experts, there are different ideas that were contemplated but have not been implemented due to lack of time or complexity. These ideas are presented in the following section of future work focusing on the Articoding game.

However, the project structure is ready to integrate these ideas and other possible extensions into the project, as it has a robust, scalable code that has an analytics system that simplifies its evaluation. The analytics facilitate ongoing user testing and product enhancement with bug fixes.

5.2. Future work

The following is a discussion of each of these ideas that can be implemented in the project with a little more time and that give even more value to the Articoding video game as an educational tool. Some of these ideas are more complex to implement as they would require at least some game server structure.

5.2.1. Shop with virtual currency

The store was proposed as a key element in the initial design of the player's progression, being able to exchange a virtual currency for aesthetic items (penguin

customization, board element appearances, etc.). For a future expansion of game content, we plan to introduce both the virtual currency and the store.

Units of this virtual currency can be obtained by obtaining stars (5 coins for each star, plus a bonus of 5 coins for obtaining the 3 stars of the level) or by completing missions. The addition of a virtual currency can serve as an incentive to continue playing and thus get all the cosmetics and items that can be obtained in the store. It should be noted that at no time has it been considered to use a real payment system to get the virtual currency. That is, you can not pay to get the virtual currency of the game but only and exclusively get it by playing and practicing.

5.2.2. Profile, user management and experience

In the profile it will be possible to consult your user statistics (time played, days in a row, etc.) and your game statistics (experience level, title). This implies a system of experience levels where the user gets experience by completing each level and finishing missions.

In addition, you will be able to add to a list of friends all those users that you want, being able to see the statistics of the added users. This is another incentive for the user to keep playing, competitiveness. In this way, you can compare your level with your friends and compete to get better statistics than the rest.

5.2.3. Missions, badges and titles

In order to offer secondary objectives to the player, it is proposed to implement a system of missions, badges and titles.

With the missions, certain objectives are established and clearly exposed to the player. This makes the player feel that they still have content to complete and encourages replayability. In addition, daily and/or weekly objectives can be included in exchange for a reward such as a cosmetic, a quantity of virtual coins, experience, etc.

On the other hand, badges and titles are rewards that can be equipped in the profile and are obtained when you achieve great goals. A badge is a medal with a distinctive symbol that represents a feat achieved by the player. Each badge is accompanied by a title, which consists of a text naming the achievement completed by the player. For example, the player receives the title and badge "Trapped in an infinite loop" if he reaches 500 times using the loops in his code. The titles and badges can be displayed in the user's profile, so that all players can see them.

5.2.4. Community-created levels

Currently in the game there is only the possibility to create levels and play them locally. This can be a problem for a teacher to distribute the created levels to his students' computers. Therefore, it is interesting to incorporate a system for sharing levels at the community level.

Thus, teachers will be able to share with their students levels designed by themselves in a more comfortable way and they will also be able to share them with other teachers, creating a community of teachers in the game. In addition, a new use case is opened in which the teacher can ask his students to create an activity and then the students can hand it in using this proposed system.

Since the original content of the game is very limited, this functionality gives the possibility to continue gaining experience, completing quests and obtaining titles by playing the levels created by the community of players.

5.2.5. Automatic validation of user-created levels

In the final version of the video game, the complexity of the levels created by the user and the way to solve these boards is not assessed. To ensure that the student correctly applies the concepts learned, a more complex system of level validation would be necessary.

This system should be able to evaluate the complexity of the user-created board based on different factors such as the variety or quantity of elements used. On the other hand, it would be necessary a system that by means of heuristics would obtain the best solution to any board, in this way it would be possible to analyze if the solution proposed by the user is the optimal solution or if there are failures in the resolution. Finally, the implementation of a code evaluation system (similar to the one proposed by Dr. Scratch) would allow to know the state of the learning process in which the player is and adjust his experience accordingly.

5.2.6. Code hints

Currently, the hints offered are visual aids on the board, indicating the moves to be made. However, these hints do not offer any help on the code cards to be used to solve the level. As future work, a hint system is proposed that, when necessary, offers hints that indicate to the player some of the code cards needed to pass the level. This system could highlight these cards so that the user has to place them manually. Or, on the other hand, the cards could be placed automatically in the code area (respecting the cards already placed).

5.2.7. Category: functions

Following the gradual approach to teaching programming concepts, the next concept to be introduced in the application is that of functions. The *ublockly* library offers an implementation of this type of blocks, but it is currently very limited.

However, functions can be a breakthrough in the acquisition of CT skills, since they are based on concepts such as abstraction or pattern recognition. In addition, the problem model adopted by the application (based on movements on the board) will make it easy to introduce the teaching of these concepts.

5.2.8. User testing

By making a detailed analysis of the interaction data collected from user testing in real environments we can find out which are the problems that a player has in the interaction with the game, which may be a reflection of misconceptions about the programming elements or a bad game design. These user tests and analyses will allow the improvement of the videogame, both as an entertainment element and as an educational tool.

In addition, in the future, analysis of these real-time traces to determine if the player is having trouble solving a level or finding it too easy may allow for two things: (1) assessment of the user's knowledge by the teacher and learn about general problems in the class. (2) creating an adaptive game based on the player's needs and shortcomings.

Capítulo 6

Aportación de los participantes

"No elegimos como empezamos en esta vida. La verdadera grandeza es que hacemos con lo que nos toca." - Victor Sullivan

6.1. Trabajo colaborativo

Este proyecto se ha llevado a cabo por un equipo de trabajo formado por tres personas y dos directores de proyecto.

Para gestionar las tareas y llevar un plan de desarrollo sólido y continuado, se ha creado un proyecto de Pivotal Tracker como herramienta ágil de producción. En dicha herramienta se han ido asignando y completando tareas cada semana, poniendo deadlines para los distintos hitos del desarrollo.

Cabe destacar que durante todo el desarrollo del trabajo se ha mantenido una comunicación constante de todos los integrantes mediante Discord, una aplicación de chat de voz. Esto ha permitido el trabajo colaborativo en tareas complejas y la toma de decisiones respecto a problemas que se han presentado durante el transcurso del proyecto.

Este último capítulo expone las contribuciones más importantes que ha realizado cada uno de los componentes del equipo de trabajo de este proyecto. Las aportaciones de cada uno de los tres miembros del equipo de trabajo se lista a continuación.

6.2. Aportaciones de Dany Faouaz Santillana

Aportaciones de La Mansión Paranormal:

- Puzle Hacking Ético.
- Sacadas fotos de los fondos de las salas.
- Puzle Laberinto de Números.
- Mejora visual del Puzle Anillas.
- Cutscene inicial extendida.
- Trazas de dificultades de todos los puzles.

Aportaciones de Articoding:

- Creación de repositorio de GitHub.
- Creación de proyecto de Unity.
- Sistema de Tabs y Paneles para el diseño móvil.
- Terminado Swipe entre paneles.
- Radial ProgressBar (barra de progreso circular que indica el progreso de la categoría).
- Ajuste de escala y arreglos de la interfaz del diseño móvil.
- DraggableUI (componente para poder arrastrar un elemento de UI).
- Sistema de Tweens (animaciones por código).
- Paneles de Tutoriales y Enunciado (descartado) en la escena del nivel.
- Inicialización de Enunciado (StatementManager).
- Carga de estados iniciales.
- Reset del nivel.
- Mejora del StatementManager: se pueden añadir títulos, párrafos, imágenes, etc.
- Ciclo de juego cerrado y conectado con la escena del Menú Principal.
- Las tarjetas ya no se pueden mover fuera de la zona de código y se esconden.
- Aplicada fuente de texto en toda la UI.
- BoardManager.
- BoardObject.
- LaserManager y LaserRay.
- ILaserReflector y BasicMirror.
- LaserEmitter.
- LaserReceiver.
- Rotación de objetos del tablero.
- Arreglo de movimiento de objetos.
- Movimientos y rotaciones interpoladas en el tiempo.
- PopupPanel.

- Shader para oscurecer la pantalla e iluminar una zona con un Popup.
- Popup Trigger: trigger que lanza un popup cuando se pasa el ratón por encima de un elemento de la UI o un objeto del tablero.
- PopupManager.
- Sistema de tutoriales.
- TutorialTrigger.
- Guardado y carga de tutoriales ya mostrados al usuario.
- Creados tutoriales de UI.
- Creados tutoriales de Menú Principal.
- Corregido error del rebote del láser con el espejo.
- Cambiada orientación del tablero al plano.
- Tipos de celdas.
- Integración de analíticas de Unity (descartado).
- Trazas de: Nivel empezado, nivel finalizado y código ejecutado
- Escena de la sala del pingüino empezada.
- Trazas de: tablero.
- Integrado tracker de uAdventure.
- Mejora visual de ventanas de UI.
- Arreglo de error de la barra de progreso de la categoría.
- Mejora visual del panel de tutoriales.
- Mejora visual del creador de niveles.
- Arreglo de los iconos de las categorías.
- Configurador de resolución de pantalla en el menú de opciones.
- Añadido paquete de localización al proyecto.
- Selector de idiomas en el menú de opciones.
- Localización de la descripción de las categorías.
- Localización de LevelData.
- Hecho que se seleccione el idioma dependiendo del idioma del equipo.
- Diseño de niveles de la categoría 4: Bucles.
- Pantalla de carga.
- Implementación de Simva y conexión con el Tracker.

Aportaciones de la memoria:

- Resumen. Palabras clave.
- Abstract. Keywords.
- Capítulo 1. ¿Qué es el PC?
- Capítulo 2. PC en STEM. PC en otras asignaturas. Un-plugged. Plugged. Lenguajes de programación usados. Otros enfoques. Análisis de juegos relacionados con el PC o la programación.

- Capítulo 4. ¿Qué es Articoding? Gameplay/jugabilidad. Mecánicas y dinámicas. Sistema de pop-ups. Sistema de tutoriales. Sistema de temario. Entorno de desarrollo (Unity). Pivotal Tracker. Git. Repositorios externos. Canales de comunicación. Limitaciones.
- Capítulo 5. Futuro trabajo.
- Chapter 1.
- Chapter 5.

Otras aportaciones:

- Análisis de juegos: May's Journey, ProgrammingHero, CodeCombat, BeBlockly.
- SpriteBox: Code Hour y CodeSpark Academy.
- GDD Articoding.
- Elaboración de guía docente de La Mansión Paranormal.
- Elaboración de guía docente de Articoding.
- Presentación de La Mansión Paranormal.

6.3. Aportaciones de Álvaro Poyatos Morate

Aportaciones de La Mansión Paranormal:

- Puzle Electricista.
- Diálogos de Jim Silver, Jack Cooper, el mayordomo y hombre de hojalata.
- Diálogo final de los tres hermanos.
- Objetos pistas para acceder a algunos puzles.
- Dificultades Puzle Hacking Ético.
- Dificultades Puzle Laberinto de Números.
- Dificultad mejorada del puzle Formas y Colores.
- Dificultades Electricista.
- Aleatoriedad de estados iniciales de algunos puzles.
- Pistas de todos los puzles.
- Trazas de acciones dentro de los puzles.
- Cambio de apariencia de todos los personajes.
- Trazas de estado inicial de puzles.
- Cambio al formato de las trazas.

Aportaciones de Articoding:

- Arreglado y probado el proyecto que viene en la librería ublockly en la versión de Unity 2020.1.6f.
- Nueva categoría y nuevas tarjetas propias para instanciar objetos y moverlos.
- Integración de ublockly en el proyecto de Articoding.
- Actualizada versión de ublockly donde se limitan los bucles y las recursiones infinitas.
- Arreglado error de tarjetas invisibles.
- Traducción de las tarjetas de código de ublockly al español mediante un json.
- Ahora el nivel tiene un textAsset con las categorías y tarjetas disponibles para ese nivel.
- Incluido botón AllActive que activa todas las tarjetas (solo en el editor de Unity).
- Mejorada estructura del json con las tarjetas activas: incluida la opción de desactivar las tarjetas que se indican mediante el booleano “activate”.
- Añadida la opción de indicar las tarjetas máximas del inventario en el json.
- Cuando se usa el máximo de tarjetas de un tipo, se desactiva esa tarjeta y se le cambia el color a gris.
- Corregido error al borrar a la vez varias tarjetas conectadas entre sí.
- Guardado y carga del tablero.
- Escena de BoardCreator.

- Se pueden mover, eliminar y crear objetos en el BoardCreator.
- Interfaz del BoardCreator: ventana para poder crear el tablero con unas dimensiones específicas e introducir el nombre para guardarlo.
- Creados niveles de la categoría 1: Variables.
- Poder cambiar la intensidad del láser.
- Añadida condición de victoria (se completa el nivel si todos los receptores reciben un láser durante 2 segundos).
- Ajuste del tablero al espacio de la pantalla (FitBoard()).
- Tarjetas nuevas: mover láser, girar láser y cambiar intensidad del láser.
- Se le puede pasar la cantidad de movimiento a la tarjeta mover láser.
- Contador de tarjetas que quedan en el inventario.
- Límites del tablero del nivel: se han añadido casillas vacías alrededor de los niveles.
- Diseño de niveles de las categorías 2 y 3.
- Creados niveles de la categoría 2: Tipos de datos.
- Creados niveles de la categoría 3: Operadores básicos.
- Tutoriales categoría 1.
- Sistema de progreso y guardado.
- Funcionalidad de Drag&Drop del creador de niveles.
- Ventana con parámetros del objeto seleccionado en el creador.
- Se puede cambiar el tipo de la casilla en el creador.
- Creados niveles de la categoría 4: Bucles.
- Creados niveles de la categoría 5: Condicionales.
- Tarjeta de si están todas las celdas de un tipo ocupadas.
- Animación de los elementos colisionando con obstáculos (cuando intentas mover un elemento donde hay un obstáculo).
- Ahora el reset del nivel ya no destruye y crea todos los elementos otra vez.
- Flechas de movimiento en la zona de código para indicar que se puede mover la zona de código.
- Bloqueado que se pueda abrir el inventario cuando hay un tutorial activo.
- Arreglo de finalización del nivel: cuando cae un láser se acaba el nivel.
- Cartel de bucles infinitos.
- Flechas de movimiento y de giro de las pistas.
- La ventana del objeto seleccionado en el creador ahora muestra el nombre completo del objeto.
- Se guardan los niveles creados por el usuario.
- Arreglos creador de niveles.
- Nombre de nivel en la escena de nivel.
- Validación del nivel en el creador de niveles.
- Niveles de introducción de cada categoría.

- Arreglo del TextField del worskpace de ublockly.
- Sincronización de la localización de ublockly con el sistema de localización.
- Localización de assets de initial states.
- Localización de textos fijos de ublockly.
- Trazas del creador de niveles.

Aportaciones de la memoria:

- Capítulo 2. Relación entre PC y programación. PC sin programación.
- Capítulo 3. Pruebas con usuarios y resultados.
- Capítulo 4. Desarrollo de la idea y su evolución. Sistema de Guardado. Sistema de creador de niveles.
- Capítulo 5. Futuro Trabajo: Evaluación automática de los niveles creados. Categoría: funciones. Pistas sobre el código.
- Chapter 5.

Otras aportaciones:

- Análisis de juegos: GrassHopper. LightBot. Scratch Jr. Dragon Architect. Blockly Games.
- SpriteBox: Code Hour y CodeSpark Academy.
- GDD Articoding.
- Scrum Master.
- Elaboración de guía docente de La Mansión Paranormal.
- Elaboración de guía docente de Articoding.
- Presentación de La Mansión Paranormal.

6.4. Aportaciones de Arturo García Cárdenas

Aportaciones de La Mansión Paranormal:

- Creadas salas del piso inferior.
- Puzle Anillas.
- Creadas salas de piso de arriba.
- Puzle Formas y Colores.
- Sala de fuera y llave.
- Diálogo John Gold y Lady Ada Mantine.
- Sala de despacho y libro.
- Mejora visual de puzles.
- Dificultades del puzle Anillas.
- Upgrade de uAdventure en el proyecto.
- Trazas de progresión de puzles.
- Simva configurado.
- Sistema de estrellas.
- Escena de créditos.
- Cutscene inicial y final.

Aportaciones de Articoding:

- Concept art del diseño móvil.
- Logo del juego.
- Panel de Perfil en el diseño móvil.
- Scroll del panel de Perfil en el diseño móvil.
- Comenzado Swipe entre paneles.
- Panel de Play en el diseño móvil.
- Planteados todos los paneles
- Mejora visual de UI: aplicada paleta de colores.
- Mejora del TabManager: desplazamiento entre tabs cíclico.
- Footer con controles.
- Menú de opciones y botón para abrirlo.
- Menú de confirmación de salida del juego.
- Prefab Category.
- Panel de selección de categoría.
- Prefab LevelCard.
- Panel de selección de nivel.
- Texto de CurrentTab.
- Cursor (descartado en el futuro).
- Modelado e implementado Receptor.

- Modelado e implementado Espejo.
- Modelado e implementado Emisor de láser.
- Mejora visual de los láseres.
- Partículas de láser y del láser moviéndose.
- Modelado e implementado Obstáculo.
- Modelada e implementada Puerta.
- Mejorada iluminación de la zona de juego.
- Órbita de la cámara alrededor del tablero del nivel.
- Botón para centrar el nivel.
- Posibilidad de hacer zoom del nivel con la rueda del ratón.
- Ahora el botón de centrado también resetea el zoom.
- Limitado para que solo se pueda hacer zoom si el ratón se encuentra dentro de la vista de juego.
- Sistema de recompensas.
- StarController.
- Panel de Nivel completado.
- Panel de Has fallado.
- Posibilidad de minimizar los paneles de Nivel completado y Has fallado para poder ver el código y el estado del tablero.
- Modelado del pingüino.
- Rigging del pingüino.
- Sala del pingüino: añadidas texturas, modelos importados, etc.
- Se cuentan los pasos realizados en el nivel: movimiento, giro, activar/desactivar puerta, etc.
- Incluidos los pasos mínimos de cada nivel.
- Arreglado error de estrella de pasos mínimos.
- Animaciones del pingüino: idle y walk.
- Importado modelo de puerta de la sala del pingüino.
- Skybox de la zona de juego.
- Iconos de UI.
- Level ahora contiene una imagen en vez de una descripción.
- Ahora se muestra una imagen del nivel cuando lo seleccionas en el menú de selección de niveles.
- Hechas todas las capturas de los niveles e implementadas.
- Arreglo de zoom del nivel.
- Añadidas capturas de los niveles en el panel de tutoriales.
- Localización de los textos fijos.
- Animación de caída de los láseres y espejos.
- Mejora de animaciones de movimiento erróneo de los objetos.
- Órbita de la cámara en el creador.

- Diseño de niveles de las categorías 1: Variables y 5: Condicionales.
- Localización de los textos dinámicos de los tutoriales.
- Traducción de todos los textos de los tutoriales (título y descripción).
- Tomadas todas las imágenes de las tarjetas de código para la localización de texturas.

Aportaciones de la memoria:

- Resumen.
- Capítulo 3. Contexto y Narrativa. Puzles. Retroalimentación de la evaluación y versión final.
- Capítulo 4. Estilo visual. Narrativa. Sistema de Progresión y Recompensas. Sistema de Pistas.
- Capítulo 5. Contribuciones. Futuro trabajo.
- Capítulo 6. Trabajo colaborativo. Aportaciones de Dany Faouaz Santillana. Aportaciones de Álvaro Poyatos Morate. Aportaciones de Arturo García Cárdenas.
- Chapter 1.

Otras aportaciones:

- Análisis de juegos: CodeHero, KIBO y Duolingo.
- SpriteBox: Code Hour y CodeSpark Academy.
- Elaboración de guía docente de La Mansión Paranormal.
- Elaboración de guía docente de Articoding.
- Vídeo tráiler de La Mansión Paranormal.
- Presentación de La Mansión Paranormal.

Bibliografía

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Educational Technology and Society*, *19*(3), 47–57.

Araujo, A. L. S. O., Andrade, W. L., Guerrero, D. D. S., & Melo, M. R. A. (2019). How Many Abilities Can We Measure in Computational Thinking? *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 545–551. https://doi.org/10.1145/3287324.3287405

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12. *ACM Inroads*, *2*(1), 48–54. https://doi.org/10.1145/1929887.1929905

Bauer, A., Butler, E., & Popovic, Z. (2017). Dragon architect: Open design problems for guided learning in a creative computational thinking sandbox game. *ACM International Conference Proceeding Series*, *Part F1301*. https://doi.org/10.1145/3102071.3102106

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017). Development of Computational Thinking Skills through Unplugged Activities in Primary School. *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, 65–72. https://doi.org/10.1145/3137065.3137069

Calvo-Morata, A. (2020). *Uso de Técnicas de Learning Analytics para la Validación, Mejora y Aplicación de Juegos Serios en la Clase Aplicado al Ciberbullying.*

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). Designing ScratchJr. *Proceedings of the 12th International Conference on Interaction Design and Children - IDC '13*, 1–10. https://doi.org/10.1145/2485760.2485785

Freire, M., Serrano-Laguna, Á., Iglesias, B. M., Martínez-Ortiz, I., Moreno-Ger, P., & Fernández-Manjón, B. (2016). Game Learning Analytics: Learning Analytics for Serious Games. In *Learning, Design, and Technology* (pp. 1–29). Springer International Publishing. https://doi.org/10.1007/978-3-319-17727-4_21-1

Gouws, L. A., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities. *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '13*, 10.

https://doi.org/10.1145/2462476.2466518

Hidalgo, E. S. (2019). Adapting the scrum framework for agile project management in science: case study of a distributed research initiative. *Heliyon*, *5*(3), e01447. https://doi.org/10.1016/j.heliyon.2019.e01447

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, *9*(0), 522–531. https://doi.org/10.1016/j.procs.2012.04.056

Kong, S.-C. (2016). A framework of curriculum design for computational thinking development in K-12 education. *Journal of Computers in Education*, *3*(4), 377–394. https://doi.org/10.1007/s40692-016-0076-z

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Marcal, A. S. C., Soares, F. S. F., de Freitas, B. C. C., & Belchior, A. D. (2007). Mapping CMMI Project Management Process Areas to SCRUM Practices. *31st IEEE Software Engineering Workshop (SEW 2007)*, 13–22. https://doi.org/10.1109/SEW.2007.102

Román-gonzález, M., Moreno-león, J., & Robles, G. (2019). Computational Thinking Education. In S.-C. Abelson & H. Kong (Eds.), *Computational Thinking Education*. Springer Singapore. https://doi.org/10.1007/978-981-13-6528-7

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Selby, C. (2013). Computational Thinking : The Developing Definition. *ITiCSE Conference 2013*, 5–8.

Serrano-Laguna, Á., Martínez-Ortiz, I., Haag, J., Regan, D., Johnson, A., & Fernández-Manjón, B. (2017). Applying standards to systematize learning analytics in serious games. *Computer Standards and Interfaces*, *50*, 116–123. https://doi.org/10.1016/j.csi.2016.09.014

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Smeddinck, J. D./Streicher, A. (2016). (2016). A Brief History of Serious Games Phil. *Entertainment Computing and Serious Games*, *9970*(October), 17–41.

http://link.springer.com/10.1007/978-3-319-46152-6

Snalune, P. (2015). The Benefits of Computational Thinking. ITNOW, 57(4), 58–59. https://doi.org/10.1093/itnow/bwv111

Takeuchi, L. M., & Vaala, S. (2014). LevelUp learning: A national survey on teaching with digital games (Joan Ganz Cooney Study). *Games & Learning*, 66 p. http://www.joanganzcooneycenter.org/publication/level-up-learning-a-national-survey-on-teaching-with-digital-games/

Vahldick, A., Farah, P. R., Marcelino, M. J., & Mendes, A. J. (2020). A blocks-based serious game to support introductory computer programming in undergraduate education. *Computers in Human Behavior Reports*, *2*(September), 100037. https://doi.org/10.1016/j.chbr.2020.100037

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. M. (2010). Computational Thinking: What and Why? *Thelink - The Magaizne of the Varnegie Mellon University School of Computer Science*, *March 2006*, 1–6. http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms. *TechTrends*, *60*(6), 565–568. https://doi.org/10.1007/s11528-016-0087-7

Yaroslavaski, D. (2014). *Relating Lightbot to Programming*. 5. http://light-bot.com/Lightbot_HowDoesLightbotTeachProgramming.pdf

Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills? *Computers & Education*, *141*(103633), 13. https://doi.org/10.1016/j.compedu.2019.103633

Webgrafía

Pons Alfonso, J. V. (2007). *¿Qué son los "Serious Games" (juegos serios)?* ¿Qué Son Los "Serious Games" (Juegos Serios)? http://www.exelweiss.com/blog/356/serious-games-juegos-serios/

BeBlockly. (n.d.). https://beblocky.com/

Bebras. (n.d.). https://www.bebras.org/about.html

Bebras Contest. (n.d.). https://bebras.ehu.eus/

Blockly. (n.d.). https://developers.google.com/blockly

Blockly Games. (n.d.). https://blockly.games

Baker, S. (2017). *Blockly VS Scratch: What's best for me?* RobotLab Blog. https://www.robotlab.com/blog/blockly-vs-scratch-whats-best-for-me

Center for game science. (n.d.). http://centerforgamescience.org

Code Hero Game. (n.d.). https://codeherogame.wordpress.com/

Code with google. (n.d.). https://edu.google.com/code-with-google/

CODE.org. (n.d.). https://code.org/

CodeCombat. (n.d.). https://codecombat.com/

CodeSpark Academy. (n.d.). https://codespark.com/

Codinism Inc. (n.d.). http://www.codinism.com/

Discord. (n.d.). https://discord.com/

Dragon Architect. (n.d.). http://dragonarchitect.net

Duolingo. (n.d.). https://www.duolingo.com/courses/all

Games For Change. (n.d.). http://www.gamesforchange.org/

Google Meet. (n.d.). https://meet.google.com/

Grasshopper. (n.d.). https://grasshopper.app/es_419/

Kinder Lab Robotics Store. (n.d.). https://www.shop.kinderlabrobotics.com/main.sc

LightBot. (n.d.). https://lightbot.com/

May's Journey. (n.d.). https://maysjourney.com/

Programamos. (n.d.). https://programamos.es/el-proyecto/

Programming Hero. (n.d.). https://www.programming-hero.com/

Project Bloks. (n.d.). https://projectbloks.withgoogle.com/

PuposeGames. (n.d.). https://www.purposegames.com/es/

Rossi, B. (2015). *Python overtakes French as the most popular language taught in primary schools*. InformationAge. https://www.information-age.com/python-overtakes-french-most-popular-language-taught-primary-schools-123460073/

RAGE. (n.d.). http://rageproject.eu/

Scratch Jr. (n.d.). https://www.scratchjr.org/

Sprite Box: Code Hour. (n.d.). https://spritebox.com/hour.html

uAdventure - eUCM. (n.d.). https://www.e-ucm.es/uadventure/

Vista Cenital - GamerDic. (n.d.). GamerDic. https://www.gamerdic.es/termino/vista-cenital/

Anexo A

Taxonomía de los Juegos Serios

A continuación se adjunta una tabla donde se clasifican los usos de los juegos serios según el tipo de juego y el campo de conocimiento. Esta tabla ha sido extraída de la presentación:

https://thedigitalentertainmentalliance.files.wordpress.com/2011/08/serious-games-taxonomy.pdf

Haciendo un análisis rápido, se puede decir que tanto La Mansión Paranormal como Articoding se encuentran en la casilla "Games for education-Education".

	Games for Health	Adver games	Games for Training	Games for Education	Games for Science and Research	Production	Games as Work
Government & NGO	Public Health Education & Mass Casualty Response	Political Games	Employee Training	Inform Public	Data Collection / Planning	Strategic & Policy Planning	Public Diplomacy, Opinion Research
Defense	Rehabilitation & Wellness	Recruitment & Propaganda	Soldier /Support Training	School House Education	Wargames / planning	War planning & weapons research	Command & Control
Healthcare	Cybertherapy / Exergaming	Public Health Policy & Social Awareness Campaigns	Training Games for Health Professionals	Games for Education and Disease Management	Visualization & Epidemiology	Biotech manufacturing & design	Public Health Response Planning & Logistics
Marketing & Communi-cations	Advertising Treatment	Advertising, marketing with games, product placement	Communication & Sales	Product Information	Opinion Research	Machinima	Opinion Research
Education	Inform about diseases/risks	Social Issue Games	Train teachers / Train workforce skills	Learning	Computer Science & Recruitment	P2P Learning Constructivism Documentary	Teaching Distance Learning
Corporate	Employee Health Information & Wellness	Customer Education & A eness	Employee Training	Continuing Education & Certification	Advertising / visualization	Business Simulation	Command & Control
Industry	Work safety	Sales & Recruitment	Employee Training	Workforce Education	Process Optimization Simulation	Nano/Bio-tech Design	Command & Control

Anexo B

Tabla de PC en asignaturas

En este anexo se encuentra una tabla donde se hablan sobre los distintos beneficios y aplicaciones que tiene el pensamiento computacional en las distintas materias en la educación.

Esta tabla ha sido extraída y traducida de: https://www.thetech.org/sites/default/files/techtip_computationalthinking_v3.pdf

Esta tabla ha sido utilizada como apéndice en la guía docente de La Mansión Paranormal.

	Descomposición	**Identificación de Patrones**	**Abstracción**	**Algoritmos**
Matemáticas	Analizar una forma y descomponerla en formas geométricas simples->Tangram El concepto de la suma gaussiana es más fácil de comprender si se divide en problemas más pequeños	En geometría, la simetría se basa en el reconocimiento de patrones. La resolución de sistemas de ecuaciones complejos mediante métodos más simples anteriormente aprendidos (Resolver un sistema de ecuaciones ternario mediante estrategias de resolución de un sistema de ecuaciones binario)	El análisis de datos consiste en observar grandes cantidades de datos y ser capaz de obtener la información importante	Aprender el concepto de operaciones simples(sumas , restas, multiplicaciones, divisiones) como una sucesión de instrucciones Entender el orden de los operadores(cuadrados, paréntesis, sumas…) y ser capaz de aplicarlos en cualquier problema.
Ciencia	Una pregunta científica puede dividirse en experimentos más pequeños que al ser combinados resuelvan la pregunta más compleja. Inferir el ecosistema a partir de las cualidades de un animal.	La clasificación y el ordenamiento en biología se basa en la identificación de patrones comunes	Las leyes y teoremas científicos abstraen resultados de un conjunto de experimentos Los modelos científicos convierten sistemas complejos en solo la información necesaria	Los procedimientos científicos o los pasos a seguir en un experimento consisten en un conjunto de instrucciones a seguir en un orden concreto.
Lengua y Literatura	La escritura de un texto se puede descomponer en secciones (introducción, argumento central, conclusión) y combinarlas para crear un argumento.	Entender el concepto de fonema como patrón que se repite en diferentes palabras y mejorar la pronunciación de nuevas palabras	Un análisis de un libro consiste en identificar las principales ideas y temas de este abstrayéndose de los detalles menos importantes.	Las diferentes estructuras poéticas se pueden entender como un algoritmo (ritmo, métrica y el orden en el que las líneas deben escribirse)
Ciencias sociales	Las estructuras sociales (gobiernos, partidos, asambleas) suelen dividirse en secciones menores con diferentes responsabilidades Analizar los diferentes momentos históricos como un conjunto de elementos sociales y culturales que lo influenciaron	Analizar comportamientos similares en diferentes etapas históricas y sus causas.	El análisis de eventos históricos no se centra en los pequeños detalles sino en las tendencias sociales y temáticas del periodo. Los diferentes tipos de mapas suponen una abstracción de la información relevante (mapas físicos, políticos…).	Los procesos formales en los diferentes sistemas sociales pueden ser vistos como un conjunto de pasos a seguir en un orden específico.

Anexo C

Tablas de juegos analizados

Tras realizar el análisis de los juegos que se encuentra en el Capítulo 2, se tomó la decisión de realizar una serie de tablas para resumir la información. Estas tablas pretenden compactar este análisis.

Características I

Juegos	Seleccionar avatar	Coleccionar objetos	Evitar obstáculos	Enemigos	Actividades de programación		
					Diseñar un algoritmo	Arreglar errores en un programa	Otros
Programming Hero					☑		Modifica código para modificar su comportamiento. Se programa un meta-juego como objetivo durante el proceso de aprendizaje.
Grasshopper							Seleccionar la opción que produce el resultado deseado.
Code Hero			☑	☑	☑		Programar soluciones para avanzar de nivel.
CodeCombat	☑	☑		☑	☑		Modificar el código inicial que te da el nivel para completarlo de manera correcta. No arregla errores.
Duolingo	☑	☑					
Light Bot	☑	☑	☑		☑		Niveles diseñados para encontrar la solución correcta.
KIBO			☑		☑		Programar el encendido de luces como output de lo que se está ejecutando.
BeBlocky	☑	☑	☑		☑		Recolocar bloques de código que te dan como base del nivel.
Scratch JR	☑				☑		El jugador crea sus propios proyectos.
SpriteBox	☑	☑	☑		☑		Niveles de plataformas que superar resolviendo puzles de programación
CodeSpark Academy		☑	☑	☑	☑		Colocar las instrucciones para conseguir el objeto clave del nivel
May's Journey		☑	☑	☑	☑	☑	Programar soluciones para avanzar de nivel.
Blockly Games			☑		☑		Varios juegos diferentes en los que resolver problemas
Dragon Architect					☑		Modo sandbox en el que el jugador crea sus construcciones.

Características II

Juegos	Edad *denota el autor estimación	Variedad de actividades	
		Programar el movimiento de un personaje	Otros
Programming Hero	9+		Preguntas tipo test o de completar código para todos los conceptos.
Grasshopper	12+		Niveles específicos para enseñar ciertos conceptos.
Code Hero	12+		Mover, rotar o escalar objetos para superar niveles.
CodeCombat	9+	?	
Duolingo	10+		Poder realizar pruebas más complejas para saltarse niveles que son fáciles para ti.
Light Bot	10 - 12	?	
KIBO	6 - 10	?	Libertad para experimentar con comportamientos del robot.
BeBlocky	8 - 14*	?	Todos los niveles tienen objetivos a cumplir.
Scratch JR	5 - 7	?	Libertad a la hora de crear proyectos .
SpriteBox	7 a 9	?	
CodeSpark Academy	5 a 7	?	Modo crear donde diseñar niveles y programar comportamientos
May's Journey	N/D	?	Se programa el movimiento de elementos del escenario
Blockly Games	10 a 12		
Dragon Architect			

Conceptos de Programación

Juegos	Forma de Programación				Conceptos enseñados						
	Lenguajes de programación	Iconos visuales	Bloques de código	Otras representaciones	Secuencia de comandos	Variables	Condicionales	Bucles	Recursión	Funciones	POO
Programming Hero	Python; HTML			El código aparece en texto plano formateado.	☐	☐	☐	☐	☐	☐	☐
Grasshopper	JavaScript		☐		☐	☐	☐	☐		☐	
Code Hero	Java; C#			El código puede mostrarse en pantallas del entorno 3D.	☐	☐	☐	☐			
CodeCombat	Python; JavaScript; Coffee Script; C++ (con suscripción)			El código aparece en texto plano formateado.	☐	☐	☐	☐		☐	☐
Duolingo											
Light Bot		☐			☐				☐	☐	
KIBO		☐	☐	El código se representa por secuencias de bloques físicos.	☐		☐	☐			
BeBlocky	JavaScript (Blockly)		☐		☐		☐	☐			
Scratch JR		☐			☐		☐	☐		☐	
SpriteBox	Swift; Java	☐			☐	☐		☐			
CodeSpark Academy		☐			☐			☐			
May's Journey	Javascript			Texto plano	☐		☐	☐		☐	
Blockly Games	Blockly JavaScript		☐		☐		☐	☐		☐	
Dragon Architect	Blockly		☐		☐		☐	☐		☐	

Ayudas y evaluación

Juegos	Feedback			Ayudas			Evaluación			
	Indica qué instrucción se está ejecutando	Diferentes velocidades de ejecución	Feedback cuando se produce un error	Tutoriales	Ayudas Desbloqueables (Hints)	Lecciones antes de los problemas	Puntuación de nivel	Recompensas por compleción	Recompensa a la solución más correcta	Logros / Desafíos
Programming Hero			☐			☐	☐			
Grasshopper			☐	☐	☐	☐		☐		☐
Code Hero	☐		☐	☐		☐				☐
CodeCombat	☐	☐	☐	☐	☐	☐		☐	☐	☐
Duolingo			☐			☐	☐	☐		☐
Light Bot	☐	☐	No se permiten errores.	☐						
KIBO	☐		☐							
BeBlocky	☐	☐	☐	☐		☐	☐	☐		
Scratch JR	☐		No se permiten errores.	☐						
SpriteBox	☐	☐		☐		☐	☐			
CodeSpark Academy	☐			☐	☐	☐	☐	☐		
May's Journey		☐	☐	☐		☐				
Blockly Games	☐				☐					
Dragon Architect	☐	☐		☐						

Plataforma

Juegos	Web	Android	iOS	Escritorio	Disponible gratuitamente online
Programming Hero		?	?		?
Grasshopper		?	?		?
Code Hero				?	?
CodeCombat	?				?
Duolingo	?	?	?		?
Light Bot	?	?	?		?
KIBO					
BeBlocky		?			?
Scratch JR		?	?		?
SpriteBox	?	?	?		?
CodeSpark Academy	?	?	?		?
May's Journey	?				?
Blockly Games	?			?	?
Dragon Architect	?				?

Anexo D

Guía docente Mansión Paranormal

Durante la planificación de las pruebas de la Mansión Paranormal, se realizó una guía docente sobre el juego. El objetivo de la guía docente es servir como herramienta a los profesores durante la realización de pruebas en el aula.

La guía docente contiene un resumen del contenido y las mecánicas principales del juego así como los principales conceptos de cada puzle y cómo resolverlos.

Mansión Paranormal

JUEGO, PUZLES Y SU RELACIÓN CON EL PENSAMIENTO COMPUTACIONAL

La Mansión Paranormal es un **juego educativo de tipo narrativo** que pretende **fomentar el pensamiento computacional** (*Computational Thinking*) mediante **puzles** que el jugador debe superar para poder avanzar y resolver el misterio. El juego utiliza una idea similar a la de los *escape rooms*.

Los puzles incluidos en el juego están diseñados para cubrir los 4 aspectos en los que se divide el **pensamiento computacional**:

- **Descomposición**: transformar un problema complejo en varios problemas de menor complejidad.
- **Identificación de patrones**: encontrar secuencias que se repiten en uno o varios problemas.
- **Abstracción**: obtener la información importante de un problema obviando las partes innecesarias.
- **Algoritmos:** plantear la resolución de un problema como una serie de pasos a seguir.

En las páginas siguientes se describe la mecánica de los puzles o desafíos que presenta el juego y cómo se relacionan con cada uno de los aspectos anteriores.

Uso del juego

El público objetivo al que se dirige el juego es a usuarios de entre **10 a 16 años**.

El contexto de uso es **utilizar el juego como elemento motivador en clase bajo la dirección del profesor**. Por su carácter narrativo e interactivo creemos que el juego puede suponer una forma interesante de introducir estas competencias de una forma más interactiva, lúdica y motivante para los alumnos. El profesor podría posteriormente complementar el juego con una explicación de los conceptos de pensamiento computacional usados en el juego (p.ej., introducción informal del concepto de algoritmo).

La plataforma de juego es PC (Windows/Linux).

Narrativa del juego y Dinámica de juego

Eres un detective junior que debe resolver el misterio de la mansión Fernández Manjón: desde que el dueño desapareció han ocurrido cosas extrañas y se escuchan sonidos que provienen del despacho de la mansión. Al entrar, todas las

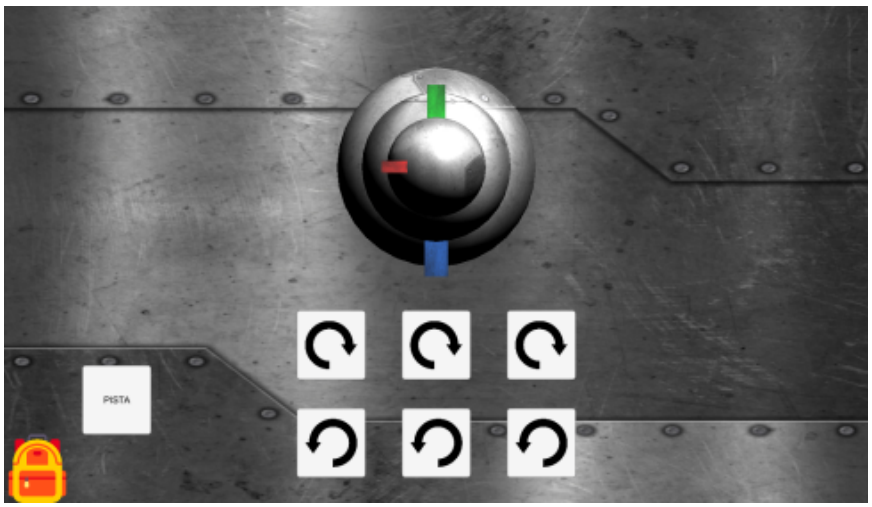
puertas se cierran. Debes resolver los puzles para poder escapar. Para resolver el caso contarás con la ayuda de los detectives hermanos John, Jim y Jack, que te darán pistas y consejos que te serán de utilidad para resolver los intrincados puzles. Además, se ha convocado al mayordomo de la mansión y a la señora Ada Mantine para ser interrogados.

El jugador puede explorar libremente la mansión moviéndose entre distintas escenas, donde puede interactuar con los elementos que se presentan en ellas como: personajes con los que hablar, objetos que inspeccionar, etc. Así podrá conocer a los personajes y los secretos que esconden, investigar los escenarios en busca de pistas y resolver el misterio que esconden las cuatro paredes de la mansión. Para avanzar el jugador tendrá que resolver distintos puzles.

Cada puzle cuenta con 3 niveles para ofrecer una dificultad creciente. En los puzles se proporcionan pistas para ayudar al jugador a encontrar la solución si no lo logra por sus propios medios. En la mansión se encuentran 4 puzles con mecánicas distintas que deben ser resueltos para acceder al puzle final, el cual permite entrar al despacho y resolver el misterio.

Puzles incluidos en el juego

Las anillas locas



- **Identificación de patrones.** El jugador debe identificar cuántos grados gira cada anilla con el uso de las distintas palancas. Las palancas provocan el giro de las anillas en sentido dextrógiro (en el sentido de las agujas del reloj) o en sentido levógiro (sentido contrario a las agujas del reloj). Estos patrones de giro se deben usar para la resolución del puzle.
  En todas las dificultades, cada columna de palancas se asocia a dos anillas empezando desde la anilla más pequeña hasta la anilla más

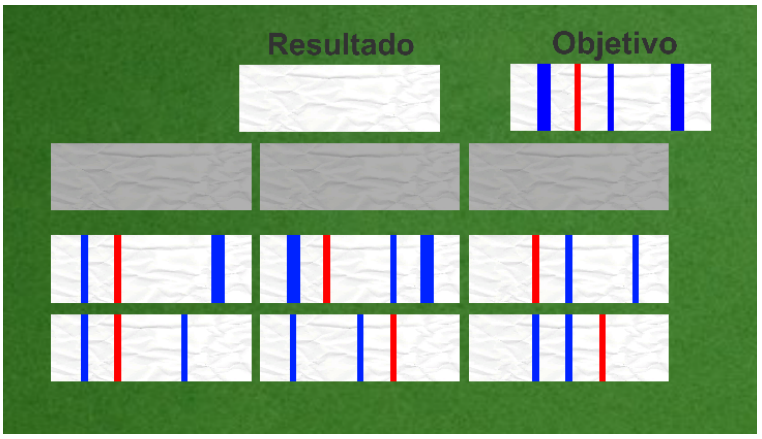
grande (la última palanca se asocia a la anilla más grande y a la más pequeña). En todos los casos la palanca girará la primera anilla asociada 45º y la segunda 90º. Hay tres niveles de dificultad con un número creciente de palancas y anillas.
Estos ángulos de giro hacen que las anillas puedan tener 8 posiciones diferentes, una por cada 45º. Es decir, en los ejes vertical, horizontal o en las diagonales.

	Dificultad		
	FÁCIL	**MEDIA**	**DIFÍCIL**
Nº de palancas (2 por anilla)	4	6	8
Nº de anillas	2	3	4

- Con el aumento de la dificultad y, en corcondancia con el aumento del número de anillas, el jugador deberá idear una forma coherente de **descomponer el problema** en problemas más pequeños. Esto incluye cómo colocar todas las anillas en la misma diagonal o colocar todas las anillas en un mismo eje.

- **Diseñar un algoritmo** usando los patrones identificados y la descomposición de problemas para conseguir llegar a la solución.

- Este algoritmo se puede resumir en los siguientes pasos:
    - Colocar todas las anillas en la misma ”X” (la formada por los ejes horizontal y vertical o por las diagonales)
    - Colocar todas las anillas en una recta de la X (pulsando las palancas un múltiplo de 2 veces)
    - Repetir hasta completar el puzle:
        - Pulsar la palanca asociada a la anilla que se quiere alinear 4 veces (la otra anilla volverá a su posición inicial)



- **Identificación de patrones.** El jugador debe identificar cómo se combinan las distintas tarjetas entre sí. Debe llegar por sí mismo a la conclusión de que el color azul significa "suma" y el rojo "resta". Además, también deberá **relacionar** el ancho de las bandas de colores con la cantidad que representan, ya que una banda azul de ancho 3 combinada con una banda roja de grosor 2 dan una banda azul de ancho 1 (+3 + -2 = +1). El ancho de cada banda puede tomar valores en el rango [-4, 4]. El número de bandas por tarjeta aumenta con el nivel de dificultad del puzle siendo 4 en el primero, 5 en el segundo y 6 en el tercero. Además desde el segundo nivel, pasan a ser 3 las tarjetas que forman la solución en lugar de las 2 del primer nivel.

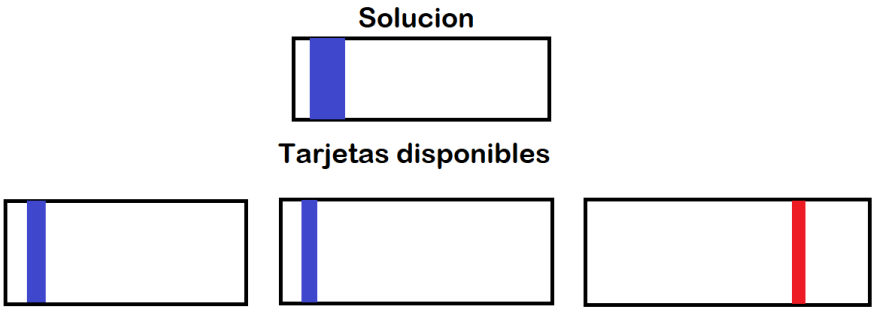
	Dificultad		
	FÁCIL	**MEDIA**	**DIFÍCIL**
No. de bandas	4	5	6
No. de tarjetas solución	2	3	3

- **Depuración.** Para conseguir que el jugador tenga realimentación y vea el resultado que va obteniendo al combinar tarjetas, hay una tarjeta "Resultado" que muestra dicho progreso. Con esta realimentación visual, mediante prueba y error, el jugador debe ser capaz de identificar los patrones mencionados anteriormente y poder llegar a obtener las relaciones que se buscan.

- En los siguientes niveles de dificultad se introducen más columnas a las tarjetas y un mayor número de tarjetas que forman la solución. Se busca que el jugador entienda que **descomponer la búsqueda de**

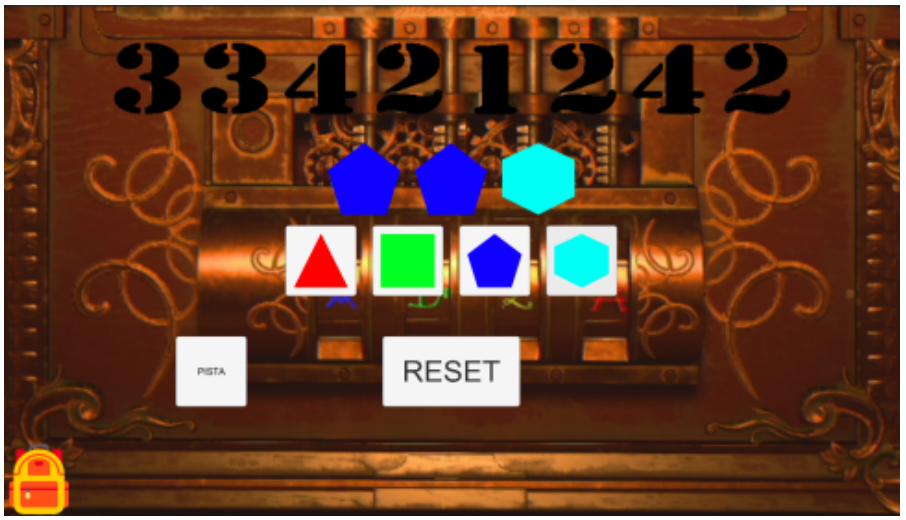
la solución en grupos de tarjetas o incluso en segmentos concretos facilita la obtención de la solución correcta.

- Se dan situaciones en las que alguna de las tarjetas puede no formar parte de la solución porque al combinar ciertas secciones con las de las demás tarjetas, la columna resultante es diferente a la de la solución. En la imagen se ve que la tercera tarjeta, aunque se combine con otra, nunca va a poder formar parte de la solución. El jugador puede **ignorar** la información innecesaria y no tener en cuenta estas tarjetas para la solución. También ocurre en el caso contrario, que claramente una de las tarjetas tenga que formar parte de la solución porque es la única que produce una sección correcta. Teniendo esto en cuenta, el jugador deberá **diseñar el algoritmo** que combine las tarjetas de la forma adecuada para llegar al resultado buscado.



- El **algoritmo** a seguir es el siguiente:
  - Analizar la solución que se pide y las tarjetas dadas.
  - Descomponer la solución en en grupos de tarjetas o de segmentos.
  - Emparejar tarjetas que den la solución o que se acerque a ella.
  - Utilizar la tarjeta resultado (la de depuración) como una tarjeta más y combinarla con una tercera tarjeta dando el resultado final.

Formas y colores



- **Abstracción.** El jugador debe averiguar qué relación existe entre la secuencia numérica mostrada y las formas geométricas con colores que se pueden introducir. De esta forma, tiene que eliminar mentalmente la información irrelevante (por ejemplo, la posición de los botones o los colores de las formas geométricas) y darse cuenta de que la relación que le llevará a la solución es **nº secuencia = nº de lados - 2** o bien **nº de lados = nº secuencia + 2.**

- **Depuración.** El jugador puede ver en todo momento la solución que está escribiendo y compararla con la secuencia de números (que es la solución objetivo que tiene que lograr). Además, por si se equivoca, dispone de un botón “Reset” para borrar la solución que estaba planteando y empezar de nuevo.

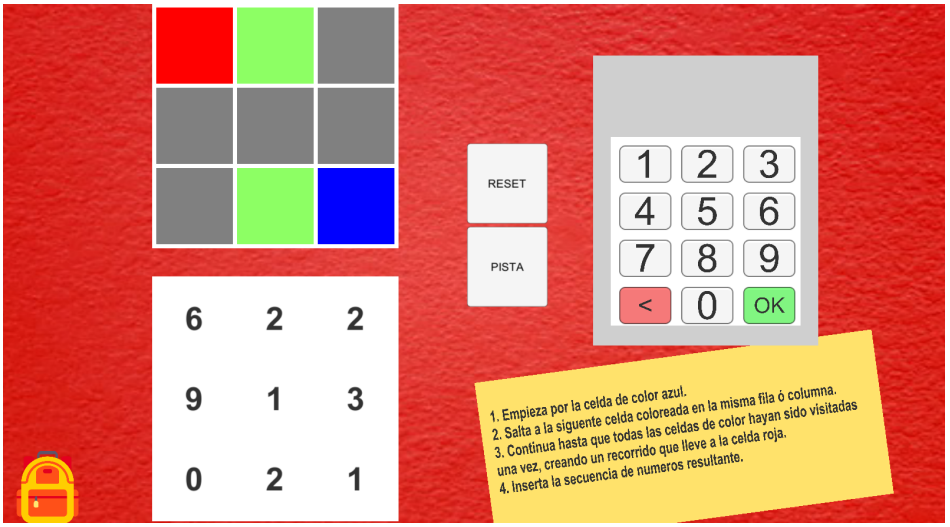
- En las diferentes dificultades se añaden más elementos a la secuencia, se cambia el orden y se alteran los diferentes botones.

	Dificultad		
	FÁCIL	**MEDIA**	**DIFÍCIL**
Longitud Secuencia	6	8	10
Orden aleatorio	No	Sí	Sï
Formas irregulares	No	No	Sí

- El **algoritmo** al que tiene que llegar el jugador es:
    - Recorrer la secuencia de números y repetir por cada uno los siguientes pasos:
        - Coger el número y sumarle dos.

- Pulsar la forma geométrica con el número de lados igual al resultado del paso anterior.

Laberinto de números



- **Abstracción.** Al jugador se le presentan dos tableros, uno con colores y otro con números. El jugador solo tiene en cuenta los números que le interesan para conseguir el código, ignorando completamente el resto.

- El **diseño del algoritmo** no debe realizarlo el jugador, si no que se le da en forma de lista de pasos. Aun así, el jugador debe de ser capaz de entender el funcionamiento de dicho algoritmo y saber aplicarlo para conseguir la solución del puzle.

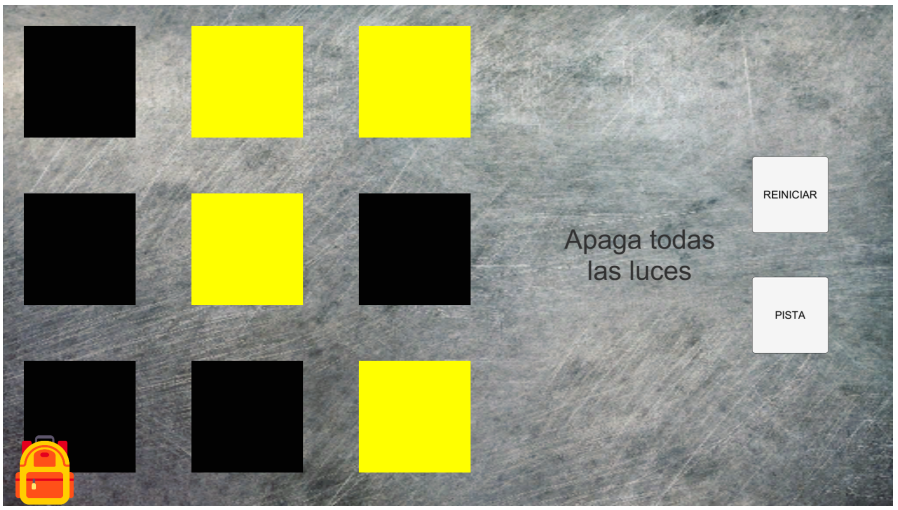
- Al aumentar los niveles de dificultad aumentan tanto la longitud de la secuencia a introducir así como el tamaño de la matriz.

	Dificultad		
	FÁCIL	**MEDIA**	**DIFÍCIL**
Longitud Secuencia	4	6	8
Tamaño Matriz	3x3	5x5	7x7

- El **algoritmo** que el jugador debe encontrar, entender y aplicar es el siguiente:
  - Empezar en la celda de color azul
  - Repetir estos pasos hasta llegar a la celda roja (incluida):

- Introducir el número con posición equivalente a la de la celda actual en el teclado numérico.
- Realizar un salto a una celda coloreada que se encuentre en la misma fila o columna, ignorando las celdas previamente visitadas.

Electricista



- **Identificación de patrones.** El jugador debe ser capaz de identificar que al interactuar/activar una casilla, las casillas adyacentes también cambian de estado (si está apagado se enciende y viceversa).

- Lograr una mayor **abstracción** al entender la lógica de estados.

- Al incrementarse el nivel de dificultad aumentan tanto el tamaño del tablero como el número de casillas pulsadas para crear el estado inicial del tablero.

	Dificultad		
	FÁCIL	**MEDIA**	**DIFÍCIL**
Tamaño del tablero	5x5	4x4	3x3
Pulsaciones Iniciales	5	5	7
Número de estados	2	2	2

- Similar a uno de los problemas de ejemplo del bebras contest (programming lights) https://www.bebras.org/examples.html

- El **algoritmo** a encontrar debería ser:

- Empezar apagando las luces que apaguen las máximas casillas adyacentes.
- Interactuar con las casillas sobrantes.
- Repetir los dos primeros pasos desplazando las luces encendidas a las esquinas hasta llegar a una solución.

Otros aspectos del juego

Gamificación

Para aumentar la motivación del jugador, se ha introducido un sistema de puntuación del progreso de los usuarios en cada nivel de dificultad de cada uno de los minijuegos. Este sistema se basa en estrellas, el jugador puede obtener hasta 3 estrellas por nivel de dificultad. Cada una de las estrellas se otorgan en función de los siguientes parámetros:

- La **primera estrella** se otorga por completar el minijuego, de esta forma el jugador siempre recibirá al menos una estrella, para animarle a continuar en el juego.

- La **segunda estrella** se consigue si el jugador consigue resolver el puzle en un número determinado de pasos. Este número de pasos permite un pequeño margen de fallos para mantener motivado al jugador y tratar de evitar un comportamiento aleatorio.

- La **tercera y última estrella** se entrega siempre que el jugador no haya usado pistas para resolver el problema tratando de motivar al usuario a buscar la solución sin ayuda.

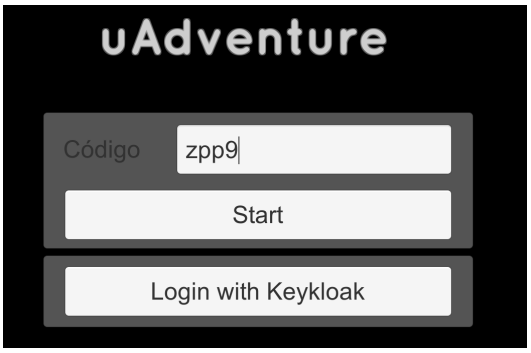
- El **tiempo empleado** no tiene ninguna influencia en las estrellas asignadas ya que se busca que el jugador reflexione, piense y se esfuerce en buscar una solución, tratando de evitar comportamientos aleatorios que podrían motivarse con una restricción de tiempo.

Rejugabilidad y configuración de los puzles

Al tratarse de un juego planteado como una “escape room”, el acceso a los minijuegos (puzles) está restringido a una única vez por juego. Esto quiere decir que una vez que se completa un puzle, no es posible acceder a él otra vez. Se podría volver a jugar al comenzar de nuevo el juego desde cero, ya que la configuración inicial de todos los puzles se genera de forma aleatoria. De esta manera, cada partida tendrá un caso diferente de cada uno de los puzles y su solución será diferente. Por un lado, esto impide que un jugador que ya ha jugado el juego le cuente la solución a otro jugador que todavía no ha jugado. Por otro lado, dos jugadores que estén jugando simultáneamente en dos ordenadores diferentes no pueden compartir soluciones y tienen que resolver cada uno su puzle por separado.

Ejecución del juego y uso de códigos

Para poder acceder al juego, se requiere un código de acceso. Una vez introducido el código en la pantalla inicial del juego, hay que pulsar el botón “Start”.



Estos códigos sirven para identificar las distintas sesiones de juego y poder recoger información de forma anónima. Antes de las pruebas, os proporcionaremos una lista de códigos (en un fichero PDF), para que los asignéis a vuestros alumnos. Cada alumno utilizará su código único para acceder al juego. De esta forma, podremos obtener la información de interacción sin saber quienes son los alumnos concretos a los que pertenece cada código (pseudo-anonimización en origen).

Group: 04-3-2021:					
No.	Nombre	Código			
1		P987	P987	P987	P987
2		ZXBL	ZXBL	ZXBL	ZXBL
3		3XLF	3XLF	3XLF	3XLF

La información que se recoge se fundamenta en el comportamiento del usuario. Por ejemplo, se capturan las decisiones que han tomado en algunas conversaciones, los pasos que han seguido para resolver los

distintos puzles, etc. En resumen, se hace un seguimiento de la sesión de juego del usuario. Por este motivo, se requiere conexión a internet y un código distinto para cada usuario.

La información recogida se usará para determinar si el juego cumple el objetivo de potenciar los aspectos del pensamiento computacional esperados y para mejorar la experiencia de usuario. En ningún momento se recoge información personal del usuario ni del equipo con el que se juega. A continuación, se muestra un extracto del tipo de información que se envía al servidor de la Universidad Complutense de Madrid.

```
1615206441725,accessed,area,Outside
1615206441744,initialized,game,La mansión paranormal
1615206441745,progressed,game,La mansión paranormal,progress,0
1615206445362,initialized,completable,MetaPuzle
1615206445362,progressed,completable,MetaPuzle,progress,0
1615206445478,used,gameobject,INITIAL STATE: Cells Active -> |Value(X Y)|: |1(0 0)| |1(1 0)| |1(2 0)| |1(0 1)| |1(0 2)| |1(2 2)|
1615206447152,used,gameobject,GridBox 0 0 checked
1615206447917,used,gameobject,GridBox 2 2 checked
1615206470811,used,gameobject,GridBox 1 0 checked
1615206471454,used,gameobject,GridBox 1 2 checked
1615206474440,progressed,completable,MetaPuzle,PUZZLE_3_DIFICULTY,1,progress,0.3333333
```

Anexo E

Guía docente Articoding

Al igual que con la Mansión Paranormal, se realizó una guía docente sobre el juego. En este caso se hace una descripción más detallada del contenido del juego, los modos de juego, las tarjetas disponibles, la progresión de niveles, la configuración del tablero, los elementos del tablero.

Articoding

Guia docente

Autores
Dany Faouaz Santillana
Arturo García Cárdenas
Álvaro Poyatos Morate

Directores
Prof. Dr. Baltasar Fernández Manjón
Dr. Antonio Calvo Morata

¿Qué es Articoding?

Articoding es un **juego serio educativo** que tiene como objetivo **fomentar el desarrollo del pensamiento computacional** (*Computational Thinking*) mediante **la enseñanza de conceptos básicos de programación usando programación visual**. El jugador debe superar niveles resolviendo problemas que se plantean en un escenario en forma de tablero. El objetivo del jugador es conseguir guiar el haz de luz de un láser hacia un objetivo utilizando la programación visual por bloques (similar a Scratch). Cuando se alcance el objetivo, el jugador puede avanzar al siguiente nivel.

Los niveles diseñados incluidos en el juego introducen al usuario de manera progresiva los conceptos básicos de programación (e.g., variables, tipos de datos, bucles), ofreciendo una experiencia de aprendizaje motivadora y agradable.

En las páginas siguientes se describe el juego y sus contenidos y cómo se relacionan con cada uno de los aspectos anteriores.

Contextos de uso del juego y público objetivo

El público objetivo al que se dirige el juego es jóvenes de entre **12 a 16 años**.

El contexto de uso es **utilizar el juego como elemento motivador en clase bajo la supervisión del profesor**. Por su carácter interactivo y visual el juego supone una forma interesante de introducir los conceptos de programación y las competencias del pensamiento computacional de una forma interactiva, lúdica y motivante para los alumnos. El profesor puede complementar la actividad de juego con una explicación de los conceptos programación y de pensamiento computacional usados en el juego (e.g., introducción informal del concepto de algoritmo).

Por su carácter progresivo también puede usarse en modo no supervisado, de forma que los alumnos ejerciten y apliquen conceptos de programación que pueden haber sido previamente presentados en clase (**modo de ejercicios**). Además, el juego introduce los principales elementos a utilizar, proporcionando tutoriales e información para que se pueda usar incluso de forma libre o como actividad complementaria para alumnos que hayan acabado sus asignaciones de clase (**modo libre**).

Por último, el juego también ofrece un **modo de creación de niveles.** En este modo los estudiantes tienen que crear sus propios tableros o niveles de juego. Una vez creado el nivel, el jugador tiene que proporcionar además una solución a su propio nivel. De esta forma se garantiza que el nivel creado sea válido y solucionable. Cuando el jugador valida un nivel que ha creado, se guarda para poder jugarlo más tarde. Desde el punto de vista educativo, este modo de juego

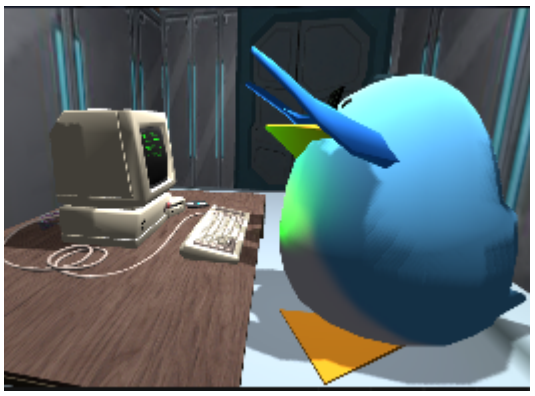
ofrece la oportunidad a los jugadores de aplicar los conceptos aprendidos de una manera diferente y más creativa. Además motiva a los jugadores a seguir aprendiendo nuevos conceptos para así crear niveles más complejos. Este modo puede ofrecer a los profesores evidencias de lo que realmente han aprendido los alumnos con el juego.

La plataforma de juego es PC (Windows/Linux) para jugar en un ordenador con ratón.

Articoding cuenta con dos versiones, una progresiva por niveles, en las que el jugador debe ir resolviendo los niveles disponibles para desbloquear nuevos y así avanzar. Y otra con todos los niveles abiertos y directamente accesibles al jugador para que el profesor pueda utilizarla en clase indicando a los alumnos qué aspectos específicos desea que practiquen. En este caso es deseable que los alumnos tengan experiencia previa con el juego y estén familiarizados con los elementos de juego y la interacción con el entorno.

Narrativa y dinámica de juego

El protagonista del juego es Albert, un pingüino científico que ha sido encerrado en un laboratorio lleno de ordenadores, láseres, espejos y otros obstáculos. Albert debe escapar del laboratorio haciendo uso de sus dotes como programador resolviendo cada nivel para avanzar al siguiente.

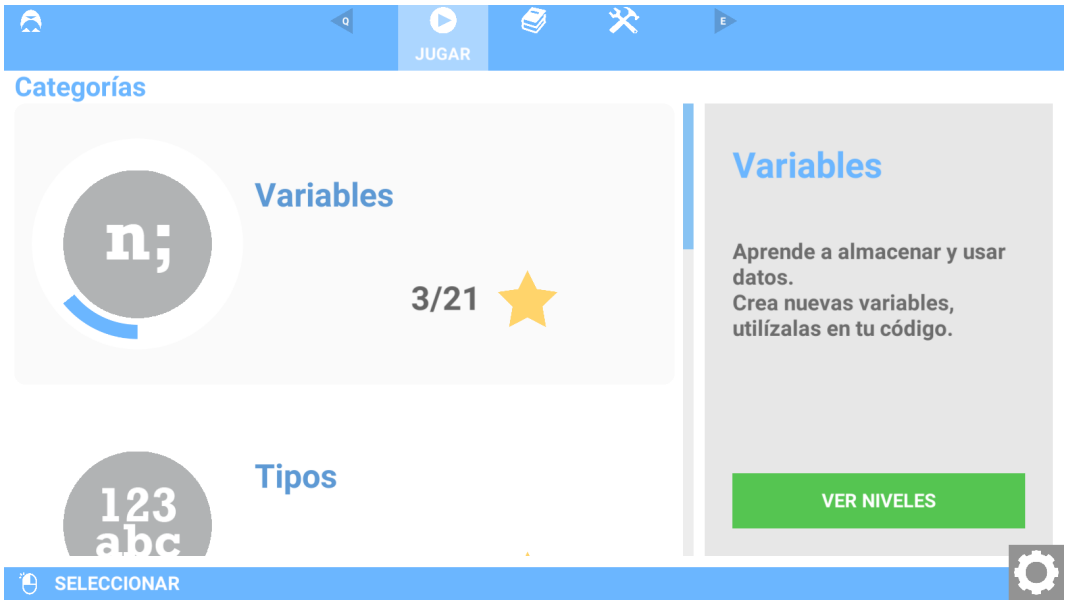


Al inicar el juego, el jugador empieza en el menú principal. Desde el menú principal se tiene acceso a estas tres secciones:

- Sección de **JUGAR**.
- Sección de **TUTORIALES**.
- Sección de **CREACIÓN**.

Sección de JUGAR (Modo normal)

En esta sección el jugador puede seleccionar el nivel que quiere jugar dentro de los que tiene desbloqueados. En esta sección se encuentran los niveles donde se enseñan los contenidos principales del juego.



Sección de JUGAR en el Menú Principal de Articoding

Cuando se juega un nivel por primera vez, se le muestran al jugador los tutoriales necesarios para explicar los elementos y bloques de código que se introducen en ese nivel. Una vez explicados, estos conceptos se añaden a la ***sección de tutoriales***.

Sección de TUTORIALES

En esta sección el jugador puede consultar lo aprendido cuando quiera. Esta sección puede encontrarse tanto en el menú principal como en la pantalla de juego. De esta manera, el jugador puede consultar y repasar en cualquier momento los conceptos aprendidos.

Sección de TUTORIALES en el Menú Principal de Articoding

Sección de TUTORIALES en la pantalla de juego de Articoding

Sección de CREACIÓN

En esta sección se podrá acceder al Modo Creación, un modo en el que el jugador puede crear y resolver nuevos niveles usando su creatividad y aplicando los conceptos aprendidos.

Sección de CREACIÓN en el Menú Principal de Articoding

Además, en esta sección también se encuentran todos los niveles previos que hayan sido creados de forma válida en el Modo Creación.

Pantalla de Juego

La pantalla de juego de los niveles se divide en tres zonas diferenciadas. En primer lugar en la parte superior está el encabezado donde se encuentran de izquierda a derecha el logo del juego, el nombre del nivel que se está jugando, el botón de play (para ejecutar el código), el botón para usar pistas, el botón para volver a leer los tutoriales desbloqueados y el botón de salir.

Debajo del encabezado, en la parte izquierda, se encuentra el panel de código. Es aquí donde se colocan las tarjetas para solucionar el nivel. En este panel, a la izquierda se encuentra el inventario de tarjetas, desde donde se arrastran a la zona de código. Además si se arrastra una tarjeta de la zona de código al inventario la tarjeta se eliminará. También se pueden duplicar tarjetas de código pulsando click derecho sobre ellas. Solo se duplicará la tarjeta si se dispone de las tarjetas necesarias en el inventario. Al duplicar una tarjeta, todas las tarjetas que estén enganchadas a ella (hacia abajo) se duplicarán.

Por último, en la parte derecha inferior al encabezado, está la vista del juego. Aquí se muestra el tablero y los elementos del nivel.

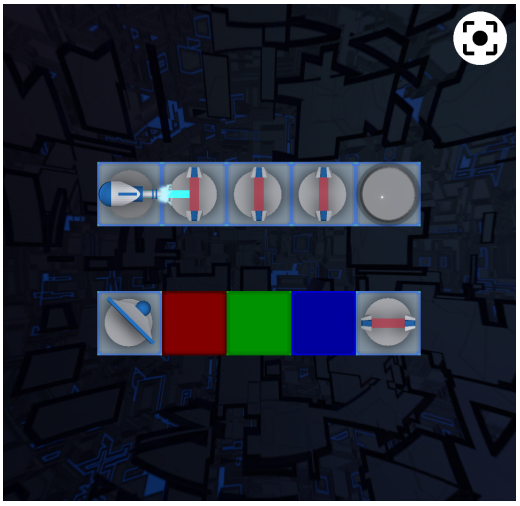
Tablero y elementos del juego

Todos los niveles se presentan en forma de tablero y estos están formados por casillas. A continuación se muestran los elementos que pueden aparecer en el tablero de juego y se describe cuál es su funcionalidad.

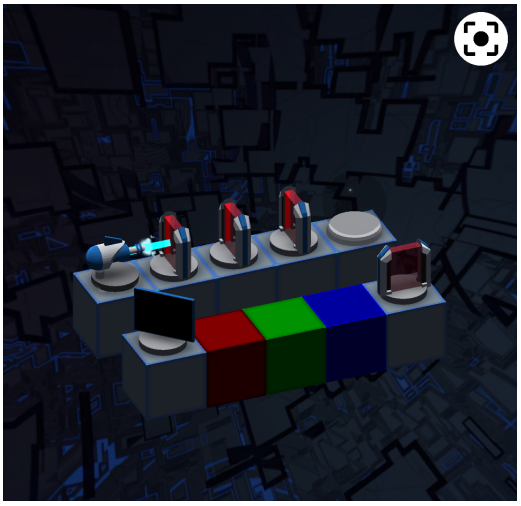
El tablero consiste en un rectángulo formado por casillas, sobre las que se colocan los diferentes elementos. Las casillas pueden ser de 5 tipos (Normal, Vacía, Roja, Verde o Azul). Si un elemento se mueve a una celda de tipo Vacía, el elemento se caerá del tablero y no se podrá seguir utilizando. Si el elemento que cae es un láser, el nivel se considerará fallado. Todos los tableros están rodeados de celdas vacías.

Todos los tableros se empiezan mostrando en vista cenital[58]. Aún así, el jugador también tiene la posibilidad de visualizar el tablero en 3D. Para pasar a esta vista y ver el tablero desde otro ángulo, basta con pulsar click izquierdo y arrastrar el ratón sobre la vista de juego.

[58] Visto desde arriba



Vista cenital de un nivel de Articoding



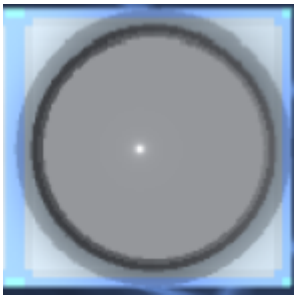
Vista 3D de un nivel de Articoding

Ya que el juego requiere tener en cuenta una referencia espacial, se puede volver a la vista cenital pulsando el botón de reseteo de la vista. El botón tiene el siguiente aspecto.

Botón de reseteo de la vista cenital

Esto es un **Emisor de Láser**. Este es el elemento principal del tablero junto con el Receptor del Láser. El jugador puede **mover** y **girar** este elemento por el tablero. También se puede **cambiar la intensidad** del láser para modificar su estado.El láser estará apagado si su intensidad es 0 o encendido si la intensidad es mayor que 0. El objetivo es hacer que el haz de luz del Emisor de Láser llegue a un Receptor de Láser.

Esto es un **Receptor de Láser**. Este elemento se activa si un haz de luz de un láser lleva a él. Este elemento **no se puede mover ni girar**.
Cuando todos los elementos del tablero de este tipo estén activos, se **completa el nivel**.

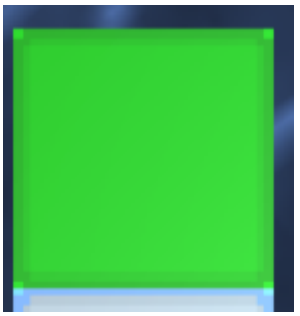


Esto es un **Espejo**. Este elemento deflecta los haces de luz de los láseres. El jugador puede **mover** y **girar** los espejos de este tipo. Los espejos son útiles para cambiar la dirección de un haz de luz en 90º.

Esto es una **Puerta**.Una puerta bloquea los haces de luz cuando se encuentra cerrada. Este elemento **no se puede mover ni girar**, pero si se puede **abrir** o **cerrar**.

Esto es un **Obstáculo**. Este obstáculo bloquea los haces de luz de los láseres. También bloquea el movimiento del resto de elementos del tablero. Este elemento **no se puede mover ni girar**.

Esto es una **Celda Especial**. Esta celda puede ser de tres colores diferentes: **rojo**, **azul** o **verde**. Al formar parte del suelo del tablero, estas celdas pueden tener objetos encima. El jugador podrá **comprobar** si todas las celdas de un color tienen algún objeto. **No se pueden mover ni girar**.



Resolución de niveles predeterminados del juego

Para resolver un nivel y poder avanzar en el juego, el jugador puede elegir y añadir instrucciones en el código visual y podrá comprobar el resultado de la ejecución. Si el jugador no logra resolver el nivel se le ofrecerán pequeñas ayudas o pistas para que no se quede atascado. Estas ayudas consisten en pistas visuales sobre los movimientos o cambios que tienen que realizar los objetos del tablero (además de movimientos hay también cambios de estado como activar el láser). Una vez el jugador obtenga la solución correcta, el nivel se dará por completado. Además, se proporcionará realimentación al jugador sobre cómo de bien se ha resuelto el nivel. El jugador podrá decidir si continúa o no con el siguiente nivel. Esta estructura de progreso es similar en todos los niveles.

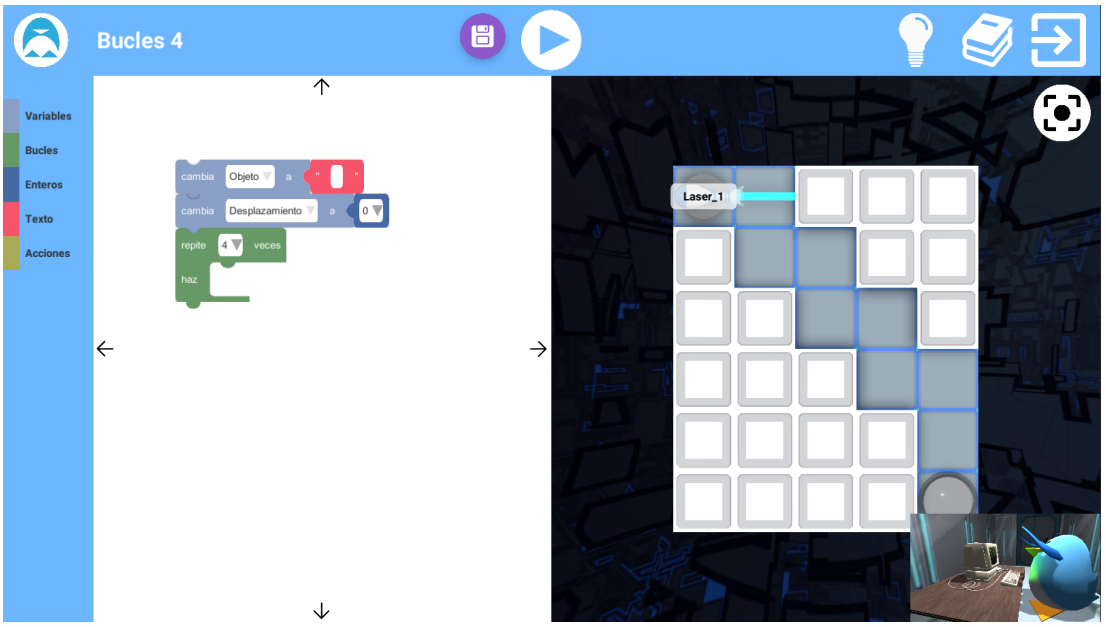
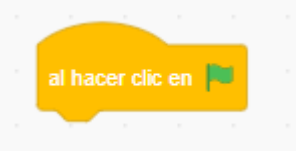


Imagen de la pantalla de juego de Articoding

Hay que destacar que a diferencia de Scratch el juego no dispone de un bloque de entrada desde el que se ejecuta el código. En su lugar, el código se ejecuta de arriba abajo, empezando por la tarjeta de más arriba.



Bloque de entrada de Scratch (no existe en Articoding)

Modo de creación de niveles por parte del jugador

Adicionalmente, el juego ofrece un modo creación al jugador con el que podrá crear sus propios niveles. Para que un nivel creado se considere válido, el usuario deberá ser capaz de proporcionar una solución correcta para el nivel que ha creado.

Este modo de juego empieza con un tablero vacío y debajo una fila con los elementos que se pueden seleccionar y añadir al tablero. Primero el jugador debe decidir el tamaño del tablero a crear expresado en número de filas y columnas. Para colocar un objeto en el tablero el jugador debe arrastrar el objeto desde la fila de selección hasta el tablero. los objetos que hayan sido colocados en el tablero, pueden ser girados pulsando el click izquierdo del ratón sobre ellos. Además, hay objetos como la puerta o el láser que pueden tener varios estados (e.g.,abierta o cerrada, encendido o apagado). Cuando se selecciona uno de estos elementos, aparece un panel en la esquina superior izquierda donde se puede cambiar el estado del objeto.

También es posible cambiar el tipo de celda del tablero. Para esto, basta con hacer click derecho sobre la celda que se quiere cambiar. La rotación de celdas es la siguiente: Normal -> Vacía -> Roja -> Verde -> Azul -> Normal. Inicialmente el tipo de todas las celdas es el normal.

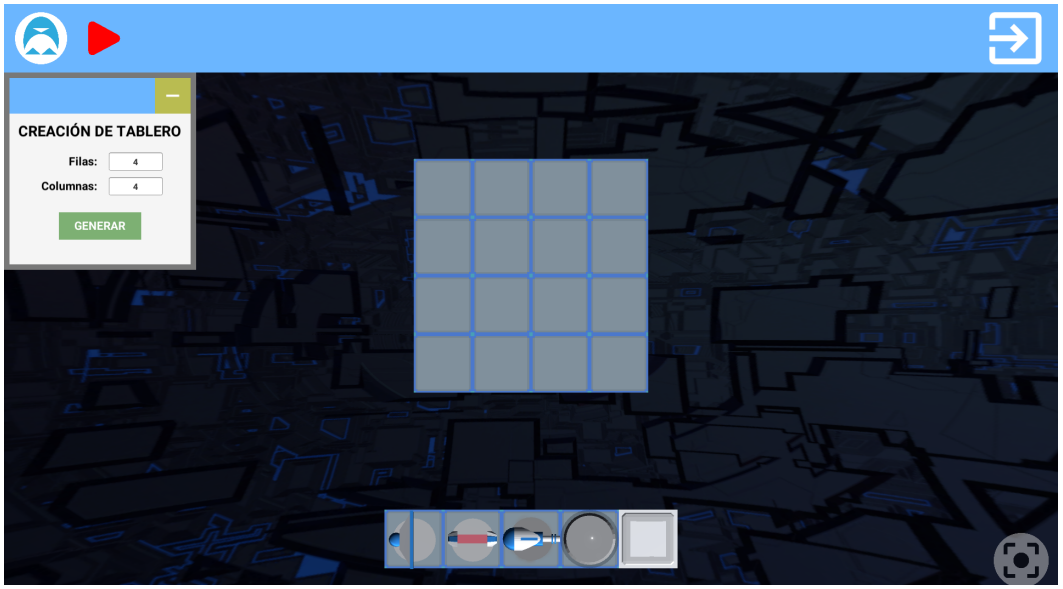


Imagen del modo Creación de Articoding

Una vez el jugador crea un tablero, es necesario solucionarlo para que se guarde. Para poder solucionar un nivel creado, el tablero tiene que tener el mismo número de láseres que de receptores, con al menos un receptor apagado. Si no se cumplen estas condiciones el botón de *play* de la esquina superior izquierda aparecerá en *rojo*. Cuando se cumplan estas condiciones, el boton play cambiará de color a

blanco y se podrá ejecutar para solucionar el nivel (si las instrucciones proporcionadas son correctas).

La resolución del nivel creado funciona igual que la resolución de uno de los niveles por defecto del juego. La diferencia es que en este modo no hay ningún límite a los bloques que se pueden utilizar. Si el jugador es capaz de resolver el nivel que ha creado, se considera un nivel válido y se guarda. Como se ha mencionado previamente, el jugador puede acceder a estos niveles creados desde el menú de creación en el menú principal.

Contenidos incluidos en el juego

Todas las categorías de niveles cuentan con un nivel de introducción donde la interacción del jugador se limita de modo que no se pueden ni crear ni borrar bloques. Estos niveles introducen de manera simple y visual el concepto o los conceptos que se tratarán en esa categoría. En estos niveles el jugador solo tiene que entender el código y darle al play para visualizar la ejecución.

Cada nivel cuenta con unos bloques determinados. Estos son diferentes en cada nivel. Además, existe un número de usos máximos por cada bloque del nivel. Los niveles están diseñados de tal forma que todos sean completables con los bloques que se proporcionan. En la mayoría de los casos, este límite de bloques se utiliza para forzar al jugador a utilizar el concepto que se quiere enseñar. (p.ej. si se le quiere enseñar que una variable puede utilizarse con el mismo valor en varias tarjetas, se le limitan las tarjetas de enteros para que tenga que reutilizar una variable en distintas partes del código).

Las tarjetas de código se pueden dividir en dos tipos: tarjetas de instrucción y tarjetas de valor. Por un lado las **tarjetas de valor** devuelven un valor y solo tienen una conexión saliente en su lado izquierdo. El valor que devuelven depende del tipo de tarjeta. El valor de la tarjeta se puede introducir directamente en algunos casos. En otros se modificara el valor de la tarjeta mediante un desplegable.

Por otro lado las **tarjetas de instrucciones** tienen una conexión entrante en la parte superior y una conexión saliente en su parte inferior (se pueden conectar otras tarjetas de instrucciones por arriba y por abajo). Además, estas tarjetas pueden tener **huecos** en los que se pueden conectar tarjetas de valor (actúan como parámetros de la instrucción). Dentro de la tarjeta también puede haber campos modificables mediante desplegables.

Conexión entrante de una tarjeta

Conexión saliente de una tarjeta

Hueco para tarjeta de valor

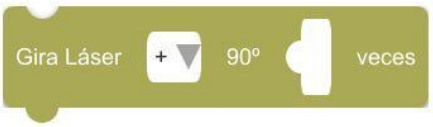
Campo desplegable de una tarjeta de instrucciones

Los niveles del juego se clasifican en 5 categorías distintas, cada una con 7-9 niveles:

1. Variables

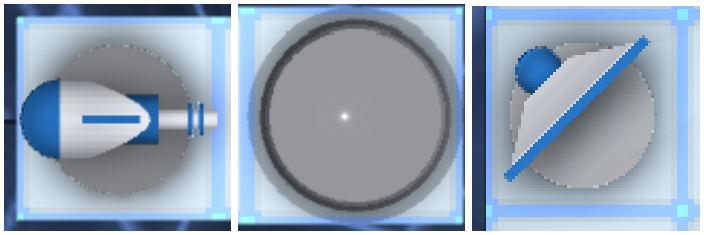
En esta categoría se introduce el concepto de lo que es una variable. Durante la compleción de la categoría, el jugador utilizará repetidamente las variables. La representación visual de las instrucciones tiene “huecos” o partes vacías donde se pueden añadir otras tarjetas (e.g., parámetros, valores).

Tarjetas introducidas

	Mueve el láser una cantidad de casillas (parámetro 1) en una dirección.
	Gira el láser un múltiplo 90 grados en una dirección (indicado por el signo + o - si es en sentido contrario) un número de veces (ese número de veces viene dado por el parámetro 1)
	Tarjeta que crea un valor numérico

	Tarjeta para modificar el valor de una variable (en este caso la variable desplazamiento)
	Tarjeta para obtener el valor de una variable
	Cambia la intensidad del láser a un valor numérico (parámetro 1) 0 apagado, >0 encendido

Elementos introducidos: laser, receptor, espejo

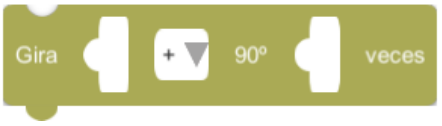


2. Tipos de datos

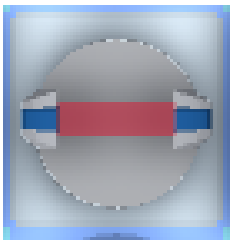
En la categoría de tipos de datos se introducen los distintos valores que puede guardar en las variables. En esta categoría se les introducen las cadenas de texto, los booleanos, etc.

Tarjetas introducidas

	Tarjeta que devuelve una cadena de texto. Utilizado para referirse a los objetos.

	Tarjeta que devuelve un valor lógico o booleano (su valor puede ser falso o cierto).
	Mueve un objeto (indicado por el parámetro 1) una cantidad de celdas (dado por parámetro 2) en una dirección (en este caso a la derecha).
	Cambia el estado de una puerta (parámetro 1) a un valor booleano (parámetro 2)
	Gira un objeto(parámetro 1) un múltiplo 90 grados en una dirección un número de veces(parámetro 2)

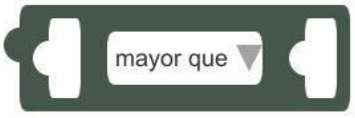
Elementos introducidos: puerta



1. Operadores básicos

En esta categoría el jugador aprenderá cómo puede manipular los datos. Se les enseña a construir expresiones booleanas, operaciones matemáticas, etc.

Tarjetas introducidas

	Devuelve el resultado de la (suma, resta, multiplicación ó división) de dos valores numéricos
	Devuelve el resultado de la comparación (>, <, >=, <= ó !=) de dos valores numéricos.
	Devuelve el resultado de la (conjunción o disyunción) de dos valores booleanos.
	Devuelve como resultado la negación de un valor booleano

	Devuelve verdadero si todas las celdas de un color están ocupadas, si no devuelve falso.

Elementos introducidos: celda especial de color

2. Bucles

En esta categoría se presenta por primera vez el concepto de bucle. Una parte del código que se repite un número de veces determinado.

Tarjetas introducidas

	Repite un conjunto de instrucciones un número de veces.

3. Condicionales

Utilizando el concepto de bucle visto anteriormente, se aprovecha para introducir un bucle de tipo while (que se ejecuta mientras se cumpla una condición). Utilizando este concepto, el jugador utilizará los condicionales (if, if-else) para modificar el comportamiento en ejecución de un bucle.

Tarjetas introducidas

	Evalúa una condición booleana. Si se cumple, ejecuta el conjunto de instrucciones que se encuentre en el cuerpo “haz”. Si no se cumple, ejecuta el conjunto de instrucciones que se encuentre en el cuerpo “si no”.
	Repite un conjunto de instrucciones mientras que una condición booleana se cumpla.

Otros aspectos del juego

Realimentación al jugador: sistema de estrellas

Para aumentar la motivación del jugador, se ha introducido un sistema de puntuación del progreso de los usuarios en cada nivel de dificultad de cada uno de los niveles. Este sistema se basa en estrellas, el jugador puede obtener hasta 3 estrellas por nivel. Cada una de las estrellas se otorgan en función de los siguientes parámetros:

- La **primera estrella** se otorga por completar el nivel en la primera ejecución. De esta forma se recompensa al jugador por plantear y resolver el problema de forma reflexiva tratando de que no abuse de la prueba y error.

- La **segunda estrella** se consigue si el jugador resuelve el nivel en el mínimo número de pasos posible (distinto en cada nivel). Se considera un paso cada acción que modifica el estado del tablero (mover, girar o cambiar estado de un objeto). Esto motiva que el jugador busque las soluciones óptimas de los problemas que se le plantean.

  Cuando el jugador hace más pasos de los necesarios, en la estrella correspondiente saldrá la cantidad de pasos por encima del óptimo. (e.g. Si ha realizado 4 pasos más de los necesarios aparecerá un +4)

- La **tercera y última estrella** se obtiene siempre que el jugador no haya usado pistas para resolver el problema, tratando de motivar al usuario a buscar la solución sin ayuda. Se pretende incentivar a que el usuario esté atento a los tutoriales del juego en cada nivel y así no tenga que recurrir a pistas.

El **tiempo empleado** no tiene ninguna influencia en las estrellas asignadas ya que se busca que el jugador reflexione, piense y se esfuerce en buscar una solución, tratando de evitar comportamientos aleatorios que podrían motivarse con una restricción de tiempo.

Localización del juego

El videojuego se ha desarrollado en español pero está traducido y adaptado (lo que vulgarmente se denomina “localizado”) al inglés (en versión beta, no revisado ni probado por nativos). Esto puede ser útil si quiere ser utilizado en un entorno bilingüe o para fomentar el uso y el aprendizaje del inglés.

Rejugabilidad y configuración de los niveles

Al contar con niveles separados y categorizados en contenidos de enseñanza, el jugador tiene la posibilidad de rejugar y practicar los niveles

que haya desbloqueado tantas veces como quiera. Además, si no se han conseguido las 3 estrellas de cada nivel, el jugador puede volver a jugar para intentar conseguirlas y competir por tener el 100% de las estrellas del juego.

El hecho de contener niveles establecidos hace, que en un ámbito escolar, los alumnos puedan trabajar en equipo o compartir las soluciones a los niveles. Esto puede ser positivo ya que potencia la cooperación y compartición de ideas para resolver problemas si se juega al juego en grupos o en conjunto con el resto de la clase. Si no se desea esta situación, el profesor deberá tenerlo en cuenta en la clase para que este tipo de comportamientos no ocurran.

La principal fuente de rejugabilidad del juego es el modo de creación de niveles. Este modo ofrece al jugador la opción de crear tantos niveles como quiera. Además, la libertad que se le ofrece al jugador en cuanto a los elementos del tablero y las formas de solucionarlo significa una gran cantidad de contenido. Sin embargo, al ser contenido generado por el usuario, la principal limitación es la motivación y la creatividad del usuario para crear nuevos niveles. Es por este motivo que se recomienda utilizar el creador de manera que desafíe al usuario, por ejemplo, mediante el uso de pequeñas condiciones a las creaciones de los estudiantes, que les motiven a elaborar niveles y soluciones más sofisticadas.

Este modo de creación da una mejor idea de lo que realmente han aprendido los alumnos y de si dominan o no los conocimientos y conceptos de programación implicados.

Ejecución del juego y uso de códigos

Para poder acceder al juego, se requiere un código de acceso. Una vez introducido el código en la pantalla inicial del juego, hay que pulsar el botón “Empezar”.

Estos códigos sirven para identificar las distintas sesiones de juego y poder recoger información de forma anónima. Antes de las pruebas, os proporcionaremos una lista de códigos (en un fichero PDF), para que los asignéis a vuestros alumnos. Cada alumno utilizará su código único para acceder al juego. De esta forma, podremos obtener la información de interacción sin saber quienes son los alumnos concretos a los que pertenece cada código (pseudo-anonimización en origen).

Group: 04-3-2021:					
No.	Nombre	Código			
1		P987	P987	P987	P987
2		ZXBL	ZXBL	ZXBL	ZXBL
3		3XLF	3XLF	3XLF	3XLF

La información que se recoge se fundamenta en el comportamiento del usuario. Por ejemplo, se capturan las interacciones con los bloques de código, la forma en la que se colocan, los valores elegidos, variables, etc. En resumen, se hace un seguimiento de la sesión de juego del usuario. Por este motivo, se requiere conexión a internet y un código distinto para cada usuario.

La información recogida se usará para determinar si el juego cumple el objetivo de potenciar los aspectos del pensamiento computacional esperados y para mejorar la experiencia de usuario. En ningún momento se recoge información personal del usuario ni del equipo con el que se juega. A continuación, se muestra un extracto del tipo de información que se envía al servidor de la Universidad Complutense de Madrid.

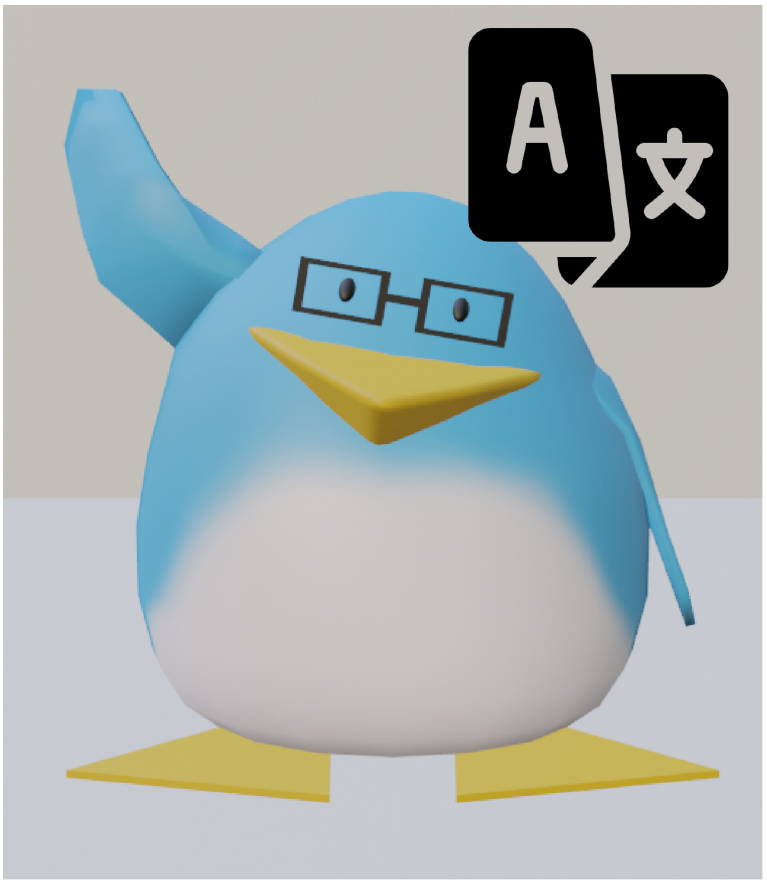
Anexo F

Trabajo de internacionalización y localización

Como parte de la asignatura Usabilidad y análisis de videojuegos, se planteó la localización del juego Articoding como proyecto final. Como parte de este proyecto final se realizó una memoria detallada del proceso de internacionalización y localización. Además, se incluyeron deficiencias y limitaciones encontradas en el paquete de localización utilizado.

Internacionalización y Localización del juego del TFG

Proyecto Final de UAJ



Grupo 09
Arturo García Cárdenas
Álvaro Poyatos Morate
Dany Faouaz Santillana

06/06/2021

Índice

1. Motivación y breve resumen del trabajo

1.1. Motivación

Nuestro TFG tiene como objetivo la enseñanza de la programación y el desarrollo de habilidades del pensamiento computacional. Hemos realizado pruebas formativas con expertos y profesores hispanohablantes (España y Latinoamérica) con la idea de que se pudiese usar en colegios como herramienta de aprendizaje.

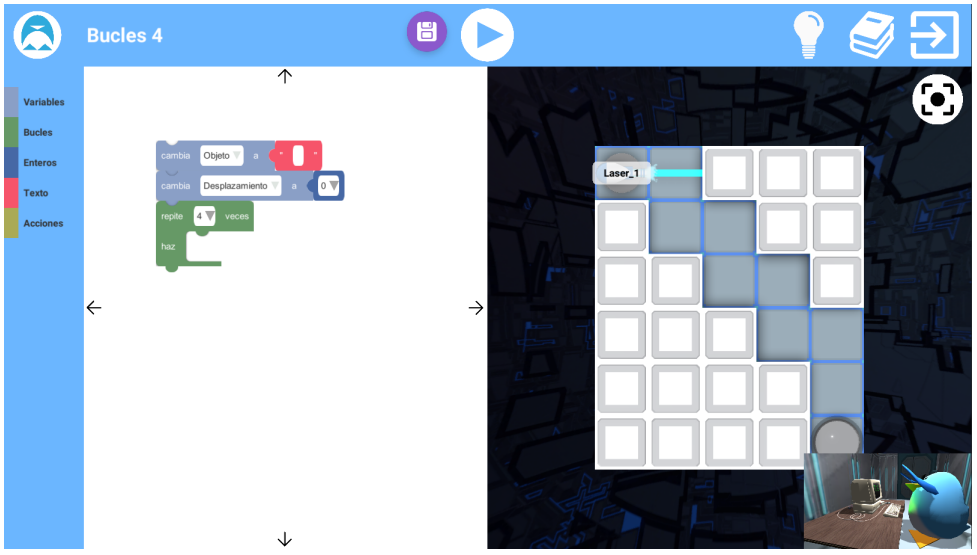
Pero hoy en día es sabido que el inglés es el idioma más usado en todo el mundo. Es por ello que, con el propósito de aumentar la visibilidad del proyecto y poder hacer pruebas en colegios de habla inglesa, se ha decidido internacionalizar y localizar el juego.

1.2. Resumen del trabajo

La idea consiste en implementar la **Internacionalización y Localización** del juego que estamos desarrollando para el TFG, Articoding. Se usarán dos idiomas de referencia:

- **Español (principal)**
- **Inglés (alternativo)**

Articoding es un **juego serio educativo** que pretende **fomentar el desarrollo del pensamiento computacional** (*Computational Thinking*) mediante **la enseñanza de conceptos de programación**.



Ejemplo de nivel de Articoding.

El jugador debe superar niveles en forma de tablero utilizando programación visual por bloques (similar a Scratch) para poder avanzar. El juego utiliza la librería externa *ublockly* para simular la programación por bloques de Scratch en Unity.

Ejemplo de código de *ublockly* en Articoding.

La aplicación **iniciará por primera vez teniendo en cuenta el idioma principal del sistema**, eligiendo adecuadamente el Locale del juego que corresponda. La próxima vez que se abra la aplicación se selecciona el idioma con el que se haya cerrado la aplicación. Aún así **se puede cambiar el idioma desde el menú de opciones**, en el menú principal (no se puede seleccionar un idioma si estás dentro de un nivel).

El **contenido total** que se espera Localizar e Internacionalizar es:

- Todos los textos que no se cambian por código.
- Todos los textos de las tarjetas de código (implica sincronización de Localización propia con un sistema externo de Localización)
- Textos flotantes de los objetos del tablero
- Todos los textos teóricos de los pop-ups (tutoriales)
- Todos los textos de las descripciones de las categorías de niveles
- Localización de Texturas de banderas (un pequeño ejemplo en el que se cambiará la imagen de una bandera dependiendo del idioma seleccionado)
- Localización de Texturas de los tutoriales

Se hará uso del **paquete de Unity de Localización** v0.10.0-preview[59].

[59] Quick Start Guide | Localization | 0.10.0-preview

2. Objetivos

Los **objetivos** que queremos conseguir con este proyecto son:

- **Aumentar la visibilidad** del juego, pudiendo hacer posibles pruebas en otros países aparte de España.
- **Hacer más accesible** el juego.
- **Mejorar la escalabilidad del proyecto**, permitiendo añadir otros idiomas si fuese necesario.
- **Analizar las deficiencias** encontradas en el proceso de desarrollo e idear posibles soluciones para resolverlas.

3. Detalles de implementación

Antes de empezar con los detalles de implementación, se van a mencionar los detalles del paquete más relevantes. El paquete ofrece las características más básicas de un sistema de localización, entre ellas están:

- **Configuración básica** del Sistema de Localización.
- Selector de idiomas (**LocaleSelector**)
- Tablas de **localización de textos**
- Tablas de **localización de assets** (Imágenes, fuentes, archivos de texto, etc)

Puesto que el paquete es de unity, las tablas no se guardan/generan en un formato de texto plano, sino que se generan archivos .asset propios de Unity. La gestión de tablas interna es simple. Se tiene una base de datos de tablas y cada vez que se crea una tabla se añade a la base de datos correspondiente.

Para implementar la funcionalidad de localización, el paquete ofrece una serie de componentes que facilitan el proceso, de los usados en este proyecto se encuentran los siguientes:

- StringLocalizeEvent
- SpriteLocalizeEvent

Estos son componentes que reciben una señal cuando deben actualizarse por un cambio de Locale y automáticamente modifican el texto o sprite, respectivamente, utilizando los valores de la tabla de localización que corresponda.

Además, el paquete ofrece una clase template **LocalizedAsset<T>** para localizar cualquier asset que se pueda serializar en el editor de Unity. También ofrece características como cadenas interpoladas, soporte de plurales, conexión con GoogleSheets o la posibilidad de importar y/o exportar tablas de textos en CSV o XLIFF.

3.1. Localización de textos fijos

Para localizar aquellos Text de UI que siempre van a contener el mismo texto, se ha añadido un componente **StringLocalizeEvent** a cada elemento y se han creado entradas en la tabla de localización con las traducciones en inglés y español. Todo esto se ha hecho desde el inspector y sin necesidad de cambiar el código.

3.2. Localización de texturas

Se han hecho uso de las Tablas de Localización de Assets para que las imágenes de las tarjetas de código que aparecen en la sección de Tutoriales se cambien cuando se seleccione otro idioma.

3.3. Localización de textos dinámicos

Sólo y exclusivamente para los elementos del juego que se modifican de manera dinámica, es necesaria una adaptación del código.

Si se localiza un texto, en vez de hacer referencia a un componente Text en un MonoBehaviour, se sustituye por una referencia al componente **StringLocalizeEvent** de la librería de localización. Para la modificación dinámica de textos, el mencionado MonoBehaviour debe de tener referencias de las distintas entradas de tablas (**LocalizedString**) que sea necesaria. Por ejemplo, en la localización de los tutoriales se han referenciado a todas las entradas de tablas que referencian los distintos tipos de tutoriales que existen. De esta manera, tanto en la creación de los botones con texto como en el título de los tutoriales se hacen uso de estas referencia a entradas de tablas.

3.4. Sincronización de la Localización de ublockly

Para lograr la sincronización del sistema de localización de la librería *ublockly* con la localización de la aplicación, se ha modificado el tipo de Asset que recibe el ScriptableObject encargado de la localización de los bloques de código (BlockResSettings). En vez de recibir un

array con los diferentes TextAssets con las traducciones de los bloques, se le pasa un **LocalizedAsset<TextAsset>** para que al cargar el TextAsset necesario lo haga acorde con la Locale seleccionada.

3.5. Sincronización de initial states de los niveles

Cada nivel del videojuego cuenta con unos bloques de código que se ofrecen como estado inicial del nivel. Frecuentemente en estos estados iniciales se muestran variables, cuyo nombre debe ser localizado.

Para conseguir esto, las estructuras de datos que contienen la información de cada nivel han sido modificadas para recibir una referencia a un **LocalizedAsset<TextAsset>** en lugar de un TextAsset. Así, cada nivel tendrá referencia a una entrada de una tabla de assets, con los diferentes archivos de estados iniciales localizados. Cuando se cargue un nivel, se cargará el TextAsset que corresponda según el Locale que esté seleccionado.

3.6. Selector de Locale

El paquete cuenta con un sistema de **LocaleSelectors** que se deben configurar en las preferencias del proyecto. Estos LocaleSelector determinan la forma de elegir el idioma al iniciar la aplicación. Se pueden elegir varios LocaleSelector y el sistema se encarga de quedarse con el primero capaz de darle la información del Locale a utilizar. Se han añadido 4 LocaleSelectors en este orden de prioridad:

- **PlayerPrefsLocaleSelector**: que se encarga de coger el idioma de los PlayerPrefs y guardarlo.
- **ComandLineLocaleSelector**: que se encarga de elegir idioma vía comandos de consola.
- **SystemLocaleSelector**: que se encarga de elegir un Locale en base al idioma del sistema.
- **DefaultLocaleSelector**: que elige un Locale determinado por el desarrollador. Este se ha elegido en caso de que no existiera el idioma del sistema, para que elija el inglés.

3.7. Entorno

El entorno de desarrollo usado para desarrollar el proyecto final es el motor de videojuegos Unity.

La herramienta usada es el paquete de Unity Localization ver. 10.0.0-preview.

Para algunas traducciones se ha utilizado la herramienta Deepl Translate.

3.8. Pruebas

Como el desarrollo de este proyecto se ha realizado sobre el proyecto del TFG y de forma simultánea, se ha tenido que mantener un control de versiones correcto. De esta forma, se ha ido mejorando el producto al mismo tiempo que se ha ido incorporando la localización e internacionalización del mismo.

Además, esta parte del desarrollo se ha incluido en la evaluación formativa con expertos y profesores. Esto ha hecho que tengamos más cuidado a la hora de enviar builds y que tengamos que comprobar que la localización esté actualizada y las traducciones sean correctas.

Cabe destacar que, al implementar las mejoras y las correcciones de errores, éstas se hacían primero en español y luego se adaptaban al inglés. Este proceso resultaba muy cómodo y rápido debido a la escalabilidad del proyecto.

3.9. Repositorio

Todo el código del proyecto se encuentra públicamente en el siguiente repositorio de GitHub:

https://github.com/WeArePawns/Articoding

4. Resultados obtenidos

Tras finalizar el desarrollo del proyecto, vamos a realizar un análisis de los objetivos alcanzados y de los resultados obtenidos.

4.1. Objetivos alcanzados

De los objetivos propuestos al inicio del proyecto, se han alcanzado los siguientes:

1. **Aumentar la visibilidad**. El juego podría distribuirse en casi todo el mundo, ya que dispone del idioma inglés. Pero al estar adaptado también al español, conseguimos abrirnos adicionalmente al público hispanohablante.
2. **Hacer más accesible el juego**. Con la inclusión del español, se permite el uso del juego a un público más joven en países hispanohablantes. Esto permite que se pueda utilizar el juego como herramienta de enseñanza de la programación con alumnos de colegios.
3. **Mejorar la escalabilidad del proyecto**. Gracias al buen uso del paquete de localización y del código escrito para la internacionalización, se ha conseguido un proyecto escalable donde es fácilmente localizable a otro idioma. Bastaría con añadir una columna a las tablas de localización e integrar los recursos pertinentes (textos traducidos, texturas, etc.).
4. **Analizar las deficiencias**. Aunque el paquete de localización es muy útil e intuitivo, hemos encontrado algunos defectos que se podrían mejorar:
    a. Carga de tablas asíncronas.
    b. No se permite la concatenación de cadenas localizadas.
    c. No se pueden generar texturas de forma automática, puesto que se usan tablas de localización donde se referencian a assets guardados en archivo
    d. Adaptación de la UI al resultado de localizar los textos.

    En el apartado de *Limitaciones encontradas* se habla más en detalle de estas deficiencias.

4.2. Limitaciones encontradas

Aunque se hayan alcanzado todos los objetivos planteados, durante el desarrollo e implementación, se han identificado ciertas limitaciones que supone utilizar el paquete de Unity Localization ver. 10.0.0-preview.
A continuación se listan las limitaciones del paquete que se han encontrado en el desarrollo y junto con una breve descripción de cómo se ha resuelto:

- **Las tablas se cargan de manera asíncrona** por lo que tarda en inicializar los textos. Para solucionar este problema, se ha implementado una pantalla de carga que espera a que todas las tablas de localización se carguen. Además, se ha aprovechado esta pantalla de carga para enmascarar los

tiempos de espera entre cambios de escena o de inicialización de niveles, aportando una mejor experiencia de usuario.

- **Los textos dinámicos formados por concatenación** no se pueden parametrizar. Aunque el paquete admite cadenas de textos con parámetros, si el parámetro es otra cadena localizada no funciona. Para solucionarlo se ha tenido que incorporar a la tabla de cadenas localizadas una entrada por cada una de las permutaciones posibles, eligiendo vía código la opción que tocase en cada momento. No es la solución más elegante ya que se podía haber utilizado dos textos en vez de uno. Puesto que el mismo texto se reutilizaba para varios string (titulos de seccion), utilizar esta ultima opcion hubiera supuesto un cambio mayor en el código. Además, se intentó utilizar la opción de *cadenas interpoladas* (**smart strings**) pero llevaba a una situación de llamadas recursivas infinitas.

- **No se pueden localizar texturas generadas dinámicamente**. Quizá una de las peores limitaciones encontradas del paquete de localización. Idealmente, se debería de poder preparar una textura de manera que se pudieran cambiar los textos que aparecieran en ella de manera dinámica. Una posible implementación sería la de utilizar las tablas de referencias y guardar además de la textura, información del texto para que dinámicamente se generen. Información como la fuente del texto, el tamaño, el contenido y la posición en la que empieza. Teniendo distinta información para los distintos Locales, se podrían reutilizar textura ahorrando en memoria y se ganaría en comodidad, escalabilidad, etc. La solución adoptada ha sido la única que nos permitía el paquete, generar manualmente texturas con distintos textos en los distintos idiomas y asignarlos uno a uno en la tabla de referencias.

- **Requerimiento de adaptación de algunos elementos de la UI**. Aunque inicialmente ya se partía de una interfaz mayoritariamente flexible y adaptable a textos con diferentes tamaños de fuente o longitud de cadena, se ha tenido que modificar ciertos elementos que al cambiar de idioma no se adaptan de manera correcta. Esto no es una limitación del paquete de localización en sí, sino una consecuencia de utilizarlo con texto de UI sin tener en cuenta tamaños.

Se debe tener en cuenta que el paquete es relativamente reciente y está en constante actualización. De hecho, se ha utilizado una versión no actualizada por temas de compatibilidad pero existen versiones más recientes con mejoras de usabilidad. Hasta el día de hoy, ninguna versión tiene en cuenta ninguna de las limitaciones descritas anteriormente.

4.3. Pipeline de trabajo

Para poder añadir nuevo contenido y localizarlo, se deben seguir los siguientes procedimientos:

- **Textos**. Se deben exportar las tablas en CSV desde Unity, modificar la tabla en una herramienta externa e importar el CSV actualizado a Unity.
- **Texturas**. Se deben asignar las nuevas texturas mediante la ventana de tabla de localización de assets en Unity a las entradas correspondientes. El traductor recibiría un archivo con los textos a traducir y posteriormente el artista gráfico introduciría los textos en las texturas correspondientes.
- **Tarjetas de código (ublockly)**. Es necesario agregar nuevas entradas en el archivo de texto del idioma correspondiente o crear un archivo nuevo si se añade un nuevo idioma.

5. Conclusiones

Para concluir, todos los objetivos planteados inicialmente han sido cumplidos con creces. Es cierto que se ha encontrado dificultades durante el desarrollo pero ninguna que no se haya podido solucionar.

Para la magnitud y contenido del juego que se ha internacionalizado y localizado, la elección del paquete ha sido acertada. Esto nos ha proporcionado justo lo que requerimos para localizar de manera cómoda el juego ya que el juego no contaba con cinemáticas, fuentes de texto, modelos o aspectos culturales que localizar. Por otro lado, si el juego hubiera sido más sofisticado y hubiera incluido elementos como los mencionados anteriormente, el paquete elegido se hubiera quedado corto a la hora de ofrecer comodidad y flexibilidad en la usabilidad.

Además, el uso de este paquete ha hecho que se tenga que modificar el código de manera interna, creando una dependencia directa con el sistema de localización, lo cual puede suponer problemas en un proyecto de gran magnitud.

6. Adenda

El reparto y gestión de las tareas se ha hecho mediante el uso de Pivotal Tracker. Se ha mantenido una comunicación constante mediante Discord, realizando reuniones diarias para tomar decisiones sobre las tareas (también para hablar sobre el TFG). Para gestionar los cambios se ha usado un repositorio de GitHub (mencionado anteriormente).

A continuación se detallan las contribuciones individuales de cada miembro del equipo:

6.1. Aportaciones de Arturo García Cárdenas

- Localización de todos los textos fijos.
- Localización de los textos dinámicos de los tutoriales.
- Traducción de todos los textos de los tutoriales (título y descripción).
- Tomadas todas las imágenes de las tarjetas de código para la localización de texturas.

6.2. Aportaciones de Dany Faouaz Santillana

- Localización de textos de ScriptableObjects (Category, Level, etc.).
- Pantalla de carga (espera de carga de las tablas de localización).
- Localización de las texturas de tarjetas de código en los tutoriales.
- Localización de la información de los BoardObjects (elementos del tablero, nombre identificador y argumentos).

6.3. Aportaciones de Álvaro Poyatos Morate

- Sincronización de la localización de ublockly con el sistema de localización.
- Traducción de los bloques de ublockly.
- Localización de assets de initial states.
- Localización de textos fijos de ublockly.