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

Rafael Moreno Serrano

22 de agosto de 2021

Ã■ndice general

Š=Š=Š=Š=Š=Š=Š=Š=Š=Š=Š=Š=Š=Š=Š=Š= **I Introducción** **3**

1. Introducción **4**
1.1. Ficheros 4

2. Código C++ **5**
2.1. Clase *AbrirPuerta* 5
2.1.1. Función *BeginPlay* 5
2.1.2. Función *TickComponent* 6
2.2. Identificación del problema real 7
2.3. Identificación del problema técnico 7
2.3.1. Funcionamiento 7
2.3.2. Entorno 8
2.3.3. Vida esperada 8
2.3.4. Ciclo de mantenimiento 8
2.3.5. Competencia 8
2.3.6. Aspecto externo 8
2.3.7. Estandarización 9
2.3.8. Calidad y fiabilidad 9
2.3.9. Programación de tareas 9
2.3.10. Pruebas 9
2.3.11. Seguridad 9

3. Objetivos **10**
3.1. Objetivo principal 10
3.2. Objetivos específicos 10
3.3. Objetivos formales 10

4. Antecedentes **11**

5. Restricciones **12**
5.1. Factores Dato 12
5.2. Factores estratégicos 13

6. Recursos **14**
6.1. Recursos Hardware 14
6.2. Recursos Software 14
6.3. Recursos humanos 15

II ANALISIS Y ESPECIFICACIÓN DE REQUISITOS 16

7. Análisis y especificación de requisitos **17**

8. Requisitos de información **18**

9. Requisitos funcionales **19**

10. Requisitos no funcionales **20**

III MODELADO DE REQUISITOS 21

11. Analisis funcional **22**

Parte I

Introducción

CapÃ■tulo 1

Introducción

En este documento se mostrará todo el código diseñado y realizado para el proyecto.

El código principal está realizado en el lenguaje C++, y tiene las características y funciones propias del motor gráfico Unreal Engine, ya que en todo momento se codifica en base a éste. Así mismo, se mostrará los Blueprints creados, al ser un lenguaje de script como tal, y se tratará de la misma manera.

Así, este documento se divide en las funciones creadas con C++ y con Blueprints, siendo a su vez separadas por clases y/o documentos.

1.1. Ficheros

CapÃ■tulo 2

Código C++

A continuación mostrará cada una de las clases codificadas en lenguaje C++:

2.1. Clase *AbrirPuerta*

Esta clase es creada para poder abrir una clase por medio de un *trigger* de superposición, y que ésta se cierre al cabo de un tiempo.

2.1.1. Función *BeginPlay*

Esta función se activa al comienzo, y guarda en variables el actor que tendrá la capacidad de abrir la puerta (en este caso el jugador), y la rotación inicial de la puerta.

```
// Called when the game starts
void UAbrirPuerta::BeginPlay()
{
    Super::BeginPlay();
    Owner = GetOwner();

    ActorThatOpens = GetWorld()->GetFirstPlayerController()->GetPawn()
    actorRotation = Owner->GetActorRotation();

}
```

2.1.2. Función *TickComponent*

Esta función comprueba en cada tick si el actor está activando el trigger. En caso positivo la puerta se abre y se guarda el momento en el que se ha abierto. Una vez pasado un tiempo, la puerta se cerrará.

```
void UAbrirPuerta::TickComponent(float DeltaTime, ELevelTick TickType, FAct
{
    Super::TickComponent(DeltaTime, TickType, ThisTickFunction);
    if (PressurePlate->IsOverlappingActor(ActorThatOpens))
    {
        OpenDoor();
        LastDoorOpenTime = GetWorld()->GetTimeSeconds();

    }

    if (GetWorld()->GetTimeSeconds() - LastDoorOpenTime > DoorCloseDelay)
        CloseDoor();
    }
}
```

2.2. 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.

2.3. 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:

2.3.1. 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

2.3.2. 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

2.3.3. 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

2.3.4. 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

2.3.5. 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

2.3.6. Aspecto externo

El programa tendrá el aspecto de un videojuego al uso, controlándose mediante teclado o «controller»

2.3.7. Estandarización

2.3.8. 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

2.3.9. Programación de tareas

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

2.3.10. Pruebas

2.3.11. Seguridad

(poner que no se ha contemplado tener ningún tipo de seguridad)
Preguntar estos dos apartados

CapÃ■tulo 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

CapÃ■tulo 4

Antecedentes

CapÃ■tulo 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

CapÃ■tulo 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

CapÃ■tulo 7

Análisis y especificación de requisitos

CapÃ■tulo 8

Requisitos de información

CapÃ■tulo 9

Requisitos funcionales

CapÃ■tulo 10

Requisitos no funcionales

Parte III

MODELADO DE REQUISITOS

CapÃ■tulo 11

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.

La parte interactiva