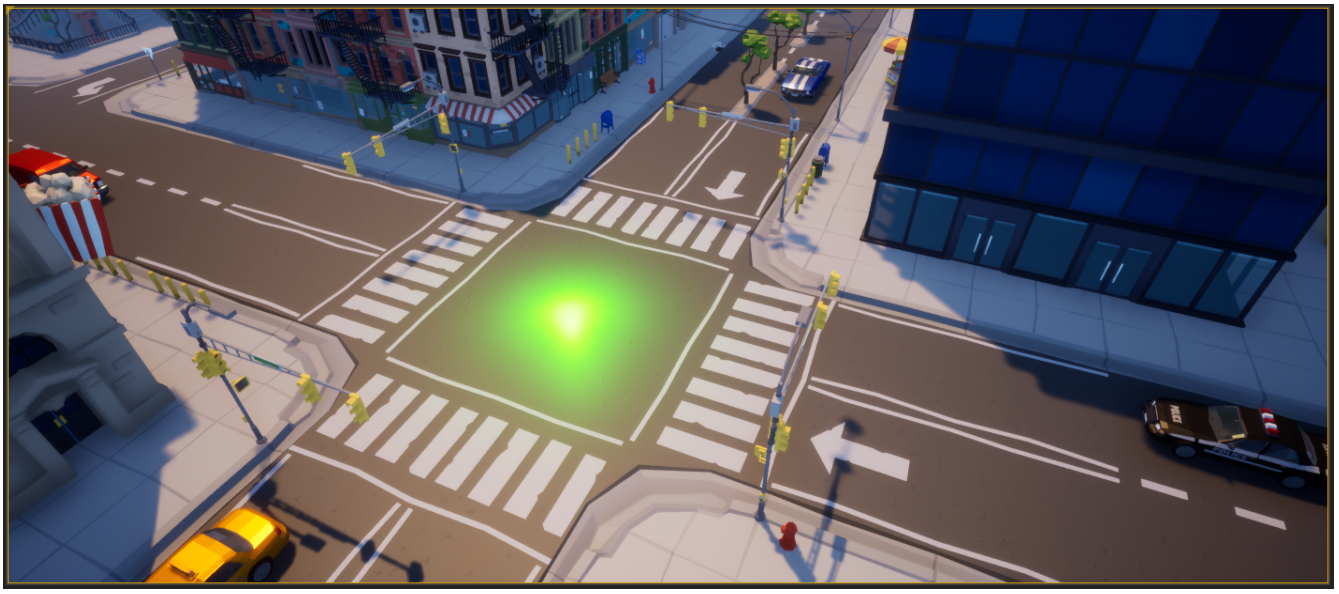
MEMORIA EJERCICIO TRÁFICO



María López Ausín

27/11/20

Diseño y desarrollo de videojuegos

2020 - 2021

ESNE

ÍNDICE

1. Descripción general 3
2. Descripción técnica 3
3. Diario de desarrollo 5
4. Bibliografía 5

DESCRIPCIÓN GENERAL

El objetivo de esta práctica es la realización de un ejercicio de **control de tráfico** en **Unreal Engine 4**. Consiste en un escenario con coches (o desplazadores) que respetarán los semáforos, los cuales cambian en función de si hay coches en el cruce o no.

DESCRIPCIÓN TÉCNICA

En este apartado explicaré los componentes técnicos del proyecto.

Ha sido desarrollado creando varias clase de C++ AActor. Estos son los actores con los que podemos crear instancia de los mismos, modificando las variables que queramos.

Dentro de C++, cabe destacar que contamos con los headers donde se declaran variables y funciones que más tarde se definirán y usarán en los cpp.

En primer lugar tenemos los **desplazadores**. Se trata de un objeto de clase **"Car"**, que a su vez es un actor. Controla el movimiento del mismo y tiene una función que le ordena parar o continuar con su movimiento. En el Tick se llama todo el rato a la función Move, que lo hará si el bool de la propia clase está en true.

```
void ACar::Move(float dT)
{
    if (canMove)
    {
        SetActorLocation(GetActorLocation() +
            GetActorForwardVector() * dT * displacementSpeed);
    }
}
```

Por otro lado, contamos con la **TrafficLight**. Es el elemento central en nuestra escena y el que decide cuándo pasan los desplazadores en el cruce y en qué orden. Con el primer contacto de un coche, la luz se pondrá en rojo y se comunicará con el resto de coches para que paren hasta que el primero haya salido de la zona.

```
ACar* _coche = Cast<ACar>(OtherActor);
cars.Add(_coche);

if (isGreen)
{
    isGreen = false;
    SetLightColor();
}
else
{
    _coche->SetCanMove(false);
}
```

```
cars.RemoveAt(0);
if (cars.Num() == 0)
{
    isGreen = true;
    SetLightColor();
}
else
{
    cars[0]->SetCanMove(true);
}
```

En un puntero a un TArray, guardo por orden de llegada los coches, los cuales serán eliminados cuando salgan de la zona de influencia. Así, el semáforo llamará al primer elemento, el cual sería el siguiente en orden de llegada.

En su constructor se creará el Collider y se hará un SetRootComponent. Para que los eventos de colisión funcionen y los podamos sobreescribir, se tendrá que llamar a una función del collider. Además, se crea la luz, atachándola al collider.

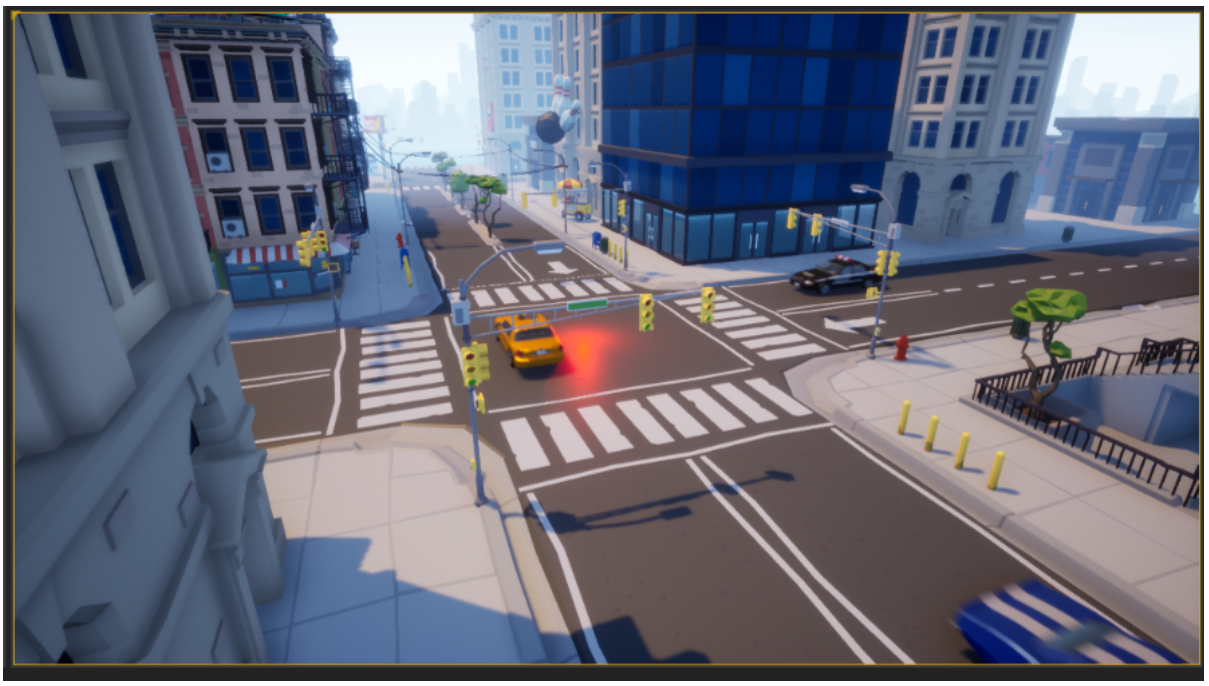
```
this->boxCollider->OnComponentBeginOverlap.AddDynamic(this, &ATrafficLight::OnTriggerEnter);
```

Por último, contamos con el **RotationTrigger**. Es un trigger que se encuentra al final de cada camino que recorren los coches utilizado para darles la vuelta a los mismos y que vuelvan por el mismo recorrido.

```
if (OtherActor != nullptr && OtherComp != nullptr)
{
    ACar* _coche = Cast<ACar>(OtherActor);
    FRotator rotation = FRotator(0, 180, 0);
    _coche->AddActorLocalRotation(FQuat(rotation));
}
```

Con un Quaternion hago una rotación local.

Para finalizar, he montado una pequeña escena por gusto propio con un paquete de assets de Unreal.



DIARIO DE DESARROLLO

En esta práctica, Unreal no ha ayudado nada. Desde crasheos aleatorios hasta que no pudiera crear el proyecto desde cero y haya tenido que crearlo desde otro proyecto.

Me lo he pasado bastante bien montando la escena y creo que es un sistema que puede llegar a crecer para ser usado en otros proyectos. Además, me ha agradado trabajar con C++ ya que es un lenguaje que me gusta y he aprendido más de él. Los blueprints son cómodos pero poder hacer lo que tú quieras con código me supera.

BIBLIOGRAFÍA

https://docs.unrealengine.com/en-US/index.html
https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/ProgrammingWithCPP/
Unreal Engine 4 Game Development in 24 Hours; Aram Cookson, Ryan DowlingSoka, Clinton Crumpler