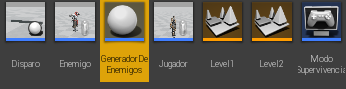
## Generación de enemigos

Vamos a crear un nuevo **Actor** no visible que se encargue de crear nuevos enemigos.

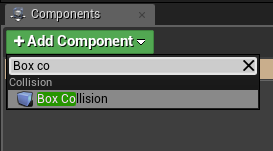
Creamos una nueva clase Blueprint de tipo **Actor**

|  |  |
| --- | --- |
|  |  |

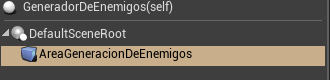
La llamamos **GeneradorDeEnemigos.**

****

Abrimos la clase y agregamos un componente **Box Collision.**

****

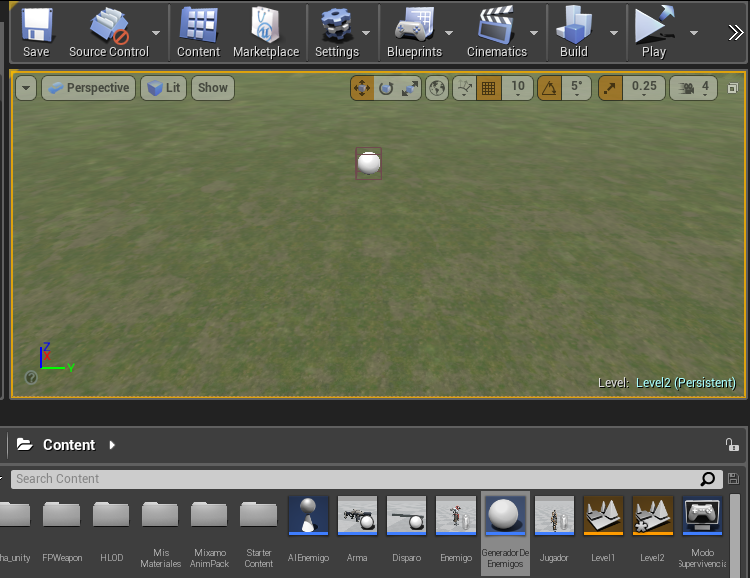
Renombramos el componente como **AreaGeneracionEnemigos.**

****

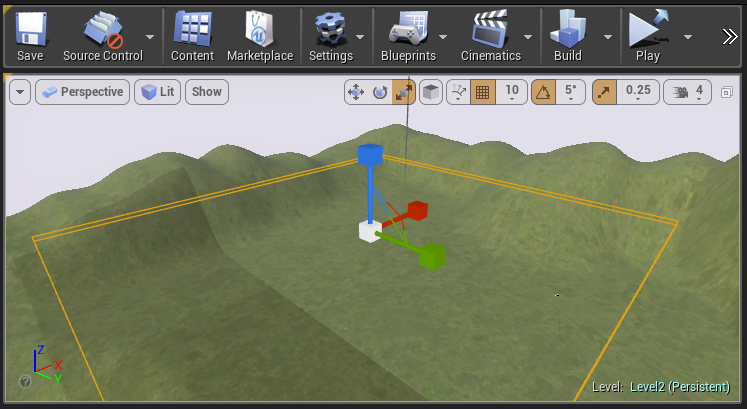
Compilamos y guardamos los cambios.



Arrastramos el **GeneradorDeEnemigos** al Nivel.



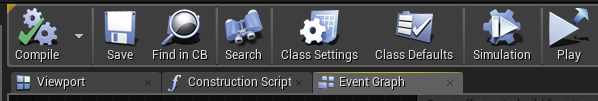
Lo **trasladamos** y **escalamos** para que ocupe una zona relativamente grande y lo subimos en el eje Z para que este en el cielo.



Los enemigos se van a generar en esa área y caer sobre el mapa.

Abrimos de nuevo la clase **GeneradorDeEnemigos**

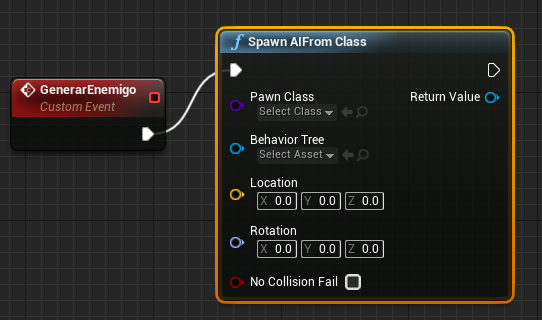
Abrimos al **Event Graph.**



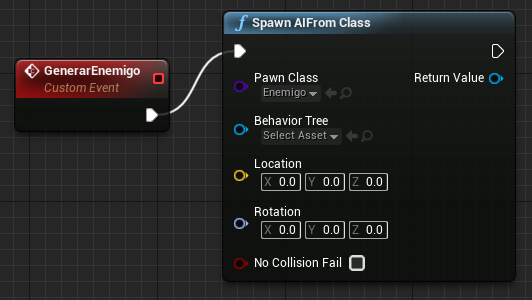
Creamos un nuevo **Add** **Custom Event,** y lo llamamos **GenerarEnemigo.**

****

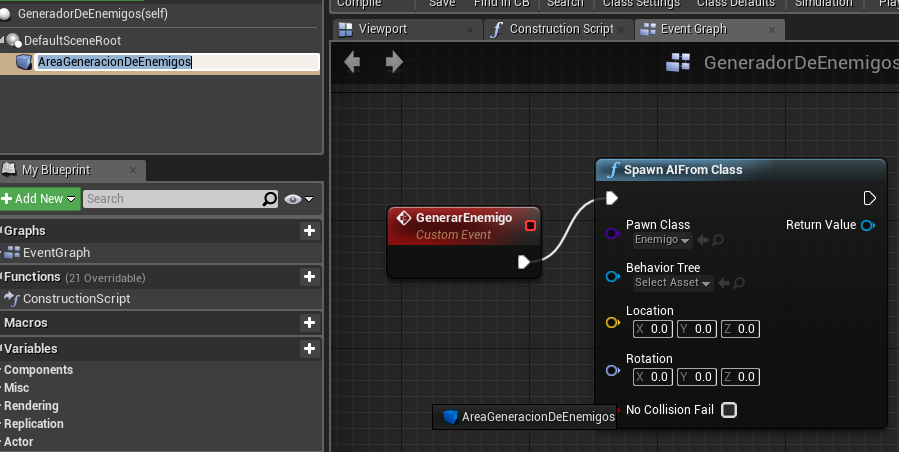
Desde el evento llamamos a un **Spawn AIFrom Class**, para generar un peón (Pawn) controlado por AI en el juego.

****

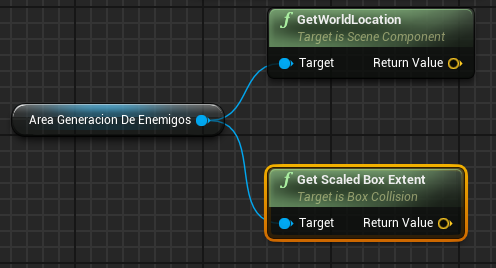
Configuramos la variable **Pawn class** y establecemos que se crea un nuevo **Enemigo.**

****

Arrastramos el componente **AreaGeneracionDeEnemigos** al área Blueprint.

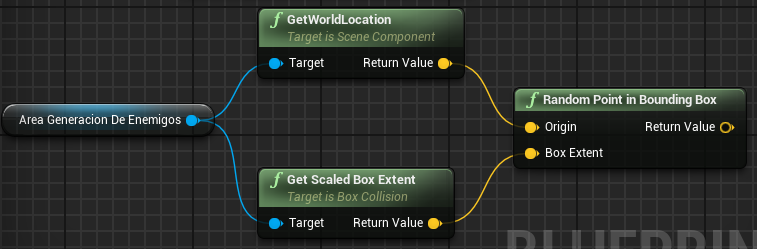


A partir de ahí obtenemos el **GetWorldLocation y Get Scaled Box Extent** (Punto del localización del elemento y sus dimensiones)

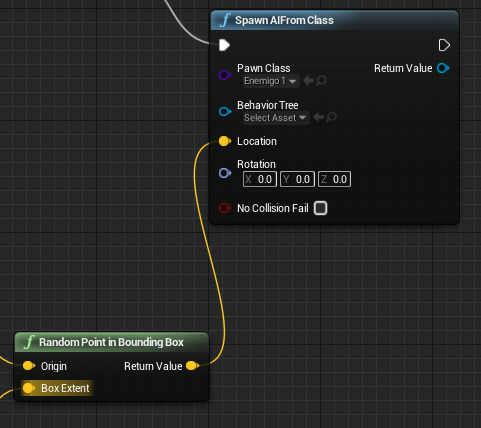
****

Asociamos ambas salidas a una nueva función **Random Point in Bounding Box,** esta función nos da un punto aleatorio.

* Origin - Get World Location
* Box Extent - Get Scaled Box Extent

****

El punto aleatorio retornado se asocia al atributo **Location** del nodo **Spawn AIFrom Class**

****

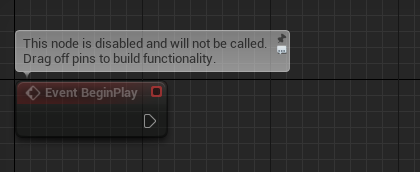
**¿Cuándo se llama al evento Generar Enemigo?**

Vamos a controlarlo desde el **GameMode.**

#### ModoSupervivencia (GameMode)

Abrimos la clase **ModoSupervivencia.**

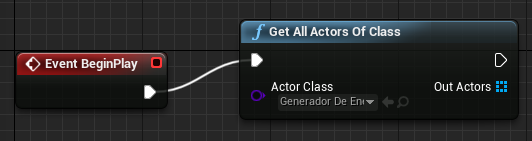
En la vista **eventGraph,** localizamos el nodo **Event BeginPlay** (inicio del juego).



A partir del **Event BeginPlay** invocamos la función **Get All Actors Of Class**

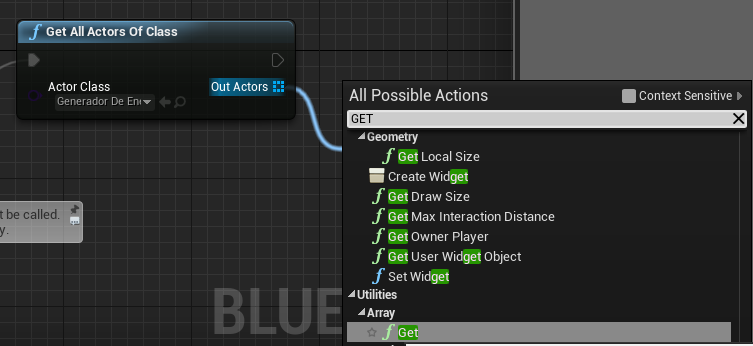
Configuramos el nodo:

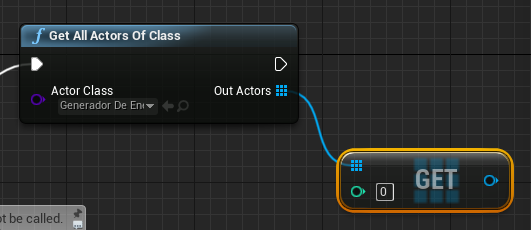
* Actor Class: **GeneradorDeEnemigos**.



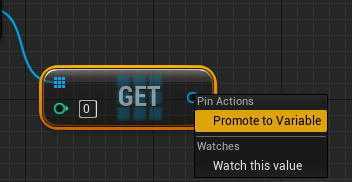
Como retorna una lista de Actores nos tenemos que quedar con el primero (posición 0).

Unimos la salida **Out Actors** a un nodo de tipo **Array - Get.**

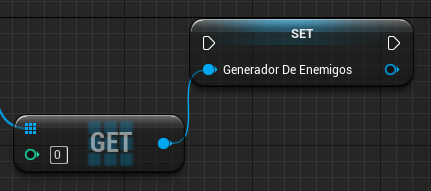




Guardamos el objeto obtenido con el **Get(0)** una variable (haciendo un **promote to variable**), renombramos a la variable como **GeneradorDeEnemigos**.

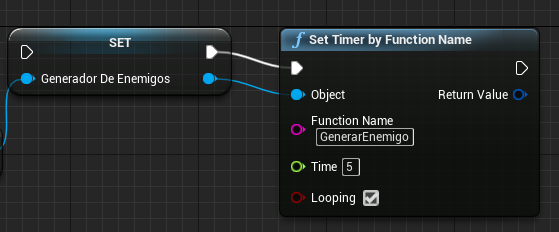


Renombramos a la variable a **GeneradorDeEnemigos**.

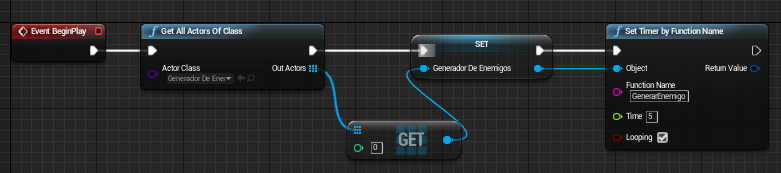


A continuación del **SET** creamos un nodo **Set Timer by Function Name**

* Object: recibe el objeto Generador de enemigos
* Function Name: invoca la función GenerarEnemigo
* Tiempo: 5 (cada 5 segundos)
* Looping : activado.



Nos aseguramos de que el flujo está correctamente conectado (la flecha blanca debe pasar por todos los nodos).



Esta función generaría un enemigo cada 5 segundos, compilamos y guardamos los cambios.



Al probar el juego veremos que cae del cielo un nuevo enemigo cada 5 segundos.



|  |
| --- |
| Get All Actors of class nos permite obtener una lista con todos los elementos de una clase determinada que están actualmente en el juego, si sabemos que solo hay un elemento podemos directamente ir a la posición GET [0], si la lista contiene más elementos podemos recorrerlos todos, ForEachLoop. <https://wiki.unrealengine.com/Blueprint_Essentials_-_12_-_For_Each_Loop>  <https://wiki.unrealengine.com/Blueprint_Essentials_-_9_-_For_Loops> |

## Regeneración de Nivel (Eliminar todos los enemigos al perder)

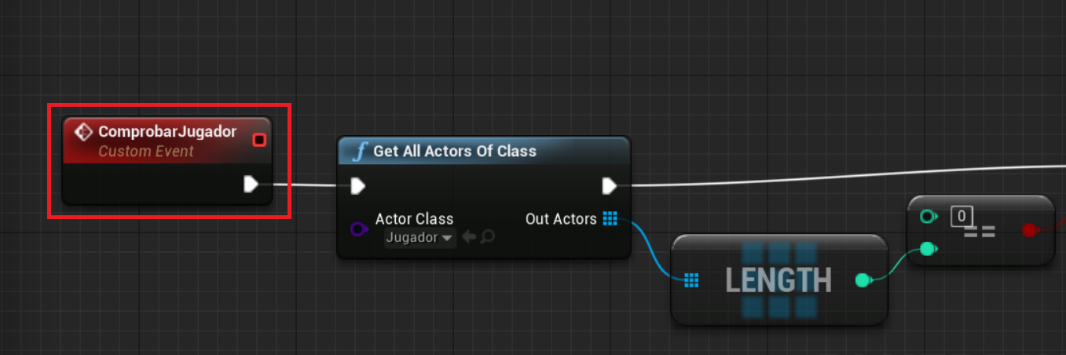
Cuando los enemigos eliminan al jugador y reiniciamos el nivel vamos a eliminar todos los Enemigos.

|  |
| --- |
| **Importante.** Realmente lo que estamos haciendo de “Destruir manualmente” todos los enemigos no sería necesario en este caso, porque al abrir un nuevo nivel, el nivel aparecería con los enemigos originalmente declarados (no con los que se han generado después).  Esto es solo un ejemplo para que veamos como destruir todos los enemigos, puede sernos útil en otros casos, como por ejemplo al coger una bomba que destruye todos los enemigos |

Abrimos la clase **ModoSupervivencia.**

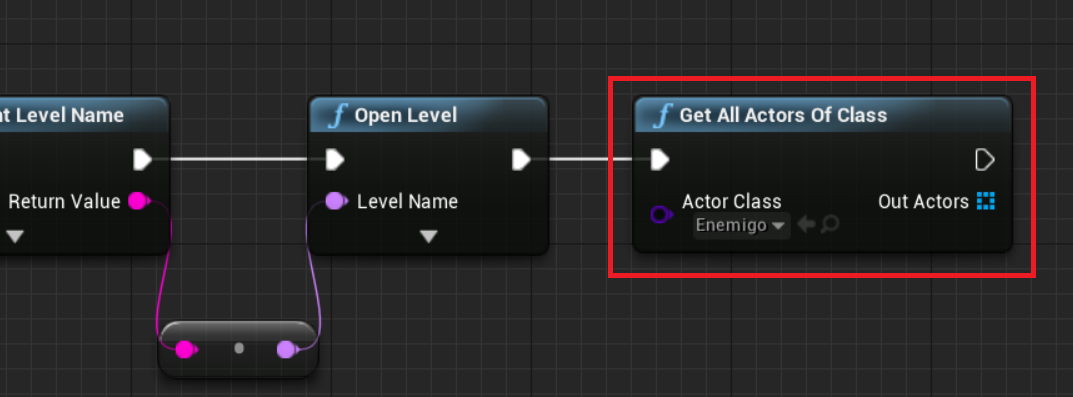
Seleccionamos la vista **Event Graph**.

Localizamos el evento **ComprobarJugador** , nos dirigimos a la última caja que se ejecuta en el evento y continuamos implementando el flujo.

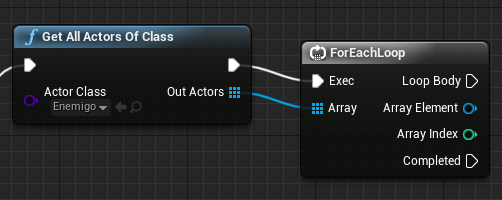


Desde el evento creamos una función **Get All Actors of Class**

* Indicamos que la clase es **Enemigo**

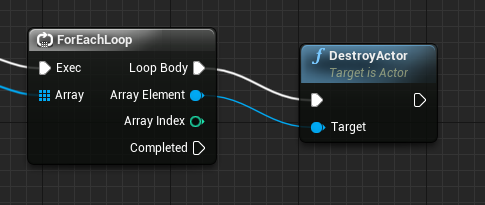


Creamos un **ForEachLoop** para recorrer la estructura **Out Actors.**



Para cada elemento del array (**Array Element**) llamamos a **DestroyActor**.

Unimos el flujo del **Loop Body** con el **Destroy Actor.**

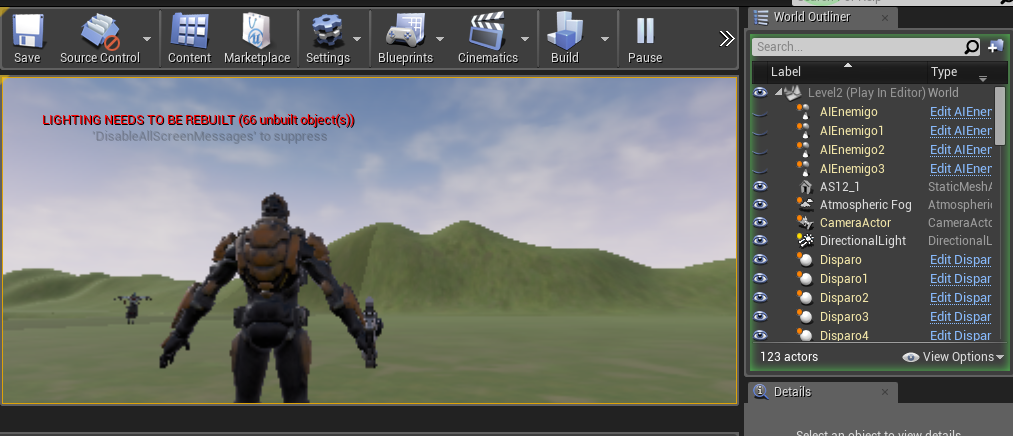


Nos aseguramos de que el flujo (línea blanca) es correcto para todos los nodos.

Guardamos los cambios y compilamos



Ejecutamos el juego y validamos el funcionamiento, cuando matan al Jugador todos los enemigos deberían desaparecer.

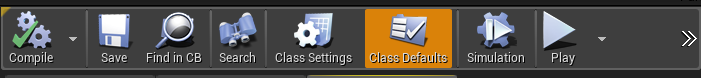


Detectamos otro error, los disparos no se están destruyendo nunca, cada vez que disparamos tenemos un nuevo actor. Cuanto más disparamos más Actores hay en el mapa.

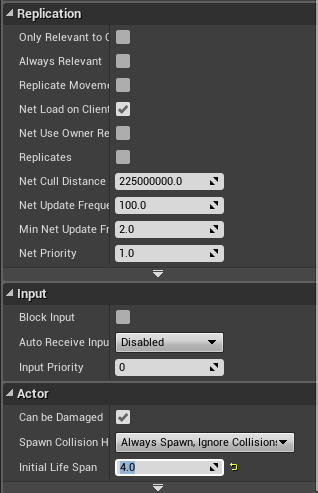
Accedemos a la clase **Disparo**



Entramos en **Class Defaults** o en el componente Disparo(self)

****

Modificamos la propiedad **Initial Life Span = 2 (segundos),** permite especificar cuánto tiempo va a seguir vivo el objeto.

****

Comprobamos que el número de actores creados en el juego no aumenta de forma constante como antes.

