Game Interaction

This game is about managing resources and evolving your valley using them. You can move arround using `WASD` and explore the region. Using numbers `1,2,3` you can switch between your three tools and use them by `left-clicking` on the appropiate resource.

By pressing the `I` on your keyboard you can open/close the Inventory to see the resources you've collected. Be careful because the size is limited.

Now, you can start doing constructions by `right-clicking` on an empty cell. You can plant vegetables to earn food and if you have enough wood and rock you can start building buildings.

Evaluation

Particle Effects

- **Chopping rocks:** there is a mesh particle effect which spawns a great amount of stones with physics that collides with the floor
- **Building Buildings:** There is some kind of dust effect with emits steam particles over the building under construction.

Lights

- **Sun:** animating the sun on the level blueprint emulates a night/day system from which the other lights in the scene decides whether or not to bright
- **Torch:** The main character has a socket on its left hand with a torch attached. Even though, to avoid too much movement when walking, the spotlight source is attached to the hip and offsetted a little bit higher to avoid large shadows from the grass. This lights starts to emit light when the night has fallen.
- **Building windows:** As well as the torch, the windows of the buildings bright during the night. This is aimed by using an emissive material and using a glowing effect with a threshold in the Post Processing Volume.

Sounds

- **Ambient sound:** forest ambient sound during the whole gameplay.
- **Music:** loopable melody during the whole gameplay.
- **Building:** played during the building construction
- **Next Level:** played when the player achieves enough experience and the level increase. It also comes with a screen pulse post processing effect.
- **Pickup:** played when you collect something to your inventory
- **Plant:** played when you plant a vegetable.
- **Rocks:** played when you chop a rock, emulating the sound of the stone particle system
- **Steps:** played when the player is moving around the scene
- **Tree-falling:** played during the tree cut-down.

Terrain

The game happens inside a valley delimited by high mountains and a sea. Al the stepable ground is a flat surface part of the terrain.

All the trees, rocks and grass are placed using the foliage tool which spawned only three type of actors: tree, rock and grass.

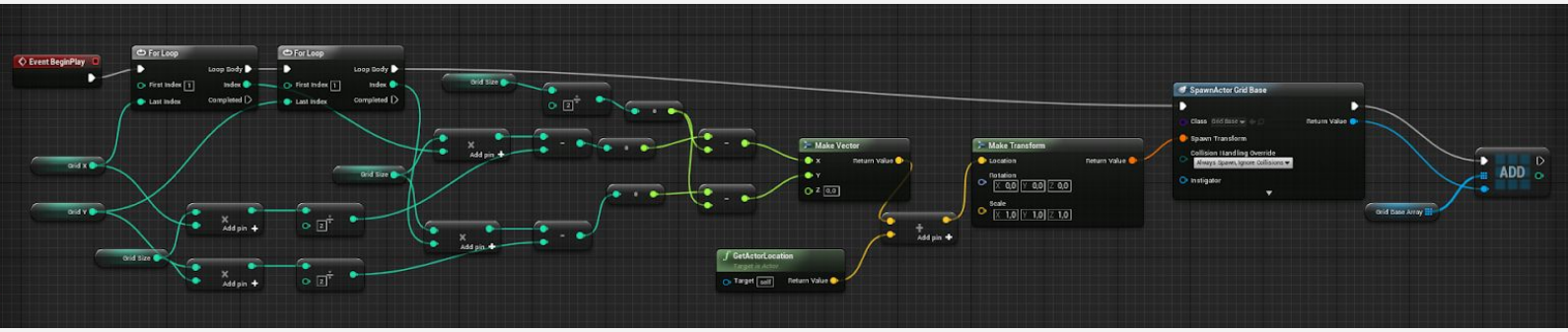
Physics

As it was said before, one of the elements of the game that uses physics are the rock particles which have gravity and collide with the terrain. Another element in which physics are important are trees. When trees are cutdown the physics are enabled to make them fall and their collisions are disabled but the floor, restricting its movement on the z axis.

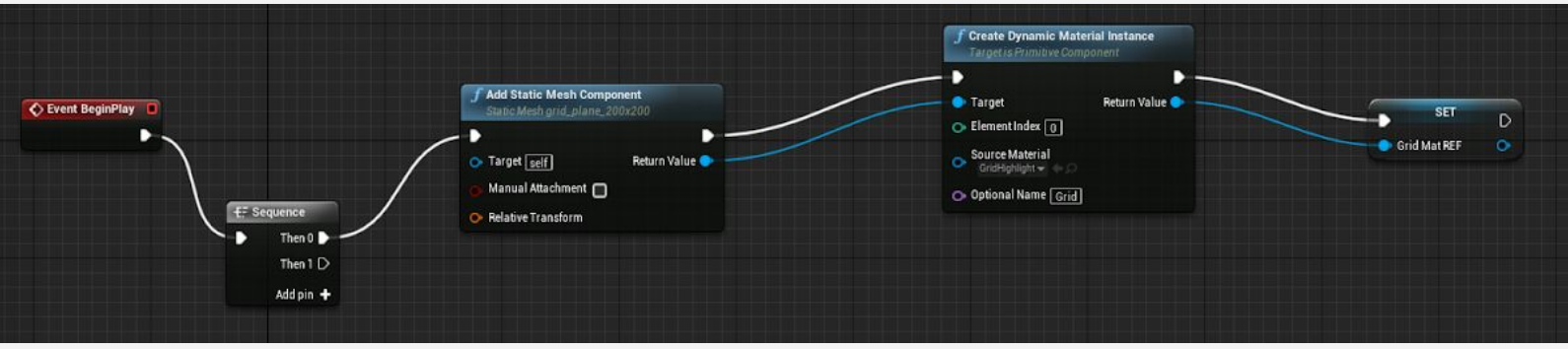
Blueprints review

Grid Implementation

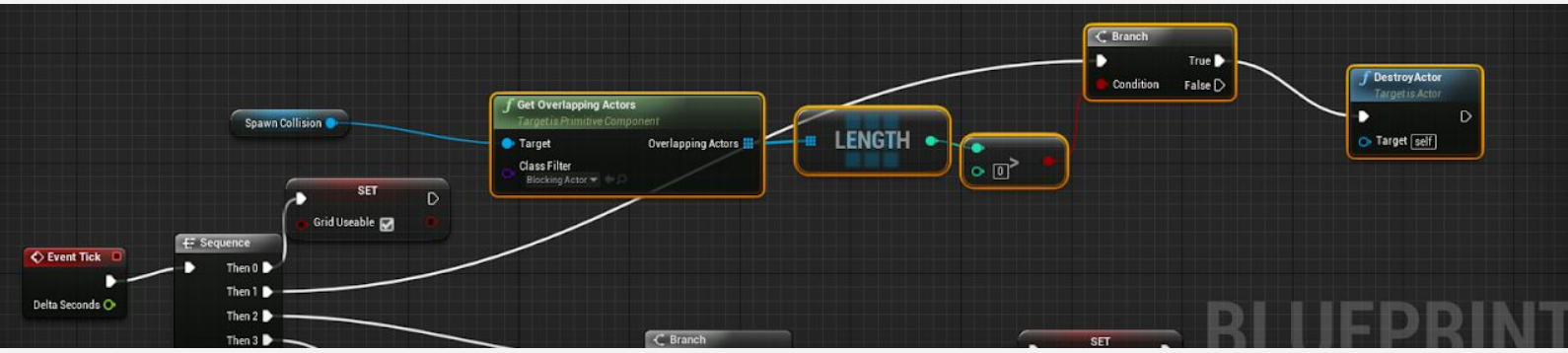
The game is grid based, so to help level designers decide the extensions or wheter or not has no be a cell there's an automatic creation of the grid. On the scene you have to place de "Grid Setup" blueprint in the floor and that will be the center of the grid.



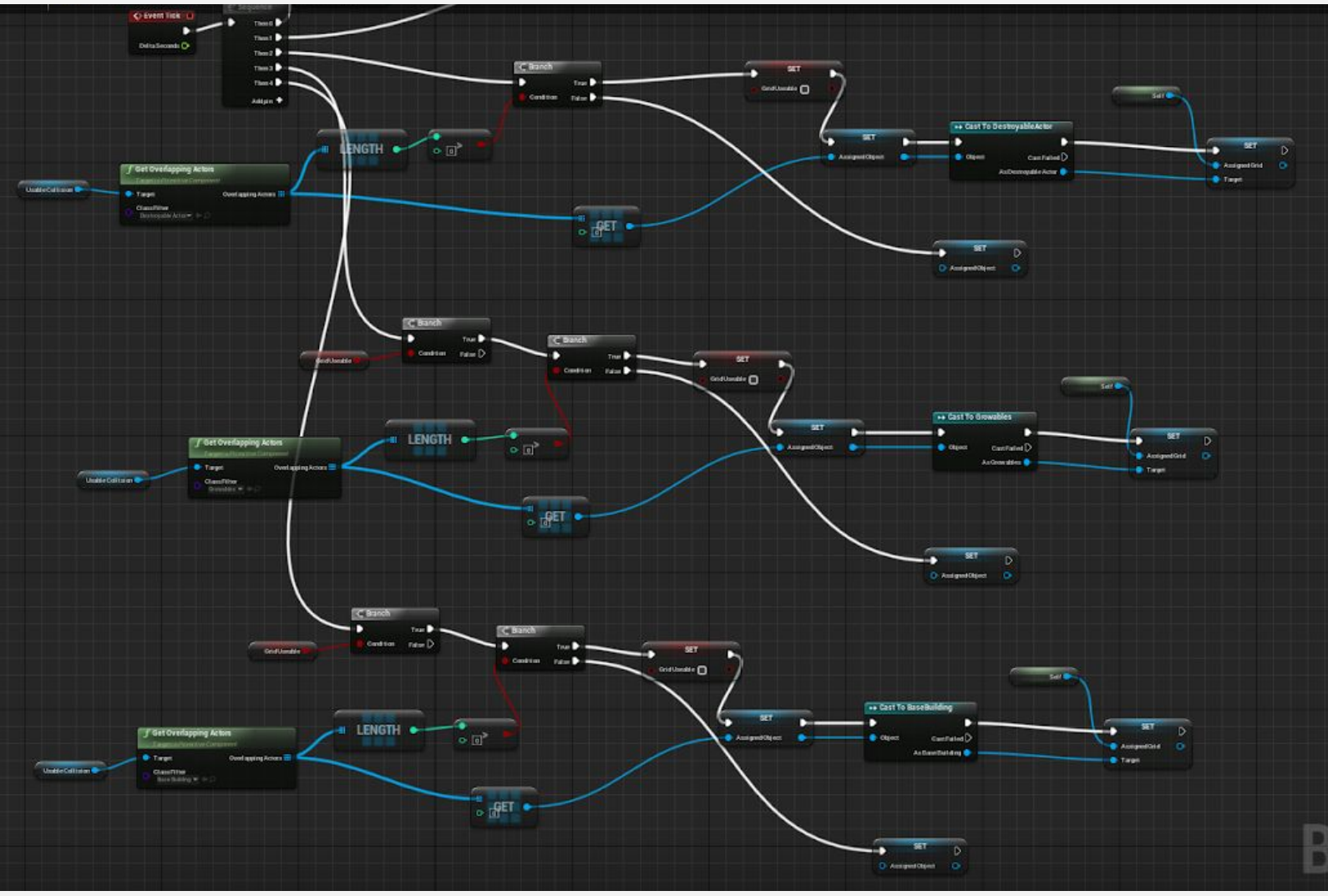
As we see the grid cells are instances of a "Grid Base" blueprint. At the start this blueprint creates a plane which will be used to highlight it and depend on its color we'll know if the cell is free or its occupied by an object.



Then, we can see that this blueprint has two colliders. The "Spawn Collision" collider is used to know if the cell instanciated in that position can stay there of it has to be destroyed. This is very easy for the level designer to implement because we have another actor called "Blocking Actor" that we can place all over the scene to tell de cells that they cannot be spawned on that position. On the case of our game this us used on the mountains and the sea that surrounds the scene.



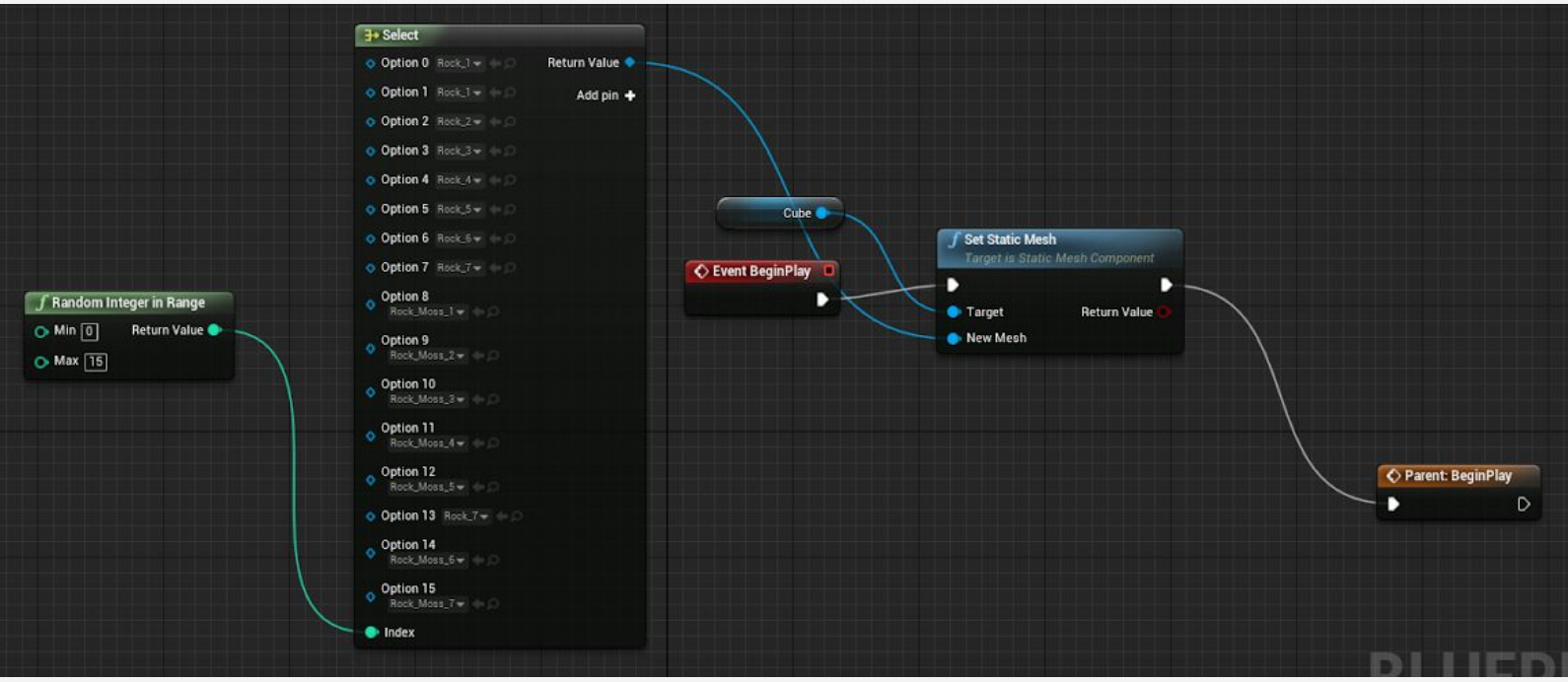
The other collider is used to detect which actor is upon that cell: DestroyableActor like rocks and trees, Growables for the vegetables or a Building.



Destroyables

Using the parent class DestroyableActor it has been created to childs: rocks and trees.

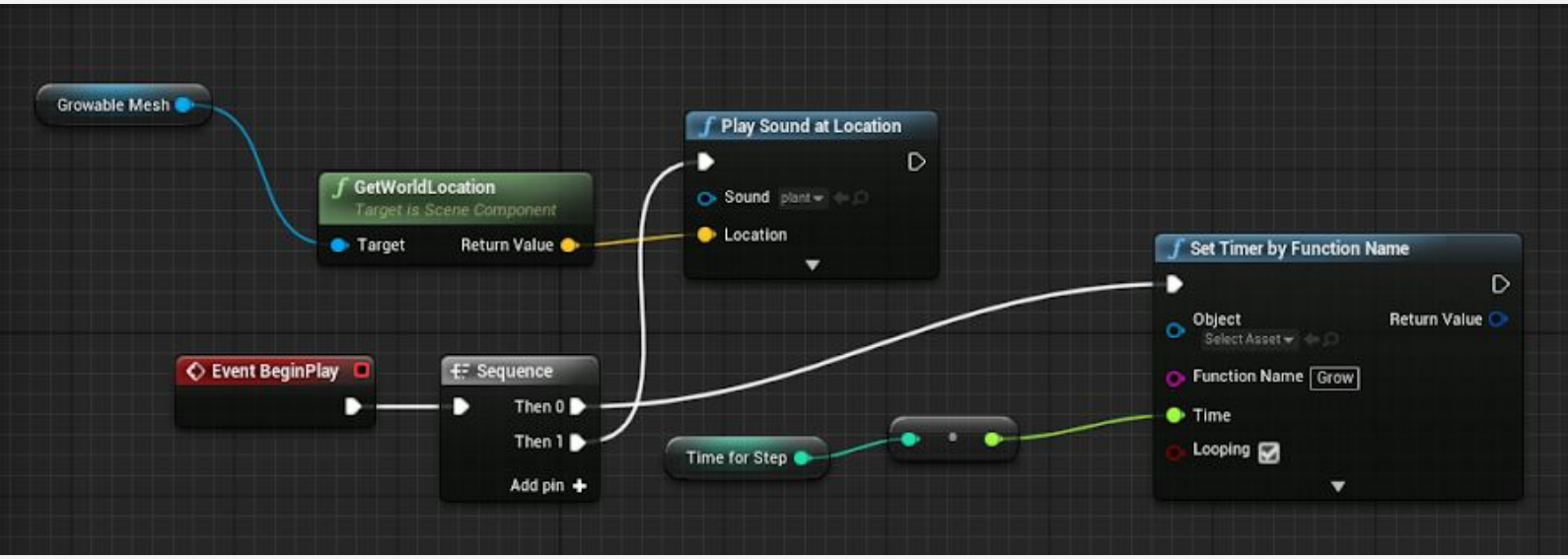
This classes are quite passive because the actions that we made on them are controlled by the playerController. The parent only set the grid to usable after the actor is destroyed. The child classes are more interesting in terms of level designing. As we've seen on the foliage we only have tree two types of destroyable actor instantiated on the scene but when be play we see that this actors change randomly their mesh every time. This is made on this blueprint. So the level designer only has to spawn trees wherever he wants and then tell the blueprints which meshes he wants to use.



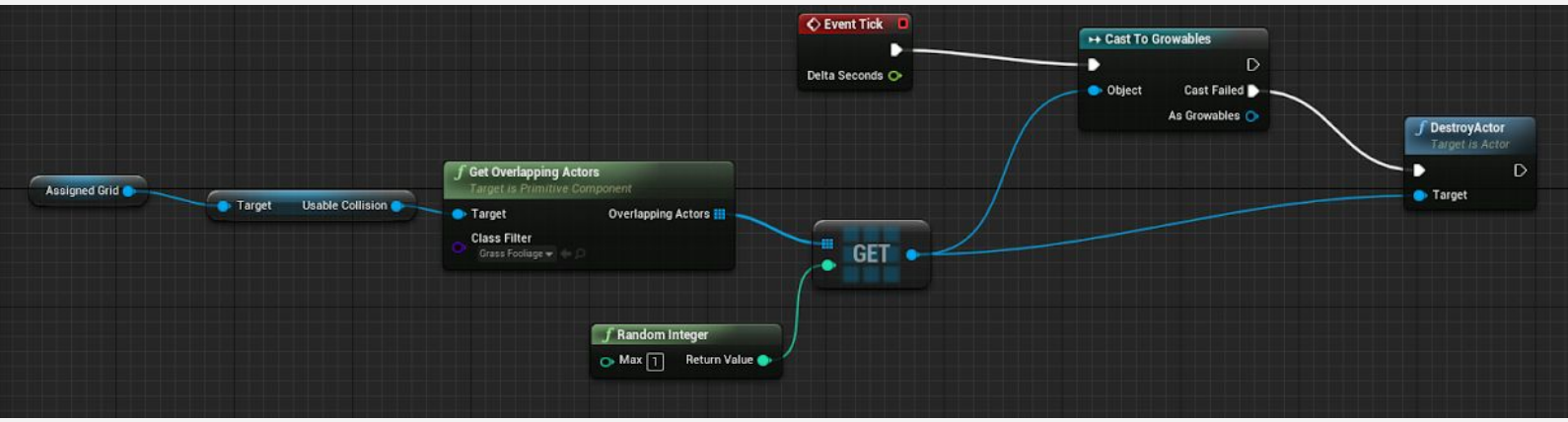
Growables

Using the same parent child structure we have a class called Growables that is the responsible to plant and grow the vegetables.

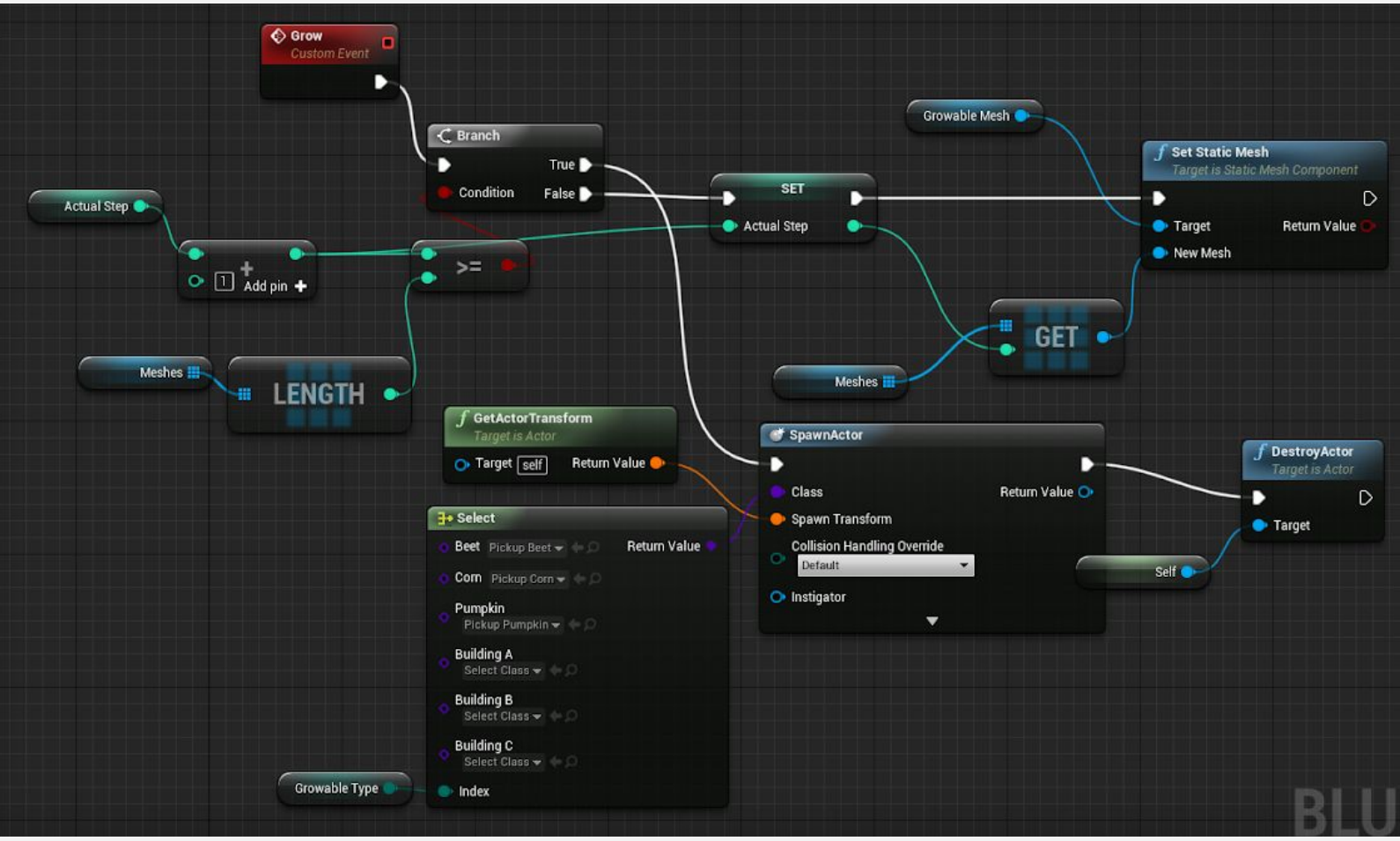
When one of this blueprints it's spawned it plays a sound and starts a timer that will call the "Grow" function ever X seconds:



To make it more realistic and visual it supressed the grass foliage from the cell that is spawned gradually



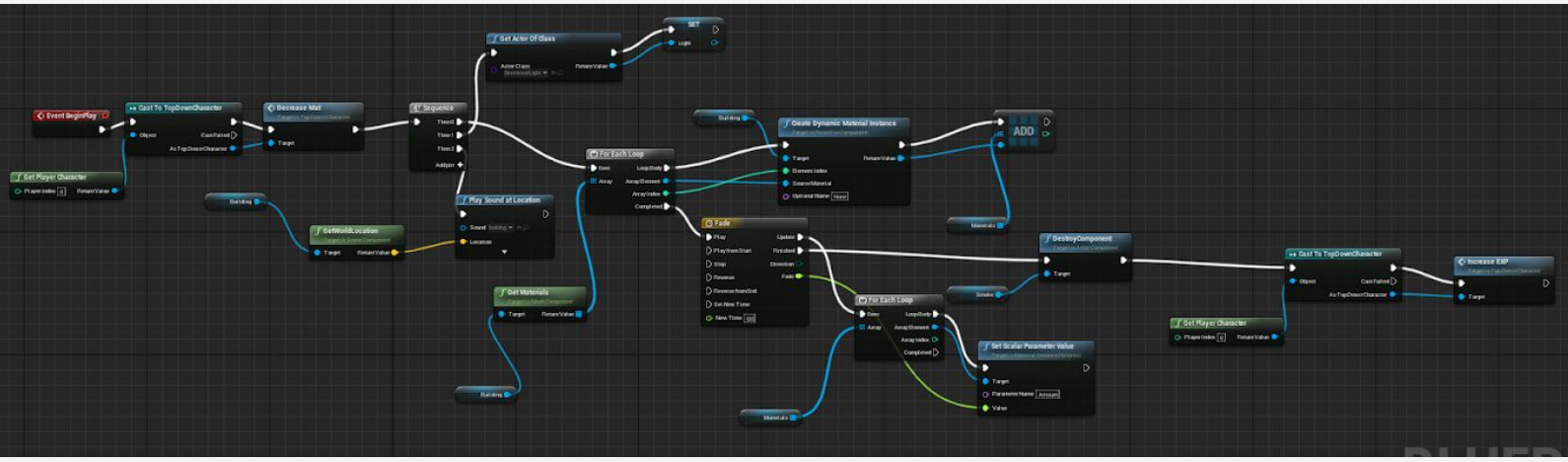
Every time that the Grow function is called it updates the mesh for the vegetable a step more grown until it reaches the last one, that it deletes itself and it spawns the associated “Pickable actor” on its location

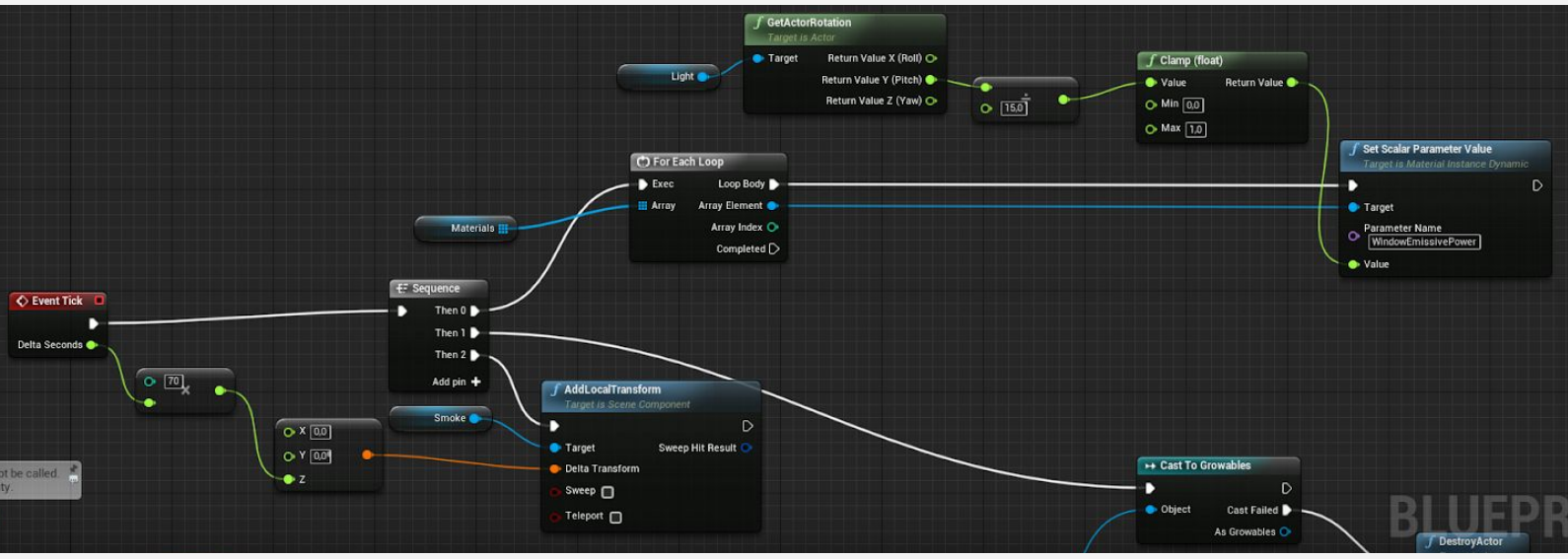


Buildings

In this case the also have a parent class and three child instances for having three different buildings.

When a building it's spawned it makes the player decrease the inventory of wood and rocks and gets a reference to the sun to change the material of the windows amb made them glow at night. After that, it plays a sound and it starts a building effect which that has been made using a shader and a particle emitter. When its done it increases player's experience. As well as the growables it also suppresses the grass on the ground.

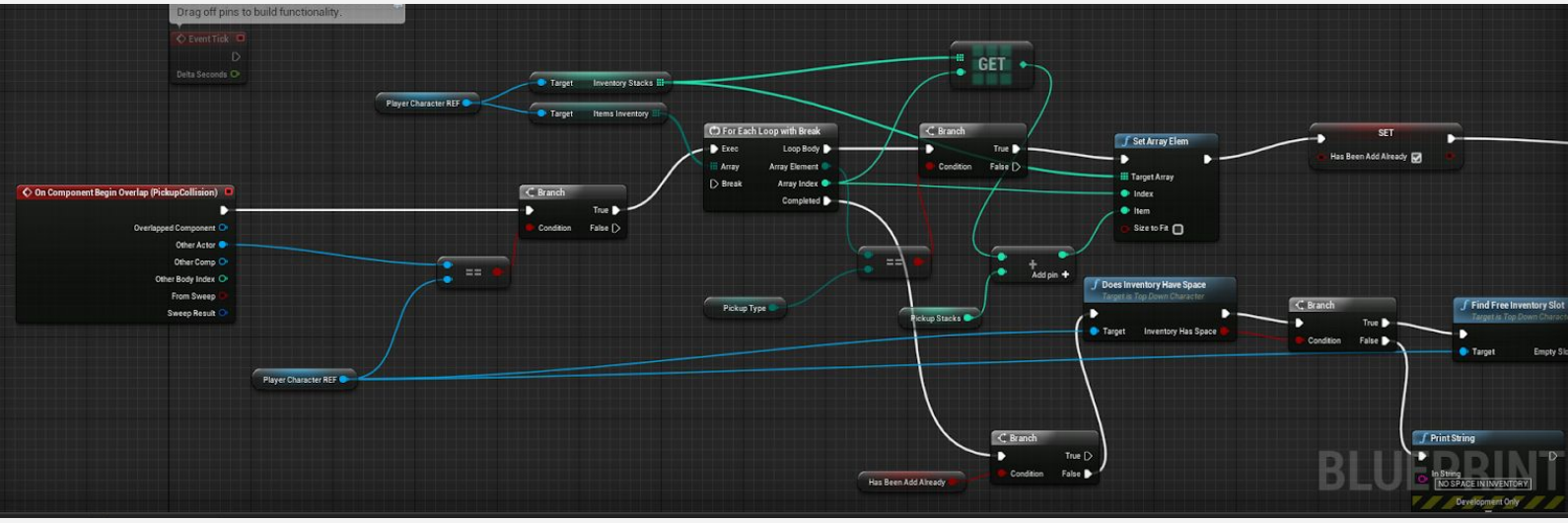




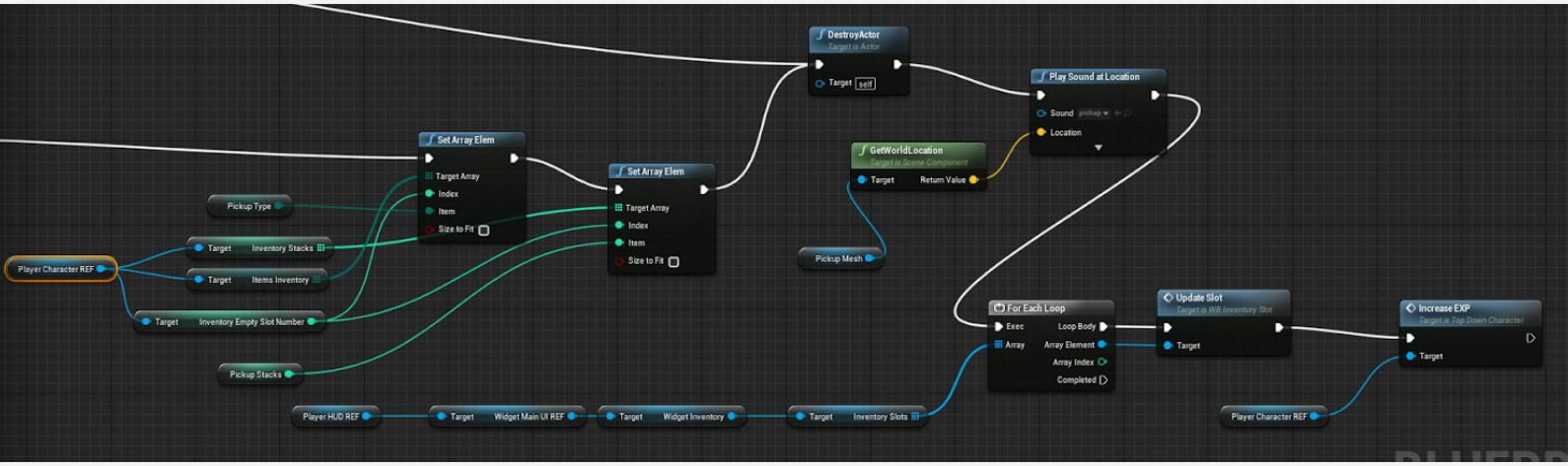
Pickables

This blueprint only have code on the OnComponentBeginOverlap and its quite large so let's split it in two halves.

The first one its the responsible to check if the player already have that pickable in its inventory and in that case it only adds more of that. If not, it uses two functions that will be explained later in the document to check if we have any free space in the inventory



If the player does have space it simply resets the inventory array, the widget, plays a sound and increases player experience

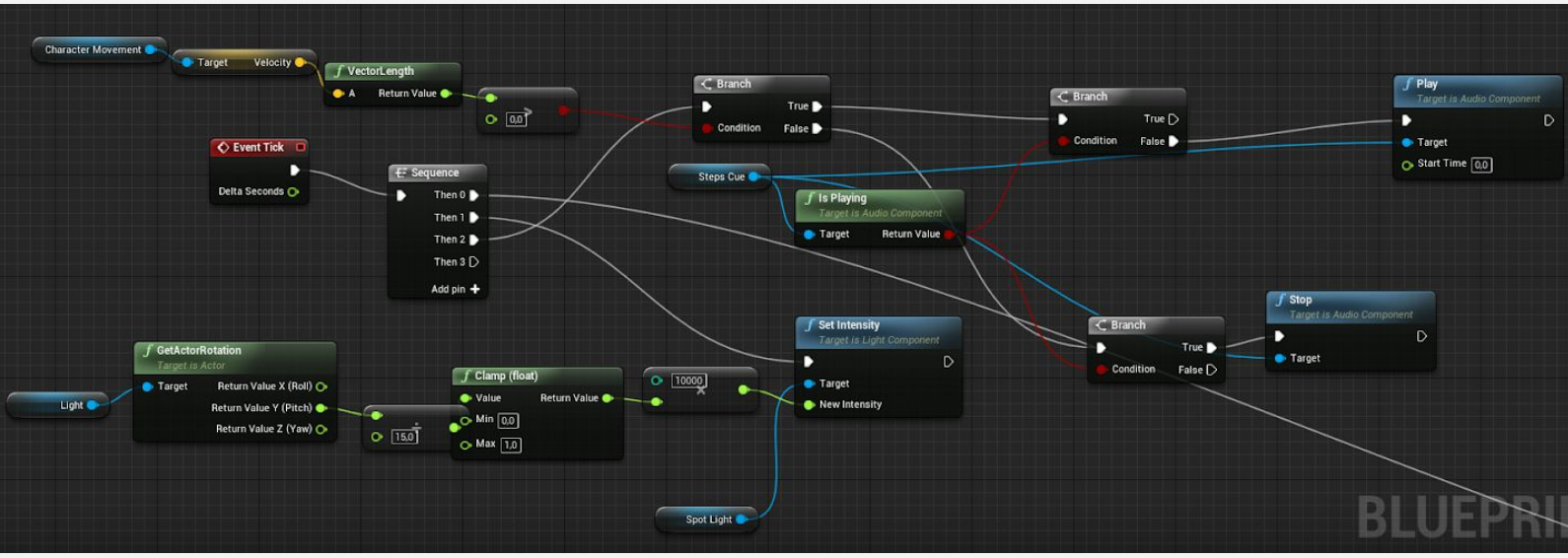


The only thing that changes between siblings is the mesh, the type and the quantity rewarded.

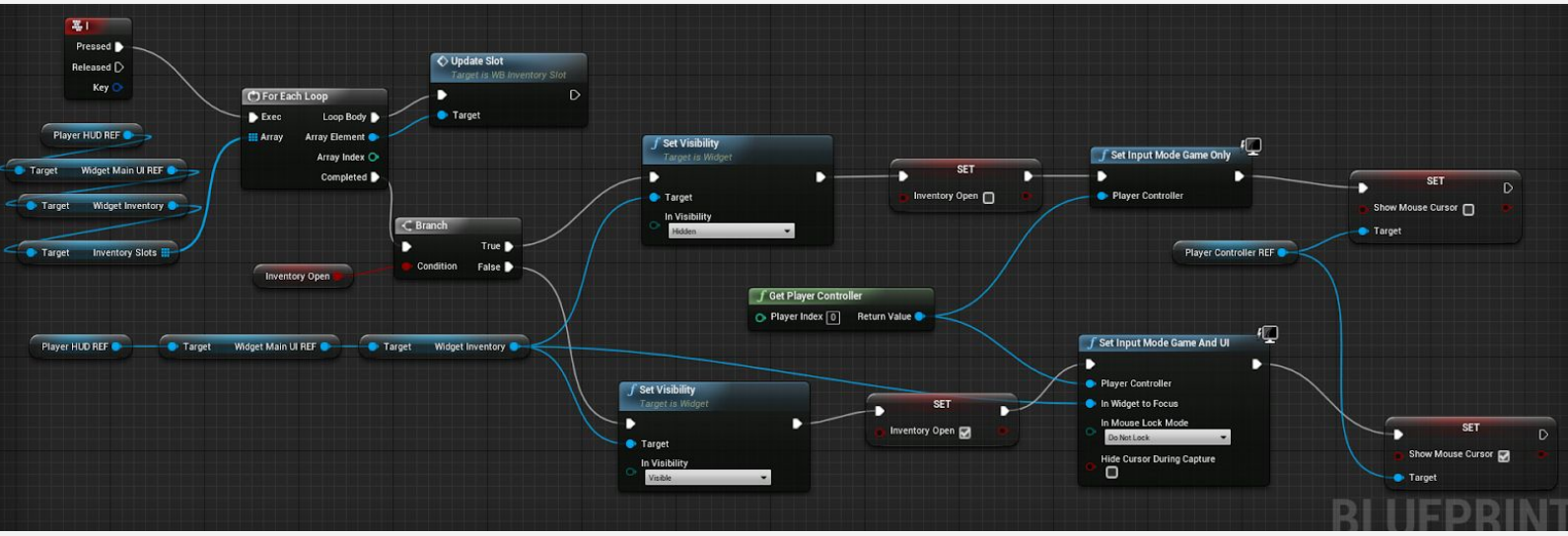
Character

This character has been created modifying the TopDownCharacter and it has a different mesh which has been retargeted form the 3rd person animation blueprint. It also has 3 new sockets for the tool, torck, and lightsource and also an audio component on its foot to play the steps sound.

On the event tick we can see that we control the sound of the steps and the intensity of the light depending of the sun rotation

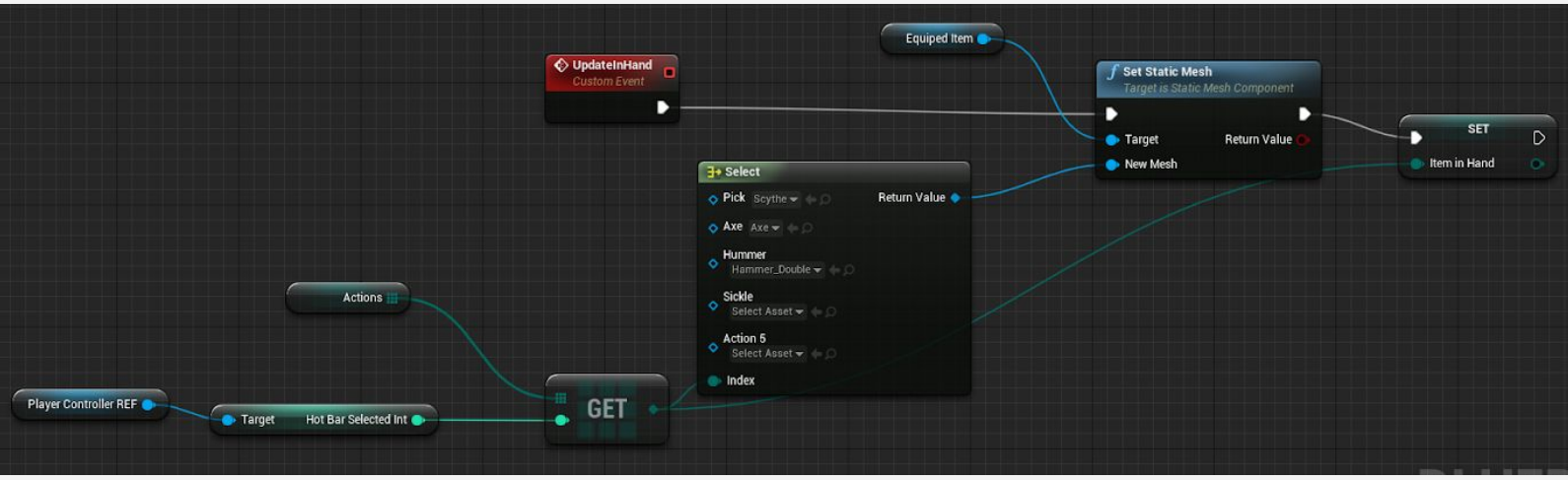


When the player presses the I key the character opens the inventory widget and toggle the game input to make the cursor interact or not with the UI

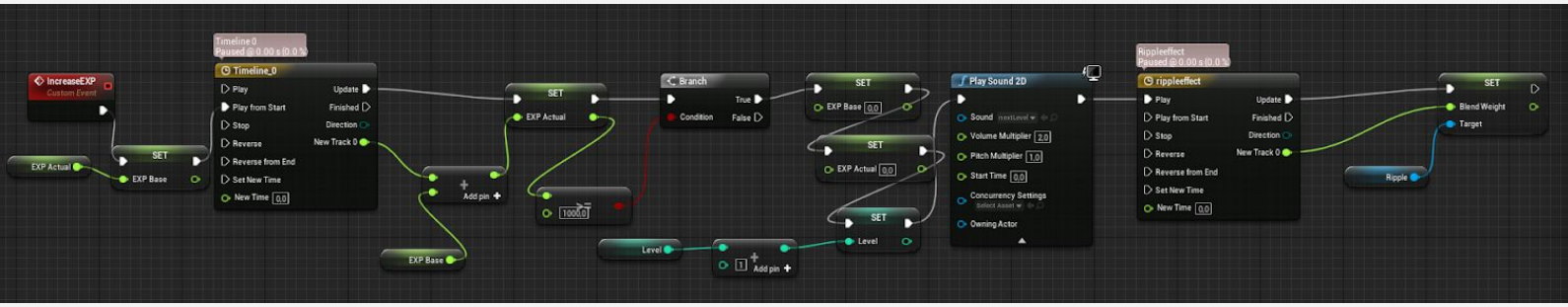


Then we can see that there are 3 custom events

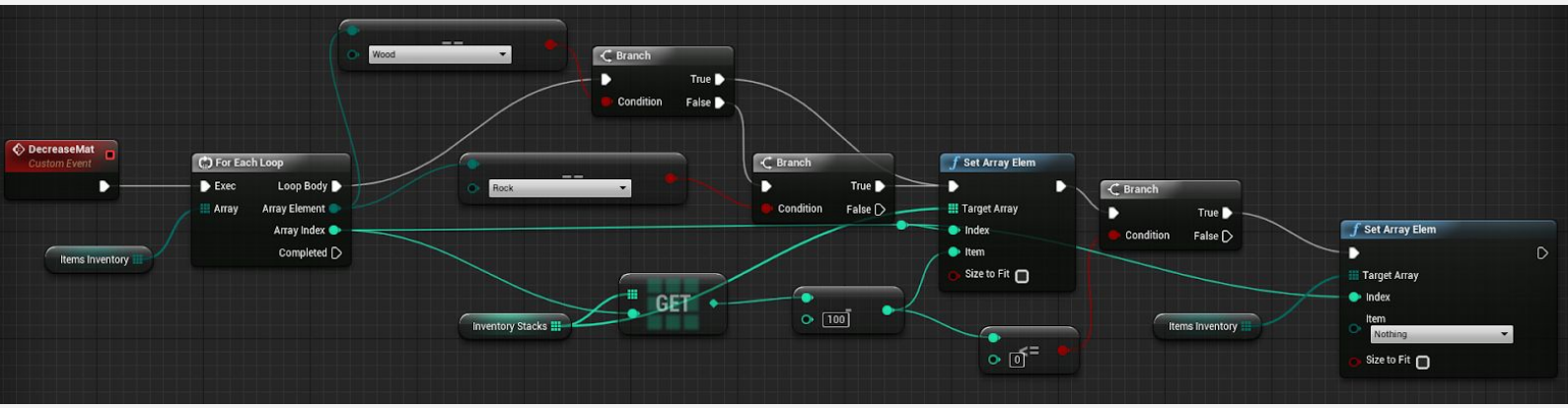
The first one is the Update in hand which is the responsible to set the tool that the player has choose.



The second one is the “Increase experience” which is the responsible to control and increase the experience smoothly and the levels and play the ripple effect when passing a level.

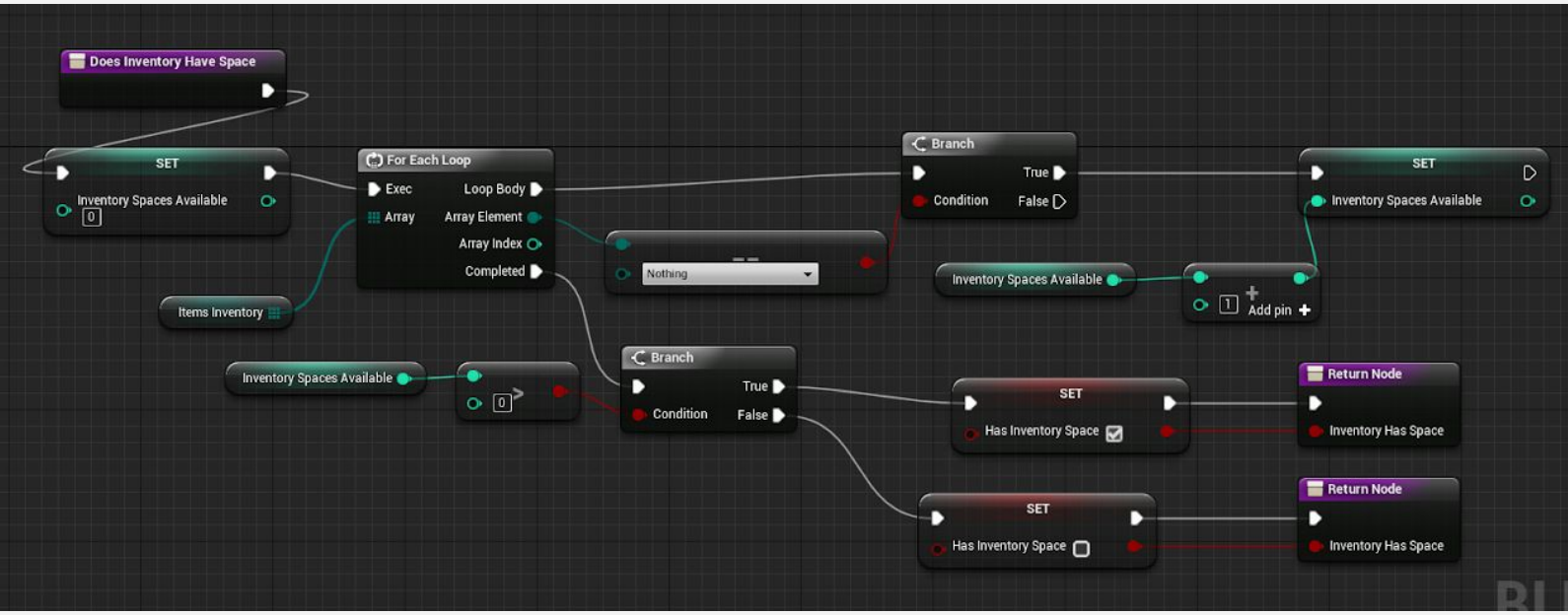


The last one is the “Decrease Mat” event which is the responsible to subtract wood and rock from the inventory after a building is built.

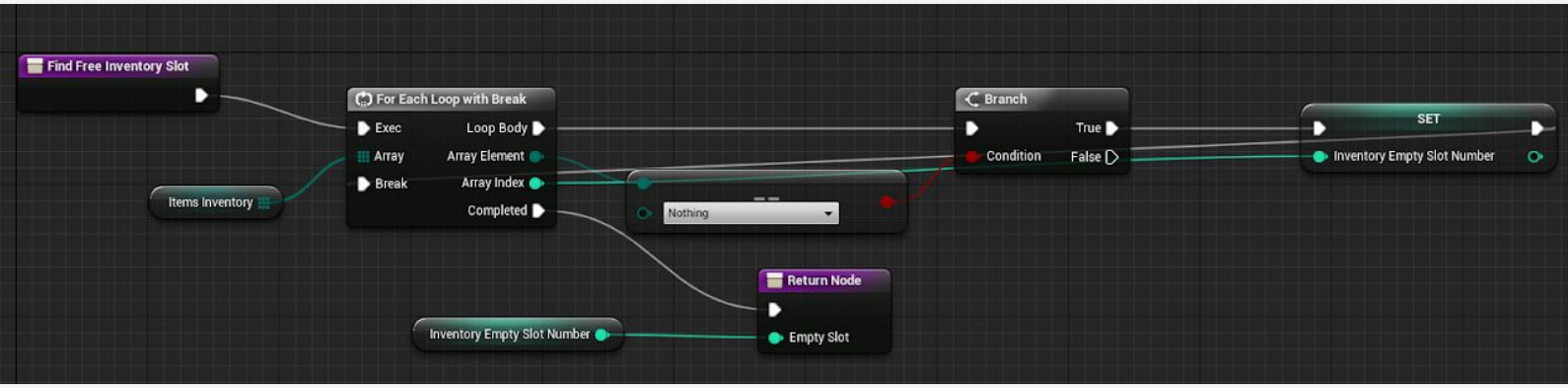


Then we have two useful function to know about the inventory space.

The first one is the “HasInventorySpace” that like its name says it returns true or false whether the inventory has a free space or not.



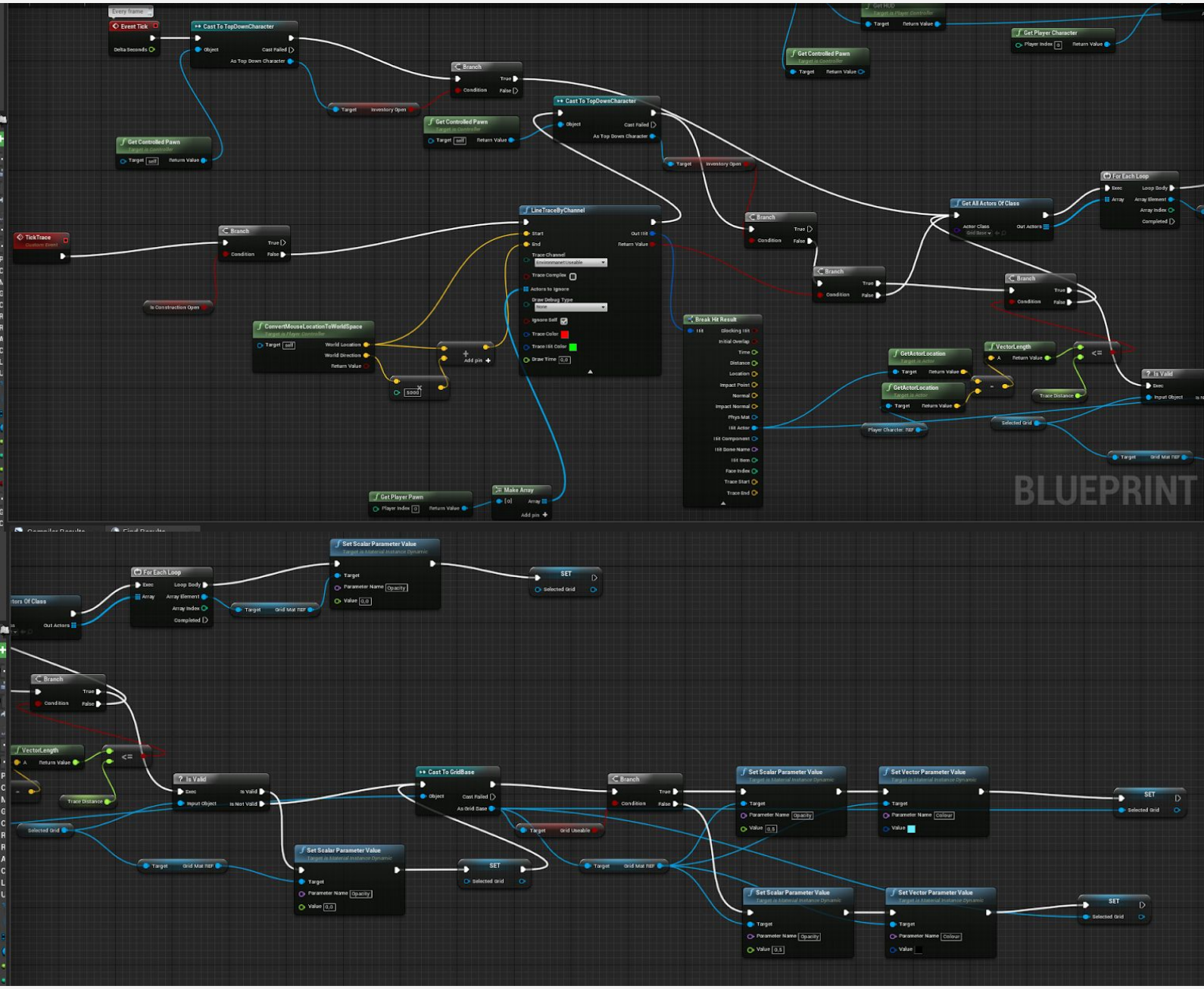
If the first one has returned True then we can call the second one to get the first free space in the inventory:



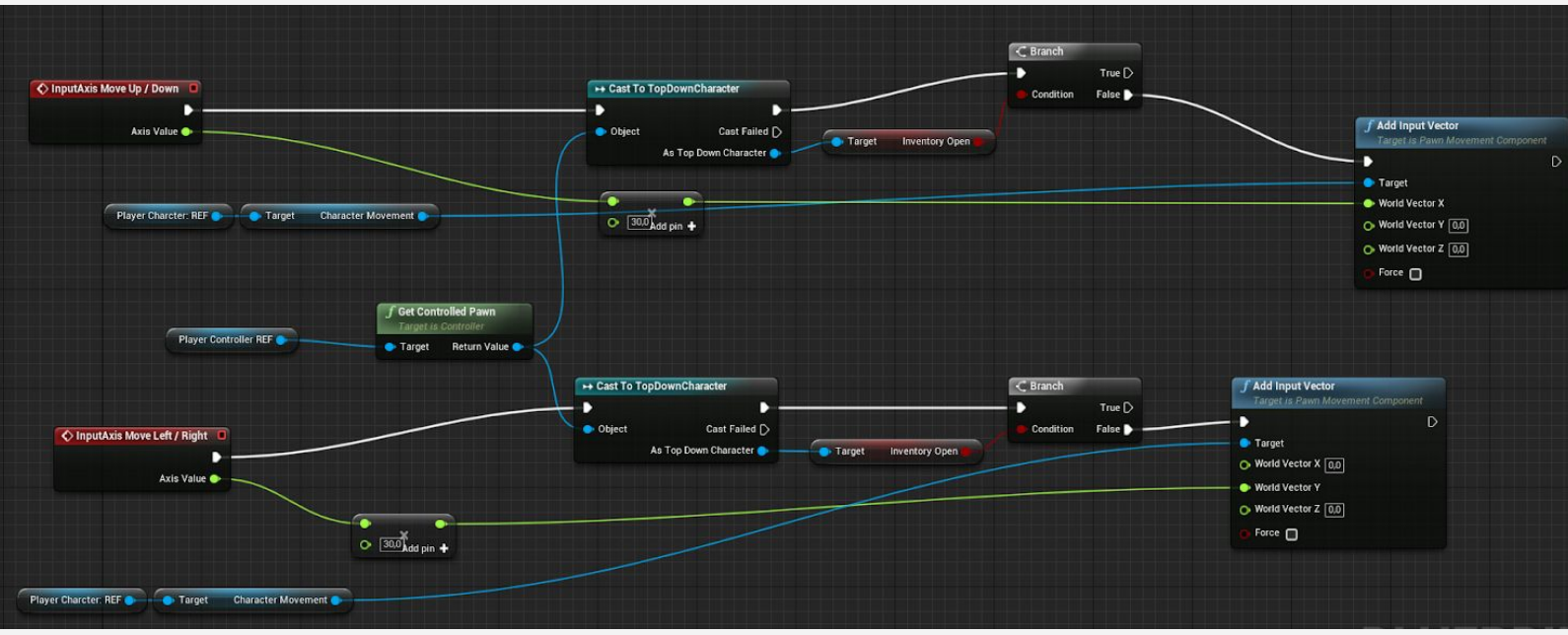
Player Controller

This is the largest blueprint in the project which is the one that basically controls the player input.

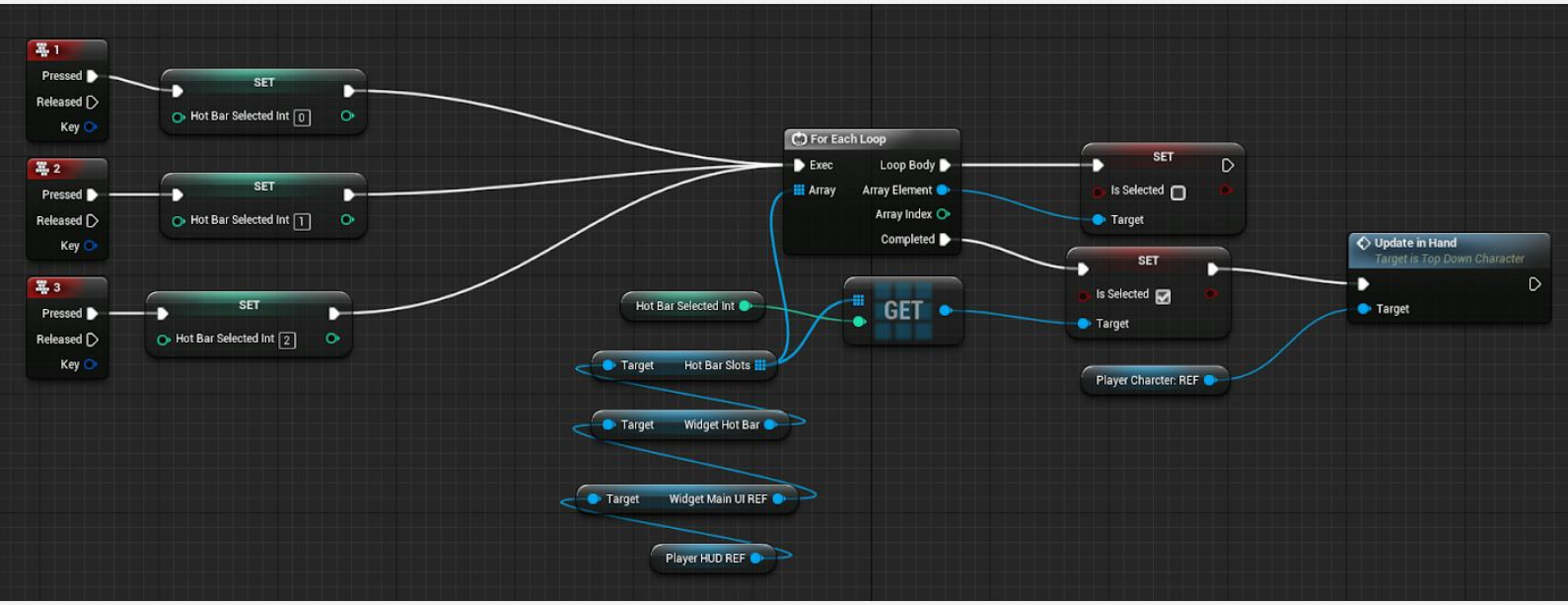
Let's start talking about the grid selection. The workflow it's quite easy, we create a raycast from the mouse position and we detect if it has collided with a Cell and looking the the cell is close enough or not to be selected. Always checking before if the inventory is closed, otherwised the raycast is not done. After that what the blueprint does is un-highlight all the cells and highlight the selected cell and depending if its usable or not the color will be one or another. For performance issues you can see that the raycast is not done every frame but every 50ms using a timer that calls the event TickTrace



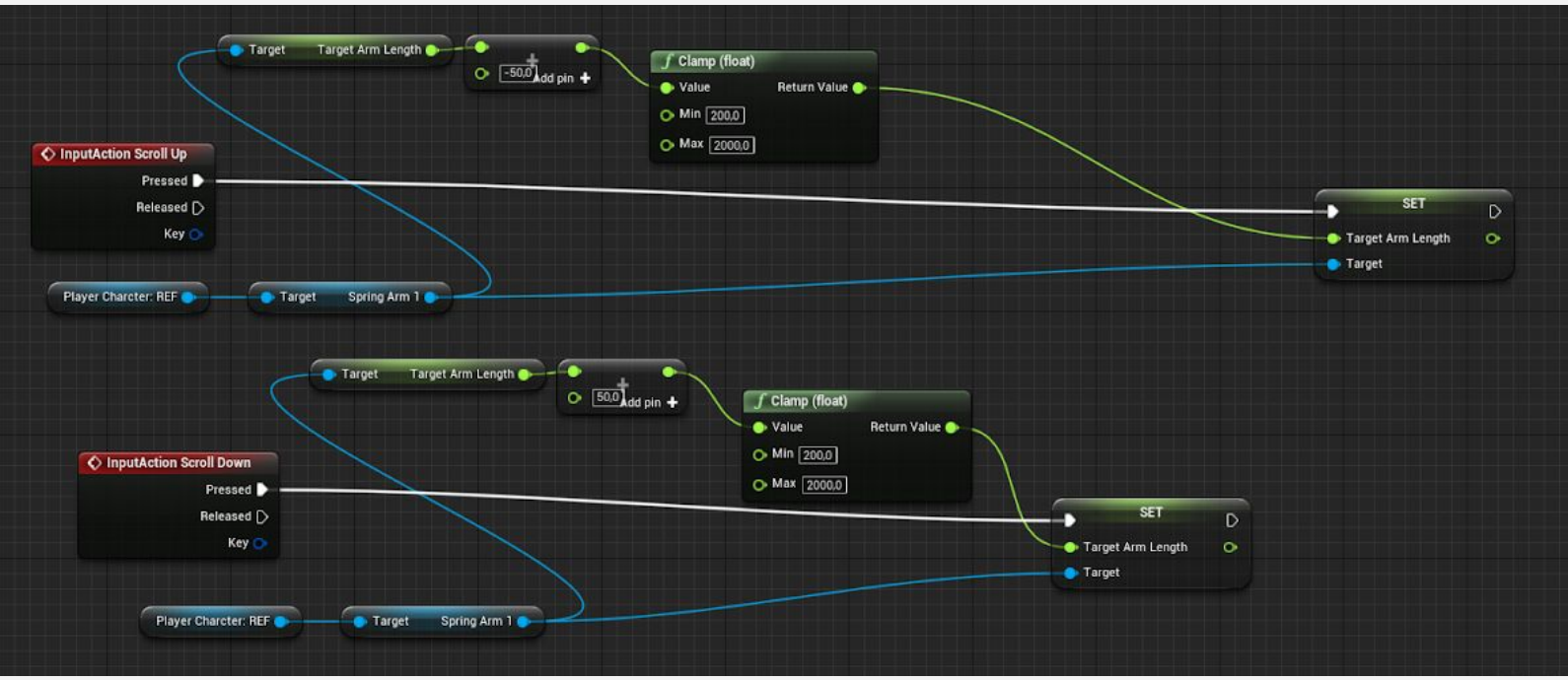
Then we deleted the click-to-go from the Top-Down-Controller and it uses a simple WASD movement using the input axis set on the project settings



To change the tool selected we use key numbers and we can see that we reset the tool depending of the input and we call the event UpdateInHand Commented above in the document

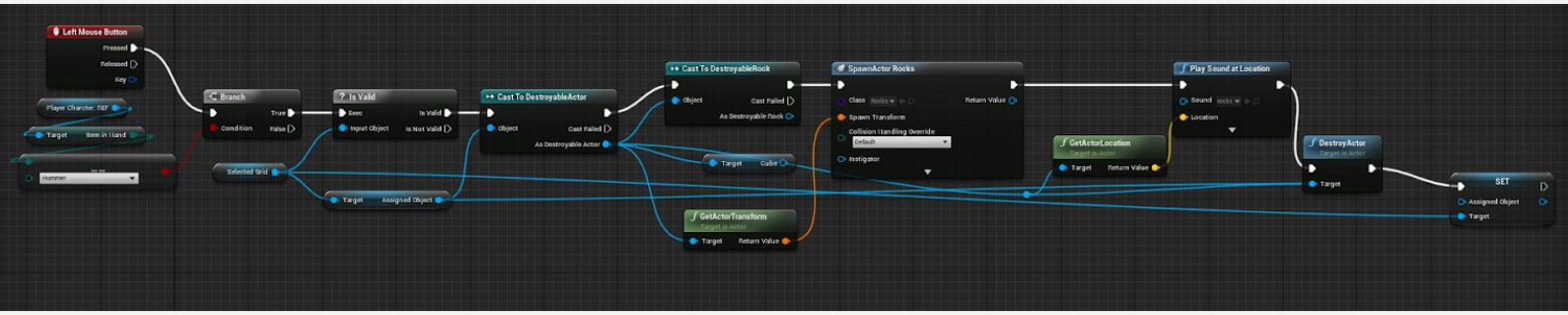


Another simple chunk of code is the one that allows the player to zoom in or zoom out the camera using the mouse wheel

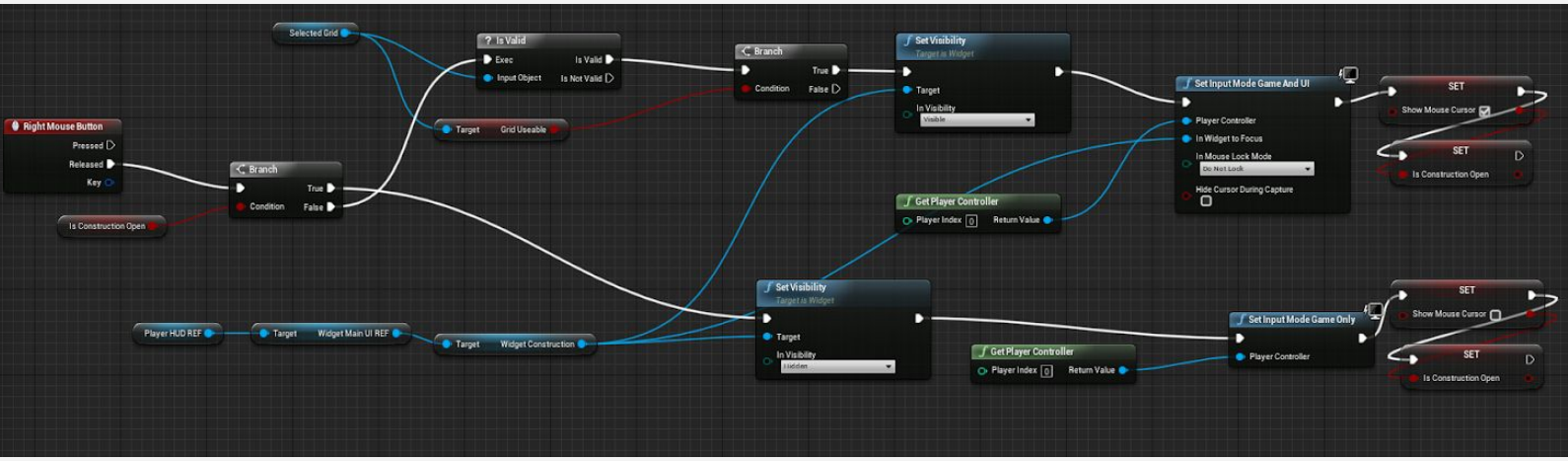


When the player presses the left button the blueprint checks which tool has equipped and if it can interact with the object that is in the selected grid.

In the case of the Hammer and the rocks in destroys the actor and spawns the pickable object with the particle system and the sound. In the case of the tree and the axe it enables de physics simulation and after a delay when the tree has fallen it destroys it and spawns the wood.



To open the construction panel the player has to right click on a empty cell and it will open



Construction Button

When the construction panel is open and you click on one of the six options it checks if you have enough materials to build it, spawns the object on the selected grid and closes the menu