

Interaktive Räume : HDRI, Platforms and Respawn



Adding an HDRI backdrop and Lighting

Add an „Editor Sphere“ from Unreal Content (activate the new content under Settings)

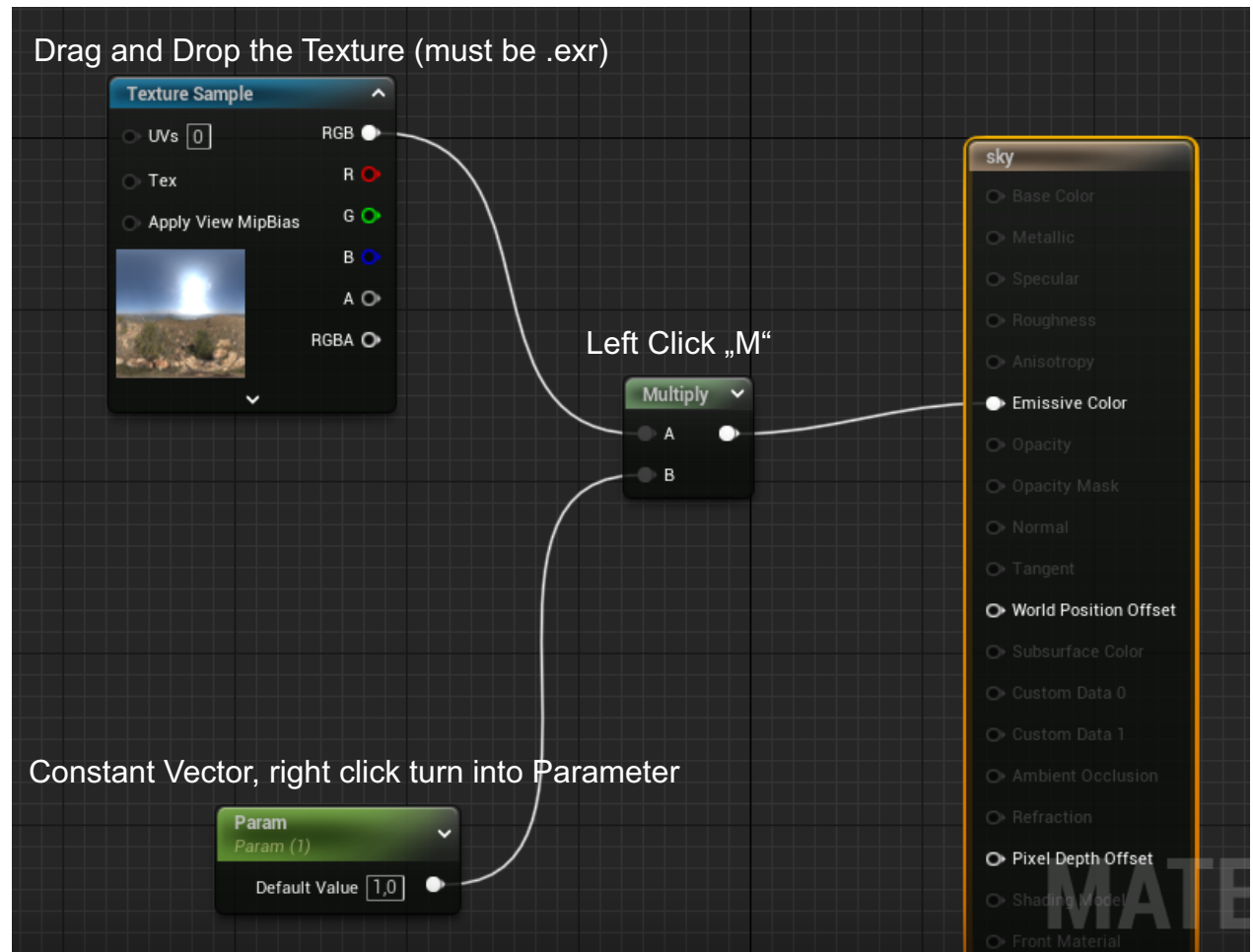
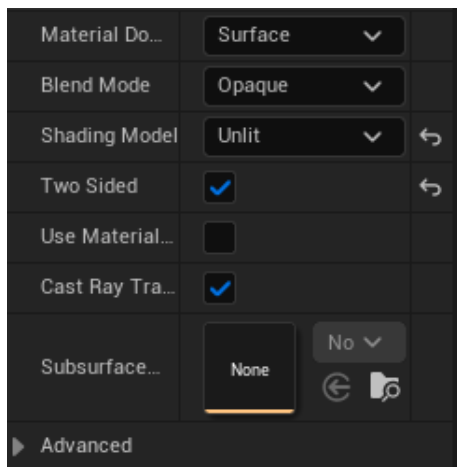
Then create a new Material, set it to unlit and double sided and connect it through a Multiply node with the Emissive Color.

Also create a Constant vector for the B Socket and turn it into a Parameter.

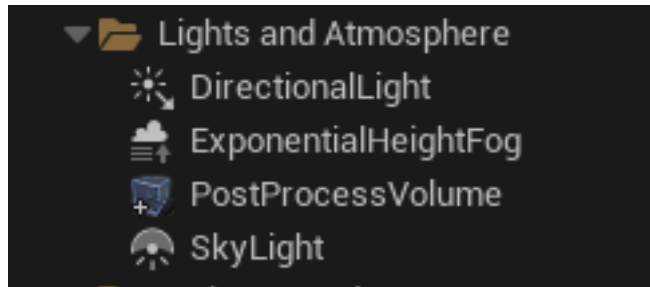
Drag and drop **.EXR HDRI** Map from <https://polyhaven.com/hdris> or Ai generated from www.blockadelabs.com, download or convert into **.exr** (use convert.io for conversions)

Save the newly created Material and drag it onto the Editor Sphere.

Now You need to add a skylight, apply its size to the Editor Sphere and click the recapture Button for the skylight so it uses the Editor Spheres colors for Lighting.



Adding an HDRI backdrop and Lighting



Other Important components we need for a perfectly lit Scene:

Directional Light: Functions as the Sun, try to match where the Sun is in your HDRI by Pressing and Holding „CTRL + L“ and simultaneously moving the Mouse to control the Location of the „Sun“

ExponentialHeightFog:

Adds a Fog that can simulate our Atmosphere and adds to the visibility of Bloom and Rays of God, be careful, can easily be overused.

PostProcessVolume:

Has an influence on everything, can be only for a certain area or the whole scene, can be used to modify your Bloom or Lensflare etc. (if for the whole scene, make sure Infinite Extent (Unbound) is selected.)

it makes sense to put all your Lighting Components in the same Folder.

Adding an HDRI backdrop and Lighting

Rays of God

In the „Directional Light“ search for „Light shaft“ and select the relevant checkboxes. for a stronger effect you can also enable Bloom (can be every extreme)

For more Atmosphere we can also add an „Environmental Fog“ actor and a „postprocessing Volume“. In the Postprocessing volume you can for example add „Lens flare“ and „Depth of Feld or respectivley have better control over the average Exposure behaviour.

If you can live without „God Rays“ the HDRI Background“ Plugin is the easiest way to get a functioning HDRI in your scene. Just remember to download your images in the HDR format or convert them to it (www.convert.io)

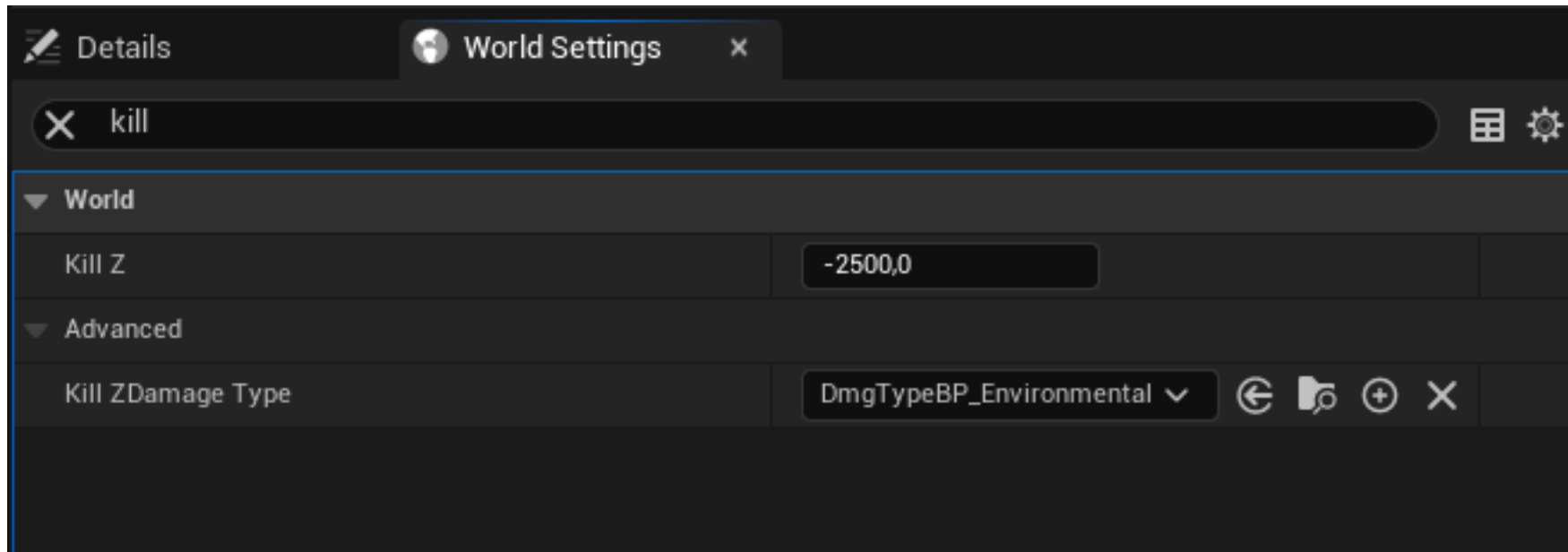


Respawns and Checkpoints

We want to create Checkpoints in our Level so we don't always have to restart from the Menu after falling or missing a jump.

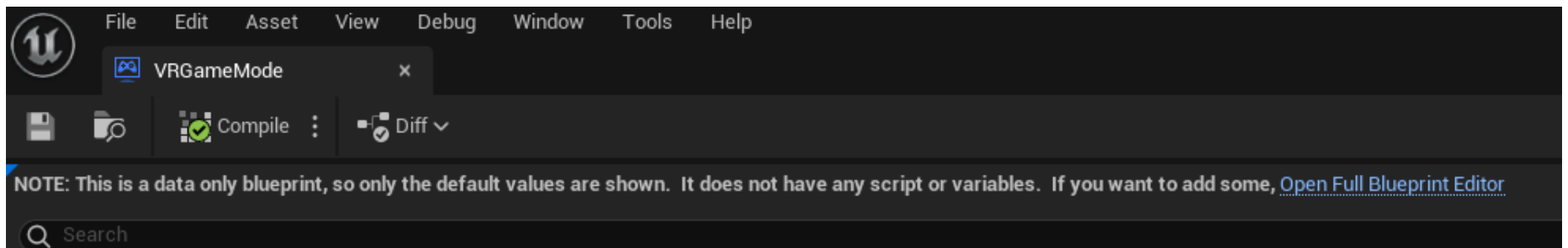
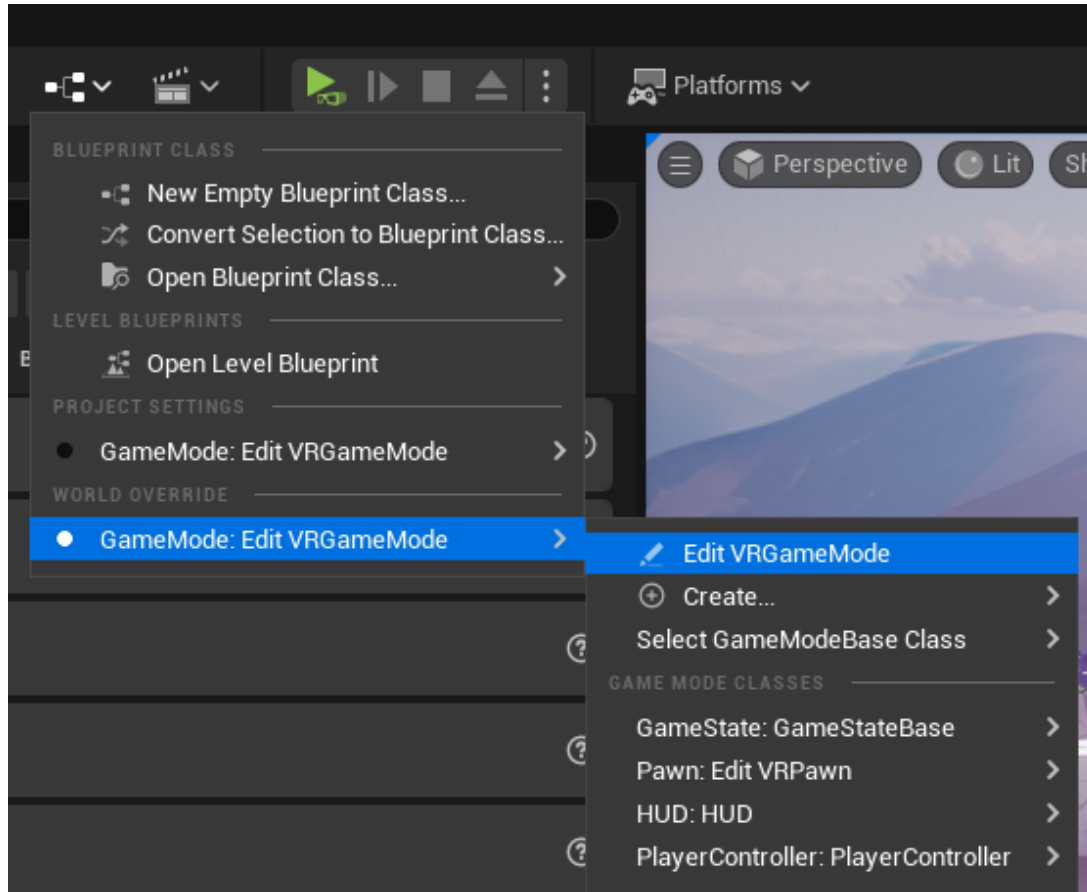
So at first we need to establish a Kill Zone where the Character (and most other dynamic Objects) „die“ on passing through.

This can be done in the **World Settings**. Adjust the value to your scene.



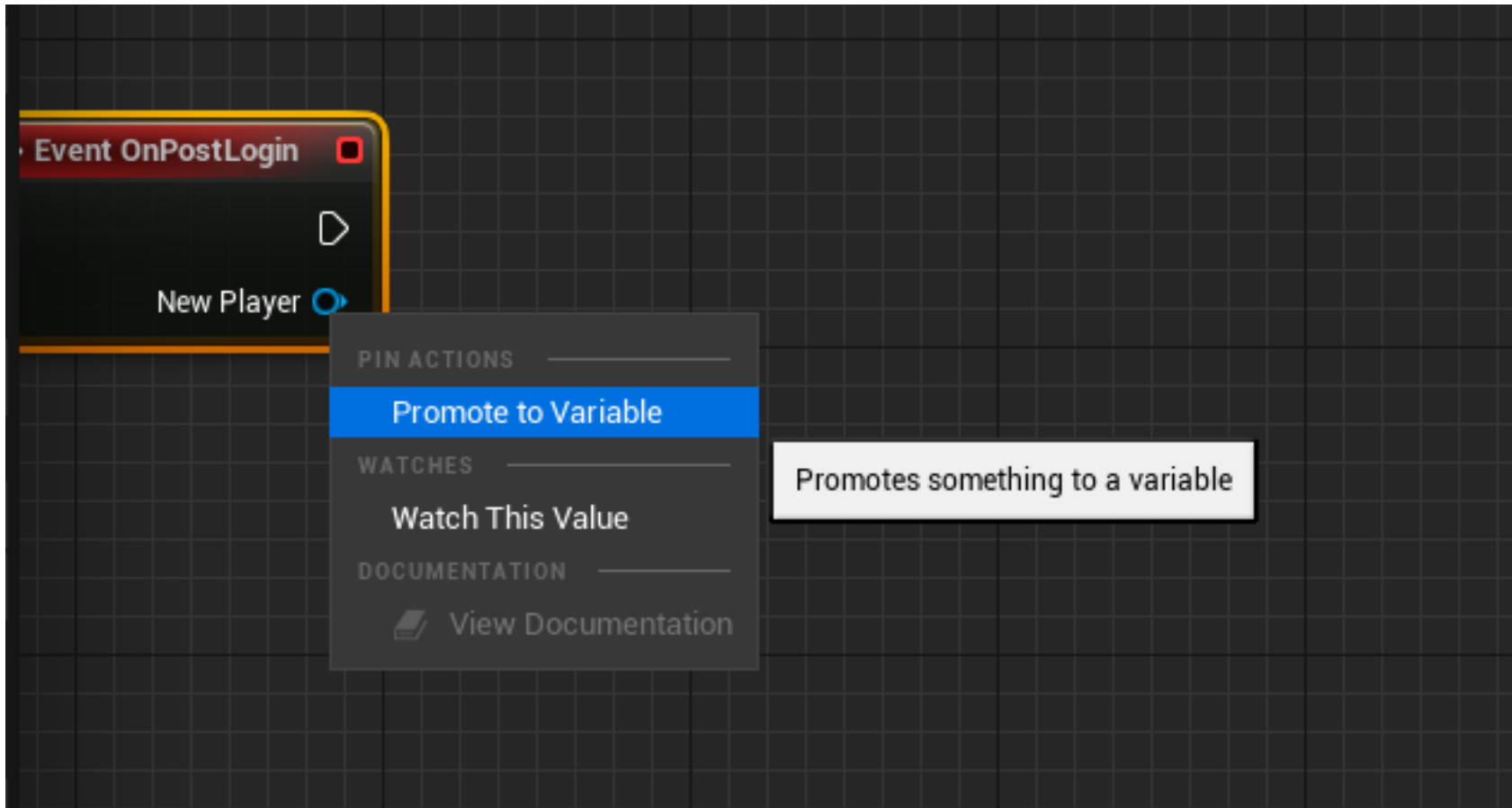
Respawns and Checkpoints

Click the Blueprints icon and Select „Edit VRGameMode“. Doing this the first time you won't see much, click on „Open Full BlueprintEditor“ to continue.



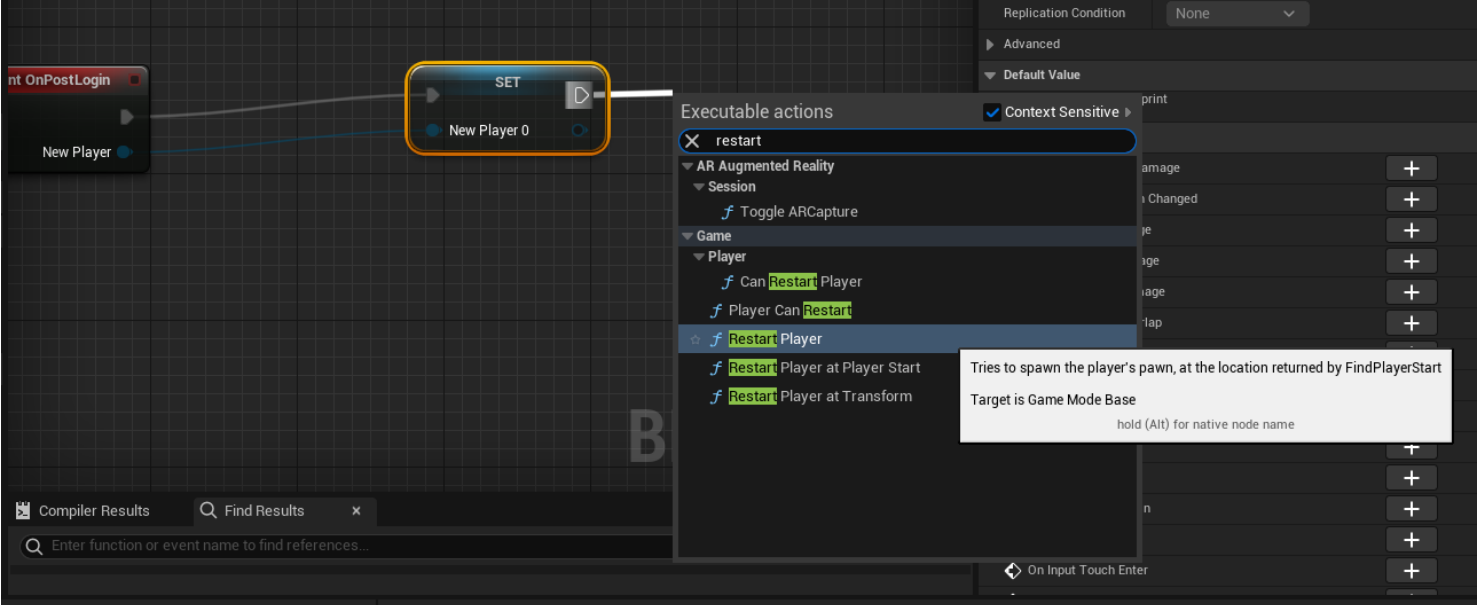
Respawns and Checkpoints

Add a „OnPostLogin“ Event, right click „New Player“ and select „Promote to variable“. This gives you a „Set“ Node (depicted on the next Page)



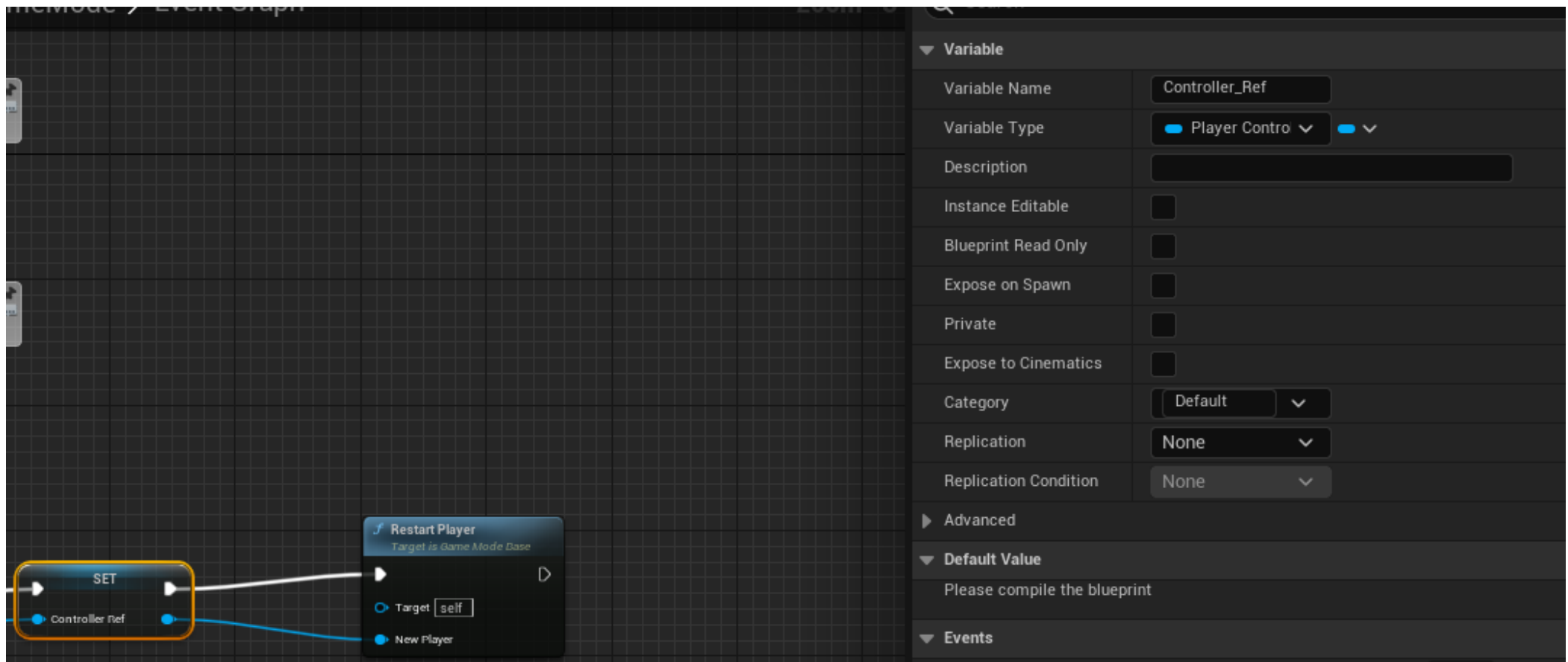
Respawns and Checkpoints

Drag out a „Restart Player“ Node.



Respawns and Checkpoints

Rename the „Set“ Node Variable to „Controller_Ref“

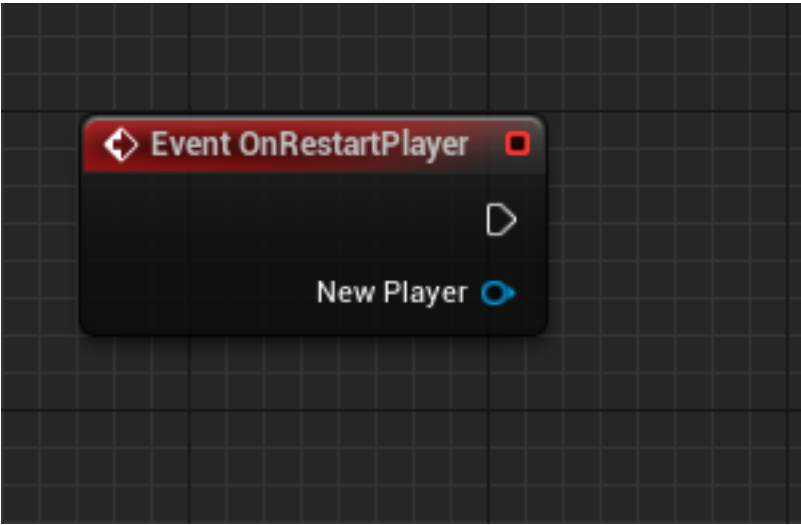
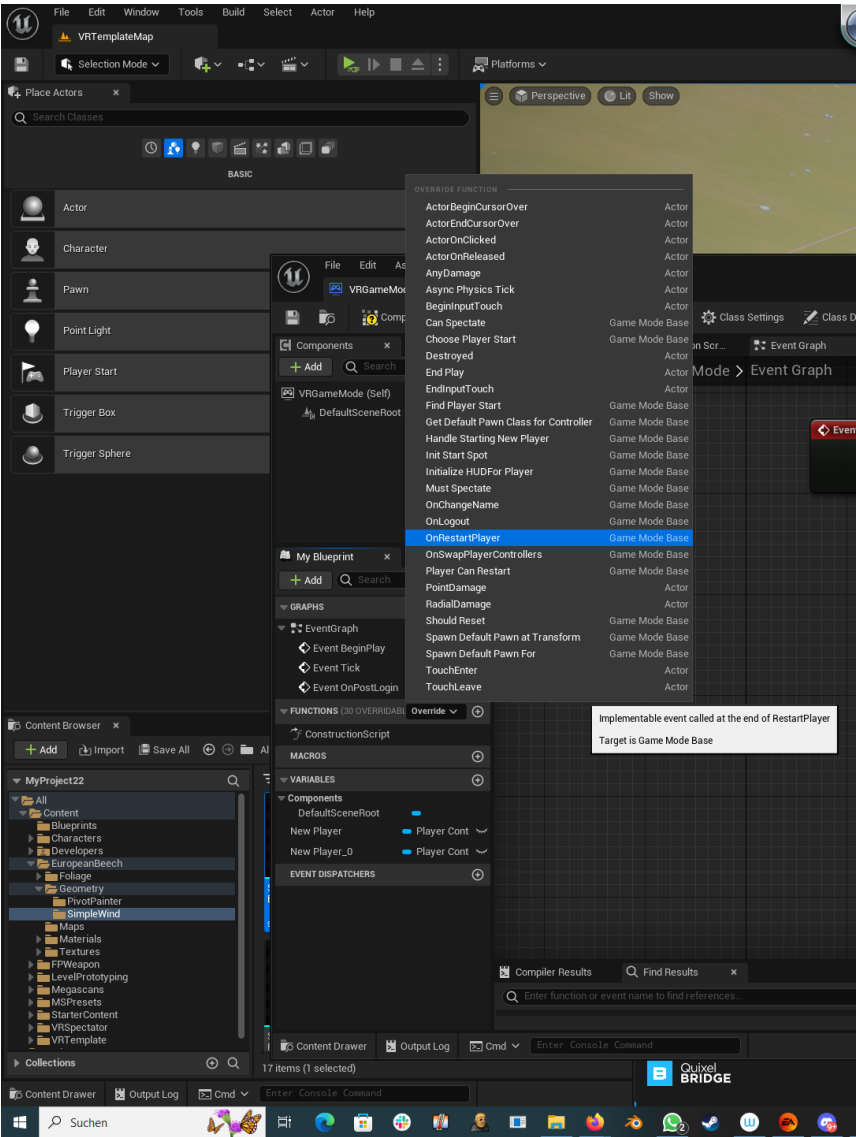


The image shows a screenshot of the Unreal Engine Blueprint Editor. On the left, a Blueprint graph is visible with a 'SET' node and a 'Restart Player' node. The 'SET' node is highlighted with a yellow box, and its variable is labeled 'Controller_Ref'. The 'Restart Player' node has a 'Target' set to 'self' and a 'New Player' input. On the right, the 'Variable' configuration panel is open, showing the following settings:

Variable	
Variable Name	Controller_Ref
Variable Type	Player Contro <input type="checkbox"/> <input checked="" type="checkbox"/>
Description	
Instance Editable	<input type="checkbox"/>
Blueprint Read Only	<input type="checkbox"/>
Expose on Spawn	<input type="checkbox"/>
Private	<input type="checkbox"/>
Expose to Cinematics	<input type="checkbox"/>
Category	Default <input type="checkbox"/>
Replication	None <input type="checkbox"/>
Replication Condition	None <input type="checkbox"/>
▶ Advanced	
▼ Default Value	
Please compile the blueprint	
▼ Events	

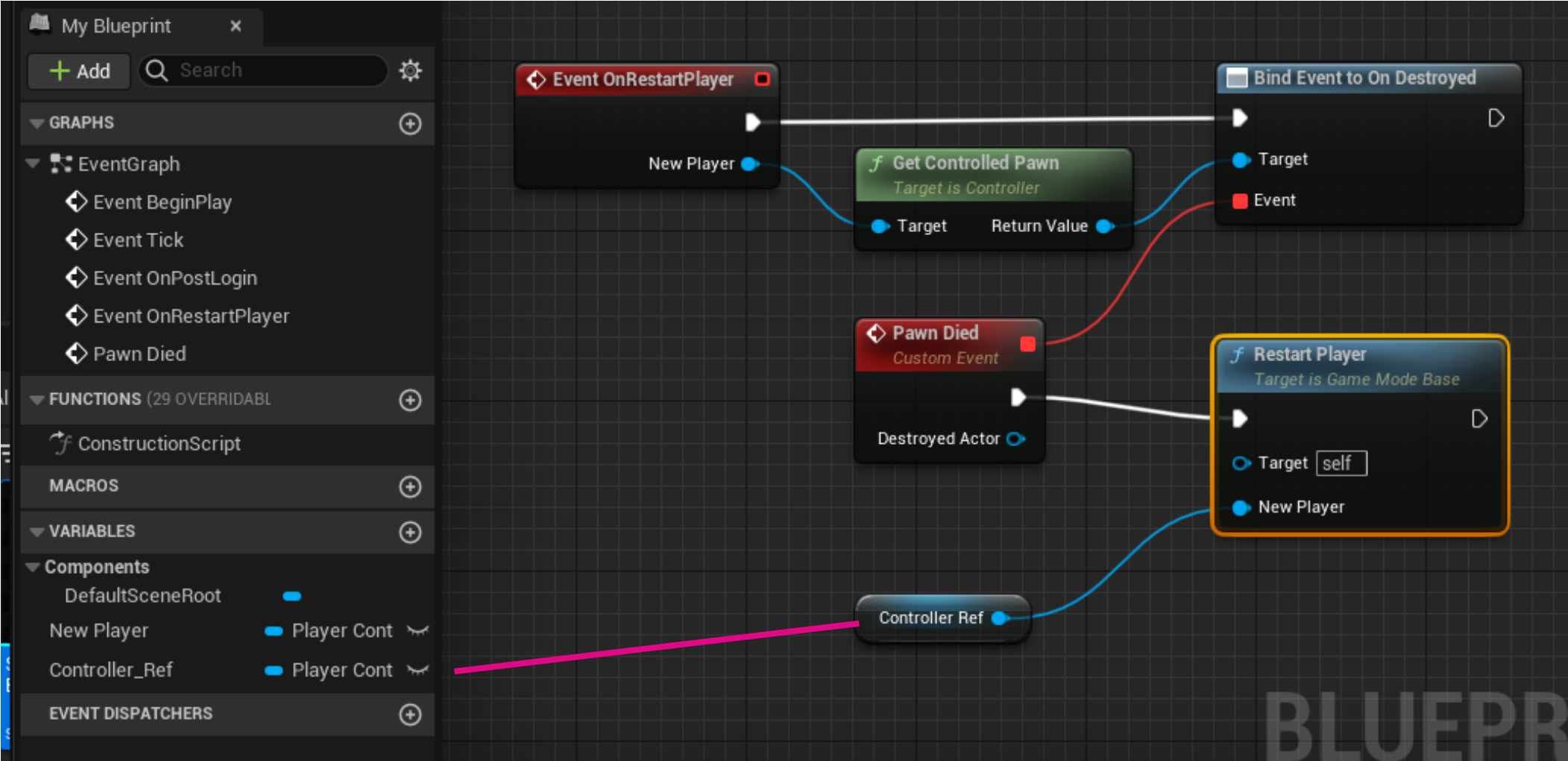
Respawns and Checkpoints

Add an Event „OnRestartPlayer“



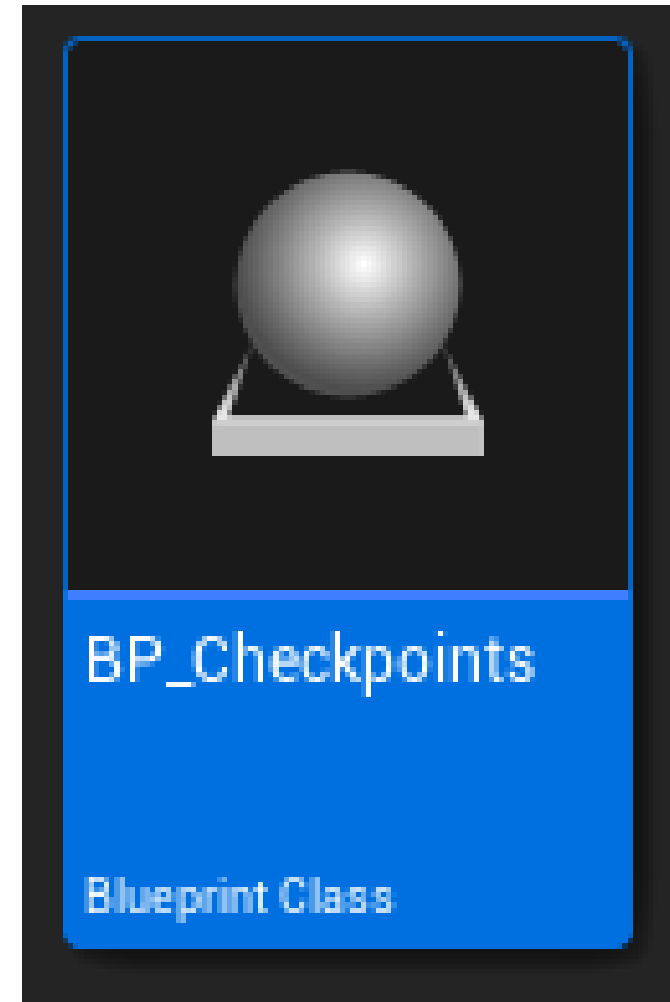
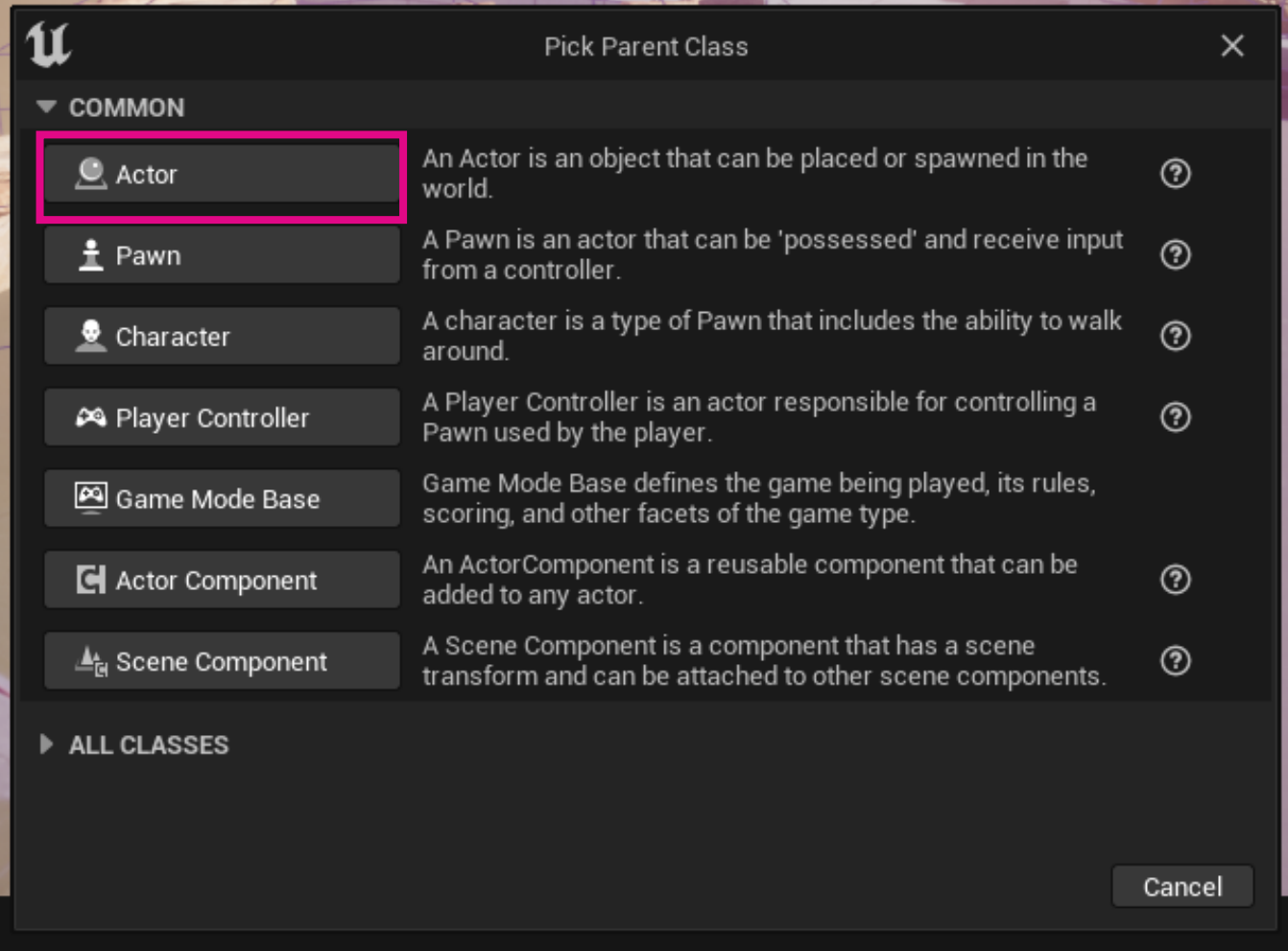
Respawns and Checkpoints

Build the Setup shown below, then compile.



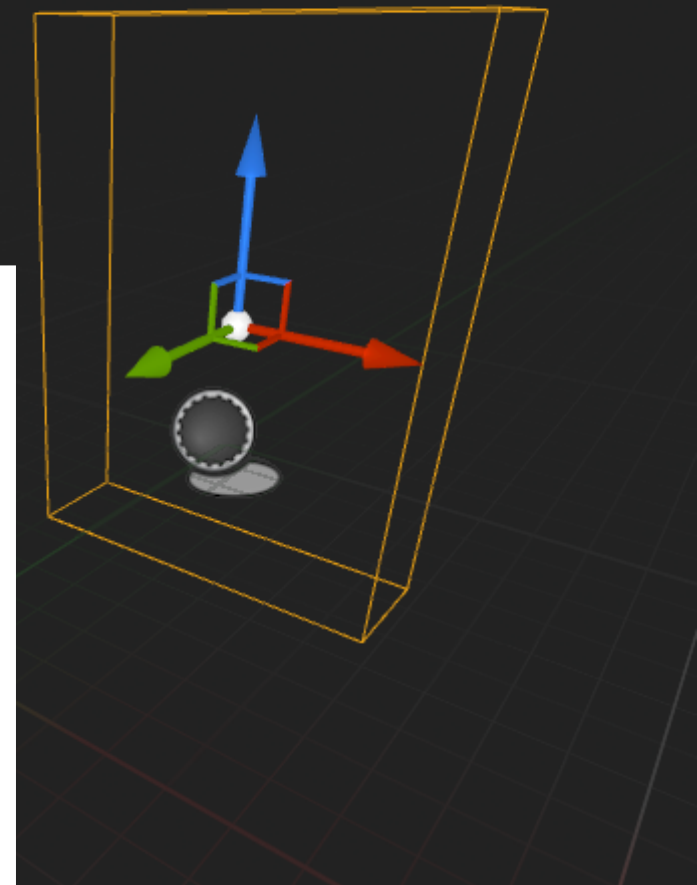
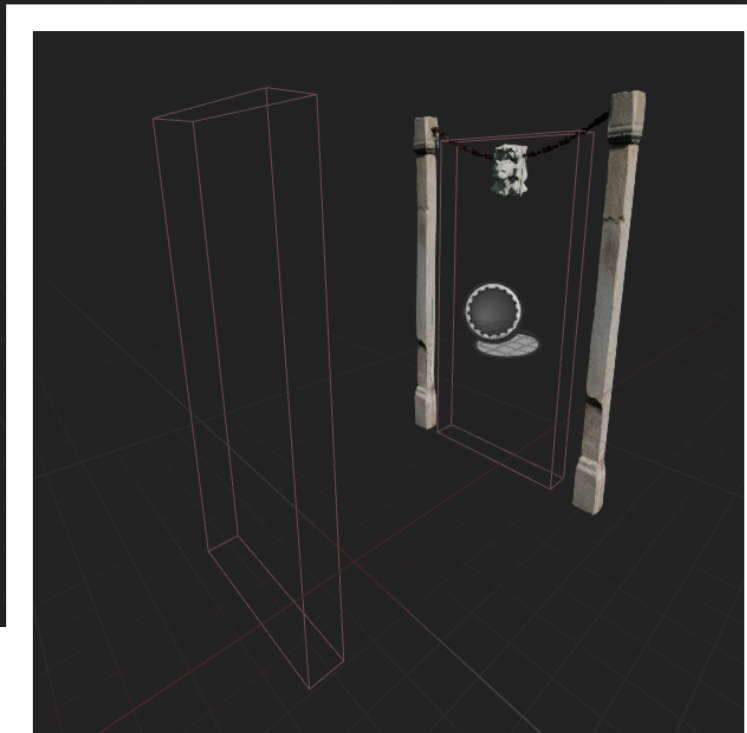
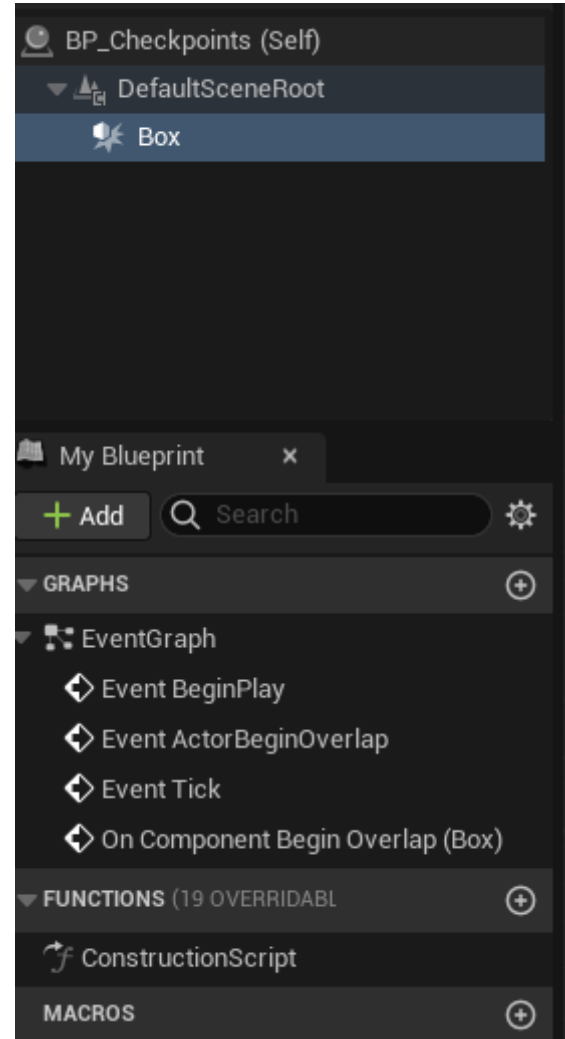
Respawns and Checkpoints

Create a new Blueprint and call it „BP_Checkpoints“



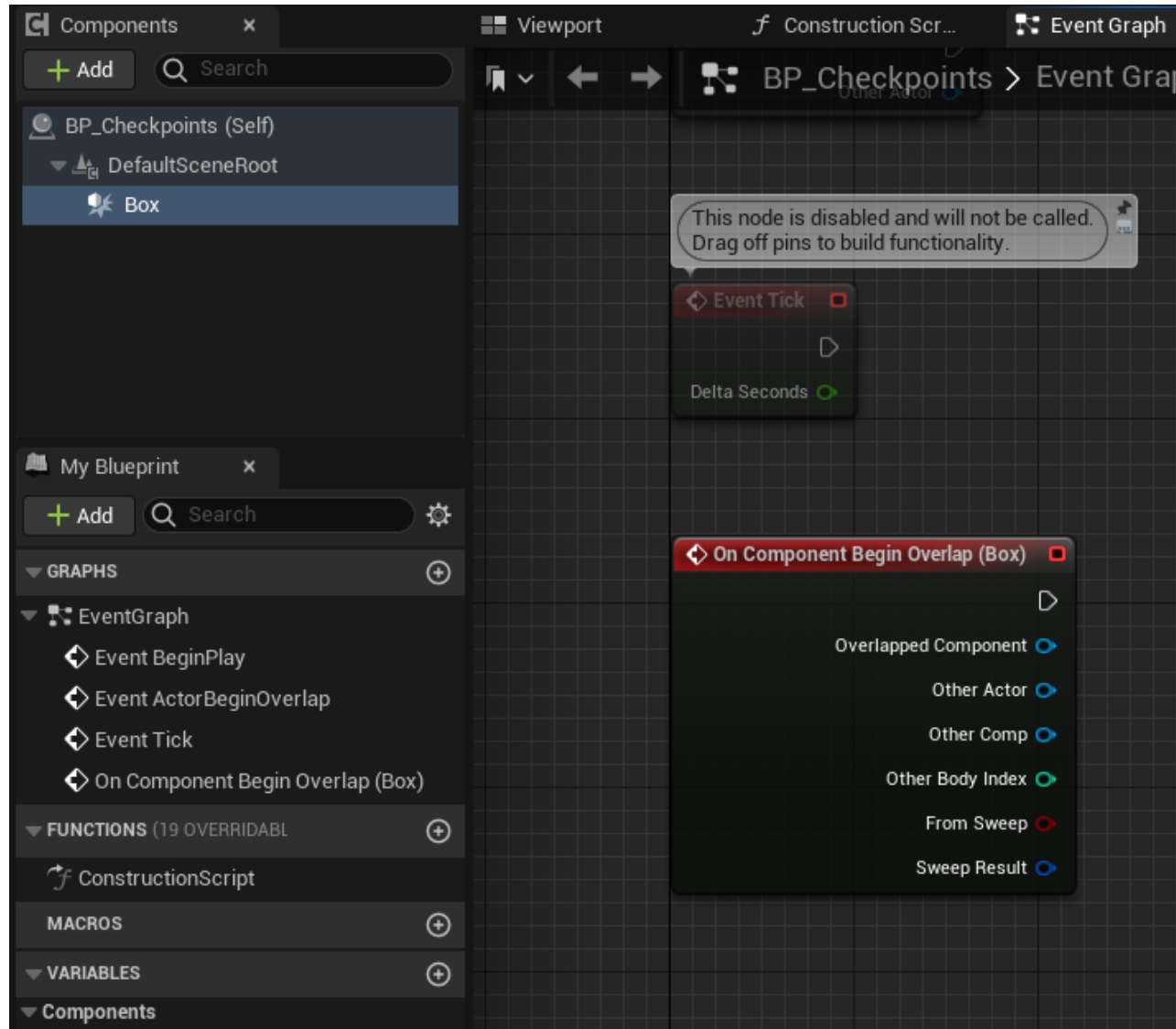
Respawns and Checkpoints

Add a Box Collision and scale it to your needs. Think of it as a Gate you pass through to activate your Respawn Point.



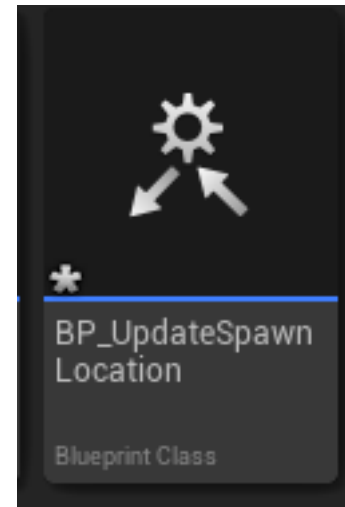
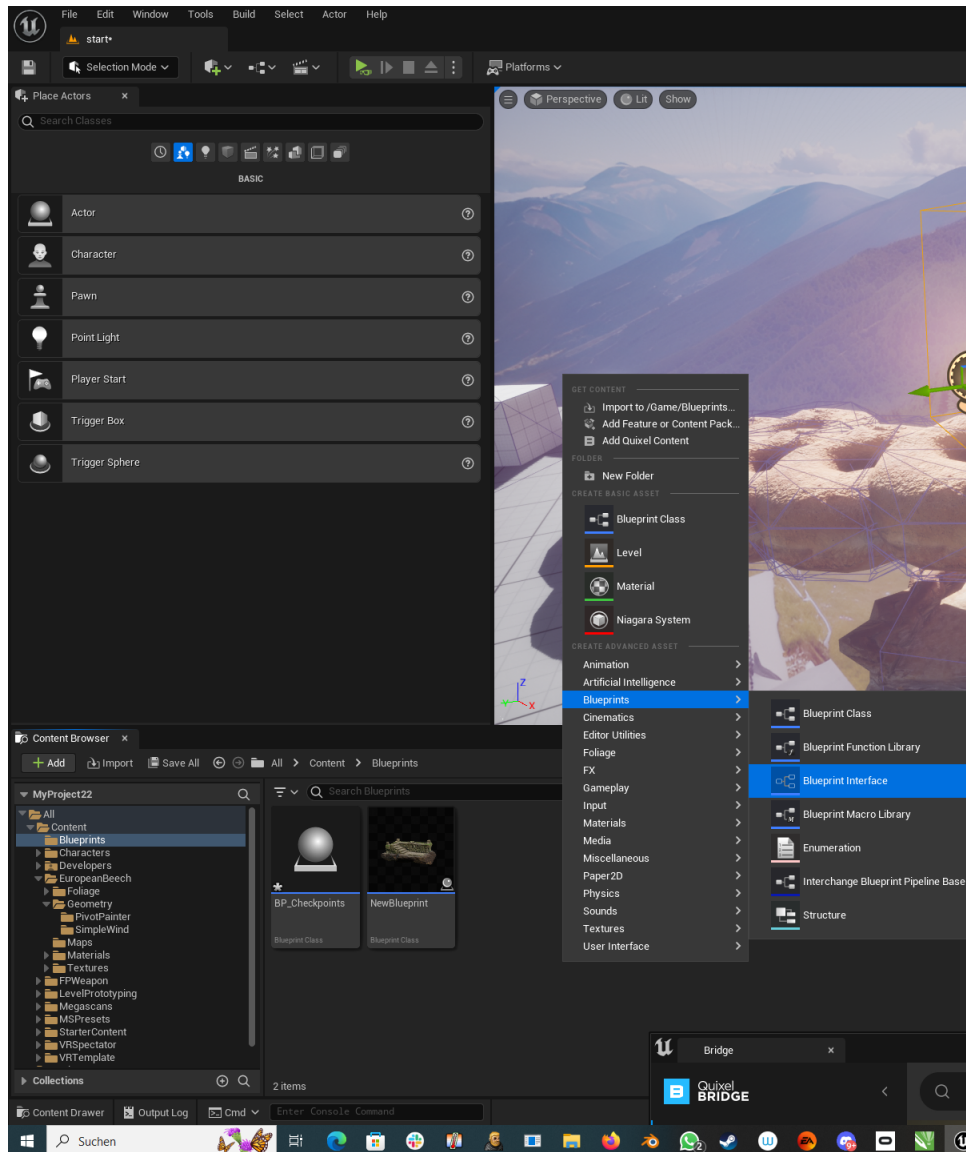
Respawns and Checkpoints

Add a „On Component begin Overlap“ Event for the Box



Respawns and Checkpoints

Now we need a Blueprint Interface. Go to your Blueprints Folder and right click, add, rename etc.



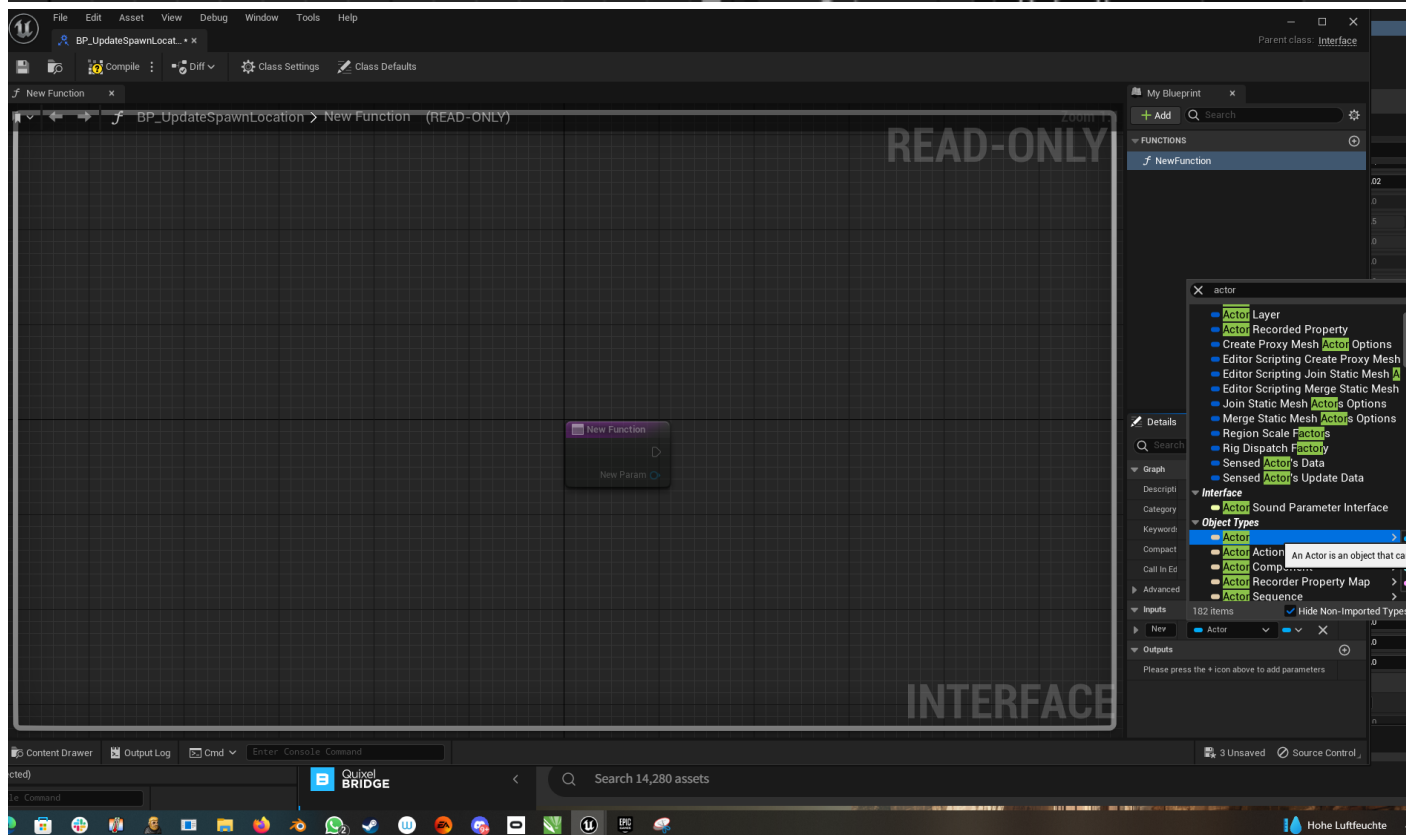
Respawns and Checkpoints

Go back to the „Gamemode Blueprint“ and under „Class Settings“ add the Interface we just created, save and compile.

The screenshot displays the Unreal Engine 5.3 interface for editing the VRGameMode+ blueprint. The top menu bar includes File, Edit, Asset, View, Debug, Window, Tools, and Help. The main viewport shows the Event Graph with several nodes: Event OnPostLogin, SET, Restart Player, Event OnRestartPlayer, Get Controlled Pawn, Bind Event to On Destroyed, Pawn Died, and another Restart Player node. The Details panel on the right is open to the 'Class Settings' tab, where the 'Interfaces' section is highlighted in pink. Under 'Implemented Interfaces', the 'BP_UpdateSpawnLocation' interface is selected, with a tooltip showing 'BP Update Spawn Location'. The bottom status bar indicates '3 Unsaved' and 'Source Con'.

Respawns and Checkpoints

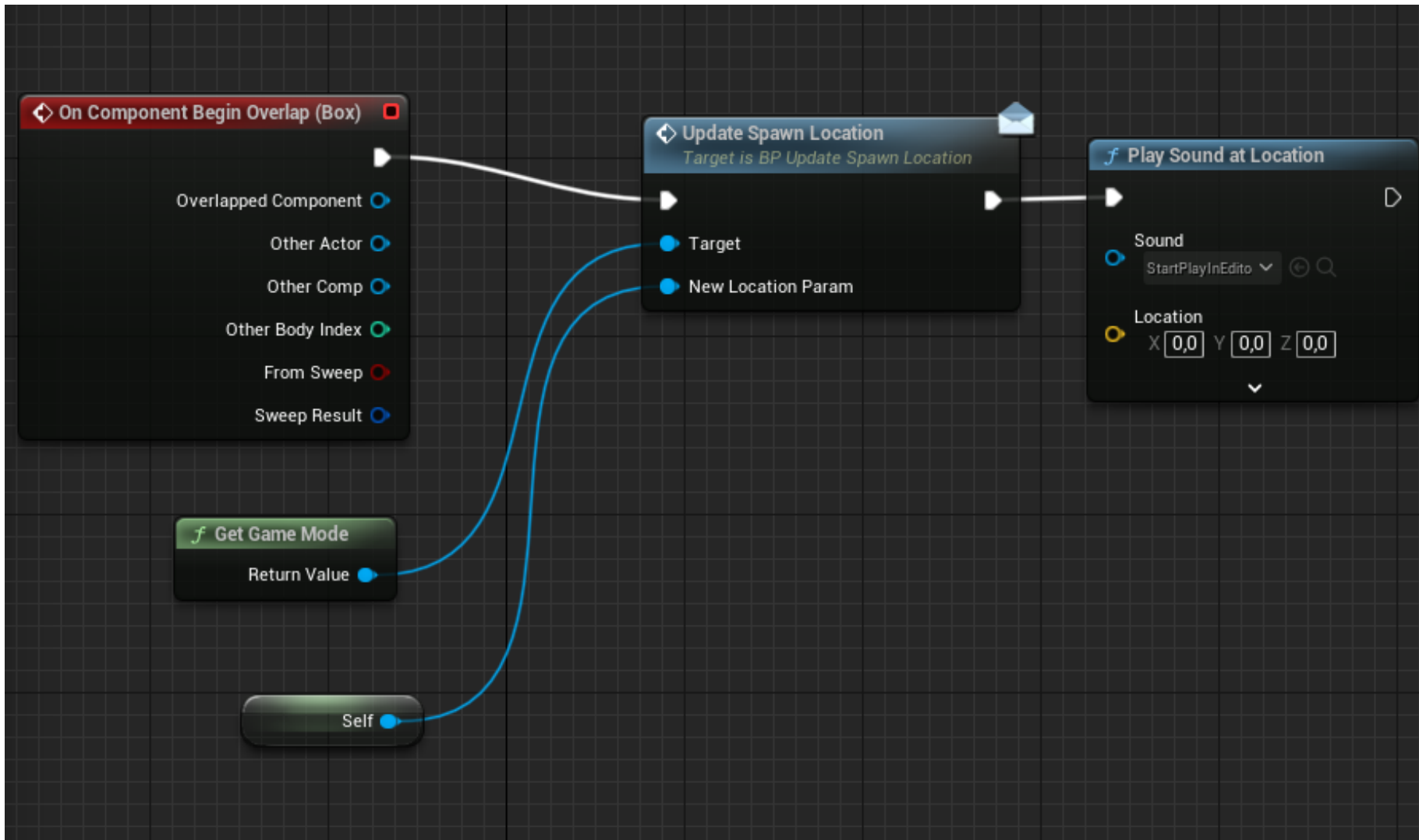
After opening the Interface, select the Box and rename it to „Update Spawn Location“. Under Inputs rename it to „NewLocationParam“ and make it an Actor (beige). Save and compile.



Respawns and Checkpoints

Now lets go back to the BP_Checkpoints Blueprint. Add an Update Spawn Location. From the Target drag out a „Get Game Mdoe“ and from „NewLocation Param“ drag out a „Get reference to self“.

I wanted some kind of audio confirmation for my checkpoint so i added a „Play sound at Location“. Save and compile.



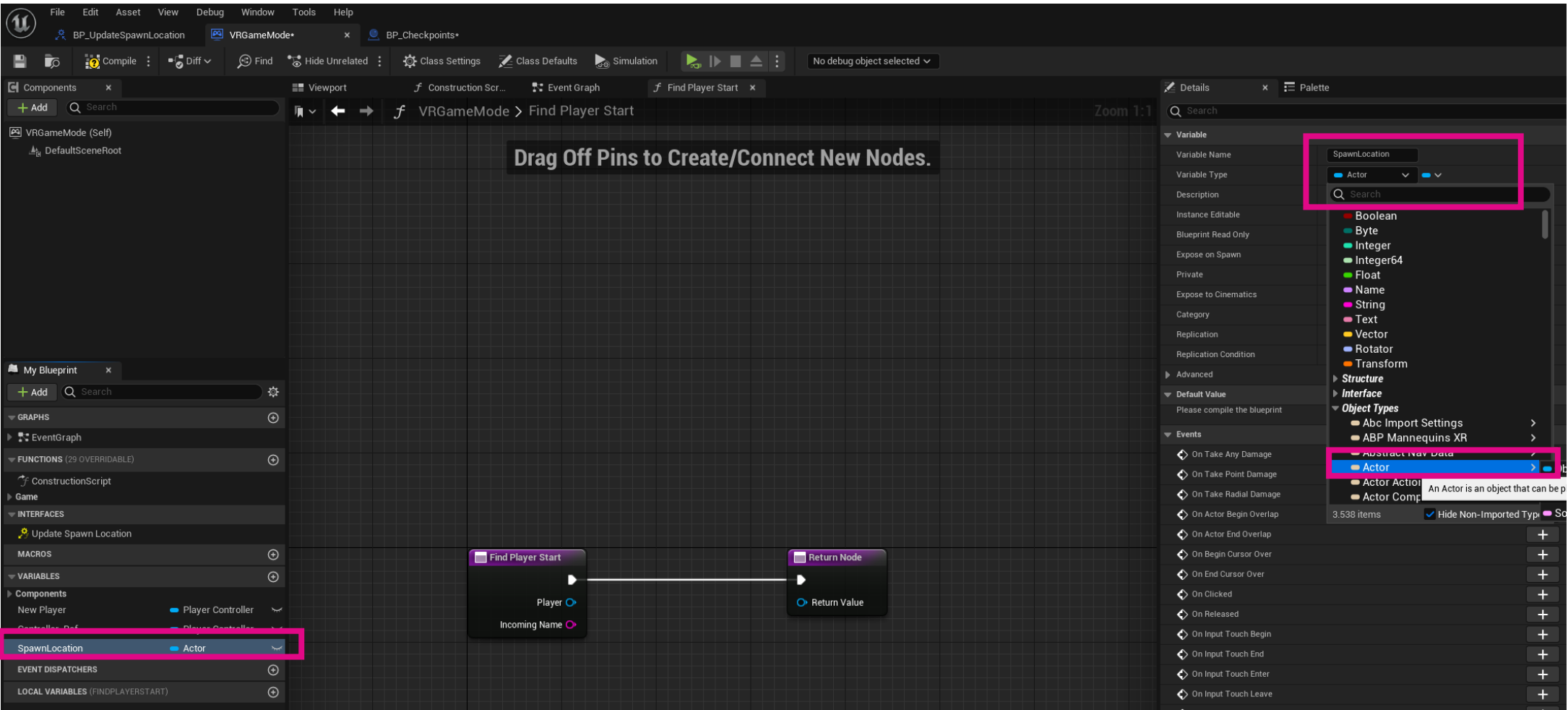
Respawns and Checkpoints

Back in the Gamemode Blueprint go to Functions - Override - Find Player Start

The screenshot shows the Unreal Engine Blueprint Editor interface. On the left, the 'My Blueprint' panel is open, displaying a search bar and a list of categories: GRAPHS, FUNCTIONS (30 OVERRIDABLE), INTERFACES, MACROS, VARIABLES, Components, and EVENT DISPATCHERS. The 'FUNCTIONS' category is expanded, and a search for 'find' has been performed. The search results show 'Find Player Start' from the 'Game Mode Base' class. A tooltip for this function is visible, stating: 'Return the specific player start actor that should be used for the next spawn. This will either use a previously saved startactor, or calls ChoosePlayerStart. Target is Game Mode Base'. In the main blueprint workspace, an 'Event OnRestartPlayer' node is connected to a 'New Player' node, which is currently selected.

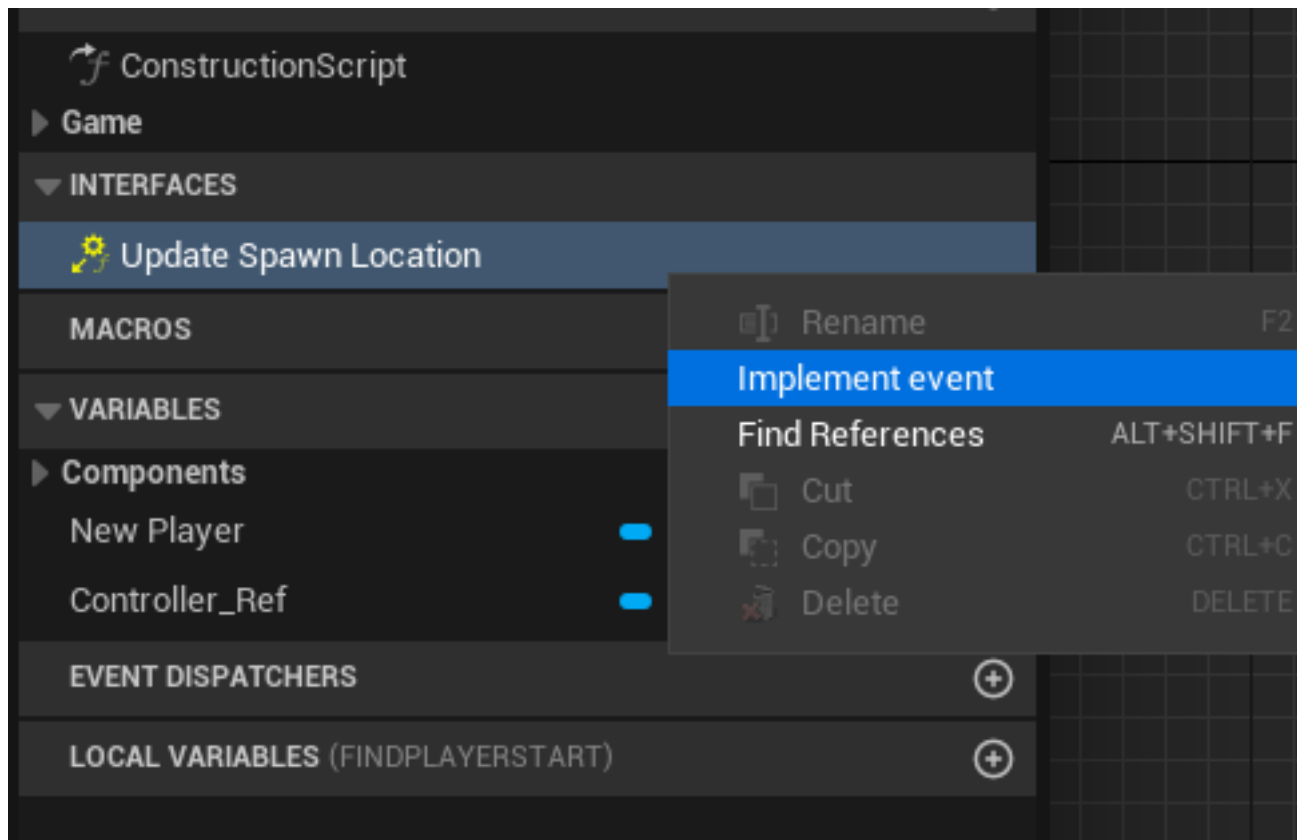
Respawns and Checkpoints

Add a new Variable called PlayerSpawn and make it an Actor



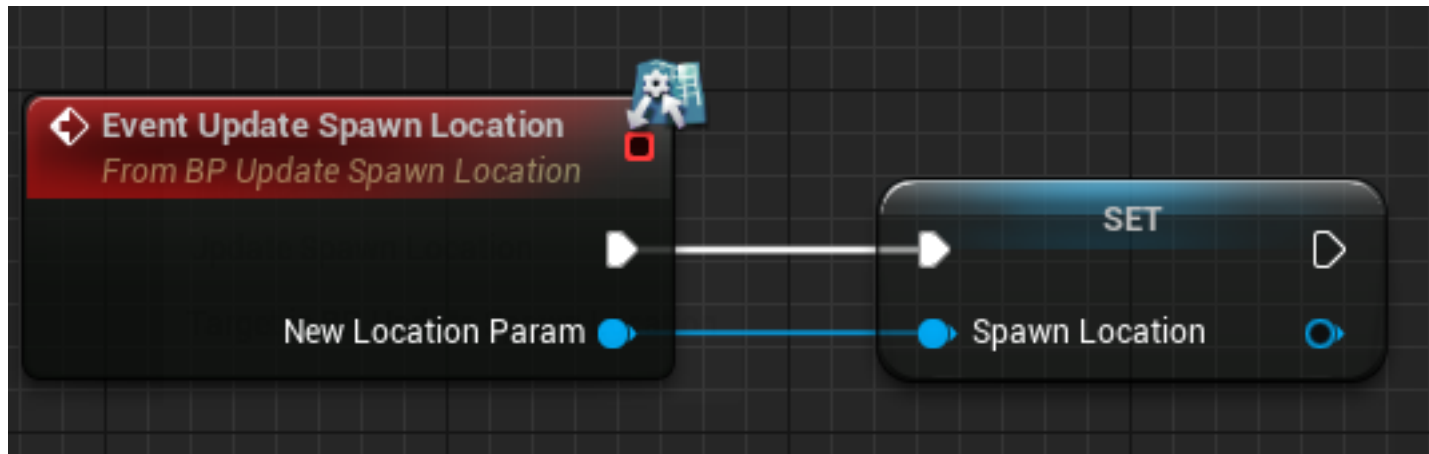
Respawns and Checkpoints

Implement the Event and go back to the Event Graph



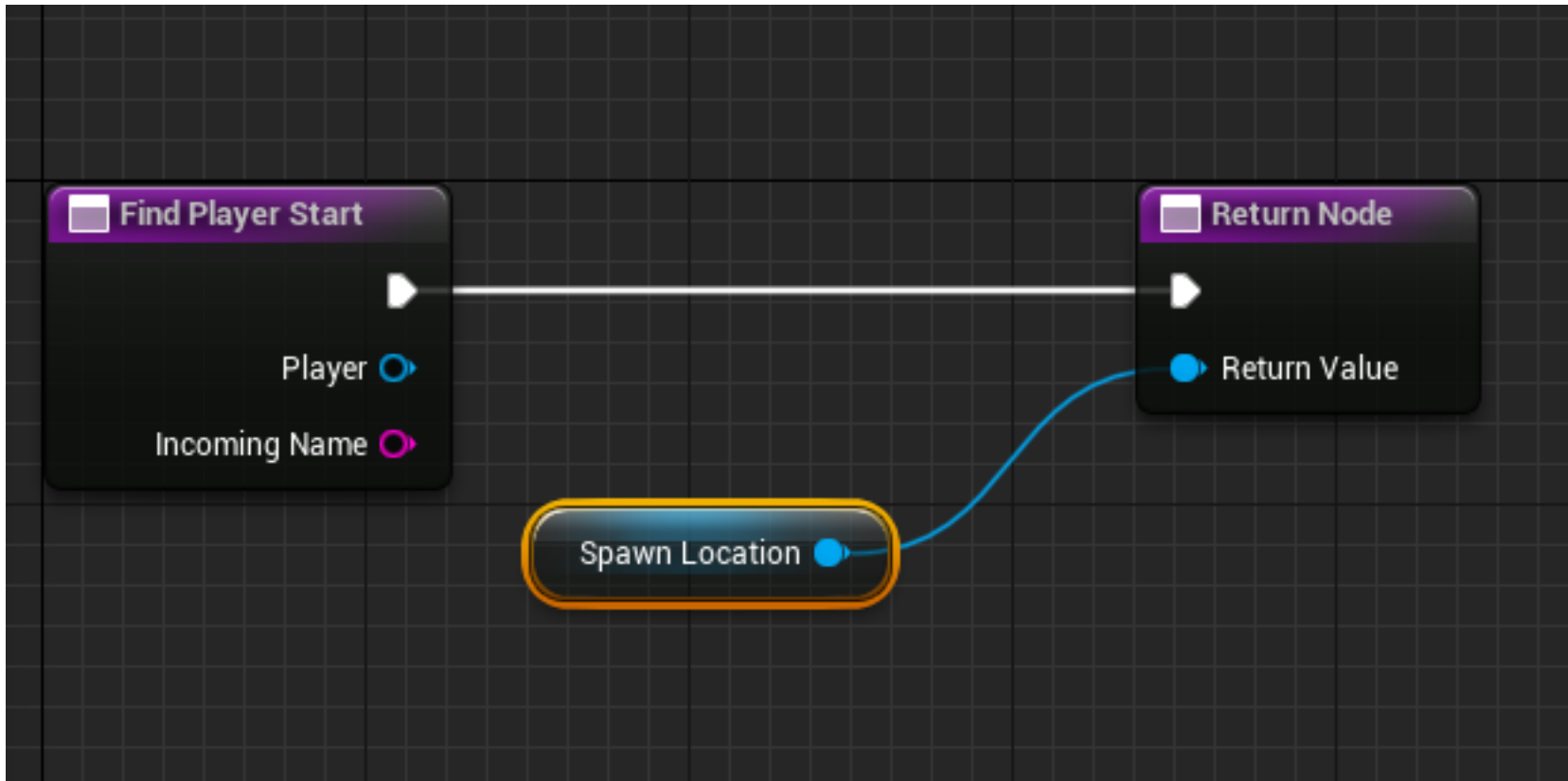
Respawns and Checkpoints

Drag and drop the spawn location (select „Set“) and connect it as shown below



Respawns and Checkpoints

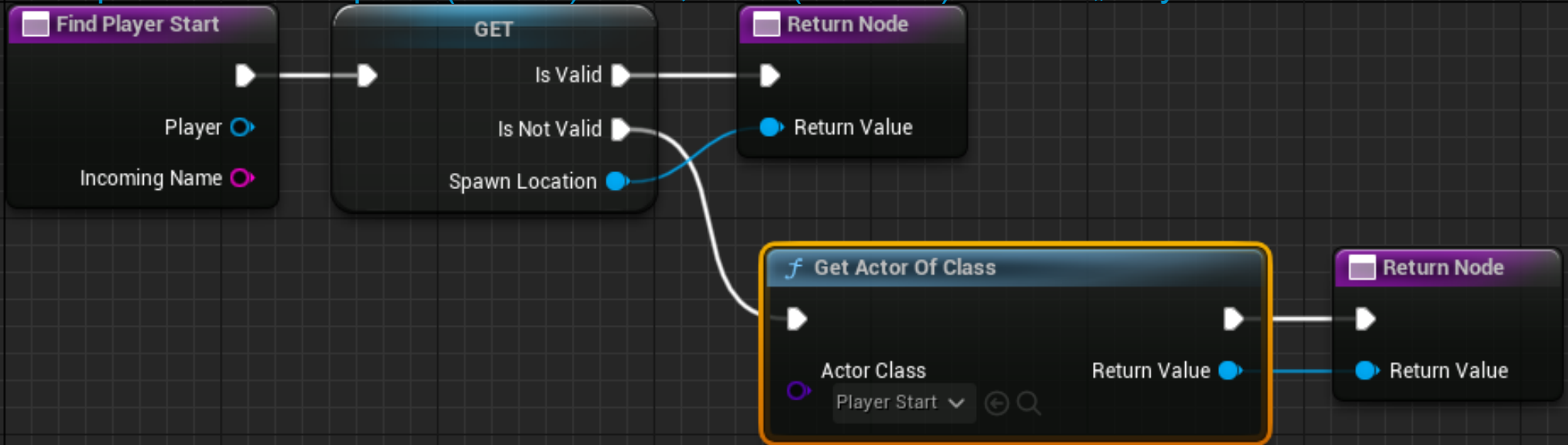
Do the same thing in the Functions Window but drag the Spawn location directly into the „ReturnValue“ Socket



Respawns and Checkpoints

The basic setup is complete, but now we are unfortunately missing the original Start Position. To remedy that lets copy the Setup below. Right click on „SpawnLocation“ and turn it into a „Validated Get“, connect it. The other Node is called „Get Actor of class“ and under Actor class select Player Start. The second Return Node can just be copied.

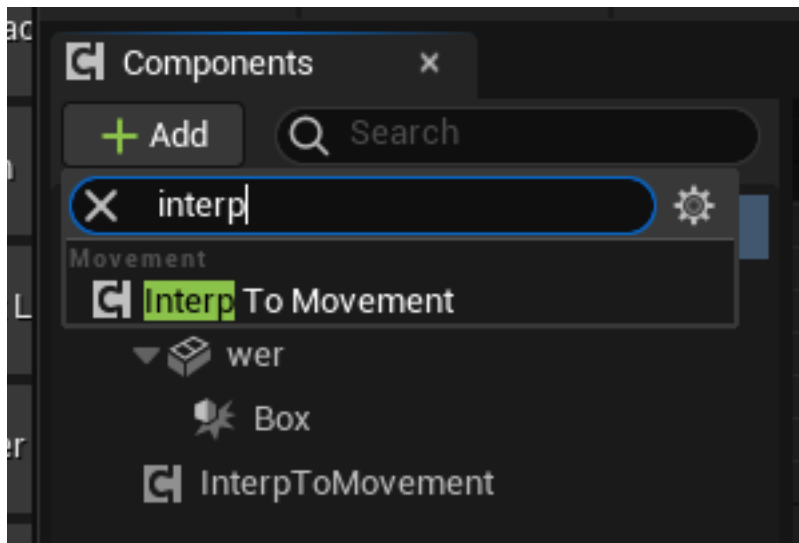
If we passed a checkpoint (is valid) use it, if not (not valid) use the „Player Start“



Simple Moving Platform

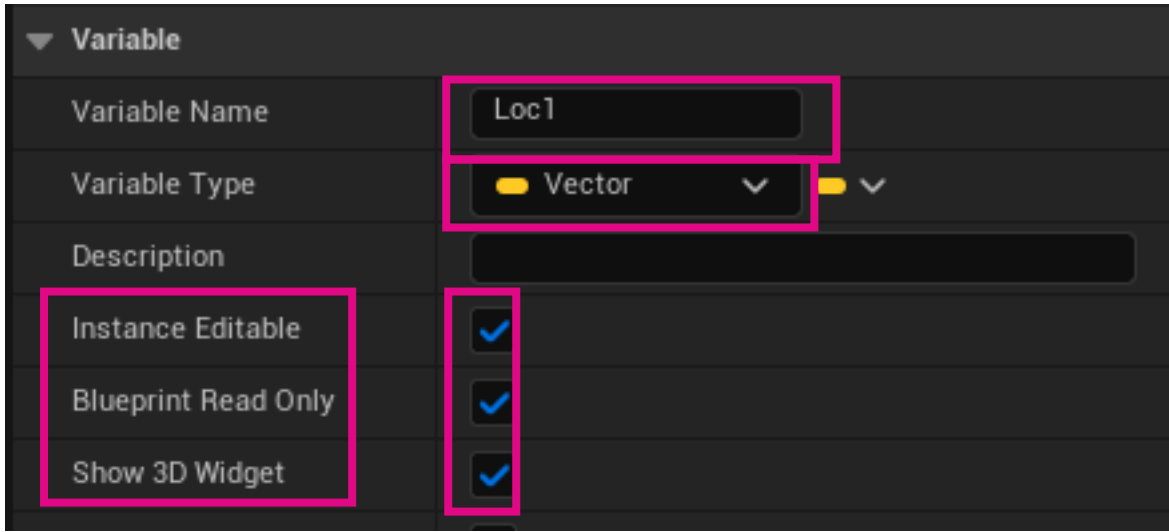
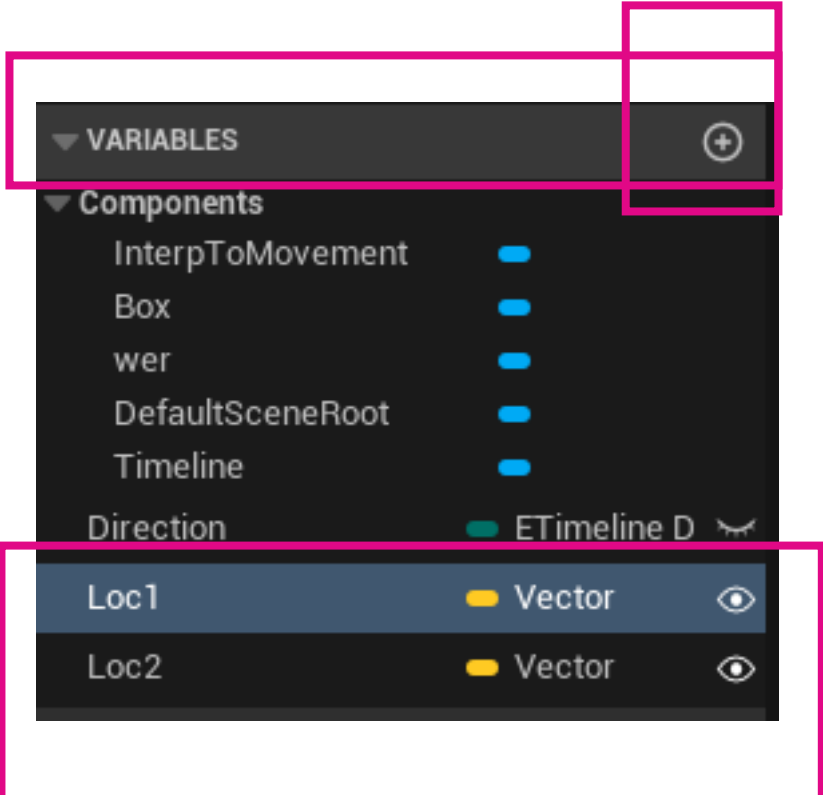
Creating a Simple moving platform between 2 Points

First, create a new **Blueprint** with a **simple Box** as a Placeholder
then, add an **„Interp to movement“ component.**



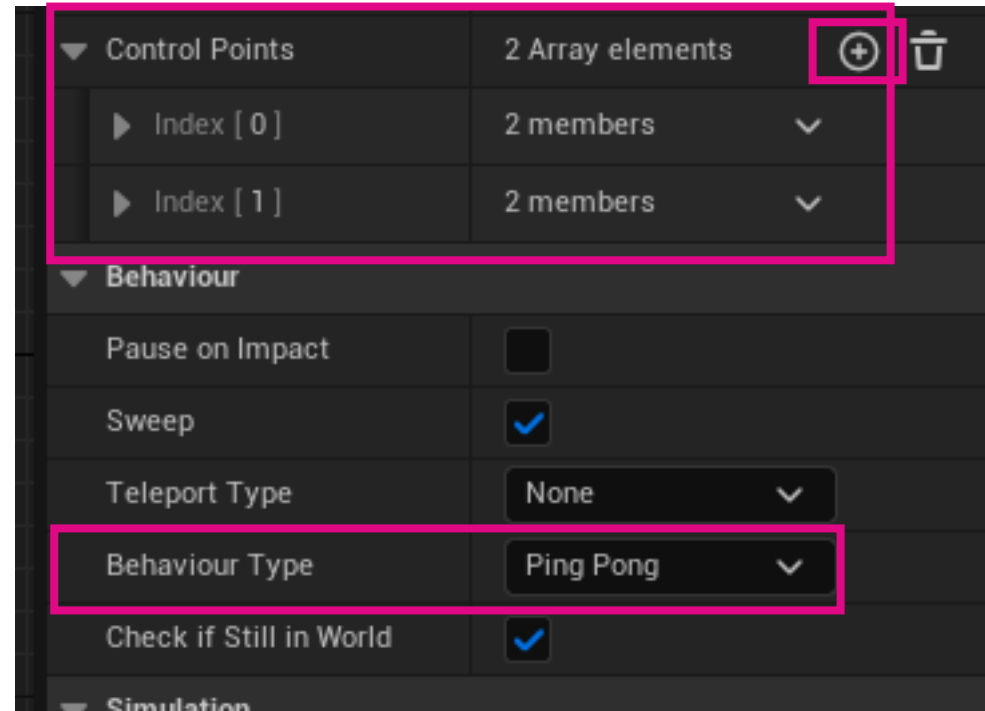
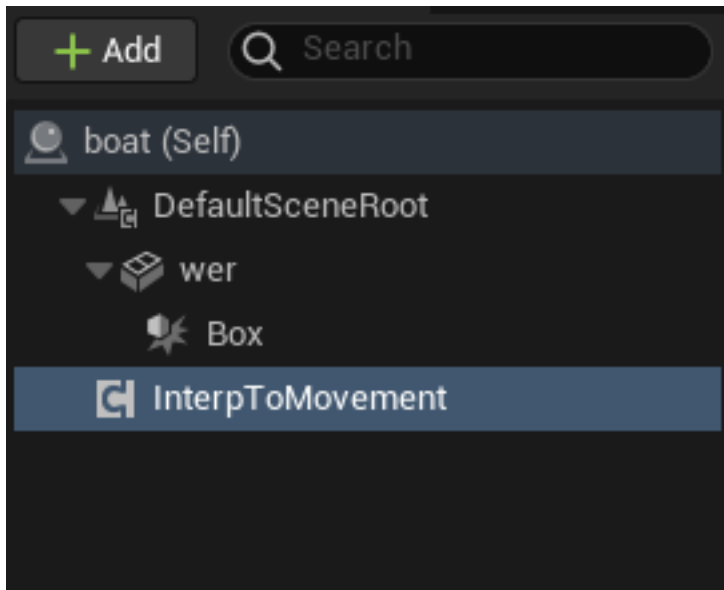
Simple Moving Platform

Add 2 new Variables and turn them into Vectors (yellow), name them Loc1 and Loc2.



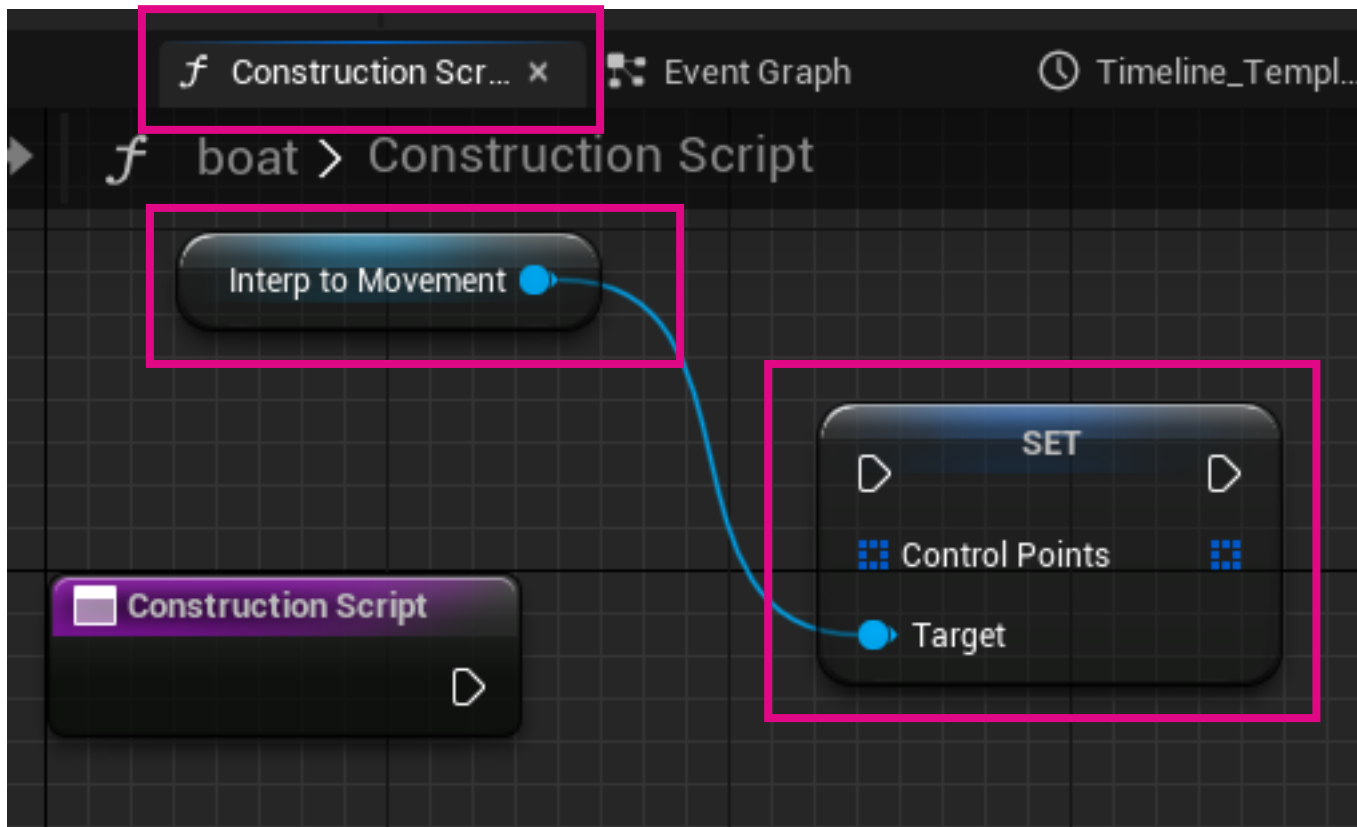
Simple Moving Platform

Select „Interp to Movement“ and on the right side under „Behaviour“ select „Ping Pong“ and under „Control“ add 2 Control Points



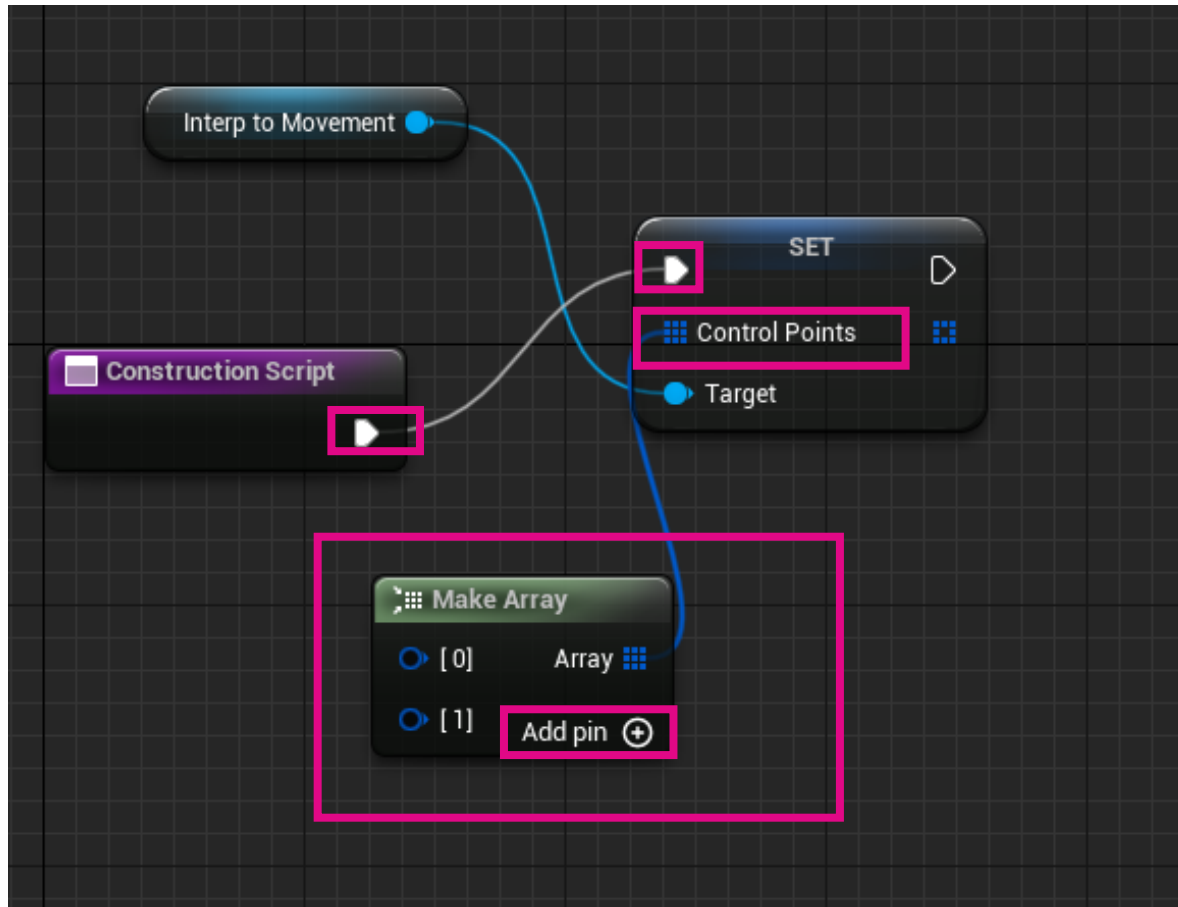
Simple Moving Platform

Go to control Scripts, drag in „Interp to movement“ and add a „Set Controlpoints Variable Node



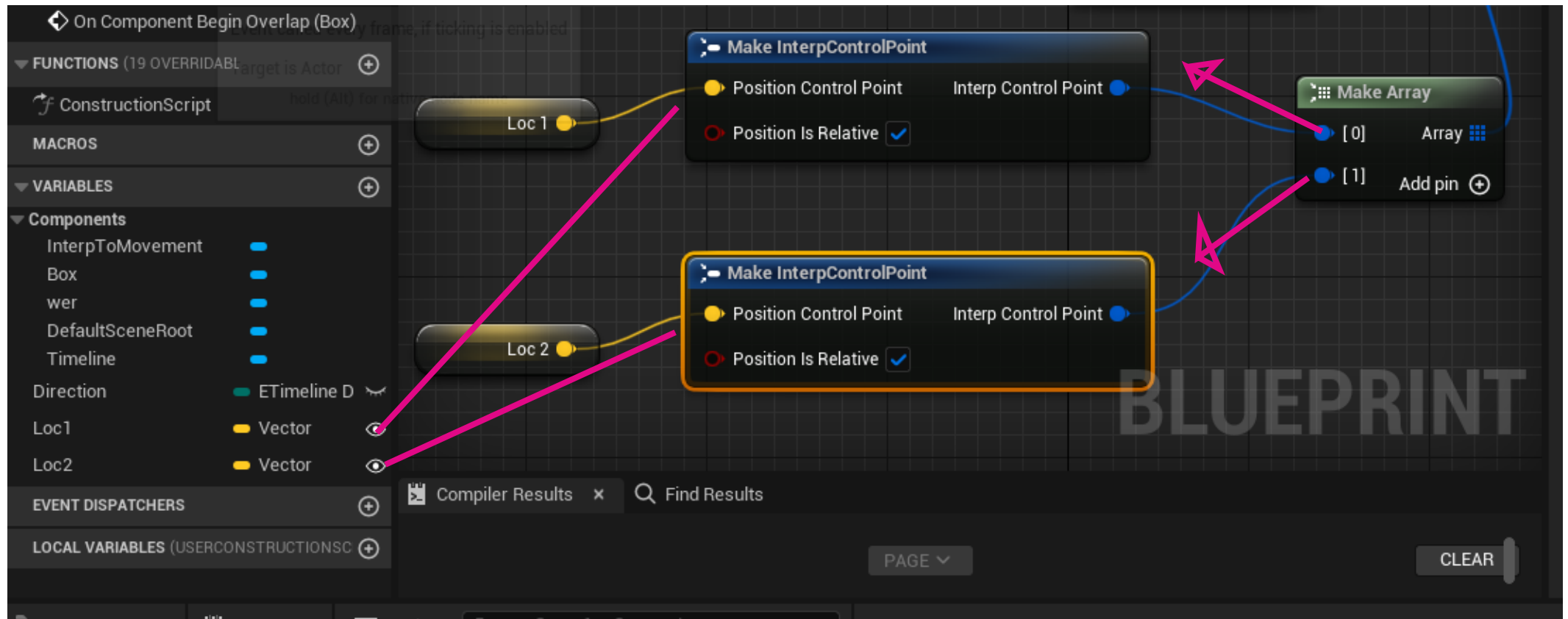
Simple Moving Platform

Connect the „Construction Script“ to the „Set Controlpoints Node“.
From the „Control Points“ Socked drag out a „Make Array Node“
and add a second pin.



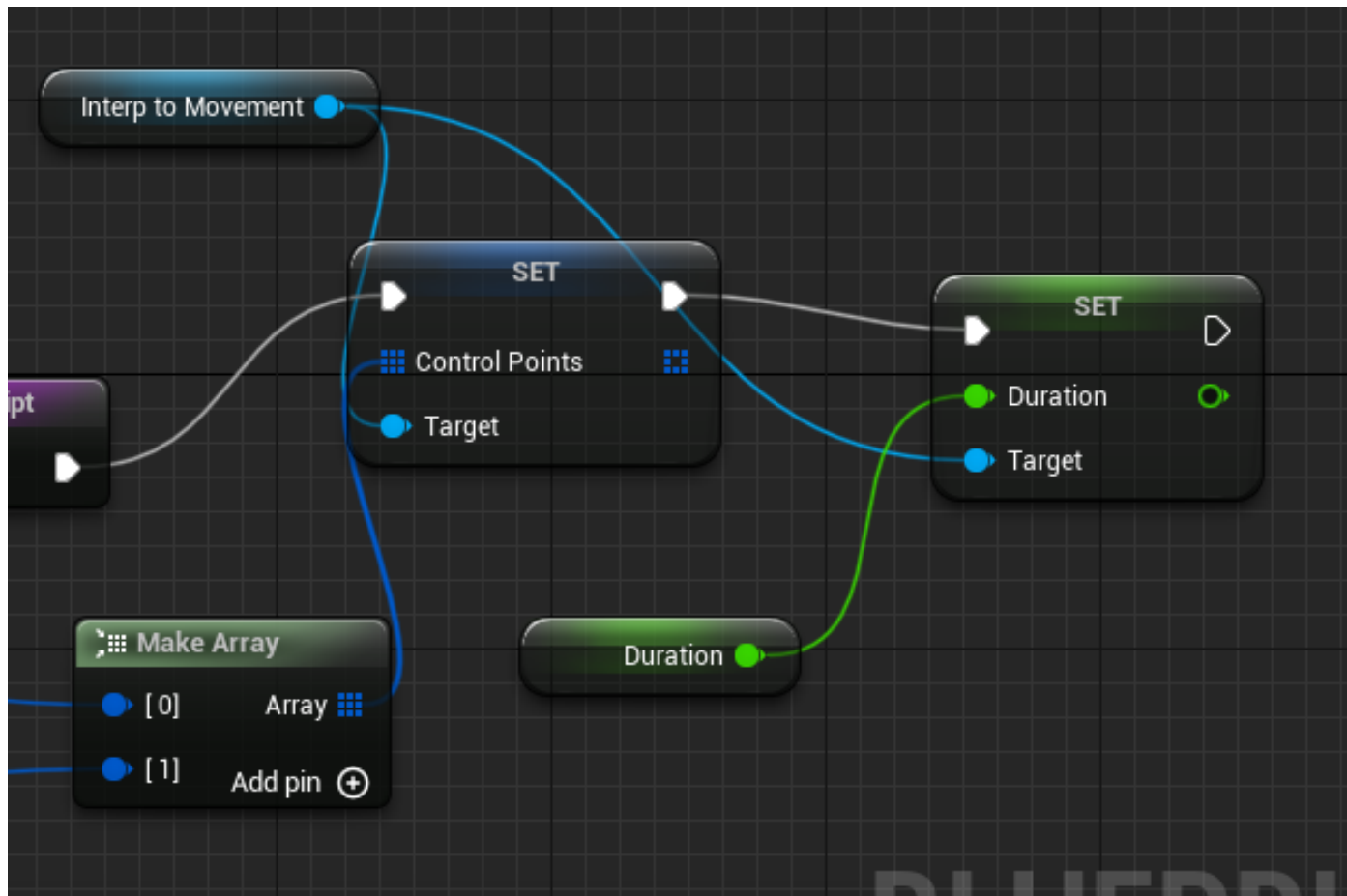
Simple Moving Platform

From the two Pins drag out 2 „Make InterpControlpoint“ Nodes and connect the „Loc1“ and „Loc2“ Vectors you created to the „Position Sockets.“



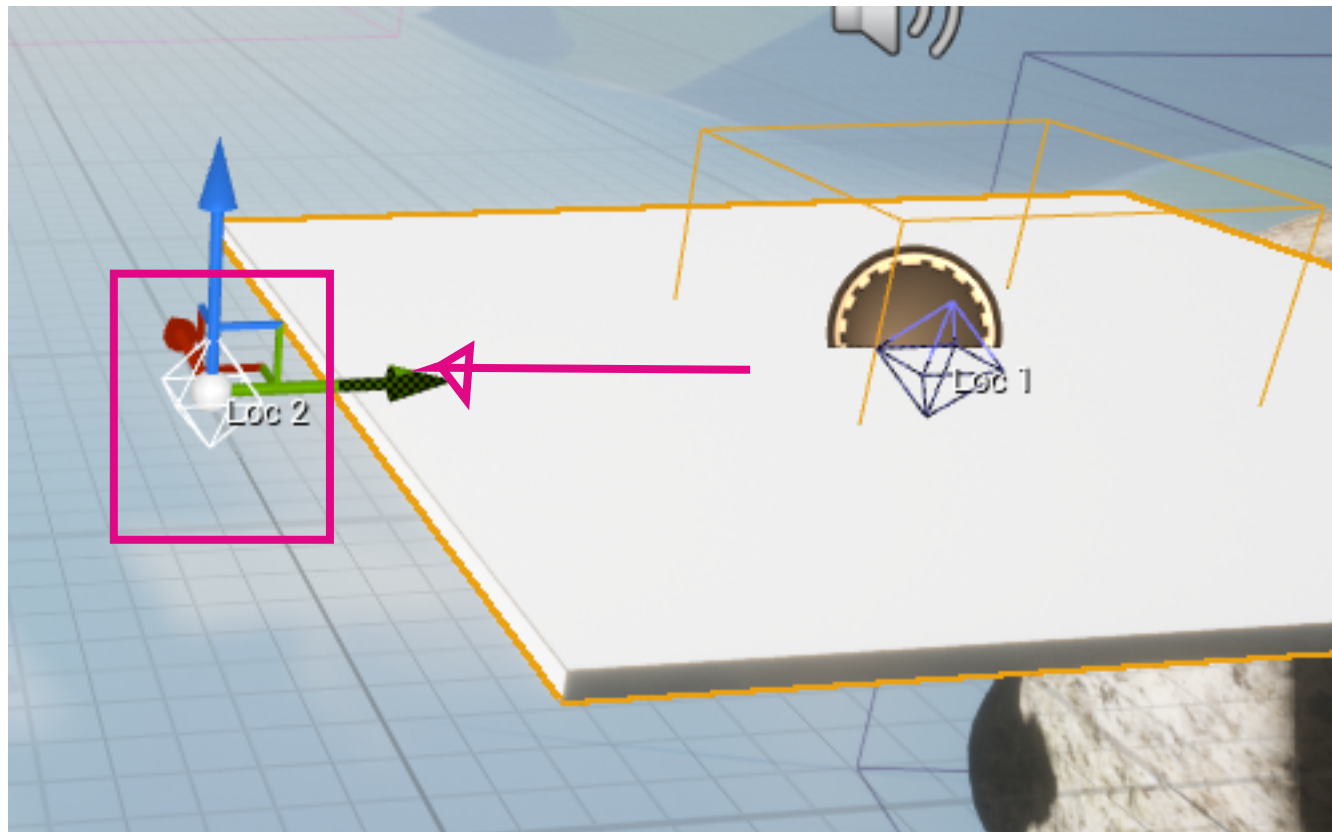
Simple Moving Platform

Finally drag out a „Set Duration Node“ from the Set Control „Points Node“ connect the „Interp to movement“ to target and add a Float Variable (set to float, click yes if asked, instance editable on the right side side and after compiling set a Value under „Default Value“



Simple Moving Platform

In the Editor you can now drag out the second location to where you want to go, and voila, finito. if it's too slow, just change the Duration Value.



Simple Moving Platform

Now you can go back to the blueprint and replace your Placeholder with a properly designed Platform/Boat/Whatever.



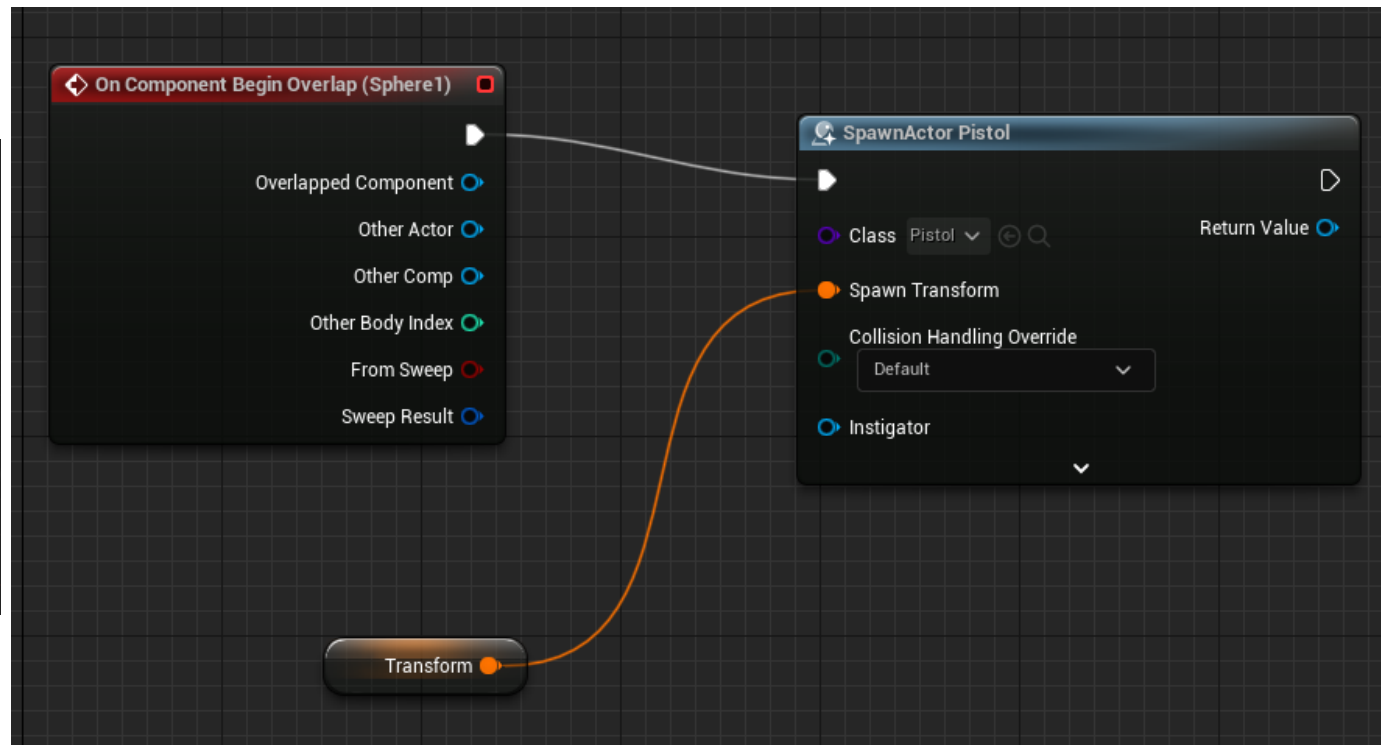
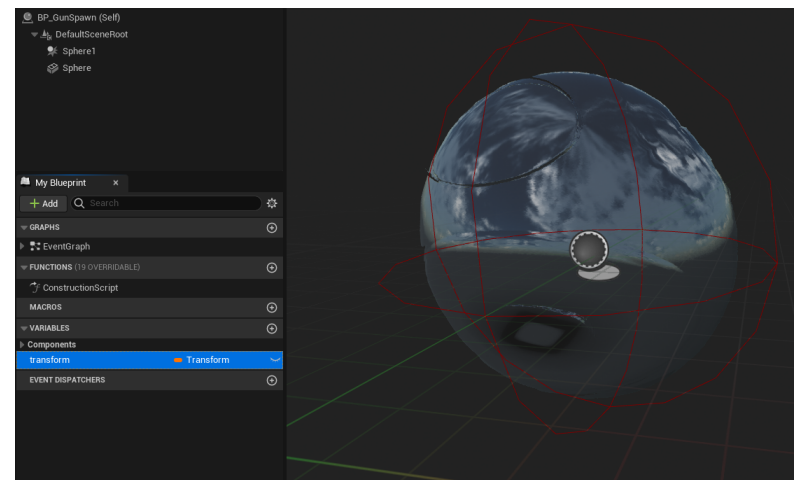
Simple „Dispenser“

Create a new Blueprint, add a Sphere, give it a transparent / Glass Material and add a „Collision Sphere“ around it.

Create an „Overlap“ Event for the Collision Sphere.

To make things „appear“ you need a „Spawn Actor From Class“ Node. Under „Class“ select whatever you want to spawn.

The trickier Part is to set the Spawn Location. For this make a Transform (orange) Variable, combine and choose your „Spawnpoint“ Coordinates. Be very careful that the „Spawnpoint“ and the „Dispenser Sphere“ don't overlap.

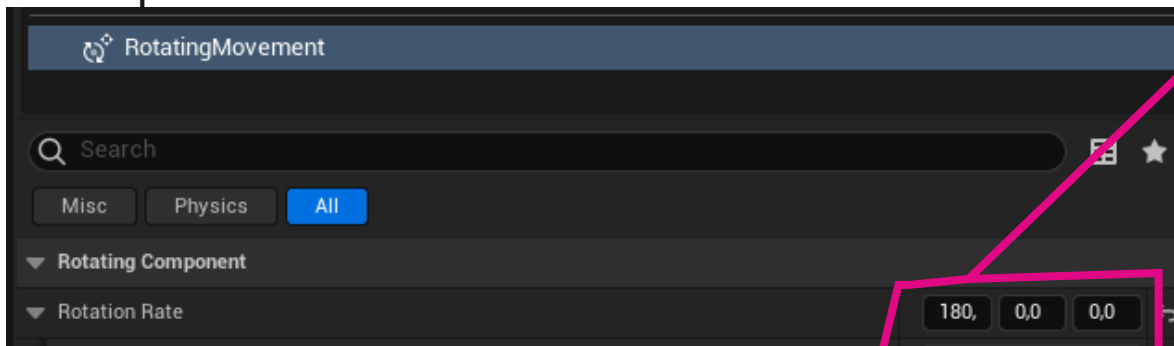


Simple „Dispenser“

If any other collision shapes „touches“ the Dispenser shape a new Object will appear. Of course you can set up an „Automatic trigger Arm“ like this rather easily.



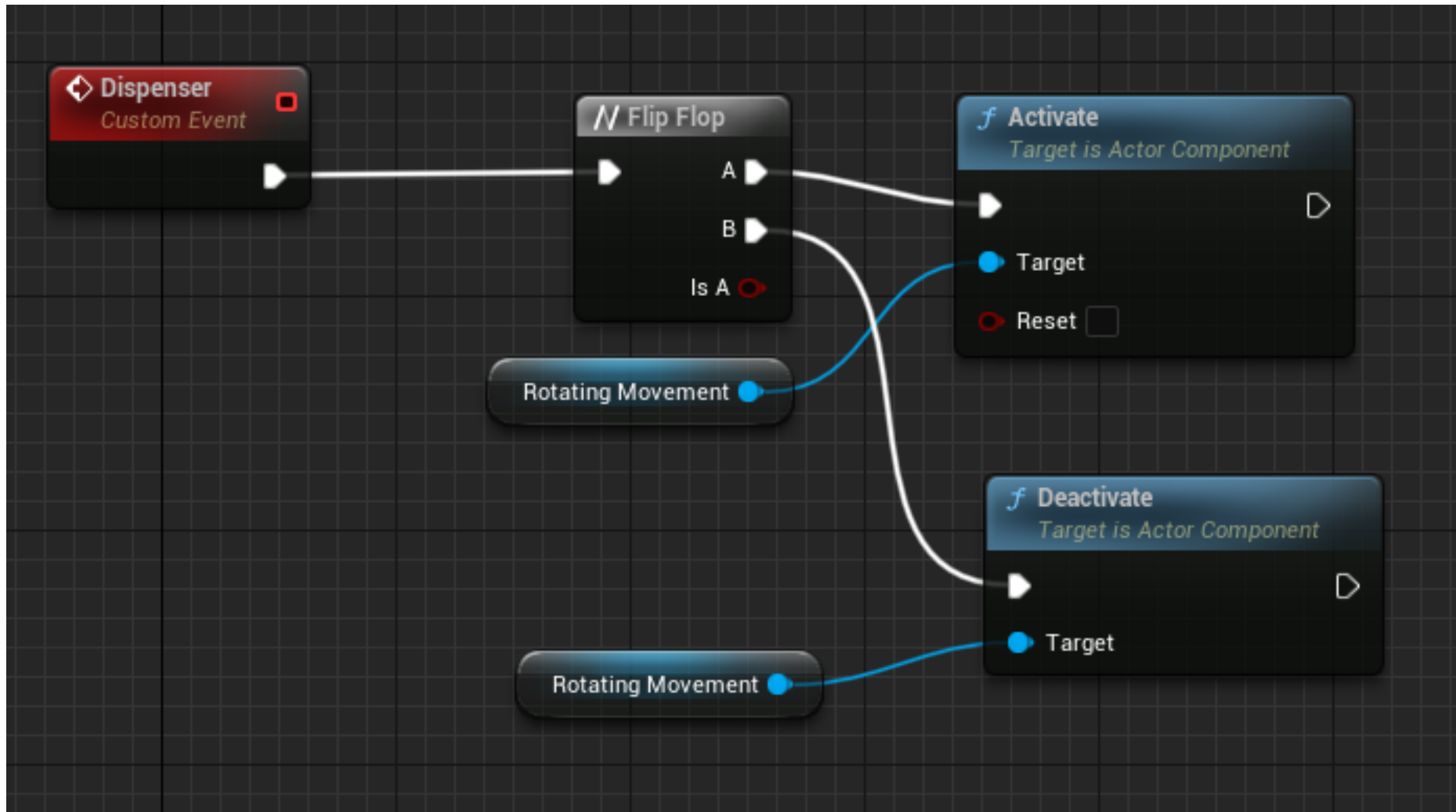
Drag an Object into the Scene, set it to **Movable**, add a „Box Collision“ and a „Rotating Movement“ Component and voila. Set the rotation axis and Speed here: Make sure that both collision shapes overlap



This can be very performance heavy, so meaby ist better to use a timeline with a fixed amount of „rotations“ over time or

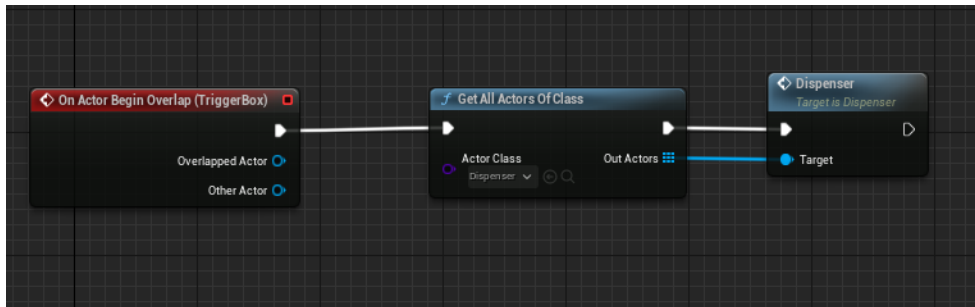
Simple „Dispenser“

i added and arm with a collision at the end to a blueprint, added a Rotating movement Component. then i set up a custom event that will be triggered by an independent trigger somewhere in the level. Because auf the flip flop the rotation starts at the first overlap and stops at the next one and again and again.



Simple „Dispenser“

I added a „Box Trigger“ to the Level and in the Level Blueprint made the following setup.



If the Trigger gets and Overlap, it gets the „Dispenser Arm BP“ and starts (or stops) the Event. To start the Event i just placed a „GrabCube“ above the Trigger that falls down as soon as the level is starting. To stop it i set up another, slowly rotating arm that stops (and starts) it after a long time.

