INTRODUCTION TO UNREAL ENGINE PART 2

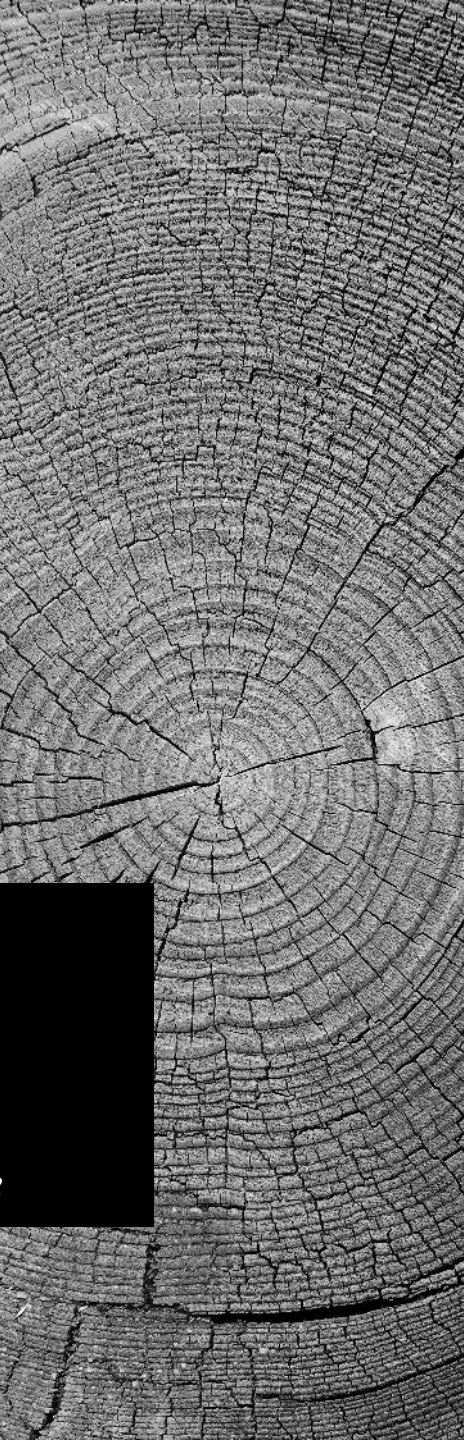
By:

Natasha Mangan

Course:

Game Engine Architecture

Unreal Engine 5.3.2



ABOUT THIS INTRODUCTION

It will be based on Unreal Engine version 5.3.2

It will include:

Blueprint implementation of last week's C++

Going deeper into some of the blueprints from last week too (still basic though ;D)

THE PROJECT

Use the same project as last week!

SETUP

- Use the same project as the C++ implementation.
  - Or download the zip file from the canvas and follow along from there. (this is last weeks ”mostly” blank project)
- Create a new map called ”game_bp_map”
- Set startup map to game_bp_map in project settings

CREATE SOME BLUEPRINTS

Right-click in the appropriate folder (name of folder), select Blueprint from the menu and then do the following:

- Create a blueprint Gamemode, ”BP_ThirdPersonBPOnlyGameMode”
- Create a blueprint PlayerController, ”BP_PlayerControllerBPOnly”
- Create a blueprint Character, ”BP_PlayerCharacterBPOnly”

GAME MODE CONTROLS

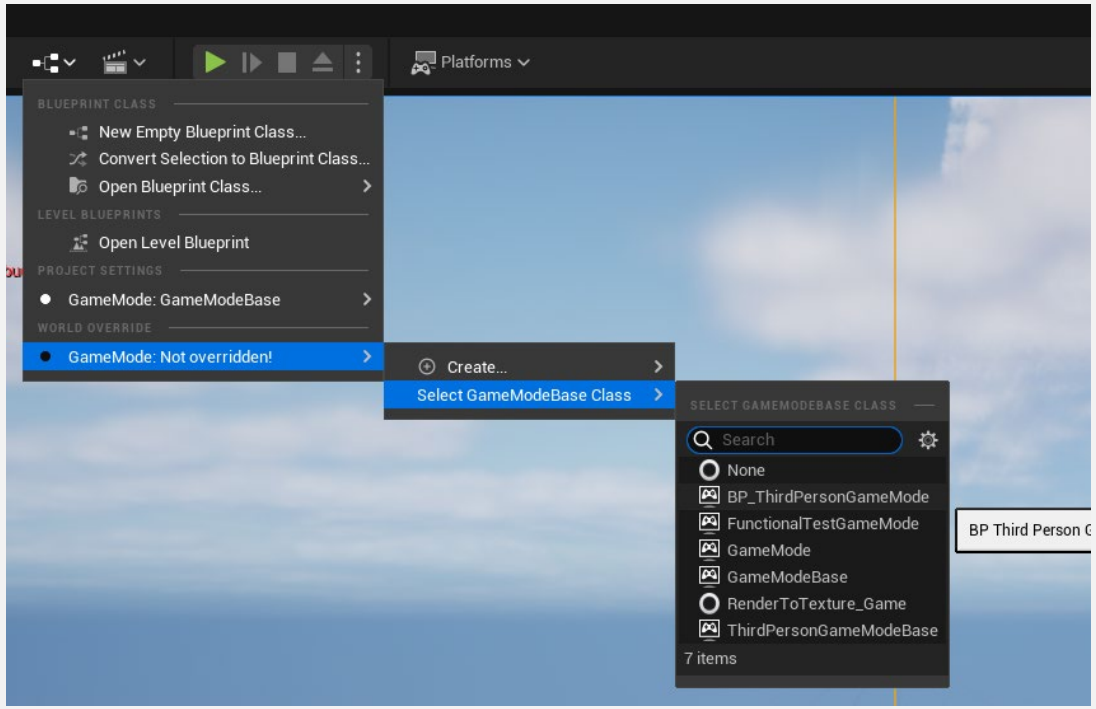
In the game mode BP we just created:

1. Set Default Pawn Class: BP_PlayerCharacterBPOnly.
2. Set Player Controller Class: BP_PlayerControllerBPOnly

Compile and Save!

SET THE LEVEL GAME MODE OVERRIDE

- Go to world override -> select gamemodeabase class -> BP_ThirdPersonBPOnlyGameMode



- Note this picture is from last week ;D
- Now our character spawns where the player start is (and facing direction)

MENU SYSTEM

HUD

Menu Screens

PLAYER CONTROLLER - SETUP HUD

In the player controller class create a custom function called ”Open HUD”:

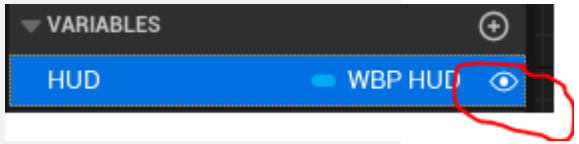
- Right-click in the Event Graph
- Select ”Create Custom Function” (type in ”cust” first in the search bar!)
- Call it Open HUD.

Side Note:

You can drag out a node and select ”Re-Route Node” to create a new anchor node in the line, this lets you neaten up blueprints!

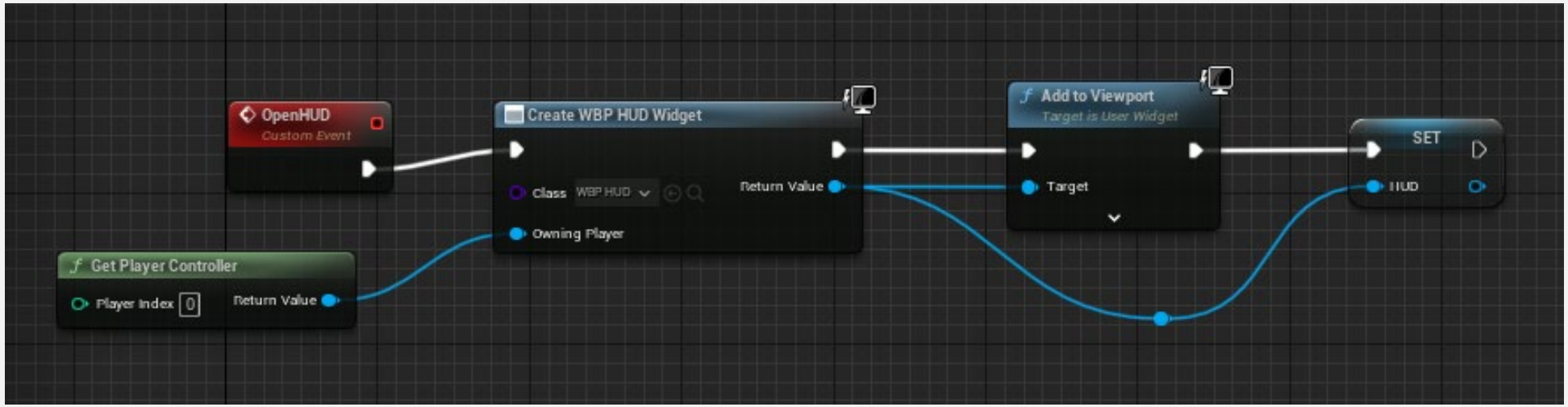
Drag out the following nodes:

1. ”Create widget” set class type to ”WBP_HUD”.
2. “Get the player controller“ and drag the node into the “create widget”.
3. Drag the node from “create widget” to “Set to Viewport“ (you may need to uncheck context sensitivity to find it)
4. Drag the node from “create widget”, in the menu select “Promote to Variable”. (turn context sensitivity back on, then it’s at the top)
5. Name the Variable HUD.
6. In the details panel of that Variable select “Instance Editable” or click The little eye icon
7. Compile and Save



By creating the variable, we have access to it and can call it / manipulate it from other blueprints.

PLAYER CONTROLLER BLUEPRINT

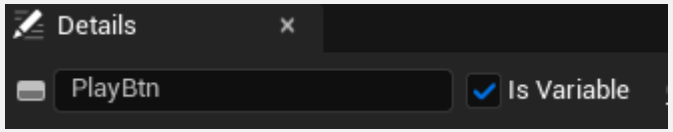


IN OUR WBP_MENU

Create a button and anchor it to the centre of the canvas.

Select the button:

- Name the button ”PlayBtn”.
- Add text component to the button as a child and set the text to ”Play” (also toggle Size to content – under anchor settings)
- At the top right of the details panel, select IsVariable.



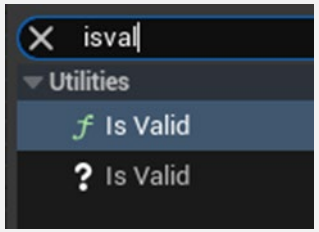
- In Graph mode, select the button and Scroll to the bottom of the details panel and select OnClick (plus sign)

From the OnClick Node:

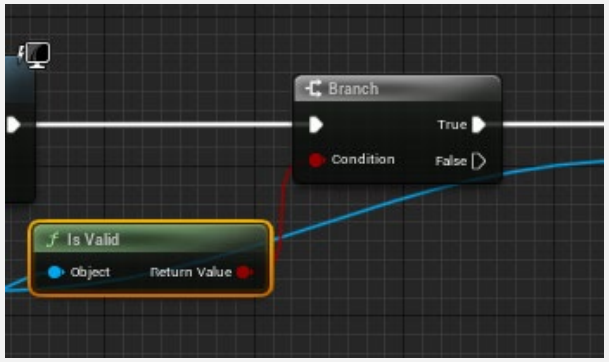
1. Get Player Controller
2. Drag node from Player controller to “Set the input mode to Game Only”.
3. Drag node from Player controller to “set show cursor” to false.
4. “Set Game Paused” to false.
5. Drag node from Player controller to “Cast to BP_PlayerControllerBPOnly”

IN OUR WBP_MENU

6. Drag node from the cast to “IsValid”. Choose the variable version noted by f



7. Drag out from IsValid to “Branch” (basically an If statement)

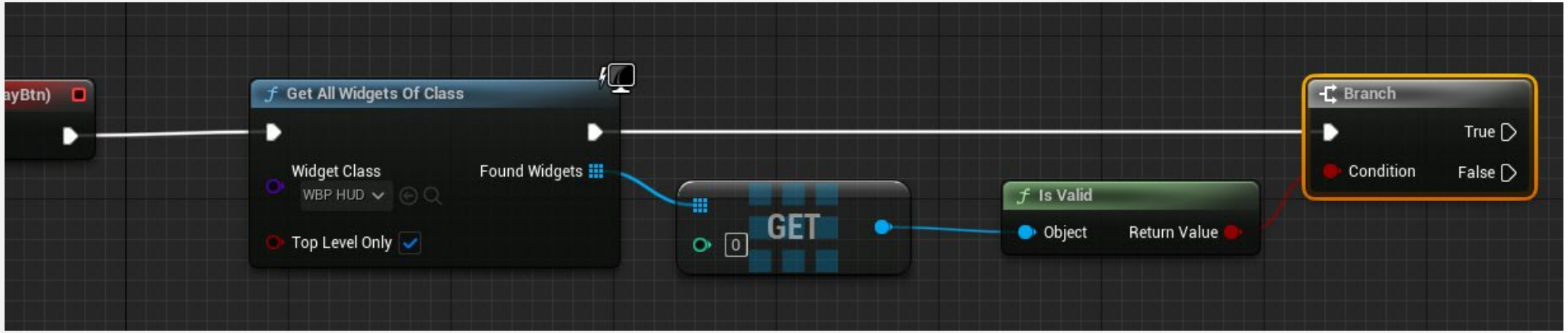


8. Drag node from Cast to “OpenHUD” (our custom function from earlier)
9. Drag out from True to “OpenHUD”
10. Drag out from False (& Cast Failed) to “Print String” (this is how we debug log to screen)
    7. Set message colour to Red
    8. Set the timer to 3 seconds
    9. Set message to “Failed to Cast to Player Controller”.
11. Drag out from “OpenHUD” to “Remove from Parent” (always have this last!)

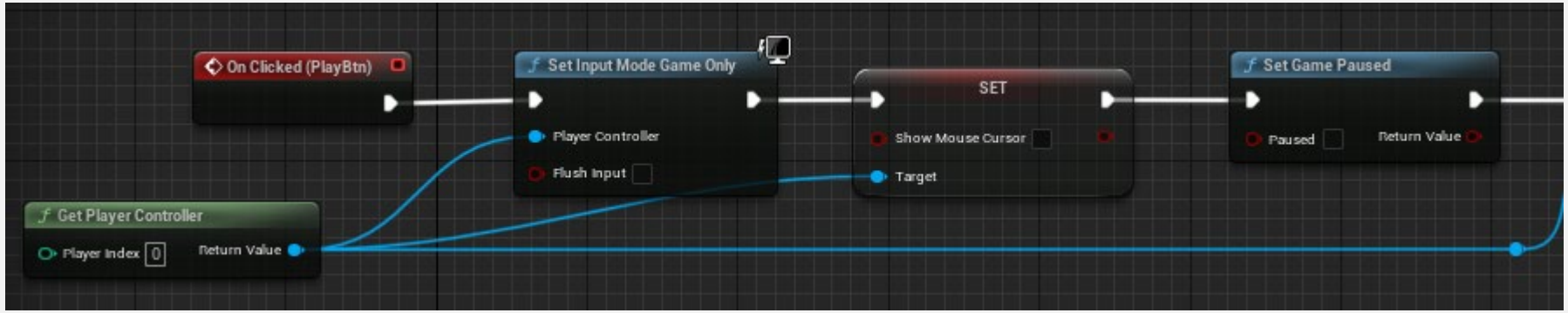
IN OUR WBP_MENU

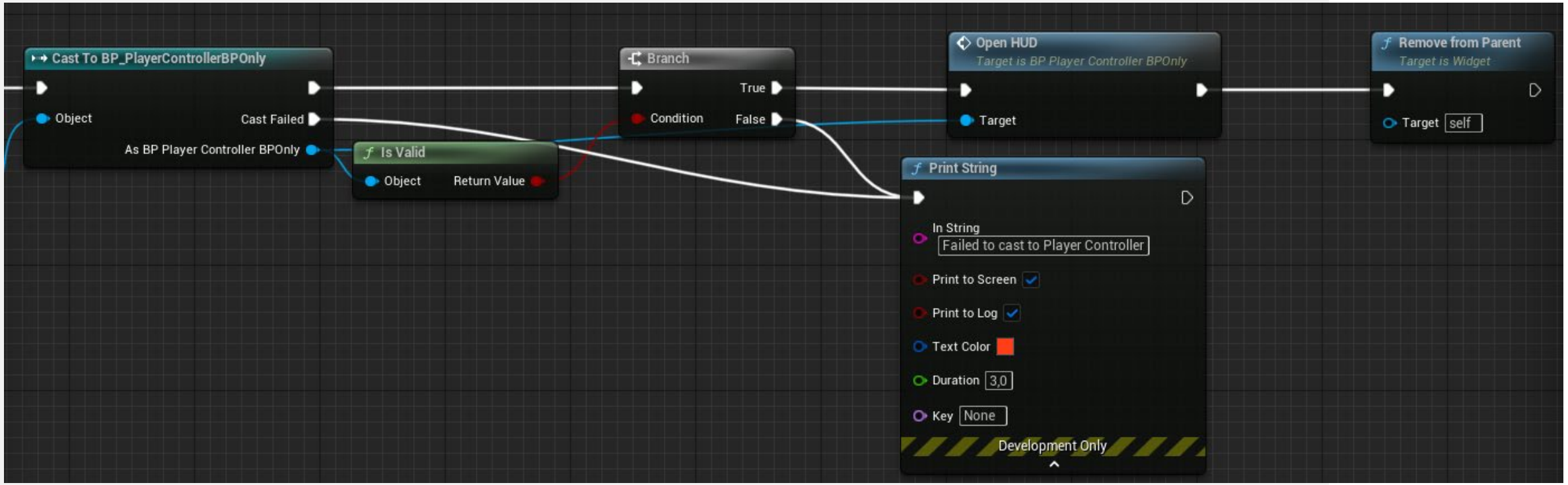
“Remove from Parent” :

- It removes current widget from the viewport.
- The viewport is an array of items.
- If you want to get something specific from the viewport you need to:
  - Get all widgets of class type WBP_HUD
  - Pull out from the array of found items
  - Get a copy (you can manually change the 0 on this node too (this is array element 0))
  - Check isValid



GRAPH OF BP_MENU





SETUP MENU

Make sure you are in the new map we made called, ”game_BP_map”

In the World Blueprint Settings, On Begin Play:

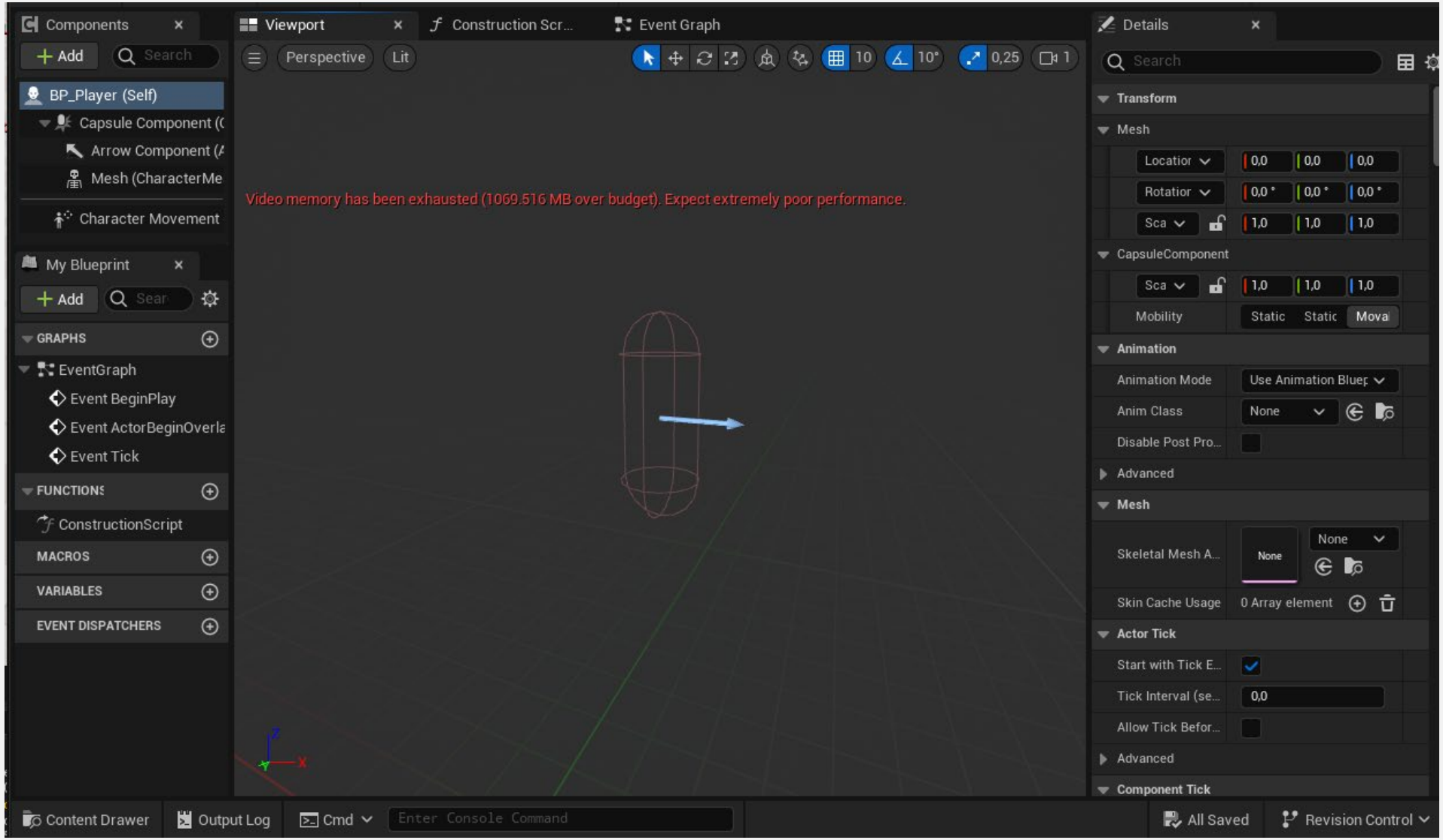
1. “Create Widget” set class type to “WBP_Menu”.
2. “Get the player controller“ and drag the node into the “create widget”.
3. Drag the node from “create widget” to “Set to Viewport“
4. “Set Game Paused” to true.
5. Drag node from Player controller to “Set the input mode to UI Only”.
6. Drag node from Player controller to “set show cursor” to true.
7. Compile and Save

PLAYABLE CHARACTERS

Third Person

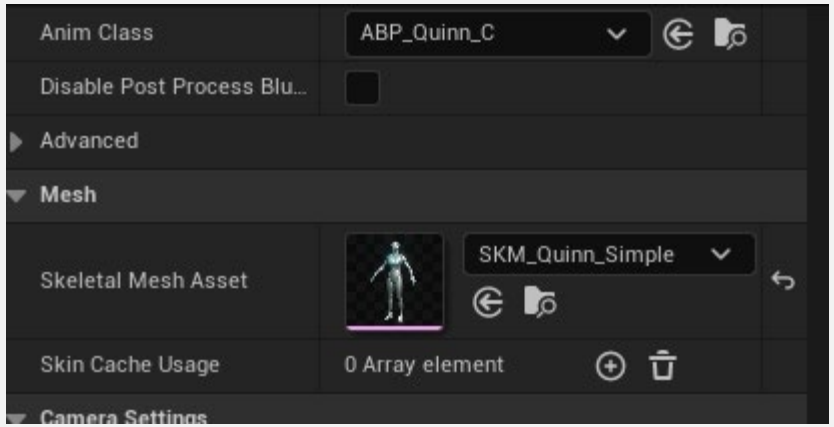
BP_PLAYERCHARACTERBPONLY

- Open BP_PlayerCharacterBPOnly



BP_PLAYERCHARACTERBPONLY

Select the Mesh component and add in the ABP and Skeletal mesh



In Components Select the Add button :

- Make a child of the Capsule: Spring Arm Component
- Make a child of the Spring Arm Component: Camera

BP_PLAYERCHARACTERBPONLY

Select Capsule

- Set radius to 42
- Set Half Height to 96

Select Spring Arm Component

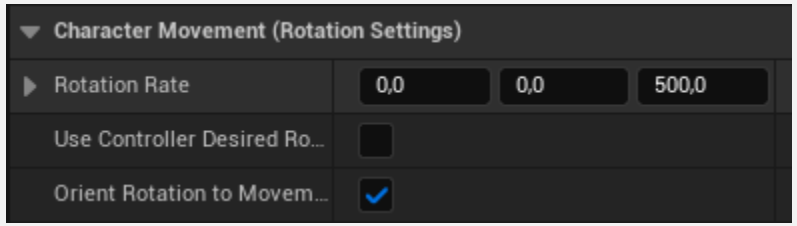
- Select Use Pawn Control Rotation
- Inherit Yaw, Pitch and Roll (Z, Y, X respectively)
- Set the target arm length to 400
- Mess with the Socket and Target settings if you wish (to get desired angle)

Select Camera Component

- UnSelect Use Pawn Control Rotation

Select Character Component

- Select Orient rotation to movement
- Set the rotation rate to 500 on the z-axis (Yaw)

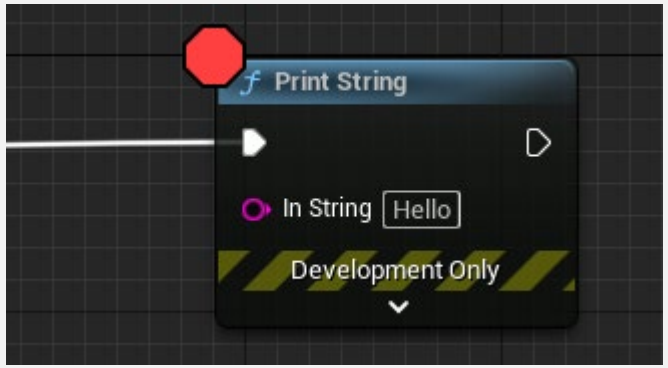


DEBUGGING

DEBUGGING

You first need to set a breakpoint

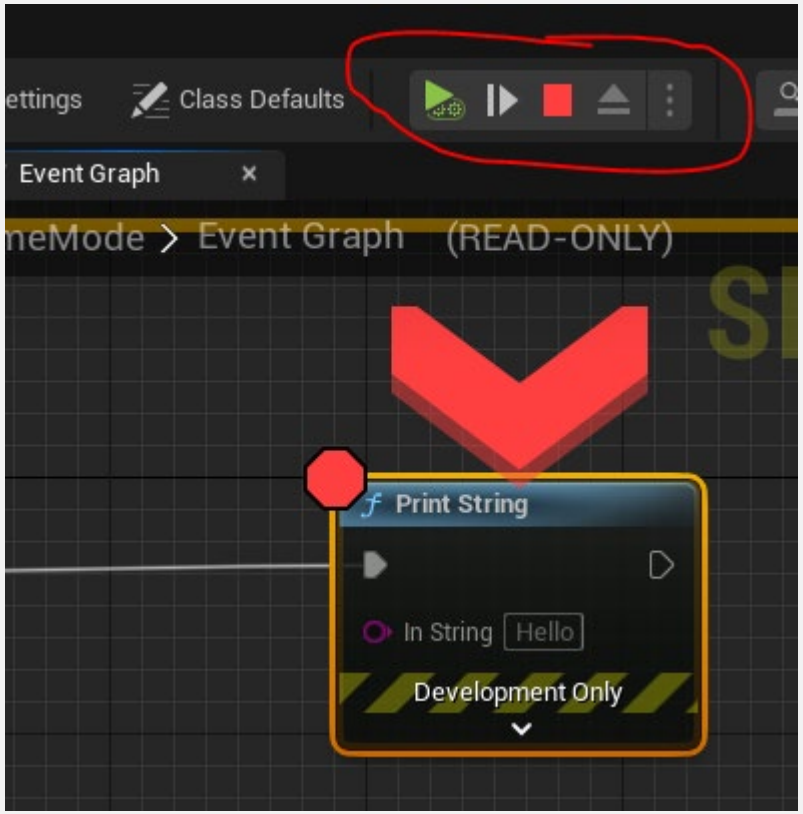
- Right-click while selecting a node in the BP, and select “Toggle Breakpoint”



DEBUGGING

Then when you press play the breakpoint will hit (if the function / node is called)

- Then you can step through the node step by step using the controls on the top of the blueprint.



ENHANCED INPUT

ENHANCED INPUT SYSTEM

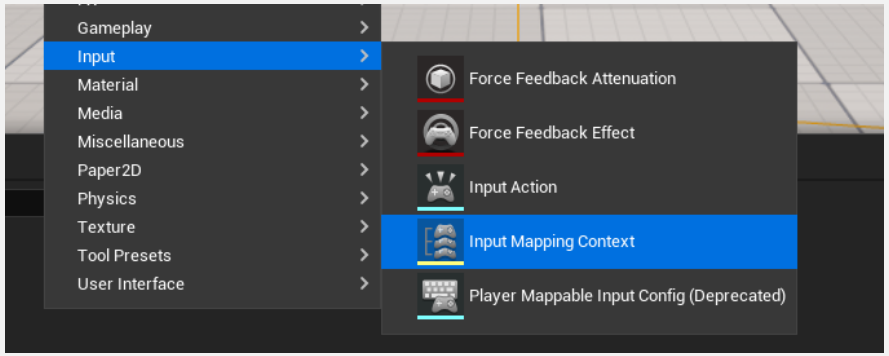
We need to setup the input system (we did this last week, this is a refresher)

We need:

- Input Actions
- InputMappingContext

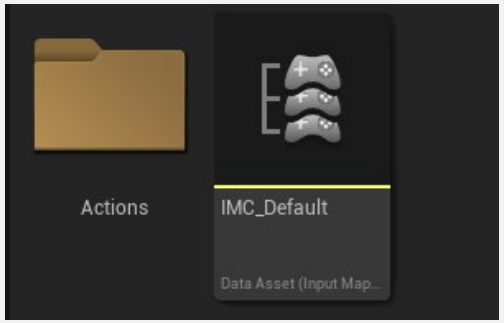
- First lets setup the InputMappingContext:
  1. Make a new folder in Blueprints called “Input”
  2. Inside “Inputs” make a new folder called “Actions”

  3. In the Input Folder, right-click -> hover over input -> select Input Mapping Context

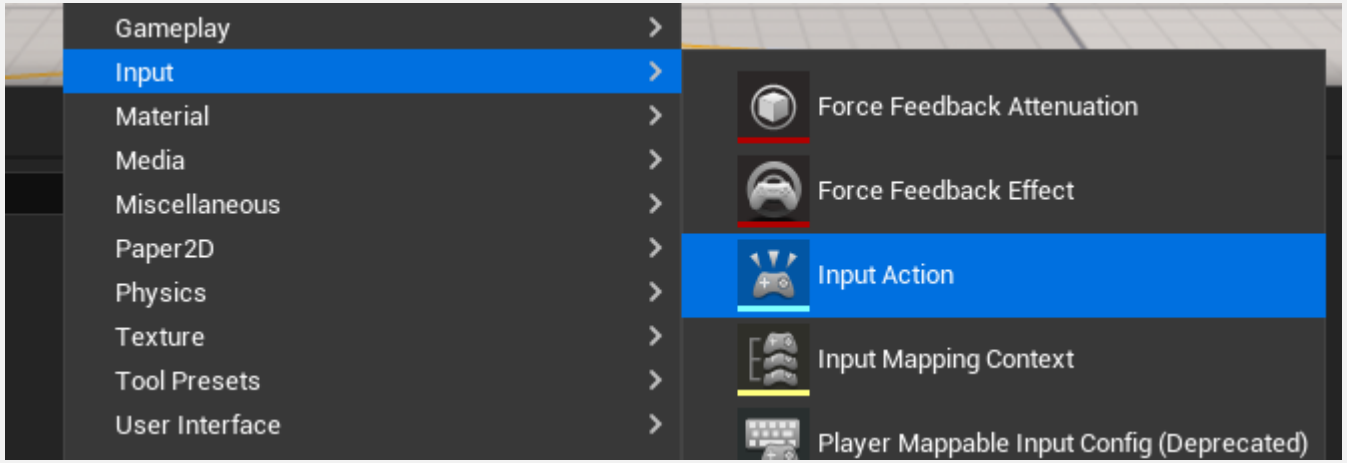


ENHANCED INPUT SYSTEM

4. Name it “IMC_Default” (InputMappingContext_Default)



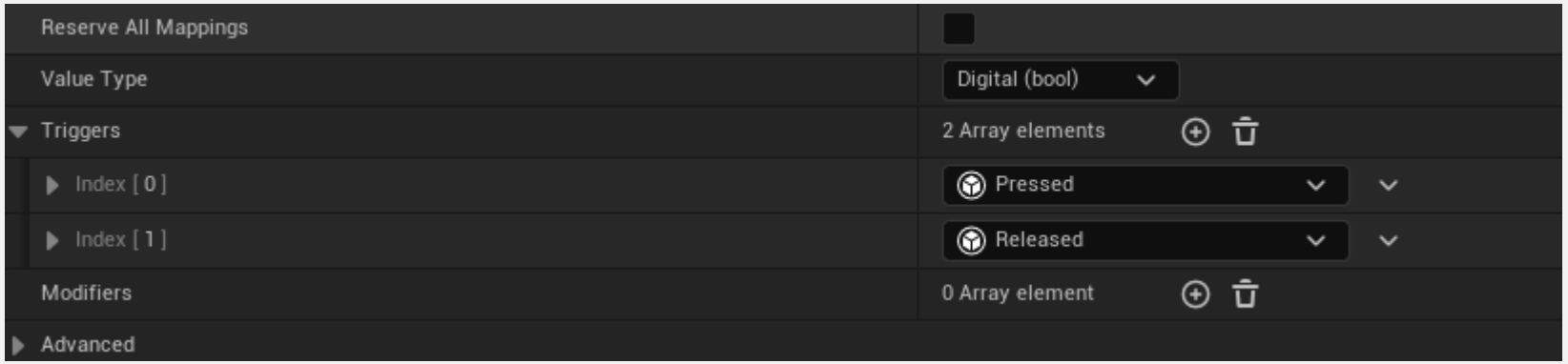
5. Open the Actions folder
6. Right-click -> Input -> Input Action



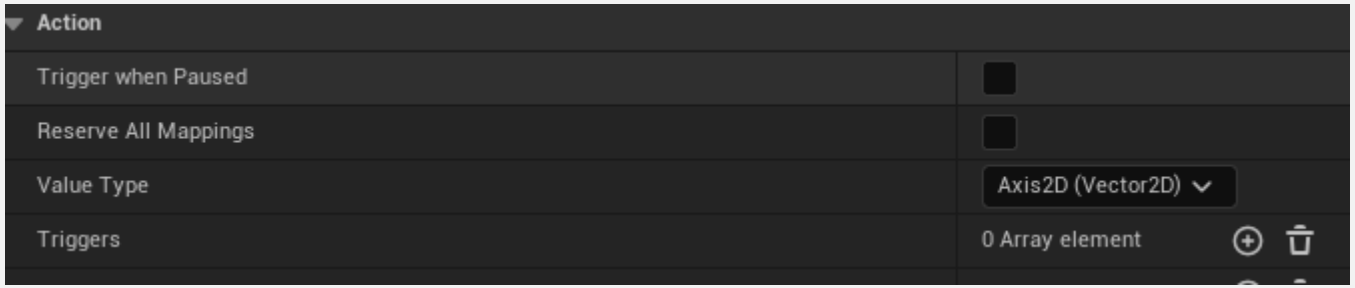
7. Name it “IA_Jump”, Repeat 6 and 7 for IA_Move and IA_Look

ENHANCED INPUT SYSTEM

7. Open IA_Jump, under Action
   - Create 2 triggers (press the plus sign twice)
   - Set the first trigger to Pressed
   - Set the second trigger to Released



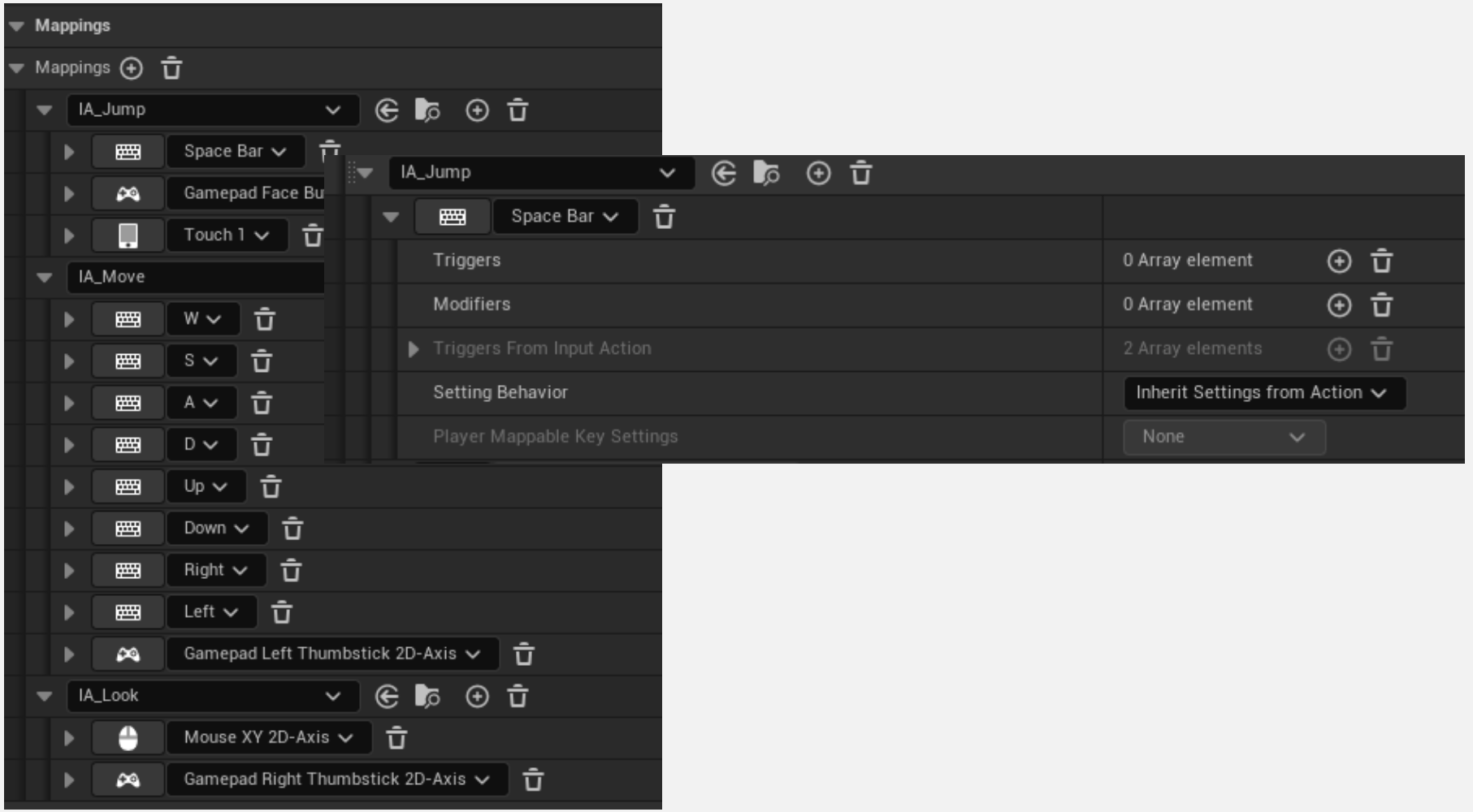
8. Open IA_Move, under Action
   - Change the value type to Axis2D(Vector2D)



9. Repeat step 8 for IA_Look

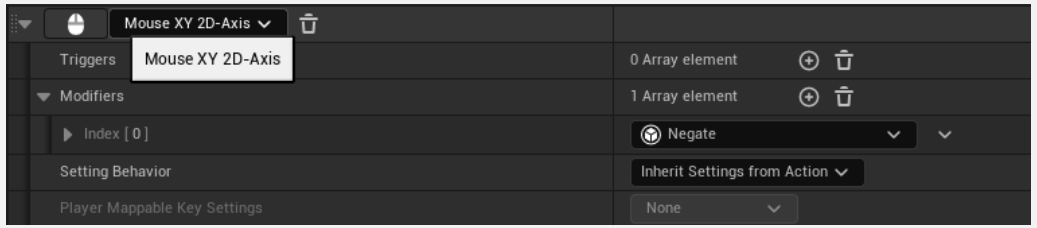
ENHANCED INPUT SYSTEM

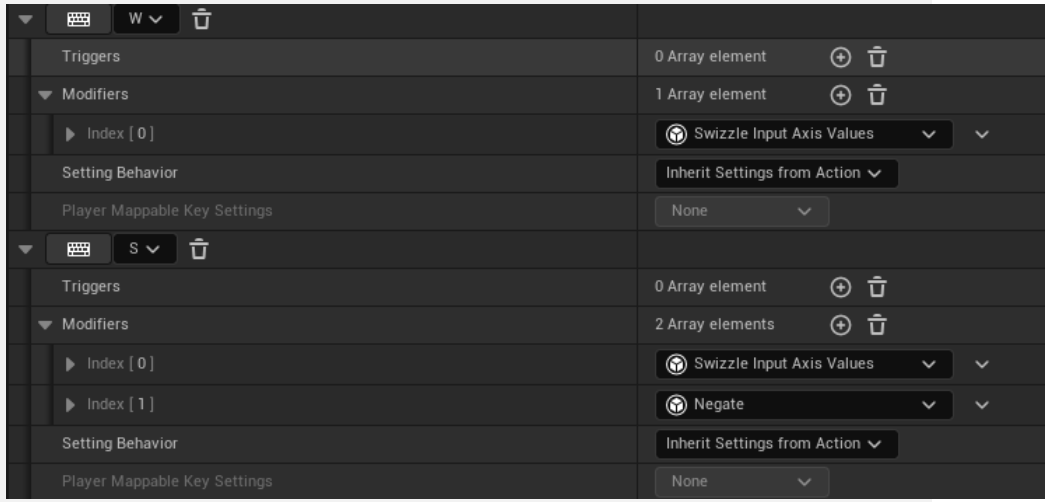
10. Open IMC_Default
11. Create 3 Mappings, for each IA Action



ENHANCED INPUT SYSTEM

12. In each Control you set, you need to adjust the modifiers
    - Jump – No Modifiers, it is inherited from Action
    - Forward – 1 Modifier -> Swizzle Input Axis Values
    - Backwards – 2 Modifiers -> Swizzle Input Axis Values + Negate
    - Move Right – No Mods
    - Move Left – 1 Mod -> Negate
    - Gamepad Thumbstick L – 2 Mods -> Dead Zone + Scalar
    - Mouse XY – 1 Mod -> Negate
    - Gamepad Thumbstick R – 1 Mod -> Dead Zone





IN BP_PLAYER

Assign our input mapping context and Assign our input actions

In BeginPlay

- We cast to our BP_PlayerControllerBPOnly
- Check IsValid
- Drag out our player controller to get “EnhancedInputLocalPlayerSubsystem”
- Drag put from that to get “Add Mapping Context”
- Select our “IMC_Default”

EnhancedInputAction IA_Jump

- Triggered -> Jump (from character movement component)
- Completed -> StopJump (from character movement component)

EnhancedInputAction IA_Look

- Triggered -> Add Controller Yaw Input -> Add Controller Pitch Input (tie in Action Value X to Yaw and Y to Pitch)

IN BP_PLAYER

EnhancedInputAction IA_Move

- Triggered -> Add Movement Input (tie in Input Action X)
  - Get Control Rotation
  - Split the vector (right-click on the vector out node)
  - Drag out Yaw to get “Get Right Vector”
  - Plug in Yaw and Roll respectively
  - Drag the vector node from “Get Right Vector” out to World Direction slot on “AddMovementInput”
- Triggered -> Add Movement Input (tie in Input Action Y)
  - Get Control Rotation
  - Split the vector
  - Drag out Yaw to get “Get ForwardVector”
  - Plug in Yaw only!
  - Drag the vector node from “Get Forward Vector” out to World Direction slot on “AddMovementInput”

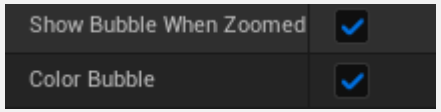
IN BP_PLAYER

Let’s organise things a bit!

Select all the begin play nodes and right-click “Create Comment from Selection”

Change the text to Begin Play, optional:

- Select a comment colour
- And enable:



- Do the same for each enhanced input action
- For Move make two more comments for the move right and move forward (can do inside the bigger comment)

MATERIALS

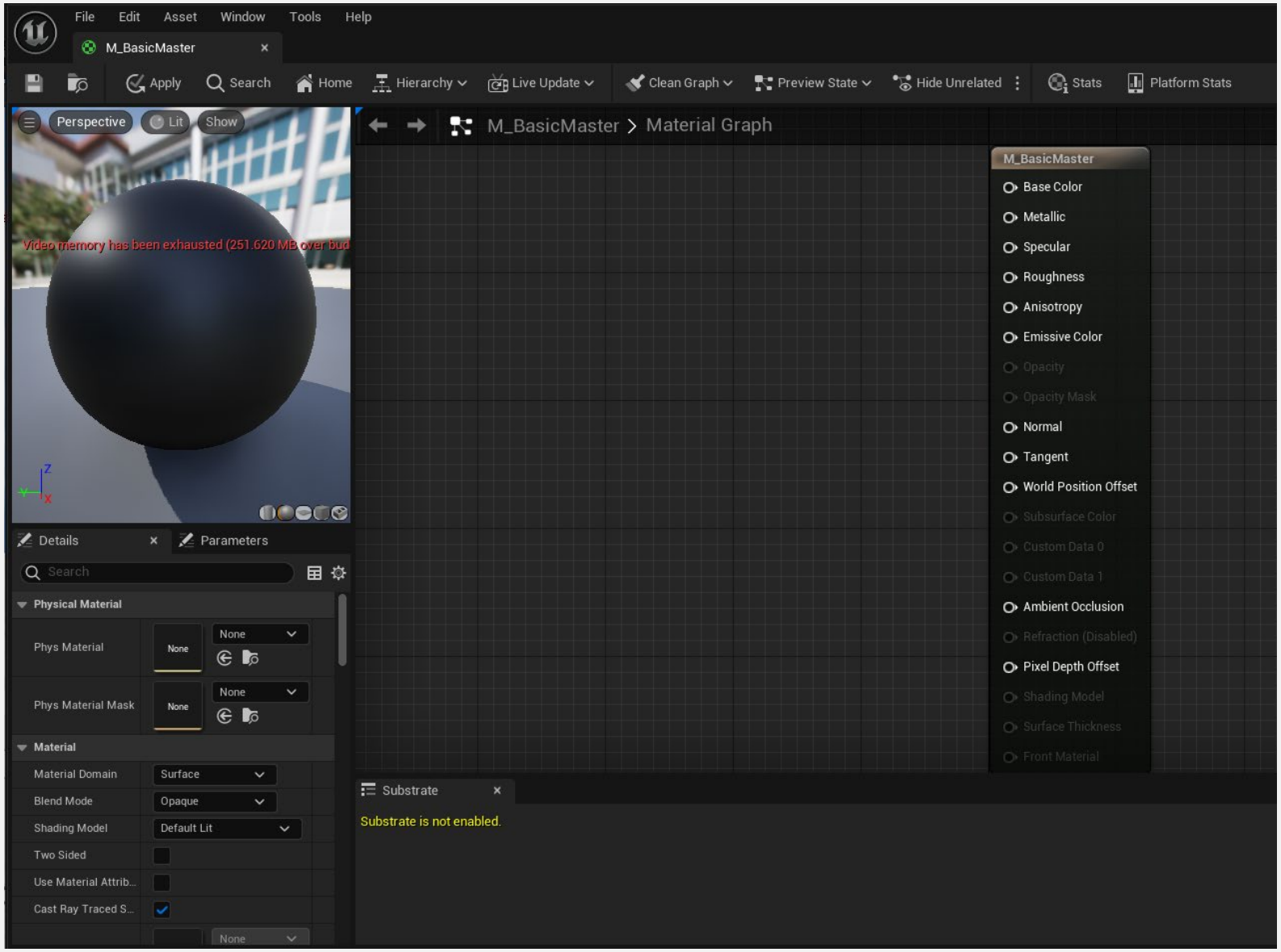
Parent and Children

Self-Coded Shading

SETUP A MATERIAL

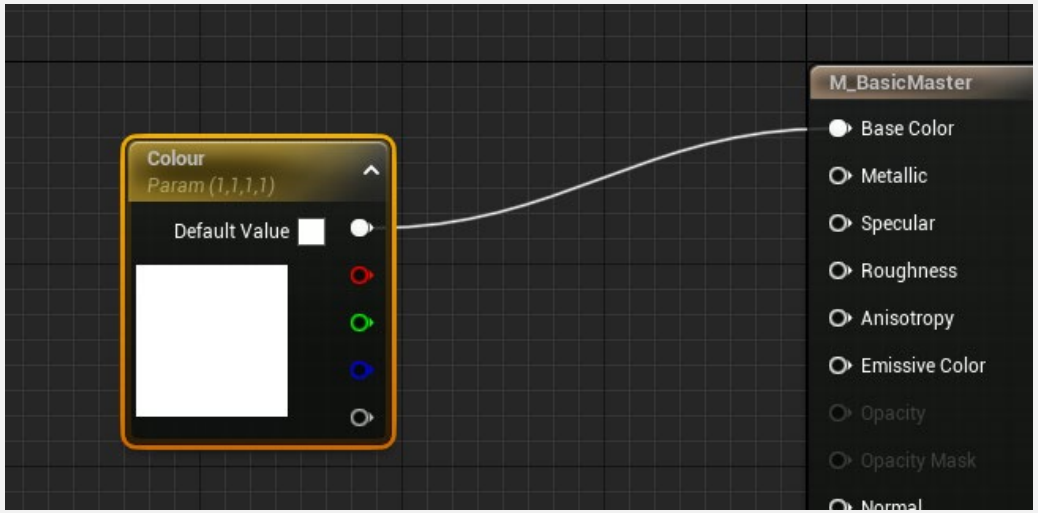
- Create a folder called Materials.
- Create a folder called Master.
- Inside Master, Right-click and select Material.
- Call this material ”M_BasicMaster”.

SETUP A MATERIAL



SETUP A MATERIAL

- Right-click and select ”VectorParameter”
  - This allows use to set custom colours
  - Drag the default node to the Base colors slot
  - Click Apply in the top left hand side (similar to Compile!)



- ”ScalarParameter” on the other hand allows us to use custom floats
- Constants of any type create a variable for you to use locally.
  - Right-click and select Convert to parameter to have a global variable (use in bps and children).

CUSTOM SHADERS

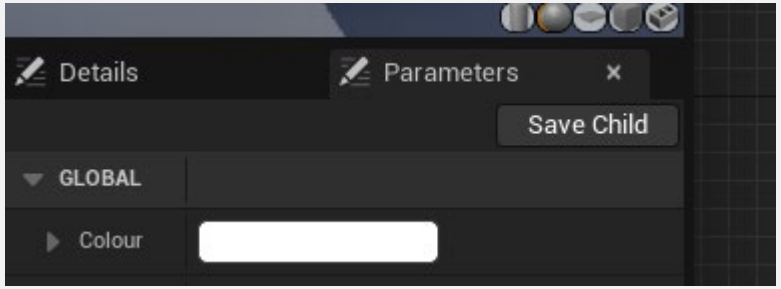
- Right-click and select ”Custom”
  - This allows us to write our own HLSL code
  - The inputs have to be sourced from outside the custom node.
  - The output can be directly connected to the base slot of the material.
  - Example: https://docs.unrealengine.com/5.3/en-US/custom-material-expressions-in-unreal-engine/

SETUP A MATERIAL

Create a child of this material (Material Instance)

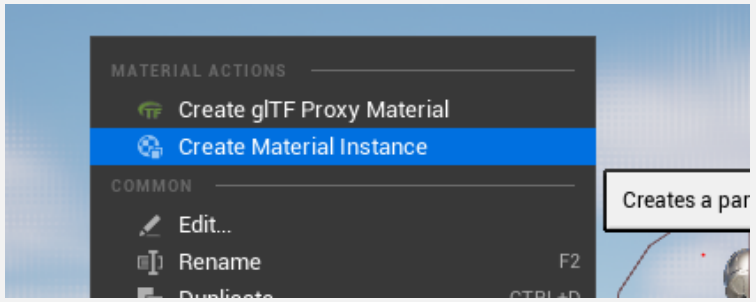
Option 1:

- Select Parameters Tab on the left side
- Select Save Child.



Option 2:

- Right-click on the material in the content drawer
- Select Create Material Instance

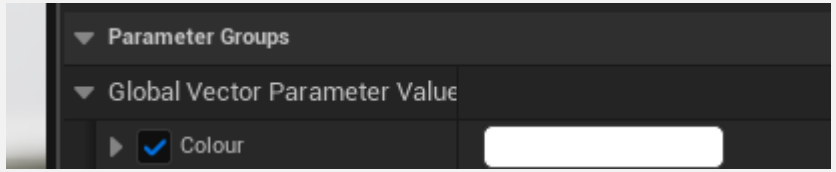


- Name it ”M_Green”
- Make sure it is saved in Materials not Materials->Master!

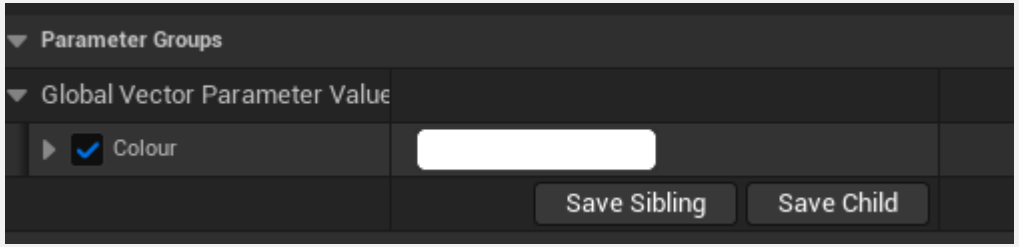
SETUP A MATERIAL

In the M_Green

- On the right-hand side, enable Colour and change the colour to green.
- Save the material



- You can also save as a Sibling for the colours Red, Blue etc.
- Or Save as Child. (but I do not recommend this)

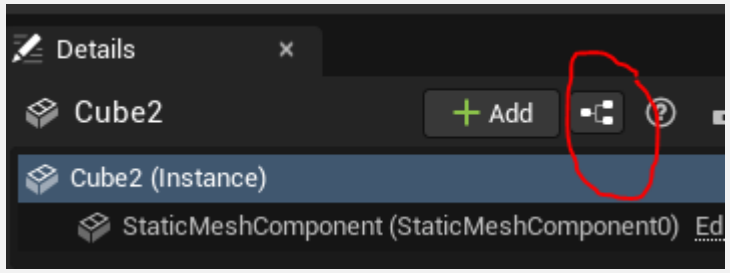


- You should try to never use Master Material in the scenes and on game objects. Use the instanced material!

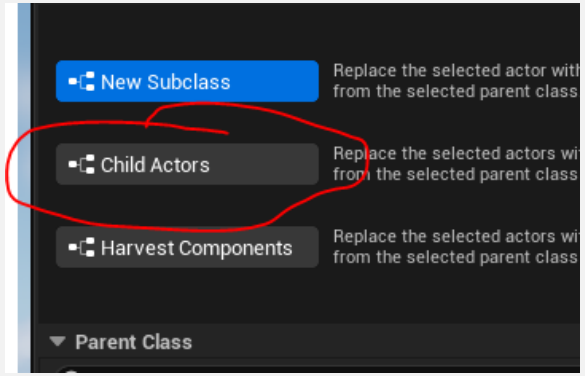
SET THE MATERIAL TO CUBES

In Scene drag and drop a cube from the menu.

- In the details panel select, create Blueprint.



- Name this BP_ColourCubes
- Choose New Subclass



- Child actors take the instanced mesh and you will not be able to swap out the materials.
- Save it in the Actors folder (create it first) in Blueprints
- In the new BP, save and compile

SET THE MATERIAL TO CUBES

- Make a few children of the BP_ColourCubes
- Name them BP_ColourCubes_Red etc. (for each colour)
- Move the BP_ColourCubes into Masters folder within the Actors folder
- In each child BP, change the material to the instanced materials created earlier, each of the colours.

SET THE MATERIAL TO CUBES

- Make a few children of the BP_ColourCubes
- Name them BP_ColourCubes_Red etc. (for each colour)
- Move the BP_ColourCubes into Masters folder within the Actors folder
- In each child BP, change the material to the instanced materials created earlier, each of the colours.

COLLISION HANDLING

OnEnter

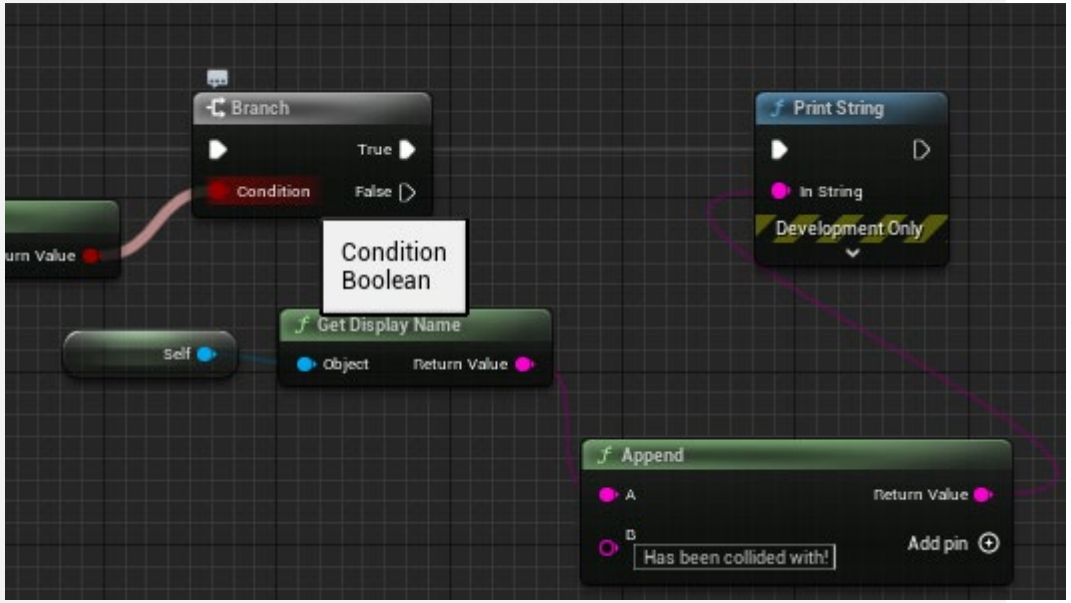
BP_COLOURCUBES

In BP_ColourCubes (Master):

- Select the Cube Static Mesh
- Scroll down until you find collision options.
- Choose:
  - Generate Overlap Events
  - Scroll further down to Events
  - Select OnComponentBeginOverlap

In OnComponentBeginOverlap:

- Drag out to ”Cast to BP_PlayerCharacterBPOnly”
- Check isValid
- On branch true, drag out to Print String
- Get reference to Self
- Drag out from Self to Get Display name.
- From Getdisplay name drag out to ”Append”
- Add in Slot B ” Has Been collided with!”



BP_COLOURCUBES

- Add all the BP_ColourCubes_Red etc. to the scene!
- Now everytime our player collides with the cube, we get a nice little output message.
- This is how we collide with things!

SOUND

Sound Cues

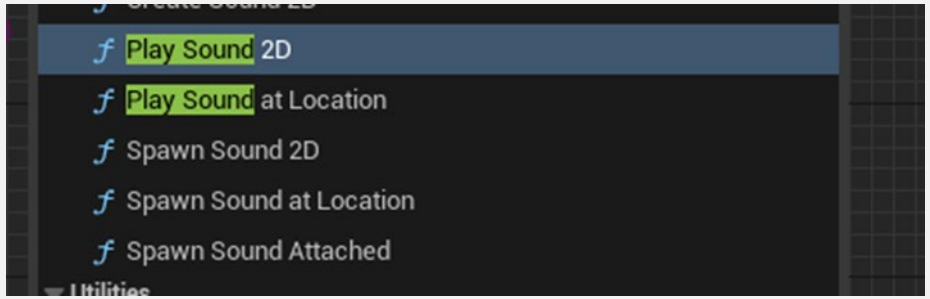
Master Controls

SOUND CUES

- Now when we collide we also want to hear something
- Make a folder called Sounds
- Inside make a Sound Cue called ”SC_Hit”.

In BP_ColourCubes

- Drag out from print string and Select Play Sound 2D



- Select our Sound Cue ”SC_Hit”
- Compile and Save!

SOUND CUES

- Now when we collide we also want to hear something
- Make a folder called ”Audio”. In the Sounds folder.
- Import .wav files (can find a random one from the internet or your files) **
- Unreal likes .wav files it does not like .mp3! (So convert them if needed!)
- Save your .wav files in the Audio folder

In SC_Hit

- Open the content drawer and drag and drop your .wav file in the sound cue.
- OR Get the node ”Wave Player” and select a sound from the drop down on the left side.
- Connect the node to Output.
- Save!
- You can test the sound by pressing Play Cue

**You can also use the unreal content to find a wav file to use

SOUND CUES

- Now when we collide get our message and a sound!

**You can also use the unreal content to find a wav file to use

ANIMATION CONTROLLER

POSSESSION

Switching Characters

Camera work

SEQUENCER

Movies

Clips

Cutscenes

LIGHTING

ARTIFICIAL INTELLIGENCE

WORKING WITH 2D

Paper Sprites

CLOSING NOTES

Next Week