

Implementación de un tutorial interactivo de programación con Unreal Engine en forma de videojuego puzzle en primera persona.

Rafael Moreno Serrano

1 de noviembre de 2021

Índice general

I	Introducción	5
1.	Introducción	6
2.	Definición del problema	7
2.1.	Identificación del problema real	7
2.2.	Identificación del problema técnico	7
2.2.1.	Funcionamiento	8
2.2.2.	Entorno	8
2.2.3.	Vida esperada	8
2.2.4.	Ciclo de mantenimiento	8
2.2.5.	Competencia	9
2.2.6.	Aspecto externo	9
2.2.7.	Estandarización	9
2.2.8.	Calidad y fiabilidad	9
2.2.9.	Programación de tareas	10
2.2.10.	Pruebas	10
2.2.11.	Seguridad	11
3.	Objetivos	12
3.1.	Objetivo principal	12
3.2.	Objetivos específicos	12
3.3.	Objetivos formales	12
4.	Antecedentes	13
4.1.	Origen de los videojuegos	13
4.2.	Historia de Unreal Engine	14
4.3.	Unreal Engine	15
4.4.	Proyectos similares	16

ÍNDICE GENERAL	2
4.4.1. Dreams	16
4.4.2. Content Examples from Unreal Engine 4	17
5. Restricciones	19
5.1. Factores dato	19
5.2. Factores estratégicos	19
6. Recursos	21
6.1. Recursos Hardware	21
6.2. Recursos Software	21
6.3. Recursos humanos	22
II ANALISIS Y ESPECIFICACIÓN DE REQUISITOS	23
7. Análisis y especificación de requisitos	24
7.1. Requisitos de información	24
7.2. Requisitos funcionales	26
7.3. Requisitos no funcionales	27
III MODELADO DE REQUISITOS	28
8. Analisis funcional	29
8.1. Caso de uso 0. Contexto del programa	30
9. Análisis dinámico	31
9.1. Diagramas de secuencia	32
9.1.1. Diagrama de secuencia de botón explicativo	32
IV DISEÑO DEL SISTEMA	33
10. Diseño de niveles	34
10.1. Nivel 1	34
10.2. Nivel 2	38
10.3. Nivel 3	41

<i>ÍNDICE GENERAL</i>	3
V PRUEBAS	45
11. Pruebas	46
11.1. Prueba de caja negra	46
11.2. Pruebas de caja blanca	47
11.2.1. Nivel 1	47
11.2.2. Nivel 2	47
11.2.3. Nivel 3	48
VI CONCLUSIONES	50
12. Conclusiones y futuras mejoras	51
12.1. Conclusiones	51

Índice de figuras

1.	Marketplace de Unreal Engine	16
2.	Captura de Dreams	17
3.	Content Examples from Unreal Engine 4 en el editor	18
4.	Caso de uso: Contexto del Sistema	30
5.	Diagrama de secuencia de botón de explicativo	32
6.	Aspecto de la zona 1 del nivel 1	35
7.	Trigger del nivel 1 para pasar a la siguiente zona	36
8.	Segunda zona del nivel 1	37
9.	Trigger de la cámara activado en el nivel 1	38
10.	Aspecto de la primera zona del nivel 2	39
11.	Aspecto de la segunda zona del nivel 2	40
12.	Aspecto del test del nivel 2	41
13.	Aspecto de la primera zona del nivel 3	42
14.	Aspecto de la segunda zona del nivel 3	43
15.	Aspecto de las opciones del test del nivel 3	44

Parte I

Introducción

Capítulo 1

Introducción

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 examinar 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, siendo 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 a cabo 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 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 proyecto 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 específicos

Se deberán cumplir los siguientes objetivos específicos:

3.3. 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

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 puzzle o "The Elder Scrolls: Arena" (1994) como videojuego de aventura rol. Este proyecto será de tipo puzzle, el cual alcanzó una fama enorme con el lanzamiento de "Portal" (2007) que popularizó el género de puzzle 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"[6]. 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[7]. 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 [8]. 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, cinematográfica, 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.

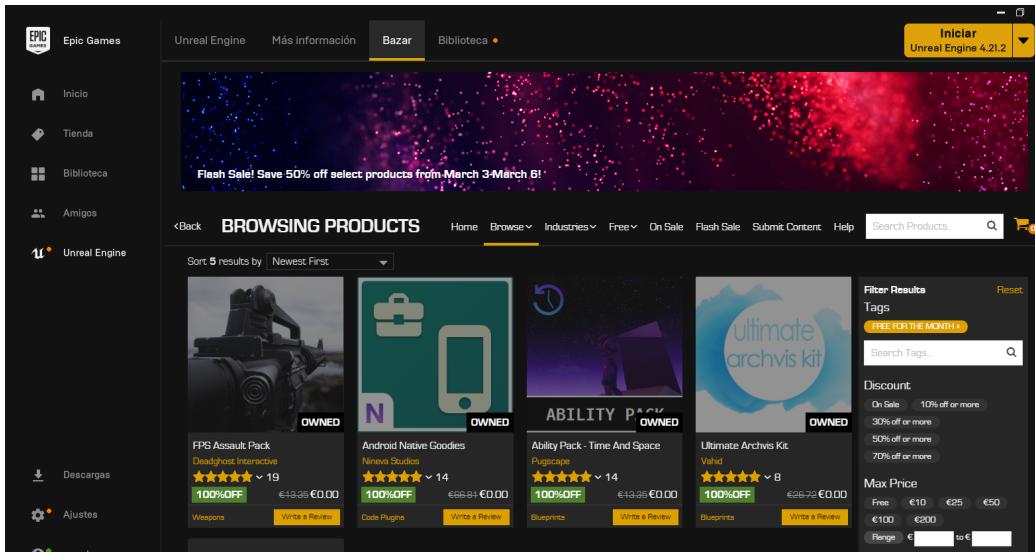


Figura 1: Marketplace de Unreal Engine

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 comprobarlo. 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.

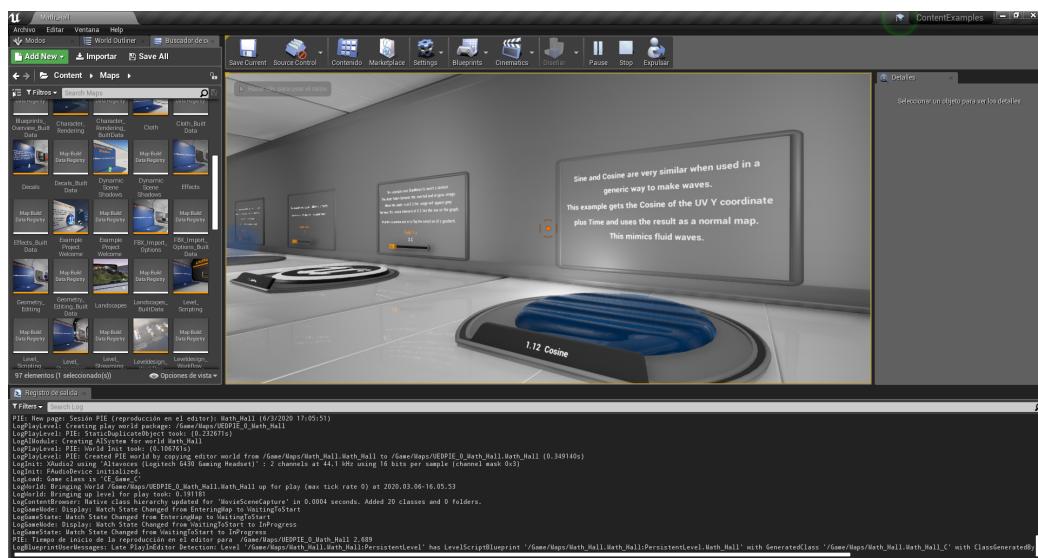


Figura 3: Content Examples from Unreal Engine 4 en el editor

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 gratuito, 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 gratuita.

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 i5-7400 @ 3.00 GHz
- 16GB de memoria RAM
- Unidad HDD de 1TB de capacidad
- Unidad SSD 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, estudiante 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:

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 abordará de manera más amplia más adelante.

Estos requisitos 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 podrá interactuar con cualquier botón que haya en el escenario, estando éstos señalizados debidamente
- **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 se realiza con diagramas de casos de uso. Los diagramas de caso de uso es un diagrama UML con el que se representan procesos empresariales, así como sistemas y procesos de programación orientada a objetos, siendo este último nuestro caso.

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

En los diagramas de caso de uso se busca garantizar que sean comprensibles para todo el mundo de un vistazo, así que se utilizan elementos estandarizados para elaborarlo. Estos son los siguientes:

- Actor: tanto si es una persona como un sistema, se representa con el dibujo de una figura humana esquemática.
- Sistema: el sistema al que se refiere el caso de uso tiene forma de rectángulo.
- Caso de uso: se muestra como una elipse que suele incluir un texto describiendo brevemente el proceso.

La relación entre estos elementos se representará con unas líneas de conexión llamadas asociaciones.

Además, dispondremos de una tabla de información para cada 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:

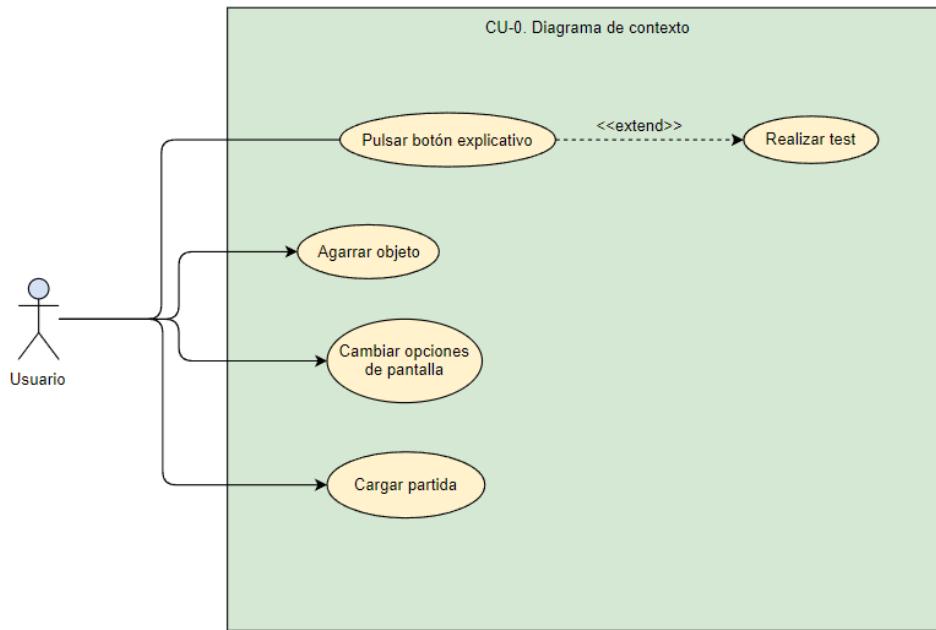


Figura 4: Caso de uso: Contexto del Sistema

Caso de Uso:	Caso de uso 0: Diagrama de contexto
ID:	1
Descripción:	Este diagrama muestra las distintas elecciones que se p...
Casos de uso:	CU-1. Pulsar botón explicativo: El usuario pulsa un botón para i... CU-1.5. Realizar test: El usuario pulsa un botón explicativo ... CU-3. Agarrar objeto: El usuario coge un objeto para lle... CU-3. Cambiar opciones de pantalla: El usuario cambia la resolución de l... CU-4. Cargar partida: El usuario selecciona esta opción pa...
Flujo principal:	

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 de botón explicativo

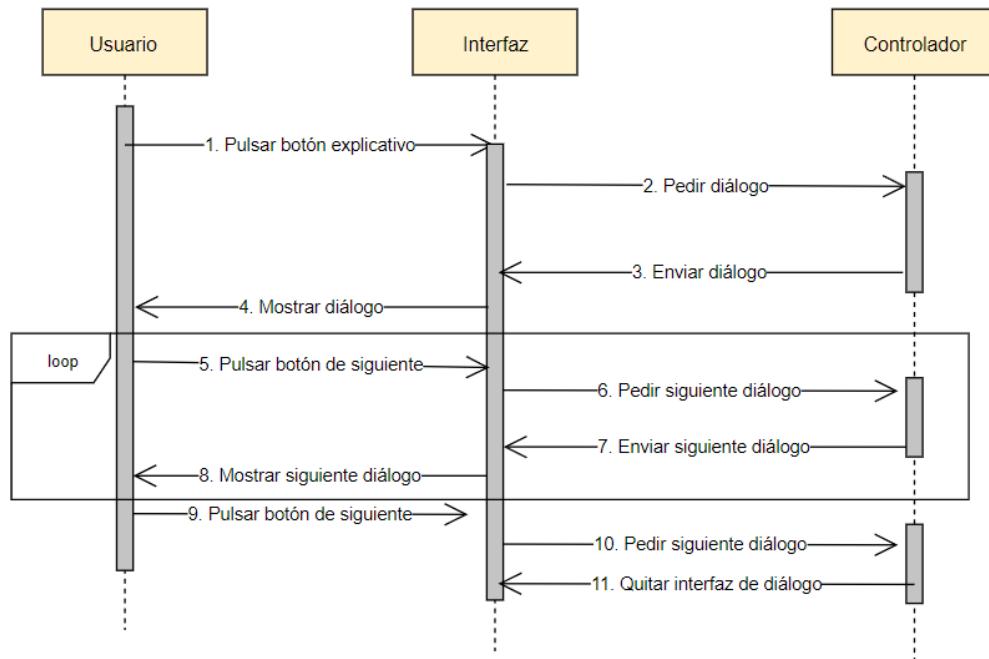


Figura 5: Diagrama de secuencia de botón de explicativo

Parte IV

DISEÑO DEL SISTEMA

Capítulo 10

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

10.1. 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 la figura 6



Figura 6: Aspecto de la zona 1 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 Uri.

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, como se ve en la figura 7 en el que tienes que soltar una esfera para abrir la puerta a la siguiente zona

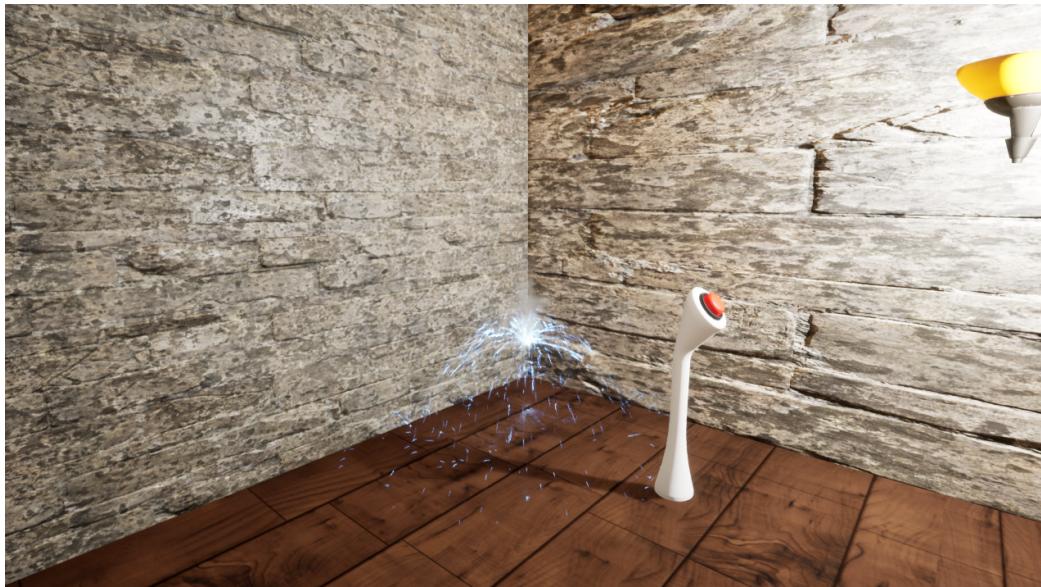


Figura 7: Trigger del nivel 1 para pasar a la siguiente zona

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

La siguiente zona, al comienzo, tiene el aspecto de la figura



Figura 8: 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, y queda de la manera que se ve en la figura9



Figura 9: Trigger de la cámara activado en el nivel 1

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

10.2. Nivel 2

Al aparecer en el nivel 2, vas a ver la habitación que se muestra en la figura 10. Este nivel está basado en enseñar conceptos de C++ específicos de Unreal Engine.



Figura 10: Aspecto 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, te 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.

La segunda zona tiene el aspecto que se aprecia en la figura 11.

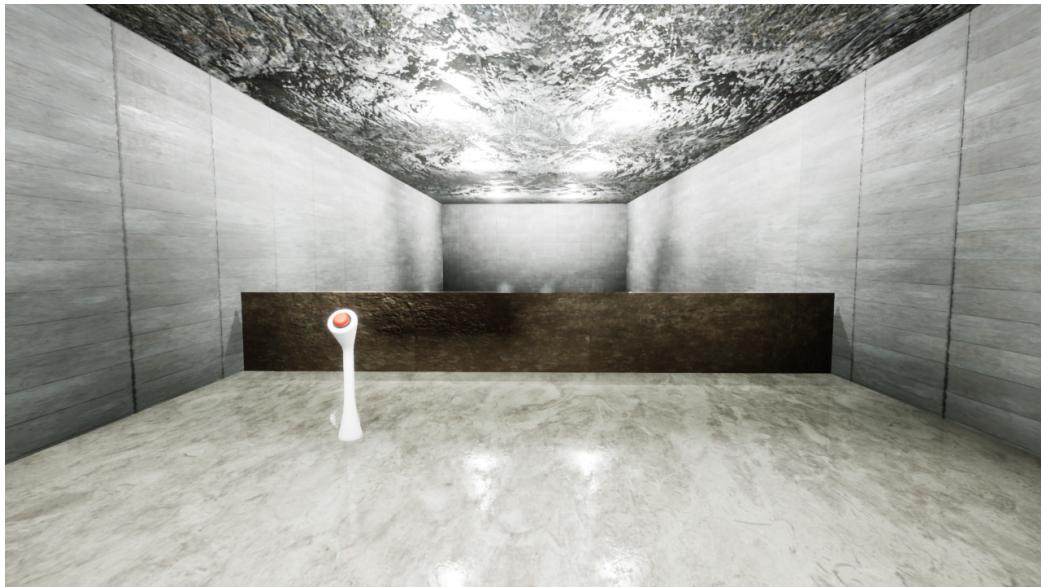


Figura 11: Aspecto de la segunda zona del nivel 2

Hay un único botón, y al pulsarlo, la cámara apuntará al fondo de la habitación. En ese momento comenzará un diálogo de test. Cuando te planteé una pregunta, te dará 3 opciones, ordenadas de forma aleatoria, que se verán como en la figura 12



Figura 12: Aspecto del test del nivel 2

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

10.3. 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 imagen correspondiente en la pared. La sala es como se puede observar en la figura 13.



Figura 13: Aspecto 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, la cual tiene el aspecto de la figura 14.



Figura 14: Aspecto 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. Se verá como en la figura 15



Figura 15: Aspecto de las opciones del test del nivel 3

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

Parte V

PRUEBAS

Capítulo 11

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 general, o una parte o módulo específicos. En

11.1. 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 interíormente.

11.2. 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.

11.2.1. Nivel 1

CASO	DESCRIPCIÓN	
1	El usuario interacciona con uno de los botones explicativos	Se muestra
2	El usuario pulsa el botón de agarrar mirando a la esfera	
3	La esfera entra en contacto con el trigger	
4	El personaje pasa a través de la puerta	
5	El usuario intenta pasar al final del nivel sin haber pulsado el botón	
6	El usuario intenta moverse cuando hay un diálogo en pantalla	
7	El usuario pulsa el botón de menú de pausa	El n
8	El usuario elige la opción de «Salir» en el menú de pausa	
9	El usuario elige la opción de «Seguir» en el menú de pausa	Vuelve a dar

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

11.2.2. Nivel 2

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

CASO	DESCRIPCIÓN
1	El personaje aparece en el nivel
2	El usuario interacciona con uno de los botones explicativos
3	El usuario interacciona con todos los botones explicativos
4	El personaje pasa a través de la puerta
5	El usuario interacciona con el botón de test
6	El usuario elige la opción «TickComponent» en la primera pregunta
7	El usuario elige la opción «Constructor» en la primera pregunta
8	El usuario elige la opción «BeginPlay» en la primera pregunta
9	El usuario elige las opciones «BeginPlay» o «TickComponent» en la segunda pregunta
10	El usuario elige la opción «Constructor» en la segunda pregunta
11	El usuario elige las opciones «UE_LOG» o «include» en la tercera pregunta
12	El usuario elige la opción «UPROPERTY» en la tercera pregunta
13	El usuario intenta moverse cuando hay un diálogo en pantalla
14	El usuario pulsa el botón de menú de pausa
15	El usuario elige la opción de «Salir» en el menú de pausa
16	El usuario elige la opción de «Seguir» en el menú de pausa

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

11.2.3. Nivel 3

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

CASO	DESCRIPCIÓN	
1	El personaje aparece en el nivel	
2	El usuario interacciona con uno de los botones explicativos	
3	El usuario interacciona con todos los botones explicativos	
4	El personaje pasa a través de la puerta	
5	El usuario interacciona con el botón de test	
6	El usuario elige las opciones 2 o 3 en la primera pregunta	
7	El usuario elige la opción 1 en la primera pregunta	Se muestra diálogo
8	El usuario elige las opciones 1 o 2 en la segunda pregunta	
9	El usuario elige la opción 3 en la segunda pregunta	Se muestra diálogo
10	El usuario intenta moverse cuando hay un diálogo en pantalla	
11	El usuario pulsa el botón de menú de pausa	
12	El usuario elige la opción de «Salir» en el menú de pausa	
13	El usuario elige la opción de «Seguir» en el menú de pausa	
14	El usuario interacciona con el portal final	

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

Parte VI

CONCLUSIONES

Capítulo 12

Conclusiones y futuras mejoras

En este apartado se expondrán las conclusiones que se han obtenido a lo largo de la realización de este proyecto, así como las futuras mejoras a las que está abierto el mismo si así fuese necesario.

12.1. Conclusiones