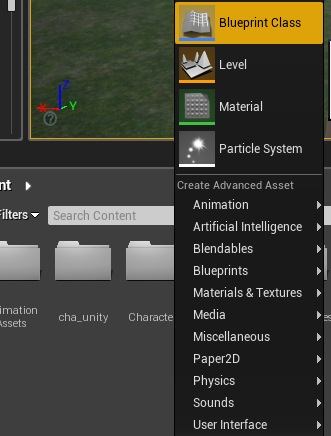
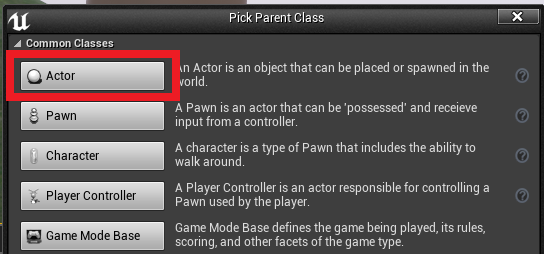
# Recolectable

Vamos a crear un elemento “recolectable” (moneda, munición o similar), para ello nos situamos en content y creamos una nueva clase que herede de **Actor**





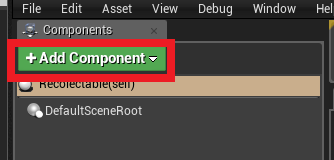
Llamamos a la clase **Recolectable**



Abrimos la clase.

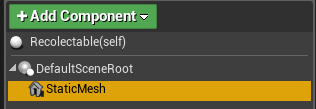
Al partir del elemento básico **Actor**, nos encontramos con que no tiene componentes agregados.

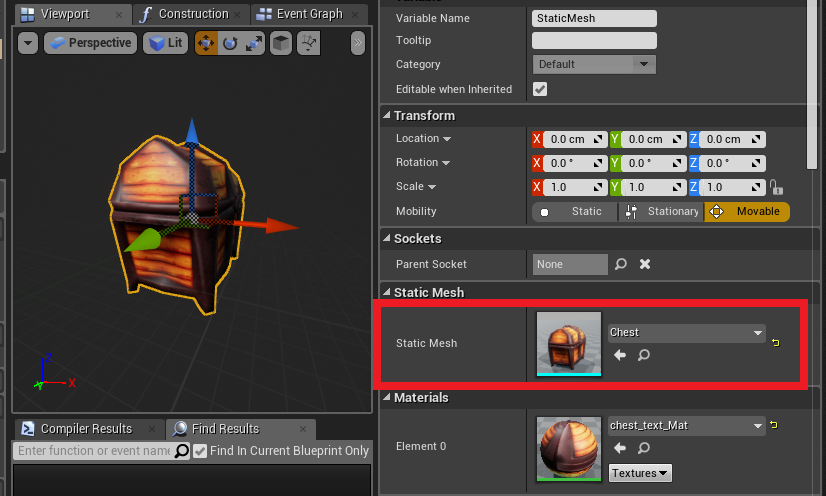
Agregamos un componente de tipo Static Mesh, (malla 3D sin esqueleto interno)





Seleccionamos el nuevo componente **Static Mesh,** en las propiedades que aparecen en la parte derecha asignamos una **Static Mesh = Chest**.

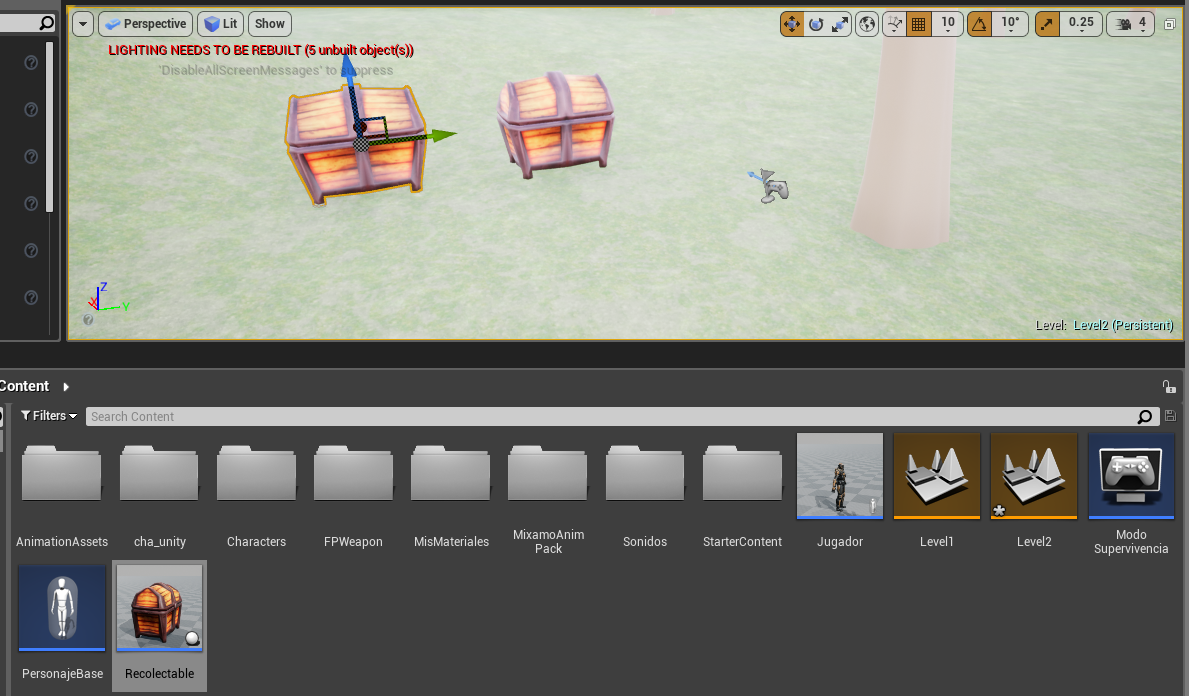




Compilamos y guardamos



Abrimos el nivel y arrastramos dos recolectables cerca de la posición de inicio del jugador (Elemento Player Start)



Probablemente haya que subirlos un poco en el eje de las Z.

Probamos a ejecutar el juego



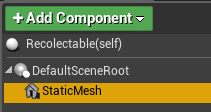


# Tipos de colisión

Por defecto los **recolectables** están bloqueando al **jugador**. Para cambiar el tipo de colisión abrimos el **recolectable**.



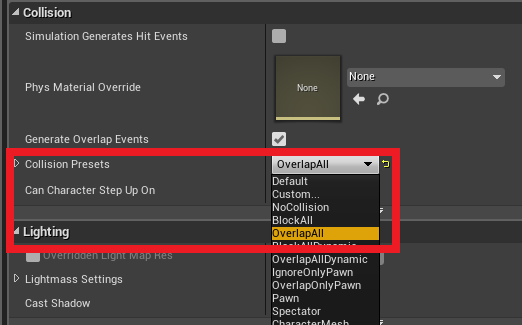
Seleccionamos el componente **StaticMesh**



Modificamos su propiedad, **Collision Presents**. Realmente esta propiedad puede tomar muchos valores, algunos de los más comunes son:

* **Block all :** bloquea físicamente el paso
* **Block all Dynamic :** bloquea físicamente el paso solo de elementos dinámicos
* **OverlapAll:** No bloquea físicamente el paso, pero registra el evento Overlap (Superposición) si algo lo “atraviesa”
* **NoCollision :** No bloquea físicamente el paso y tampoco registra el evento Overlap (Superposición) si algo lo “atraviesa”

Probamos por ejemplo el modo **OverlapAll**.



Compilamos y guardamos





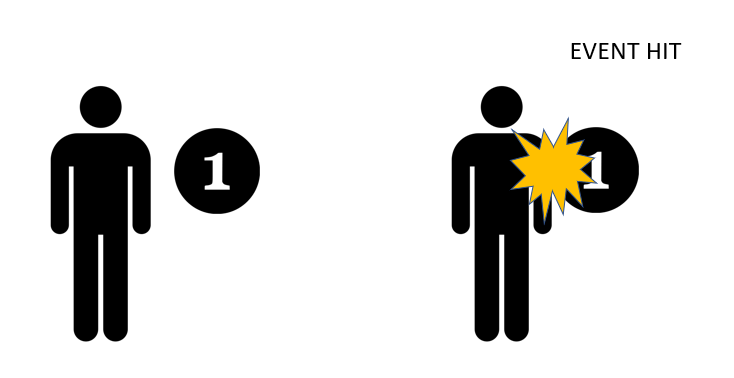
Volvemos a poner el modo **Block all.**



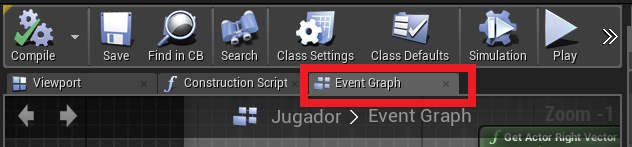
## Hit (golpes)

Una forma de registrar “colisiones” entre el **Jugador** **controlable** y otros elementos es registrando eventos **Hits**. (Solo se suele usar para colisiones relativas al jugador)

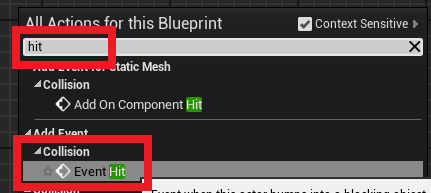
Para que se produzca un Hit tiene que haber un “golpe físico”.



Abrimos el **Recolectable** y accedemos a la pestaña de **Event Graph.**

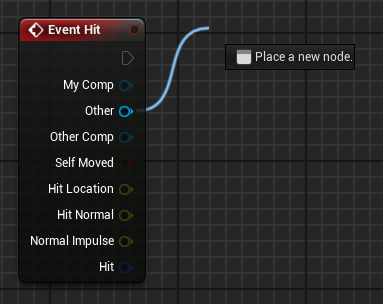


Buscamos una superficie libre, botón derecho y buscamos el evento **Event Hit**

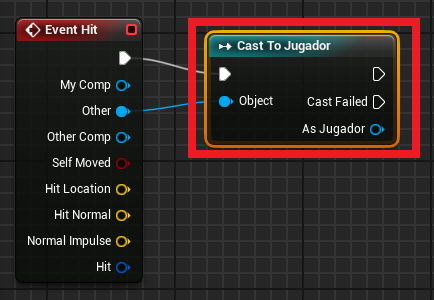


Este evento se llama automáticamente cada vez que se registra un golpe, recibimos muchísima información acerca del golpe (Actores y componentes involucrados, posición exacta del golpe, fuerza , etc).

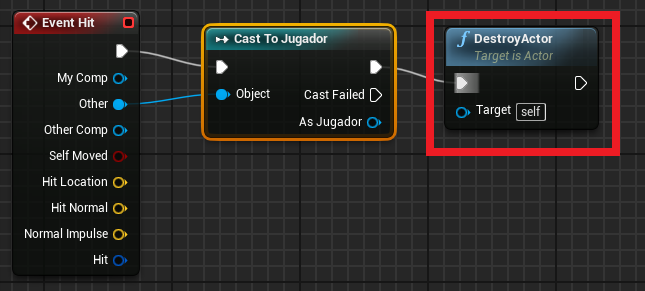
Solo vamos a la variable **Other** (El otro Actor involucrado en el golpe además del recolectable)



Para comprobar si el actor **Other** es un Jugador, hay que hacer un **Cast To Jugador.** (Esta es la forma de comprobar de que tipo es una variable)



El Cast, genera dos flujos (el de éxito y el de fallo Cast Failed), por el flujo de éxito llamamos al **DestroyActor (recibiendo self>recolectable) como parámetro.**



|  |
| --- |
| ¿Diferencias entre lo anterior y esto? |

Compilamos y guardamos

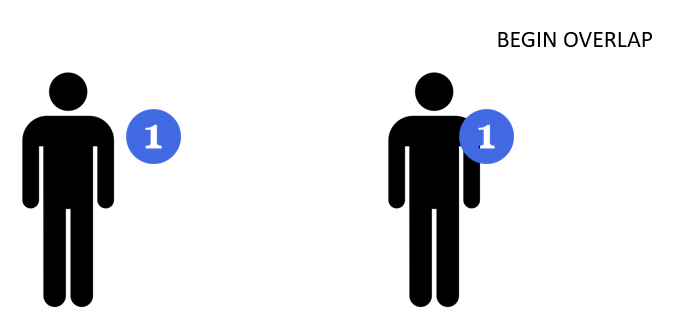


Sí chocamos con un **recolectable** este se destruye.



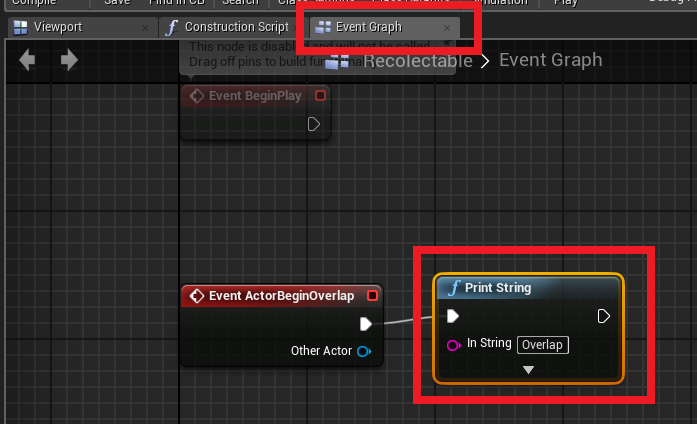
## Overlap (Superposiciones) - Recomendado

Overlap significa literalmente superposición ambos elementos tienen que estar en el mismo espacio físico.

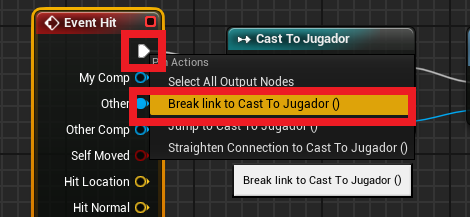


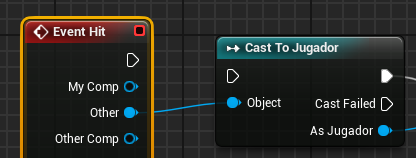
Abrimos el **Event Graph** y localizamos el evento **ActorBeginOverlap** (ya está pre-incluido en el Event Graph por ser uno de los más usados)

Ejecutamos la función **Print String,** al detectar un Overlap



Para que el elemento no desaparezca rompemos el flujo de ejecución de **Event Hit** (botón derecho sobre la flecha blanca -> **Break link to Cast To Jugador()** )





Compilamos y guardamos



Sí probamos el juego vemos que no hay ningún mensaje de Overlap en la pantalla (parte superior izquierda).



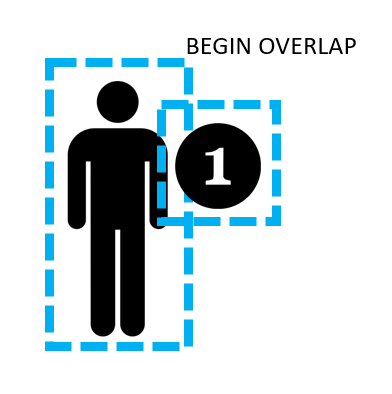
Sí la malla de un elemento bloqueante físicamente no puede haber superposición.



Para que hubiera overlap/superposición, deberíamos optar por una de estas opciones:

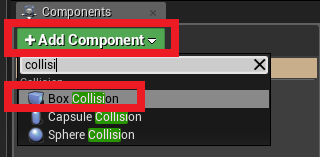
1. Que la malla del objeto fuera overlapeable (CollisionType). Esto haría que el objeto fuera “etéreo” en todos los casos
2. Dejar la malla como esta y agregar área de overlap que sobresaliera de la malla 3D. Esto haría que siguiente manteniendo sus propiedades bloqueantes y pudiera por ejemplo bloquear a los enemigos.

Vamos a probar la segunda.

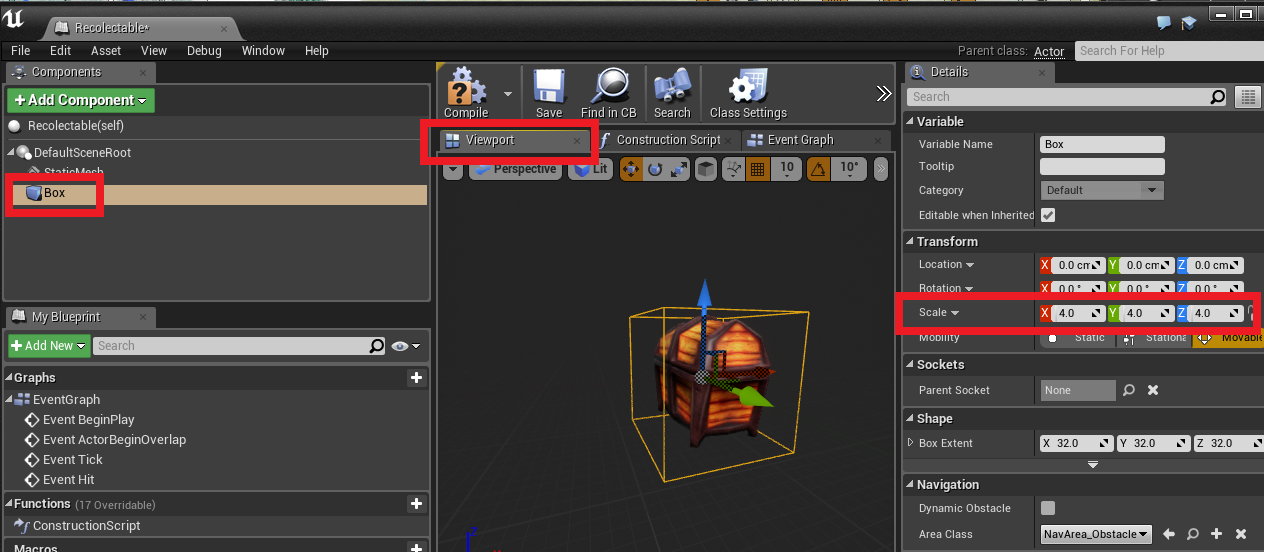


Abrimos el **Recolectable**.

Agregamos un nuevo componente de tipo **Box Collision** (todos los elementos Collision son iguales, solo cambia la forma).



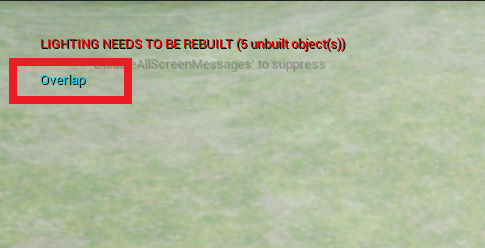
Seleccionamos la **Box** de colisión y le cambiamos el factor de escalado para que sea algo más grande que la malla.



Compilamos y guardamos



Al entrar en superposición con el área de colisión se muestra el **PrintString** de Overlap.



Aunque **ActorBeginOverlap** es el más utilizado hay 2 eventos de invocación automatica:

* **ActorBeginOverlap :** al **inicio** de una superposición
* **ActorEndOverlap :** cuando se **separan** de una superposición

También una función:

* **isOverlapingActor :** la cual podemos invocar en cualquier parte del código, nos permite saber si en ese momento concreto está habiendo una superposición.

