Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en Ingeniería Informática

Trabajo Fin de Grado

Icarus: Desarrollo de un Producto Software Interactivo Basado en Mecánicas de Point-and-Click y Aventura Gráfica

Autor: Víctor Valdeolmos Rabadán

Tutor(a): José María Barambones Ramírez

Madrid, junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Icarus: Desarrollo de un Producto Software Interactivo Basado en Mecánicas de Point-and-Click y Aventura Gráfica
Junio 2024

Autor: Víctor Valdeolmos Rabadán

Tutor:
José María Barambones Ramírez
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Este documento recoge el diseño y desarrollo de un producto software interactivo basado en mecánicas de point-and-click y aventura gráfica utilizando el motor de videojuegos Unity y el lenguaje C#. Dicho producto consiste en la resolución de un caso policiaco. Para ello, los jugadores deberán examinar el escenario e interactuar con los objetos que encuentren a su paso. Tras ello, conseguirán unas pistas y podrán unirlas para obtener diferentes deducciones hasta llegar a la resolución del misterio.

Este videojuego será utilizado para medir ciertos patrones motivacionales de los usuarios basado en el *Motivational Trait Questionnaire* (MTQ) y clasificar el tipo de jugador según la Taxonomía de Bartle. Para este propósito, se utilizarán dos juegos más desarrollados por alumnos de la ETSIINF donde cada juego examinará ciertas acciones del usuario y al unir todas las métricas evaluadas se definirá su perfil motivacional y de tipo de jugador.

El producto se ha implementado en una versión jugable conocida como *"vertical slice"* durante cuatro iteraciones de trabajo. A lo largo de este proyecto, se ha realizado un *Game Design Document* (GDD), detallado en el capítulo 3, que explicará las decisiones que se han tomado en el diseño del juego. Mientras tanto, en el capítulo 4 se hablará sobre la fase de desarrollo, donde se explica cómo se ha implementado el producto.

A continuación, tras haber completado la implementación del juego, se comenzó con la fase de evaluación donde se evalúo la experiencia de usuario y se comparó el perfil motivacional y de jugador obtenido por las métricas de nuestros juegos con los de los tests MTQ y Bartle Test. Los resultados obtenidos no fueron del todo satisfactorios ya que los perfiles medidos por nuestros juegos distaban de los generados por los tests. Pero se extrajo información valiosa de cara a la mejora de este sistema. Asimismo, se realizó una auditoría de accesibilidad para saber dónde mejorar en este aspecto.

Tras ello, se analizaron los resultados y se extrajeron conclusiones para orientar el desarrollo futuro del videojuego. Aquí se detallan las más importantes.

- La explicación de la mecánica de unión de pistas no fue lo suficientemente aclaratorio y varios jugadores manifestaron su confusión ya que no sabían qué hacer con las pistas obtenidas.
- Se observó que los jugadores no reaccionaban ante las notificaciones que había en pantalla, lo que dificultó el entendimiento del juego en algunas fases.
- Había un *bug* importante con el sistema de *outline* que no se mostraba correctamente, lo que complicó la experiencia de usuario.

Se discutieron aspectos que podían ser implementados en futuras líneas de desarrollo tras arreglar los contratiempos detectados durante la evaluación. Entre estos aspectos destaca la mejora de las métricas medidas para la obtención de los perfiles de usuario, la implementación de nuevas mecánicas como un inventario general, un sistema de misiones o estadísticas RPG y también se planteó expandir el prototipo con nuevos escenarios y casos por resolver.

Abstract

This document covers the design and development of a point-and-click and graphic adventure video game product using Unity as game engine and C# language. This product involves solving a police case. For this purpose, players will examine the scene and interact with the objects that they find along the way. From there, they will gather clues and they will combine them to make some deductions until they solve the mystery.

This video game will be used to measure certain motivational traits of users based on the *Motivational Trait Questionnaire* (MTQ) and classify the type of player according to Bartle's Taxonomy. To do this, two more games developed by students from the ETSIINF will be used and each game will inspect some user actions and combining all the evaluated metrics, the user's motivational and player type profile will be defined.

The product has been implemented in a playable version known as *"vertical slice"* over four iterations of work. Throughout this project, a *Game Design Document* (GDD) has been developed, detailed in Chapter 3, which explains the decisions made during the game design process. Meanwhile, Chapter 4 discusses the development phase, which will explain how the product has been implemented.

Moreover, after completing the implementation of the game, the evaluation phase started, where the user experience was assessed, and the motivational and player profiles obtained by our games were compared with the results of the MTQ and Bartle Test. The results were not entirely satisfactory as the profiles measured by our games differed from those generated by the test. Nevertheless, valuable information was obtained in order to improve the system in the future. Additionally, an accessibility audit was conducted to identify aspects for improvement in this area.

Following this, the results were analyzed, and conclusions were drawn to guide future phases of development of the video game. Here are the most important findings:

- The explanation of the clue mechanism was not clear enough, and several players expressed their confusion as they did not know what to do with them.
- It was observed that players did not respond to on-screen notifications, which did not help in the understanding of the game in some phases.
- There was a significant *bug* with the *outline* system, which did not display correctly, making more difficult the user experience.

There were some discussions of the following aspects that could be implemented in future development after fixing the issues detected during the evaluation. These aspects include improving the metrics used to obtain much better results on user profiles, implementing new mechanics such as a general inventory, a mission system, or some RPG features, and expanding the prototype with new scenarios and cases to solve.

Agradecimientos:

Primero, querría agradecer a mi tutor ya que este trabajo no hubiera sido posible sin su dedicación y esfuerzo. Has sido una fantástica guía y no podría estar más satisfecho de haber hecho este proyecto contigo.

A mi hermano Jesús, que siempre ha estado presente en los buenos y malos momentos, apoyándome siempre que ha sido necesario. Sin ti no hubiese conseguido llegar hasta donde he llegado.

A mis padres que hicieron que me embarcase en el viaje de la Ingeniería Informática y gracias a ellos tomé la mejor decisión posible. Siempre habéis sido una gran fuente de consejos y vuestro apoyo y respaldo ha sido esencial.

A mis amigos de la universidad que me habéis acompañado en este viaje y que me habéis dado las fuerzas y ganas para poder continuar. Sois lo mejor que me llevó de esta etapa.

Agradecer a mis amigos del instituto, en especial a mi amiga Elena, por su apoyo siempre que lo necesito y estar siempre ahí.

Por último, dar las gracias a todos los alumnos de la ETSIINF que aportaron su tiempo para realizar la evaluación de usuario del proyecto y a mis compañeros que participaron en el estudio de los perfiles motivacionales de los usuarios.

Tabla de contenidos

1 Introducción

1.1 Motivación

A lo largo de la historia reciente se han diseñado muchas pruebas y teorías para poder medir las motivaciones de una persona, su personalidad o diferentes aspectos psicológicos. Sin embargo, cuando un usuario está realizando uno de estos tests, puede mentir o no ser del todo preciso en sus respuestas ya que sabe que está siendo evaluado. Incluso, en muchas ocasiones, la percepción de uno mismo puede no ser la correcta. Por lo que tal vez puede ser más efectivo realizar estas mediciones en otros contextos donde el usuario se sienta más cómodo y menos observado. Para este propósito, los videojuegos pueden ser una buena herramienta al tener los jugadores la percepción de que lo que está haciendo no tiene incidencia en el mundo real.

Por lo tanto, los videojuegos pueden ser utilizados para un propósito serio, más allá del lúdico. Actualmente, los enfoques basados en juegos para la salud mental están en sus inicios. Además, las primeras investigaciones sugieren de los beneficios potenciales para conseguir posibles cambios psicológicos y de conducta, además de que pueden ser una buena manera de detectar ciertos patrones psicológicos y motivacionales. [1] [2]

El objetivo fundamental de este proyecto será la consecución de un perfil motivacional y el tipo de jugador, gracias a la colaboración con dos proyectos más, donde cada uno conseguirá examinar determinadas métricas que sean más evaluables en su videojuego. Cuando una persona juegue a los 3 juegos se podrá obtener el perfil motivacional y clasificar el tipo de jugador de dicho usuario evaluado. Con ello comprobar si 3 prototipos que no son diseñados específicamente para este propósito son capaces de obtener resultados concluyentes y establecer un buen punto de partida.

Para la obtención de esta meta se desarrollará un producto software interactivo siguiendo las pautas y metodologías ligados al desarrollo de software.

1.2 Objetivos

La meta del proyecto consiste en diseñar y desarrollar un producto software interactivo que sea capaz de examinar los rasgos motivacionales de un usuario utilizando de referencia el *Motivational Trait Questionnaire* (MTQ) [3] y el Bartle Test [4] para poder clasificar al jugador. Para ello se aplicarán los métodos y procesos de Ingeniería de Software, basado en diseño de videojuegos y sistemas interactivos. Para poder alcanzar estos propósitos se plantean estos objetivos más específicos.

- Elaborar un *Game Development Document* (GDD) donde se recojan las directrices y decisiones de diseño en la producción del producto.
- Desarrollar una versión jugable del producto en estado Alfa o *Vertical Slice* siguiendo lo recogido en el GDD.
- Definir un perfil motivacional y clasificar a los jugadores en base a sus acciones medidas en los prototipos jugables.

1.3 Planificación

Para la planificación del proyecto se realizó un diagrama de Gantt, que se muestra en la figura 1.2. En este diagrama se muestran las tareas a realizar durante el proyecto y la duración de las mismas, indicando la semana de inicio y la final. Cada 2 semanas, se realizaron reuniones junto al tutor para poder efectuar distintos ajustes si hubiese sido necesario y revisiones sobre el trabajo elaborado.

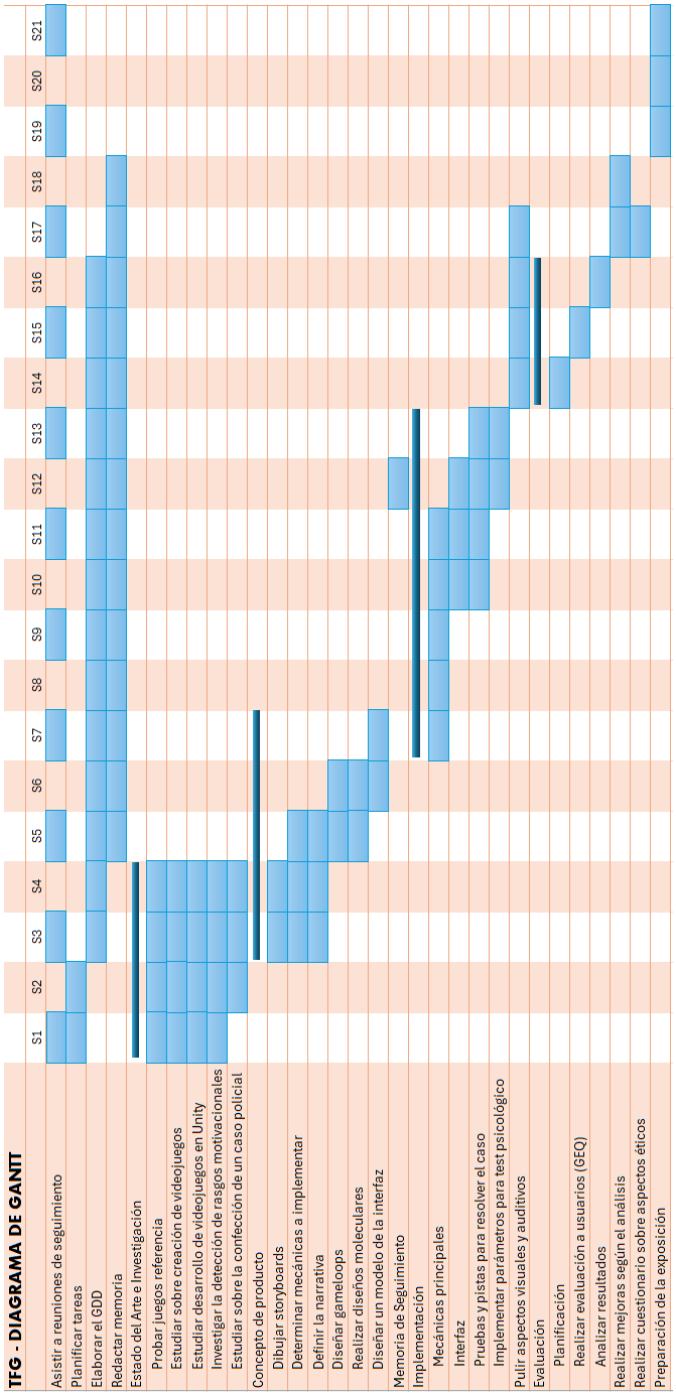

Figura 1.1 Diagrama de Gantt

1.4 Estructura del documento

Los contenidos que conformarán el contenido de este documento son los siguientes:

2 - Estado del Arte. Investigación y documentación de trabajos previos acerca del desarrollo de videojuegos, el uso de estos productos para detectar rasgos motivacionales y de personalidad y las herramientas disponibles para realizar un videojuego.

3 - Diseño. Resumen sobre las decisiones tomadas para el diseño del producto que se recogen en el GDD.

4 - Desarrollo. Descripción sobre la implementación de la *Vertical Slice.*

5 - Evaluación y resultados. Evaluación de la experiencia de usuario y la obtención de los perfiles motivacionales y la personalidad de los usuarios.

6 - Discusiones y líneas futuras. Planteamiento de líneas futuras de trabajo y análisis de los resultados.

7 - Conclusión. Reflexiones finales tras la consecución del proyecto.

2 Estado del Arte

2.1 Ingeniería de software aplicada al diseño y desarrollo videojuegos

Los videojuegos son aplicaciones software que están instalados en dispositivos hardware como pueden ser un ordenador o una consola. Para poder realizar estos productos es necesario un equipo multidisciplinar que necesita conocimientos en sonido, arte, programación o inteligencia artificial, entre otros. Esto produce que los proyectos de desarrollo de estos productos software sean altamente complejos. Esta complejidad conlleva muchos retos y problemas en el proceso de diseño y desarrollo ya que envuelve actividades en la disciplina artística (storyboards, animaciones, sonido, etc.) además de cumplir con unos requisitos tecnológicos y funcionales del producto. [5]

Para el desarrollo y diseño de videojuegos se siguen metodologías similares a las utilizadas en un producto software. Los principales enfoques utilizados son las metodologías tradicionales y las metodologías ágiles. [5] [6]

2.1.1 Metodologías tradicionales

Las metodologías tradicionales de desarrollo software se distinguen por fijar al inicio de los proyectos los requerimientos y procesos a seguir de una forma muy rígida. Esto conlleva una baja flexibilidad a la hora de realizar ajustes a lo largo de la vida del proyecto. [7]

Cuando los videojuegos comenzaron a ser más complejos por la mejora de los hardware donde se ejecutaban, se empezó a integrar el modelo en cascada, comúnmente utilizada en proyectos software, para el desarrollo y diseño de los videojuegos. Esta metodología emplea la idea de desarrollar un proyecto software a través de una serie de fases que se ejecutan de forma secuencial. Cada fase está diseñada para reducir el riesgo antes de moverse a las siguientes etapas del desarrollo. Con este método, la gran mayoría del diseño está realizado pronto y luego la gran mayoría del testing está realizado más tarde. Esto puede provocar que alguna característica del desarrollo haya que rediseñarla si se encuentran errores durante el testeo. [6]

2.1.2 Metodologías ágiles

Las metodologías ágiles en el desarrollo de software se están haciendo cada vez más populares, especialmente en la industria de los videojuegos, gracias a su gran flexibilidad y capacidad para ajustarse rápidamente a posibles cambios. Estas metodologías posibilitan que los equipos sean más eficientes y productivos al tener una comprensión clara de las tareas que tienen que abordar en cada etapa. [7] [6]

A partir de los años 80, se comenzó a intentar buscar alternativas a las diversas metodologías tradicionales y el modelo de cascada. Para ello, se diseñaron algunos modelos iterativos que consistían en hacer pequeñas iteraciones de trabajo donde se trabajaba sobre todas las fases de desarrollo, que incluían,

análisis, diseño, scripting, integración y testeo. En 2001, se decidió referirse a ellos como metodologías ágiles tras la publicación del 'Manifiesto Ágil' donde se constataron los valores y principios de estos nuevos métodos. Esto ha llevado a que varias de estas metodologías ágiles como Scrum o XP, que son los más utilizados en el desarrollo de videojuegos, comparten una filosofía común. [6]

2.2 Estudios psicológicos y motivacionales utilizando videojuegos

Sherry propuso en 2004 que jugar a un videojuego produce un estado de flujo psicológico, que representa el atractivo principal de esta actividad. Este flujo se caracteriza por: una intensa concentración en la tarea, una fusión de acción y conciencia, poca o ninguna autoconciencia, una percepción de que las habilidades cumplen con todos los desafíos, una sensación de que el tiempo pasa más rápido y la participación en la actividad es en sí misma su propia recompensa. Este flujo que ocurre al disfrutar de uno de estos productos ha hecho que cada vez se utilicen los videojuegos en distintas áreas como la educación. [8]

El crecimiento de la popularidad de los videojuegos ha producido que investigadores de diversas disciplinas como la psicología, hayan empezado a examinar posibles correlaciones psicológicas, sociales y situacionales del uso y experiencia de videojuegos [9] [10]. Por ejemplo, varios científicos de la universidad de Auckland, en unión con otros de diferentes zonas del mundo, han explorado el concepto del uso de juegos serios para poder detectar y evaluar una discapacidad psicológica [1] [10]. Con ello consiguieron mejorar la eficacia de las tecnologías ya existentes para este propósito.

2.2.1 Motivational Trait Questionnaire

Para analizar la precisión del perfil motivacional conseguido por nuestros juegos se ha seleccionado como referencia la prueba *Motivational Trait Questionnaire* (MTQ) creado por Kanfer y Heggestad. Este cuestionario evalúa las características motivacionales de las personas mediante varias preguntas que exploran la disposición de una persona para enfrentar desafíos, su consistencia en alcanzar metas y su confianza en su habilidad para triunfar [3] [2].

Kanfer y Heggestad se basaron en su propia estructura llamada *Motivational Traits and Skills* (MTS) para realizar el MTQ. En este marco se definen los conceptos de rasgos y habilidades motivacionales.

- Los rasgos motivacionales se refieren a las diferencias estables que tiene cada individuo en sus preferencias, relacionadas con la aproximación o evitación de situaciones, así como en la manera en la que invierten esfuerzo para alcanzar sus metas.
- Las habilidades motivacionales se definen como competencias de autorregulación integradas que se aplican durante el esfuerzo por alcanzar una meta específica.

2.2.1.1 Rasgos Motivacionales

Los rasgos motivacionales se dividen en dos grupos llamados Logro y Ansiedad. Mientras que los rasgos del grupo Ansiedad se caracterizan por tendencias de evitación, el grupo de Logro incluye rasgos que se caracterizan por tendencias de acercamiento [3] [2].

En el conjunto de Logro tenemos los siguientes rasgos motivacionales:

- Maestría Personal (PM): Una persona con un valor alto en este rasgo establece estándares de excelencia en términos de mejora personal y persiste en esforzarse por alcanzarlos a pesar de las frustraciones y dificultades que se pueda encontrar. Estas personas muestran, generalmente, preferencias por tareas que desafíen sus habilidades y capacidades. Son individuos competitivos consigo mismos y siempre buscan ser lo mejor que se pueda ser.
- Excelencia Competitiva (CE): Una persona con un nivel alto en este rasgo adopta estándares normativos de excelencia. Estas personas definen el éxito en relación con los demás y no valoran tanto la calidad absoluta de su rendimiento, ya que lo importante es que su rendimiento mejore al de los demás. Son muy competitivos y a menudo intentan crear competencia en situaciones que no tienen por qué serlo. Además, estas personas tienen un fuerte deseo de ser respetadas por sus logros conseguidos.
- Trabajo Duro (HW): Una persona con un valor alto en este rasgo se espera que ejerza grandes cantidades de esfuerzo al completar una tarea, independientemente de su nivel de disfrute intrínseco en la realización de la tarea. Estos individuos son trabajadores y diligentes. Además, tienen un fuerte deseo de mantenerse ocupadas y les resulta complejo simplemente relajarse y no hacer nada.

Mientras tanto, en el complejo de Ansiedad se encuentran los siguientes rasgos motivacionales:

- Evitación del Fracaso (FA): Se espera que una persona con un alto nivel en este rasgo evite activamente las situaciones orientadas al logro siempre que sea posible debido a la ansiedad causada por la posibilidad de experimentar el fracaso.
- Ansiedad por el Logro (AA): Una persona con un valor alto en este rasgo se espera que refleje una tendencia a experimentar respuestas de ansiedad en situaciones de logro, como cumplir un plazo en un trabajo o participar en una competencia deportiva.

2.2.2 Bartle Test

La taxonomía de Bartle es una clasificación de los jugadores de videojuegos basado en un artículo realizado por Richard Bartle [4] [11]. Este escritor basó su estudio tras analizar los videojuegos de tipo *Multi-User Dungeon* (MUD) [12]. El videojuego más popular adscrito a este género es *Dungeons &* Dragons. Tras este escrito, se diseñó una prueba llamada "Bartle Test" que diese porcentajes a cada jugador en base a su forma de jugador. La teoría desarrollada por el inglés constituye cuatro tipos diferentes de jugador:

- **Triunfador:** los triunfadores son usuarios que buscan mejorar su nivel y alcanzar los máximos puntos posibles. La exploración solo la utilizarán

para conseguir nuevo equipamiento o tesoros que ayuden al anterior propósito. Socializar les servirá para descubrir nuevos métodos que otros jugadores puedan saber para conseguir masterizar el juego. También matarán a otros jugadores si esto les hace escalar en algún tipo de clasificación.

- **Explorador:** el disfrute de los exploradores reside en que el juego exponga sus costuras internas a ellos. Para este propósito, intentan realizar acciones poco convencionales en entornos salvajes, buscar características interesante como *bugs* o simplemente conocer como funciona la lógica interna del juego. Para ellos la alegría real solo viene del descubrimiento de algo nuevo y de explorar todo el entorno posible que haya en el juego.
- **Asesino:** este tipo de jugador disfruta al imponerse sobre otros. Cuanto mayor sea la angustia causada, mayor será la diversión de estos usuarios. Estos jugadores pueden hacer uso algunas veces de la exploración o la socialización, pero siempre con el objetivo en mente de encontrar vulnerabilidades. Para ellos, el fin justifica los medios.
- **Socializador:** están interesados en las personas y en lo que estos tienen que decir. El juego para ellos es un medio para interactuar con jugadores o personajes humanos. Para ellos podrá ser satisfactorio observar cómo los jugadores a su alrededor van creciendo como individuos. Luchar contra otros jugadores solo estará excusado en el caso de que sea un acto de venganza contra un usuario que haya hecho daño a un amigo. En general, son jugadores que disfrutan conociendo a personas, entendiéndolas y generando vínculos duraderos con ellos.

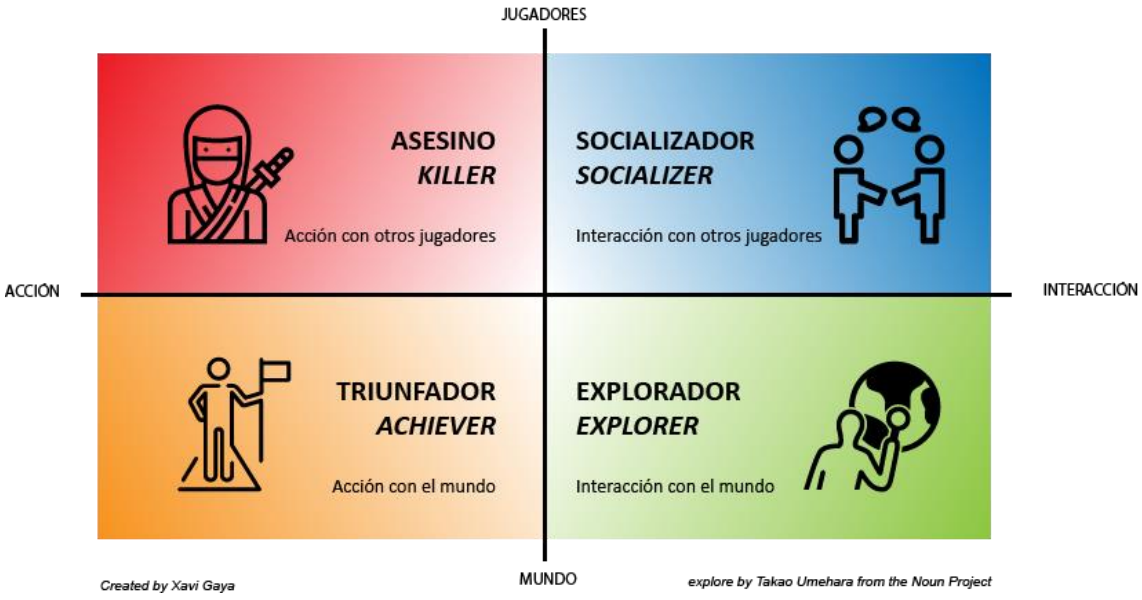

Figura 2.1: Tipos de jugadores propuestos por Richard Bartle en su taxonomía [13]

2.3 Herramientas

2.3.1 Motores de videojuegos

La primera vez que se acuñó el término 'motor de videojuegos' en la historia fue por parte de la compañía ID Sofware en 1993 tras el lanzamiento de Doom. La empresa explicó este concepto como la tecnología "detrás del juego" [14]. La creación de los motores creó un nuevo paradigma para el desarrollo de videojuegos, permitiendo el uso y reúso de un código base para crear nuevos productos que compartiesen un núcleo común [15].

Un motor de videojuegos es un entorno de desarrollo software con configuraciones y ajustes que optimizan y simplifican el desarrollo de los videojuegos a lo largo de una gran variedad de lenguajes de programación. Estos motores pueden incluir renderizado de gráficos para 2D y 3D, además de un motor de físicas que es capaz de realizar simulaciones de ciertos sistemas físicos, como puede ser el movimiento de un fluido. Además, los motores también pueden incluir inteligencia artificial que responda a las acciones del jugador, un sistema de sonido para controlar los efectos sonoros y un motor de animaciones [16] [17].

2.3.1.1 Unity

Unity es uno de los motores de videojuegos más utilizados en la actualidad dentro de la industria, con casi la mitad de los juegos del mercado desarrollados en este motor en el 2018. Este entorno se caracteriza por su compatibilidad, versatilidad y su facilidad de uso, haciéndolo ideal para desarrolladores novatos y proyectos pequeños y medianos [15] [18] [19].

Este motor incluye una variedad de componentes avanzados como el motor físico de Nvidia, PhysX, un sistema de animación llamado Mecanim, un editor de terrenos o MLAPI para el desarrollo de productos multijugador en red. Para escribir scripts en esta plataforma será necesario utilizar C#, Javascript o Boo. [15] [18] [19].

Unity tiene una comunidad de usuarios enorme y bastante activa que hace que el aprendizaje de esta plataforma sea más sencillo y rápido debido a la alta disponibilidad que hay de tutoriales y guías en la red. [15] [18] [19].

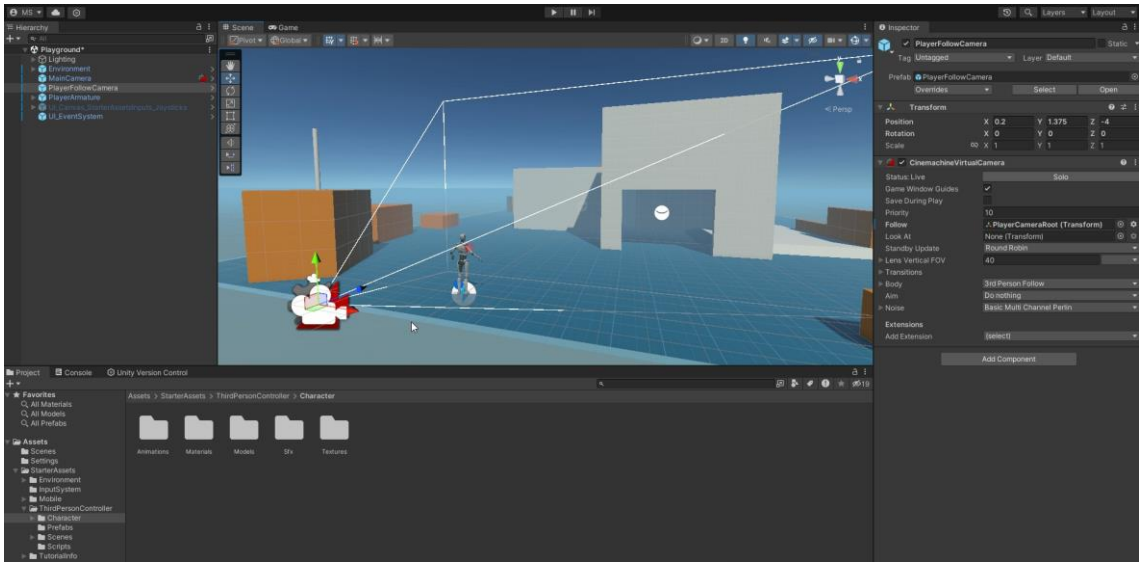

Figura 2.2: Editor de Unity [20]

2.3.1.2 Unreal Engine

Unreal Engine fue desarrollado por Epic Games y fue lanzado al mercado en el año 1998. Este motor fue creado con C++ y ofrece un alto grado de portabilidad, soportando una amplia gama de plataformas. Unreal Engine también ha sido utilizado en la producción de cine y series, como en el caso de la serie de televisión 'The Mandalorian'. Este entorno ofrece una posibilidad para obtener una mayor potencia en gráficos comparando con sus competidores, sin embargo, tiene una mayor curva de aprendizaje y su interfaz es menos intuitiva [15] [18] [19].

Los principales beneficios de utilizar Unreal Engine son el tener un código reutilizable utilizando librerías, el concepto de programación orientada a objetos del que hace uso y la tecnología para realizar modelos humanos [15]. La versión 4 de Unreal Engine lanzada en el año 2014, introdujo un sistema llamado Blueprints, que consiste en un lenguaje de programación visual que permite a los desarrolladores diseñar elementos del juego y mecánicas haciendo uso de una interfaz basada en nodos [15] [18] [19].

En la última versión lanzada en el año 2022, introdujeron 3 sistemas revolucionarios al motor llamados Nanite, Lumen y 'World Partition System'. El primero consiste en un sistema virtualizado de geometrías que acelera y mejora la creación del nivel de detalle y renderización de objetos. Lumen es una nueva utilidad para la iluminación de las escenas, que la hace más realista. Por último, el 'World Partition System' es un sistema que mejora la forma en la que Unreal Engine se comporta al crear mundos abiertos grandes [18].

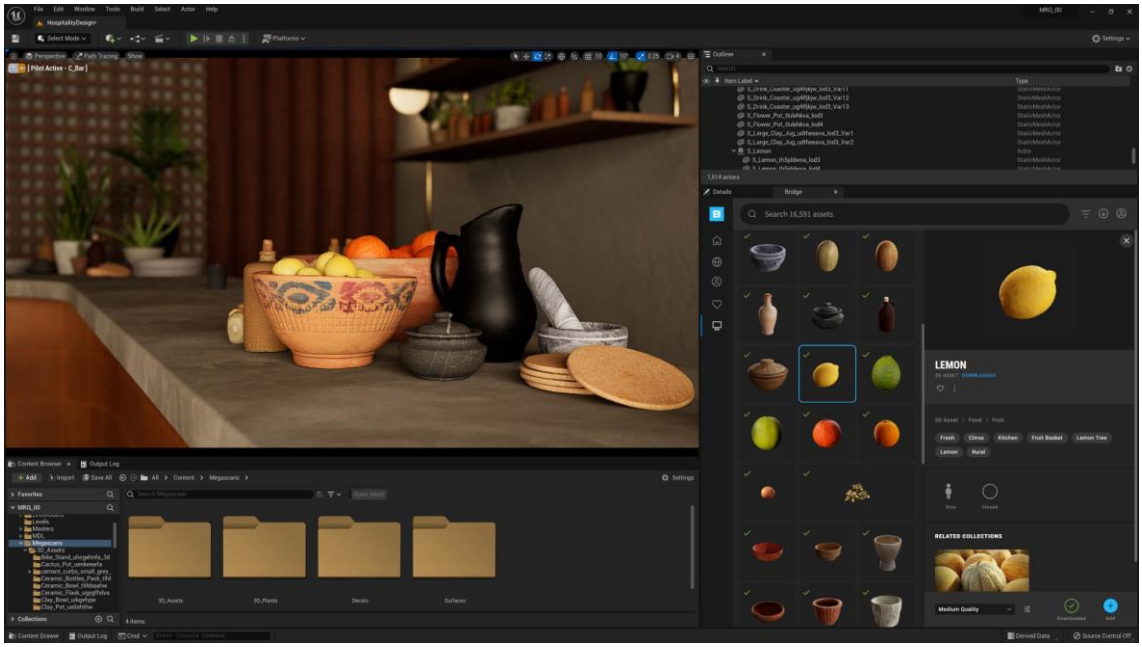

Figura 2.3: Editor de Unreal Engine 5 [21]

2.3.1.3 Gamemaker

Gamemaker fue lanzado al mercado en 1991 bajo el nombre de Animo y es un motor pensado para desarrollar juegos con gráficos 2D. En recientes versiones se ha permitido utilizar gráficos 3D, aunque su uso es bastante limitado en este caso. [22]

Gamemaker se caracteriza por utilizar un lenguaje de programación visual de arrastrar y soltar para el desarrollo de videojuegos en su motor. También, se puede utilizar un lenguaje llamado Game Maker Language para la programación. Este lenguaje estuvo diseñado para programadores novatos y así poder crear juegos sin tener un gran conocimiento en programación. Por lo tanto, Gamemaker es un buen motor para desarrolladores que quieran aprender, por su facilidad de uso, pero no es un motor que destaque en otros aspectos respecto a su competencia. [22]

Figura 2.4: Editor de GameMaker [23]

2.3.1.4 Cryengine

Cryengine fue creado por Crytek y lanzó su primera versión en 2002 aunque esta versión solo fue un demo tecnológica creada para Nvidia. En 2004, presentaron ya el motor evolucionado y también fue lanzado Far Cry que es el juego más característico que ha tenido este motor. [19] [24]

Para programar en Cryengine es necesario utilizar uno de los dos lenguajes de programación que soporta el motor que son C++ o C#. Cryengine permite el desarrollo de videojuegos para múltiples plataformas. [19]

Este motor tiene una gran curva de aprendizaje ya que tiene una interfaz menos amigable y hay menos información sobre este motor en la red ya que la comunidad es más pequeña. Aun así, se trata de un motor de videojuegos muy potente que puede producir gráficos muy realistas y un gran rendimiento. Además, Cryengine tiene un sistema gráfico llamado Flow Graph que permite controlar la lógica del juego desde el nivel de diagramas gráficos. [19]

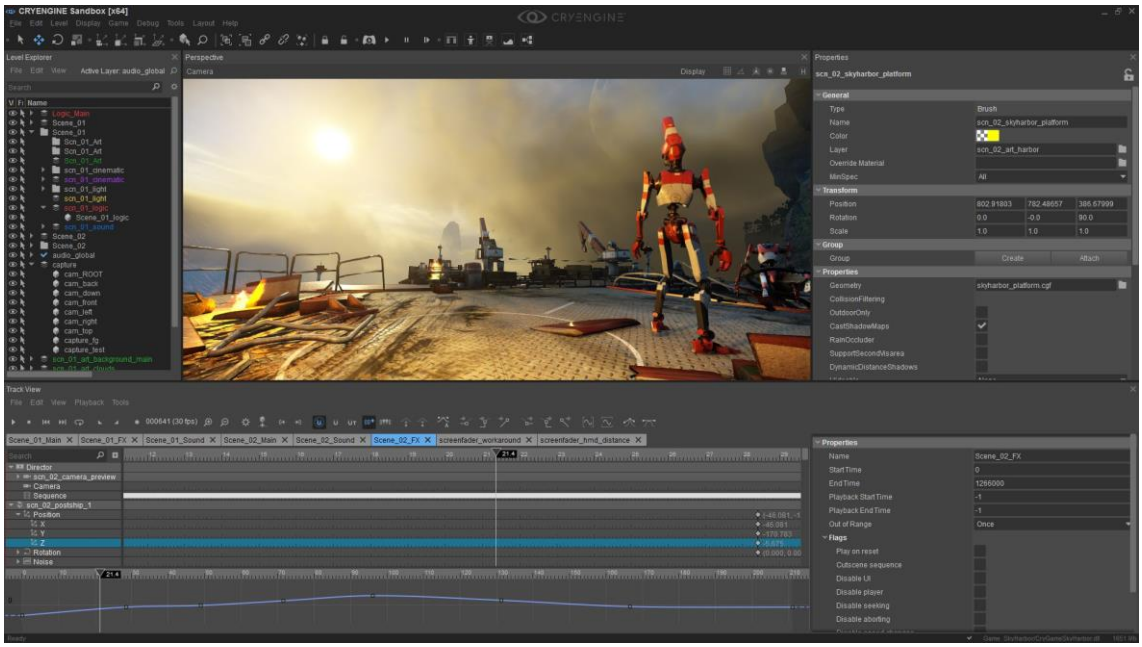

Figura 2.5: Editor de CryEngine [25]

2.3.1.5 Godot

Godot es un motor de videojuegos de código abierto en el que se puede desarrollar juegos para múltiples plataformas y para gráficos en 2D y 3D. Godot ha sido desarrollado para poder utilizar varios lenguajes de programación. Aunque tiene un lenguaje propio llamado GDScript, también se puede utilizar C#, C++ y Visual Script. [26]

Este entorno utiliza un motor de físicas de alta fidelidad llamado Bullet y también tiene uno de baja fidelidad llamado GodotPhysics. Además, Godot proporciona buenas herramientas para la construcción de videojuegos multijugador. [26]

Godot está diseñado para ser eficiente y ligero, de esta manera se minimiza el uso de recursos comparado con otros motores. Asimismo, este motor permite la modificación del código fuente ya que es accesible para los desarrolladores y así satisfacer sus necesidades. Esta combinación hace que Godot sea particularmente adecuado para desarrolladores que trabajen en proyectos con recursos limitados. [26]

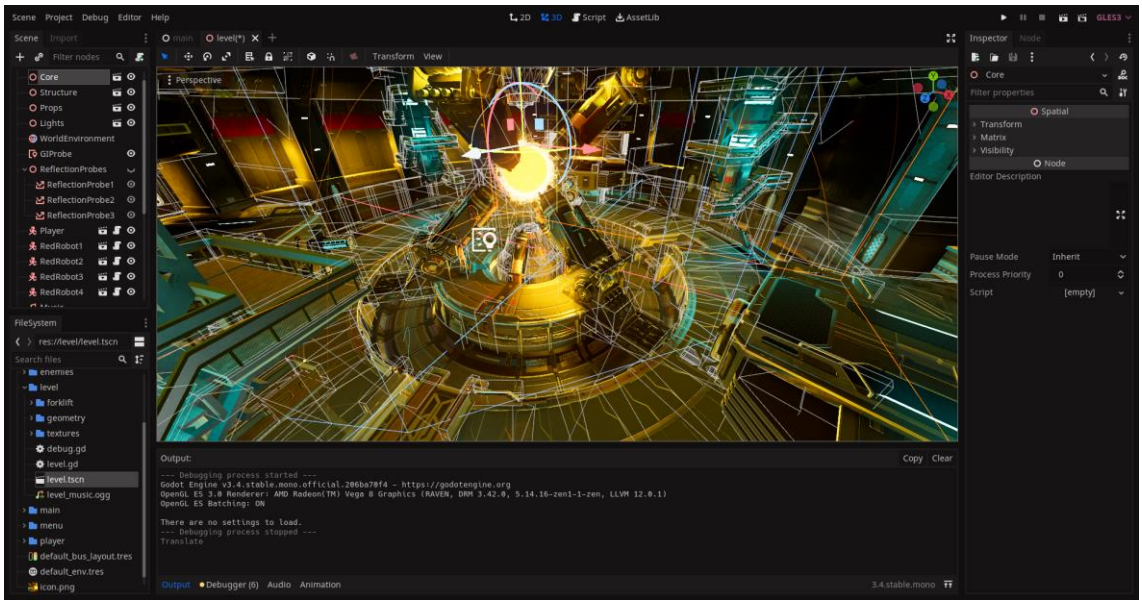

Figura 2.6: Editor de Godot [27]

3 Diseño

Durante este capítulo se expondrán los contenidos más relevantes que se hallan en el *Game Design Document* (GDD), adaptado para las necesidades del proyecto.

3.1 Descripción del juego

Este videojuego consiste en la resolución de un caso policiaco donde el jugador tendrá que interactuar y explorar el entorno para encontrar las pistas que permitan resolver el misterio. Las pistas obtenidas se mostrarán en la interfaz de usuario (UI) donde el jugador tendrá que utilizar la funcionalidad de la misma para poder descifrar el enigma.

3.1.1 Género

El juego se enmarca en los siguientes géneros:

- **Aventura:** se basa en la investigación e interacción con el entorno y personajes para ir avanzando en la trama. En este tipo de juegos, el jugador tendrá que enfrentar desafíos donde podrá resolverlos utilizando el ingenio o el conocimiento de las mecánicas de juego. Juegos que son ejemplo de este género son: las sagas de *"The Legend of Zelda"* o *"Uncharted"*.

Figura 3.2: Captura del videojuego *"The Legend of Zelda: Tears of the Kingdom"* [28]

- **Misterio:** se caracteriza por la presentación de un misterio donde se ha cometido un crimen. El enigma no se resuelve hasta el final del relato. A modo de inspiración, se ha utilizado juegos como *"Disco Elysium"*, *"Sherlock Holmes: Chapter One"* o "*L.A. Noire"* que implementan un misterio y dan herramientas al jugador para resolverlo.

Figura 3.2: Captura del videojuego *"Sherlock Holmes: Chapter One"* [29]

- **Point-and-click:** este género está diseñado basándose en el uso del ratón. Los jugadores tendrán que utilizar este periférico para poder mover al personaje, interactuar con objetos o ejecutar acciones. *"12 Minutes"* o *"Return to Monkey Island"* son videojuegos que se encuadran en el uso de estas características.

Figura 3.3: Captura del videojuego *"Return to Monkey Island"* [30]

3.1.2 Personajes

- **Personaje jugable:** el personaje que será controlado por el jugador es el inspector Katagiri.
- **Compañero:** es un personaje no jugable. Es una inteligencia artificial de la nave del protagonista, llamada Watson, que le irá acompañando. Proporcionará ayuda al jugador y le hará un tutorial al comienzo del

juego. También dará su apoyo al inspector a lo largo de la trama para poder resolver el caso.

- **IA de la nave Icarus:** es un personaje no jugable. Formará parte de la resolución final del juego y revelará secretos al protagonista al alcanzar la deducción final.
- **Científicos:** son dos personajes no jugables. Serán científicos se levantan del proceso de hibernación por orden de la nave, tras la exploración del jugador. Se puede hablar con ellos y proporcionarán dos pista para el inspector. Asimismo, se les puede hacer preguntas opcionales, que no son necesarias para el cumplimiento del juego, para saber más acerca de ellos y el contexto de la misión que tenía la nave Icarus.

3.2 Mecánicas del juego y *Gameplay*

El juego está diseñado y desarrollado en un entorno tridimensional y emplea una perspectiva isométrica para presentar el mundo al jugador. El mundo del videojuego, tanto personajes como objetos, sigue las leyes de la física del mundo real que simula el motor gráfico. Estas son las principales mecánicas que presenta el juego:

- **Movimiento:** El personaje podrá ser movido por el jugador utilizando el ratón y pulsando a la zona donde quieres que se desplace el protagonista. Además, se puede cambiar la rotación de la cámara con las teclas "W" y "E" para rotar a izquierda y a derecha, respectivamente, siempre siendo el punto fijo de rotación el personaje jugable.
- **Interacción con objetos:** Se puede interactuar con objetos clicando sobre ellos con el ratón. Cuando se interactué, se abrirá un cuadro de diálogo donde saldrán las opciones que puedes hacer con ese objeto. Estos pueden ser una pista o un objeto opcional con el que aprender más sobre la trama y el contexto de la misma.
- **Creación de deducciones:** Tras recoger pistas, estas se mostrarán en la interfaz en un inventario de pistas. Cuando tengas dos o más, estas podrán ser unidas. Si las dos pistas sean unidas conforman una deducción, esta se mostrará en el espacio de deducciones y las pistas se eliminarán al haber sido utilizadas. Sin embargo, si las dos pistas unidas no llevan a nada, entonces se verá sobreimpresionado en la interfaz que se ha equivocado el jugador y las pistas seguirán en el inventario para seguir siendo usadas.

3.3 *UI* del juego

Durante el juego existen distintas interfaces de juego con las que el jugador puede interactuar y ver información necesaria para que se pueda avanzar en el videojuego. Se han diseñado las interfaces de usuario de forma que sean lo más intuitivas y que tengan la mayor claridad posible.

- **Interfaz de pausa:** Mientras el jugador esté disfrutando del producto, podrá acceder a la interfaz de pausa mediante la tecla "Escape". En primera instancia, el usuario se encontrará un fondo blanco con distintas

opciones en la parte superior, donde puede elegir la opción a la que accede. Para interactuar con esta interfaz, el jugador tendrá que clicar con el ratón para acceder a las distintas partes del menú. En la zona superior izquierda se muestra el botón “Q” con el que el usuario puede volver al juego para proseguir. Esto se puede observar en la figura 3.4.

Figura 3.4: Interfaz general de pausa en el juego

Cuando el jugador pasa el ratón por cualquiera de estas opciones mencionadas, esta se pondrá en naranja como se puede ver en la ilustración 3.5.

Figura 3.5: Interfaz con la opción “Mente” señalada

A su vez, cuando se selecciona una de ellas, esta se quedará en naranja permanentemente mientras el jugador siga dentro de ese menú. Con ello, el usuario tendrá un recordatorio contextual de la zona de la interfaz que ha accedido en cada momento. La única opción disponible que está implementada es la del apartado “Mente”. Este funcionamiento se muestra en la figura 3.6.

Figura 3.6: Interfaz con la opción “Mente” seleccionada

- **Interfaz “Mente”:** Cuando se accede a la opción “Mente” dentro del menú, se abren dos opciones llamadas “Pistas” y “Deducciones”. Estos dos apartados funcionan igual que los anteriores, si el usuario posa el cursor en una de estas opciones se ilumina de naranja y se pondrá permanentemente de este color si es seleccionada.
- **Interfaz “Pistas”:** Como se muestra en la figura 3.7, esta interfaz contiene varias partes diferenciadas. En el centro del menú, se mostrarán las pistas recogidas a lo largo del juego. Para poder seleccionar las pistas, hay que clicar con el ratón en el texto mostrado por cada pista. Debajo de esto, se encuentra una zona de retroalimentación, donde a izquierda y derecha se muestran las dos pistas que se han escogido para efectuar una deducción. El recuadro negro que se presenta entre las dos opciones elegidas mostrará el texto “Deducción exitosa” o “Deducción fallida” si la deducción efectuada es correcta o incorrecta, respectivamente. Si se alcanza la deducción que permite llegar al final del juego, en este cuadrado se mostrará el texto “Deducción final alcanzada”. Cada vez que se haga una deducción exitosa, se eliminarán las dos pistas unidas con éxito. Por otro lado, en la zona inferior derecha de esta interfaz, muestra un recordatorio contextual de que al pulsar la tecla “Y” se puede acceder a una ayuda para poder.

Figura 3.7: Interfaz de pistas con dos pistas conseguidas

- **Interfaz "Deducciones":** Esta pantalla exhibe las deducciones exitosas alcanzadas mediante la unión de pistas en la anterior interfaz. Además, existe otro apartado de ayuda pulsando la tecla "Y" y está posicionado el recordatorio en la misma zona que en la interfaz de "Pistas". En esta zona no se puede interactuar hasta llegar a la deducción final, donde se podrá acceder al final del juego pulsando la tecla "E". Se puede ver el diseño final de esta interfaz en la figura 3.8.

Figura 3.8: Interfaz de deducciones con una deducción conseguida

- **HUD:** A lo largo del juego se verán sobreimpresionado en la pantalla notificaciones o ayuda para que el jugador entienda y sepa cómo debe ejecutar algunas acciones. Por un lado, cada vez que el jugador recoja una pista, se le mostrará una notificación en la parte superior derecha de la imagen. Esto se puede observar en la figura 3.9.
  Por otro lado, en la ilustración 3.10 se puede ver otro ejemplo de este HUD donde aparece un recordatorio contextual para que el jugador sepa cómo puede interactuar. De esta manera se busca hacer más sencilla e intuitiva la experiencia de juego.

Figura 3.9: Notificación de pista encontrada en la zona superior derecha

Figura 3.10: Recordatorio contextual para indicar que con la tecla "Z" se sale de esta interacción

- **Interfaz de diálogos:** Como se puede apreciar en la imagen 3.11, al interactuar con un objeto o un personaje, aparecerá un cuadro de dialogo con el que el jugador podrá interaccionar. En el cuadrado negro superior se muestra el texto del hablante o del objeto además del nombre del mismo. Debajo de esto, se encuentran las opciones con las que puede responder el jugador, donde puede haber una o varias opciones. Esta interfaz se ha obtenido del repositorio en línea oficial de Unity.

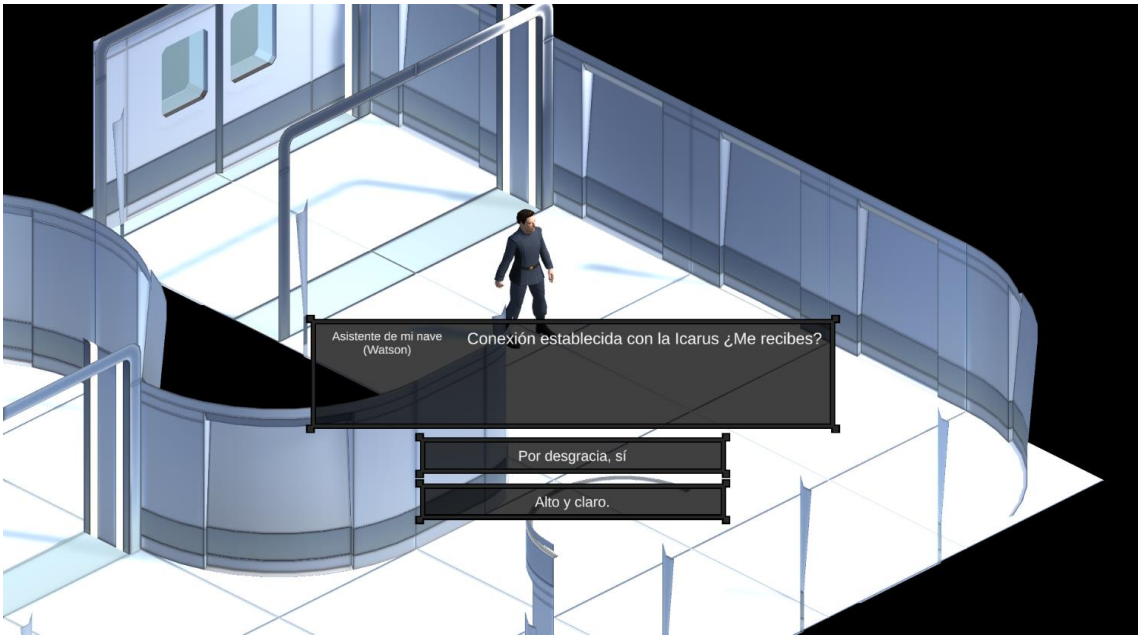

Figura 3.11: Ejemplo del cuadro de diálogo con dos opciones a elegir

3.4 Diseño de niveles

El diseño de niveles de este producto se ha realizado con la idea de hacer un caso policiaco donde se podrán adquirir hasta 8 pistas y realizar 4 deducciones. Para ello se ha diseñado un *game loop* principal, mostrado en la ilustración 3.12, que engloba y estas son las fases del mismo:

I. **Explorar el entorno:** el usuario se desplazará por el escenario, donde irá descubriendo formas de interactuar con el entorno que le rodea y lo que puede hacer.
II. **Interacción con objetos o NPCs:** después de realizar la exploración, el jugador podrá interactuar con los objetos designado para ello.
    a. **Interacción con objetos opcionales:** Durante el juego, el jugador se encontrará varios objetos que son opcionales a la hora de desarrollar la trama. Será voluntad del jugador poder interactuar con ellos al no ser necesarios para completar el arco argumental principal de la narrativa.
III. **Coleccionar pistas:** Tras haber interactuado con los distintos elementos interactivos, el jugador habrá coleccionado pistas, las cuáles pueden ser vistas desde la interfaz de pausa del juego.
IV. **Efectuar deducciones:** Una vez se hayan obtenido pistas suficientes para efectuar una deducción, el jugador tendrá que utilizar la interfaz de pistas para unir dos pistas y formar deducciones que se mostrarán en el apartado de deducciones del menú.

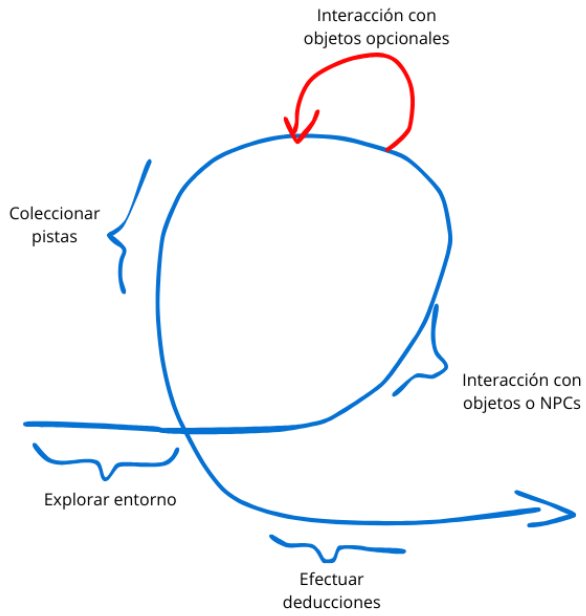

Figura 3.12: *Game loop* principal del juego

Durante el juego se puede acceder a 4 salas distintas que conformarán el escenario del mismo. Se ha buscado un diseño no lineal para así conceder más libertad al jugador a la hora de moverse por el escenario y así adaptarse a cada tipo de jugador. Por ello, el jugador podrá moverse y obtener las pistas de cada sala sin seguir un orden determinado. Aunque, sí que hay un elemento de linealidad en el diseño del nivel ya que para acceder a la sala de seguridad se necesitará haber visitado antes las dependencias del capitán donde tendrás acceso a la contraseña para abrir la zona de seguridad.

A continuación, se van a ir mostrando los distintos *game loops* que hay en cada habitación que se basan directamente en el principal, mostrado anteriormente. Sin embargo, hay algunas excepciones en cada *game loop*, ya que, por un lado, en cada sala puedes encontrar un número diferente de pistas. Asimismo, por otro lado, no es necesario que dos pistas que encuentres en la misma sala formen posteriormente una deducción, por lo que a la hora de formar una deducción puede ser necesario haber obtenido pistas de dos salas distintas. El usuario tendrá que visitar las 4 habitaciones que hay en el escenario del juego para poder alcanzar el final del nivel.

3.4.1 Habitación 1: Sala de mando

La primera habitación tiene como objetivo que el jugador empiece a conocer el contexto narrativo en el que se enmarca la historia. Asimismo, el jugador conocerá a dos científicos que estaban en estado de hibernación. Se podrán encontrar tres pistas y efectuar una deducción con ellas. El *game loop* es el siguiente y se puede observar en la figura 3.13:

I. **Exploración de la sala:** el jugador tendrá que buscar los elementos interactivos que hay en la sala, concretamente los proyectores.
II. **Interacción con proyectores la sala:** tras la búsqueda de objetos, el jugador deberá interactuar con los proyectores que hay en la sala. Cada uno de ellos provocará un efecto distinto, uno abrirá una compuerta para poder hablar con los científicos y el otro proporcionará una pista.
    a. **Interacción con objetos opcionales:** en esta estancia de la nave se encuentran varios ítems opcionales con los que poder interaccionar antes o después de haberlo hecho con los proyectores y se podrán tomar algunas decisiones con estos objetos.
III. **Interacción con NPCs:** tras haber interactuado con los proyectores, el jugador deberá hablar con los NPCs que acaban de aparecer tras haber desbloqueado la compuerta que llevaba a ellos dentro de la habitación de mando.
IV. **Coleccionar pistas:** en esta estancia se coleccionan tres pistas tras haber hablado con los NPCs y de haber observado el proyector.
V. **Efectuar deducción:** después de conseguir las pistas, se podrá efectuar una deducción entre las pistas “Protocolo de emergencia activado” y “IA asume el control” en la interfaz.

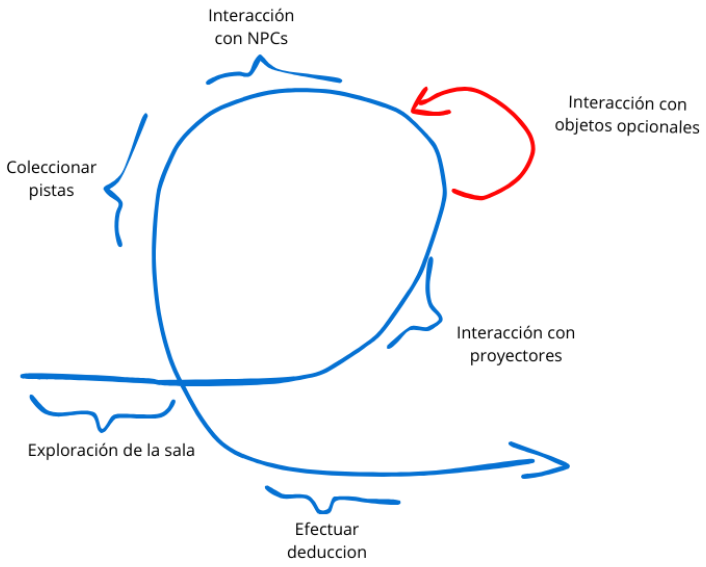

Figura 3.13: *Game loop* de la habitación 1

3.4.2 Habitación 2: Zona común de la tripulación

La segunda zona habilitada para poder explorar está diseñada para que el jugador conozca más a la tripulación que han destinado para la misión que iba a realizar la nave Icarus y se revelará más información con la que el usuario pueda comenzar a establecer teorías sobre lo ocurrido. En esta habitación se podrán conseguir tres pistas y establecer una deducción. Su *game loop* se puede ver en la imagen 3.14.

I. **Exploración de la zona:** el jugador deberá examinar la estancia para encontrar una silla, una mesa y un panel con los que podrá interactuar para continuar la narrativa, además de algunos objetos opcionales.
II. **Interacción con objetos principales:** el usuario interactuará con los objetos encontrados, mencionados anteriormente.
    a. **Interacción con objetos opcionales:** a su vez, se podrá interactuar con objetos opcionales que aportan más contexto sobre los tripulantes.
III. **Coleccionar pistas:** Tras haber interactuado con los objetos, las pistas serán coleccionadas y podrán ser utilizadas para efectuar deducciones en la interfaz de usuario.
IV. **Efectuar deducción:** se podrá establecer una deducción al unir las pistas "Indicios de violencia" y "Comida sin terminar" en el menú.

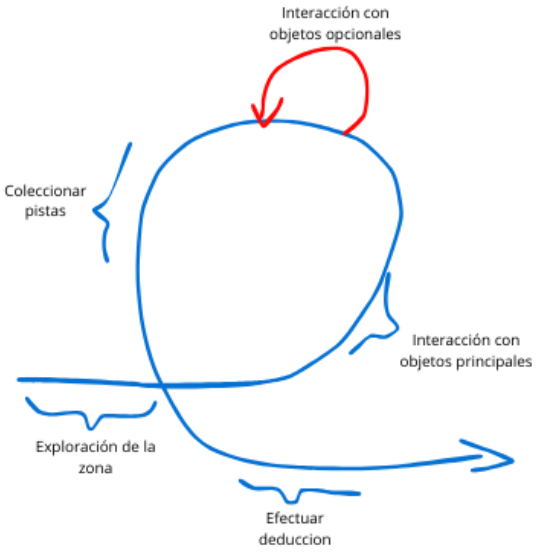

Figura 3.14: *Game loop* de la habitación 2

3.4.3 Habitación 3: Dependencias del capitán

La tercera habitación accesible en el escenario tiene como labor dar información sobre quién era el capitán de esta nave y además se conocerá su testimonio a través de su diario. Aquí se podrá conseguir una pista y establecer una deducción con una pista conocida en la Habitación 2. El *game loop* de esta sala se puede observar en la figura 3.15.

I. **Exploración de la sala:** el jugador tendrá que explorar los aposentos del capitán para encontrar los elementos interactivos.
II. **Interacción con el cofre:** tras explorar, el jugador podrá interactuar con el cofre para seguir con la narrativa de la historia.
    a. **Interacción con objetos opcionales:** de forma opcional, el jugador puede interactuar con otros ítems que se encuentran en la habitación para conocer más sobre la historia del capitán de la nave.

III. **Colecciona pista:** después de interactuar con el cofre, el jugador coleccionará una pista que se enseñará en la interfaz.
IV. **Efectuar deducción:** tras haber obtenido esta pista, el jugador podrá efectuar una deducción en la interfaz con las pistas "Tensión en la tripulación" y "Tripulación multicultural".

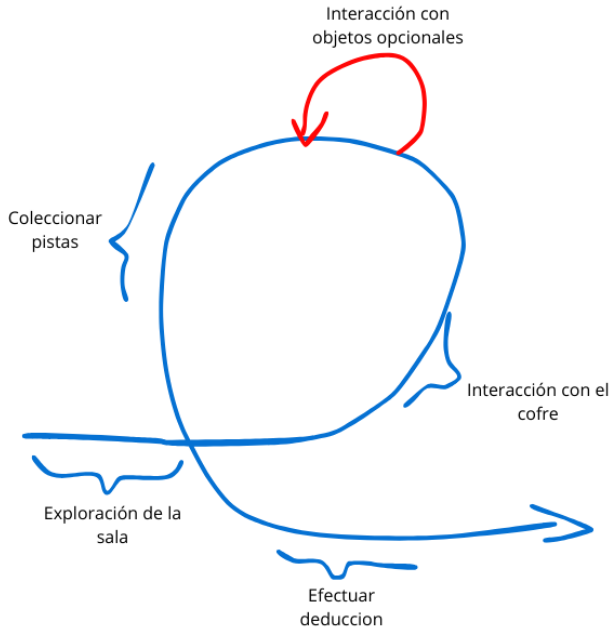

Figura 3.15: *Game loop* de la habitación 3

3.4.4 Habitación 4: Sala de seguridad

La última sala será accesible a través de una investigación secundaria en las dependencias del capitán. La zona de seguridad será la última habitación accesible en el juego y donde se encontrará la última pista que dará lugar a establecer la última deducción. El *game loop* es el siguiente y viene representado en la ilustración 3.16.

I. **Obtención de la contraseña:** para poder acceder a la sala de seguridad es necesario obtener una contraseña numérica que se podrá conseguir con los siguientes pasos:
   a. **Exploración de las dependencias del capitán:** el jugador deberá explorar las dependencias del capitán en busca de la contraseña de la puerta de seguridad.
   b. **Interacción con el escritorio:** después de la exploración el jugador tendrá que interactuar con el escritorio donde se obtiene que la contraseña es el nombre de su libro favorito.
   c. **Interacción con la estantería:** tras esta revelación, la interacción con la estantería desvela tres libros que el capitán tenía. La solución será el único libro que contiene una cadena numérica que en este caso es "1984".
II. **Abrir la puerta de la sala:** habiendo obtenido la contraseña, el jugador deberá abrir la sala utilizando para ello el panel que hay al lado de la puerta.

III. **Exploración de la sala:** una vez haya sido abierta la estancia, comienza la fase de exploración donde se buscarán los posibles objetos interactivos que haya repartidos por la sala.

IV. **Interacción con los monitores:** tras el estudio de la habitación, el jugador solo podrá interactuar solo con los monitores.

V. **Colecciona pista:** después de la interacción con los monitores se obtiene otra pista que se mostrará en el menú de pistas habilitado para ello.

VI. **Efectuar deducción:** por último, el usuario deberá establecer una deducción en la interfaz entre las pistas "Tripulación encarcelada" y "Importancia objetivo de la misión", obtenida en la Habitación 1.

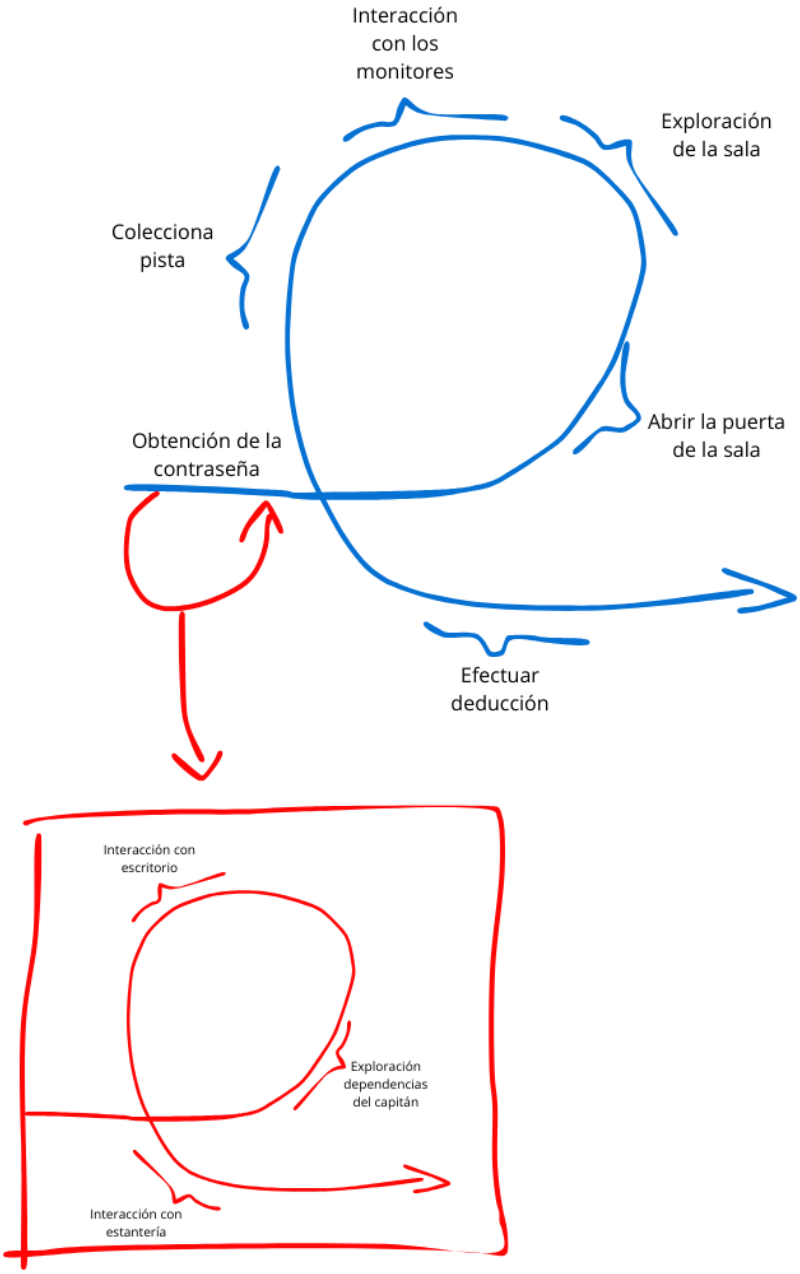

Figura 3.16: *Game loop* de la habitación 4

3.5 Estilo visual

Para este producto se ha buscado un estilo visual de lo más realista posible para poder aumentar la inmersión del jugador. A su vez, se ha utilizado un apariencia *low poly*, en la mayoría de los casos, para conseguir un estilo lo más similar posible, aunque los modelos fuesen de distintas fuentes. También esta apariencia ayuda a encontrar una mayor optimización de los recursos de cada sistema, haciendo que el juego sea más accesible para cada máquina.

Los objetos que se han usado proceden del repositorio en línea oficial de Unity llamado *"Unity Asset Store"* y los iconos con letras en 2D para los recordatorios contextuales provienen de la web "*Flaticon"*. Los modelos de los personajes que aparecen en el juego, concretamente los dos científicos, han sido obtenidos de la página *"Mixamo"*.

3.6 Métricas para el MTQ y Bartle Test

Para la obtención de los rasgos motivacionales del MTQ y de la personalidad del jugador del Bartle Test, se tuvieron que diseñar y establecer las métricas que el juego iba a tener que medir. Todas estas métricas se han diseñado para medir ciertas interacciones del usuario con el juego. Se han tenido que añadir algunas de estas interacciones para que algunas situaciones pudiesen ser medidas. A continuación, se detallarán las métricas utilizadas, con una explicación de porque se han utilizado.

3.6.1 MTQ

Dentro de los rasgos motivacionales que se pueden obtener del MTQ, se decidió descartar la medición de los rasgos de "Búsqueda de Competición" y "Otros Objetivos Referentes" ya que determinamos que nuestros juegos no podían obtener con suficiente precisión unas métricas que asegurasen estos rasgos motivacionales. Sin embargo, los cuatro restantes sí que han podido ser medidos, estos son: Determinación, Deseo de aprender, Evitación de fallos y Masterización.

En este juego, se determinó que algunas de las métricas que mejor se podían medir, dado el tipo de juego que es, pertenecían a los rasgos motivacionales de Determinación, Deseo de aprender y Evitación de fallos.

3.6.1.1 Determinación

Para el rasgo "Determinación" se han hecho mediciones de hasta dos métricas durante este juego y son las siguientes:

Acción medida	Peso para el rasgo	Explicación
Utiliza los apartados de ayuda de la interfaz de pistas y deducciones	5% por cada ayuda (hasta 10%)	Si el jugador utiliza los apartado de ayuda, significa que está determinado a seguir mejorando en el juego.
Elige la opción: "¿Algo más que deba saber?"	5%	Si el jugador hace uso de esta opción hace notar que está determinado a intentar conocer todo lo posible para tener éxito en el juego

Figura 3.17: Métricas para determinación

3.6.1.2 Deseo de aprender

Para el rasgo "Deseo de aprender", se midieron dos acciones distintas y son las siguientes:

Acción medida	Peso para el rasgo	Explicación
Utiliza los apartados de ayuda de la interfaz de pistas y deducciones	5% por cada ayuda (hasta 10%)	Si el jugador utiliza los apartado de ayuda, significa que está quiere aprender más sobre el juego y tener toda la información posible para poder utilizar mejor las mecánicas disponibles.
Interacciona con objetos (opcionales) en el juego	15% cada vez que supera estos números (3,5,7 objetos inspeccionados)	Si el jugador decide inspeccionar y conocer más sobre algunos objetos que son opcionales para la trama principal del juego, esto hace ver que el jugador quiere aprender más sobre el contexto narrativo del juego lo que significa que le gusta aprender.

Figura 3.18: Métricas para deseo de aprender

3.6.1.3 Evitación de fallos

Para esta métrica se obtuvieron datos sobre dos métricas distintas y son las que vienen a continuación:

Acción medida	Peso para el rasgo	Explicación
Utiliza los apartados de ayuda de la interfaz de pistas y deducciones	5% por cada ayuda (hasta 15%)	Si el jugador utiliza los apartado de ayuda, significa que quiere evitar fallar a toda costa y prefiere antes asegurarse mirando la ayuda proporcionada por el juego
Evita tomar decisiones que puedan llevar a un error	10% por cada decisión (hasta 30%	Si el jugador decide no tomar una decisión que puede implicar un fallo y se le ha avisado previamente de que esto podía ocurrir, esto hace notar que el jugador prefiere evitar cualquier posibilidad de error que se pueda evitar

Figura 3.19: Métricas para evitación de fallos

3.6.2 Bartle Test

Sabiendo que la taxonomía de Bartle tiene cuatro tipos de jugadores que son: Socializador, Explorador, Triunfador y Asesino, se ha decidido implementar algunas acciones para la medición de los jugadores de tipo Exploradores y Socializadores. En este caso si se ha determinado que se podían realizar las mediciones necesarias para poder catalogar con exactitud a todos los tipos que conforman esta prueba.

3.6.2.1 Socializador

En el caso de los socializadores, este producto tendrá el peso del 100% obtenido en las métricas ya que se decidió que este era el juego más acorde para poder obtener métricas de este tipo de jugadores al ser el único que tenía NPCs con los que se podía interactuar. A continuación, se puede observar qué métricas han sido tenidas en cuenta para este propósito.

Acción medida	Peso para el rasgo	Explicación
Tiene una conversación adicional (opcional) con un NPC	16.6% por cada vez que el jugador tenga una conversación de este tipo (hasta 100%)	Si el jugador decide tener un diálogo adicional con un NPC y que es opcional para el desarrollo de la trama, significa que al jugador le gusta socializar ya que disfruta de la interacción con personajes no jugables.

Figura 3.20: Métricas para socializador

3.6.2.2 Explorador

Para los Exploradores, también este juego tendrá el 100% del peso medido. Como se mencionó en el anterior caso, se decidió así ya que este juego era el más adecuado para obtener unas métricas precisas para medir si un jugador es explorador o no. Esto se eligió así dado que este juego era en el que más había que realizar un trabajo de exploración por parte del jugador. Se midieron dos acciones distintas y vienen desglosadas en la siguiente tabla:

Acción medida	**Peso para el rasgo**	**Explicación**
Interacciona con objetos (opcionales) en el juego	17.5% cada vez que supera estos números (2,4,6,8 objetos inspeccionados) (hasta 70%)	Si el jugador interacciona con distintos objetos que son opcionales para la realización del juego significa que el jugador explora el entorno que le rodea y es capaz de encontrar objetos en este.
Obtiene el secreto oculto	30%	Si el jugador encuentra el secreto oculto que hay a lo largo del juego, significa que ha hecho una buena labor de investigación y de exploración del escenario.

Figura 3.21: Métricas para explorador

4 Desarrollo

Durante el desarrollo de este producto software se trabajó con iteraciones para conseguir la implementación del juego. Para ello, se eligieron minuciosamente que tareas iban a ser desarrolladas antes en función de la importancia que cada una tenía. Este capítulo se dividirá en dichas iteraciones para poder mostrar la implementación hecha en cada una de ellas y las decisiones tomadas para alcanzar el estado final de la *Vertical Slice*.

- **Iteración 1:** durante esta iteración se comenzó la implementación del producto priorizando el desarrollo de las mecánicas necesarias para el juego entre las que se encuentran el movimiento del jugador, movimiento de la cámara, la lógica de los objetos Pista y Deducción y la gestión de un inventario de ambos objetos.
- **Iteración 2:** en esta fase de producción, se decidió atajar el desarrollo de mecánicas más complejas como la creación de la interacción entre jugador y objeto o toda la interacción del jugador con la interfaz
- **Iteración 3:** en el transcurso de esta iteración, se priorizó la formación del escenario y se hizo la colocación de todos los assets necesarios para poder garantizar la experiencia de juego.
- **Iteración 4:** esta fase fue dedicada a mejorar lo que existía dentro del juego entre lo que destaca: la inserción de los apartados de ayuda, recordatorios contextuales para hacer más accesible el juego, animaciones y uso de *triggers*.

4.1 Primera Iteración

Durante las dos primeras fases del proyecto se decidió no implementar nada que tuviese que ver con el escenario o con aspectos visuales. De esta manera, se buscó centrarse en implementar y probar las mecánicas que se fuesen introduciendo. Se implementó el movimiento del jugador mediante point-and-click, el movimiento y rotación de la cámara. También se crearon unos *Scriptable Objects* para crear la lógica de las pistas y de las deducciones. Por último, se hizo que los objetos pudiesen ser interactivos.

Para ello, se compuso un entorno básico compuesto de figuras geométricas que Unity permite insertar en las escenas. Se introdujo un rectángulo que hiciese de base y posteriormente se posicionaron unos cubos sobre el suelo que simulasen los objetos con los que se podía interactuar. Tras ello se introdujo una capsula para representar al personaje jugable. Dicho escenario se puede observar en la figura 4.1.

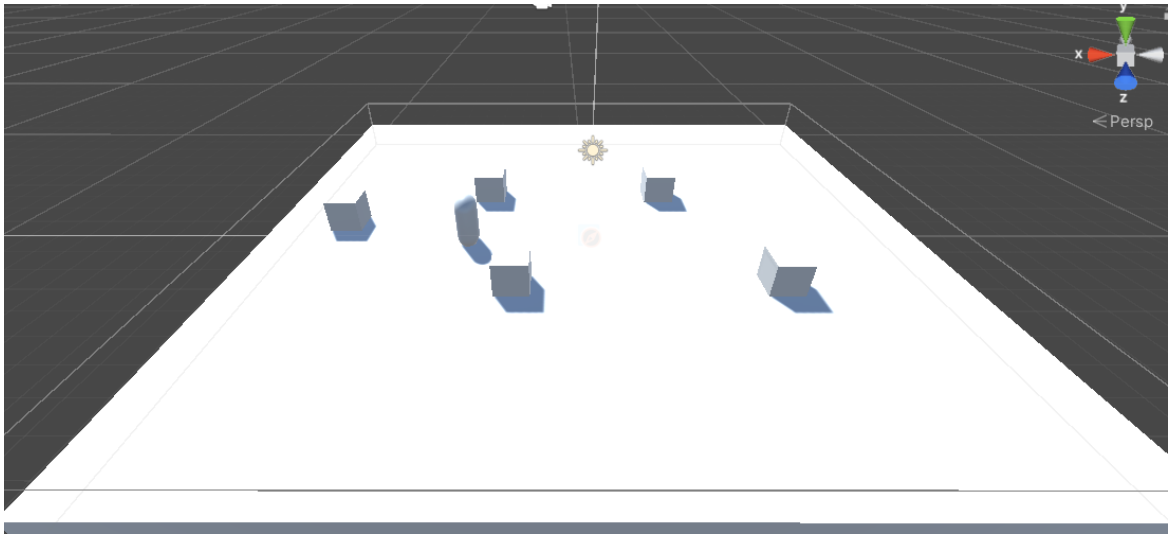

Figura 4.1: Entorno básico para implementar las mecánicas

4.1.1 Movimiento del jugador:

Tras establecer el escenario de pruebas para introducir las mecánicas, se decidió comenzar con la creación del movimiento del personaje mediante point-and-click.

Unity tiene un paquete llamado *NavMesh* [31] que consiste en una estructura abstracta de datos utilizada para ayudar a personajes a hacer *pathfinding* en un espacio determinado. Básicamente, su principal acción es la de reducir el número de nodos a lo largo del escenario para así aumentar la optimización a la hora de poder mover al personaje. Además, hace más sencillo la implementación del movimiento del personaje.

Antes de realizar ningún script, se tuvo que importar dicho paquete al proyecto. Tras ello, se tuvieron que añadir a los componentes *NavMeshSurface* y *NavMeshAgent* al escenario y al agente, respectivamente. El primer componente se configuro de tal manera que se le diesen propiedades al escenario para que un *NavMeshAgent* se pueda mover por él y define la zona donde el *NavMesh* se tiene que crear. El segundo componente, asignado al agente, fue implementado para que el paquete asignase a este objeto la capacidad de moverse por el escenario. Dentro de este componente hay que configurar algunas variables para ajustar mejor el movimiento del personaje y cómo evita los obstáculos. Además, en la imagen 4.2 se puede observar cómo quedo el escenario tras la inserción de estos componentes donde destaca una zona marcada en azul en el suelo que indica las zonas donde el jugador se puede mover.

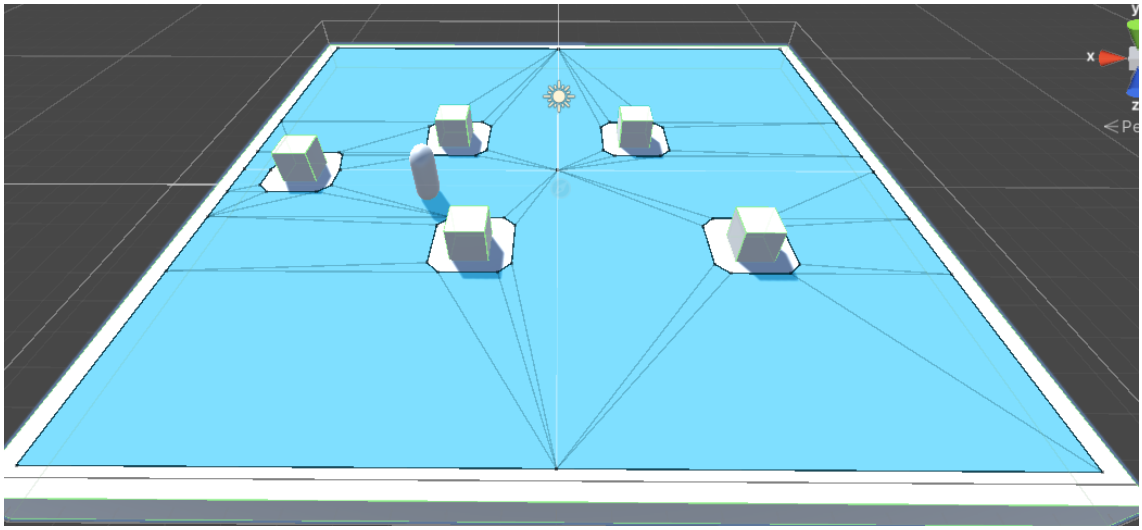

Figura 4.2: Suelo marcado con la zona donde el *NavMeshAgent* puede andar por el *NavMeshSurface*

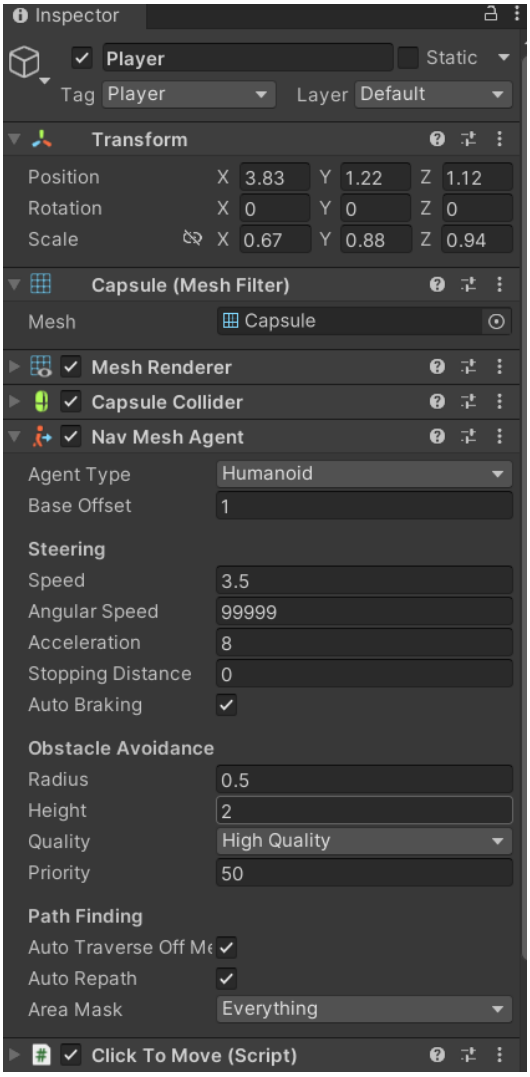

Figura 4.3: Componentes del objeto “Player”

Tras dejar bien configurado los anteriores componentes, se comenzó el desarrollo del script "ClickToMove" que se iba a asignar al objeto . Primero, se declaró una variable privada para el componente *NavMeshAgent* del objeto y en el método "Start" se obtenía y almacenaba en esta variable una referencia a este componente del mismo objeto al que está asignado este script. También se creó una variable de tipo *LayerMask* para identificar el terreno donde se puede mover el jugador. Después, se configuró el método "Update" que es llamado una vez por fotograma. Este método es el encargado de manejar la lógica de entrada del usuario. Primero, se crea un rayo desde la cámara hacia la posición del cursor del ratón y se almacena. Tras ello, se verifica si el rayo golpeo algún objeto y si ese objeto corresponde a la capa de terreno permitido. Esto se hizo utilizando una máscara de capa bit a bit. Por último, se verifica si el usuario ha pulsado el clic izquierdo para realizar el movimiento, si esto es así, se utiliza la función "SetDestination" proporcionada por el paquete *NavMesh.* Por otro lado, se le asignó el tag "Player" al jugador para su posterior uso donde se requiera la identificación del jugador. En la ilustración 4.3 se puede observar los componentes incluidos en el objeto "Player".

4.1.2 Rotación y movimiento de la cámara

En el diseño del juego se determinó que este producto iba a necesitar una cámara isométrica, por lo tanto, esta cámara tenía que estar por encima del objeto "Player" a una distancia y ángulos determinados para conseguir ese tipo de proyección.

Lo primero que se hizo para ello, fue asignar una cámara como "hijo" a un objeto vacío que seguía directamente al jugador. De esta manera, la cámara estaría apuntando a este objeto vacío y al hacer la rotación, en lugar de rotar sobre sí misma, esta rotaría alrededor de este elemento vacío, dejando al objeto "Player" siempre en el centro de la imagen. Por lo tanto, el objeto que realmente rota es este objeto vacío, aunque se da la sensación visual de que es la cámara la que lo hace.

Para poder controlar este movimiento y rotación de la cámara se realizó un script llamado "CameraController". Este script se asignó directamente al objeto vacío que se mencionó anteriormente. Se declararon tres variables públicas necesarias para el manejo de la rotación y el tiempo de interpolación para la misma. Al ser públicas, se podían cambiar su valor desde la interfaz de Unity, sin necesidad de cambiar el script. Además, también se declaró una variable pública para almacenar el *GameObject "*player". Para que la cámara se moviese con el movimiento del jugador, se utilizó el método "Update" para que se ejcutase en cada fotograma. En él se le asignaba una nueva posición al objeto vacío basándose en la actual posición del objeto "Player", por si éste se había movido.

Una vez hubiese acabado la ejecución de "Update", se utilizó el método "LateUpdate", que es llamado inmediatamente después del anterior. Este método se encargaba de manejar la rotación. Se decidió utilizar esta opción para evitar condiciones de carrera al cambiar la rotación mientras el objeto vacío se está moviendo. Este método comprobaba si se utilizaban las teclas "W" y "E" para rotar y dependiendo de la tecla hacerlo para un lado o para otro. De todas formas, ambas opciones incrementan la rotación en el eje Y y tras ello se suaviza la transición. Para este propósito se utilizaron las fórmulas Quaternion.Euler y Quaternion.Lerp.

4.1.3 Lógica de objetos Pista y Deducción

Para poder establecer la lógica de los objetos Pista y Deducción se utilizó la clase de Unity llamada *ScriptableObjects* [32]. Esta se utiliza para poder crear objetos de datos que son usados para guardar grandes cantidades de datos. Con esta clase, se reduce el uso de memoria por parte del proyecto. Los datos de estos objetos se almacenan como archivos que no necesitan estar adjuntos a un objeto en la escena de Unity. De esta manera, los datos son reutilizables y persisten entre escenas del juego. Gracias a esto podremos definir las variables que tendrán los objetos Pista y Deducción para poder realizar la función que se busca. La figura 4.4 muestra la creación de varios objetos de tipo "Pista" para mostrar el funcionamiento buscado.

Figura 4.4: Objetos de tipo "Pista" creadas

Esta lógica se incluyó en dos scripts diferentes llamados "Pista" y "Deduccion" que heredarán de *ScriptableObject*. Se utilizó en ambos un atributo de Unity llamado "CreateAssetMenu" que permite la creación de nuevas instancias de tipo "Pista" o "Deduccion" desde el editor de Unity. Tras ello se definieron las variables necesarias para cada una en el script. A continuación, en las tablas 4.5 y 4.6 se pueden observar las variables introducidas en cada caso.

Variable	**Tipo de Variable**	**Descripción**
pistaName	string	El nombre de la pista que será el utilizado para mostrarlo posteriormente en la interfaz
description	string	Una breve descripción más detallada acerca de la pista.

Figura 4.5: variables en Pista

Variable	**Tipo de Variable**	**Descripción**
DeducName	string	El nombre de la deducción que será el utilizado para mostrarlo posteriormente en la interfaz de usuario.
pista1	Pista	Primera pista necesaria para alcanzar esta deducción.
pista2	Pista	Segunda pista necesaria para alcanzar esta deducción
description	string	Breve descripción sobre esta deducción para aportar más información
enabled	bool	Esta variable es un indicador de si la deducción ha sido habilitada o no, dependiendo de si el jugador ha unido la pista1 con la pista2.

Figura 4.6: variables en Deduccion

4.1.4 Inventarios de pistas y deducciones:

Para la realización de estas instancias se siguió el patrón de diseño Singleton [33]. Esto también se conoce como patrón de instancia única y es utilizado para manejar clases e instancias. Este patrón nos permite asegurarnos de que una clase tenga sola una instancia creada y proporciona un punto de acceso global a ésta. Se tomo esta decisión ya que de esta manera se asegura que no haya varios inventarios diferentes y así solo hay una instancia que sea manejada para este propósito. Además, es necesario que se permita el acceso a estos inventarios de manera global y sencilla ya que estos van a estar en constante cambio y muchas interacciones a lo largo del juego necesitarán realizar llamadas a estos inventarios para hacer comprobaciones.

Se realizó un script para cada inventario llamados "InventarioPistas" e "InventarioDeducciones" aunque la lógica en ambos es muy similar. Las variables que conforman estos inventarios son una lista del objeto a almacenar y una referencia estática a la única instancia creada por estos inventarios, siguiendo el patrón Singleton.

Primero, se utilizó el método "Awake" que es llamado cuando el script se inicializa por primera vez. En este método se busca inicializar correctamente el inventario. Su funcionamiento consiste en que si la instancia es "null" significa que no hay no hay una instancia existente de este inventario, por lo que se asigna "this" a la referencia estática. Tras ello se asegura que el objeto no se destruya al cargar una nueva escena mediante la función "DontDestroyOnLoad". Por otro lado, si la instancia no fuese "null", el objeto duplicado sería destruido para mantener una sola instancia activa.

El inventario de pistas tiene un funcionamiento algo distinto al de deducciones. En el de deducciones se optó porque al inicio del juego el inventario tenga todas las deducciones posibles a obtener y se activan si el usuario las consigue. Mientras que el de pistas se inicializa vacío y va añadiendo las pistas a medida que el jugador las va recogiendo. Por lo tanto, para el inventario de pistas se diseñaron dos funciones estáticas llamadas "SetPista" y "HasPista".

"SetPista" comprueba que la instancia del inventario ha sido creada y si lo está, añade la pista a la variable lista que almacena las pistas.

"HasPista" hace la misma comprobación que SetPista para saber si la instancia fue creada. Si estuviese creada, devuelve true si la pista pasada a la función está en la lista y devuelve false si no lo está. Esta última función también está definida para el inventario deducciones con el nombre "HasDeduccion".

4.2 Segunda Iteración

La segunda iteración se dirigió hacia la implementación de las mecánicas principales del juego una vez ya fueron establecidas las mecánicas básicas para el desarrollo del mismo. Durante esta fase se creó la interacción con objetos y con la interfaz por parte del jugador.

4.2.1 Creación de interacción con objetos:

Para la realización de la interacción entre el jugador y los objetos se usó de nuevo la técnica de *raycast* como se tuvo que hacer para desarrollar el movimiento del jugador por el escenario. Para hacer esta interacción se desarrollaron 4 scripts diferentes llamados “Interactable”, “TextInteractable”, “CollectablePistas” y “PlayerInteraction”.

4.2.1.1 Interactable

Este script compone una clase abstracta que servirá de base para todos los objetos interactivos en el juego. También se definió una enumeración llamada “ObjectType” que categoriza los diferentes tipos de objetos interactivos que se pueden encontrar. Los valores de esta enumeración son: “pista”, “text”.

Además, se crearon dos variables llamadas “isInteracting” y “objectType” para indicar si el objeto estaba siendo interactuado y para indicar el tipo de objeto interactivo, respectivamente.

Por último, se añadió el método abstracto “Interact” que será implementado por las clases que hereden directamente de “Interactable”. Este método especificará la acción a realizar cuando el objeto sea interactuado por el jugador durante el juego.

Al utilizar una clase abstracta para hacer de “padre” de las clases que alberguen las interacciones, se busca la reutilización de código a través de la herencia ya que se permite que distintos objetos interactivos compartan la misma lógica y se podrán añadir nuevos tipos de objetos según sea necesario.

4.2.1.2 TextInteractable

Esta clase hereda directamente de la clase abstracta “Interactable” explicada anteriormente. En él se define un tipo específico de elemento interactivo que inicia una conversación cuando se realiza una interacción con el objeto. Este script se asigna directamente al objeto que necesitamos que interactúe con el jugador.

Para realizar esta conversación se utilizará el paquete importado llamado *DialogueEditor*. Por lo tanto, para poder iniciar esta conversación primero se debe crear una variable de tipo *NPCConversation* llamada “myConv” que albergará la conversación que debe mostrar cuando se interactúe con el objeto. Esta variable será privada, pero se le añadirá el atributo *[SerializeField]* para que a pesar de esta condición pueda ser visible y editable desde el inspector de Unity.

A continuación, se implementa el método “Interact” que ha sido heredado de la clase “Interactable”. Primero, se comprueba si ya se está interactuando con ese objeto con la variable “isInteracting”. Si esto es falso, marcará esta variable booleana como true y se utilizará la función “StartConversation” de *DialogueEditor* para iniciar la conversación.

4.2.1.3 CollectablePistas

Este script compone una clase que también hereda de la clase abstracta “Interactable”. El comportamiento que se ha desarrollado para este script es al interactuar con el objeto, iniciar una conversación, tras ello añadir una pista al inventario del jugador y mostrar una notificación temporal al acabar esta interacción. Este script tiene que ir asignado a un objeto y hará que se comporte como una pista.

Se asignan tres variables en la clase que son:

- **Pista:** en esta variable pública se introduce la pista a almacenar por el objeto y que luego será mostrada en la interfaz posteriormente una vez entre en el inventario.
- **pistaNoti:** hace referencia a un objeto de la escena de Unity para activarlo temporalmente cuando se colecciona la pista.
- **myConv:** igual que en la clase “TextInteractable”, se almacena una conversación específica que se inicia al interactuar con el objeto.

A continuación, se mostrará el flujograma en la figura 4.7 para detallar el proceso de interacción que se sigue.

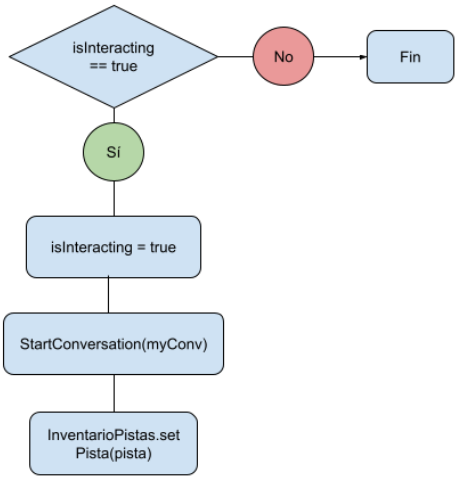

Figura 4.7: Flujograma de la clase “CollectablePistas”

4.2.1.4 PlayerInteraction

Esta clase manejará todas las interacciones del jugador con objetos interactivos de la escena menos los de la interfaz. Para ello, se utilizará un sistema de *raycasting* que detectará si un objeto es interactivo o no. Con este sistema, se

lanzará un rayo desde la cámara principal hacia la posición del cursor del ratón. Si el impacto se produce en un objeto interactivo se inicia una corutina para gestionar la interacción y además el jugador iniciará antes un movimiento para llegar al punto más cercano posible al que se encuentra dicho objeto.

Para la implementación de este script se han utilizado cuatro variables distintas para el siguiente propósito:

- **clickToMove:** hace referencia al componente "ClickToMove" del jugador que maneja el movimiento del jugador. Se obtendrá y se almacenará este componente al inicio.
- **interactRoutine:** variable de tipo IEnumerator que servirá para iniciar la corutina necesaria.
- **currentInteractable:** servirá para almacenar cualquier objeto de tipo "Interactable" con el que el jugador esté interactuando.
- **panelLayer:** hace referencia a un *gameObject* de la escena de Unity para bloquear posibles interacciones mientras el usuario esté utilizando la interfaz de usuario.

4.2.2 Interfaz de usuario

Para la realización de la interfaz de usuario, primeramente, se procedió a crear todos los elementos en la escena de Unity para luego darles su funcionalidad. Con este propósito, se tuvo que crear un *canvas* [34] que es el área donde deben estar todos los elementos de UI, por lo tanto, estos serán "hijos" del *canvas*. Este *canvas* permanecerá desactivado a lo largo del juego mientras el jugador no lo active.

Se desarrolló la siguiente jerarquía de objetos dentro del *canvas* que se puede ver en la figura 4.8.

Figura 4.8: Jerarquía de elementos del *canvas* en el inspector de Unity

Esta jerarquía fue planteada de tal forma que a medida que el usuario fuese entrando en las diferentes partes del menú, estas se fuesen desactivando y

activando. Para ello, fue crucial el uso de objetos vacíos que agrupasen los contenidos de cada parte de la interfaz, además de organizar y dar estructura a dicha jerarquía. En la figura 4.9, se presenta la utilidad de los componentes más importantes de la interfaz.

Componente	**Descripción**
Frame	Objeto que contiene el fondo de la interfaz y que siempre estará activado si el canvas está activado.
Header	Objeto vacío para agrupar los elementos “Tabs” y “Left”.
Body	Objeto vacío para agrupar los elementos a cambiar a medida que se va cambiando de pestaña.
Tabs	Alberga la lógica del cambio de pestañas en la zona superior.
Left	Muestra el botón con el que se sale del menú.
TabsMente	Alberga la lógica del cambio de pestañas dentro del menú “Mente”.
Switches	Objeto vacío para agrupar los elementos a cambiar dependiendo si estamos dentro de “Pistas” o “Deducciones”
Pistas (Switches)	Objeto vacío que agrupa todos los elementos que se enseñarán en el apartado de “Pistas”. Este se activará en caso de que se entre a esta parte del menú.
Deducciones (Switches)	Objeto vacío que agrupa todos los elementos que se enseñarán en el apartado de “Deducciones”. Este se activará en caso de que se entre a esta parte del menú.
Grid	Zona donde se almacenarán todas las pistas recogidas y albergará toda la lógica de unión de pistas.
PistaSeleccionada	Zona donde aparecerá la primera pista seleccionada para realizar una deducción.
DeducciónAviso	Objeto que se actualizará cada vez que se realice una deducción para indicar si ha sido correcta o incorrecta.

Figura 4.9: Descripción componentes de la jerarquía de la UI

A continuación, se mostrarán en las ilustraciones 4.10, 4.11 y 4.12 la interfaz resultante en el editor de Unity.

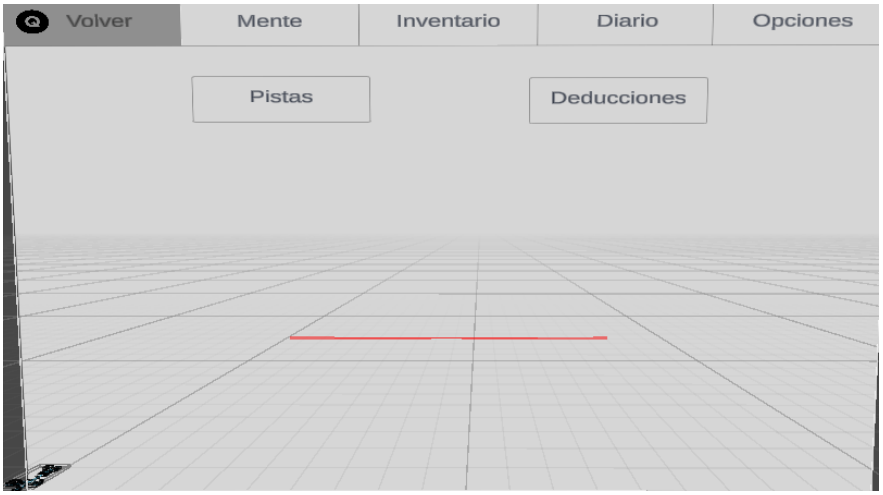

Figura 4.10: Interfaz de “Mente” donde se puede acceder a “Pistas” o “Deducciones”

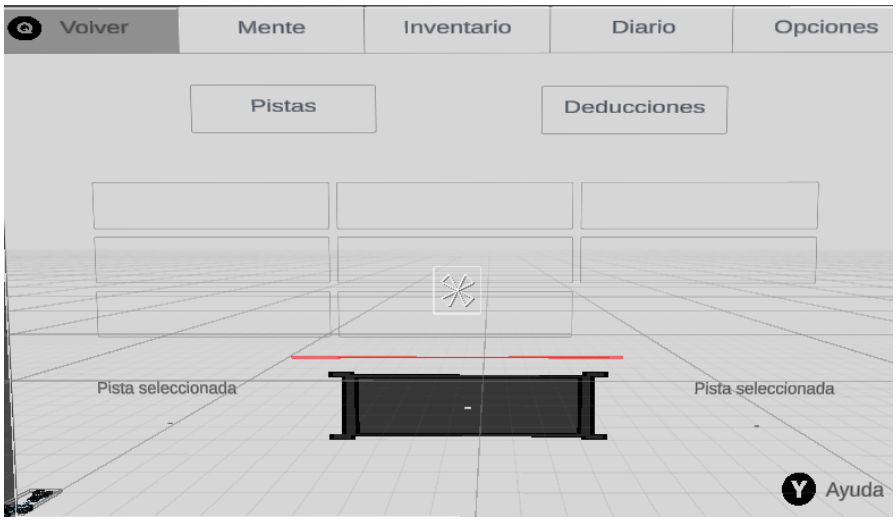

Figura 4.11: Interfaz de pistas con los huecos para las pistas

Figura 4.12: Interfaz de deducciones completa

La interfaz está diseñada de tal manera que sea lo más intuitiva posible y facilite al jugador la gestión de la información. Estas son las funciones de la UI:

- **Navegación por pestañas:** se podrá alternar entre las vistas de diferentes apartados de la interfaz como "Pistas" o "Deducciones", haciendo así que cada parte del menú sea más sencilla de distinguir.
- **Interacción con pistas:** en la sección "Pistas", las pistas se presentarán dentro del "Grid" y podrán ser seleccionadas para realizar deducciones. Cada vez que se consiga efectuar una deducción de forma correcta, se eliminarán las dos pistas utilizadas de la lista.
- **Muestra de deducciones:** a medida que se vayan consiguiendo unir las pistas correctamente, se irán mostrando las deducciones conseguidas en el apartado "Deducciones".
- **Notificaciones y ayuda:** hay secciones de ayuda y notificaciones en el uso de la interfaz que proporcionarán asistencia contextual.
- **Interacción mediante diálogos:** se importó un asset llamado *DialogueEditor* para poder crear diálogos interactivos.

4.2.2.1 Navegación por Pestañas:

La navegación por pestañas dentro de la interfaz se buscó que fuese lo más intuitiva posible. Para ello se diferenciaron dos partes claras que hiciesen más sencillo la implementación de esta mecánica. A continuación, en la ilustración 4.13 se pueden comprobar los objetos implicados en ello.

Figura 4.13: Componentes implicados en la implementación de esta navegación

Como se puede observar en la anterior figura, el objeto "Tabs" es el "padre" de los objetos "Tab" que serán los botones que pulse los jugadores. El objeto "padre" es vacío y alberga la lógica del funcionamiento mediante el script "TabGroup" que le será asignado a su objeto. Por otro lado, a cada botón se le asignará otra clase llamada "TabButton" que será encargada de recibir las interacciones de los usuarios y redirigirlas a "TabGroup" para que maneje el tipo de interacción. Cada objeto de tipo "Tab" es una imagen con fondo blanco con un texto dentro que especifique cada zona a explorar.

Para este propósito se utilizaron las siguientes interfaces que son parte del sistema de eventos de Unity: *IPointerClickHandler* [35], *IPointerEnterHandler* [36] *y IPointerExitHandler* [37]. Estos se encargan de detectar cuando un usuario hace clic o cuando el puntero del ratón entra o sale del área de un objeto de la UI. En la figura 4.14 se pueden ver los métodos que tiene la clase "TabButton".

Método	Descripción
OnPointerClick	Método que es llamado cuando se hace clic en la pestaña y llama al método “OnTabSelected” de “TabGroup” para gestionar el comportamiento.
OnPointerEnter	Método que es llamado cuando el puntero entra en área de la pestaña y llama al método “OnTabEnter” de “TabGroup” para gestionar el comportamiento.
OnPointerExit	Método que es llamado cuando el puntero sale del área de la pestaña y llama al método “OnTabExit” de “TabGroup” para gestionar el comportamiento.
Start	Método de inicialización que utiliza el método “Subscribe” de “TabGroup” pasando una referencia de sí mismo.

Figura 4.14: Métodos de la clase “TabButton”

Como se ha mencionado anteriormente, la clase “TabGroup” será la encargada de gestionar el grupo de pestañas. Controlará la apariencia cambiando el color del fondo de las imágenes en caso de que se haya seleccionado alguna de las pestañas. También se cambiará el color del botón cuando el puntero del ratón esté encima de una de ellas. Este funcionamiento se puede observar en la imagen 4.15.

Figura 4.15: Interfaz de pausa con el tab “Mente” seleccionado y el tab “Diario” apuntado

A continuación, se mostrará en las figuras 4.16 y 4.17, las variables utilizadas para esta clase y cada método implementado para el funcionamiento de esta clase.

Variable	Tipo de Variable	Descripción
tabButtons	List<TabButton>	Lista que almacena los “TabButton” que se inicializan con el comienzo del juego.
normalColor	Color	Color que se utilizará cuando el botón no sea seleccionado o no tenga el puntero encima de él.
highlatedColor	Color	Color que se utilizará cuando el puntero del ratón esté encima del botón.

selectedColor	Color	Color que se utilizará cuando el botón haya sido seleccionado.
selectedTab	TabButton	Variable que almacenará el último "TabButton" que haya sido seleccionado.
objectsToSwap	List<GameObject>	Lista de los elementos de la escena que tendrán que rotar cuando se produzca cambio de pestaña.

Figura 4.16: variables en TabGroup

Método	**Descripción**
Subscribe(TabButton button)	Método que añade al botón pasado a la lista de botones. Comprobará si la lista ha sido inicializada y la creará si no lo estaba.
OnTabEnter(TabButton button)	Método que dará color al Tab por donde pase el cursor del ratón mientras no sea el botón que está seleccionado. Invocará antes el método ResetTabs.
OnTabExit(TabButton button)	Invoca el método ResetTabs para restaurar todos los Tabs menos el seleccionado si lo hubiese
OnTabSelected(TabButton button)	Método que da color al botón seleccionado y ejecutará un bucle para activar el contenido necesario que está vinculado con cada pestaña. Almacenará en la variable "selectedTab" el botón que se haya seleccionado.
ResetTabs()	Método que ejecuta un bucle que itera sobre todos los botones en la lista "tabButtons" y que reinicia el color de los botones salvo el del botón que estuviese seleccionado si lo hubiese.

Figura 4.17: métodos en TabGroup

4.2.2.2 Interacción con Pistas y Muestra de Deducciones:

Para la realización de la interacción con las pistas, se siguió un enfoque parecido al de la navegación por pestañas. Por lo tanto, se utilizaron las mismas interfaces de Unity y se albergó la lógica de los botones y del conjunto por separado. Sin embargo, en este caso, se tenía que realizar la mecánica de unión de pistas.

Primero, se introdujeron diferentes componentes de la UI en la escena de Unity. Para ello, se creó un objeto vacío que se le asignó el componente de Unity *GridLayoutGroup* [38] que se utiliza para organizar unos mismos elementos en una cuadrícula de forma uniforme. De esta manera, se busca que los objetos que formen parte de este componente tengan el mismo tamaño y espacio.

Esto se utilizó para que las pistas fuesen organizadas en figuras iguales a lo largo de la cuadrícula. De esta forma, se introdujeron imágenes como "hijos" de este objeto vacío y se les asigno el componente *GridElement* para que se ajustase

al *layout*. Tras ello, se crearon las zonas de pista seleccionada, que consiste en la actualización de un objeto de *TextMeshPro*. Por último, en el apartado de pistas se posicionaron debajo de la lista de pistas una imagen con un elemento *TextMeshPro* para mostrar si la unión de pistas había sido correcta o incorrecta y así dar retroalimentación al usuario. En la figura 4.18 se pueden observar los elementos creados en el inspector de Unity.

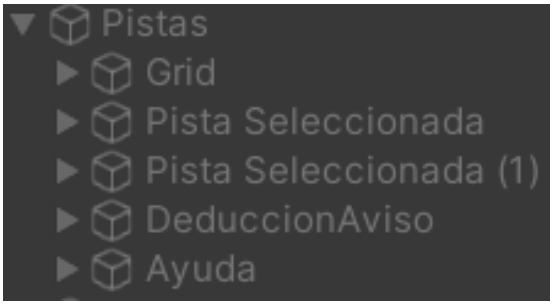

Figura 4.18: Objetos creados para la unión de pistas

Por otro lado, se introdujeron para el apartado de deducciones unas formas circulares con un texto al lado para mostrar las deducciones que se habían ido descubriendo a cada momento. Cada uno de estos elementos estaba dentro de un objeto vacío que agrupase cada círculo y cada texto con su respectiva deducción para así poder desbloquearlas de una manera correcta.

Tras introducir todos los elementos gráficos en la escena de Unity era necesario comenzar la implementación de la lógica de los botones y las pistas. Se crearon dos scripts llamados "Slot" y "SlotGroup". El primero fue asignado a cada imagen que fuese "hijo" del objeto vacío con el componente *GridLayoutGroup* y el segundo fue atribuido al "padre" de estos elementos.

Como en el manejo de pestañas, la clase "Slot" era la encargada de recibir la interacción del usuario mediante el ratón y llamar al método necesario en la clase "SlotGroup". En este caso, se añadió un método llamado "AddPista" que cambiaba el texto del "Slot" por el nombre de la pista para mostrarlo por pantalla y también asignaba a la variable pista del "Slot" la nueva pista. Este método se diseñó para la actualización de pistas que se explicará más adelante. A continuación, se pueden apreciar las variables de la clase "Slot" en la figura 4.19.

Variable	**Tipo de Variable**	**Descripción**
pista	Pista	Variable pública de tipo Pista que almacena la pista que se muestra en el slot.
tabGroup	SlotGroup	Variable pública de tipo SlotGroup que sirve para almacenar el objeto "padre" del slot.
pistaTexto	TMP_Text	Variable pública de tipo *TextMeshPro* que se utiliza para almacenar el nombre de la pista que será mostrado por pantalla.

Figura 4.19: variables en Slot

Mientras tanto, como se ha mencionado anteriormente, la clase “SlotGroup” se encarga de gestionar la interacción del usuario con los “Slot”. A su vez, esta clase también se encargaba de la mostrar por pantalla las pistas seleccionadas en cada momento y de desbloquear las deducciones a medida que se fuesen efectuando de forma correcta. Se van a mostrar las variables utilizadas en este script en la figura 4.20 y el funcionamiento de esta mecánica a través del flujograma de la ilustración 4.21.

Variable	**Tipo de Variable**	**Descripción**
tabButtons	List<Slot>	Lista que almacena todas los objetos de tipo “Slot” que hay.
selectedTab	Slot	Variable que almacena el objeto de tipo “Slot” seleccionado por última vez.
actualPista	Pista	Variable que almacena un objeto de tipo “Pista” seleccionado por última vez.
anteriorPista	Pista	Variable que almacena un objeto de tipo “Pista” anterior a la selección de “ActualPista”.
deduccUI	List<GameObject>	Lista de objetos de la escena de Unity que almacena los elementos a desbloquear de la UI cada vez que se efectúa una deducción correcta.
inventario	InventarioDeducciones	Referencia al inventario de deducciones.
pistaTexto1	TMP_Text	Variable de tipo *TextMeshPro* que servirá para actualizar la interfaz cuando se elige una pista.
pistaTexto2	TMP_Text	Variable de tipo *TextMeshPro* que servirá para actualizar la interfaz cuando se elige una pista después de haber elegido una anteriormente.
deduccionNoti	TMP_Text	Variable de tipo *TextMeshPro* que se actualizará cada vez que se efectúe una deducción para indicar por pantalla si ha sido correcta o no.
pistas	List<pistas>	Lista con todas las pistas disponibles.

Figura 4.20: variables en SlotGroup

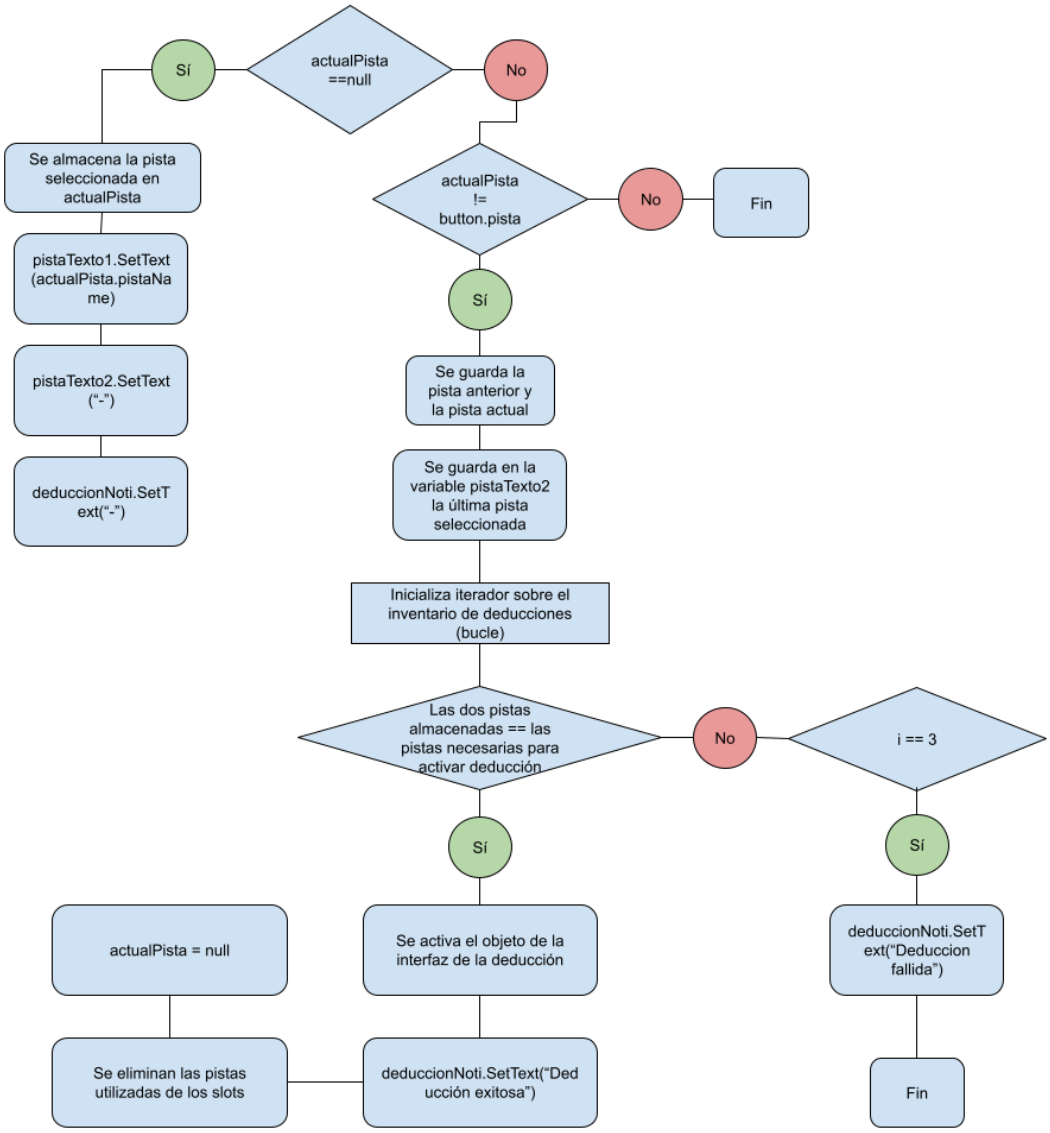

Figura 4.21: Flujograma de la clase SlotGroup

4.2.2.3 Actualización de pistas en UI

Durante el juego, el usuario tendrá que recoger pistas y poder interactuar con ellas en la interfaz de usuario. Por lo tanto, se decidió implementar una clase que llamase a un método cada vez que una pista fuese recogida para mostrarla en el menú. El script creado llamado "InventarioPistasUI" fue asignado al objeto *Canvas.*

Esta clase tiene una referencia al "InventarioPistas" que se haya creado como instancia única y luego un *array* con los objetos de tipo "Slot" que haya para mostrar las pistas.

Cada vez que se recoja una pista será llamado el método UpdateUI para poder actualizar lo mostrado en la interfaz. Primero, se itera sobre todos los slots disponibles en un bucle. Si el índice es menor que la cantidad de pistas en el

inventario, se añade la pista al correspondiente “Slot” utilizando el método de esta clase llamado “addPista” que se explicó anteriormente. En cualquier otro caso, no se realiza ninguna acción ya que no hay más pistas descubiertas que mostrar en el menú.

4.2.2.4 Interacción mediante diálogos

La interacción de los diálogos se realizó mediante el asset *DialogueEditor* que se obtuvo en la Unity Asset Store. Este paquete permitía la opción de diseñar diálogos con nodos como se muestra en la imagen 4.22.

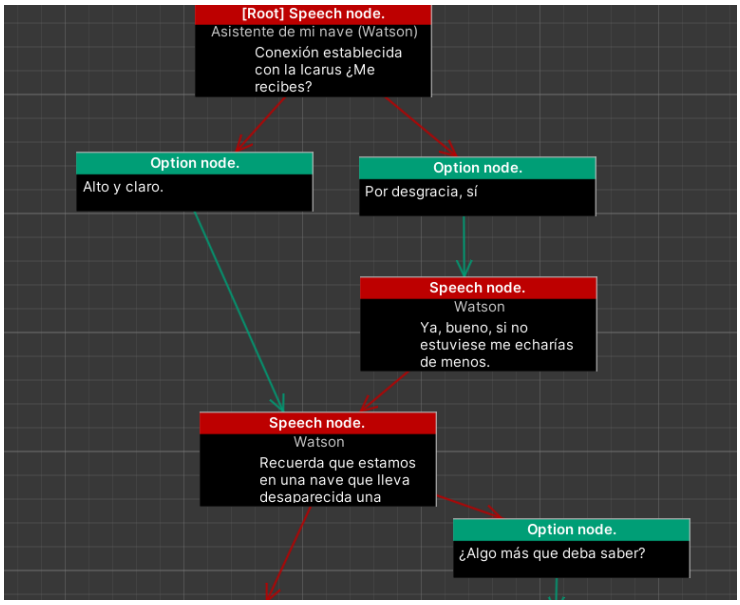

Figura 4.22: Editor para crear los diálogos

Para utilizar esto, se necesitaba realizar un objeto vacío y asignarle la clase *NPCConversation* que venía con el asset, cada vez que se quisiese hacer un diálogo diferente para cada objeto. Por último, se modificaba el diálogo con ese sistema de nodos para satisfacer las necesidades del juego y se tenía que utilizar el método *StartConversation* pasándole la referencia del diálogo que se quería iniciar. También se decidió bloquear la interacción del usuario con el juego mientras este cuadro de diálogo estuviese activado. Para ello, se introdujo en las clases que gestionan el movimiento y la interacción una condición en la que, si el objeto de la interfaz de diálogos está activado, no realizase la lógica de estas dos acciones. De esta manera, cuando el cuadro esté activado, no se podrá interactuar con nada que no sea este mismo objeto.

A modo de organización, se decidió que cada objeto o NPC que tuviese una conversación tuviese este objeto vacío como “hijo” para facilitar la estructura y la jerarquía de elementos en el inspector de Unity.

4.3 Tercera Iteración

Tras finalizar las principales formas de interacción del usuario se decidió empezar a realizar todo el escenario del juego incluyendo la inserción de los objetos con los que el jugador interactuaría. De esta manera, se formarían la narrativa del videojuego y los *game loops* de cada habitación.

4.3.1 Escenario de Juego

El diseño del escenario de juego fue pensado de tal forma que satisficiera las necesidades para poder desarrollar la narrativa y diesen coherencia. Se importaron diferentes *assets* de la Unity Asset Store priorizando que diesen la inmersión buscada.

Cada habitación fue diferenciada por pasillos para crear una clara diferenciación entre cada sala. Primero, se creó la estructura de suelos y paredes de cada habitación dentro de un objeto vacío que englobase todo. Este elemento vacío iba a albergar el componente *NavMeshSurface* para crear la zona donde se pudiese andar en el suelo. Luego se le añadiría cada objeto nuevo como "hijo" para que este componente los tuviera también en cuenta a la hora de calcular el camino más corto para que el agente se moviese. Estos objetos, algunos decorativos y otros esenciales para la interacción, fueron colocados de forma estratégica.

En las ilustraciones 4.23, 4.24, 4.25, 4.26 y 4.27 se pueden apreciar los resultados conseguidos en este apartado.

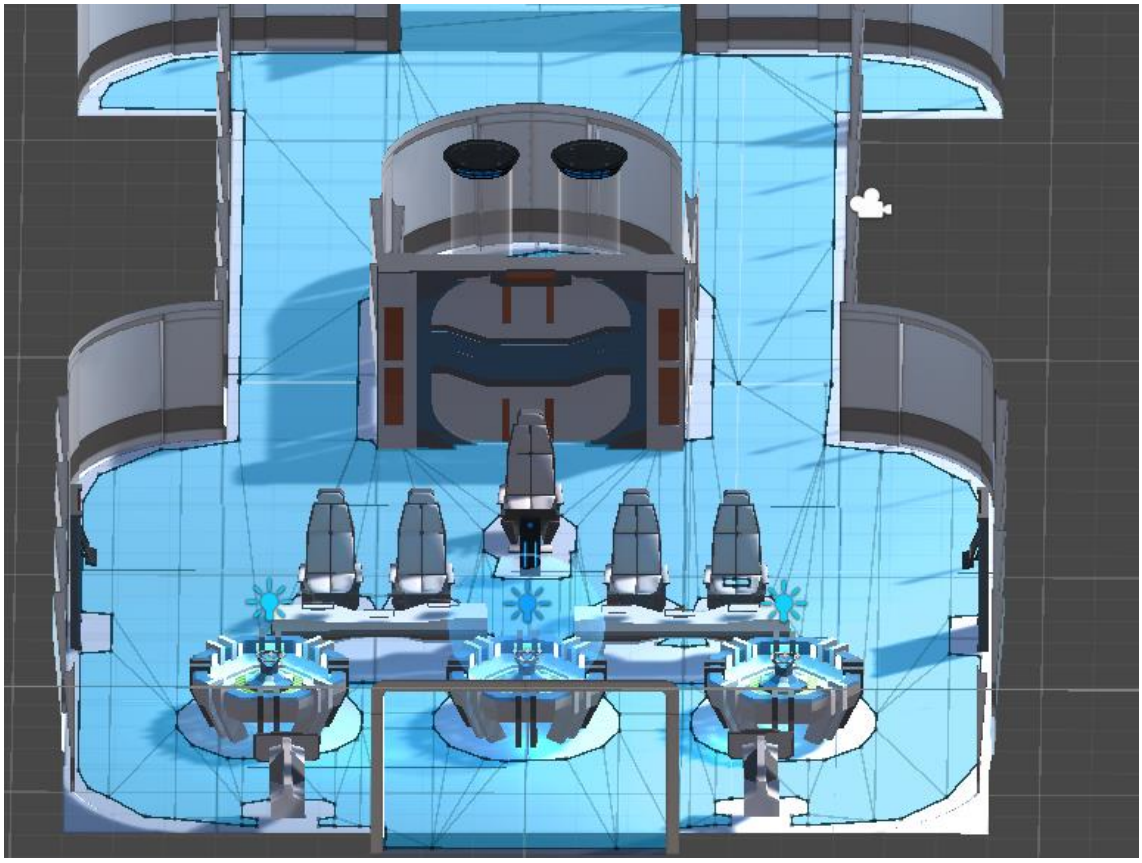

Figura 4.23: Puente de mando de la nave

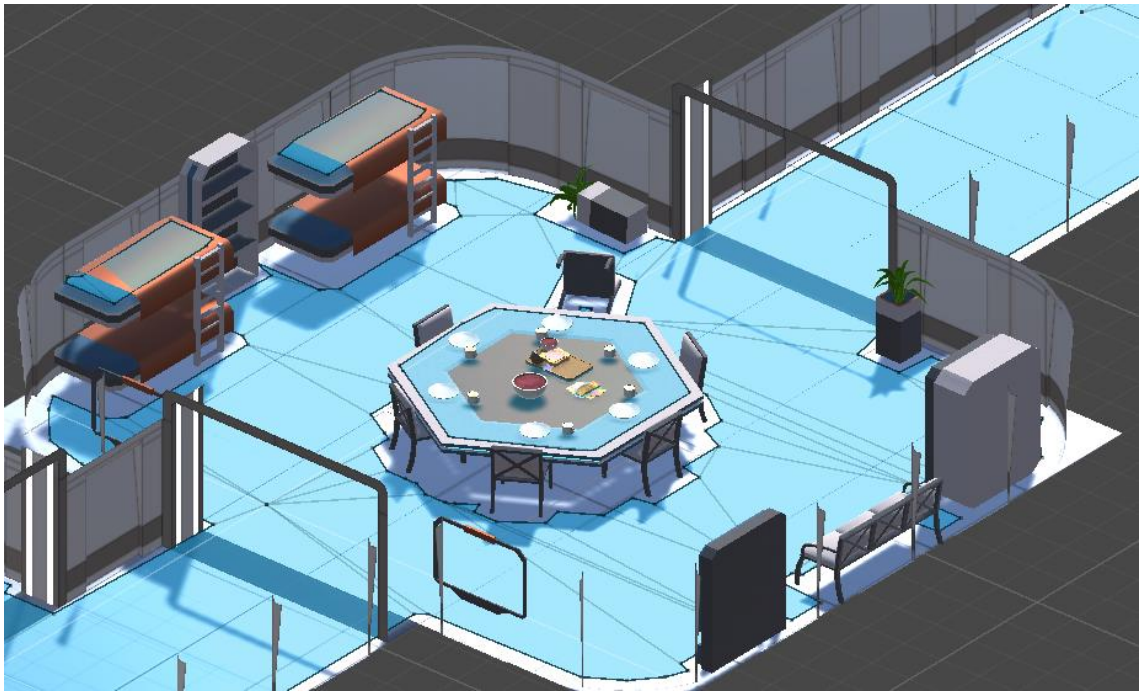

Figura 4.24: Sala común de la tripulación

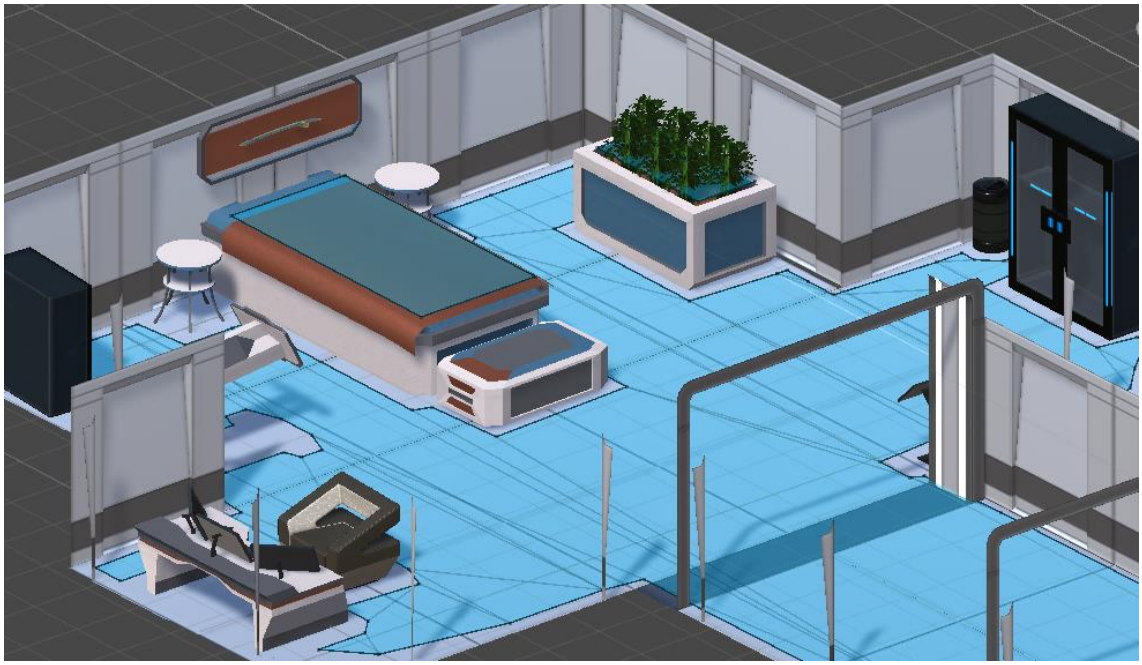

Figura 4.25: Dependencias del capitán

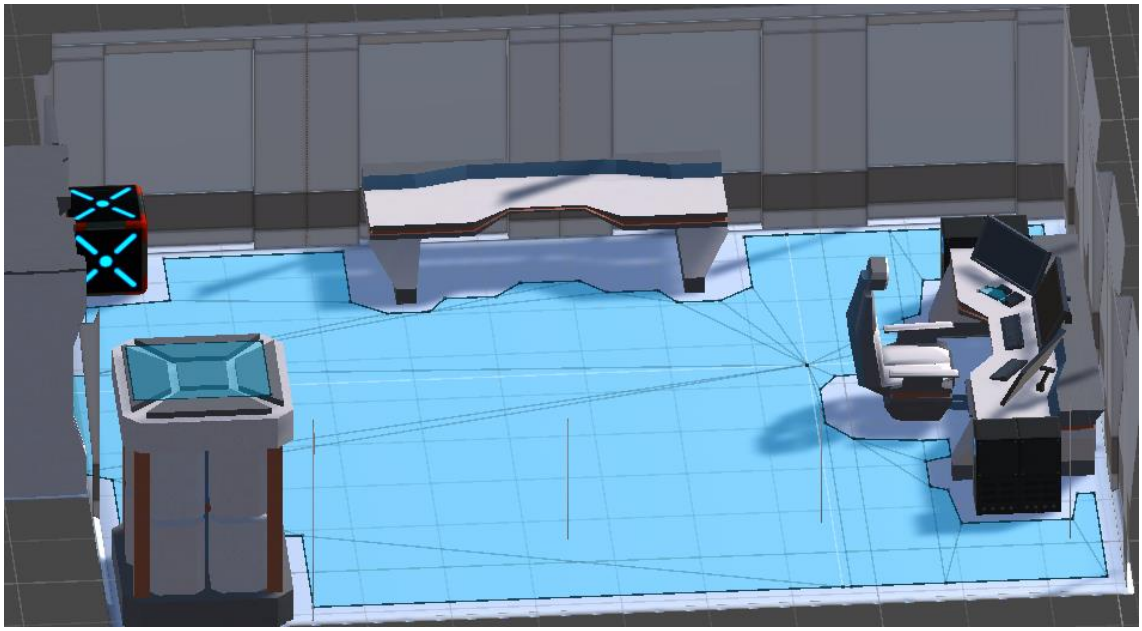

Figura 4.26: Sala de seguridad

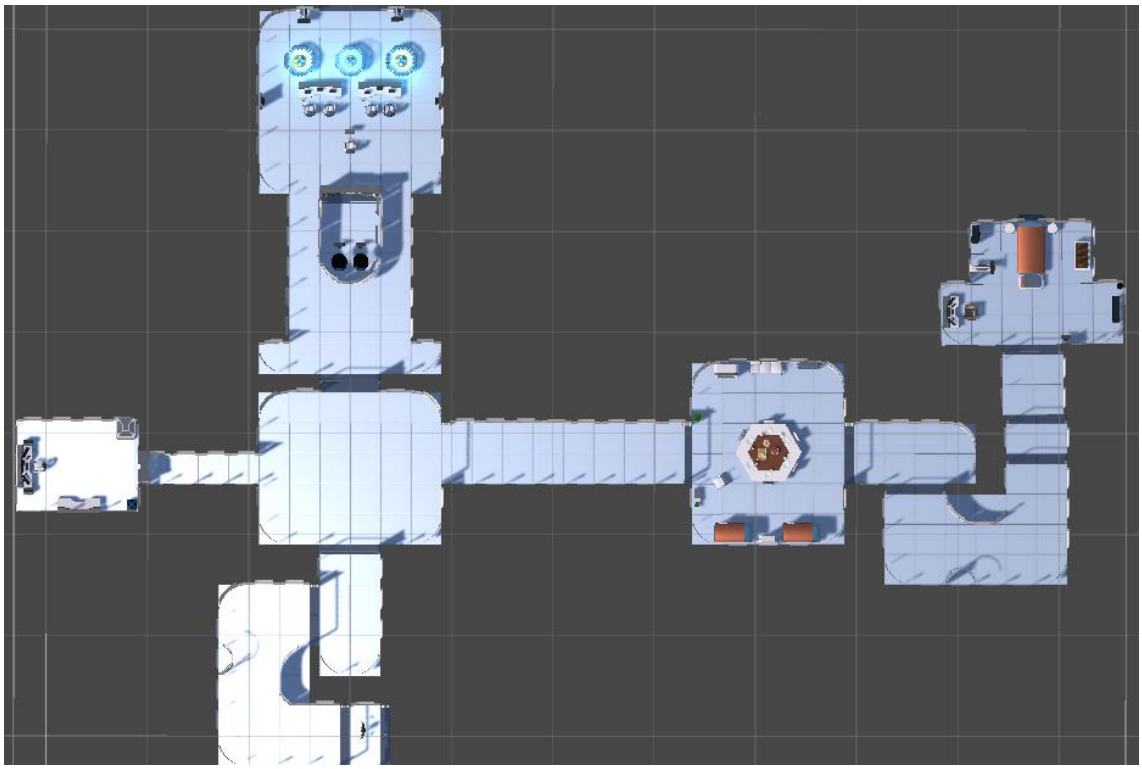

Figura 4.27: Distribución de la nave

4.3.2 Contorno de Objetos

Los juegos isométricos y de point-and-click suelen añadir un contorno a los objetos llamado *outline* que añadirá un borde de un color llamativo alrededor de la figura de los elementos al posar el puntero del ratón por encima del objeto. De esta manera, se busca resaltar los objetos seleccionables para hacer más intuitivo el producto. Para este propósito, se importó un paquete de la tienda oficial de Unity. Los objetos se marcarían como se muestra en la imagen 4.28.

Figura 4.28: Proyector con el *outline*

Para hacer funcionar esta herramienta, se tuvo que designar el tag “Selectable” a los objetos interactivos que debían ser señalados por su capacidad de ser interactivos. Se volvió a utilizar la técnica del *raycast* para poder identificar estos elementos. Además, se desarrolló una clase llamada “OutlineSelection” y se muestra en la figura 4.29 su flujo.

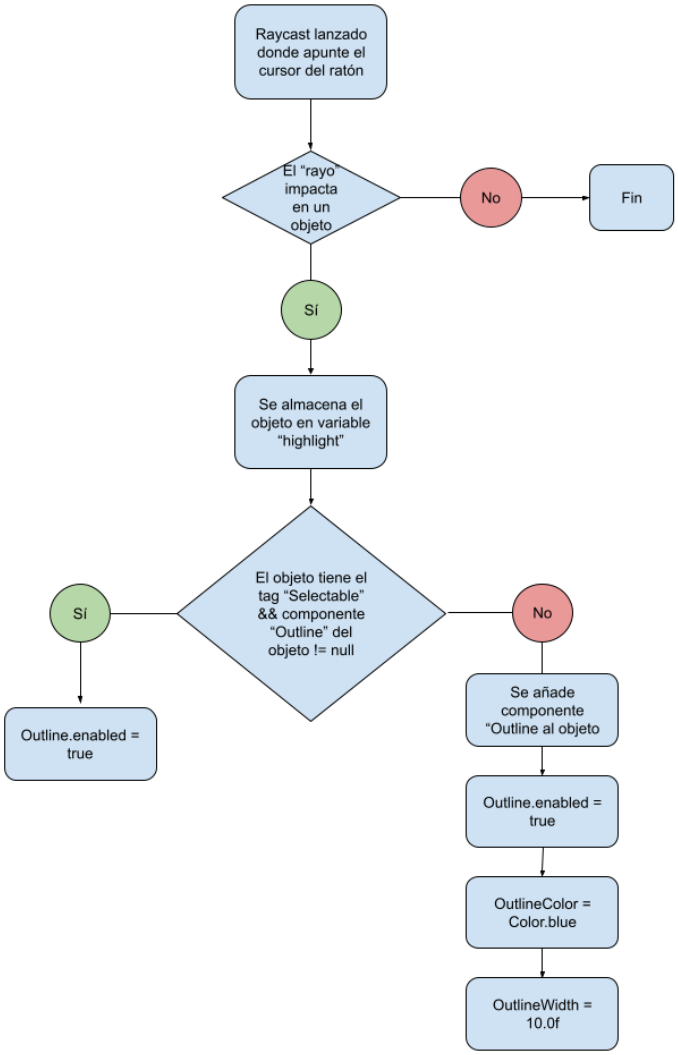

Figura 4.29: Flujograma del funcionamiento de la clase “OutlineSelection”

4.3.3 *Keypad* y Cambio de Cámara

Para satisfacer el *game loop* de la habitación 4 se descargó un asset de la Unity Asset Store que permitía introducir una contraseña y que cuando esta fuese correcta se efectuase una acción, que, en este caso, iba a ser una animación de una puerta. Este objeto se puede observar en la figura 4.30.

Figura 4.30: *Keypad* utilizado para el desbloqueo de la puerta

Se realizó un script llamado "KeypadCamera" para poder controlar el cambio de cámara entre la principal y la cámara que está apuntando al *keypad*. Además, también verificará si se presiona la tecla Z para poder volver a la cámara principal. En este caso, no se siguió el enfoque basado en *raycasts* para captar la pulsación del ratón en el objeto. Se utilizó el método *OnMouseDown* [39] que es llamado cuando el usuario clica en el *collider* del objeto. Por lo tanto, se tiene que asignar el script al elemento que vaya a ser pulsado para realizar la interacción con el *keypad.* Se eligió la puerta para tener este script asignado ya que iba a ser más intuitivo para el usuario.

Por lo tanto, si se pulsa en el objeto seleccionado, activará la cámara que apunta al *keypad* y desactivará la principal. Por otro lado, si se teclea la tecla Z se hará el proceso contrario.

4.4 Cuarta Iteración

Durante la cuarta iteración se hizo hincapié en perfeccionar lo ya existente, además de añadir algunas funciones que hiciesen la experiencia de usuario más inmersiva e intuitiva. Se preparó el juego para valorar la experiencia de usuario que se realizó durante esta fase del proyecto. Por ello, como se ha mencionado, se estuvo dedicando tiempo a dejar las mecánicas lo más pulidas posible y que el producto no tuviese ningún tipo de *soft lock*, que son situaciones donde el juego sigue aparentemente jugable, pero resulta imposible continuar.

Se decidió introducir unos modelos de personajes para representar al personaje principal y a los NPCs que están durante el juego. También se introdujeron algunos iconos 2D y el HUD para ayudar al jugador. Por otro lado, se implementó el uso de unos *triggers* para aumentar la experiencia narrativa y para introducir un tutorial donde se explicase la interacción del jugador con el entorno.

4.4.1 Modelos y Animación de Personajes

Se descargaron varios modelos humanos para caracterizar a los personajes que aparecen en el juego. El personaje principal fue adquirido en la tienda oficial de Unity, mientras que los NPCs fueron conseguidos en la plataforma *Mixamo.* Los personajes descargados venían todos por defecto con la "pose en cruz" también conocida como *T-pose*. Esta es la pose en la que están este tipo de modelos antes de ser animados. Por lo tanto, era necesario animar a los personajes para darle vida y realismo. Estas animaciones fueron también adquiridas de los mismos sitio web que los personajes. A continuación, se muestra en la imagen 4.31 al modelo del personaje principal con la "pose en cruz".

Figura 4.31: Modelo del personaje principal con *T-pose*

Por lo tanto, se tuvo que añadir el componente *Animator* al objeto "Player" para poder animarlo. Se utilizaron dos animaciones, una llamada "Idle" para cuando

el personaje estuviese sin moverse y otra llamada “Walk” para cuando el personaje tuviese que andar. Esto se puede observar en la figura 4.32.

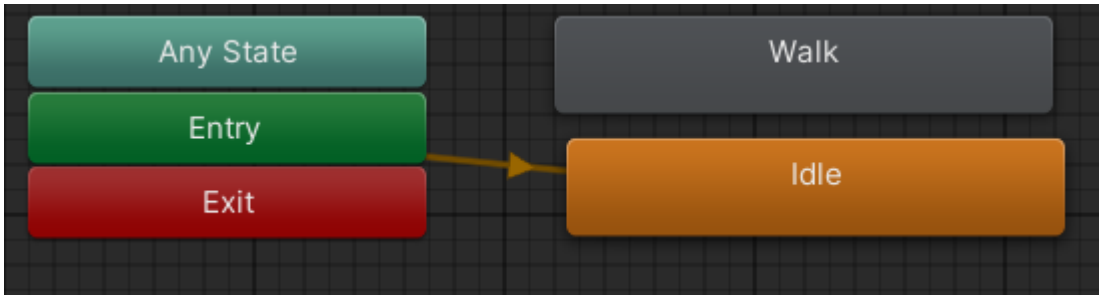

Figura 4.32: Componente *Animator* del objeto “Player”

Tras ello, era necesario darle la lógica a través de un método que gestionase las animaciones que se tuviesen que ejecutar en cada momento. Se decidió realizar un método llamado “SetAnimations” dentro de la clase “ClickToMove” que controlaba el movimiento del jugador. En él se comprobará si el jugador no tiene velocidad, si esto es verdadero se utilizaría la animación “Idle” y en el caso contrario se usaría la animación “Walk”.

Además, se desarrolló un método nombrado “FaceTarget” para rotar el modelo del personaje de modo que mire en dirección a su objetivo de destino. Primero, se calcula la dirección hacia el destino, tras ello se crea un objeto de tipo *Quaternion* que albergará la rotación necesaria para alcanzar la orientación deseada. Por último, se interpolará la rotación actual del personaje hacia la rotación creada anteriormente.

4.4.2 Uso de *Triggers*

Los *triggers* sirven para provocar un evento cuando, por ejemplo, el jugador entra dentro del *collider* del objeto al que se le ha dado el componente de ser *trigger*. Para este propósito, los *colliders* tienen una opción dentro del inspector de Unity llamada *isTrigger* que hay que marcar y de esta manera los *collider* no afectarán a la física del objeto. Se implementó el uso de *triggers* en el juego utilizando objetos vacíos y se les añadió un componente llamado *box collider* a modo de *trigger* para identificar la zona donde debía de ejecutarse el evento.

Para hacer uso de esta función, se tuvo que desarrollar una clase llamada “ColliderConv” que iba a ser asignada a estos objetos vacíos. Este script utiliza el método proporcionado por Unity llamado *OnTriggerEnter* [40] que se ejecutará cuando un objeto entre dentro del *trigger* del elemento al que esté asignada la clase. Dentro de este método, se comprobará si el objeto colisionado tiene el *tag* “Player” para así saber si es el jugador el que ha entrado y de esta manera iniciar la interacción.

Esto se utilizó al principio del juego para indicar al jugador los controles del juego y también para introducir ciertos elementos narrativos para poner en contexto al usuario.

4.4.3 HUD

Se creó un *canvas* diferente al del menú para gestionar esta nueva interfaz. El HUD se refiere a toda la información que se muestra en pantalla durante el juego para ayudar al jugador a entender lo que está pasando.

En este caso, se decidió añadir notificaciones en la zona superior derecha cada vez que el jugador recogiese una pista. También, se añadieron algunos recordatorios contextuales donde se indicase con que botón se podía realizar alguna acción.

Para ello, se añadieron la activación y desactivación de estos elementos en los momentos en los que se requiriese. Por ejemplo, al recoger las pistas se añadió una activación del objeto que albergaba el cuadro de la notificación y se implementó una corutina para que se ejecutase en paralelo a la interacción. Esta corutina espera 5 segundos tras activar la notificación y luego la desactivaría.

5 Evaluación y resultados

5.1 Evaluación

5.1.1 Proceso de evaluación

La etapa final de este proyecto se basó en la evaluación de la experiencia de usuario y la comparación de los resultado obtenidos mediante las métricas medidas en el juego con los cuestionarios reales que los usuarios realizaron. Estas evaluaciones se realizaron en el MadHCILab de la Facultad de Informática de la UPM y los usuarios evaluados fueron estudiantes voluntarios. A lo largo de este proceso se fueron registrando observaciones y se anotó las métricas completadas por los jugadores. Cada sesión de juego duraba como máximo 15 minutos y se indicó a los usuarios que expresasen sus pensamientos en voz alta para poder analizar de una mejor manera los puntos a mejorar en el diseño del producto. Durante la observación se anotó si los jugadores lograban llevar a cabo las tareas marcadas para el juego:

- Completa Deducción1
- Completa Deducción2
- Completa Deducción3
- Completa Deducción4
- Completa Keyboard
- Consigue todas las pistas
- Encuentra el secreto
- Usa menús de ayuda

Estas tareas se definieron para comprobar si el jugador efectuaba las acciones definidas en los *game loops* y para comprobar otros datos importantes como si necesitó los menús de ayuda o si encontró el secreto en el juego.

Se proporcionó a los usuarios hasta cuatro cuestionarios distintos para poder realizar la evaluación y que se detallarán y estudiarán:

- Cuestionario de información personal
- Cuestionario sobre impresiones generales del juego
- *Motivational Trait Questionnaire* (MTQ)
- *Bartle Test*

5.1.2 Información personal de los usuarios

A continuación, se muestran los datos recopilados sobre la información personal del usuario. Con este cuestionario se procura obtener los perfiles de cada jugador y analizar como estos pueden haber afectado a la jugabilidad. Se pidieron los siguientes datos: edad, frecuencia de juego a videojuegos, plataforma preferida, género de videojuego más jugado y tipo de juego preferido.

En la figura 5.1 se puede apreciar la edad de los usuarios que participaron en la evaluación. Casi el 90% de los participantes se encuentran en el rango entre los 19 y 22 años. Esto se debe al entorno en el que se ejecutó esta encuesta. Por ello, la mayoría pertenecen a una edad donde se suele estar en la etapa universitaria.

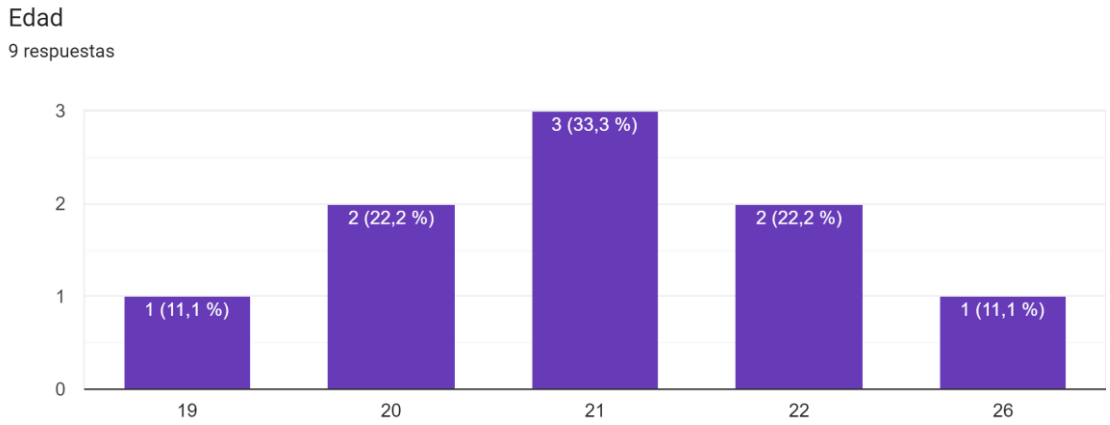

Figura 5.1: Gráfico con la edad de los usuarios participantes

Se puede observar la frecuencia de juego por parte de los jugadores encuestados en la ilustración 5.2. No se registró ninguna respuesta en las opciones de “nunca” y “raramente”, por lo que se puede concluir que la gran mayoría de usuarios estaban bastante familiarizados con los videojuegos. Esta condición suele hacer que los jugadores puedan adaptarse más fácilmente a las mecánicas del juego y pueden aportar una crítica y perspectiva más crítica basándose en sus experiencias en este ámbito. Además, estos usuarios pueden realizar acciones donde se detecten *bugs* ya que buscan el límite de las mecánicas y de lo permitido en el juego, facilitando esta labor.

Estos resultados se deben al contexto en el que se realizó la encuesta ya que al ser todos alumnos de la Facultad de Ingeniería Informática, se puede entender que la gran mayoría estén familiarizados con el mundo de los videojuegos.

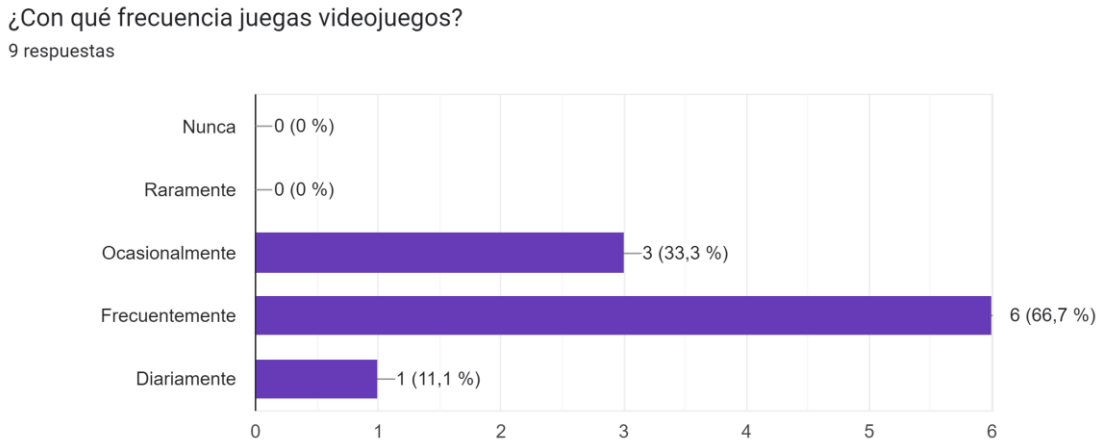

Figura 5.2: Gráfico con la frecuencia de juego de los usuarios participantes

En cuanto a la plataforma preferida por los encuestados para jugar, se permitió que los usuarios respondieran con varias opciones. En la imagen 5.3,

se puede atisbar que destaca el uso del ordenador como herramienta para jugar. Lo que significa que dos tercios de los usuarios que probaron el juego estaban familiarizados con el uso del ratón, periférico en el que está basado el juego. Por otro lado, las demás plataformas están bastante igualadas en preferencias de uso por los encuestados. Tan solo 2 de los 9 usuarios eligieron alguna de estas plataformas alternativas como su única preferencia.

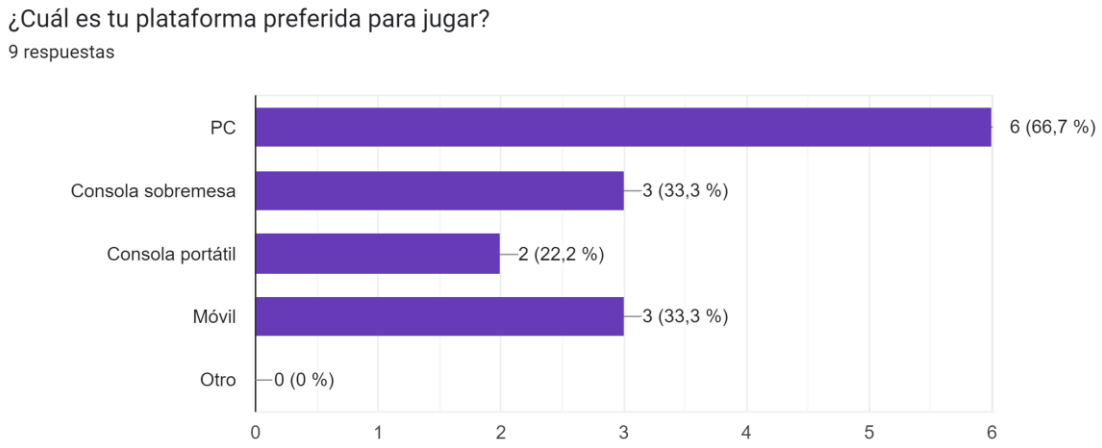

Figura 5.3: Gráfico con la plataforma preferida por los usuarios participantes

Como se puede ver en la imagen 5.4, no hay una categoría de juego que destaque entre las demás. Hay una ligera preferencia por los juegos de tipo individual, que es la categoría en la que se enmarca el producto evaluado, por lo tanto, esta también es una característica favorable para la evaluación. Hasta 7 de los 9 encuestados respondieron con alguno de los dos tipos de juego “multijugador”.

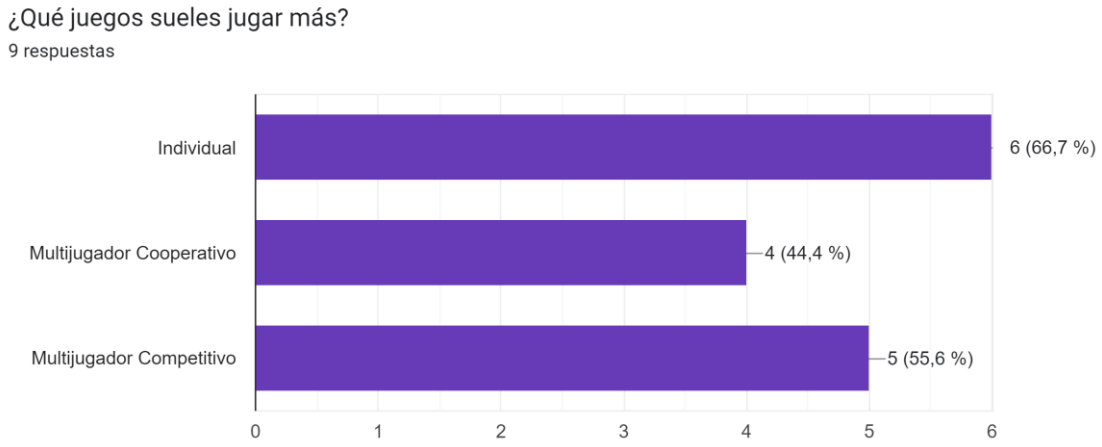

Figura 5.4: Gráfico con los tipos de juego más jugado por los encuestados

Por último, se preguntó a los encuestados por el género de videojuego que más disfrutan. Los resultados de esta pregunta se pueden observar en la imagen 5.5. Se permitió responder con varias opciones a los usuarios.

Tras analizar los resultados, se puede llegar a la conclusión de que los juegos más jugados por los encuestados pertenecen a los géneros Aventura, Estrategia y Plataformas. Los demás géneros están muy igualados entre sí y esto es beneficioso ya que se puede evaluar el producto con diferentes tipos de jugadores. La *Vertical Slice* desarrollada se puede enmarcar dentro del género de Aventuras.

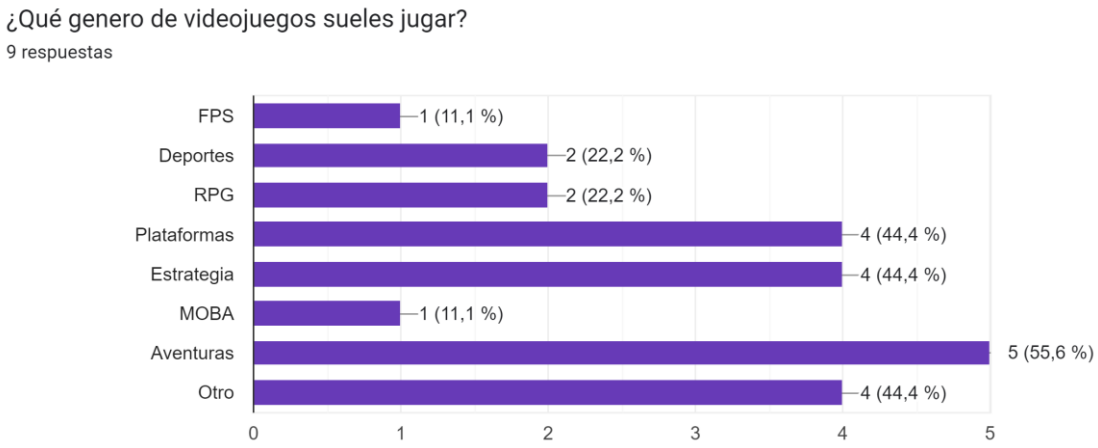

Figura 5.5: Gráfico con los géneros más jugados por los encuestados

5.2 Impresiones generales de los usuarios

Se les pidió a los usuarios rellenar una encuesta donde se les hacía diferentes preguntas para poder conocer su opinión sobre el juego probado. Las cuestiones iban dirigidas para obtener puntos fuertes y de mejora del prototipo. Se les preguntó acerca de qué les había parecido el juego en general y qué les había parecido más divertido. Por otro lado, se consultó sobre que les había costado más de entender y si les había resultado desafiante. Por último, se registraron respuestas sobre que cambiarían del juego, que emociones se han despertado en ellos y una nota del 1 al 10 del juego.

Las respuestas dadas serán analizadas a continuación para poder extraer conclusiones significativas que orienten el desarrollo y la optimización del producto.

¿Qué te ha parecido el juego en general?

Las sensaciones generales de los jugadores fueron las siguientes:

- La gran mayoría de los usuarios destacaron la inmersión durante la sesión de juego.
- Un usuario apuntó que le falta añadir algún apartado para mejorar la accesibilidad del juego.
- Falta de ayuda inicial: un usuario manifestó que le hubiese gustado tener algo más de contexto inicial ya que se sintió algo perdido al principio.

- Un jugador comentó que le motivaba el hecho de tener que relacionar las pistas conseguidas para resolver el misterio.

¿Qué te ha parecido más divertido?

Los usuarios destacaron los siguientes puntos sobre esta pregunta:

- Varios usuarios mencionaron que la ambientación del juego les gustó mucho ya que les aumentaba la curiosidad e intriga.
- Un usuario destacó la investigación de la nave como punto a favor en la diversión
- Tres usuarios comentaron que les pareció muy buena la narrativa del videojuego.
- Otro usuario también apuntó que buscar la correlación entre las pistas que habías encontrado daba satisfacción cuando se resolvía ya que era una meta a corto plazo.
- Para un jugador este juego no tenía nada de divertido, ya que no es el tipo de videojuego al que jugaría habitualmente.

¿Qué te ha costado más entender o no te ha gustado?

Estos son los principales aspectos que han sido más difíciles de entender para los usuarios:

- Movimiento de la cámara: un jugador destacó que la cámara le costó entenderla y que a veces no respondía como esperaba. Esto pudo ocurrir dado que había un *bug* al rotar la cámara cuándo se mantenía más de 3 segundos pulsados uno de los dos botones de rotar.
- Unión de pistas: Hasta tres jugadores no comprendieron correctamente cómo funcionaba esta función.
- Notificación de pistas: Otro usuario apuntó a que tuvo que revisar frecuentemente el menú de pistas porque no sabía que objetos le estaban proporcionando pistas.
- Información inicial: Otro encuestado destacó que hubo mucha información al principio del juego y que le resultó complicado recordar todo.
- Varios usuarios mencionaron que les hubiera gustado que hubiese una lista de objetivos en pantalla durante el juego o alguna clase de indicación que les ayude a saber que van por buen camino.
- Contraseña de la puerta: dos usuarios apuntaron que sería necesario que hubiese una indicación de que a la puerta bloqueada le falta por añadir una contraseña ya que esto no era intuitivo.
- *Outline:* Un usuario mencionó que el *bug* que había al mostrar el *outline* de los objetos dificultaba en algunos momentos la experiencia de juego.

¿Te ha resultado desafiante?

Para esta pregunta la gran mayoría respondió con un “sí” o con un “no”. Hasta 4 jugadores sintieron que el producto no era desafiante y 2 usuarios sí que les pareció desafiante. Los 3 encuestados restantes respondieron que necesitaban poder jugar más para poder responder a esta pregunta.

¿Qué cambiarías del juego?

Estos son los principales cambios que los jugadores plantearon para el juego:

- Un jugador propuso que la rotación de la cámara se hiciese con el ratón y que también se pudiese elegir caminar con las teclas “WASD”.
- Un jugador mencionó que añadiría un mapa de la nave para poder situarse mejor.
- Otro usuario pensó que el juego mejoraría si fuese en primera persona.
- Un encuestado mencionó que se debería haber hecho una mejor diferenciación del personaje que está hablando en cada momento.
- Un usuario señaló que añadiría una mejor visión de ciertos ángulos para detectar ciertos objetos.

¿Volverías a jugar al juego si se ampliase?

Casi todos los encuestados mencionaron que sí que volverían a jugar al juego si este fuese ampliado en mecánicas y escenarios.

¿Qué emociones ha despertado en ti?

La gran mayoría de usuarios mencionaron que les despertó curiosidad e intriga. Hubo un usuario que menciono que sintió sensación de atasco al no saber que hacer en algún momento. Otros usuarios destacaron que tenían motivación por explorar y descubrir que había pasado en la historia.

Dale una nota al juego del 1 al 10

Las notas de los usuarios se pueden observar en la ilustración 5.6. Todas están dentro de un intervalo entre 6 y 10. La nota media obtenida fue de 8 puntos sobre 10. Al preguntarles a los jugadores el por qué habían dado esa nota muchos mencionaron que, aunque el prototipo tiene margen de mejora, piensan que la idea tiene potencial para poder ser muy interesante.

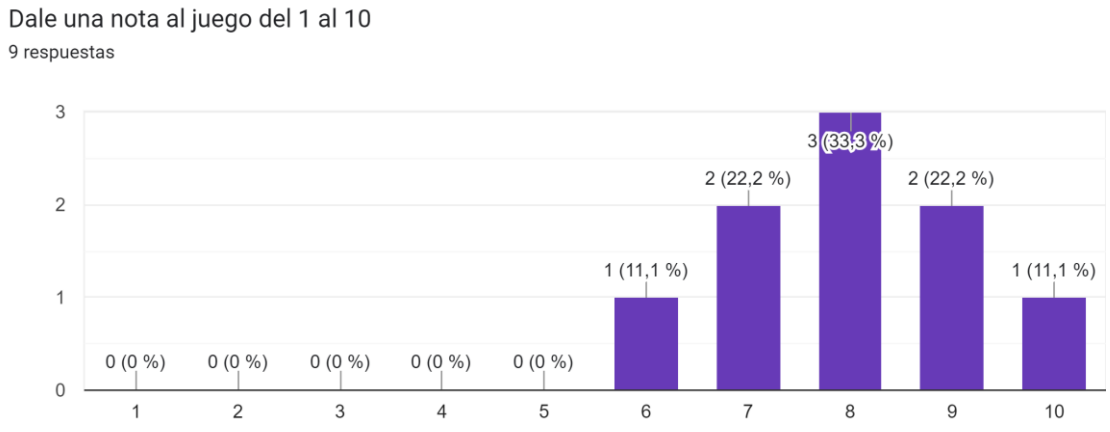

Figura 5.6: Gráfico con las notas dadas al prototipo por los encuestados

5.3 Comparación entre métricas obtenidas y los tests:

Tras realizar los tests de información personal del usuario y de impresiones generales del juego, se pidió a los jugadores que hiciesen el Bartle Test y el *Motivational Trait Questionnaire* (MTQ). Con esos resultados se podrán hacer comprobaciones con las métricas obtenidas por los juegos para poder ver su eficiencia. Para realizar este análisis fue necesario unir toda la información recabada entre los 3 proyectos para poder conseguir todos los datos completos de cada jugador.

5.3.1 Bartle Test

A continuación, se pueden apreciar en las figura 5.7, 5.8, unas tablas con lo obtenido por nuestros juegos y los resultados del Bartle Test realizado por los jugadores. En dichas tablas las siglas BT significará "Bartle Test" y BTMP "Bartle Test Métricas Propias".

Para sacar los resultados del Bartle Test se cuantificaron las respuestas dadas que se podían enmarcar más con una personalidad. Por lo tanto, de las 29 preguntas que tenía la prueba, se han obtenido los porcentajes en cada personalidad de cada jugador (sobre 100). Antes de realizar los análisis de los resultados, cabe destacar que hay que tomarlos con algo de cautela ya que en cada juego se hicieron sesiones de 15 minutos y, en muchos casos, los usuarios no pudieron acabarlos. Por lo tanto, algunas de las métricas que se evaluaban en la fase final de los prototipos no pudo ser medida de forma correcta ya que el usuario no tuvo oportunidad de cumplirla.

Id jugador	**Asesino BT**	**Socializador BT**	**Triunfador BT**	**Explorador BT**
1	0%	60%	73%	67%
2	87%	33%	33%	47%
3	33%	60%	20%	87%
4	20%	67%	47%	67%
5	0%	73%	53%	73%
6	20%	67%	40%	73%
7	47%	67%	27%	60%
8	73%	40%	20%	67%
9	40%	40%	53%	67%

Figura 5.7: Porcentajes para cada personalidad del jugador (sobre 100) obtenidos por el Bartle Test.

Id jugador	Asesino BTMP	Socializador BTMP	Triunfador BTMP	Explorador BTMP
1	0%	100%	0%	17,5%
2	0%	100%	18%	65%
3	0%	100%	18%	35%
4	67%	100%	24%	65%
5	0%	100%	18%	52,5%
6	0%	100%	0%	100%
7	0%	100%	18%	35%
8	78%	100%	46%	17,5%
9	100%	0%	54%	17,5%

Figura 5.8: Porcentajes para cada personalidad del jugador (sobre 100) obtenidos por las métricas medidas durante la sesión de juego.

Tras comparar los resultados obtenidos por las métricas medidas por nuestra cuenta, se pueden extraer las siguientes conclusiones:

- Es necesario ajustar las métricas medidas en el apartado de "Socializador" ya que la mayoría de los jugadores las completaron dejando lugar a unos resultados que pueden ser poco concluyentes. Esto puede ser debido a que la gran mayoría de los jugadores pensaron que para completar el juego era necesario realizar todos los diálogos con los NPCs, a pesar de que venía indicado que esto era opcional. La gran mayoría de jugadores evaluados tenían más de un 50% en este apartado según el Bartle Test.
- Habría que rediseñar las métricas medidas para la personalidad "Asesino" ya que los resultados han sido muy dispares entre sí. Hay una mayoría de jugadores que no completaron ninguna de las métricas medidas para este apartado, lo que indica que estas no funcionaron como se esperaba debido a que no se consiguió diferenciar a unos de otros a pesar de que algunos si se identifican más con este rasgo como se demuestra en los resultados del Bartle Test.
- En cuanto a la personalidad "Triunfador" se han alcanzado resultados algo mejores, aunque hay margen de mejora. En este caso, muchas de las métricas no pudieron ser capturadas satisfactoriamente por el hecho de que se evaluaban al final de los juegos.
- La gran mayoría de usuarios se encuentra con más del 50% de características de jugador tipo "Explorador". En este apartado, nuestras métricas tampoco están cerca de conseguir resultados cercanos al Bartle Test, aunque en algunos casos si está cerca.

En general, los resultados obtenidos no entran dentro de un error asumible y es necesario realizar varios ajustes para alcanzar unos resultados satisfactorios y más concluyentes. Habría que definir nuevas métricas que sean más específicas para cada personalidad y añadir nuevas mecánicas a los juegos que ayuden a esto.

5.3.2 MTQ

Se le dio a los encuestados un formulario con 82 preguntas donde se evaluaban sus rasgos motivacionales de acuerdo al *Motivational Trait Questionnaire* (MTQ). Para obtener los resultados de esta prueba se dividieron las preguntas en relación al rasgo motivacional al que hacían referencia. Por lo tanto, cada categoría estará medida sobre 100 y podría haber usuarios que obtenga valores altos en las cuatro categorías evaluadas o al revés. Esto viene muy bien para compararlo con las métricas medidas por nuestros juegos ya que nuestras mediciones también están realizadas calculando cada rasgo sobre 100.

Los porcentajes obtenidos por el cuestionario MTQ y por las métricas de nuestros juegos se pueden observar en las figuras 5.9 y 5.10. Para referirnos a las métricas medidas por nuestro juego se han empleado las siglas "MTQMP". Como ocurrió en el Bartle Test, cada sesión de juego fue de 15 minutos y en algunos casos no dio tiempo a que los usuarios acabasen. Por lo tanto, estos resultados no podrán ser todo lo precisos posibles y habrá que tomarlos con cautela.

Id jugador	Determinación MTQ	Deseo de aprender MTQ	Evitación de fallos MTQ	Masterización MTQ
1	97,6%	72,9%	85,2%	83,3%
2	78,6%	66,7%	50%	85,7%
3	64,3%	68,8%	59,3%	66,7%
4	50%	56,2%	40,7%	42,9%
5	52,4%	52,1%	53,7%	64,3%
6	76,2%	45,83%	50%	66,7%
7	78,6%	50%	18,5%	66,7%
8	90,5%	39,6%	18,5%	83,3%
9	81%	72,9%	57,4%	71,4%

Figura 5.9: Porcentajes para cada rasgo motivacional de cada usuario obtenido por el MTQ.

Id jugador	Determinación MTQMP	Deseo de aprender MTQMP	Evitación de fallos MTQMP	Masterización MTQMP
1	55%	15%	5%	0%
2	65%	45%	35%	0%
3	60%	40%	65%	50%
4	65%	40%	5%	20%
5	75%	45%	15%	0%
6	55%	55%	0%	0%
7	45%	25%	0%	50%

8	70%	15%	0%	70%
9	85%	40%	5%	30%

Figura 5.10: Porcentajes para cada rasgo motivacional de cada usuario obtenido por las métricas propias.

- Hay algunos perfiles motivacionales como los de los usuarios 3 y 8 que no están muy lejos de lo medido por el cuestionario MTQ.
- Nuestras métricas han capturado el rasgo motivacional más y menos identificativo del jugador en varios usuarios.
- Varios jugadores han obtenido un 0% en el apartado de "Masterización" con nuestras métricas a pesar de que son usuarios con un alto porcentaje en este aspecto.
- En relación al rasgo motivacional "Evitación de Fallos" no se han conseguido unos resultados lo suficientemente buenos ya que hay jugadores que en el MTQ tienen un alto valor en este aspecto y sin embargo en nuestras métricas han alcanzado un número muy bajo.
- En el rasgo de "Determinación" se han conseguido resultados aceptables ya que se ha conseguido medir decentemente este aspecto por nuestras métricas en comparación con lo determinado por el MTQ.

5.4 Auditoría de accesibilidad

Para poder evaluar la accesibilidad del prototipo se ha decidido realizar una auditoría de accesibilidad de 13 medidas del *Libro Blanco de Accesibilidad para Desarrolladores de Videojuegos* [41] para analizar mejor estos aspectos de cara a futuras implementaciones. Para valorar estas medidas se utilizaron tres valores posibles: "Bien", "Mal", "No aplicable".

Medida	**Valor**	**Observaciones**
1.Audio 3D	No aplicable	No hay elementos auditivos durante el juego.
7.Estamina	No aplicable	No existe un sistema de vidas o resistencia en este juego.
13.Modificaciones	Mal	No se observa diferencia de acciones por la manera de interactuar con un mismo control ya que cada acción tendrá asociado un botón diferente.
16.Omitir Mecánicas	Mal	El prototipo no tiene la opción de omitir mecánicas.
21.Recordatorios contextuales	Bien	Al lado de cada acción que no sea controlada por el ratón se expone la tecla que se debe utilizar para realizar la acción.

25.Sensibilidad de movimiento y de la cámara	Mal	No se permite modificar la velocidad de rotación de la cámara.
27.Volúmenes	No aplicable	No hay elementos auditivos durante el juego.
29.Anticaídas	Bien	El espacio de juego está delimitado por *colliders* que hacen que no se pueda sufrir una caída.
34.Información con códigos de color	No aplicable	El prototipo no hace uso del color como fuente de información para lograr los objetivos propuestos en el juego.
38.Personalización de colores	Mal	No se puede modificar la paleta de colores utilizada en el juego.
40. Posicionamiento de la cámara	Mal	No se puede cambiar el posicionamiento de la cámara ya que está sigue al jugador permanentemente y no está implementada la opción de cambiar este aspecto.
41. Realización de acciones automáticas	Mal	No hay alternativas en el juego para activar la realización de acciones automáticas.
43.Sensibilidad al apuntar	Mal	No se puede cambiar la sensibilidad del apuntado del ratón.

Figura 5.11: Auditoría de accesibilidad

6 Discusión y líneas futuras

Después de realizar una minuciosa observación y un análisis durante la evaluación del producto se han identificado diferentes problemas y se darán soluciones adecuadas orientadas a siguientes áreas de desarrollo. También se expondrán vías de mejora para obtener resultados más precisos a la hora de generar el perfil motivacional y de personalidad de jugador de los usuarios.

1) **Rotación de la cámara:** a lo largo de la evaluación, hubo un claro problema con la rotación de la cámara ya que esta no se comportaba correctamente. Además, algún usuario mencionó que le hubiera resultado más sencillo e intuitivo realizar esta acción con el clic derecho del ratón.
    - Para solucionar este aspecto se deberá realizar ajustes en la clase que controla la cámara y habrá que implementar y usar un *InputAction* que capte el uso del clic derecho además de implementar la nueva lógica de la rotación con el uso de este nuevo control.

2) ***Outline***: todos los usuarios necesitaron ayuda durante la sesión de juego para identificar que objetos eran interactivos y cuáles no. Esto se debió a que el *outline* no funcionó bien y en muchas ocasiones no marcaba el contorno del elemento a pesar de que el cursor estaba apuntándolo. Esto provocó confusión en los jugadores y empeoró la experiencia de usuario.
    - La solución a este error reside en el uso correcto de *layers*. En la clase que controla el *outline* no se ignoraba a determinados *layers* al lanzar el *raycast*. Esto provocaba que, aunque los usuarios observasen que estaban apuntando al objeto, había una pared o elemento que estaba interfiriendo y provocando que el "rayo" lanzado no llegase a impactar con el objeto deseado.

3) **Uso de notificaciones:** al observar a los jugadores se notó que todos no vieron las notificaciones que aparecían en la zona superior derecha al recoger las pistas. Esto era debido a que estas notificaciones aparecían a la vez que el usuario interactuaba con los diálogos. Por lo tanto, el campo de visión de los jugadores estaba en la zona inferior central de la pantalla y era muy difícil apreciar el aviso dado. Para arreglar este problema:
    - Se puede retrasar la notificación hasta que el usuario haya dejado de estar en el diálogo y de esta manera se evitaría que esté la atención del jugador en ese aspecto cuando salga el aviso.
    - Modificar la posición de la notificación a un lugar más visible y llamativo es otra opción que se podría usar para solucionar este hecho.

4) **Demasiadas indicaciones iniciales:** varios jugadores mencionaron durante la evaluación que percibían que se les había dado demasiada información al principio del juego y que era complicado retener todas estas indicaciones iniciales. La consecuencia de esto es que los jugadores podían tener dificultades para seguir adelante al no recordar ciertos aspectos. Surgen varias formas posibles de resolver esta incidencia:
    - Hacer uso de recordatorios contextuales que vayan recordando a lo largo del juego aspectos críticos para la jugabilidad.
    - Enseñar al jugador a interactuar con el juego paulatinamente a medida que va a avanzando.
    - Dar la oportunidad al jugador de volver a revisar el tutorial.

5) **Explicación de la mecánica de pistas:** durante la evaluación se observó que algunos jugadores manifestaron que no habían entendido bien la mecánica de unión de pistas y que resultaba algo confusa. Se plantea la siguiente solución:
    - Guiar al jugador durante la obtención de la primera deducción, haciendo así que el usuario pruebe e interactúe con este sistema teniendo una referencia al principio.

6) **Lista de objetivos en pantalla:** varios usuarios echaron en falta que hubiese algún tipo de identificación para que fuese más intuitivo para ellos. Para mejorar este aspecto se propone lo siguiente:
    - Añadir un nuevo elemento al HUD del juego donde se indique los objetivos a realizar y cuántas pistas faltan por recabar a lo largo del escenario, de esta manera, los jugadores sabrían qué deben hacer.

7) **Conclusión del juego sin llegar a todas las deducciones:** se observó que hubo dos jugadores que terminaron el juego sin haber llegado a las cuatro deducciones que había en total para conseguir. Con llegar a la "deducción4" donde se utilizaba la pista de la sala de seguridad se podía llegar a la opción de terminar el juego. Este comportamiento no era el esperado. Por ello se propone la siguiente solución:
    - Cambiar la comprobación de la activación de la deducción final. Se optó por verificar únicamente que la "deduccion4" era activada; ya que, a priori, era la última deducción que se debía alcanzar por el orden en el que estaban establecidas las pistas. Sin embargo, como se dio libertad al jugador a la hora de moverse por la nave, salvo para entrar en la sala de seguridad, este enfoque no era el correcto. Por lo tanto, habrá que comprobar que las cuatro deducciones posibles han sido descubiertas y hasta que no estén todas activadas, no se podrá acceder a la deducción final.

8) **Mala visión de objetos:** se detectó que había una mala visión en algunos objetos difíciles que eran complicados de observar. A pesar de la rotación de la cámara, esto resulto un problema para muchos jugadores. Esto fue debido a que, en algunos casos, había objetos como puertas o paredes que por la altura y ángulo de la cámara no permitían una visión directa del objeto. Para la resolución de este problema se plantea lo siguiente:
    - Rediseñar las zonas del escenario que tienen este problema como la sala que hay dentro del puente de mando. Cuando se abre la puerta donde están los científicos, la puerta tapa a estos NPCs y puede resultar complicado interactuar con ellos. Por ello, habría que cambiar estas partes del mapa que provocan este problema.

Tras haber desarrollado las resoluciones a los problemas detectados, se expondrán los siguientes aspectos para mejorar el juego en próximas líneas de desarrollo.

1. Se estudiará y analizará qué métricas han funcionado mejor y cuáles peor a la hora de detectar los patrones motivacionales y la personalidad del jugador. Tras ello, se buscarán nuevas métricas que mejoren los resultados obtenidos. Para este propósito, se podrían añadir nuevas mecánicas a los juegos que permitan una mejor captura.
2. Para poder detectar los patrones motivacionales de "Búsqueda de Competición" y "Otros Objetivos" se propone introducir mecánicas o elementos que puedan efectuar las mediciones de estos rasgos lo más preciso posible.
3. Se plantea añadir los elementos necesarios para que el propio juego sea capaz de medir estas métricas para el MTQ y el Bartle Test y que no sea necesario capturarlas a mano.
4. Se considera añadir al juego ciertas mecánicas que fueron pensadas en un principio, pero que no fueron implementadas por falta de tiempo. Entre estas se incluyen, por ejemplo, que haya un inventario de objetos donde se puedan examinar ciertos objetos o haya algún consumible que otorgue bonificadores al jugador. Por otro lado, también se pensó en introducir un diario de misiones donde viniesen recogidas las misiones que el jugador tenía activadas y los objetivos de estas. Estas mecánicas ayudarían para darle más contexto al jugador y para aportar nuevas maneras de resolver los casos.
5. También se sugiere hacer más niveles con nuevos escenarios, personajes y casos por resolver que aumenten la riqueza del juego.
6. Se propone añadir mecánicas en las que el usuario tenga que interactuar con el entorno y seguir un rastro, como seguir unas huellas.
7. Se plantea la implementación de unas estadísticas RPG al juego para que cada jugador pueda elegir su forma de jugar y de enfocar la resolución de un problema. También a la hora de dialogar, dependiendo de las estadísticas que tuviese el jugador, tendrá unos diálogos desbloqueados y otros bloqueados.
8. Se considera mejorar en cuestión de accesibilidad ya que más de la mitad de las medidas evaluadas durante la auditoría no están siendo cubiertas.

7 Conclusiones

Tras la finalización del proyecto, se han conseguido cumplir con éxito los objetivos propuestos al inicio:

- Se ha implementado una versión jugable del producto en estado Alfa de acuerdo al diseño planteado.
- Se ha realizado el documento *GDD* donde se recogen todas las decisiones tomadas de diseño a lo largo del proyecto.
- Se ha definido un perfil motivacional y se ha clasificado a los jugadores en base a sus acciones en los prototipos jugables.

La medición de los rasgos motivacionales y la personalidad del jugador en tres juegos, que no están dirigidos expresamente para satisfacer esta labor, ha resultado ser un buen primer paso para conseguir que, en un futuro, se pueda obtener dicha medición con una mayor precisión. Se necesita hacer un rediseño de las métricas medidas y añadir algunas nuevas para poder realizar mediciones más completas. Asimismo, se considera que si se llegase a implementar una versión ampliada de estos productos y evaluar a bastantes usuarios se podrían conseguir resultados más concluyentes en este aspecto.

Realizar una implementación del producto en cuatro iteraciones distintas ayudó mucho a poner el foco en las mecánicas esenciales al principio y llevar un desarrollo más estructurado y eficiente. Por lo tanto, aplicar esta metodología fue un acierto para la consecución de los objetivos propuestos. Asimismo, la utilización de la plataforma *GitHub* para realizar el control de versiones fue muy remarcable ya que facilitó la gestión de las implementaciones realizadas.

Aunque se tenía cierta experiencia en el desarrollo de videojuegos, el aprendizaje continuo durante la implementación ha sido esencial para hacer frente a todas las dificultades que fueron apareciendo. En relación a esto, ha resultado muy beneficioso la gran cantidad de tutoriales y de recursos, relacionados con Unity, que existen y que han facilitado mucho este proceso.

Debido a que fue complicada la implementación de algunas mecánicas, como la unión de pistas, por la aparición de *bugs,* se tuvo que descartar la adición de alguna mecánica más que hubiesen hecho más completa la experiencia de usuario. Además, la aparición de *bugs* como los del *outline* y la rotación de la cámara en la evaluación de usuario provocó que la experiencia de juego no fuera del todo satisfactoria, por lo que hacer desaparecer este tipo de problemas será esencial de cara a futuras evaluaciones.

8 Bibliografía

[1] T. M. Fleming, L. Bavin, K. Stasiak, E. Hermansson-Webb, S. N.Merry, C. Cheek, M. Lucassen, H. M. Lau, B. Pollmuller y S. Hetrick, «Serious Games and Gamification for Mental Health: Current Status and Promising Directions,» *Frontiers,* vol. 7, 2017.

[2] D. Fernández-Avilés, A. d. Antonio y E. Villalba-Mora, «Motivational traits: An objective behavorial test using a computer game,» *Frontiers,* 2022.

[3] E. D. Heggestad y R. Kanfer, «Individual differences in trait motivation: Development of the Motivational Trait Questionnaire,» *International Journal of Educational Research,* 2000.

[4] R. Bartle, «HEARTS, CLUBS, DIAMONDS, SPADES: PLAYERS WHO SUIT MUDS,» 1996.

[5] S. Aleem, L. F. Capretz y F. Ahmed, «Game development software engineering process life cycle: a systematic review,» *Journal of Software Engineering Research and Development,* 2016.

[6] C. Keith, Agile Game Development with Scrum, 2010.

[7] «Metodologías de desarrollo de software: ¿Qué son y para qué sirven?,» Valtx, [En línea]. Available: https://www.valtx.pe/blog/metodologias-para-el-desarrollo-de-software-que-son-y-para-que-sirven. [Último acceso: 16 Abril 2024].

[8] J. Seger y R. Potts, «Personality Correlates of Psychological Flow States in Videogame Play,» *Current Psychology,* vol. 31, pp. 103-121, 2012.

[9] J. Feldman, A. Monteserín y A. Amandi, «Exploring the use of online video games to detect personality dichotomies,» *Online Information Review,* 2017.

[10] R. A. Tasnim y F. Z. Eishita, «Analyzing the Distinctive Impact of Personality Traits on Serious Gameplay Experience,» de *IEEE 9th International Conference on Serious Games and Applications for Health(SeGAH),* 2021.

[11] Wikipedia, «Taxonomía de Bartle,» [En línea]. Available: https://es.wikipedia.org/wiki/Taxonomía_de_Bartle. [Último acceso: 20 Mayo 2024].

[12] F. Williamson, «Medium,» [En línea]. Available: https://medium.com/@williamson.f93/multi-user-dungeons-muds-what-are-they-and-how-to-play-af3ec0f29f4a. [Último acceso: 20 Mayo 2024].

[13] X. Gaya, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Taxonomía_de_Bartle#/media/Archivo:Taxonomía_de_Bartle.png. [Último acceso: 20 Mayo 2024].

[14] C. Politowski, F. Petrillo, J. E. Montandon, M. T. Valente y Y.-G. Guéhéneuc, «Are game engines software frameworks? A three-perspective study,» *Journal of Systems and Software,* vol. 171, 2021.

[15] C. Vohera, H. Chheda, D. Chouhan, A. Desai y V. Jain, «Game Engine Architecture and Comparative Study of Different Game Engines,» 2021.

[16] «What is a Gaming or Game Engine,» Arm, [En línea]. Available: https://www.arm.com/glossary/gaming-engines. [Último acceso: 2024 abril 18].

[17] «Motor físico,» Wikipedia, [En línea]. Available: https://es.wikipedia.org/wiki/Motor_f%C3%ADsico. [Último acceso: 2024 abril 18].

[18] K. Balasubramian, «Game Engines: All You Need to Know,» Gameopedia, [En línea]. Available: https://www.gameopedia.com/game-engines-all-you-need-to-know-about/. [Último acceso: 19 abril 2024].

[19] A.Andrade, «Game engines: a survey,» *EAI Endorsed Transactions on Serious Gaming,* 2015.

[20] Unity, «UnIty,» [En línea]. Available: https://unity.com/es/use-cases. [Último acceso: 25 Mayo 2024].

[21] Unreal Engine, «Unreal Engine,» [En línea]. Available: https://www.unrealengine.com/es-ES/blog/unreal-engine-5-is-now-available. [Último acceso: 25 Mayo 2024].

[22] A. H. Sutopo, Developing Games with GameMaker Studio.

[23] Steam, «Steam,» [En línea]. Available: https://store.steampowered.com/app/1670460/GameMaker/?l=latam. [Último acceso: 25 Mayo 2024].

[24] B. Andrzej y W. Hubert, «Comparative Study on Game Engines,» Vols. %1 de %21-2, 2019.

[25] 80 Level, «80.lv,» [En línea]. Available: https://80.lv/partners/cryengine/. [Último acceso: 25 Mayo 2024].

[26] M. Ranaweera y Q. H. Mahmoud, «Deep Reinforcement Learning with Godot Game Engine,» *MDPI,* vol. 13, 2024.

[27] J. Linietsky y F. M. Calabró, «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Godot_%28game_engine%29#/media/File:Godot3.4.png. [Último acceso: 25 Mayo 2024].

[28] Nintendo, «Nintendo,» [En línea]. Available: https://www.nintendo.com/es-es/Juegos/Juegos-de-Nintendo-Switch/The-Legend-of-Zelda-Tears-of-the-Kingdom-1576884.html. [Último acceso: 25 Mayo 2024].

[29] A. Bell, «RockPaperShotgun,» [En línea]. Available: https://www.rockpapershotgun.com/sherlock-holmes-chapter-one-review. [Último acceso: 25 Mayo 2024].

[30] H. Hawes, «WaytoomanyGames,» [En línea]. Available: https://waytoomany.games/2022/09/19/review-return-to-monkey-island/. [Último acceso: 25 Mayo 2024].

[31] Unity, «Unity Documentation,» [En línea]. Available: https://docs.unity3d.com/es/2019.4/Manual/nav-BuildingNavMesh.html. [Último acceso: 5 Mayo 2024].

[32] Unity, «Unity Documentation,» [En línea]. Available: https://docs.unity3d.com/Manual/class-ScriptableObject.html. [Último acceso: 5 Mayo 2024].

[33] A. Casero, «Keepcoding,» [En línea]. Available: https://keepcoding.io/blog/el-patron-singleton/. [Último acceso: 7 Mayo 2024].

[34] Unity , «Unity Documentation,» [En línea]. Available: https://docs.unity3d.com/es/2019.4/Manual/UICanvas.html. [Último acceso: 10 Mayo 2024].

[35] Unity, «Unity Documentation,» [En línea]. Available: https://docs.unity3d.com/2018.3/Documentation/ScriptReference/EventSystems.IPointerClickHandler.html. [Último acceso: 16 Mayo 2024].

[36] Unity, «Unity Documentation,» [En línea]. Available: https://docs.unity3d.com/2019.1/Documentation/ScriptReference/EventSystems.IPointerEnterHandler.html. [Último acceso: 16 Mayo 2024].

[37] Unity, «Unity Documentation,» [En línea]. Available: https://docs.unity3d.com/es/530/ScriptReference/EventSystems.IPointerExitHandler.html. [Último acceso: 16 Mayo 2024].

[38] Unity, «Unity Documentation,» [En línea]. Available: https://docs.unity3d.com/2018.1/Documentation/ScriptReference/UI.GridLayoutGroup.html. [Último acceso: 16 Mayo 2024].

[39] Unity, «Unity Documentation,» [En línea]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnMouseDown.html. [Último acceso: 17 Mayo 2024].

[40] Unity, «Unity Documentation,» [En línea]. Available: https://docs.unity3d.com/es/530/ScriptReference/Collider.OnTriggerEnter.html. [Último acceso: 18 Mayo 2024].

[41] A. Monedero, M. F. Simón, J. A. Delgado y E. G. Cortés, Libro Blanco de Accesibilidad para Desarrolladores de Videojuegos, AEVI y Fundación ONCE.

9 Anexos

ANEXO A: Observaciones de la evaluación de experiencia de usuario

Nº Usuario:	Fecha: 16-05-24		Hora: 16:04	
1	Datos de interés: Diálogos: IIIIII Inspeccionar: II		No evita fallos: III	
Tiempo empleado:	**Completa Deduccion1** (Sí) No	**Completa Deduccion2** (Sí) No	**Completa deduccion3** (Sí) No	**Completa Deduccion4** (Sí) No
15:00	**Completa Keyboard** (Sí) ayuda No	**Consigue todas las pistas** (Sí) No	**Encuentra el secreto** Sí (No)	**Usa menú ayuda pistas** (Sí) No
				Usa menú ayuda deducciones Sí (No)

- Usa ~~ayuda~~ ayuda pistas → salto sin querer "¿algo mas que deba saber?"
- Va a la sala puente de mando
- Lee detenidamente diálogo
- No se da cuenta notis.
- Mira deducción tras unir pistas
- Entiende el sistema pistas
- No se da cuenta hibernación → tuve que ayudar
- Reacciona a diálogos
- Ayuda sala seguridad
- Interactua bien con objetos → ~~aunque~~ c
- No llega al final, aunque ~~esta~~ alcanza deducción final
- Se olvida rotar

ORDEN

- Puente de mando (salta NPCs)
- Sala común
- Dependencias del Capitán
- Sala de seguridad (recordar contraseña)

Nº Usuario:	Fecha: 16-05-24		Hora: 15:45	
2	Datos de interés: Dialogos: III III Inspeccionar: IIIII		No evita fallos: III	
Tiempo empleado:	**Completa Deduccion1** (Sí) No	**Completa Deduccion2** (Sí) No	**Completa deduccion3** (Sí) No	**Completa Deduccion4** Sí (No)
15:00	**Completa Keyboard** Sí (No)	**Consigue todas las pistas** (Sí) No	**Encuentra el secreto** (Sí) No	**Usa menú ayuda pistas** (Sí) No
				Usa menú ayuda deducciones Sí (No)

- Intenta moverse con tutorial
- Clicka al principio probando ↯
- Pulsa sin querer en los dialogos
- Bug cuando ~~ju~~ utilizas W y E a la vez
- Doble dialogo - puente de mando
- Objeto terminal - no funciona → Sala puente de mando
- Se olvida de hablar con un científico
- No entiende sistema de pistas
- Lee los dialogos detenidamente
- Colliders mal y problema layers
- Interactua con casi todo
- Tira deducciones a voleo, ¿explicar mejor?
- Da muchos clicks
- No le ha dado tiempo a terminar
- Problema outline y dificulta mucho el juego → Necesita ayuda para interactuar
- No accedió a la ~~ter~~ interfaz cuando se explica en el tutorial.

Orden

- Puente de mando → Dependencias capitan
- Sala común → Sala Seguridad (falta tiempo)

Nº Usuario:	Fecha: 16 - 05 - 24		Hora: 15:17	
3	Datos de interés: Dialogos opcionales: HTT I Inspeccionar: HHT		No Evita fallos: III	
Tiempo empleado:	**Completa Deduccion1** Sí (No)	**Completa Deduccion2** (Sí) No	**Completa deduccion3** Sí (No)	**Completa Deduccion4** (Sí) No
15:00	**Completa Keyboard** (Sí) No	**Consigue todas las pistas** Sí (No)	**Encuentra el secreto** (Sí) ✗ (No)	**Usa menú ayuda pistas** Sí (No)
				Usa menú ayuda deducciones Sí (No)

- Se mete en el menú, tras primer collider → sigue tutorial
- Bug Puerta hibernación, seguridad → arreglar
- Va primero a la sala de mandos y luego a sala común
- No se da cuenta de las notificaciones, o no reacciona ante ellas
- Dialogo sala de seguridad necesita de entrar en puente de mando
- Llega bien a ver los objetos del puente de mando cuando vuelve
- Consigue entender facilmente el sistema de pistas y deducciones
- Bug deducción final
- Ha llegado casi al final, aunque han faltado algunos objetos

Orden

- Puente de mando
- Sala Común
- Dependencias del Capitán
- Sala seguridad → (falta de tiempo)

Nº Usuario:	Fecha: 16-05-24		Hora:	
4	Datos de interés: Diálogos: IIIII Inspecciones: IIIII		No evita fallos: III	
Tiempo empleado:	**Completa Deduccion1** Sí (No)	**Completa Deduccion2** Sí (No)	**Completa deduccion3** Sí (No)	**Completa Deduccion4** Sí (No)
~~15~~ 15:00	**Completa Keyboard** Sí — (No)	**Consigue todas las pistas** Sí (No)	**Encuentra el secreto** (Sí) No	**Usa menú ayuda pistas** Sí (No)
				Usa menú ayuda deducciones Sí (No)

- "¿Algo más que deba saber?" → activado
- Demasiada info al principio para
- Entiende la Q en el menú.
- Va a la derecha y activa silla y camas
- Outline no funciona → buscar el motivo → Necesita el usuario ayuda
- Llega a la habitación y consigue interactuar con todo
- Da menos clicks y funciona mejor el juego, que en anteriores análisis
- No llega al final → busca interactuar con todo
- Notificaciones → no las ha visto → arreglar e
- ~~A~~

ORDEN

- Sala común
- Dependencias del capitán
- Puente de mando
- Sala seguridad

Nº Usuario:	Fecha: 16-05-24		Hora: 17:15	
5	Datos de interés: Diálogos: ||||| Inspecciones: |||||		No evita fallos: |	
Tiempo empleado:	**Completa Deduccion1** Sí (No)	**Completa Deduccion2** Sí (No)	**Completa deduccion3** Sí (No)	**Completa Deduccion4** Sí (No)
15:00	**Completa Keyboard** Sí (No)	**Consigue todas las pistas** Sí (No)	**Encuentra el secreto** Sí (No)	**Usa menú ayuda pistas** (Sí) No
				Usa menú ayuda deducciones Sí (No)

- Piensa que los brillos son cosas que interactuar.
- Outline Bug
- Llega a sala de seguridad sin pasar por puente de mando
- Desactivar outlines
- No se da cuenta de notificaciones
- No se ~~da~~ acuerda de rotar → recordatorio contextual
- Utiliza el panel de ayuda tras sugerirlo pistas
- Encuentra pistas, pero no llega a interactuar con pistas en interfaz.

ORDEN

- Sala Común
- Dependencias del capitán
- Sala de seguridad
- Puente de mando (sin tiempo)

Nº Usuario:	Fecha: 16-05-24		Hora: 16:53	
§6	Datos de interés: Dialogos; Inspecciones: ||||||		No Evita fallos: ||||	
Tiempo empleado:	**Completa Deduccion1** Sí (No)	**Completa Deduccion2** Sí (No)	**Completa deduccion3** (Sí) No	**Completa Deduccion4** Sí No
15:00	**Completa Keyboard** Sí (No)	**Consigue todas las pistas** Sí (No)	**Encuentra el secreto** (Sí) No	**Usa menú ayuda pistas** Sí (No)
				Usa menú ayuda deducciones Sí (No)

- ¿Algo más que deba saber? → activado
- Prueba movimiento, en sala inicial
- Observa keyboard
- Funciona mejor con pocos clicks
- Notificaciones en pistas → Problema visibilidad
- Se da cuenta de la Q
- Se da cuenta feedback de interfaz pistas → tras mencionarselo
- No rota la cámara
- Interactua con todo en puente de mando

ORDEN

- Sala Común
- Dependencias del capitán
- Puente de mando
- Sala de seguridad (falta tiempo)

Nº Usuario:	Fecha: 16-5-24		Hora: 17:40	
7	Datos de interés: Dialogos: ||||| Inspeccionar: |||			No evita fallos: ||
Tiempo empleado:	**Completa Deduccion1** (Sí) No	**Completa Deduccion2** (Sí) No	**Completa deduccion3** (Sí) No	**Completa Deduccion4** Sí (No)
15:00	**Completa Keyboard** (Sí) No	**Consigue todas las pistas** Sí (No)	**Encuentra el secreto** Sí (No)	**Usa menú ayuda pistas** Sí (No)
				Usa menú ayuda deducciones Sí (No)

- ¿Algo más que deba saber? ⇒ activado
- Se queda observando primeros pasillos.
- Va primero a puente de mando
- No ve las notificaciones → Se focalizan en dialogos.
- Entra en sala común → Interactua con camas primero → Acaba viendo las 3 pistas
- Llega a camarote → va al escritorio
- Bug outline → ayuda para ¡Interactuar
- ~~Usuario piensa~~
- Entiende deduccion3, Entiende deducción 2 y la 1
- Saca deducción 1
- No llega al final del juego por falta de tiempo. (completa keyboard)

ORDEN

- Puente de mando
- Sala común
- Dependencias del capitán
- Sala seguridad

Nº Usuario:	Fecha: 16-05-24		Hora: 18:03	
8 [redacted]	Datos de interés: Dialogos: ////// Inspeccionar: //			No evitar fallos: /
Tiempo empleado:	**Completa Deduccion1** (Sí) No	**Completa Deduccion2** Sí (No)	**Completa deduccion3** (Sí) No	**Completa Deduccion4** (Sí) No
13:00	**Completa Keyboard** (Sí) No	**Consigue todas las pistas** Sí (No)	**Encuentra el secreto** Sí (No)	**Usa menú ayuda pistas** Sí (No)
				Usa menú ayuda deducciones Sí (No)

- ¿Algo más que deba saber? Activado
- Dialogo missclick → cambiar posición
- Llega a sala común → interactua con las camas, → se va a camarote
- Se da cuenta de la contraseña → completa Keyboard
- Missclick silla → falla en encontrar pistas en la mesa
- Se salta puente de mando → va directo a sala seguridad !!
- Interactua con los 3 proyectores
- Se pasa el juego → sin obtener pista mesa !!
-

ORDEN

- Sala Común
- Dependencias del capitán
- Sala seguridad
- Puente de mando

Nº Usuario:	Fecha: 16-05-24		Hora: 18:22	
9	Datos de interés: Dialogos: Inspeccionar: !!!			No evita fallos. !!!
Tiempo empleado: 13:30 ~~15~~	**Completa Deduccion1** Sí (No)	**Completa Deduccion2** (Sí) No	**Completa deduccion3** Sí (No)	**Completa Deduccion4** (Sí) No
	Completa Keyboard (Sí) No	**Consigue todas las pistas** Sí (No)	**Encuentra el secreto** Sí (No)	**Usa menú ayuda pistas** Sí (No)
				Usa menú ayuda deducciones Sí (No)

- Prueba rotación en tutorial
- ~~Intenta~~.
- Va a sala común → interactua mesa → sigue a camarote
- Outline problema → necesita ayuda:
- Recordatorio de los libros. → Ayuda para el usuario
- Va a sala de seguridad, tras descubrir en camarote la contraseña
- Prueba a unir pistas pero → he tenido que ayudar para recordar interfaz
- Acaba el juego sin conseguir todas las deducciones.
  ↳ comportamiento erróneo
- No consigue todas las pistas.
- Prueba deducciones al unir

ORDEN

- Sala Común
- Dependencias del Capitán
- Sala seguridad
- Puente de mando

Anexo B: Informe de Originalidad

turnitin

Recibo digital

Este recibo confirma quesu trabajo ha sido recibido por Turnitin. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

Autor de la entrega:	VICTOR VALDEOLMOS RABADAN
Título del ejercicio:	Turnitin Memoria Final (Moodle PP)
Título de la entrega:	Desarrollo de un Producto Software Interactivo Basado en M...
Nombre del archivo:	4841_VICTOR_VALDEOLMOS_RABADAN_Desarrollo_de_un_Pro...
Tamaño del archivo:	5.04M
Total páginas:	93
Total de palabras:	21,955
Total de caracteres:	117,340
Fecha de entrega:	03-jun.-2024 09:40p. m. (UTC+0200)
Identificador de la entre...	2394827620

Desarrollo de un Producto Software Interactivo Basado en Máquinas de Point-and-Click y Aventura Gráfica

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Mon Jun 03 23:19:35 CEST 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)