Got project building with Lua and SFML

An important first step was to get Lua, SFML and Box2D building together within a single cpp project. Once the project was building correctly, the next step was to create some example classes and a simple main program loop to create a test bed for future work.

Created example class in C++ that works in SFML Designed and created first classes of the entity system Created drawable and point classes

Creating simple classes that showed that SFML works allowed for the creating of the first classes as shown in the class diagram. The drawable and point classes were then created as well as the surrounding framework to allow for the storing and retrieval of the objects from both Lua and in cpp tests.

The point class is purely virtual and cannot be instantiated and provides a think function that will be overridden by all sub classes that are required to do work every cycle of the main loop.

The drawable class provides the ability to draw a sprite at a set position and inherits from the point class so that it can be stored in the same vector as other entities and can then be casted as is required.

Designed and created resource manager

Creating the resource manager proved to be challenging but ultimately very satisfying when it worked correctly. It allows for the end user of the game engine to not have to bother with resource management as the core engine will handle it, all resources are loaded from file for the first time but are otherwise reused by passing a smart pointer to the resource, when the resource is no longer in use anywhere it is unloaded from the resource manager.

Expanded Lua integration with the use of the init and think(dt) functions as well as keyboard and mouse

Up to this point there was no Lua integration at all, to kick off the game engines API, initialisation and a think function were provided, this allowed the cpp environment to call the init function when the lua world needed to be created and called the think function every iteration of the main loop.

User input is a requirement too so getting when mouse buttons were pressed and released and when keyboard keys were pressed and released were all provided to the Lua API in the form of hooks that were called as and when the events happened. Getting the current mouse position as well as getting the state of all buttons and keys were also provided so the end user of the engine can decide their preferred method of handling input within their game.

input/setting and getting some information from the window Converted the resource manager to be generic and to use smart pointers Added sound to C++ then to Lua Created animation system design and began implementation Created Animation class and created animatable Got animation and animatable working in Lua Did some work on the gamestate system Created ECS design and started conversion to ECS Designed and created the camera system Created animation component Built on Linux Finished conversion to ECS and fixed in Lua Combined the drawable and animatable components into a single drawable component Added b2d Created physics component and added it to Lua Command line args added created pong and realised circle physbox was a requirement created snake View system now works in Lua created fall Documentation Other people use the engine see what they say