Capítulo 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

Capítulo 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*

Ésta función se activa al comienzo del nivel y guarda el personaje usado por el jugador como el actor que puede abrir la puerta.

```
1 void UAbrirPuerta::BeginPlay()
2 {
3   Super::Beginplay();
4   Owner = GetOwner();
5
6   ActorThatOpens = GetWorld()->GetFirstPlayerController()->GetPawn();
7   actorRotation = Owner->GetActorRotation();
8 }
```

2.1.2. Función *OpenDoor*

Esta función es la que se encarga de abrir la puerta

```
1 void UAbrirPuerta::OpenDoor().
2 {
3   Owner->SetActorRotation( actorRotation + FRotator(0.f, -90.f, 0.f));
4 }
```

2.1.3. Función *CloseDoor*

Esta función es la que se encarga de cerrar la puerta.

```
1 void UAbrirPuerta::CloseDoor()
2 {
3   Owner->SetActorRotation(actorRotation);
4 }
```

2.1.4. Función *TickComponent*

Esta función se activa en cada tick

```
1 void UAbrirPuerta::TickComponent(float DeltaTime, ELevelTick TickType,
      FActorComponentTickFunction* ThisTickFunction)
2 {
3   Super::TickComponent(DeltaTime, TickType, ThisTickFunction);
4   if (PressurePlate->IsOverlappingActor(ActorThatOpens))
5   {
6     OpenDoor();
7     LastDoorOpenTime = GetWorld()->GetTimeSeconds();
8   }
9
10  if (GetWorld()->GetTimeSeconds() - LastDoorOpenTime > DoorCloseDelay)
11  {
12    CloseDoor();
13  }
14 }
```

2.2. Clase MyBlueprintFunctionLibrary

Esta clase es la usada para crear en C++ nuevos nodos para usar en tus Blueprints

2.2.1. Función SaveGame

Esta función implementa la posibilidad de guardar partida en cualquier momento.

```
1 void UMyBlueprintFunctionLibrary::SaveGame(AActor * MyActor)
2 {
3   UMySaveGame* SaveGameInstance = Cast<UMySaveGame>(UGameplayStatics::
      CreateSaveGameObject(UMySaveGame::StaticClass()));
4   SaveGameInstance->PlayerLocation = MyActor->GetActorLocation();
5   UGameplayStatics::SaveGameToSlot(SaveGameInstance, TEXT("MySlot"), 0)
      ;
6   GEngine->AddOnScreenDebugMessage(-1, 5.f, FColor::Green, TEXT("Juego
      Guardado."));
7 }
```

2.2.2. Función LoadGame

Esta función implementa la posibilidad de cargar en cualquier momento una partida anteriormente guardada.

```
1 void UMyBlueprintFunctionLibrary::LoadGame(AActor * MyActor)
2 {
3   UMySaveGame* SaveGameInstance = Cast<UMySaveGame>(UGameplayStatics::
      CreateSaveGameObject(UMySaveGame::StaticClass()));
4   SaveGameInstance = Cast<UMySaveGame>(UGameplayStatics::
      LoadGameFromSlot("MySlot", 0));
5   MyActor->SetActorLocation(SaveGameInstance->PlayerLocation);
6   GEngine->AddOnScreenDebugMessage(-1, 5.f, FColor::Green, TEXT("Juego
      Cargado."));
7 }
```

2.3. Clase ReporteDePosicion

Esta clase implementa un sistema para conocer la posición actual de cualquier actor al comienzo del nivel, usado para fines de pruebas.

2.3.1. Función BeginPlay

Esta es la función que muestra por consola dónde se encuentra el actor que use esta clase:

```
1 void UReporteDePosicion::BeginPlay()
2 {
3   Super::BeginPlay();
4   FString Nombre = GetOwner()->GetName();
5   FString Posicion = GetOwner()->GetTransform().GetLocation().ToString
      ();
6   UE_LOG(LogTemp, Warning, TEXT("Hola, soy %s y estoy en %s"), *Nombre,
       *Posicion);
7 }
```

aaa