C++ vs Blueprints Breakdown

This project has several files that are hard to interpret. The linkage between c++ and unreal is rather unnatural when looking into each induvial part.

Briefly, I will explain where each portion of the project lives or is created.

Map: The map lives within the engine and wasn't built using code, only the engine tools such as landscape, foliage, and free mesh/texture assets were used to design the map.

There is a level map blueprint where I added collision code for when the player enters the water and/or the garage. This does call functions from the c++ class to things such as stop sound, get distance, etc.

Mower Physics: All mower physics are built inside of the AKeepMowingALawnCharacter.cpp. It handles the Timer, Distance, Position Calculations, Collision, and so on.

The slight caveat is how the mesh that the mower is wearing is attached within an engine blueprint that extends AKeepMowingALawnCharacter. The blueprint is named ThirdPersonCharacter because the project started out as a thirdperson default project. Almost all default code has been stripped out or repurposed, but the blueprint name remains the same.

Mower Blades: Mower blades were designed as a blueprint within the ThirdPersonCharacter blueprint. At the time of creating this, I didn’t know how to do LineTrace by channel within c++. I feel as though the process is actually more readable within blueprints, but if I did it again, I would make it in c++ to make its size a bit more manageable instead of a rather large blueprint.

Heads up Display (HUD): Both HUDS are widgets designed in the engine, with functions in blueprints. Their blueprints access functions within c++ to get distance, time, etc.

Within the AKeepMowingALawnGameMode class I made code that would allow me to attach a HUD to the player via c++ and the engine.

The MowerHUD is attached using this method.

The EndgameHUD is attached using blueprints in the level blueprint.

Sounds: All mower sounds are inserted into the engine as files, and has engine options that loop the sound, but manipulation of the sound is run purely through c++. Occasionally, the blueprints will call c++ functions to stop sound.