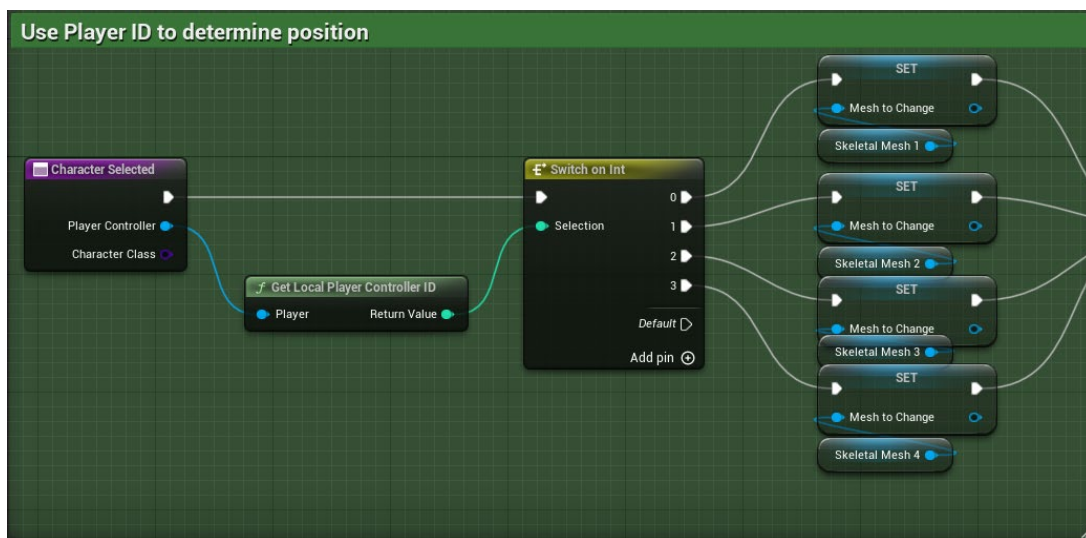


Character Animation Integration

This document will go over the implementation of trigger the character select and stage entrance animations. To implement these features, I experimented with several different approaches before finding the one that worked the best for our project. This required a bit of creative problem solving and collaboration with artists to find the best settings to display their art and animations.

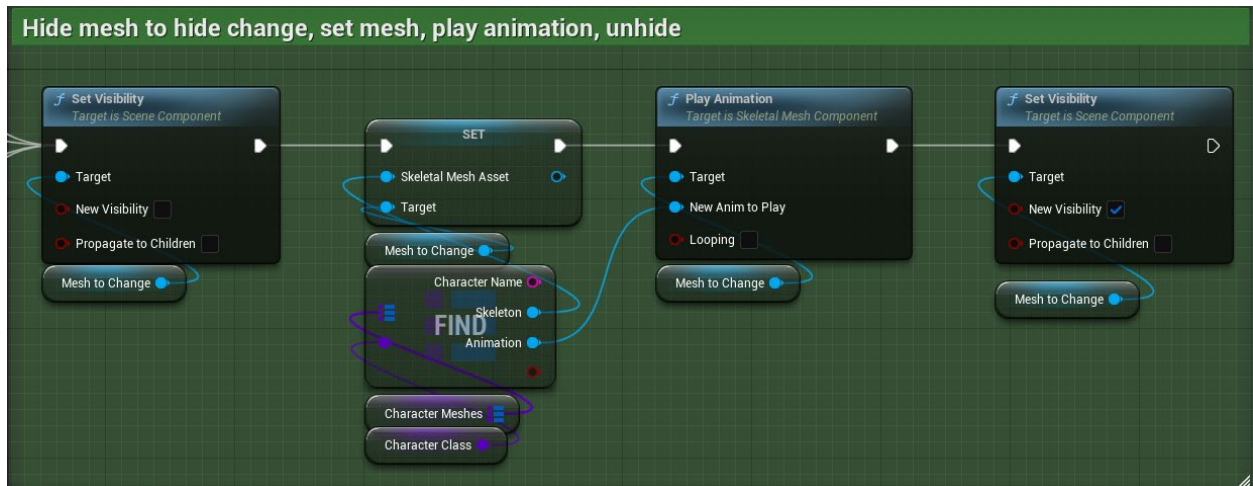
Character Select Animations

For the character selection animations, we initially explored setting up 2D Scene Capture components to capture the meshes for the selected characters and then render these to a texture for use on the widget, but it looked rather flat, so we decided to try a 3D scene instead. I made an actor with a 3D Widget component on it, and four skeletal mesh components positioned in front of each character select scripts. When a character is selected, it calls an event dispatcher which the actor subscribes to. The actor then checks which player it was that triggered the event using their Controller ID, find the skeletal mesh that corresponds to them.





1 Blueprint snippet of the determining Skeletal mesh for the character being selected.

This mesh is then made invisible, the skeletal mesh is set to the character they chose, and the animation for that character is set to play, and the mesh is made visible again. The visibility is toggled off and on because I noticed that there was a frame of two before the animation started where the mesh was in its default pose.



This is handled using a Map where the Key is the Character Class, and the value is a Struct Marissa had set up for a snapshot mode, which stores the character name, skeletal mesh, and an animation.

| BP_CEO | | 3 members |
|---------------|---|------------------|
| CharacterName | Elias Kane | |
| Skeleton |  | SKM_CEO_Proxy_01 |
| Animation |  | AS_Zoners_Idle |

2 Example of an entry in the Map used in this function.

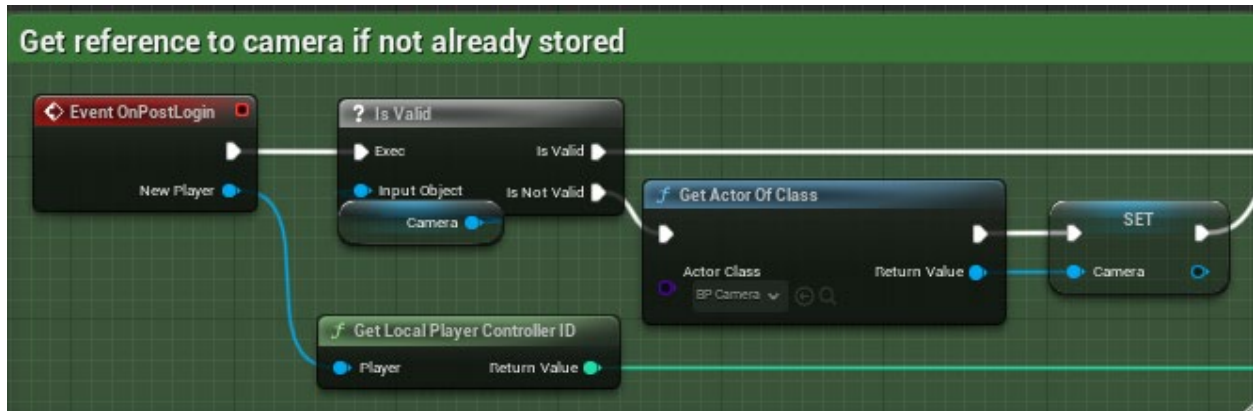
Stage Entrance Animations

The stage entrance animations were a more challenging feature than I anticipated, not because of the logic for the feature on its own, but because of how the previous system was set up. Previously, the characters were spawned as soon as the player controller connected to the game mode, in the *OnPostLogin* function. I initially was going to add the functionality for playing the animations here, but quickly found that caused issues, as the *OnPostLogin* function runs before *OnBeginPlay*, so it would be unable to reference the characters it was spawning. The solution for this ended up being to simply move the functionality to *OnBeginPlay*, but it caused quite a headache before I figured this out.

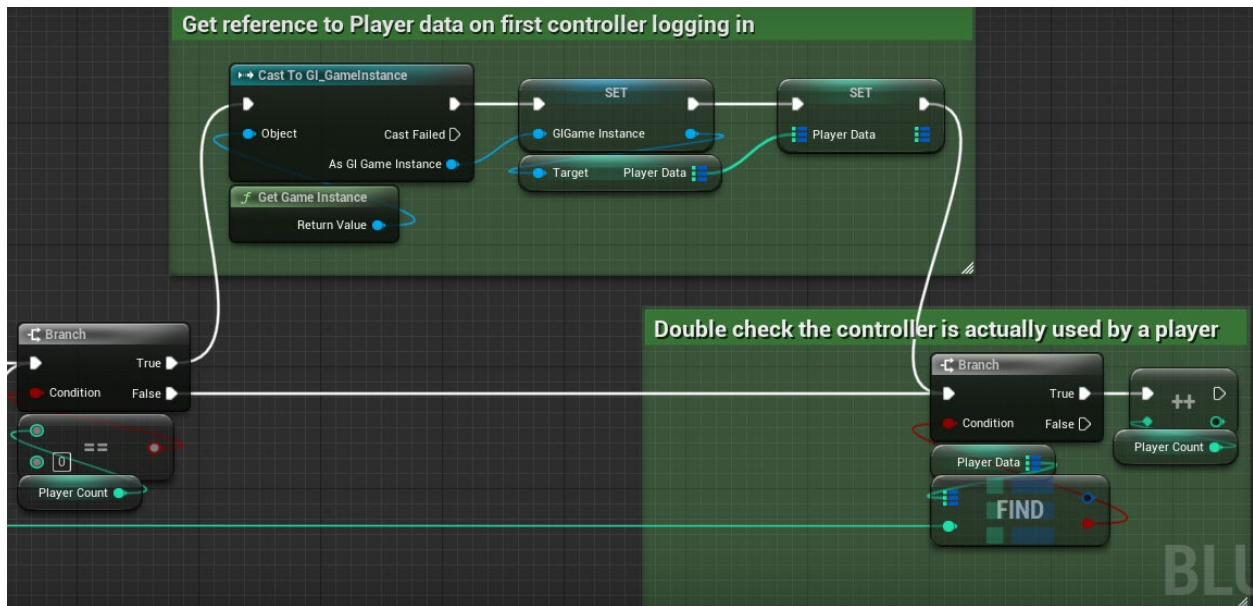
Another issue that came up was that because the spawning of characters is done in the parent game mode, there were some issues with the game modes that inherited from it after the change was made. For example, because we were delaying when the characters

were spawning in so they would appear when the animations played, the camera didn't have references until the first character spawned, and not be registered as the main camera for some reason. To fix this, I just manually set the Camera as the view target on begin play.

When *OnPostLogin* runs, I first get a reference to the camera if one isn't already set.

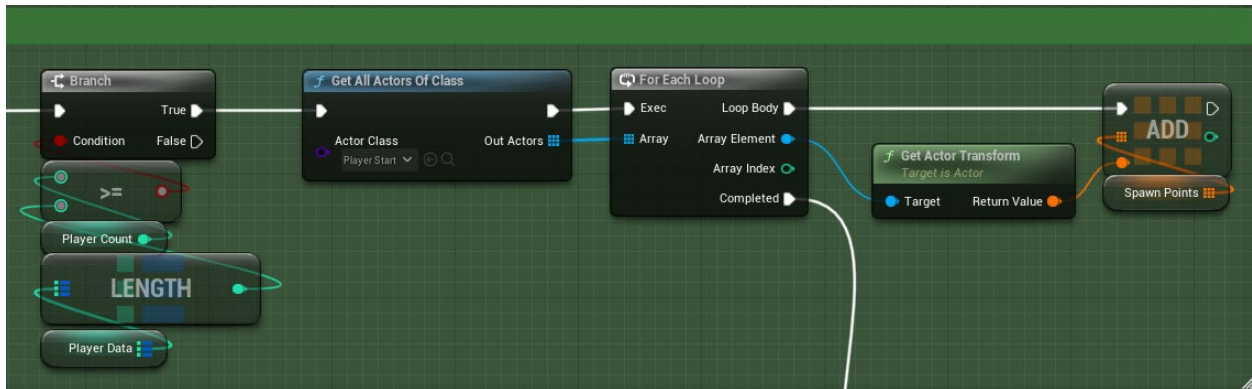


After this, I get a reference to the *PlayerData* map if we haven't counted any players yet, then if the Controller ID for the Player Controller that's connecting is registered in the *PlayerData* Map, I increment the PlayerCount.

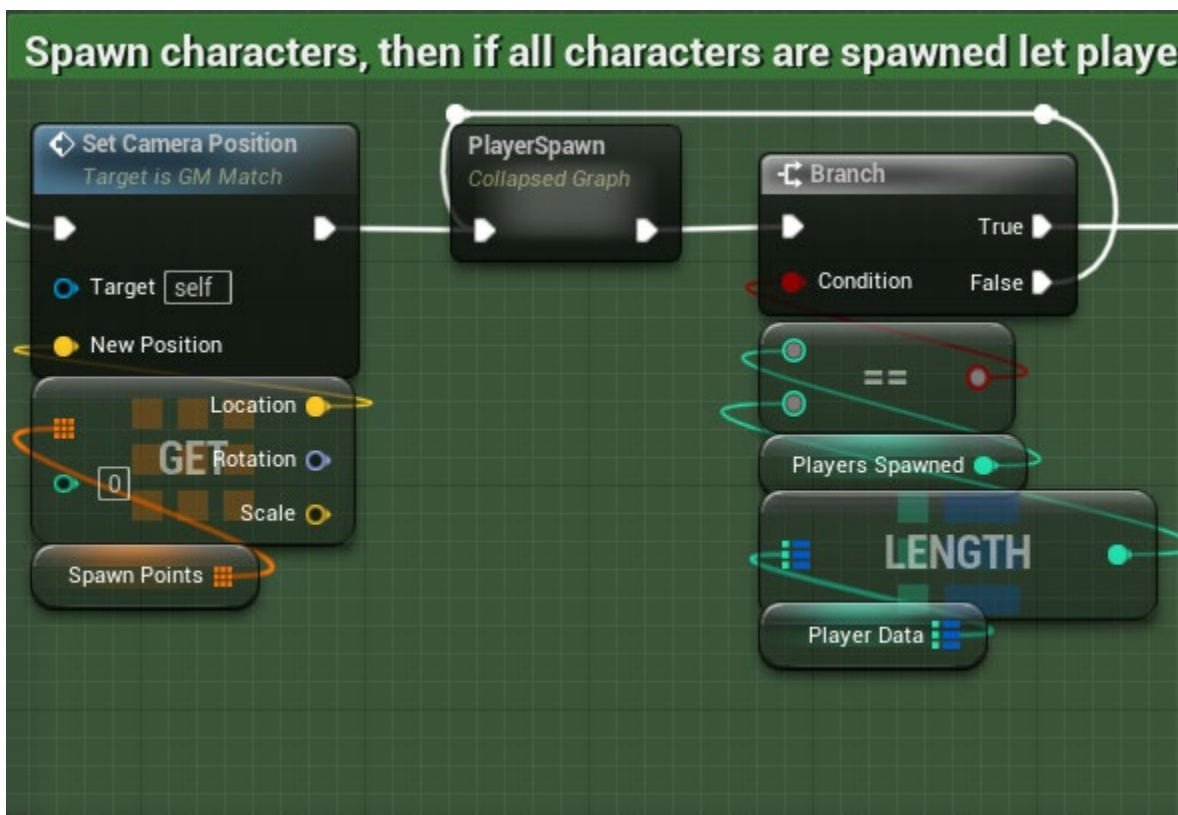


Over in *BeginPlay*, after the camera is set as view target as mentioned above, if the player count is equal to the number of players in the *PlayerData* map, then we proceed with

spawning the characters. First, we find all the spawn points on the map.

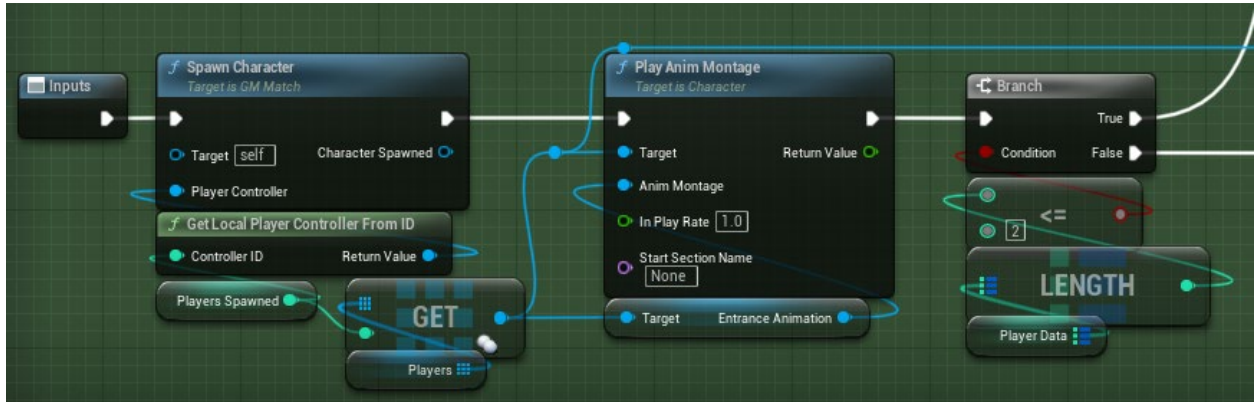


Once those are found, we set the Camera Position to the first spawn point in preparation for the first character spawning. After each character spawns and their entrance animation plays, we check to see if all characters have been spawned yet. If they haven't, we recursively call the *PlayerSpawn* graph again. I would have used a loop here, but the *PlayerSpawn* graph contains a timeline and delays, which can't be used in a loop, so I made my own.

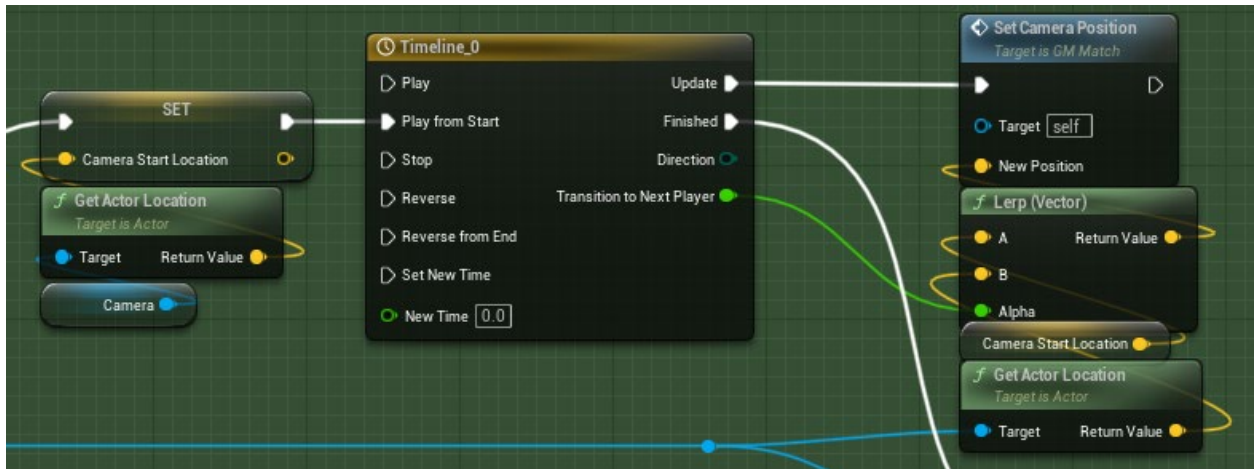


In the *PlayerSpawn* graph, the character is spawned (this uses the same function Sean made previously, so I won't go into here), then their Entrance Animation montage is set to

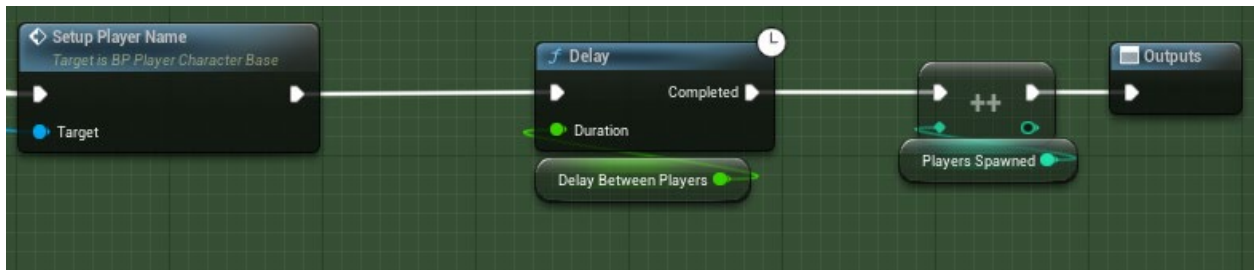
play. We then check if the number of players is 2 or less, as the camera positioning is handled differently depending on how many players there are.



If there are two or less players, the camera's position is logged, and then a timeline is used to move the camera from the first character to the second character using a lerp. If there are more than two players, this section is skipped.

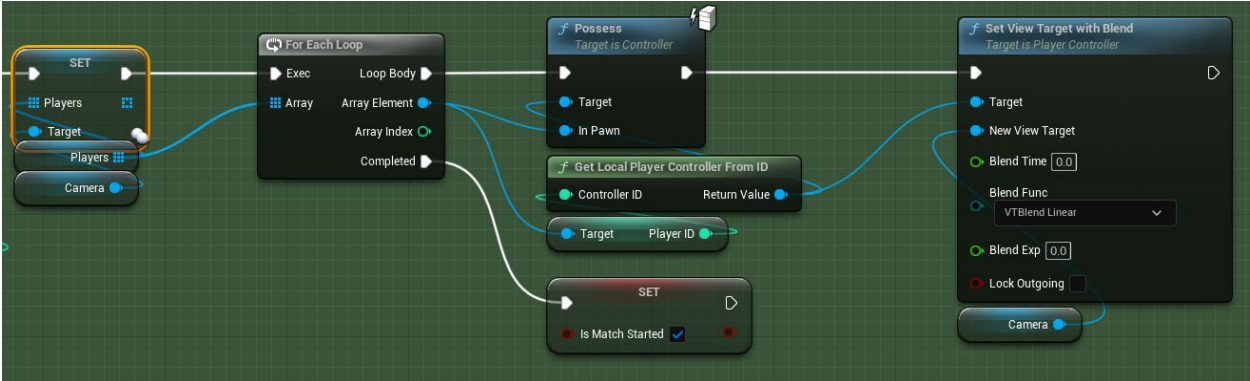


After the camera pan is done (or the animation of the first character is triggered), the character's name is Setup, and a small delay is triggered so that the next character's animation doesn't start immediately. The counter for how many characters are spawned then gets incremented, so we know when to break out of the loop.



Once all the characters are spawned and their animations have played, the camera is given reference to all the characters, and the players are able to possess their character, and the camera is set as their view target again as it would default to the first-person view for some reason.

Once this is finished, the Match is set to start, which triggers the timer.



Overall, these systems required a lot of work, troubleshooting and iteration to get it to its current state. There is still a lot of room for improvement, particularly for the stage entrance animations, but the more difficult and frustrating parts are out of the way.