Duncan Carroll

Beautiful Corridor - Unreal 4 Art Pipeline

Dynamically Importing Static Meshes

The main functionality of my corridor project is that ability to dynamically import static meshes into a level in Unreal 4. By that I mean that my project can read any given file path and load that asset into the game without having to recompile or rebuild the project. In its current state my project has a text file that it reads from where you can type in different parameters that affect an actor in the game. The text file the project reads from is called “info.txt” and is located in the Content directory of the project. The first line of the file is a number that the actor’s world scale will be set to upon creation. The second line of the file is a number that the actor’s rotation around the x-axis will be set to. And the third line of the file is the path to the static mesh you want to be loaded and attached to the actor. So by changing the file path and other parameters in the text file you can load different models onto the same actor at runtime. This project uses a text file to change the parameter so it can show the functionality on its own. It can be easily expanded later to take parameters from other classes in the game that will hold all the information about what pokemon are fighting and what models need to be loaded.

Implementation

This functionality is implemented with a C++ class in Unreal 4 that inherits from the Actor class and is called APokemon. If you place one of these objects in the level then it will read from the file and load the static mesh according to the arguments given. The APokemon object before runtime doesn’t have a static mesh component on it. The static mesh component is required to hold a static mesh and associate it with an actor so it can be manipulated in the world space. The APokemon object is written in C++ and all the code that loads and manipulate the static mesh is controlled through the cpp and h files associated with it. In the object's constructor a default static mesh component is created and attached. In the BeginPlay function the static mesh components mobility property is set to moveable to tell the engine that the mesh will be modified. This property can be set to static to increase loading times for the mesh but disallows movement like scale, rotation, etc from being changed. After the mesh component is set to moveable the class currently reads through the text file line by line and saves the information into an array. That information is then used to create a new static mesh from the path given and set the object’s scale and rotation. To integrate this into the full project I can simply have functions that are given parameters to load from instead of loading the arguments from the text file. Whatever class holds information about what pokemon are needed and spawns them will be able to call these functions as the pokemon are created. This way the same loading function on the APokemon class can be used to load whatever model is needed for the game.