Agradecimientos.

Índice general

I Introducción **10**

1. Introducción **11**

2. Definición del problema **13**

2.1. Identificación del problema real 13

2.2. Identificación del problema técnico 14

2.2.1. Funcionamiento 14

2.2.2. Entorno 14

2.2.3. Vida esperada 15

2.2.4. Ciclo de mantenimiento 15

2.2.5. Competencia 15

2.2.6. Aspecto externo 15

2.2.7. Estandarización 15

2.2.8. Calidad y fiabilidad 16

2.2.9. Programación de tareas 16

2.2.10. Pruebas 17

2.2.11. Seguridad 17

3. Objetivos **18**

3.1. Objetivo principal 18

3.2. Objetivos formales 18

3.3. Objetivos específicos 19

4. Antecedentes **20**
4.1. Origen de los videojuegos 20
4.2. Historia de Unreal Engine 21
4.3. Unreal Engine 23
4.4. Proyectos similares 24
4.4.1. Dreams 24
4.4.2. Content Examples from Unreal Engine 4 25

5. Restricciones **27**
5.1. Factores dato 27
5.2. Factores estratégicos 27

6. Recursos **29**
6.1. Recursos Hardware 29
6.2. Recursos Software 29
6.3. Recursos humanos 30

II ANALISIS Y ESPECIFICACIÓN DE REQUISITOS 31

7. Análisis y especificación de requisitos **32**
7.1. Requisitos de información 32
7.2. Requisitos funcionales 34
7.3. Requisitos no funcionales 35

III MODELADO DE REQUISITOS 37

8. Analisis funcional **38**
8.1. Caso de uso 0. Contexto del programa 39
8.2. Caso de uso 1. Pulsar botón explicativo 41
8.3. Caso de uso 2. Realizar test 43
8.4. Caso de uso 3. Agarrar objeto 44

8.5. Caso de uso 4. Cambiar opciones de pantalla 46
8.6. Caso de uso 5. Cargar partida 48

9. Análisis dinámico **50**
9.1. Diagramas de secuencia 50
9.1.1. Diagrama de secuencia Botón explicativo 50
9.1.2. Diagrama de secuencia Realizar test 52
9.1.3. Diagrama de secuencia Agarrar objeto 54
9.1.4. Diagrama de secuencia Cambiar opciones de pantalla 56
9.1.5. Diagrama de secuencia Cargar partida 57

IV DISEÑO DEL SISTEMA **60**

10. Diseño de datos **61**
10.1. Base de datos diálogo automático 61
10.2. Base de datos diálogo explicativo 63
10.3. Base de datos diálogo pregunta 65
10.4. Base de datos opciones pregunta 67

11. Diseño de niveles **69**
11.1. Nivel 1 69
11.2. Nivel 2 72
11.3. Nivel 3 73

12. Diseño de la interfaz **76**
12.1. Prototipo de la interfaz 76
12.1.1. Menú principal 76
12.1.2. Menú de opciones de pantalla 77
12.1.3. Menú de controles 79
12.1.4. Menú de pausa 79

V PRUEBAS 81

13. Pruebas 82

13.1. Pruebas de caja blanca 83
13.1.1. Menú principal 83
13.1.2. Nivel 1 84
13.1.3. Nivel 2 85
13.1.4. Nivel 3 86
13.2. Ejecución de pruebas de caja blanca 88
13.2.1. Menú principal 88
13.2.2. Nivel 1 89
13.2.3. Nivel 2 89
13.2.4. Nivel 3 90
13.3. Prueba de caja negra 91
13.3.1. Pruebas del caso de uso 1: Pulsar botón explicativo . 91
13.3.2. Pruebas del caso de uso 2: Realizar test 91
13.3.3. Pruebas del caso de uso 3: Agarrar objeto 92
13.3.4. Pruebas del caso de uso 4: Cambiar opciones de pantalla 92
13.3.5. Pruebas del caso de uso 5: Cargar partida 93
13.4. Ejecución de pruebas de caja negra 93

VI CONCLUSIONES 94

14. Conclusiones 95

14.1. Conclusiones 95
14.1.1. Conclusiones de objetivos formales 95
14.1.2. Conclusiones de objetivos específicos 96

15. Futuras mejoras 97

Bibliografía **98**

Índice de figuras

1. Marketplace de Unreal Engine 24
2. Captura de Dreams 25
3. Content Examples from Unreal Engine 4 en el editor 26

4. Caso de uso 0: Contexto del Sistema 40
5. Caso de uso 1: Pulsar botón explicativo 42
6. Caso de uso 2: Realizar Test 43
7. Caso de uso 3: Agarrar objeto 45
8. Caso de uso 4: Cambiar opciones de pantalla 47
9. Caso de uso 5: Cargar partida 48

10. Diagrama de secuencia Botón explicativo 51
11. Diagrama de secuencia Realizar Test 53
12. Diagrama de secuencia Agarrar objeto 55
13. Diagrama de secuencia Cambiar opciones de pantalla 56
14. Diagrama de secuencia Cargar partida 58

15. Boceto de la primera del nivel 1 70
16. Boceto de la segunda zona del nivel 1 71
17. Boceto de la primera zona del nivel 2 72
18. Boceto de la segunda zona del nivel 2 73
19. Boceto de la primera zona del nivel 3 74
20. Boceto de la segunda zona del nivel 3 75

21. Boceto del menú principal 77
22. Boceto del menú de opciones de pantalla 78
23. Boceto del menú de listado de controles 79
24. Boceto del menú de pausa 80

Índice de cuadros

1. Ejemplo de cuadro de Caso de uso 39
2. Tabla caso de uso 0: Contexto del sistema 41
3. Tabla caso de uso 1: Pulsar botón explicativo 42
4. Tabla caso de uso 2: Realizar test 44
5. Tabla caso de uso 3: Agarrar objeto 46
6. Tabla caso de uso 4: Cambiar opciones de pantalla 47
7. Tabla caso de uso 5: Cargar partida 49

8. Pruebas caso de uso 1: Pulsar botón explicativo 91
9. Pruebas caso de uso 2: Realizar test 91
10. Pruebas caso de uso 3: Agarrar objeto 92
11. Pruebas caso de uso 4: Cambiar opciones de pantalla 92
12. Pruebas caso de uso 5: Cargar partida 93

Parte I

Introducción

Capítulo 1

Introducción

Los videojuegos están presentes en la vida cotidiana como forma recurrente de ocio. Aunque es cierto que hay cierta reticencia, sobre todo en avanzadas edades, algo similar ha ocurrido con todas las tecnologías: ocurrió con la radio, ocurrió con la televisión y seguirá ocurriendo con todo lo que se invente y esté destinado a estar en los hogares.

El proceso de creación de un videojuego es complejo y conlleva muchos elementos. No consiste solo en diseñar los niveles, también hay que crear los personajes, la música... Para un proyecto de este tipo se requiere la colaboración de expertos de múltiples ámbitos: artistas, músicos, matemáticos... Por supuesto, el proceso implica múltiples aspectos de programación, similares a los que se abordan en el Grado de Ingeniería en Informática. Este aspecto de programación es muchas veces minusvalorado por los usuarios, que solo observan el resultado final. En este sentido, programar un videojuego puede ser complejo o tosco, pero es una tarea muy completa con la que puedes ver resultados de forma muy visual y satisfactoria.

Este proyecto está orientado a enseñar de forma amigable algunos conceptos de programación de videojuegos, sirviendo de iniciación para los usuarios. Para realizar esta tarea parece que lo más idóneo es crear un

videojuego utilizando alguna de las diferentes plataformas de desarrollo que hay disponibles. En nuestro caso, hemos elegido Unreal Engine, porque es uno de los motores más utilizados y potentes de los disponibles en la actualidad. Además, todos los meses proporciona recursos de forma gratuita y de libre uso: interfaces, modelos, música…; y todo ello facilitará toda la parte de diseño y se podrá hacer más énfasis en la programación.

Capítulo 2

Definición del problema

A continuación se mostrará la definición del problema desde diferentes puntos de vista:

2.1. Identificación del problema real

Unreal Engine posee múltiples escenarios de pruebas pre-construidos en los cuales puedes interactuar con diversos elementos del motor gráfico <referencia> y aprender de manera general cómo funcionan los mismos. A su vez, existen incontables cursos y tutoriales de programación y uso de Unreal Engine.<referencia>

Este proyecto tiene como objetivo un concepto intermedio. Poder enseñar conceptos de programación y de uso del motor al usuario de forma interactiva, a la vez que, para avanzar, debes poner a prueba los conceptos aprendidos. Ésto puede ser una manera de introducción a Unreal Engine para programadores que aún no hayan usado el programa y quieran aprender de una manera más entretenida y, a su vez, una manera en la examine si se ha entendido correctamente aquello que se acaba de explicar.

2.2. Identificación del problema técnico

La definición del problema técnico se realizará mediante la técnica Product Design Specification (PDS), la cual está formada por los siguientes apartados:

2.2.1. Funcionamiento

El programa ofrecerá al usuario, el cual ya posee conocimientos de programación, un tutorial interactivo de conceptos útiles en Unreal Engine. Dicho tutorial se compondrá de 3 niveles:

- Un nivel físico, el cual explicará los principales estados que puede tener cada uno de los actores que se encuentran en un proyecto de Unreal Engine.
- Un nivel de blueprints, el cual enseñará el método de programación propio de Unreal Engine.
- Un nivel de C++, el cual enseñará conceptos de código C++ específicos de Unreal Engine.

2.2.2. Entorno

El programa se podrá ejecutar desde cualquier ordenador con una instalación de Windows 7+. Aunque el entorno óptimo de ejecución del programa será en la versión de Windows 10.

El programa tendrá una interfaz sencilla e intuitiva, a pesar de que está dirigido a un público con conocimientos previos de programación.

2.2.3. Vida esperada

Se trata de un programa que enseña conceptos básicos de Unreal Engine en la versión 4.20, pero al no adentrarse en temas avanzados, es posible que siga siendo útil en las versiones venideras. Es por ello que realmente la vida esperada del producto es incierta, si bien se intentará que sea lo mayor posible.

2.2.4. Ciclo de mantenimiento

El programa se prestará a modificaciones y nuevos añadidos si así se viese necesario, aunque no necesitará de un mantenimiento periódico y solo se tratará en el caso de que éste contenga errores que sean descubiertos posteriormente.

2.2.5. Competencia

La llamada «Marketplace» de Unreal Engine propone una serie de muestras del motor, siento algunas de ellas interactivas, pero éstas no ponen a prueba al usuario. Es por ello que no tendría ningún competidor directo, aunque sí tendría bastantes competidores de una forma menos inmediata.

2.2.6. Aspecto externo

El programa tendrá el aspecto de un videojuego al uso, controlándose mediante teclado y ratón.

2.2.7. Estandarización

Se han usado códigos de estandarización recomendados por Unreal Engine. Éstos son:

- Organización de clases: dentro de la clase, el apartado público debe ser declarado primero, seguido por el privado.
- Aviso de copyright de Epic Games por la distribución en cada archivo fuente.
- Los nombres de las variables deben comenzar por letra mayúscula

2.2.8. Calidad y fiabilidad

Se asegurará de que el programa funcione correctamente y sea robusto mediante un profundo testeo de las opciones que ofrece.

Al ser un entorno 3D es puede dar pie a pequeños errores físicos, por lo que será necesario comprobar cualquier situación y pulir lo que fuera necesario. En cambio, al ser una prueba lineal, el abanico de errores es bastante más pequeño del que podría ser si se otorgase algún grado más de libertad.

2.2.9. Programación de tareas

Estas son las tareas que se llevarán acabo para la realización de este proyecto.

- **Estudio y análisis del problema:** En esta fase se realizará el aprendizaje de las herramientas que más tarde se usarán, principalmente el motor gráfico Unreal Engine. Además, se estudiará la peculiaridad de este, y las diferencias con respecto al resto de motores del mercado, para así enseñar conocimientos propios únicamente de Unreal Engine.
- **Diseño:** En la fase de diseño se preparará una idea general de qué va a ser el tutorial, las fases de las que constará y cómo se desarrollará cada una.

- **Implementación:** Una vez realizado el diseño del programa, se procederá a la implementación del mismo, así como la codificación. Por una parte se realizarán todos los apartados físicos y visuales, diseñados en la fase de diseño, y por otra parte se programará todo aquello que el tutorial necesite para que su funcionamiento sea el esperado.
- **Pruebas:** En esta fase se realizarán una serie de pruebas dedicadas a comprobar que el funcionamiento del programa es el esperado.
- **Documentación:** Esta tarea consiste en la realización de esta memoria técnica, así como la documentación de todo el código del programa y un manual de usuario para el uso de este.

2.2.10. Pruebas

Para una asegurar la robustez del programa, así como su correcto funcionamiento, se realizarán pruebas exhaustivas.

Se probará cada uno de los niveles de forma individual, abarcando todas las opciones disponibles para el jugador. Además, para una mayor seguridad, el programa será probado por más usuarios, con tal de corroborar el funcionamiento del mismo.

El rendimiento también será probado, y para ello se ejecutará el programa desde distintos ordenadores, comprobando que funcione correctamente siempre y cuando cumplan con los requisitos mínimos.

2.2.11. Seguridad

Más allá de garantizar la estabilidad del sistema, no se ha contemplado ningún tipo de sistema de seguridad, al no haber datos personales con los que tratar.

Capítulo 3

Objetivos

3.1. Objetivo principal

El principal objetivo de este projecto será poder mostrar a nuevos usuarios del motor gráfico Unreal Engine diferentes características propias de éste y transmitirles conocimientos básicos de su uso de forma interactiva.

3.2. Objetivos formales

Los objetivos formales del proyecto son los siguientes:

- Aprender a usar el motor gráfico Unreal Engine
- Aprender a programar en el entorno gráfico de Visual Studio
- Aprender distintos conceptos de programación en el ámbito de los videojuegos
- Aplicar los conocimientos aprendidos en el Grado en un proyecto completo

3.3. Objetivos específicos

Se deberán cumplir los siguientes objetivos específicos:

- Diseño y desarrollo de los niveles en los que constará el tutorial: los elementos que formarán, la disposición de los mismos y los eventos que ocurrirán.
- Un uso simple e intuitivo, que no genere duda ni problema alguno, para que el usuario se centre completamente en lo que el tutorial pretende enseñar.
- Un diseño de una base de datos robusta para los diálogos que aparecerán a lo largo del tutorial.
- Diseñar una interfaz intuitiva y amigable. Así el usuario sabrá fácilmente la utilidad de cada elemento y no causará agobio alguno.
- Elaborar una documentación completa de los procesos de realización del proyecto, así como un manual de código y un manual de usuario.

Capítulo 4

Antecedentes

A continuación, se exponen los antecedentes a este proyecto, empezando de manera más general para luego ir acotando a antecedentes más específicos y similares al proyecto.

4.1. Origen de los videojuegos

La considerada primera "máquina de juegos" es "NIMATRON" [2], una máquina electromecánica para jugar al juego matemático Nim. Fue mostrada en la feria mundial de Nueva York en 1940 por el Dr. Edward Uhler Condon. Jugaron unas 50.000 personas en los 6 meses que estuvo exponiéndose [3].

La primera videoconsola tenía el nombre de "Magnavox Odyssey". Anteriormente llamada "Brown Box", fue lanzada como prototipo en 1967, y permitía controlar cubos que se movían por la pantalla. Se podían programar juegos como el ping pong. Meses más tarde Atari lanzó la suya y fue la encargada de expandir la idea al mercado, debido a su gran éxito. Se lanzó el primer videojuego electrónico, el conocido "Pong".

Los primeros videojuegos 3D en primera persona fueron "Maze War", diseñado por Steve Celley, y "Spasim", lanzado por Jim Bowery, ambos

en 1974. Ambos permitían hasta 32 jugadores simultáneos. El primero fue desarrollado para la Imlac PDS-1 y posteriormente portado a Mac, Next, PalmOS y Xerox. El segundo se lanzó para el sistema educativo en red de la Universidad de Illinois (PLATO). El género no adquirió popularidad hasta "Wolfenstein 3D" (1992), y un año más tarde "Doom", ambos creados por id Software y lanzados para PC [1].

El género se fue expandiendo a más especialidades además de disparos, siendo algunos ejemplos "MYST" (1991) como videojuego de puzle o "The Elder Scrolls: Arena" (1994) como videojuego de aventura rol. Este proyecto será de tipo puzle, el cual alcanzó una fama enorme con el lanzamiento de "Portal" (2007) que popularizó el género de puzle moderno, permitiendo una gran libertad e inmersión, como se puede observar, por ejemplo, en "Antichamber" (2013).

4.2. Historia de Unreal Engine

Originalmente, no existían los motores de videojuegos. Estos eran desarrollados como una única entidad, es decir, no había separación entre la parte lógica y la gráfica, así que las rutinas de dibujado debían ser implementadas desde cero.

No fue hasta la década de los noventa cuando se comenzó a hablar del término "game engine"[4]. "Doom" y "Quake" fueron dos videojuegos que alcanzaron una gran popularidad debido a que, en lugar de crear un videojuego desde cero, se licenciaba la parte esencial del mismo para usarlo como base y crear motores de juego 3D llamados "id Tech 1" y "Quake Engine", respectivamente. También se crearon motores en dos dimensiones como el famoso "RPG Tsukūru Dante 98" en 1992, el cual sigue siendo actualizado a día de hoy bajo el nombre de "RPG Maker"[5].

En 1998, el estudio de videojuegos "Epic MegaGames" (actual "Epic Games") lanza el videojuego "Unreal", el cual presentó un nuevo motor

gráfico con el mismo nombre. Este motor adquirió tanto éxito que ganó por mucho en popularidad a "id Tech 4", el que era hasta ese momento el motor más conocido.

Esa versión fue denominada como "Unreal Engine 1", y entre sus características estaba la detección de colisión o el editor de niveles[6]. Se crearon videojuegos de éxito como "X-COM: Enforcer" o el ya nombrado "Unreal".

La segunda versión se lanzó en 2002, e hizo su debut con "America's Army", un shooter desarrollado con la intención de servir como prueba de reclutamiento para la armada estadounidense. Esta versión es capaz de crear niveles mucho más detallados y tenía un sistema de animación de esqueleto y físicas mejoradas. "Bioshock 1" o "Lineage II" usaban este motor.

En 2004 se publicó Unreal Engine 3, siendo utilizado por primera vez por el videojuego "Gears of War". A pesar de que en lo visible para el jugador era todo drásticamente nuevo y con más potencial, el aspecto técnico no cambió mucho, excepto que el sistema de sombreado era totalmente programable. Soportaba una gran cantidad de plataformas como Windows, Playstation 3, Xbox 360, Android o iOS. "Borderlands", "Mass Effect" o "Rocket League" usan este motor.

Hubo que esperar hasta 2012 para que se publicase la actual versión, "Unreal Engine 4". Fue un cambio muy importante, ya que un par de años después de su lanzamiento, fue lanzado gratuitamente para todo el mundo, incluyendo todo el código fuente escrito en C++. Ésta versión es de libre uso con un 5 % de royalties para publicadores y 100 % libre de royalties para proyectos gratuitos o privados. La propia empresa Epic, lanzó el mundialmente conocido "Fortnite" como demostración del potencial que tenía el motor en videojuegos multijugador.

4.3. Unreal Engine

El motor que se va a usar se define a sí mismo como la plataforma más abierta y completa de creación de proyectos en 3D a tiempo real[7]. Actualmente se encuentra en la versión 4.24.3., con una versión 4.25. ya en camino, y está en constante evolución con una excelente comunicación y transparencia con el usuario.

El motor consiste en una interfaz gráfica de total modificación para adaptarse a las necesidades personales. Tiene integradas herramientas de todo tipo para mayor rapidez y simpleza, entre ellas herramientas de animación, de modelaje, sonido, iluminación, cinemática, y muchas más.

Con respecto a la renderización, es un proceso que se realiza en segundo plano y da la oportunidad de seguir trabajando mientras tanto. El programa tiene una perfecta integración con Visual Studio, lo cual da la oportunidad de modificar el código del proyecto con facilidad y rapidez, y permite la compilación en tiempo real para ver los resultados al instante Uno de los conceptos interesantes que se tratará es el de la colisión de objetos.

En Unreal Engine se divide en tres tipos: block, overlap e ignore. El primero de ellos no permite al objeto A traspasar el objeto B, mientras los otros dos sí. La diferencia entre overlap e ignore es que cuando el objeto A realiza un overlap sobre el objeto B, permite realizar la acción que se programe. Unreal Engine tiene dos modos de programación. La programación por código con Visual Studio, y los llamados blueprints. Son modos totalmente integrados entre sí, siendo el uso de ambos la manera más recomendada de realizar un proyecto. Con los blueprints te permite conectar o modificar de manera más intuitiva funciones, variables, objetos...

Desde 2014, Unreal Engine tiene una tienda íntegra o “marketplace” para adquirir recursos de pago hechos por la comunidad, empresas externas o la propia Epic Games. Cada mes ofrecen los llamados “Free of the

Month", recursos que anteriormente eran de pago y los ofrecen temporalmente sin ningún coste adicional. Entre estos recursos podemos encontrar: música, interfaces, modelos de personajes o escenario, animaciones, sonidos, inteligencia artificial y más, como se puede ver en la figura 1.

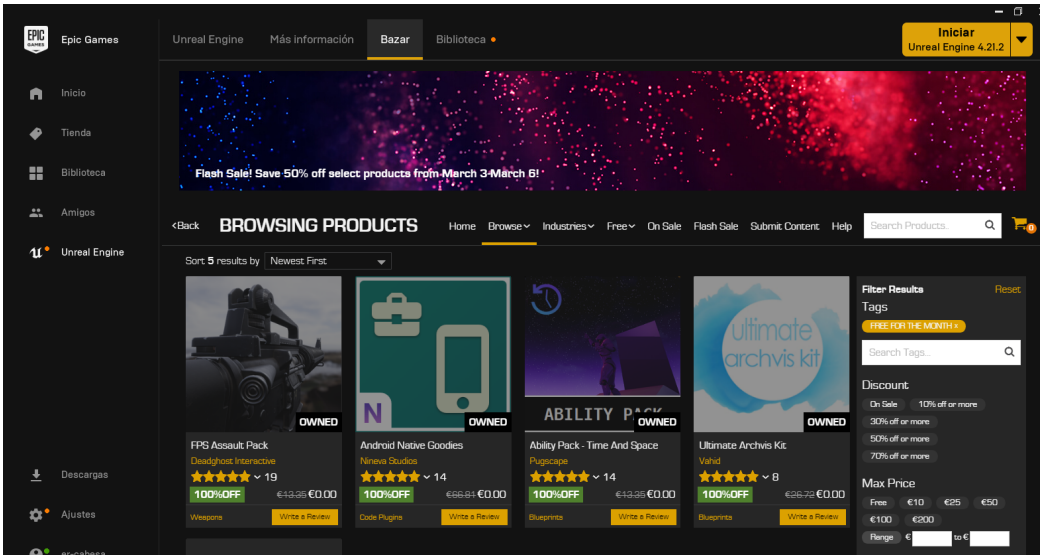

Figura 1: Marketplace de Unreal Engine

4.4. Proyectos similares

4.4.1. Dreams

Si hablamos de tutoriales interactivos sobre desarrollo de videojuegos, un ejemplo muy reciente es "Dreams", un videojuego lanzado recientemente por Media Molecule y publicado por Sony para PlayStation 4. Se trata de un entorno de creación de juegos: da las herramientas necesarias para poder crear tus propias mecánicas, niveles, música, etc. Usa muchos conceptos de programación de manera sencilla e interactiva para que sea más entendible y tiene múltiples tutoriales que enseñan dichos conceptos.

Los creadores de contenido tienen la oportunidad de publicar sus obras en la misma plataforma, siendo no solo un entorno de creación sino de disfrute y descubrimiento de las obras de la propia comunidad. Tiene manejo sencillo, lo cual permite a cualquier usuario poder realizar creaciones. A su vez, puede llegar a un nivel de complejidad muy alto. De esa manera es posible crear de todo: desde sencillos personajes, hasta complejos sistemas de inteligencia artificial. En la figura 2 se puede ver el aspecto de este videojuego.

Figura 2: Captura de Dreams

4.4.2. Content Examples from Unreal Engine 4

Es una guía gratuita que ofrece Epic para explicar una gran cantidad de funcionalidades del motor. Simula una visita a un museo, como se muestra en la figura 3, y mientras se prueba el proyecto se pueden observar los resultados de lo que se enseña. El proyecto hace una explicación breve y si se desea saber cómo funciona, se puede abrir desde el editor y comprobar-

lo. Esta guía abarca tanto temas de físicas, matemáticas o programación, como audio, animación o diseño. No hay conexión entre las zonas ni mecánicas de juego para avanzar, pero es un gran ejemplo para ver de primera mano todo lo que ofrece el motor y poder interactuar con ello en tiempo real.

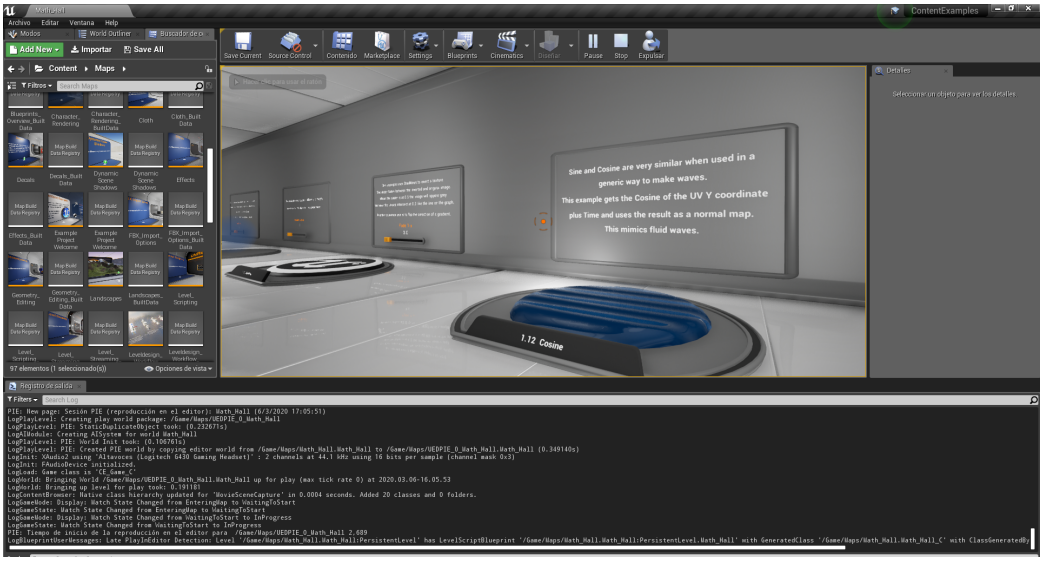

Figura 3: Content Examples from Unreal Engine 4 en el editor

Capítulo 5

Restricciones

Este apartado presentará las limitaciones o restricciones que se presentarán a lo largo del desarrollo del tipo del proyecto. Se dividirán en dos tipos: los factores dato son aquellos inherentes al propio problema, mientras que los factores estratégicos los forman decisiones realizadas a partir del diseño del mismo.

5.1. Factores dato

Estas son los distintos factores dato, inherentes al proyecto:

- El código del proyecto estará escrito en C++, ya que es el lenguaje de programación que soporta el motor gráfico Unreal Engine.
- El software usado es totalmente gratuíto, ya que Unreal Engine está disponible para cualquier persona a través de la Epic Store.

5.2. Factores estratégicos

Los siguientes puntos son las decisiones tomadas a lo largo del desarrollo del proyecto:

- El proyecto estará programado en el motor gráfico Unreal Engine, concretamente en la versión 4.24. Ésta era la última versión que existía cuando se comenzó el proyecto y, para evitar problemas de compatibilidad, no se ha trasladado a las siguientes versiones.

- Se programará el código en el entorno de programación Visual Studio. Esto es debido a que tiene integración directa con el motor y permite opciones como la compilación directa desde Visual Studio o la depuración de código.

- Se usará también la programación por blueprints, qué es el método de programación visual proporcionado por Unreal Engine, basado en un sistema de nodos y conexiones.

- Las imágenes usadas estarán en formato PNG, para así poder trabajar con transparencias, con una alta calidad y una compresión sin pérdida.

- Se usará, en su mayoría, recursos del propio motor gráfico y otros proporcionados por la propia tienda de la plataforma Epic Games, siendo éstos recursos de software libre y, al menos los que usaremos, disponibles de forma gratuíta.

Capítulo 6

Recursos

Este apartado servirá para listar los recursos usados en el desarrollo del proyecto, divididos en recursos hardware, recursos software y recursos humanos

6.1. Recursos Hardware

Se ha usado un ordenador personal con las siguientes especificaciones:

- Procesador Intel Core 15-7400 @ 3.00 GHz
- 16GB de memoria RAM
- Unidad HDD de 1TB de capacidad
- Unidad SDD de 240GB de capacidad

6.2. Recursos Software

Estos son los recursos Software usados para el desarrollo del proyecto:

- Sistema operativo Windows 10 Home

- Motor gráfico Unreal Engine, en su versión 4.24
- Entorno de programación Visual Studio
- El procesador de documentos LyX, que permite la creación y edición de documentos en LaTeX
- Clip Studio para el diseño de imágenes

6.3. Recursos humanos

A continuación se detallará el personal dedicado al desarrollo del proyecto

- El autor, Rafael Moreno Serrano, estudianto de cuarto curso del Grado en Ingeniería Informática de la Escuela Politécnica Superior de la Universidad de Córdoba
- Los directores encargados de la coordinación del proyecto:
    - D. Pedro Antonio Gutiérrez Peña
    - D. Javier Barbero Gómez

Parte II

ANALISIS Y ESPECIFICACIÓN DE REQUISITOS

Capítulo 7

Análisis y especificación de requisitos

Para que el proyecto realice correctamente su funcionamiento necesita cubrir unos requisitos que especificaremos a continuación.

Hay tres tipos de requisitos, que serán: requisitos de información, requisitos funcionales y requisitos no funcionales.

7.1. Requisitos de información

Estos requisitos hacen referencia a la información que debe contener el proyecto. En este apartado se incluye, por ejemplo, todo lo almacenado en bases de datos.

Estos requisitos serán enumerados con las siglas RI (Requisitos de Información):

- **RI 1**: Diálogos: El proyecto almacenará todo el contenido relacionado con los diálogos que se mostrarán en pantalla, tanto los que ocurren al interactuar como los que se accionan automáticamente. Éstos diálogos contendrán la siguiente información:

- NPC_ID: El código identificativo del NPC u objeto con el cual será necesario interactuar para accionar el diálogo.
- Conversation_ID: El código identificativo de la conversación dentro de un mismo NPC u objeto. Depende de la situación, podría tener un diálogo u otro.
- Line_ID: El código identificativo de la línea de diálogo dentro de una misma conversación.
- Dialogue: El texto que será mostrado en pantalla

- **RI 2**: Tests: El proyecto necesitará almacenar la información necesaria a los tests que se incluyen en el proyecto. Para ello, se requiere de la siguiente información, dividida en dos bases de datos distintas para mantener consistencia. La primera base de datos consiste en:

  - NPC_ID: El código identificativo del NPC u objeto con el cual será necesario interactuar para accionar el diálogo.
  - Conversation_ID: El código identificativo de la conversación dentro de un mismo NPC u objeto. Depende de la situación, podría tener un diálogo u otro.
  - Line_ID: El código identificativo de la línea de diálogo dentro de una misma conversación.
  - Dialogue: El texto que será mostrado en pantalla
  - IsQuestion: Indica si en la línea de diálogo actual se encuentra una pregunta y debe activarse el test.

  La segunda base de datos, a la que se accede cuando se detecta que el diálogo es una pregunta:

  - Answer_ID: el código identificativo de la respuesta a la que refiere.

  - Answer: El texto de la respuesta.
  - IsCorrect: Indica si la respuesta es correcta o no

- **RI 3**: Progreso guardado: El proyecto guardará la información relativa al progreso realizado anteriormente, el cual se guarda de forma automática al realizar diversas acciones al avanzar. La información guardada es la siguiente:
  - Nivel: Se refiere al nivel en el que se encuentra el personaje en ese momento
  - Posición: Es referido a la posición actual del actor dentro del nivel

7.2. Requisitos funcionales

Los requisitos funcionales determinan la función del proyecto. Éstos tienen que estar definidos detalladamente, pero sin adentrarse en exceso, ya que será un tema que se abarcará de manera más amplia más adelante.

Estos requsitos se numerarán con las siglas RF (Requisito Funcional):

- **RF 1:** El usuario podrá moverse libremente por el plano horizontal. Además de eso, se podrá saltar.
- **RF 2**: Además del movimiento, el usuario podrá manejar la cámara a su antojo, pudiendo girar en las tres dimensiones.
- **RF 3**: La resolución de la pantalla podrá ser cambiada tanto en el menú principal como en el menú de pausa.
- **RF 4**: El menú de pausa podrá abrirse en todo momento en la prueba del proyecto.

- **RF 5**: El progreso se guardará automáticamente al realizar cualquier avance en el tutorial interactivo.
- **RF 6**: Así mismo, el progreso se podrá cargar desde el menú principal.
- **RF 7**: El usuario podrá interrumpir su partida en cualquier momento cerrando el software, para lo cual existirá una opción en el menú de pausa.
- **RF 8**: El proyecto mostrará debidamente el concepto que se está explicando, mediante imágenes, mientras éste está siendo explicado.
- **RF 9**: El usuario pordrá interactuar con cualquier botón que haya en el escenario, estando éstos señalizados debidamento
- **RF 10**: Los tests darán la oportunidad de errar en la respuesta sin ninguna consecuencia más allá de mostrar el error cometido, además de dar la oportunidad de intentarlo todas las veces necesarias

7.3. Requisitos no funcionales

En este apartado se trataran los requisitos que no describen una funcionalidad del proyecto. Éstos se refieren a temas de accesibilidad, calidad o consistencia del proyecto.

Este tipo de requisitos son numerados con las siglas RNF (Requisito No Funcional):

- **RNF 1**: La interfaz será intuitiva en todo momento. El usuario debe entender la funcionalidad de cada botón de cada menú.
- **RNF 2**: Los controles serán sencillos y no supondrán ningún problema a la hora de avanzar, ya que en la principal premisa del proyecto

no se encuentra ninguna prueba de habilidad, sino de poner a prueba los conocimientos que se enseñan.

- **RNF 3**: El proyecto funcionará correctamente en cualquier ordenador que posea los requisitos mínimos especificados en el Manual de Usuario.
- **RNF 4**: Los tiempos de carga al iniciar el proyecto y al cambiar de nivel deberán ser lo más cortos posible, para no estropear la experiencia al usuario.
- **RNF 5**: Los tests tratarán sobre los conocimientos explicados anteriormente, y las respuestas deberán ser intuitivas si el usuario ha leído la información que se ofrece.
- **RNF 6**: El proyecto será consistente y robusto a errores y funcionar correctamente en todo momento
- **RNF 7**: El proyecto tendrá un escenario agradable y sencillo, que no distraiga al usuario del propósito principal

Parte III

MODELADO DE REQUISITOS

Capítulo 8

Analisis funcional

Un análisis funcional es aquel que especifica qué comportamiento debe presentar un sistema. En nuestro caso, lo haremos mediante casos de uso.

Los diagramas de casos de uso buscan representar todos los elementos de un sistema y sus relaciones entre sí, desde el punto de vista de un actor. Estos diagramas muestran la relación entre dicho actor y sus requisitos o expectativas del sistema.

Para definir un caso de uso, usaremos una tabla con la siguiente información:

- **Caso de uso:** Descripción concisa del caso de uso que se va a definir.
- **Id:** Número identificativo del caso de uso.
- **Descripción:** Descripción más detallada del caso de uso.
- **Actores:** Entidad externa al sistema que guarda relación con este e interactúa con él.
- **Precondiciones:** Condiciones que deben presentarse en el sistema para que el caso de uso pueda darse de forma correcta.
- **Flujo principal:** Fases principales que realiza el caso de uso.

- **Flujos alternativos:** Fases alternativas por las que puede pasar el caso de uso.
- **Post condiciones:** Estado del sistema una vez se realice el caso de uso de forma correcta.

Sabiendo esto, la tabla quedaría de la manera que se muestra en la tabla 1:

Caso de uso:	Nombre conciso del caso de uso
Id:	Número identificativo del caso de uso
Descripción:	Descripción detallada del caso de uso
Actores:	Entidad externa que interviene en el caso de uso
Precondiciones:	Condiciones necesarias para ejecutar el caso de uso
Flujo principal:	Fases principales del caso de uso
Flujo alternativo:	Fases alternativas del caso de uso
Postcondiciones:	Estado del sistema después del caso de uso

Cuadro 1: Ejemplo de cuadro de Caso de uso

8.1. Caso de uso 0. Contexto del programa

El caso de uso que aparece en la figura 4 es el correspondiente al iniciar el programa:

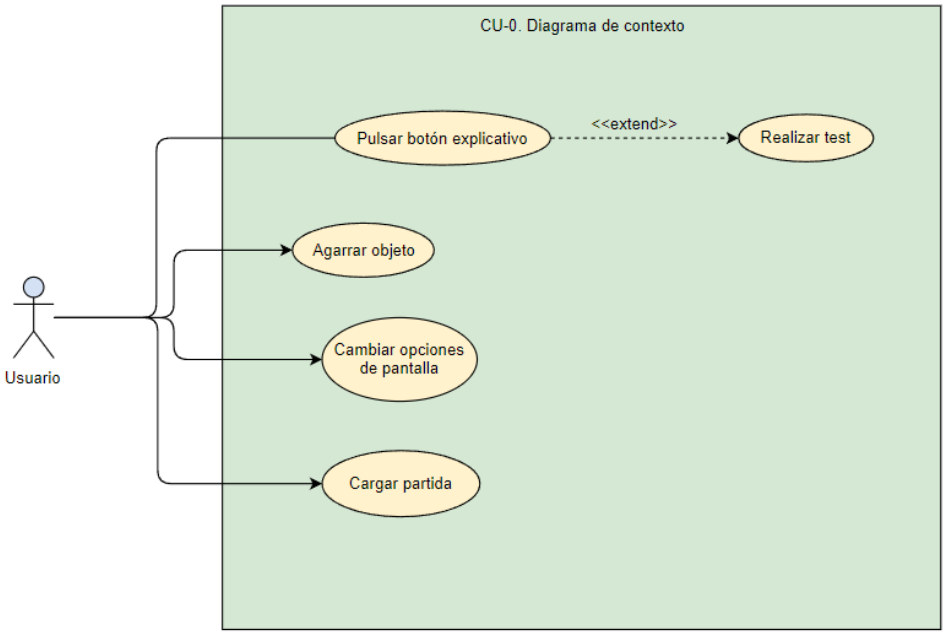

Figura 4: Caso de uso 0: Contexto del Sistema

Caso de Uso:	Caso de uso 0: Diagrama de contexto
ID:	1
Descripción:	Este diagrama muestra las distintas elecciones que se pueden realizar
Casos de uso:	CU-1. Pulsar botón explicativo: El usuario pulsa un botón para recibir una explicación CU-1.5. Realizar test: El usuario pulsa un botón explicativo y comienza un test CU-3. Agarrar objeto: El usuario coge un objeto para llevarlo consigo CU-3. Cambiar opciones de pantalla: El usuario cambia la resolución de la pantalla o el modo de ventana CU-4. Cargar partida: El usuario selecciona esta opción para cargar partida

Cuadro 2: Tabla caso de uso 0: Contexto del sistema

8.2. Caso de uso 1. Pulsar botón explicativo

En la figura 5 aparece el diagrama correspondiente a este caso de uso.

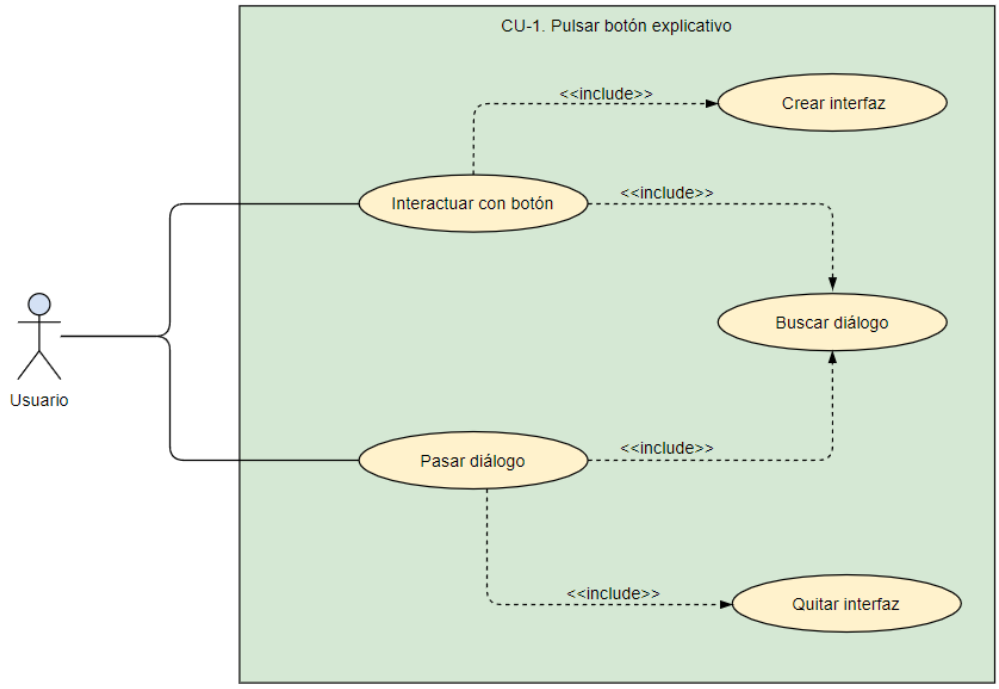

Figura 5: Caso de uso 1: Pulsar botón explicativo

Caso de uso:	Pulsar botón explicativo
Id:	1
Descripción:	El usuario recibe una lección informativa de uno de los botones explicativos
Actores:	Usuario
Precondiciones:	El usuario no debe estar en un menú El usuario debe estar en frente de un botón explicativo
Flujo principal:	1. El usuario interactúa con un botón explicativo 2. El usuario lee y avanza por los diálogos correspondientes
Flujo alternativo:	Ninguno
Postcondiciones:	El usuario ha aprendido una lección sobre el concepto correspondiente

Cuadro 3: Tabla caso de uso 1: Pulsar botón explicativo

8.3. Caso de uso 2. Realizar test

En la figura 6 aparece el diagrama correspondiente a este caso de uso.

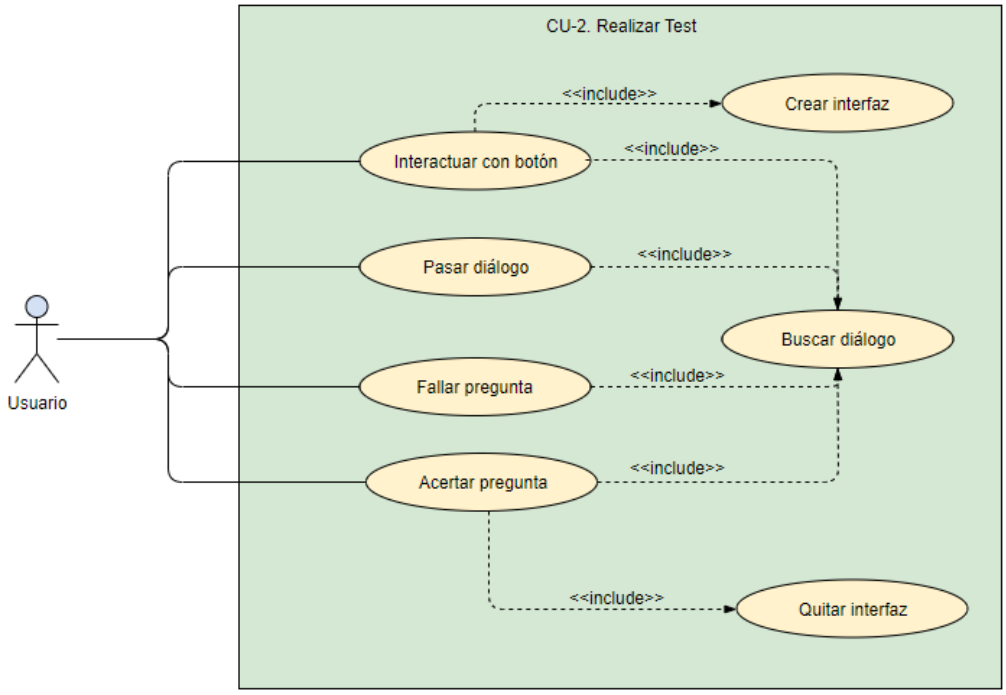

Figura 6: Caso de uso 2: Realizar Test

Caso de uso:	Realizar test
Id:	2
Descripción:	El usuario realiza un test proporcionado después de pulsar el botón apropiado
Actores:	Usuario
Precondiciones:	El usuario no debe estar en un menú El usuario debe estar en frente de un botón de test El usuario no debe haber realizado el test anteriormente
Flujo principal:	1. El usuario interactúa con un botón de test 2. El usuario avanza por el diálogo hasta llegar a las preguntas 3. El usuario responde correctamente a las preguntas
Flujo alternativo:	1. El usuario responde erróneamente a una pregunta 1.a. Se muestra una explicación de por qué es erróneo 1.b. Se repite la pregunta fallada
Postcondiciones:	El usuario ha pasado satisfactoriamente el test y se elimina la interfaz de la pantalla

Cuadro 4: Tabla caso de uso 2: Realizar test

8.4. Caso de uso 3. Agarrar objeto

En la figura 7 aparece el diagrama correspondiente a este caso de uso.

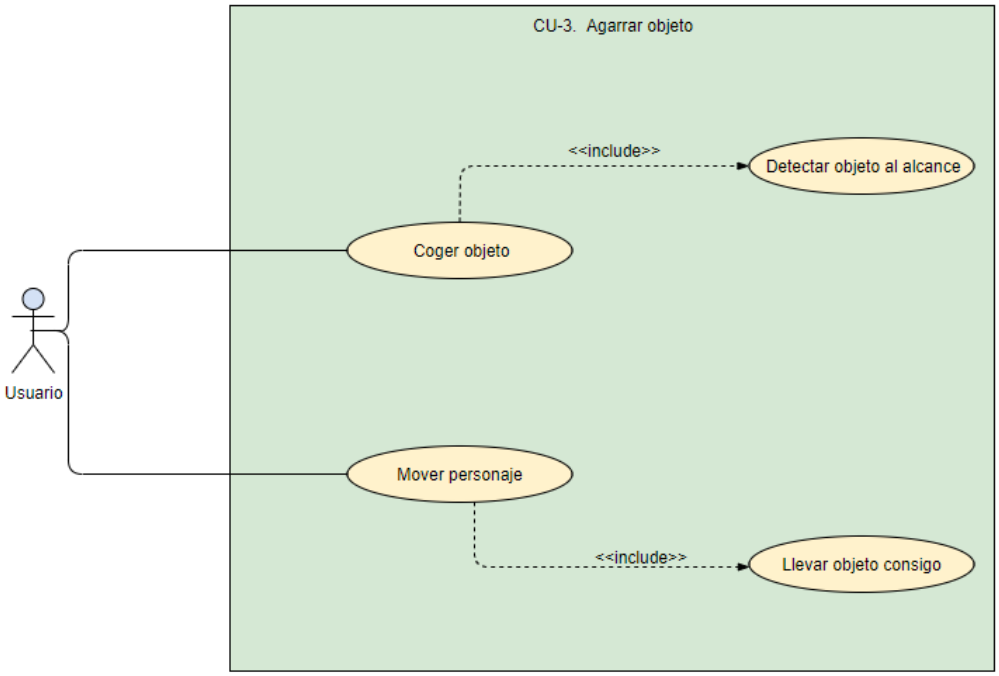

Figura 7: Caso de uso 3: Agarrar objeto

Caso de uso:	Agarrar objeto
Id:	3
Descripción:	El usuario coge un objeto y se mueve
Actores:	Usuario
Precondiciones:	El usuario no debe estar en un menú El usuario debe estar en frente de un objeto agarrable El usuario no debe estar sujetando ningún otro objeto
Flujo principal:	1. El usuario coge un objeto 2. El usuario se mueve mientras mantiene el botón de coger objeto
Flujo alternativo:	Ninguno
Postcondiciones:	El usuario lleva consigo el objeto que ha agarrado

Cuadro 5: Tabla caso de uso 3: Agarrar objeto

8.5. Caso de uso 4. Cambiar opciones de pantalla

En la figura 8 aparece el diagrama correspondiente a este caso de uso.

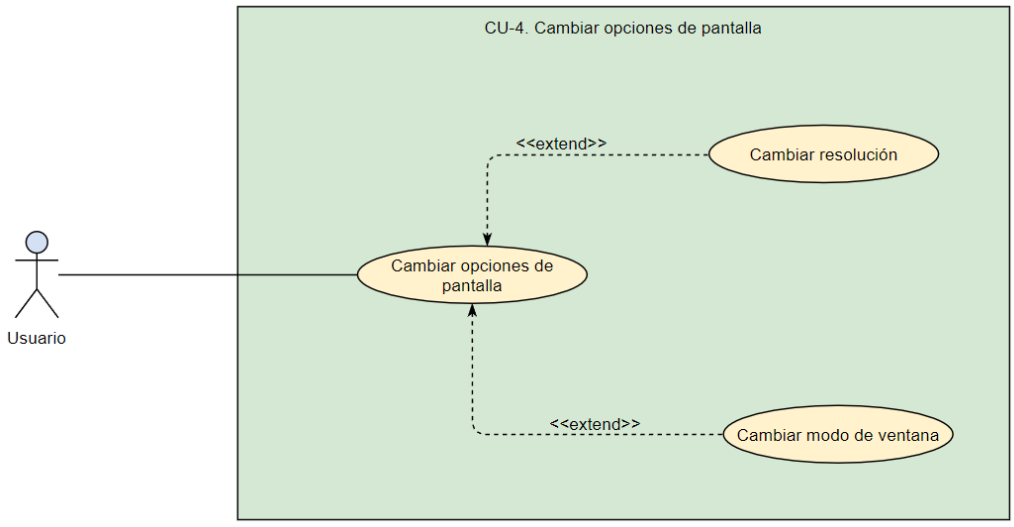

Figura 8: Caso de uso 4: Cambiar opciones de pantalla

Caso de uso:	Cambiar opciones de pantalla
Id:	4
Descripción:	El usuario cambia alguna opción de pantalla en el menú principal
Actores:	Usuario
Precondiciones:	El usuario debe estar en el menú de opciones de pantalla El usuario no debe escoger la opción que está actualmente activa
Flujo principal:	1. El usuario entra en el menú de opciones de pantalla 2. El usuario elige una opción diferente a la actual
Flujo alternativo:	Ninguno
Postcondiciones:	Cambia el modo de ventana o la resolución de la misma

Cuadro 6: Tabla caso de uso 4: Cambiar opciones de pantalla

8.6. Caso de uso 5. Cargar partida

En la figura 9 aparece el diagrama correspondiente a este caso de uso.

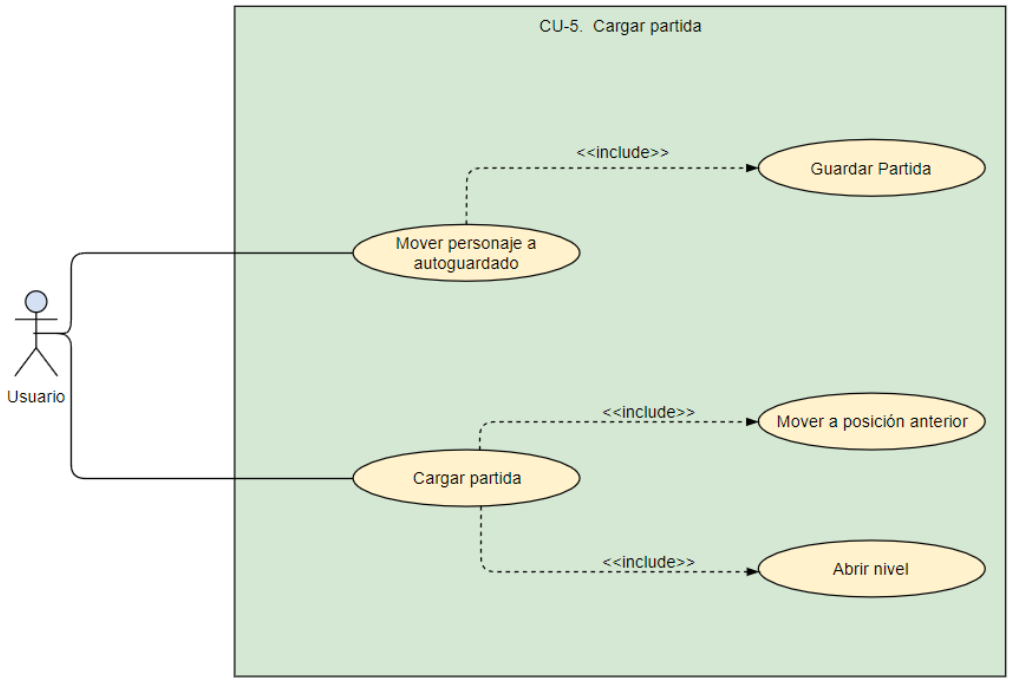

Figura 9: Caso de uso 5: Cargar partida

Caso de uso:	Cargar partida
Id:	5
Descripción:	El usuario activa una zona de autoguardado y luego decide cargar partida
Actores:	Usuario
Precondiciones:	El usuario no debe encontrarse en un menú El usuario debe tener el control del personaje
Flujo principal:	1. El usuario mueve al personaje por una zona de autoguardado. 2. El usuario sale al menú principal 3. El usuario pulsa el botón de cargar partida
Flujo alternativo:	Ninguno
Postcondiciones:	La partida se ha cargado donde anteriormente se había autoguardado

Cuadro 7: Tabla caso de uso 5: Cargar partida

Capítulo 9

Análisis dinámico

En este capítulo se verán los principales diagramas de secuencia del proyecto. Un diagrama de secuencia relaciona los distintos modelos del sistema a través del tiempo, indicando los cambios de los mismos.

9.1. Diagramas de secuencia

9.1.1. Diagrama de secuencia Botón explicativo

El diagrama de secuencia sería el que aparece en la figura 10.

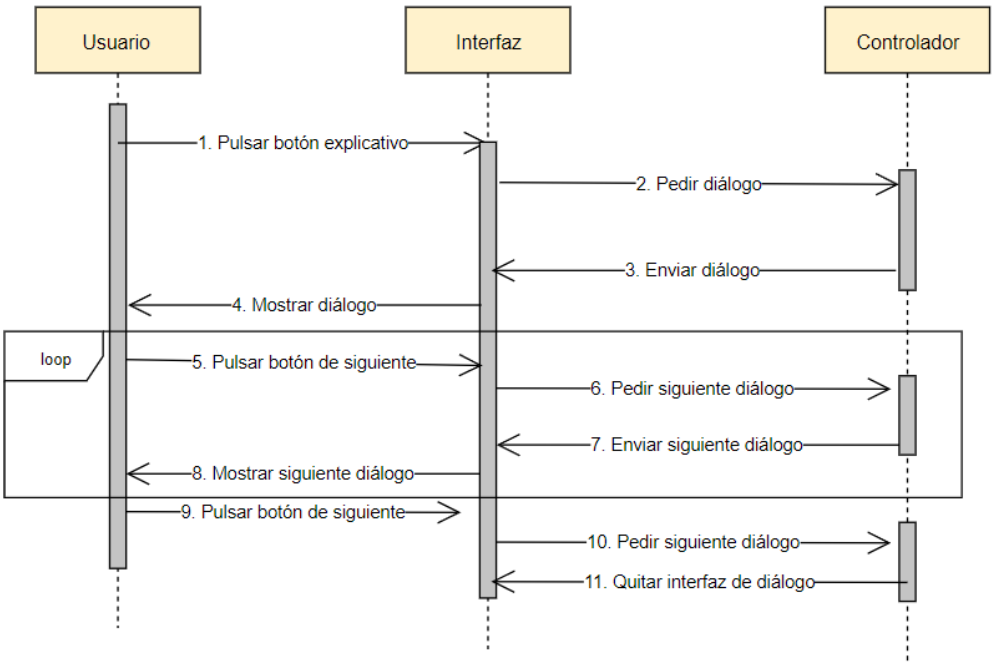

Figura 10: Diagrama de secuencia Botón explicativo

La secuencia sería la siguiente:

1. **Pulsar botón explicativo:** El usuario se acerca a un botón explicativo situado en el mapa y pulsa el botón de interactuar.

2. **Pedir diálogo:** Se crea la interfaz de diálogo y ésta pide al controlador el diálogo correspondiente al botón.

3. **Enviar diálogo:** El diálogo se encuentra y se envía a la interfaz.

4. **Mostrar diálogo:** El diálogo encontrado se muestra al usuario por medio de la interfaz en pantalla.

5. **Pulsar botón de siguiente:** El usuario vuelve a pulsar el botón de interactuar para avanzar al siguiente diálogo.

6. **Pedir siguiente diálogo:** La interfaz pide al controlador el diálogo que suceda al anteriormente mostrado.

7. **Enviar siguiente diálogo:** El controlador encuentra el diálogo y lo envía a la interfaz.

8. **Mostrar siguiente diálogo:** El diálogo recogido se muestra al usuario por medio de la interfaz.

9. **Pulsar botón de siguiente:** El usuario vuelve a interactuar para avanzar en el diálogo.

10. **Pedir siguiente diálogo:** De nuevo, se vuelve a pedir el correspondiente diálogo.

11. **Quitar interfaz de diálogo:** Se detecta que no quedan diálogos y el controlador manda quitar la interfaz de diálogo.

9.1.2. Diagrama de secuencia Realizar test

El diagrama de secuencia sería el que aparece en la figura 11.

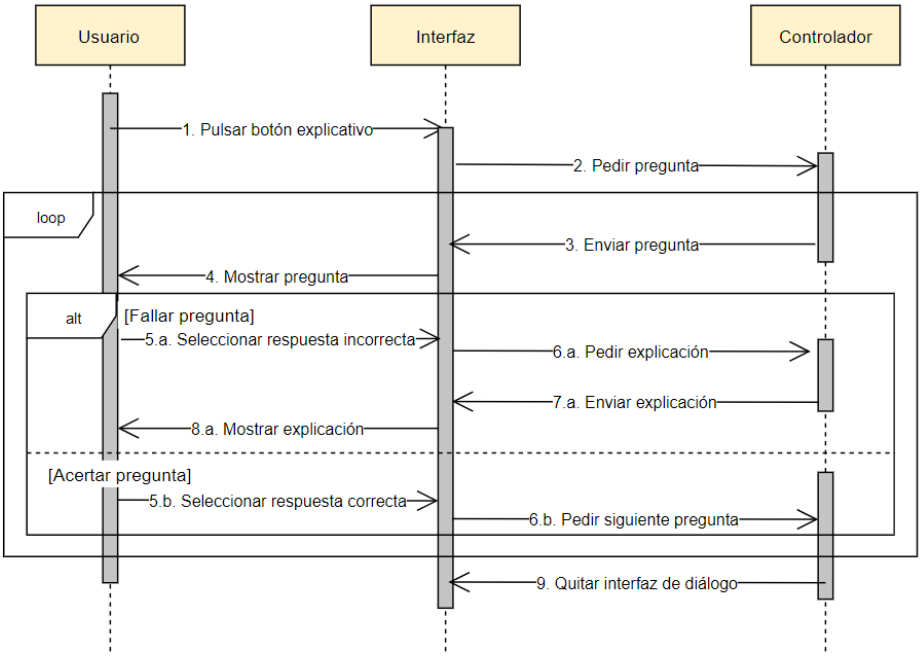

Figura 11: Diagrama de secuencia Realizar Test

La secuencia sería la siguiente:

1. **Pulsar botón explicativo:** El usuario se acerca a un botón explicativo situado en el mapa y pulsa el botón de interactuar.

2. **Pedir pregunta:** Se crea la interfaz de diálogo y ésta pide al controlador la pregunta correspondiente.

3. **Enviar pregunta:** El diálogo se encuentra y se envía a la interfaz, junto con sus respuestas.

4. **Mostrar pregunta:** El diálogo encontrado se muestra al usuario por medio de la interfaz en pantalla, junto con sus respuestas.

5. **Seleccionar respuesta incorrecta:** El usuario selecciona una respuesta incorrecta a la pregunta.

6. **Pedir explicación:** Se pide a la interfaz la explicación correspondiente a la respuesta errónea escogida.

7. **Enviar explicación:** Se envía la corrección a la interfaz.

8. **Mostrar explicación:** Se muestra la corrección al usuario por medio de la interfaz.

Camino alternativo:

5.b. **Seleccionar respuesta correcta:** El usuario selecciona la respuesta correcta a la pregunta.

6.b. **Pedir siguiente pregunta:** La interfaz pide al controlador la siguiente pregunta

9. **Quitar interfaz de diálogo:** El controlador no encuentra más preguntas y envía la opción de quitaz la interfaz de diálogo.

9.1.3. Diagrama de secuencia Agarrar objeto

El diagrama de secuencia sería el que aparece en la figura 12

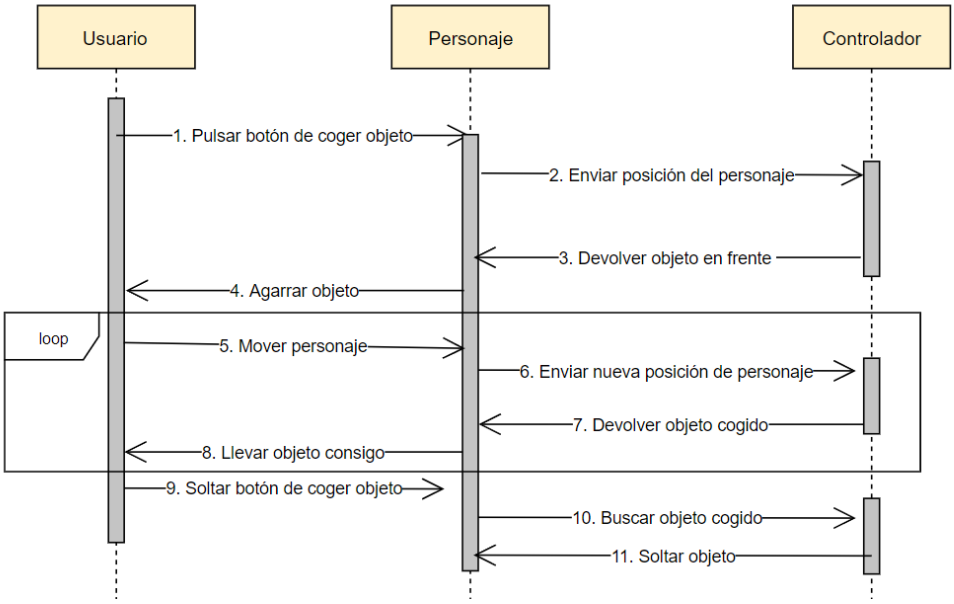

Figura 12: Diagrama de secuencia Agarrar objeto

La secuencia sería la siguiente:

1. **Pulsar botón de coger objeto:** El usuario mantiene presionado el botón de coger un objeto.

2. **Enviar posición del personaje:** Se envía la posición actual del personaje, incluído hacia dónde está mirando.

3. **Devolver objeto en frente:** Se devuelve el objeto que esté en el alcance de agarre del personaje.

4. **Agarrar objeto:** El personaje agarra el objeto devuelto.

5. **Mover personaje:** El usuario mueve el personaje a otro lugar.

6. **Enviar nueva posición de personaje:** Se envía al controlador la nueva posición del personaje.

7. **Devolver objeto cogido:** Se devuelve qué objeto tenía agarrado.
8. **Llevar objeto consigo:** El objeto se transporta junto con el personaje.
9. **Soltar botón de coger objeto:** El usuario suelta el botón que mantenía presionado.
10. **Buscar objeto cogido:** Se busca qué objeto tenía agarrado.
11. **Soltar objeto:** Se suelta el objeto que se haya devuelto.

9.1.4. Diagrama de secuencia Cambiar opciones de pantalla

El diagrama de secuencia sería el que aparece en la figura 13.

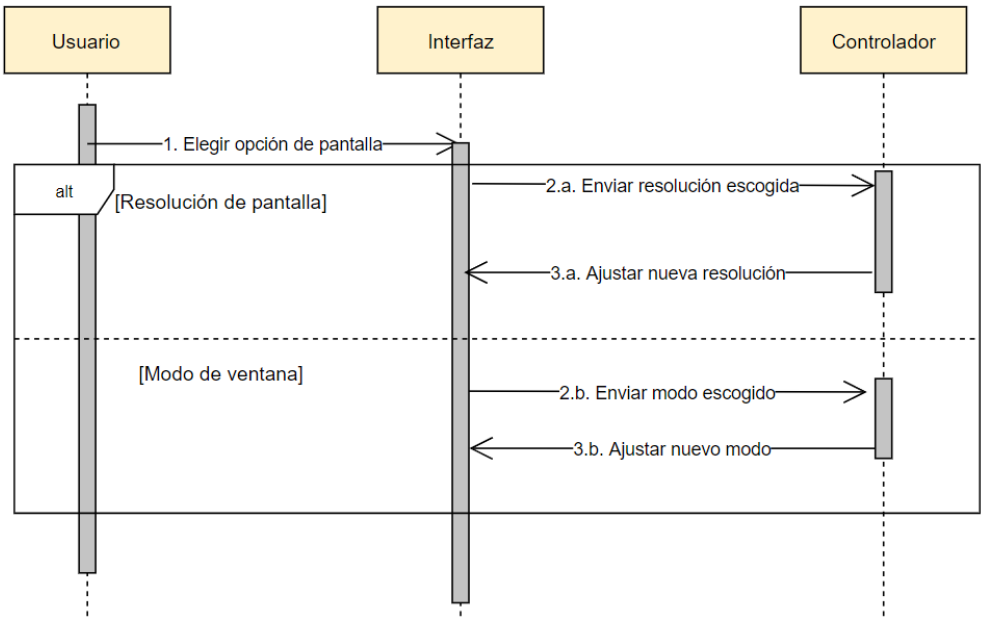

Figura 13: Diagrama de secuencia Cambiar opciones de pantalla

La secuencia sería la siguiente:

1. **Elegir opción de pantalla:** El usuario elige una nueva opción de pantalla, que puede ser una nueva resolución o un modo de ventana.

2. **Enviar resolución escogida:** Se envía al controlador la nueva resolución de pantalla escogida.

3. **Ajustar nueva resolución:** Se ajusta la nueva resolución de pantalla.

Camino alternativo:

2. **Enviar modo escogido:** Se envía al controlador el nuevo modo de ventana, que puede ser o pantalla completa o modo ventana.

3. **Ajustar nuevo modo:** Se ajusta la ventana al nuevo modo escogido.

9.1.5. Diagrama de secuencia Cargar partida

El diagrama de secuencia sería el que aparece en la figura 14

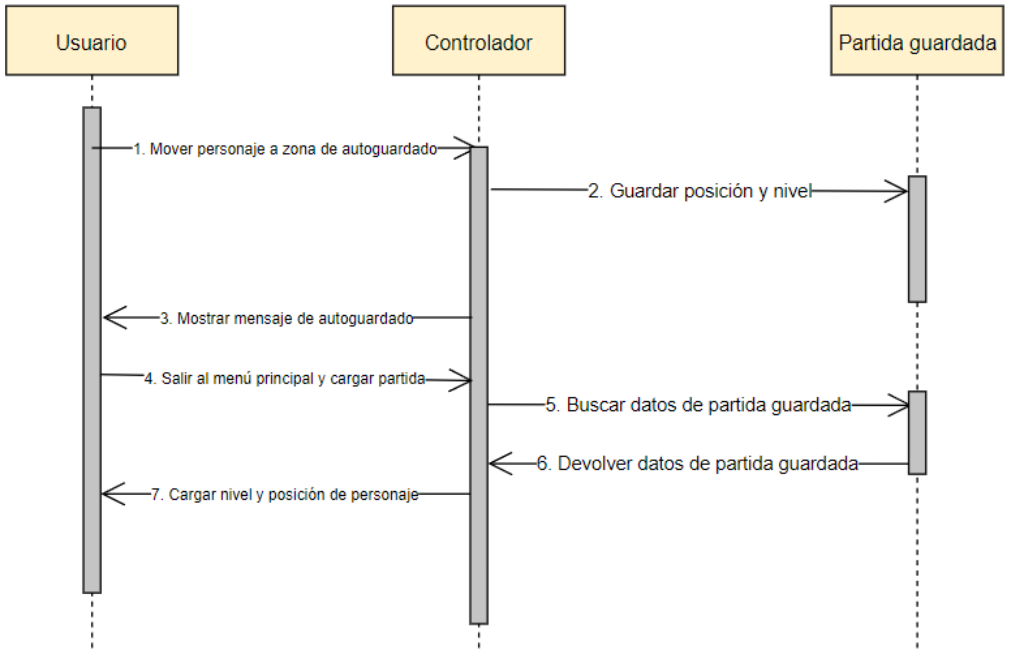

Figura 14: Diagrama de secuencia Cargar partida

La secuencia sería la siguiente:

1. **Mover personaje a zona de autoguardado:** El usuario mueve al personaje a través de una zona de autoguardado.

2. **Guardar posición y nivel:** El controlador guarda la posición actual del personaje y el nivel en el que se encuentra.

3. **Mostrar mensaje de autoguardado:** El controlador muestra al usuario un mensaje de que el autoguardado se ha realizado correctamente.

4. **Salir al menú principal y cargar partida:** El usuario, desde el menú principal, escoge la opción de cargar partida guardada.

5. **Buscar datos de partida guardada:** El controlador busca los datos de

la partida guardada anteriormente, que son la posición del personaje y el nivel.

6. **Devolver datos de partida guardada:** Se devuelven los datos de la partida guardada anteriormente.

7. **Cargar nivel y posición de personaje:** Se abre el nivel de la partida guardada con el personaje en la posición del autoguardado.

Parte IV

DISEÑO DEL SISTEMA

Capítulo 10

Diseño de datos

Las bases de datos suelen ser muy importantes en cualquier proyecto software. En el caso concreto de este proyecto, no hay muchos datos que tratar, ya que no se guardan datos personales del usuario ni de ninguna entidad. Las únicas bases de datos que se realizarán serán las que contengan los diálogos de todo el tutorial, tanto los explicativos como los de test.

Las bases de datos de los diálogos se han dividido en tres:

- Una base de datos para los diálogos automáticos al comienzo de un nivel.
- Una base de datos para los diálogos de los botones explicativos.
- Una base de datos para los diálogos de los botones de test.
- Una base de datos para las diferentes opciones de cada pregunta.

10.1. Base de datos diálogo automático

- **Descripción:** Define los diálogos que aparecen al comienzo de un nivel y no están relacionados con ningún actor.

- **Características:**
  - **Nombre:** AutoDialogue
  - **Número de atributos:** 4
  - **Atributo identificativo:** Row Name
- **Descripción de los atributos:**
  - **Row Name:**
    - **Definición:** Nombre identificativo que diferencia cada uno de los elementos de la base de datos.
    - **Dominio:** Conjunto de caracteres.
    - **Carácter:** Obligatorio.
    - **Ejemplo:** 1.
    - **Información adicional:** Este es el identificativo que proporciona Unreal Engine por defecto para sus bases de datos.
  - **Level_ID:**
    - **Definición:** Número identificativo del nivel en el que aparecerá el diálogo.
    - **Dominio:** Números naturales.
    - **Carácter:** Obligatorio.
    - **Ejemplo:** 1.
    - **Información adicional:** Por defecto es 0.
  - **Window_ID:**
    - **Definición:** Número identificativo de la ventana del diálogo.
    - **Dominio:** Números naturales.
    - **Carácter:** Obligatorio.
    - **Ejemplo:** 1.

- **Información adicional:** No puede haber dos iguales en un mismo Level_ID. Por defecto es 0.

- **Dialogue:**
  - **Definición:** Diálogo que se mostrará en pantalla.
  - **Dominio:** Conjunto de caracteres.
  - **Carácter:** Obligatorio.
  - **Ejemplo:** Hola.
  - **Información adicional:** Es un diálogo único, correspondiente al conjunto de identificadores anteriores.

10.2. Base de datos diálogo explicativo

- **Descripción:** Define los diálogos que aparecen al interactuar con un botón explicativo.
- **Características:**
  - **Nombre:** Dialogue
  - **Número de atributos:** 5
  - **Atributo identificativo:** Row Name
- **Descripción de los atributos:**
  - **Row Name:**
    - **Definición:** Nombre identificativo que diferencia cada uno de los elementos de la base de datos.
    - **Dominio:** Conjunto de caracteres.
    - **Carácter:** Obligatorio.
    - **Ejemplo:** 1.

  - **Información adicional:** Este es el identificativo que proporciona Unreal Engine por defecto para sus bases de datos.

- **NPC_ID:**
  - **Definición:** Número identificativo del NPC o botón del diálogo.
  - **Dominio:** Números naturales.
  - **Carácter:** Obligatorio.
  - **Ejemplo:** 1.
  - **Información adicional:** Por defecto es 0.
- **Conversation_ID:**
  - **Definición:** Número identificativo de la conversación del diálogo.
  - **Dominio:** Números naturales.
  - **Carácter:** Obligatorio.
  - **Ejemplo:** 1.
  - **Información adicional:** Por defecto es 0.
- **Line_ID:**
  - **Definición:** Número identificativo de la línea del diálogo.
  - **Dominio:** Números naturales.
  - **Carácter:** Obligatorio.
  - **Ejemplo:** 1.
  - **Información adicional:** Por cada conjunto de NPC_ID y Conversation_ID solo puede haber un Line_ID. Por defecto es 0.
- **Dialogue:**
  - **Definición:** Diálogo que se mostrará en pantalla.
  - **Dominio:** Conjunto de caracteres.

- **Carácter:** Obligatorio.
- **Ejemplo:** Hola.
- **Información adicional:** Es un diálogo único, correspondiente al conjunto de identificadores anteriores.

10.3. Base de datos diálogo pregunta

- **Descripción:** Define los diálogos pertenecientes a un test.
- **Características:**
  - **Nombre:** TestDialogue
  - **Número de atributos:** 7
  - **Atributo identificativo:** Row Name
- **Descripción de los atributos:**
  - **Row Name:**
    - **Definición:** Nombre identificativo que diferencia cada uno de los elementos de la base de datos.
    - **Dominio:** Conjunto de caracteres.
    - **Carácter:** Obligatorio.
    - **Ejemplo:** 1.
    - **Información adicional:** Este es el identificativo que proporciona Unreal Engine por defecto para sus bases de datos.
  - **Test_ID:**
    - **Definición:** Número identificativo del test del diálogo.
    - **Dominio:** Números naturales.
    - **Carácter:** Obligatorio.
    - **Ejemplo:** 1.

- **Información adicional:** Por defecto es 0.

- **Conversation_ID:**
  - **Definición:** Número identificativo de la conversación del diálogo.
  - **Dominio:** Números naturales.
  - **Carácter:** Obligatorio.
  - **Ejemplo:** 1.
  - **Información adicional:** Por defecto es 0.
- **Line_ID:**
  - **Definición:** Número identificativo de la línea del diálogo.
  - **Dominio:** Números naturales.
  - **Carácter:** Obligatorio.
  - **Ejemplo:** 1.
  - **Información adicional:** Por cada conjunto de Test_ID y Conversation_ID solo puede haber un Line_ID. Por defecto es 0.
- **Dialogue:**
  - **Definición:** Diálogo que se mostrará en pantalla.
  - **Dominio:** Conjunto de caracteres.
  - **Carácter:** Obligatorio.
  - **Ejemplo:** Hola.
  - **Información adicional:** Es un diálogo único, correspondiente al conjunto de identificadores anteriores.
- **IsQuestion:**
  - **Definición:** Indica si el diálogo actual es una pregunta o no.
  - **Dominio:** 1 o 0.
  - **Carácter:** Obligatorio.

    - **Ejemplo:** 1.
    - **Información adicional:** Es el número que se usa para acceder a las opciones en su base de datos correspondiente. Por defecto es 0.
- **Question_ID:**
    - **Definición:** Número identificativo de la pregunta del diálogo.
    - **Dominio:** Números naturales.
    - **Carácter:** Opcional.
    - **Ejemplo:** 1.
    - **Información adicional:** Por defecto es 0.

10.4. Base de datos opciones pregunta

- **Descripción:** Define las diferentes opciones a las preguntas de un test.
- **Características:**
    - **Nombre:** QuestionDataTable
    - **Número de atributos:** 4
    - **Atributo identificativo:** Row Name
- **Descripción de los atributos:**
    - **Row Name:**
        - **Definición:** Nombre identificativo que diferencia cada uno de los elementos de la base de datos.
        - **Dominio:** Conjunto de caracteres.
        - **Carácter:** Obligatorio.

- **Ejemplo:** 1.
- **Información adicional:** Este es el identificativo que proporciona Unreal Engine por defecto para sus bases de datos.

- **Question_ID:**
  - **Definición:** Número identificativo de la pregunta del diálogo.
  - **Dominio:** Números naturales.
  - **Carácter:** Obligatorio.
  - **Ejemplo:** 1.
  - **Información adicional:** Por defecto es 0.
- **Dialogue:**
  - **Definición:** Opción correspondiente a la pregunta.
  - **Dominio:** Conjunto de caracteres.
  - **Carácter:** Obligatorio.
  - **Ejemplo:** Hola.
  - **Información adicional:** Opción perteneciente a la pregunta correspondiente en la base de datos de preguntas.
- **Answer_ID:**
  - **Definición:** Número identificativo de la respuesta del diálogo.
  - **Dominio:** Números naturales.
  - **Carácter:** Opcional.
  - **Ejemplo:** 1.
  - **Información adicional:** Por defecto es 0.

Capítulo 11

Diseño de niveles

En este capítulo trataremos de la composición de cada uno de los niveles, sus elementos y la disposición en el mismo.

La intención ha sido la de tener un diseño sencillo e intuitivo, nada cargado, para que los elementos importantes del nivel estén claramente diferenciados

11.1. Nivel 1

El primer nivel se basa en explicar conceptos físicos de Unreal Engine. En concreto, habla de las diferencias de objetos en base a su tipo de colisión. El aspecto del nivel es el que se observa en el boceto de la figura 15

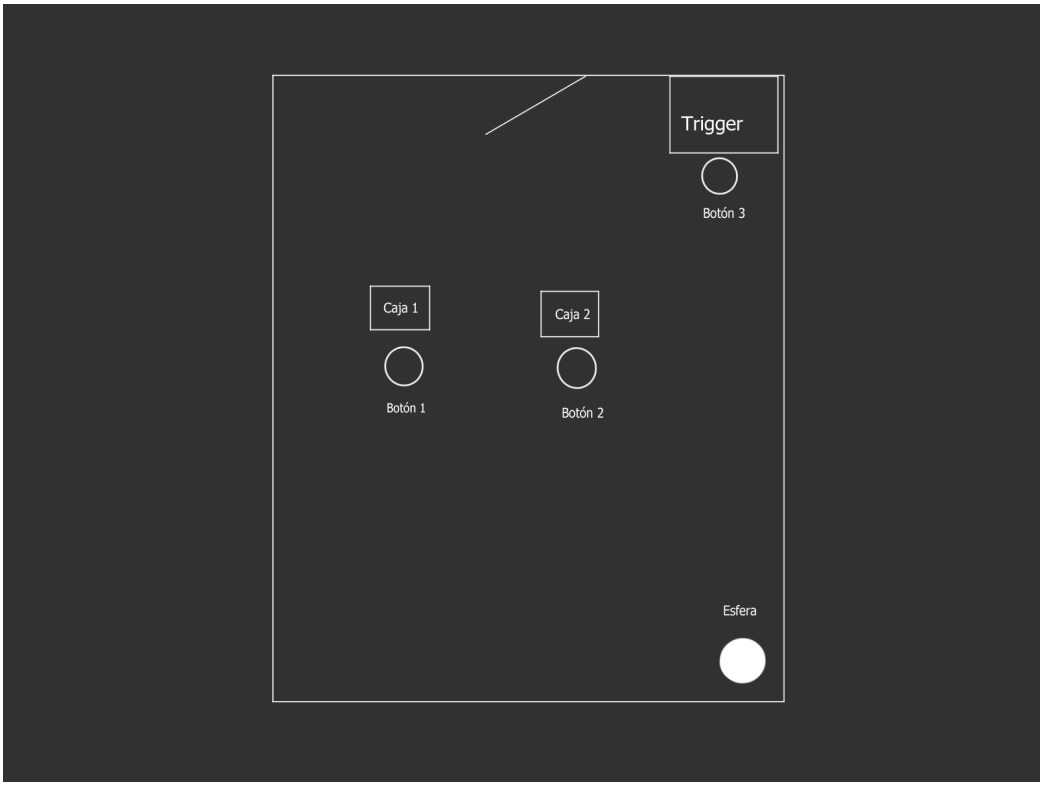

Figura 15: Boceto de la primera del nivel 1

Nada más aparecer, un diálogo aparecerá en pantalla, dando la bienvenida y haciendo una explicación general del tutorial y el nivel. Todos los diálogos están escritos desde el punto de vista de un gato llamado Mixi.

Hay tres botones explicativos en esta sala:

- El botón de la izquierda explica los objetos del tipo «block».
- El botón de la derecha explica los objetos del tipo «ignore».
- El botón del fondo explica los objetos de tipo «overlap» y los «triggers». Justo delante tiene un trigger en el que tienes que soltar una esfera para abrir la puerta a la siguente zona

Al pasar por la puerta a la siguiente zona del nivel 1, se activa un autoguardado.

En la figura 16se puede ver un boceto de la segunda parte del nivel.

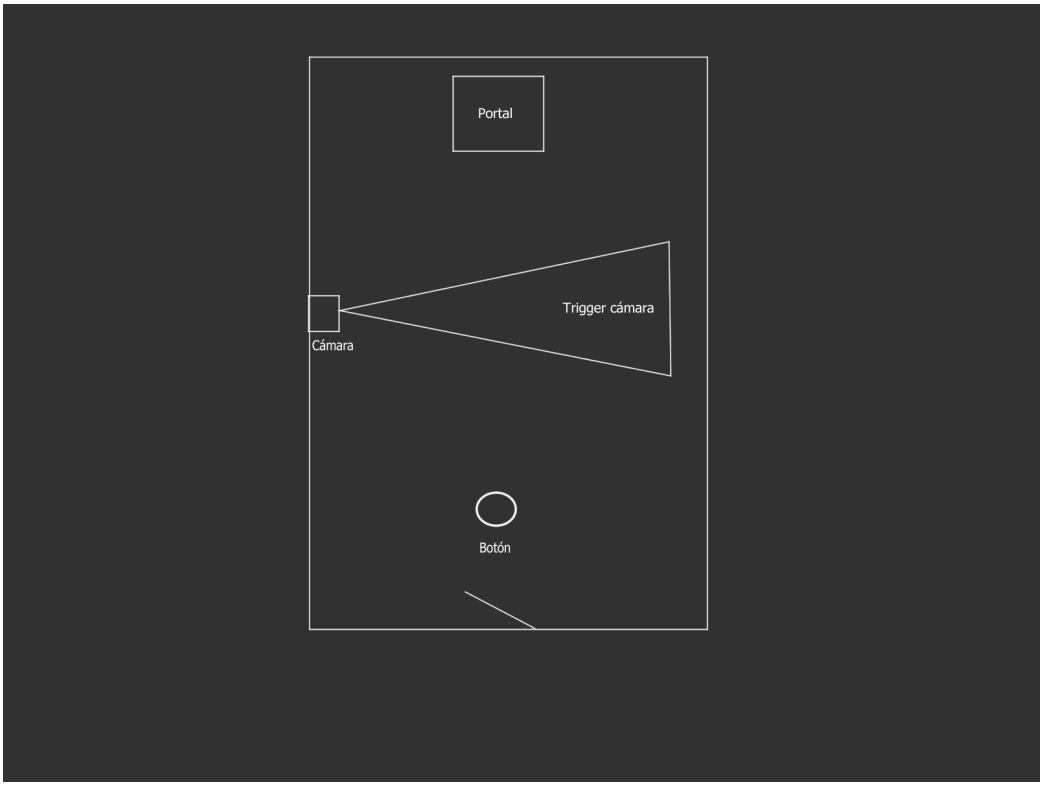

Figura 16: Boceto de la segunda zona del nivel 1

En esta zona hay únicamente un botón. Al interactuar con él, te explica el uso de los triggers como cámaras de seguridad. Al explicarlo, se hace visible el cono correspondiente al trigger de la cámara, para que el usuario pueda verlo claramente.

Ahora que el trigger es visible, el usuario puede pasar al final de la zona y entrar al portal que te lleva al nivel 2.

11.2. Nivel 2

Al aparecer en el nivel 2, vas a ver una habitación con botones. En la figura 17 se puede ver un boceto de la habitación. Este nivel está basado en enseñar conceptos de C++ específicos de Unreal Engine.

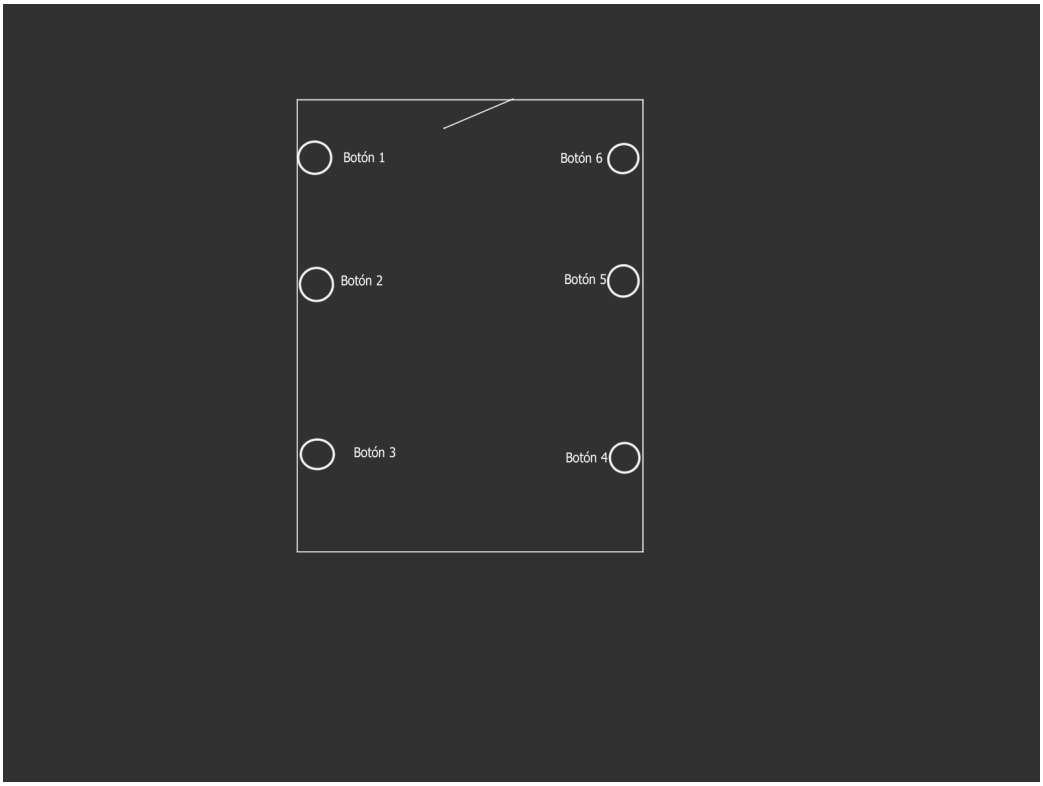

Figura 17: Boceto de la primera zona del nivel 2

A cada lado de la habitación hay tres botones explicativos, iguales que los del nivel 1. Al pulsarlos, explicarán un concepto correspondiente a la imagen que haya justo detrás de dicho botón.

Cuando se lean las seis explicaciones, se abrirá la puerta para la siguiente zona de este nivel. Al igual que en el anterior nivel, al pasar por la puerta se activará el autoguardado.

En la figura 18 se puede ver un boceto de la segunda parte del nivel.

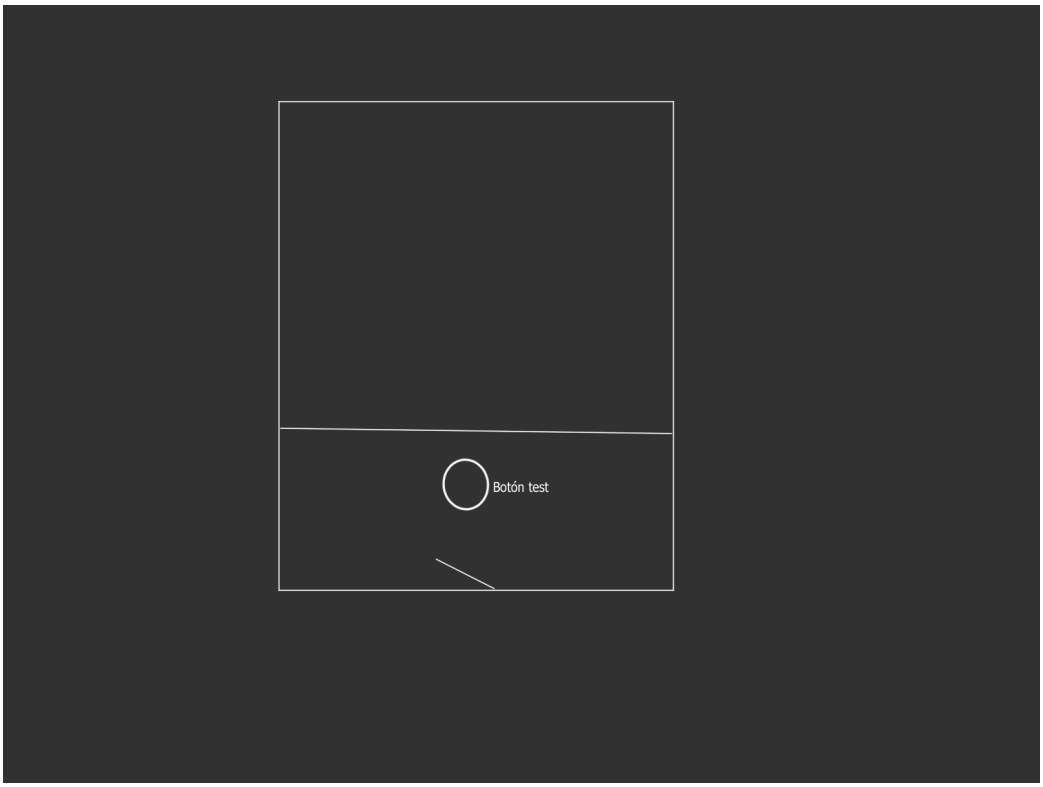

Figura 18: Boceto de la segunda zona del nivel 2

Hay un único botón, y al pulsarlo, la cámara apuntará al fondo de la habitación. En ese momento comenzará un diálogo de test. Cuando se plantee una pregunta, dará 3 opciones, ordenadas de forma aleatoria.

Al acertar todas las preguntas, una esfera se acercará al jugador, y cuando éste interactúe con la esfera, será transportado al nivel 3.

11.3. Nivel 3

El nivel 3 es el último nivel, y está dedicado a la programación por blueprints, original de Unreal Engine.

La primera zona es similar a la del anterior nivel. Varios botones explicativos en cada pared, cada uno con su imágen correspondiente en la pared. La sala será como se puede observar en el boceto de la figura 19.

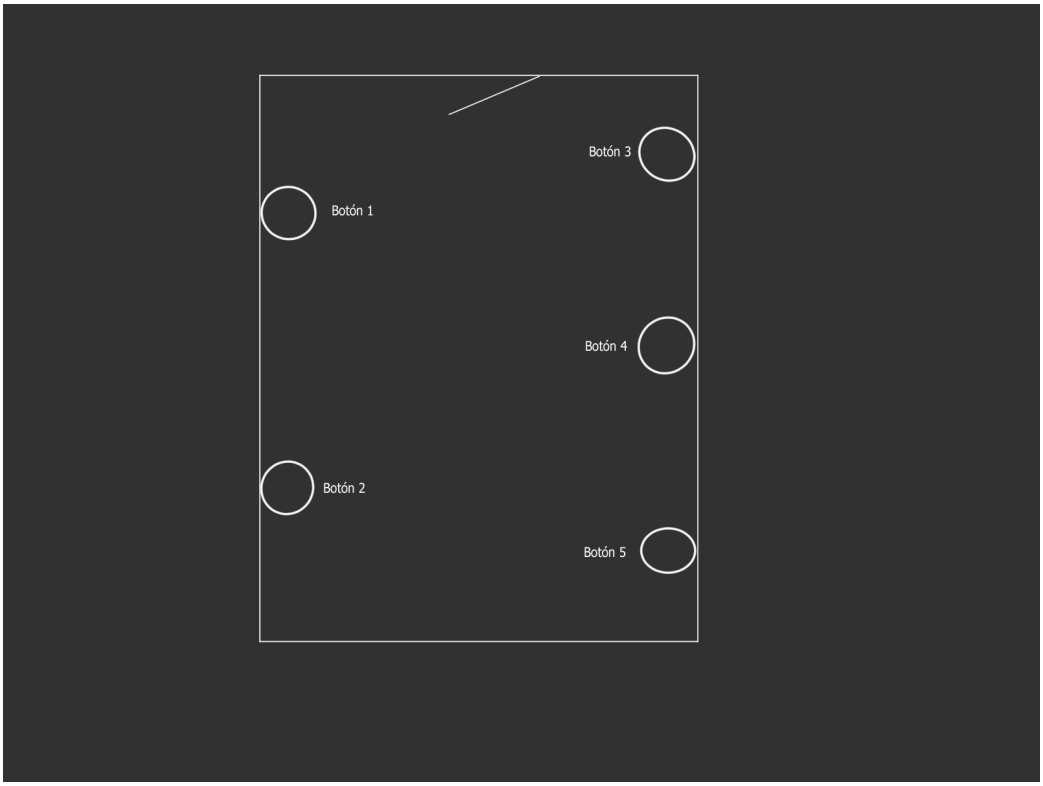

Figura 19: Boceto de la primera zona del nivel 3

Al igual que en el nivel anterior, cuando se lean todas las explicaciones, se abrirá la puerta para acceder a la segunda zona del nivel. Se puede ver un boceto de la habitación en la figura 20.

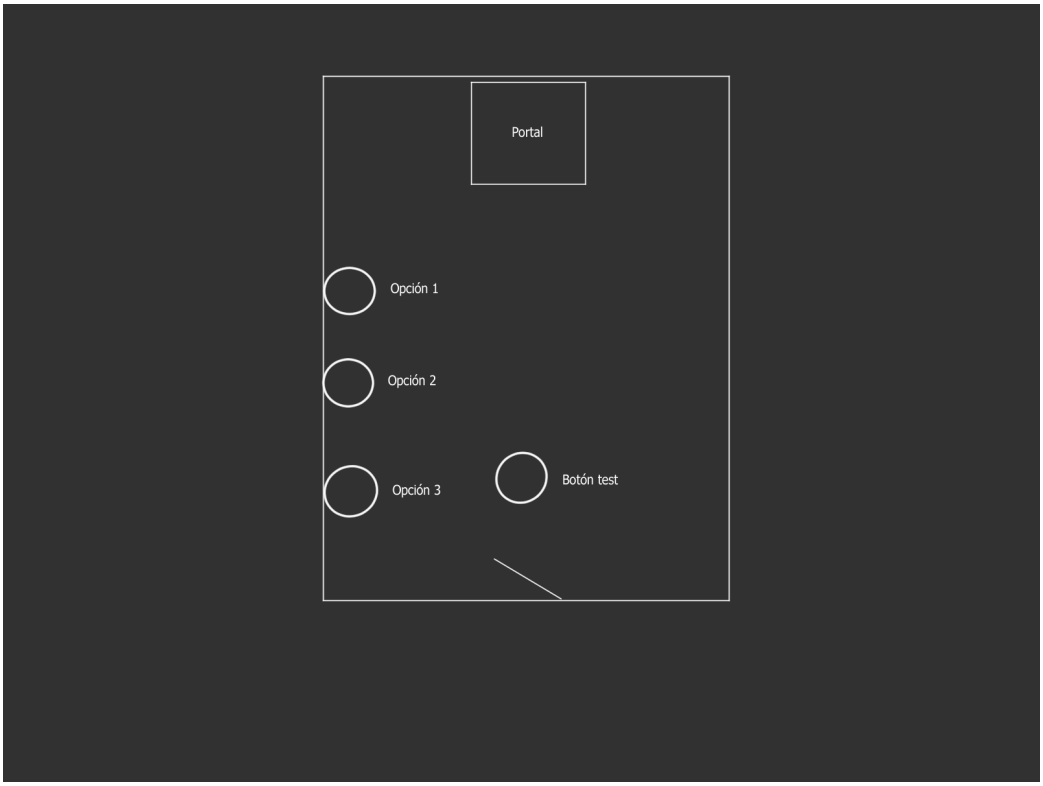

Figura 20: Boceto de la segunda zona del nivel 3

Aquí se encontrará un botón principal, en medio de la habitación. Este botón comenzará otro test, algo distinto. Al tener que elegir entre nodos «blueprints», para más comodidad, las opciones aparecerán en la pared. Cada opción tendrá su correspondiente botón.

Cuando se acierten todas las preguntas, se activa el portal del fondo. Al entrar en dicho portal, volverá al menú principal.

Capítulo 12

Diseño de la interfaz

Además de los niveles, se ha realizado un diseño de la interfaz que tendrá el tutorial. El diseño se realizará mediante bocetos, intentando que se asemejen lo mayor posible al resultado final.

El diseño de la interfaz se hará intentando que sea lo más intuitivo posible, para que el usuario vea todos los elementos rápidamente y sepa la función de cada uno.

12.1. Prototipo de la interfaz

Estos son los prototipos creados al comienzo del proyecto. Son bocetos descriptivos de qué tendrá cada ventana.

12.1.1. Menú principal

El menú principal sería simple y muy intuitivo. Constaría de una ventana, situaza en la mitad izquierda, con los siguientes botones:

- **Comenzar:** Al pulsar este botón, se iniciaría el tutorial desde el comienzo.

- **Cargar:** Se cargaría la partida desde el último autoguardado.
- **Controles:** Se abriría el listado de controles.
- **Opciones de pantalla:** Se abrirían las opciones de pantalla.
- **Salir:** Cerraría el sistema.

En la figura 21 se puede observar el boceto de este menú.

Figura 21: Boceto del menú principal

12.1.2. Menú de opciones de pantalla

Este es el menú que aparecería cuando pulsas el botón de opciones de pantalla en el menú principal. Sería sencillo y bastante intuitivo para el

usuario, y constaría de dos ventanas principales:

- **Resolución de pantalla:** ventana donde puedes elegir la resolución de pantalla que desees.
- **Modo de ventana:** ventana donde puedes elegir el modo de ventana que desees

Además, tendría un botón para volver al menú anterior.

En la figura 22 se puede observar el boceto de este menú.

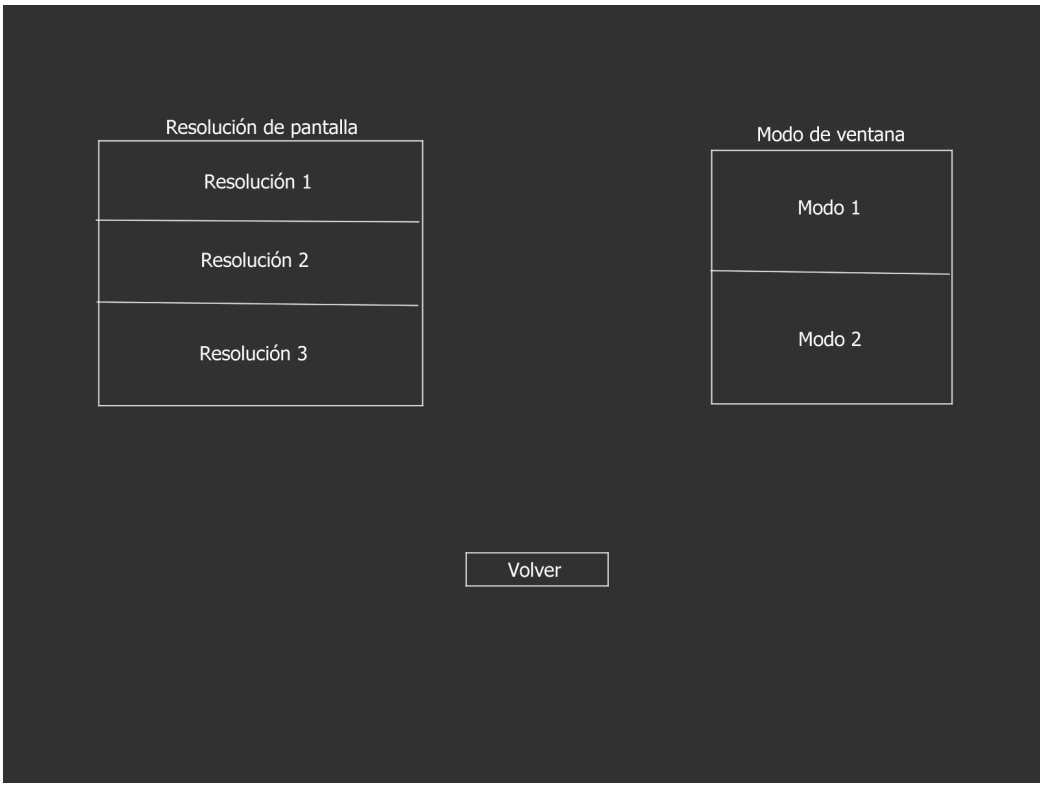

Figura 22: Boceto del menú de opciones de pantalla

12.1.3. Menú de controles

A este menú se podría acceder desde el menú principal o desde el menú de pausa.

Constaría de una ventana principal, con el listado de los controles del tutorial, y un botón para volver al menú anterior.

En la figura 23 se puede observar el boceto de este menú.

Figura 23: Boceto del menú de listado de controles

12.1.4. Menú de pausa

Este es el menú que aparece cuando pulsas el botón de pausa dentro de un nivel. De nuevo, sería muy sencillo y constaría únicamente de 3 botones:

- **Continuar:** Al pulsar este botón, se devolvería el control al personaje.
- **Controles:** Aparecería el listado de controles del tutorial.
- **Salir:** Saldría al menú principal.

En la figura 24 se puede observar el boceto de este menú.

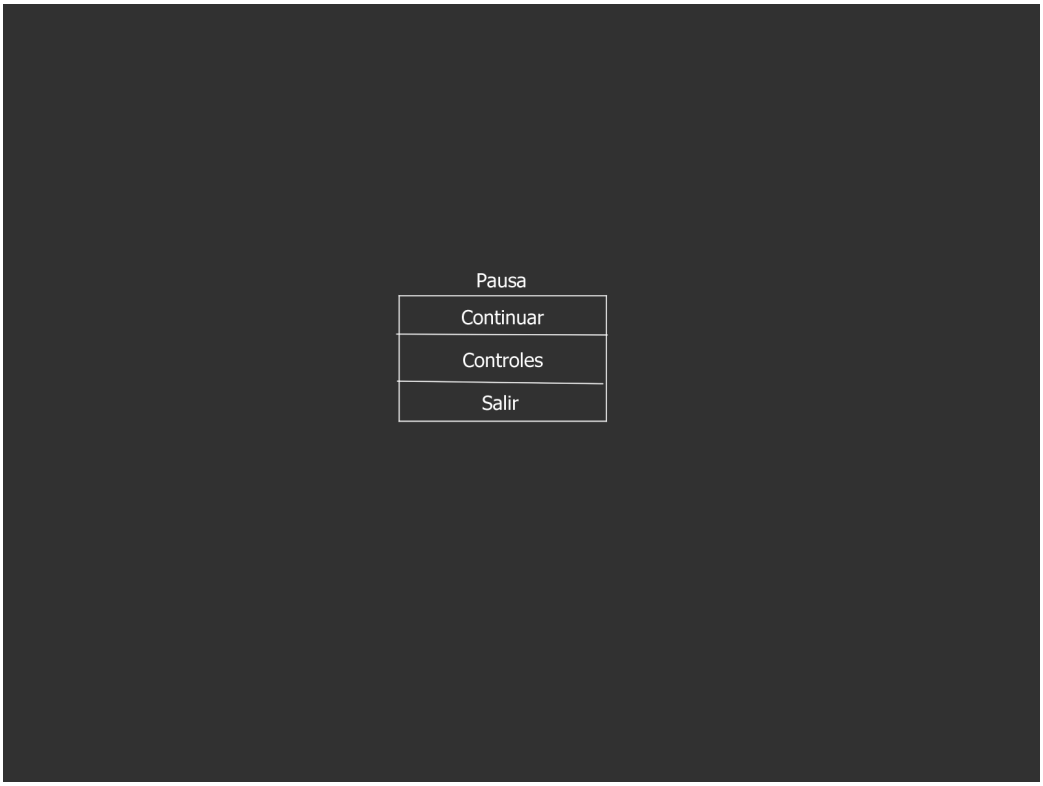

Figura 24: Boceto del menú de pausa

Parte V

PRUEBAS

Capítulo 13

Pruebas

Como en todo sistema, es necesario realizar una serie de pruebas para comprobar que funciona correctamente. Para ello, se comprobarán partes o módulos específicos para encontrar los posibles fallos que pudiera haber y, en el caso de que haya alguno, poder solucionarlo.

Con este apartado se busca garantizar en la mayor medida posible un funcionamiento adecuado del programa, sin errores de interfaz, errores gráficos o errores de comportamiento.

Como en todo proyecto software, hay dos tipos de pruebas:

- Pruebas de caja negra: se le llama así a aquel método que se usa para probar el software desconociendo la estructura interna del código o programa. Para este caso, usaremos como referencia los casos de uso definidos en el capítulo 8.

- Pruebas de caja blanca: son las pruebas de software en el que la persona que lo prueba conoce la estructura interna, y prueba el software, ya sea de manera genera, o una parte o módulo específicos.

13.1. Pruebas de caja blanca

En este apartado, dividiremos las distintas pruebas por niveles, para comprobar que cada uno de ellos funciona correctamente de manera individual, además del menú principal.

13.1.1. Menú principal

A continuación veremos los distintos casos que se pueden dar en el menú principal:

1. **El usuario pulsa el botón de comenzar:** El tutorial comienza desde el principio.
2. **El usuario pulsa el botón de cargar partida sin haber partida guardada:** No ocurre nada.
3. **El usuario pulsa el botón de cargar partida habiendo partida guardada:** Se carga la partida aguardada anteriormente.
4. **El usuario pulsa el botón de opciones de pantalla:** Aparece el menú de opciones de pantalla.
5. **El usuario pulsa el botón de controles:** Aparece el menú de controles.
6. **El usuario pulsa el botón de salir:** El programa se cierra.
7. **El usuario cambia la resolución de la pantalla a una diferente:** La resolución se cambia correctamente.
8. **El usuario cambia la resolución de la pantalla a la que ya tenía:** No ocurre nada.
9. **El usuario cambia el modo de ventana a uno diferente:** Cambia al nuevo modo de ventana.

10. **El usuario cambia el modo de ventana al que ya tenía:** No ocurre nada.

11. **El usuario pulsa el botón de volver desde la ventana de opciones de pantalla:** Vuelve a aparecer el menú principal.

12. **El usuario pulsa el botón de volver desde la ventana de controles:** Vuelve a aparecer el menú principal.

13.1.2. Nivel 1

A continuación veremos los distintos casos que se pueden dar en este nivel:

1. **El usuario interacciona con uno de los botones explicativos:** Se muestra el diálogo en pantalla correspondiente al objeto.

2. **El usuario pulsa el botón de agarrar mirando a la esfera:** Se coge la esfera.

3. **La esfera entra en contacto con el trigger:** La puerta se abre.

4. **El personaje pasa a través de la puerta:** Se activa el auto-guardado.

5. **El usuario intenta pasar al final del nivel sin haber pulsado el botón:** El personaje se transporta hacia atrás.

6. **El usuario intenta moverse cuando hay un diálogo en pantalla:** El personaje no se mueve.

7. **El usuario pulsa el botón de menú de pausa:** El menú de pausa se abre y el personaje se para.

8. **El usuario elige la opción de «Salir» en el menú de pausa:** Vuelve al menú principal.

9. **El usuario elige la opción de «Seguir» en el menú de pausa:** Vuelve a dar control al personaje y desaparece el menú de pausa.

13.1.3. Nivel 2

A continuación veremos los distintos casos que se pueden dar en este nivel:

1. **El personaje aparece en el nivel:** Se activa el auto-guardado.
2. **El usuario interacciona con uno de los botones explicativos:** Se muestra el diálogo en pantalla correspondiente a la imágen de la pared.
3. **El usuario interacciona con todos los botones explicativos:** Se abre la puerta a la segunda zona.
4. **El personaje pasa a través de la puerta:** Se activa el auto-guardado.
5. **El usuario interacciona con el botón de test:** Comienza el diálogo del test, se formula la primera pregunta y se muestran las imágenes de las distintas elecciones.
6. **El usuario elige la opción «TickComponent» en la primera pregunta:** Aparecen muchas esferas, se repite la pregunta y se muestra el diálogo correspondiente.
7. **El usuario elige la opción «Constructor» en la primera pregunta:** Se repite la pregunta y se muestra el diálogo correspondiente.
8. **El usuario elige la opción «BeginPlay» en la primera pregunta:** Se avanza a la siguiente pregunta y se muestra el diálogo correspondiente.
9. **El usuario elige las opciones «BeginPlay» o «TickComponent» en la segunda pregunta:** Se repite la pregunta y se muestra el diálogo correspondiente.

10. **El usuario elige la opción «Constructor» en la segunda pregunta:** Se avanza a la siguiente pregunta y se muestra el diálogo correspondiente.

11. **El usuario elige las opciones «UE_LOG» o «include» en la tercera pregunta:** Se repite la pregunta y se muestra el diálogo correspondiente.

12. **El usuario elige la opción «UPROPERTY» en la tercera pregunta:** Se muestra el diálogo correspondiente, se termina el test y la esfera mágica se acerca al jugador.

13. **El usuario intenta moverse cuando hay un diálogo en pantalla:** El personaje no se mueve.

14. **El usuario pulsa el botón de menú de pausa:** El menú de pausa se abre y el personaje se para.

15. **El usuario elige la opción de «Salir» en el menú de pausa:** Vuelve al menú principal.

16. **El usuario elige la opción de «Seguir» en el menú de pausa:** Vuelve a dar control al personaje y desaparece el menú de pausa.

El resultado de las pruebas ha sido el esperado, así que damos las pruebas por satisfechas.

13.1.4. Nivel 3

A continuación veremos los distintos casos que se pueden dar en este nivel.

1. **El personaje aparece en el nivel:** Se activa el auto-guardado.

2. **El usuario interacciona con uno de los botones explicativos:** Se muestra el diálogo en pantalla correspondiente a la imágen de la pared.

3. **El usuario interacciona con todos los botones explicativos:** Se abre la puerta a la segunda zona.

4. **El personaje pasa a través de la puerta:** Se activa el auto-guardado.

5. **El usuario interacciona con el portal final sin pasar el test:** No se termina el nivel.

6. **El usuario interacciona con el botón de test:** Comienza el diálogo del test y se muestra la primera pregunta. Además, aparecen en la pared las distintas opciones.

7. **El usuario elige las opciones 2 o 3 en la primera pregunta:** Se muestra el diálogo correspondiente de respuesta equivocada.

8. **El usuario elige la opción 1 en la primera pregunta:** Se muestra diálogo de respuesta correcta, se formula la siguiente pregunta y se cambian las imágenes de las opciones.

9. **El usuario elige las opciones 1 o 2 en la segunda pregunta:** Se muestra el diálogo correspondiente de respuesta equivocada.

10. **El usuario elige la opción 3 en la segunda pregunta:** Se muestra diálogo de respuesta correcta, se formula la siguiente pregunta y se cambian las imágenes de las opciones.

11. **El usuario elige cualquier opción en la tercera pregunta:** Se muestra diálogo de respuesta correcta y se termina el test, activando el portal del fondo de la habitación.

12. **El usuario intenta moverse cuando hay un diálogo en pantalla:** El personaje no se mueve.

13. **El usuario pulsa el botón de menú de pausa:** El menú de pausa se abre y el personaje se para.

14. **El usuario elige la opción de «Salir» en el menú de pausa:** Vuelve al menú principal.

15. **El usuario elige la opción de «Seguir» en el menú de pausa:** Vuelve a dar control al personaje y desaparece el menú de pausa.

16. **El usuario interacciona con el portal final:** Aparece el menú principal de nuevo.

El resultado de las pruebas ha sido el esperado, así que damos las pruebas por satisfechas.

13.2. Ejecución de pruebas de caja blanca

Después de realizar las pruebas detalladas en la sección 13.1, se han obtenido los siguientes resultados.

13.2.1. Menú principal

En el menú principal, después de probar todos los casos anteriormente detallados, se han encontrado los siguientes errores:

- Caso 2: Se genera un error y se cierra el programa.
    - Causa: intenta cargar un archivo que no existe.
- Caso 11: Aparece el menú principal pero siguen estando las opciones de modo de ventana.
    - Causa: el botón de volver no se había programado para ocultar esa ventana.

Estos errores fueron solucionados después de su detección. El resto de casos funcionaban correctamente.

13.2.2. Nivel 1

En este nivel, después de probar todos los casos anteriormente detallados, se han encontrado los siguientes errores:

- Caso 3: Al entrar en contacto la esfera con el «trigger», la puerta no se abría.
    - Causa: La esfera no estaba configurada para pueda activar eventos de superposición o «triggers».
- Caso 5: Si el usuario, sin haber pulsado el botón, intentaba pasar al otro lado en un momento específico y pegado a la pared, podía llegar a conseguirlo.
    - Causa: El trigger se movía incluso antes de pulsar el botón, permitiendo evitarlo.
- Caso 6: El usuario podía moverse mientras leía el diálogo del botón de la segunda zona.
    - Causa: Ese botón era un actor distinto al resto, al realizar alguna acción más, y no comunicaba correctamente que había un diálogo en pantalla.

Estos errores fueron solucionados después de su detección. El resto de casos funcionaban correctamente.

13.2.3. Nivel 2

En este nivel, después de probar todos los casos anteriormente detallados, se han encontrado los siguientes errores:

- Caso 2: Los diálogos de dos botones estaban intercambiados.
    - Causa: el ID de esos botones no era el correcto.

- Caso 4: El autoguardado no se activa correctamente.
    - Causa: El «trigger» no estaba correctamente añadido a la lista de «triggers» de autoguardado.
- Caso 12: La esfera mágica no se acerca al jugador.
    - Causa: La esfera no tenía el atributo de «móvil».
- Caso 13: Durante el test, el personaje, aunque no se viese, se podía mover.
    - Causa: No se indicaba al controlador que había un diálogo en pantalla.

Estos errores fueron solucionados después de su detección. El resto de casos funcionaban correctamente.

13.2.4. Nivel 3

En este nivel, después de probar todos los casos anteriormente detallados, se han encontrado los siguientes errores:

- Caso 5: El portal sí que terminaba el nivel sin haber realizado el test.
    - Causa: Estaba activado de manera predeterminada.
- Caso 6: Las imágenes no se muestran correctamente.
    - Causa: No se había renderizado correctamente la iluminación.

Estos errores fueron solucionados después de su detección. El resto de casos funcionaban correctamente.

13.3. Prueba de caja negra

Como se ha explicado, se especificará una prueba de caja negra por cada caso de uso anteriormente desarrollado. De esta manera, probaremos cada apartado del software, sin tener en cuenta cómo actúa interiormente.

13.3.1. Pruebas del caso de uso 1: Pulsar botón explicativo

Prueba	Prueba del C.U. 1: Pulsar botón explicativo
Descripción	Se pretende probar que al interactuar con un botón explicativo se muestre correctamente la explicación
Entrada	Interactuar con uno de los botones explicativos
Salida	El diálogo se habrá mostrado en su totalidad y desaparecerá la interfaz
Condiciones	El usuario deberá estar en frente de uno de los botones

Cuadro 8: Pruebas caso de uso 1: Pulsar botón explicativo

13.3.2. Pruebas del caso de uso 2: Realizar test

Prueba	Prueba del C.U. 2: Realizar test
Descripción	Se pretende probar que el test funciona correctamente de principio a fin
Entrada	Interactuar con un botón de test
Salida	El test habrá sido realizado y desaparecerá la interfaz
Condiciones	El usuario deberá estar en frente de un botón de test

Cuadro 9: Pruebas caso de uso 2: Realizar test

13.3.3. Pruebas del caso de uso 3: Agarrar objeto

Prueba	Prueba del C.U. 3: Agarrar objeto
Descripción	Se pretende probar que se agarran los objetos correctamente
Entrada	Pulsar el botón de coger frente a un objeto y mover el personaje
Salida	El objeto va junto al personaje
Condiciones	El usuario deberá estar en frente de un botón de test

Cuadro 10: Pruebas caso de uso 3: Agarrar objeto

13.3.4. Pruebas del caso de uso 4: Cambiar opciones de pantalla

Prueba	Prueba del C.U. 4: Cambiar opciones de pantalla
Descripción	Se pretende probar que al cambiar una opción de pantalla, esta se aplica correctamente
Entrada	Seleccionar una nueva resolución de pantalla o modo de ventana
Salida	La pantalla ha cambiado a la nueva resolución de pantalla o modo de ventana
Condiciones	El usuario deberá estar en el menú de opciones de pantalla

Cuadro 11: Pruebas caso de uso 4: Cambiar opciones de pantalla

13.3.5. Pruebas del caso de uso 5: Cargar partida

Prueba	Prueba del C.U. 5: Cargar partida
Descripción	Se pretende probar que el autoguardado funcione correctamente, tanto cargar como guardar
Entrada	El usuario mueve al personaje por una zona de autoguardado y luego carga la partida
Salida	El personaje aparecerá en el nivel que se encontraba y en la misma posición
Condiciones	El usuario deberá estar dentro de un nivel

Cuadro 12: Pruebas caso de uso 5: Cargar partida

13.4. Ejecución de pruebas de caja negra

Las pruebas de caja negra han dado todas los resultados esperado. Esto es debido a que las pruebas de caja blanca han sido suficientemente consistentes y se han aplicado de manera correcta.

Parte VI

CONCLUSIONES

Capítulo 14

Conclusiones

En este apartado se expondrán las conclusiones que se han obtenido a lo largo de la realización de este proyecto.

14.1. Conclusiones

Para realizar unas conclusiones sobre el proyecto, se partirán de los objetivos formulados en el capítulo 3, viendo si se han cumplido correctamente.

14.1.1. Conclusiones de objetivos formales

- Se han adquirido conocimientos amplios de manera genérica del motor gráfico Unreal Engine, tanto del diseño como de la programación. Se ha conseguido llegar a manejar de manera fluida y cómoda las herramientas que éste proporciona al usuario.
- Se ha aprendido a programar en el entorno de programación Visual Studio y a sacarle partido a las funcionalidades que este tiene, así como su integración con Unreal Engine.

- Se han aprendido conceptos de progración del ámbito de los videojuegos, tales como el uso de disparadores, física de los actores, instancias de juego y muchos más.
- Se han aplicado los conocimientos aprendidos en el Grado. Al haber programado con C++, que es el lenguaje que más se ha manejado durante el grado, se ha podido demostrar que se es capaz de poner en práctica correctamente.

14.1.2. Conclusiones de objetivos específicos

- Se han diseñado y desarrollado los niveles que componen el tutorial, y se han situado los elementos de manera que sea agradable para el usuario.
- Se ha desarrollado un control cómodo e intuitivo para el usuario. Hay pocos controles, y son intuitívos para el usuario. Además, el control del personaje es simple y no requiere ninguna habilidad.
- Se ha diseñado una base de datos robusta, en fase normal 1, para los diálogos del tutorial, y cumple su función correctamente.
- La interfaz se diseñado de manera que sea intuitiva. Hay pocos botones en cada menú y se muestran de forma clara para que el usuario intuya perfectamente la utilidad de cada nno.
- Se ha elaborado la documentación, dividida en memoria técnica, manual de código y manual de usuario. La documentación es completa y define perfectamente el proyecto.

Capítulo 15

Futuras mejoras

Además de los objetivos que se han planteado para este proyecto, se han encontrado nuevas ideas con las que, en un futuro, poder ampliarlo y mejorarlo. Estas son:

- Otro nivel para términos matemáticos que se aplican en Unreal Engine.
- Mejorar el apartado visual del tutorial.
- Poder tener varias partidas guardadas, con nombre modificable por el usuario.
- Desarrollar más opciones modificables en el menú además de las opciones de pantalla ya implementadas.

Bibliografía

[1] *AsÃ se crearon los principales motores grÃ¡ficos de la historia del videojuego*. https://www.vidaextra.com/n/asi-se-crearon-los-principales-motores-graficos-de-la-historia-del-videojuego. 2018. Fecha de acceso: 12-04-2021.

[2] *First-Person Perspective*. https://www.giantbomb.com/first-person-perspective/3015-330/. 2019. Fecha de acceso: 10-04-2021.

[3] *NIMATRON: An Early Electromechanical Machine to Play the Game of Nim*. http://www.historyofinformation.com/detail.php?entryid=4472. 2020. Fecha de acceso: 18-04-2021.

[4] *RPG Maker MV: Introduction and History*. https://moegamer.net/2016/08/03/rpg-maker-mv-introduction-and-history/. 2016. Fecha de acceso: 10-04-2021.

[5] *The History of Gaming: An Evolving Community*. https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/. 2020. Fecha de acceso: 09-04-2021.

[6] *This is Unreal*. https://hotgates.eu/this-is-unreal/. 2018. Fecha de acceso: 11-04-2021.

[7] *Unreal Engine*. https://www.unrealengine.com/en-US/. 2021. . Fecha de acceso: 11-04-2021.