**Intro**

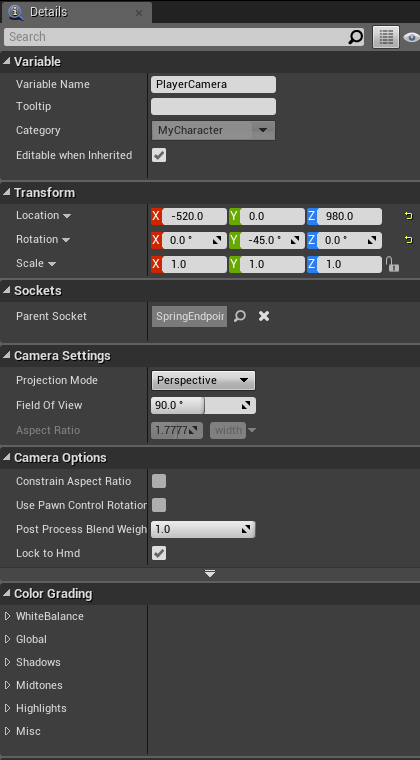
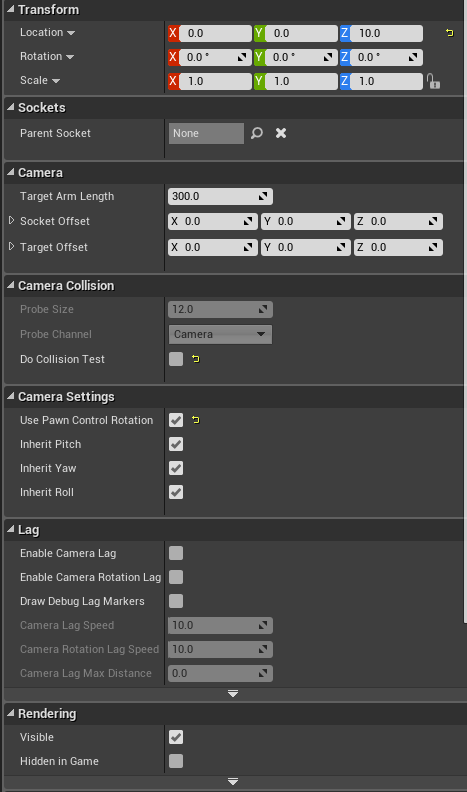
This is the tech document for the Unreal Engine 4 remake of Getting Home.  
These documents all the tech made on the prototype build.  
This document is to be used as reference when creating the final product from an empty UE4 map, rather than the 3rd person placeholder map.

**Player Movement**

The player navigates the map with the W, A, S, D keys as is conventional for movement on PC games. Using a controller, the player would use the Joystick.

**Camera**

The players camera is attached to a CameraBoom that is attached to the player.

  
Camera Boom; Camera;  


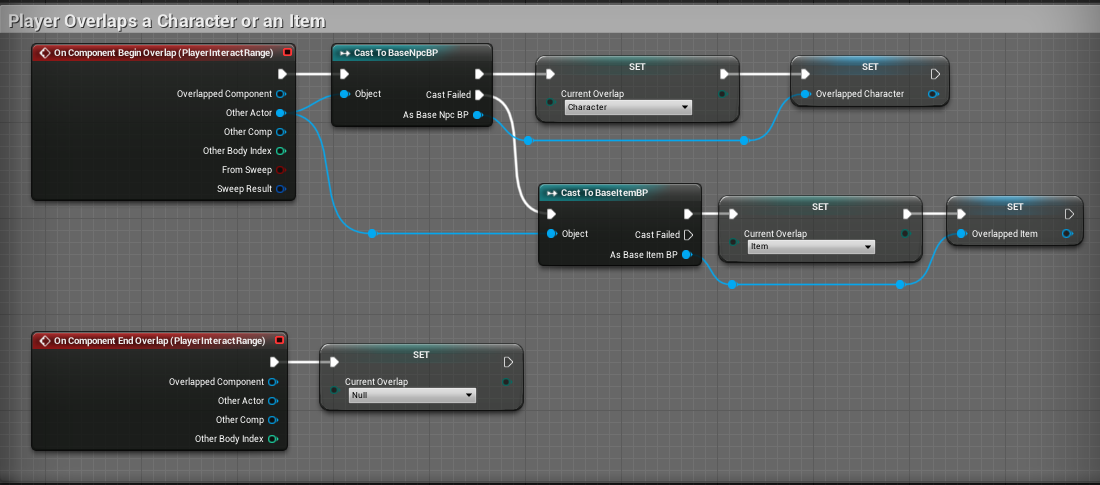
**Player Interaction**

The player interacts with the world by pressing the E key (InteractInput)

Settings – Project Settings – Input – Action Mapping – ‘InteractInput’ – E

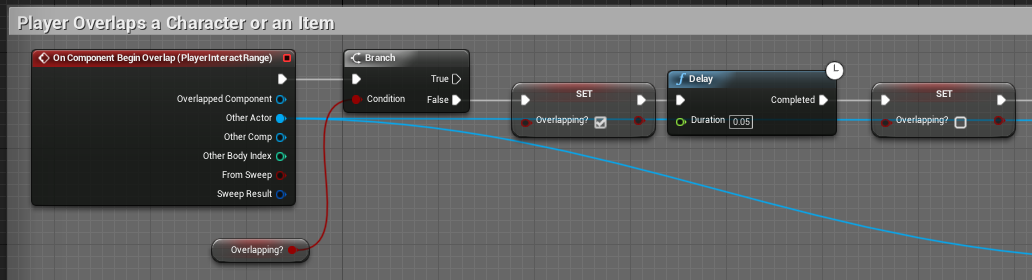
When a player gets close to an interactable element (NPC, Item, ect…) the player has a box collision that checks what the player is overlapping and sets ‘CurrentOverlap’ to either NPC or Item then updates either OverlappedCharacter or OverlappedItem accordingly;

CurrentOverlap is an Enum that tracks what type of interactable the player is currently overlapping; NPC or Item

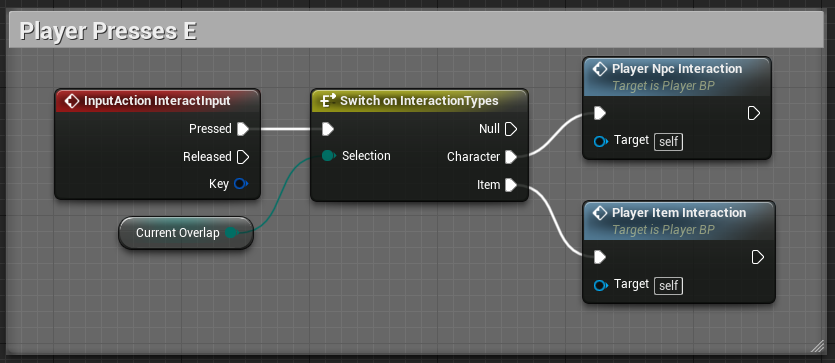


**The player should only be able to overlap ONE interactive element at a time.**

There was a bug where the player would overlap with the same object or NPC multiple times when the interactable entered the players trigger box. To solve this, I’ve added this tiny delay so that it only happens once;

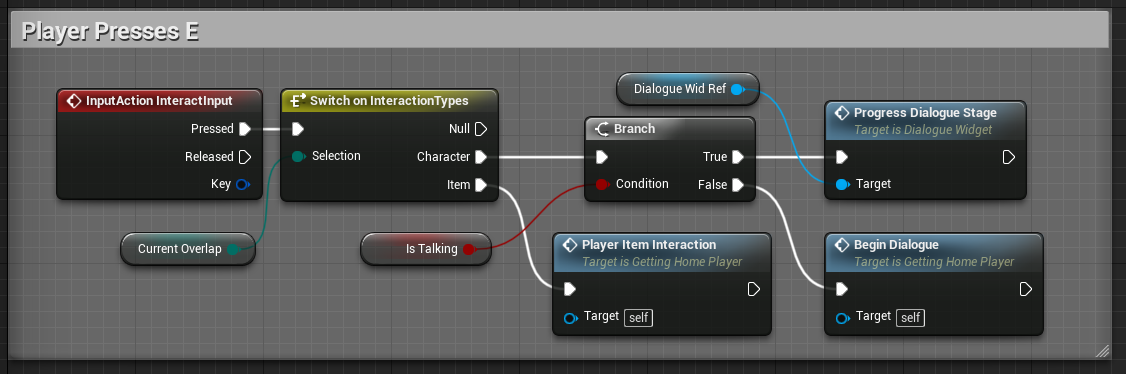


When the player presses the InteractInput there is a switch based off ‘CurrentOverlap’. If it’s an NPC, the player attempts to talk to them.  
If it’s an item, the player attempts to pick it up.

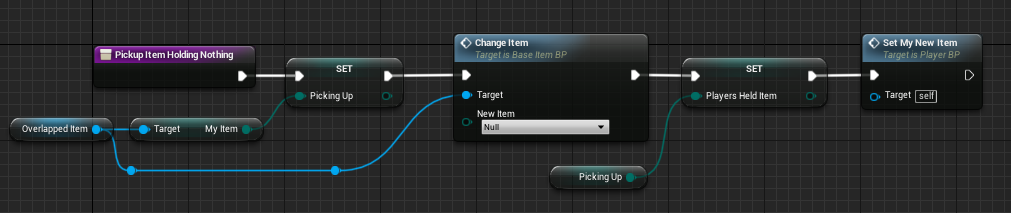


**Player to Item Interaction**

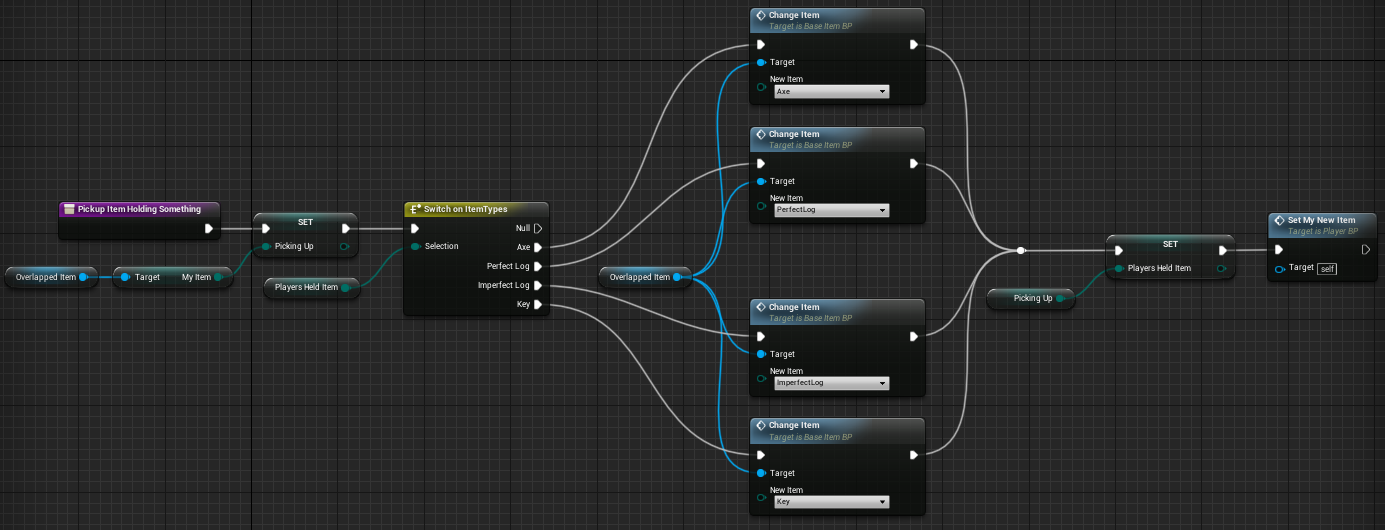
When the player attempts to interact with an item, they attempt to pick it up.  
The player can only hold ONE item at a time.  
So, when the player attempts to pick something up, the first thing we need to check is whether or not the player is already holding something; This is a switch based off of ‘PlayersHeldItem’ which is an Enum of all the items the player could pick up.



If the player is holding nothing, the player sets whatever item they are overlapping as the item they are picking up before telling the item to change its item to Null, since the player is putting nothing down. The player then sets ‘PlayersHeldItem’ to whatever item they were attempting to pick up;



If the player IS holding an item already, then the object being picked up needs to be told what the ‘PlayersHeldItem’ is, as this is what they player will be putting down in place of whatever they’re picking up;

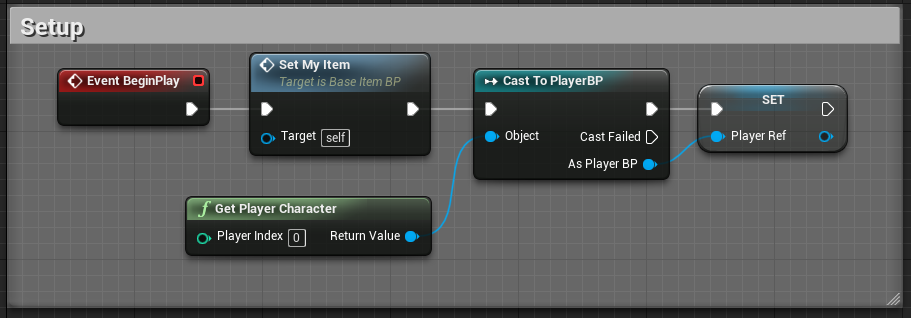


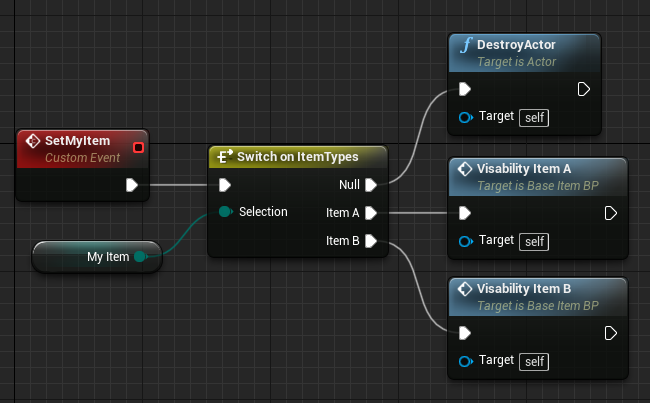
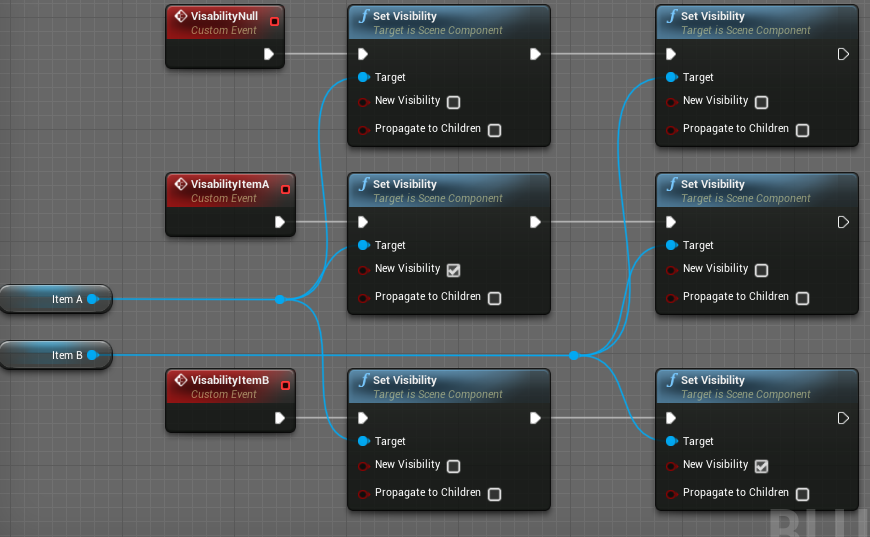
**Base Item BP**

The only things the BaseItemBp needs to do is;

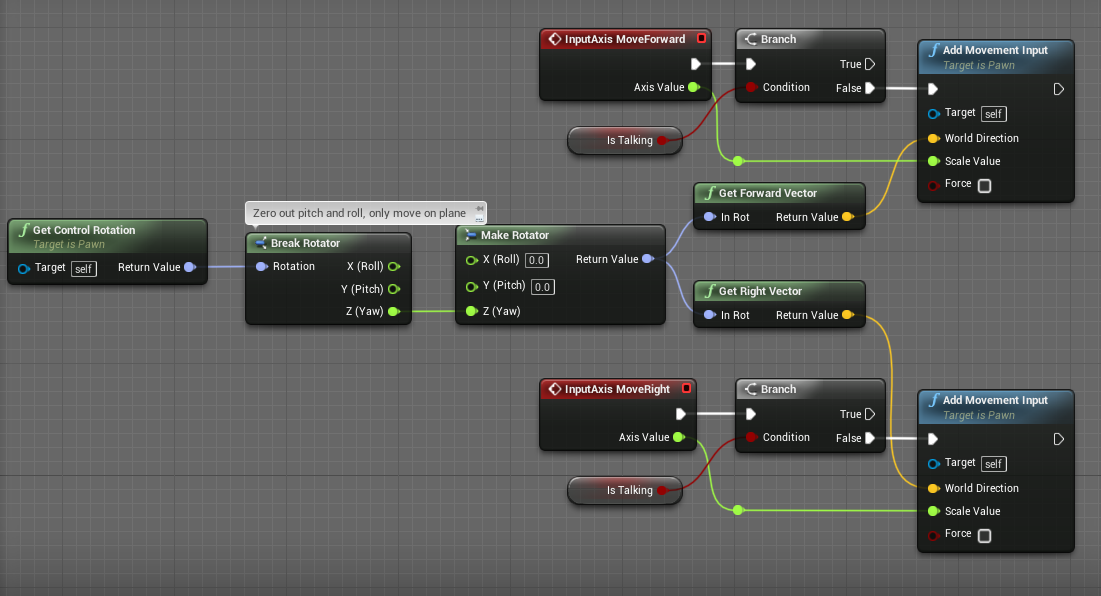
* Know what item it is ‘MyItem’
* Know what item it needs to turn into. This is told to it by the player.
* Update the visibility of all the items so that what is being shown is the same as ‘MyItem’
* Destroy itself if MyItem’ is null.

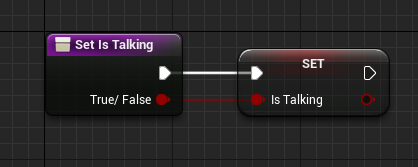
‘MyItem’ is an Enum of ‘ItemTypes’. This variable is public so the levels designer can place ONE item BP rather than a unique one for each item. This allows the levels designer to set what ‘MyItem’ is from the level editor without needing to go into Blueprints.



**Player to NPC Interaction**

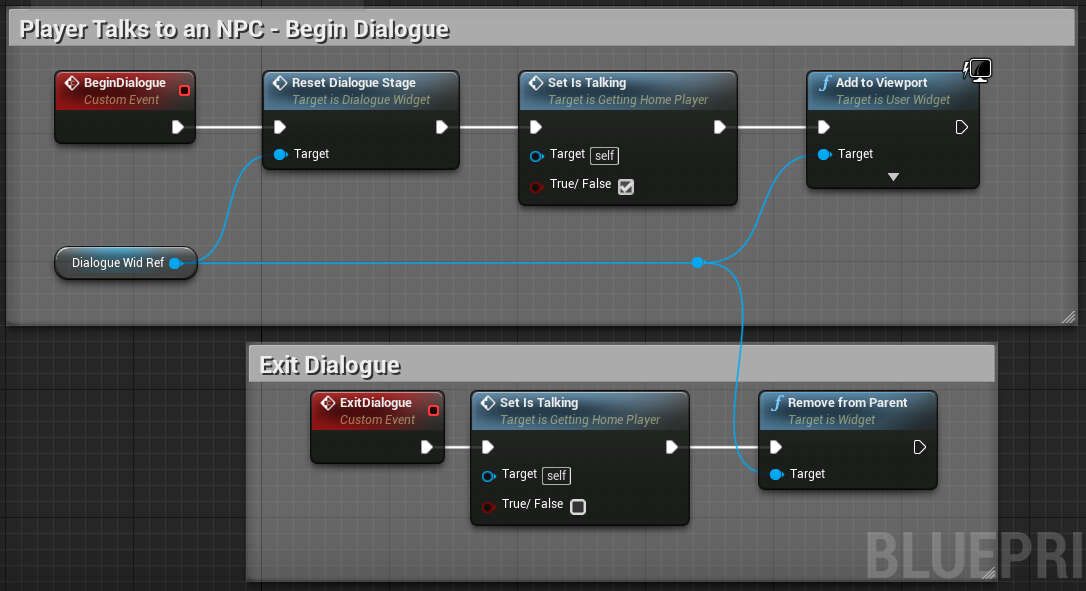
When a player attempts to interact with an NPC, they are always trying to talk to them.  
When a player talks to an NPC, they lose movement input. This is done by setting ‘IsTalking’ to True. Movement input is given back by setting ‘IsTalking’ to False.



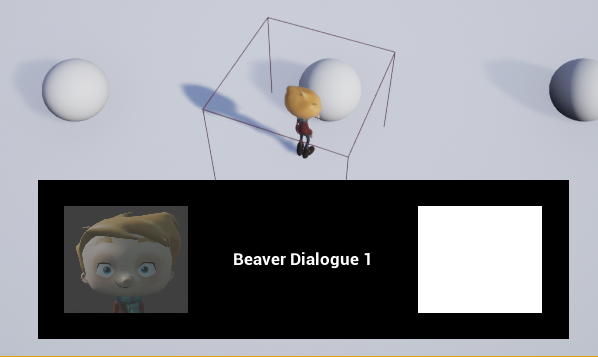


**Dialogue Widget**

When the player talks to an NPC, we add a widget to the players viewport; ‘DialogueWidget’;

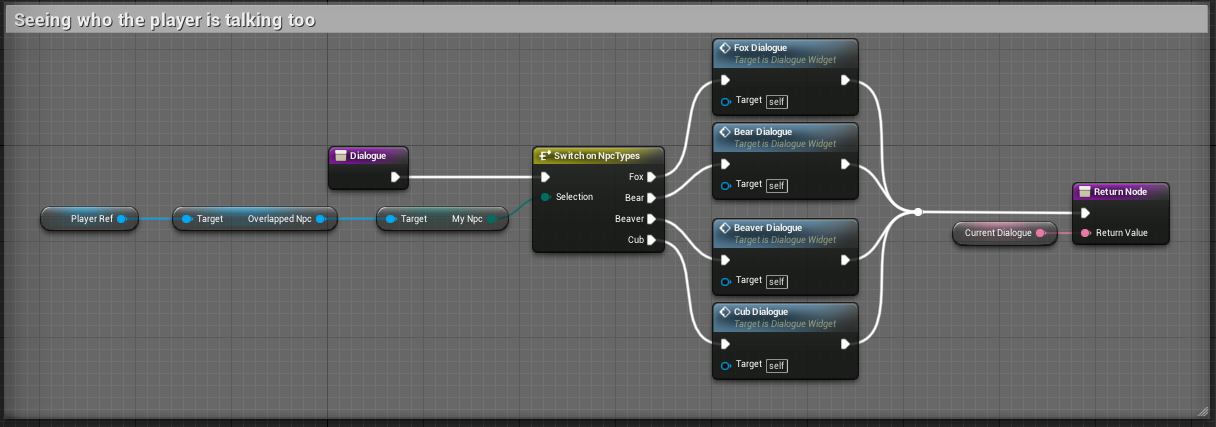


DialogueWidget displays an image of the player, the Npc they are talking too, and text.  
Whoever is NOT talking is greyed out;

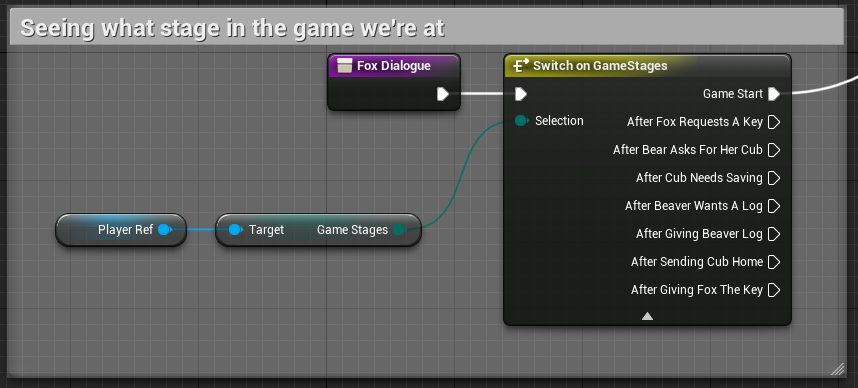


**DialogueWidget – Getting Dialogue**

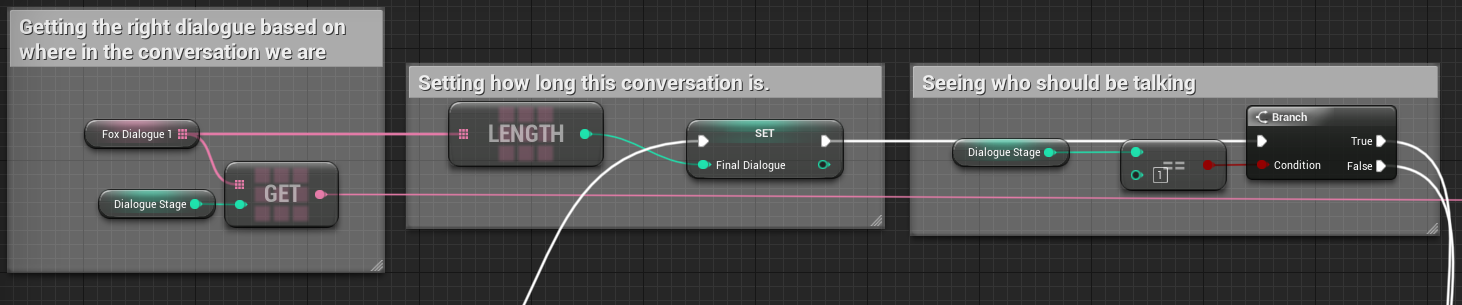
The DialogueWidget updates the text it displays based off of who the player is talking too;



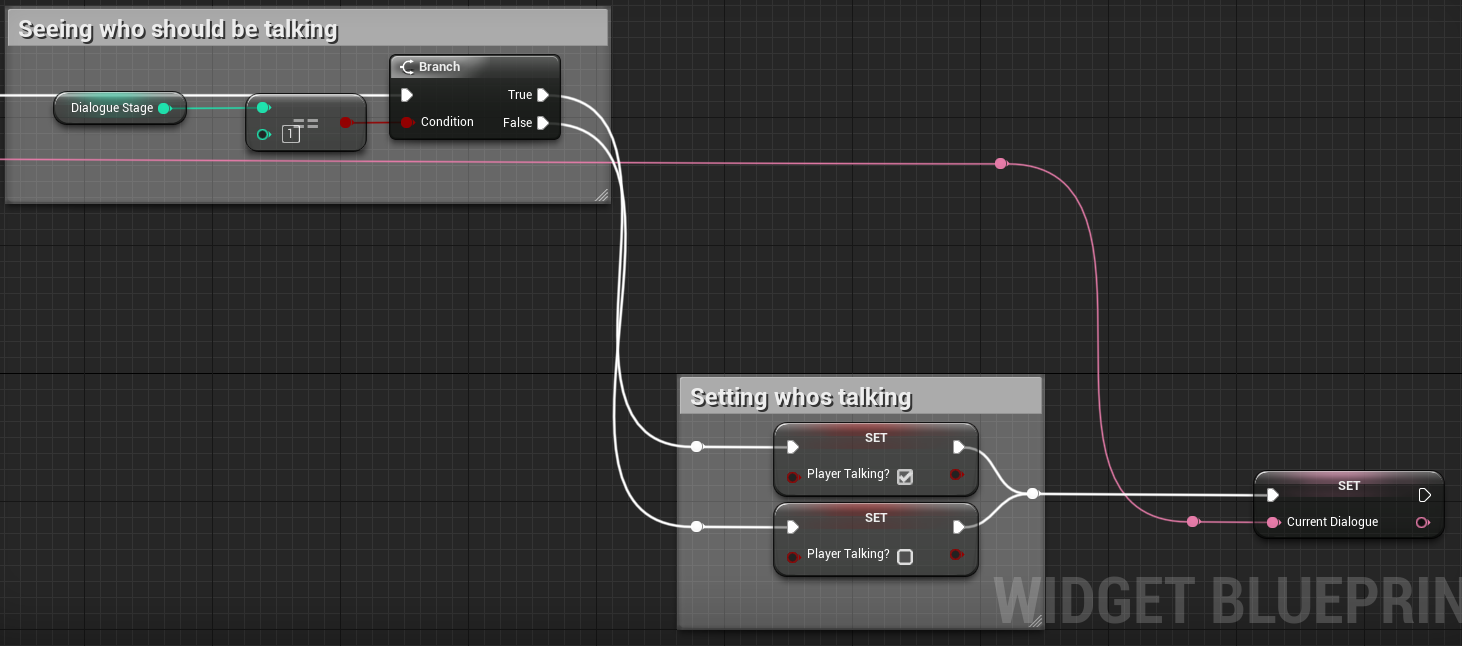
What stage of the game the player is at;



It then checks what stage of the current dialogue the player is at;

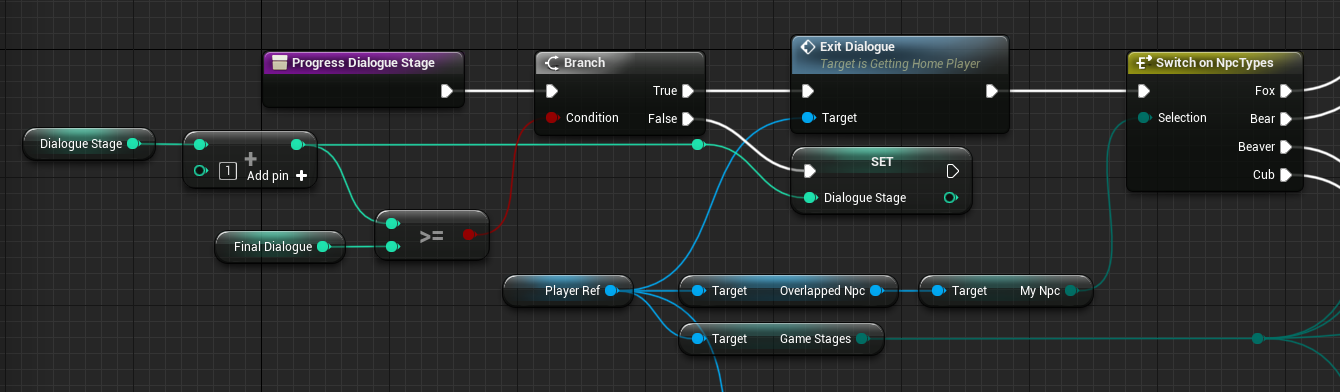


It then checks who should be delivering that line of dialogue and sets PlayerTalking to either True or False. True if the player should deliver the dialogue, false if the Npc should;



Now that we know who the player is talking too, what stage of the game the player is at, what stage of the conversation the player is at, and who is delivering that stage of the conversation, CurrentDialogue, a text variable, is set to the appropriate line of text.

What stage of the conversation the player is at is updated when the player presses the ‘InteractInput’ while already talking to an Npc. It is updated by getting the current DialogueStage, an Int variable, and adding 1 to it. We then check if the player has reached the end of the conversation. If they have, the dialogue will end when the player next presses the ‘InteractInput’ or will proceed to the next stage of the conversation if they have not.

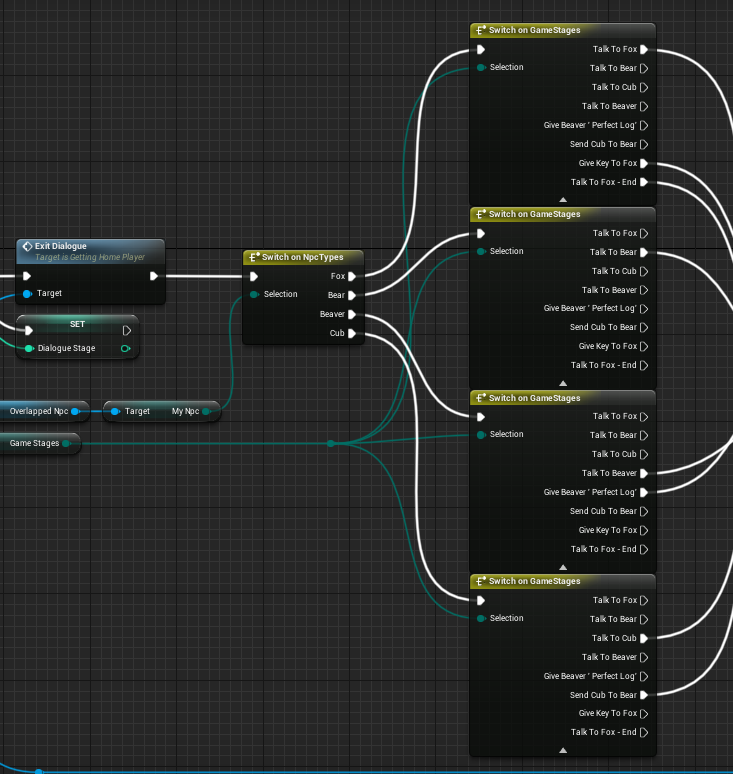


**Game Progression**

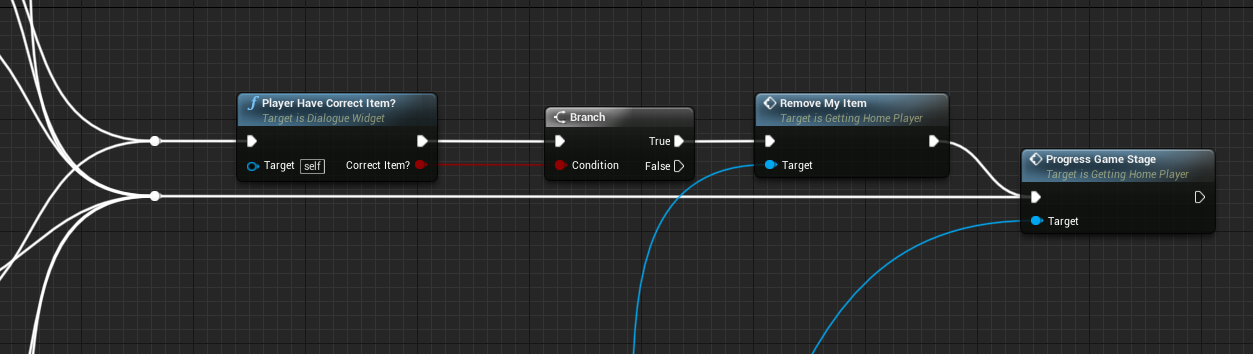
The game progresses based off of a Enum variable on the Player, ‘GameStages’. GameStages as an Enum is a list of all the different stages of the game;



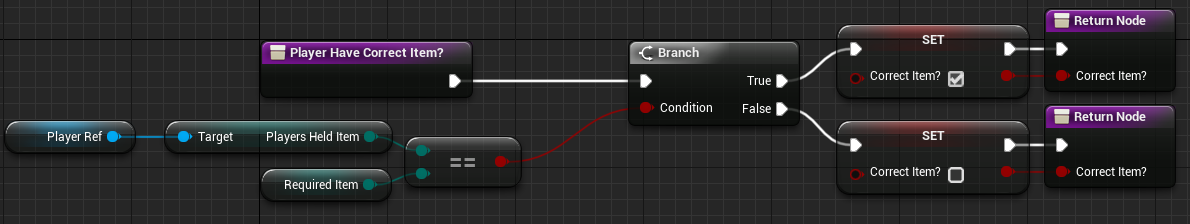
Progression along this Enum is triggered when the player speaks to the correct Npc at the correct time.



When a specific item is required to continue the game, it checks that the item the player is holding is the required item;



‘PlayerHaveCorrectItem?’ function;



We tell the DialogueWidget to set what the required item to continue the game when we update the GameStage within the Players Blueprint;

