Reflection

When starting out with the game engine, I was able to implement some really important systems that allowed me to have the game objects all stem from a single class, then I was able to implement functions that check for any possible collisions between objects so that later these functions could be used for any uses that the user wished for. I also created an ObjectManager class that served as the thing that managed all the objects inside the game world. This was really neat as I was able to use this as a way to have my characters or other objects "die" while still maintaining the objects in the code memory to then have them respawn as I have demonstrated in the Death and Respawn Event implementations in the platformer game. I think that the design was very successful as I was able to use this throughout the semester and only had to make additions based on the requirements of further homeworks. This system of the engine was left untouched for the second game except for some changes to constructing the objects in the games[1]. One thing that I would like change if I were to do this over is to be able to make this completely generalizable so that when these objects need to be created, all information is able to be gathered from the arguments themselves and the objects could be dynamically created (idk if that's the right way to put it) but I'm not super sure how to do that in C++, I'm thinking that could be through some kind of list object to hold parameters similar to the variants that were discussed when making events (which I did not use cause they scared me).

The timeline that I created has been relatively untouched since HW 2 (though I'm pretty sure I've messed that up but like, it still makes a functional game, it is just really weird).

I am relatively proud of the server-client system that I have built for this engine as this was my first time making anything like this (we didn't even go over this stuff yet in 246 at that time). I ended with a server-centric approach to everything as it was the easiest to implement. However, if I could make it better, creating a server-client system that allows for the client to do a lot of the calculations on its side would definitely be favorable. However, this system was able to be relatively untouched when creating the second game as the ObjectManager is what allowed for different types of GameObjects to be sent to the client and constructed to be displayed for the player.

The Game Object model that I developed seems very finicky and I think needs a lot of fine tuning, in HW 1 and 2 I had an Object Centric model that had some inheritance and for what it was worth, I think it was a pretty good system, though since I've been working with this type of model for other non-game programs in other object-oriented programs, I wanted to try something different. I ended up trying to go with a Component model where the components were structs that held more fields for the GameObject to access, I also wanted to give the components methods but giving those methods access to the object was finicky so I only did it for one of the components. However, I feel like I at least understood the idea behind it and I was still able to utilize this model for my second game.

I am very happy about the Event System that was designed as a part of HW 4. I know a lot of ideas for the implementation were given in class but I still felt very proud of what the system was able to accomplish and even though I only made the bare minimum of events in the first game, when I was making the second game, I was able to utilize this system for a lot of features and communication and see why events are extremely useful for just being able to relay a lot of information and store that for a more centralized way of handling things that happen in the game. When creating my second game I made a decent amount of new events and handlers but the EventManager was left completely untouched.

The overall differential in my two games (I had chosen my HW 4 game instead of the game after chords and scripts as I was not confident in my engine there to be able to create a new game as they were extremely difficult! More on that in the writeup most likely. The diff in lines were 603 [2] which I think is mostly the implementation of my game specific objects in features as I just compared my entire game files. Even then, when dividing it by the number of lines in my second game, I only had a 5.45% difference which I'd argue is a win, especially if I say that my significance level is 5.46%. Sorry about the statistics joke, I was also taking that this semester. But yeah, overall I'm very proud of the engine that I have built and the stuff that I have learned and implemented in this class.

Things that I would like to change about my design if I could start again in the future is one, I'd like to have my things completely extensible (I know that's kinda really generic and like "duh" but ya know). I'd also like to have a sort of UI, nothing crazy, just like an in terminal UI that is able to take in commands to be able to create Objects or Events or Handlers or anything else that the user can then just edit and it will be able to work just fine. Another thing that kinda falls into line with this is that if the user isn't versed in the native language of C++ then this system could create scripts for whatever is created with all necessary components visible to the scripts, and any additional/optional components could be added through the terminal UI. The ideal scenario would be that the user would be able to create an entire game in the engine using only scripts which I feel like definitely needs some intentionality behind it when just starting to create an engine as linking that into what we currently had was more difficult than I had thought. I've already mentioned this but I'd also like to understand the server-client system a bit more so that this UI and scripts could be able to access both the server and client when necessary so that everything will be synced. I would also like for my GameObject model to be more understandable, and create more components for it to allow for, again, more interactions in the game world between objects that would allow for many other games to be created in the engine.

I still really enjoy my design for the Event and Object Managers and like the ways that they are able to access a lot of really useful information and relay those to the entire engine. With that I would like to just add more functionality, especially the ObjectManager and the functionality needed to allow for more interactions between the GameObjects in a really centralized area.

Overall, I really enjoyed what I was able to produce in the class and have learned a lot about what all needs to go into an engine in order for multitudes of games to be made. I'm not sure if I'd ever create my own personalized engine (at least not until I have a great grasp on C++ and all the ins and outs of different libraries, especially v8) but I feel like if I were to work on creating tools for an existing engine, I feel like I would have some ability to understand what I need to look out for when creating tools that need to be extensible.

Appendix

[1]

```
smoore@GamePuter:~/csc481$ diff ./'Events HW 4'/Engine/objects/manager/objectManager.cpp ./2Game/Engine/obje
cts/manager/objectManager.cpp
112c112
<             DeathBox *deathBox = new DeathBox(size);
---
>             DeathBox *deathBox = new DeathBox(size, true);
122c122,129
<         } else {
---
>         } else if (type.compare("Enemy") == 0) {
>             sf::Vector2f size(to_parse["Size"][0], to_parse["Size"][1]);
>             Enemy *enemy = new Enemy(size);
>
>             parsed = enemy;
>             addObject(parsed);
>         }
>         else {
smoore@GamePuter:~/csc481$
```

[2]

```
smoore@GamePuter:~/csc481$ diff -r ./'Events HW 4'/ ./2Game/ | grep "^>" | wc -l
603
```

[3]

```
smoore@GamePuter:~/csc481$ find ./2Game/ -type f -print0 | xargs -0 wc -l | tail -n 1
  11069 total
```