Overview

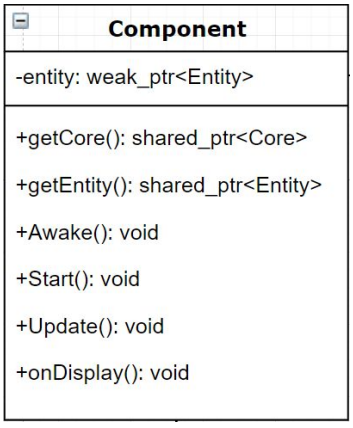
The TEFGAS Engine(Toby's Engine For Games And Stuff) will have a fully integrated ECS system with Resource Loading. It will also be able to play sound, take input from the keyboard and mouse and also be able to render in PBR.

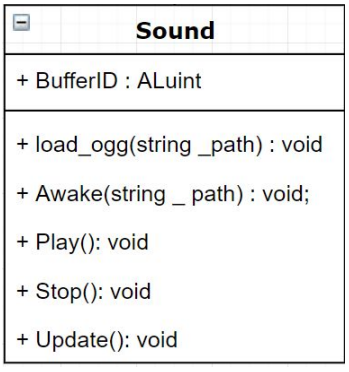
Looking into the syntax of game engines I decided to go with one more similar to unity than unreal engine as it seems the most user friendly system.

The Engine is entirely independent of any game created in it only requiring to be included as a library and is accessed through a single header file Engine.h. This allows for the engine to be tested independently of the games made in it for reliability meaning when developing a game in the engine you can be sure you don't accidentally break some low level engine code.

The ECS system will comprise of Entities that contain Components . These Components can be either Built into the Engine or created by the end user as long as it extends the Component class. Transform , Mesh and Sound are a few examples of what could be a built in component.

The Engine will revolve around a Core class which regulates everything this is what holds the main game / update loop. The core will trigger the Entities to run the update function on their components after which it will call the draw function. A Component will have a few defined Functions that it can extend from as defined below :

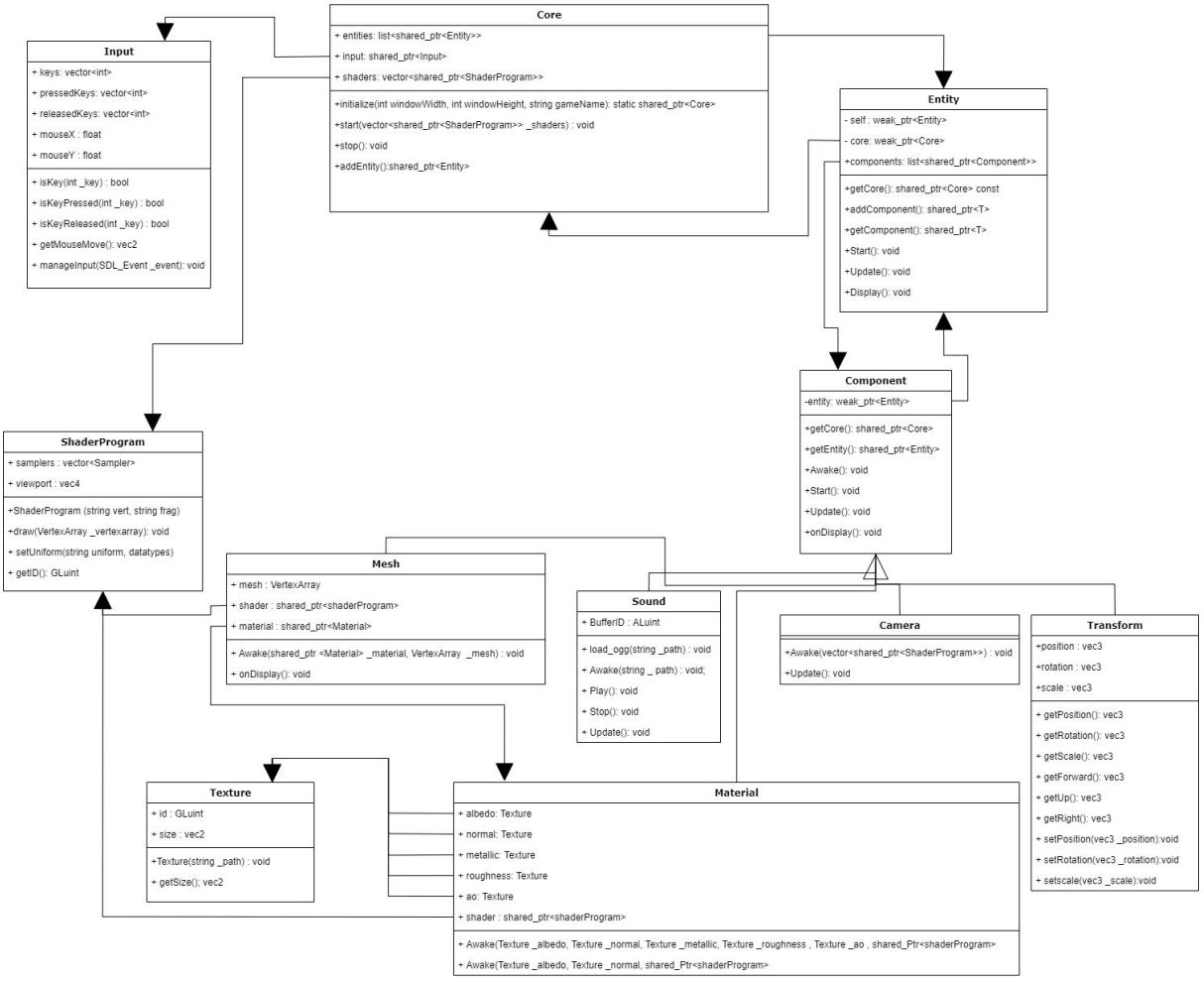




An Example of a Class that builds upon Component would be Sound. It uses the Awake() and Update() from component but also has its own Play() , Stop() and loadOgg() functions.

To make the system more user friendly I created a Material system where you can define a material that could be used on several objects it also lends itself to future development of a more user friendly Graphical Editor for creating game worlds and an improved shader pipeline. The material system also accounts for the use of both PBR and non PBR workflows

The full design of the engine can be seen below:



Overall I think my design is quite effective although after having developed it there are some things that I would like to change going forward such as defining the shaders in a similar way to how you define a shader by calling a function like Core->addShader(.....) rather than making the user make a shared pointer and then put it in a vector that they have to make themselves. I would also like to develop a Scene system allowing for designing the game world in a simple editor and being able to save and load it. The final thing that I would like to add is Tri-Box collision so that I would be able to make FPS style games with trigger volumes and enemies. Everything I have mentioned that I would like to add would be easy to implement due to the open design of the engine.