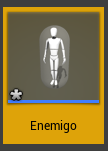
# Enemigos

Creamos una clase que herede de **PersonajeBase** (Así tendrá la variable vida y el evento **ReducirVida**)

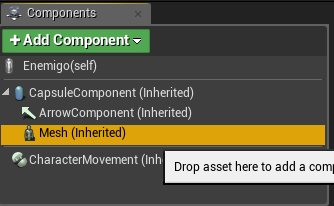


Llamamos a la clase **Enemigo**

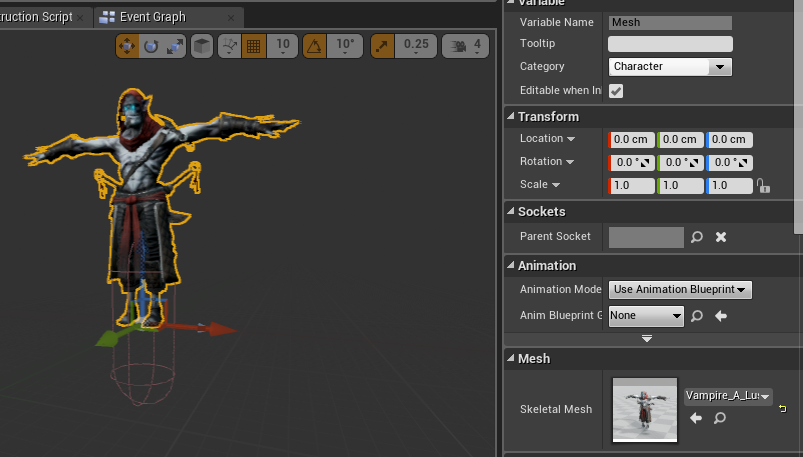


Abrimos la clase.

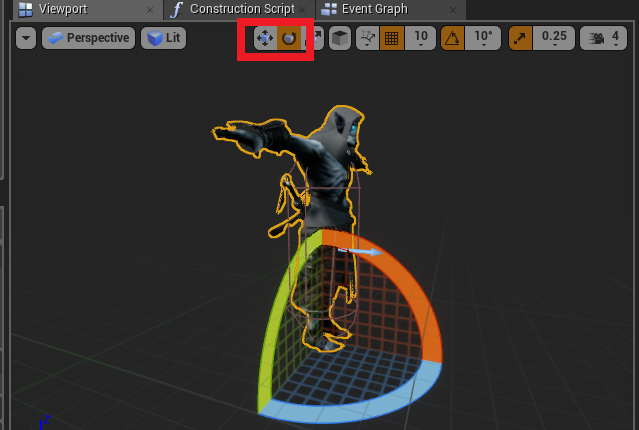
Dentro de sus propiedades seleccionamos el componente **Mesh** (malla con esqueleto).



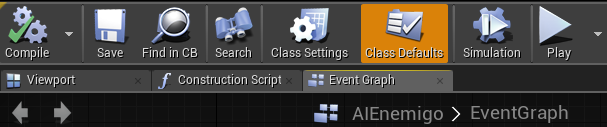
En las propiedades de la **Mesh** le asignamos el **Skeletal Mesh** **= Vampire\_A\_Lusht**



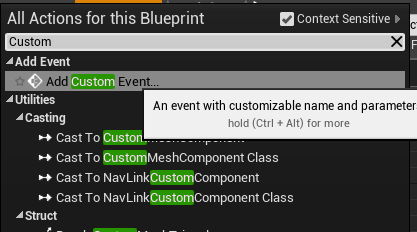
Bajamos el personaje 90 centímetros y lo rotamos 90 grados hacia la derecha (Debe estar mirando hacia la X).



Seleccionamos la pestaña **Event Graph.**



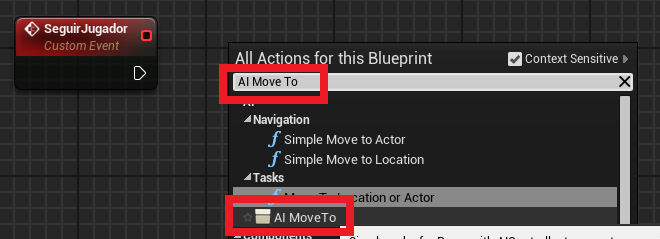
Creamos un nuevo nodo de tipo **Add Custom event**, lo llamaremos **SeguirJugador**.



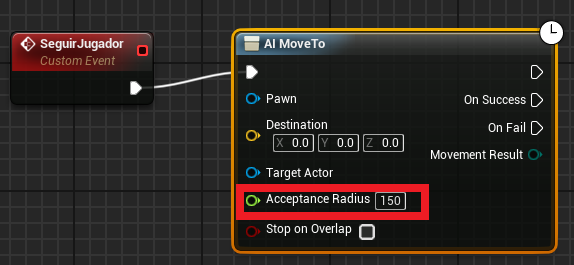


\*El nombre se puede cambiar haciendo click en el propio nodo, o en las propiedades que aparecen en la parte derecha de la pantalla.

Creamos un nuevo nodo **AI Move To** y lo conectamos al flujo de nuestro evento.

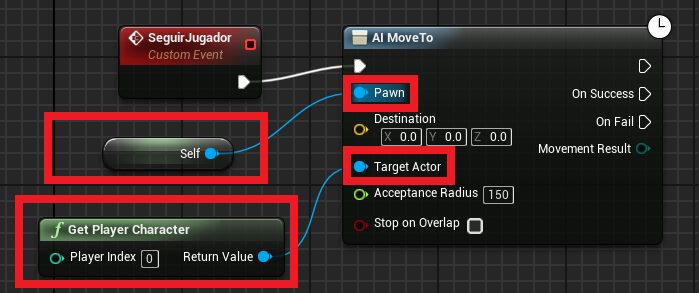


La propiedad **Acceptance Radius** es el radio que los enemigos van a dejar alrededor del jugador, por lo tanto, cuando más pequeño sea el radio más se acercaran los enemigos.

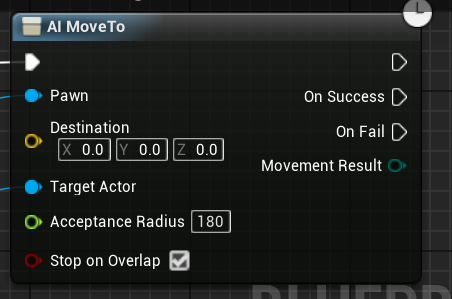
****

Ahora falta especificar dos cosas.

* **Pawn :** quien es el elemento que se va a mover, el propio enemigo **Self**
* **Target Actor**, a quien hay que seguir, en cualquier momento podemos obtener al jugador principal del juego con **Get Player Character**.



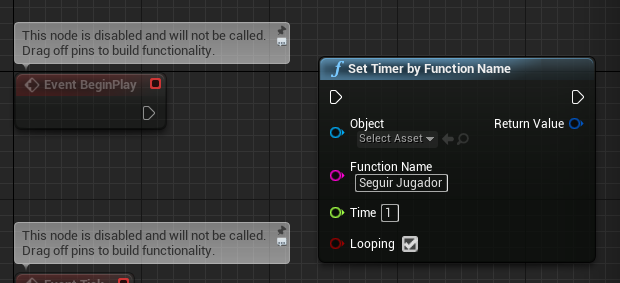
|  |
| --- |
| Sí en lugar de **Acceptance Radius** 150, ponemos 180 el enemigo se acercará un poco menos al jugador y quedará mejor cuando el enemigo haga la animación de atacar. Podemos modificar este parámetro más adelante en función de las animaciones que le queramos colocar al **Enemigo**. |



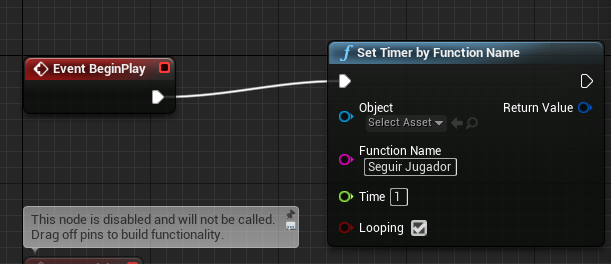
Falta determinar cuándo se llamará al evento **Seguir Jugador**.

Buscamos en el Event Graph el evento **Event BeginPlay** , creamos al lado un nodo **Set Timer By Function Name**. Este nodo llama a una función repetidamente. Configuramos:

* **FunctionName** : Nombre de la función a ejecutar
* **Time**: Intervalo de tiempo entre ejecuciones
* **Looping**: Ejecutar en forma de loop (Boolean).

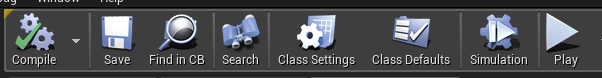


Asociamos el flujo de **BeginPlay** al nuevo Nodo.

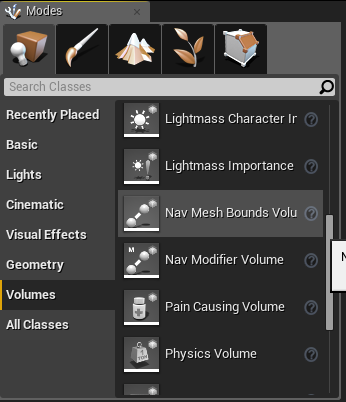


|  |
| --- |
| **Hilos**: Cada vez que usamos un **Set Timer By Function Name** o **Set Timer By Event** estamos creando una especie de hilo, esta función se usa mucho. |

Compilamos, guardamos los cambios.

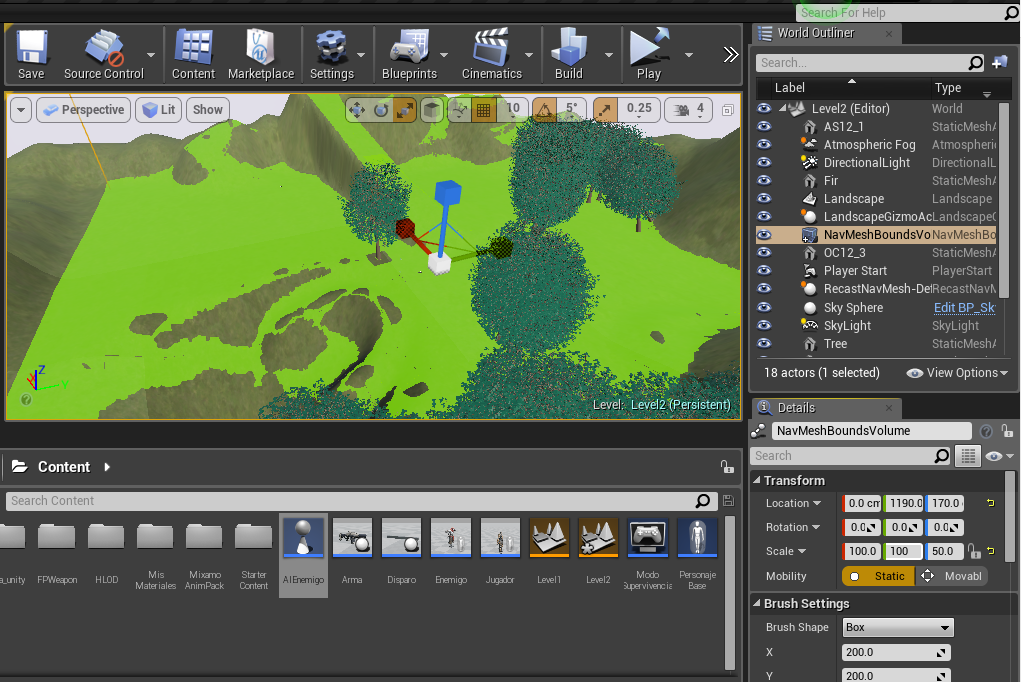


Volvemos al Mapa, nos hace falta incluir un componente de tipo **Volumes -> Nav Mesh Bounds Volume** que indica por donde se pueden mover los enemigos.

p

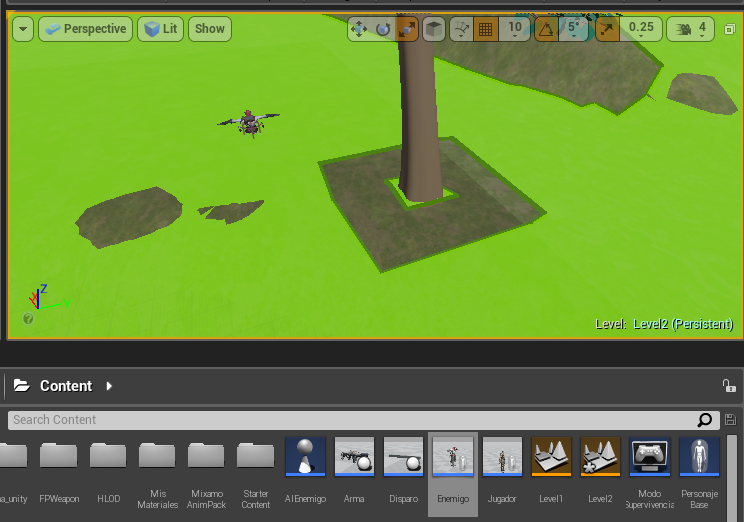
Podemos pulsar la tecla **"p"** para destacar la Nav Mesh, se dibujará en verde todas las zonas por las que el enemigo puede caminar.

Redimensionamos el cuadrado en sus 3 ejes para que ocupe todo el mapa (o casi todo).



Nos aseguramos de que el cuadrado cubra todo el suelo, es importante que el cuadrado colisione con el suelo (no puede estar encima).

Arrastramos un Enemigo al mapa (dentro del **Nav Mesh**).

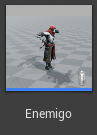


Ejecutamos el juego para comprobar que el funcionamiento es correcto.

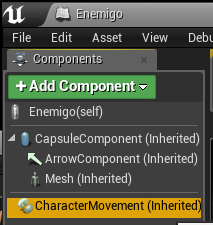


Realmente no hemos configurado el **componente que mueve al enemigo**, se está ejecutando con los valores por defecto.

Sí queremos modificarlo abrimos de nuevo la clase **Enemigo**.



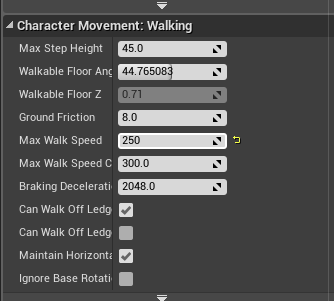
Seleccionamos el componente **CharacterMovement**.



En las propiedades de ese componente podemos cambiar su modo de funcionamiento.

Modificamos la propiedad **Max Walk Speed = 250**, para que camine más despacio.

La propiedad **Walkable Floor Angle** : permite especificar cuanta pendiente son capaces son capaces de subir.



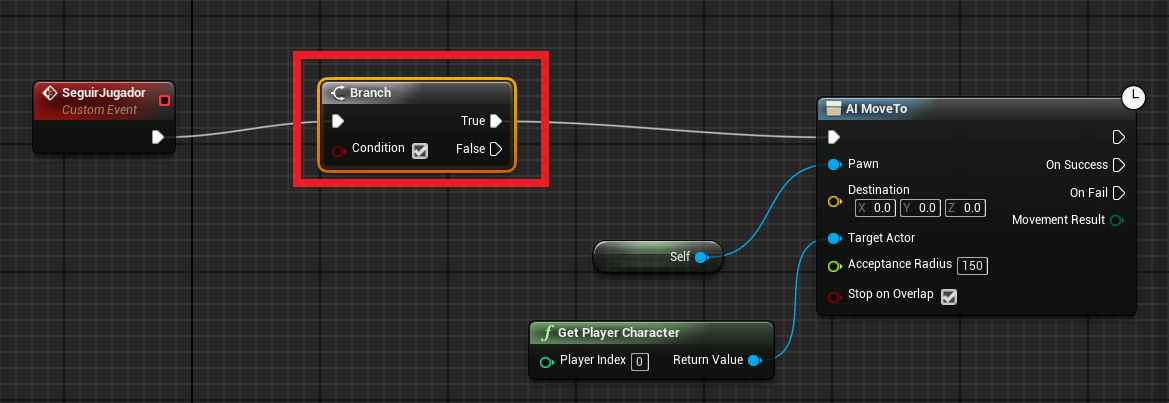
Volvemos a probar el juego, podemos observar que en ciertos momentos cuando el **Enemigo** se coloca entre la cámara y el **Jugador** se produce un Zoom (Es por el Spring Arm)

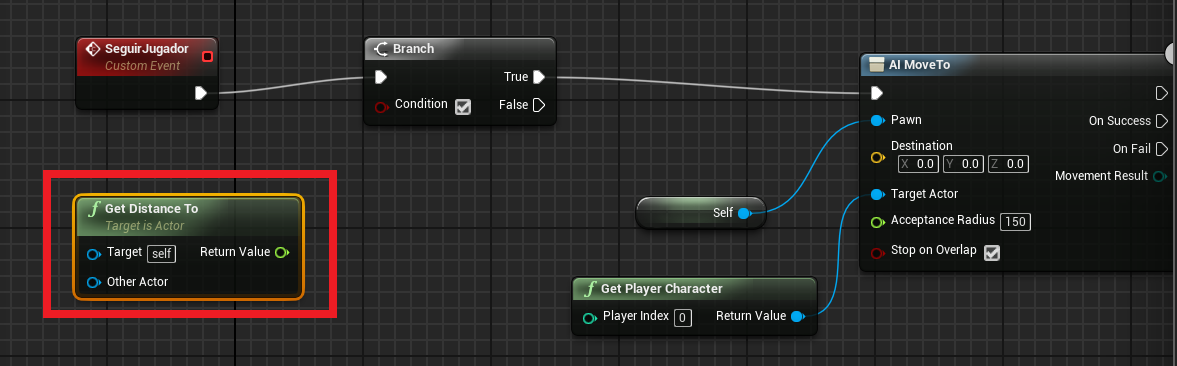
|  |  |
| --- | --- |
|  |  |

# Distancia entre Actores – Seguir cuando el Jugador está cerca

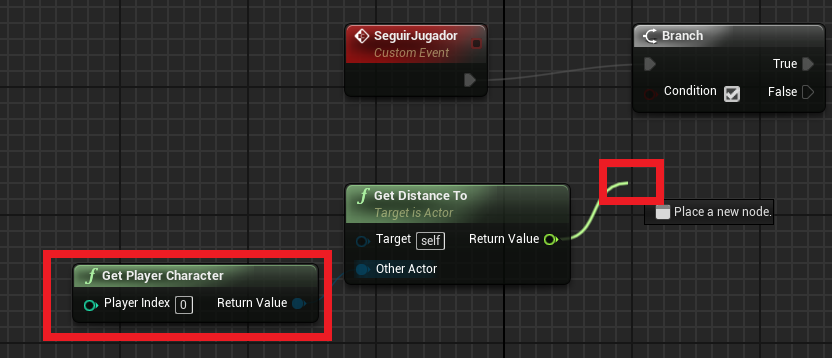
En la función **SeguirJugador**, incluimos un **Branch** antes de llamar a **AI MoveTo**

Un **Branch** es un **if,** recibe una **Condition** (expresión lógica Booleana) y se bifurca en 2 ramas **True** y **False**

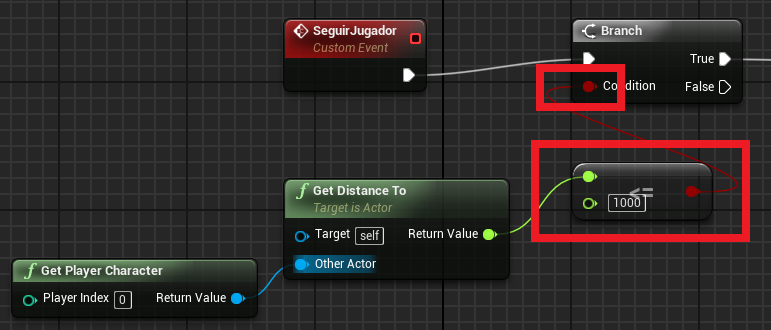


Agregamos la operación **Get Distance To,** recibe dos actores y obtenemos la distancia  
  


Los actores serán **Self** (el propio Enemigo) y **Get Player Character** (jugador controlable).



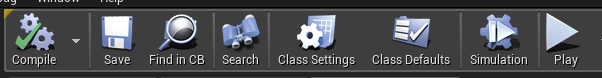
Sí la distancia es <= 1000 entonces seguimos al Jugador.



Otra cosa que mejoraría el juego es comportamiento, es que el enemigo no siguiera al Jugador sino la localización donde lo vio por ultima vez.



Compilamos, guardamos los cambios.



# Modificar la actualización del seguimiento.

Entramos en la lógica del **Enemigo** y cambiamos la frecuencia de llamada de la función **SeguirJugador**, veremos que nos empieza a seguir cada 5 segundos.

# 

Compilamos, guardamos los cambios.

