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

Rafael Moreno Serrano

28 de septiembre de 2021

Índice general

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 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 dispositivo 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 específicos, es posible que siga siendo útil en las versiones venideras. Es por ello que realmente la vida esperada del producto es incierta

2.2.4. Ciclo de mantenimiento

(empezar por lo positivo «se prestará a modificaciones y nuevos añadidos si así se viese necesario») El programa no necesitará de un mantenimiento periódico y solo se tratará en el caso de que éste contenga errores

que sean descubiertos posteriormente. Aun así, se prestará a modificaciones y nuevos añadidos si así se viese necesario

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 o «controller»

2.2.7. Estandarización

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

2.2.10. Pruebas

2.2.11. Seguridad

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.

3.2. 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 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"[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, 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...

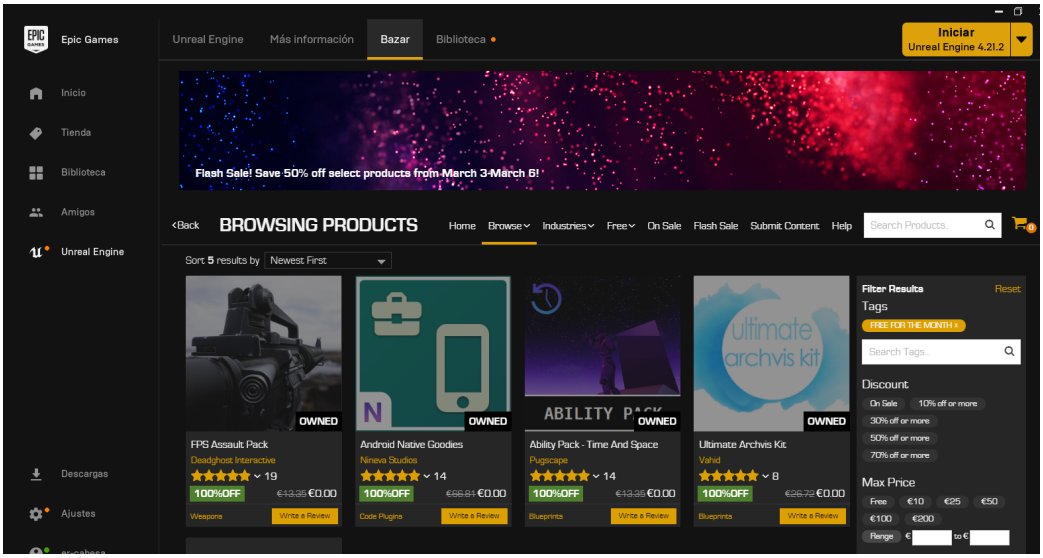

Figura 1: Marketplace de Unreal Engine

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.

4.4. Proyectos similares

4.4.1. Dreams

Si hablamos de tutoriales interactivos sobre desarrollo de videojuegos, un ejemplo muy reciente es "Dreams", un juego 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

Figura 2: Dreams

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.

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, 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

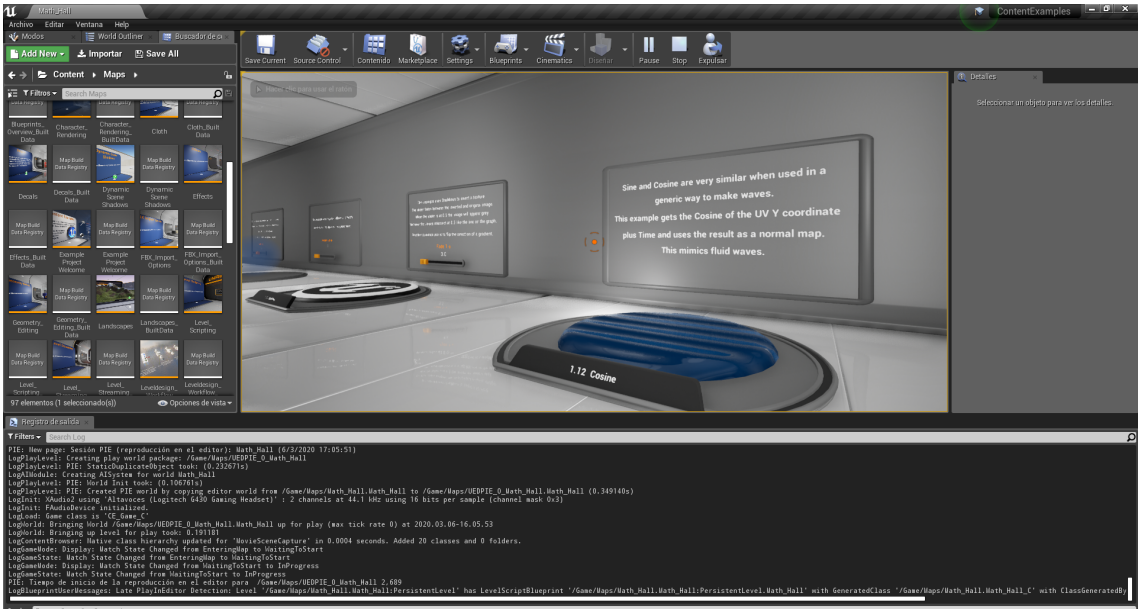

Figura 3: Content Examples from Unreal Engine 4 en el editor

y poder interactuar con ello en tiempo real.

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.
- 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
- El software usado en todo momento es totalmente gratuíto, tanto el entorno de programación Visual Studio como el propio motor gráfico Unreal Engine

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 metodo 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

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:

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 (Requisitos Funcionales):

- **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 decidir en cualquier momento si cerrar el proyecto, y éste deberá dar una opción apropiada para ello.
- **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, 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 (Requisitos No Funcionales):

- **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 en cuanto 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

En este nivel se explicarán los conceptos básicos del plano físico del editor de Unreal Engine. Se hablará de colisión, transparencia, disparadores, texturas... El nivel se dividirá en 2 habitaciones. Ambas habitaciones tienen un estilo rústico, con unos cuantos muebles decorativos y unos elementos a tratar que serán claramente diferenciables del resto, para captar la atención del usuario:

Nada más comenzar el nivel se enseñará al usuario unos cuantos conceptos de Unreal Engine por pantalla. Entre ellos los tres tipos de físicas que puede tener un actor que son los siguiente: sólido, transparente o disparador. Para mostrarlo, delante del punto de aparición habrá dos cubos contiguos, uno será atravesable y el otro no. Además tendrán texturas a una escala diferente para explicar el escalado de textura. Justo en frente de cada cubo habrá un botón. Al pulsar el botón explicará el estado respectivo del cubo que se encuentre delante.

Lo siguiente será la descripción de un trigger, que será el concepto más importante de esta parte. Para ello, en una esquina la cual se indicará de forma llamativa, habrá otro botón explicativo. Éste tratará la explicación de cómo seleccionar mediante código un

En esta segunda habitación habrá un claro ejemplo práctico de disparador o «trigger». Éste será la figura del vigilante. En un principio, el acompañante avisa al usuario de la dificultad de pasar hacia el otro lado, y tras un intento, el rango de visión se hará visible. Una vez hecho, se dará la posibilidad de modificar el tamaño del campo de visión para mostrar visualmente la aplicación de este y poder pasar a la siguiente fase

Este nivel comenzará en una sala con diversas pizarras mostrando par-

tes de código blueprint. La finalidad principal de esta zona será la de demostrar que los blueprints es otro tipo de programación y, aunque sea más gráfica, sigue siendo necesario tener conceptos claros de programación. La principal ventaja es la de la claridad visual de los eventos y la posibilidad de ver el orden de acción de los mismos.

Parte IV

CONCLUSIONES

Capítulo 9

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 futuas mejoras a las que está abierto el mismo si así fuese necesario.

9.1. Conclusiones