Reflection

My engine went through many changes throughout the course of this semester. In the beginning I tried to implement a design that would allow for future changes and additions to my engine without needing to be refactored too much, but since I was unaware of what the future requirements would be I was only able to plan for so much. I do think I was successful in my overall design though since some of the implementations, such as static platforms, required very little changes in later revisions.

As the requirements for the engine grew throughout the semester, I was forced to refactor my main function constantly, but I was pleased with how little I needed to refactor most of the classes and other functions I implemented. Sometimes a new requirement would force some refactoring, such as choosing an object-centric game object model. To implement this, I needed to create a new class (GameItem) that all my other game objects would inherit from. I needed to do some refactoring to each object model to inherit from this new class, but it was very simple to do with the way I had previously designed and implemented my game objects.

One of the requirements that caused the most refactoring was the event system. This is due to the fact that each item that could have events associated with it needed a new onEvent function to handle whatever events that were subscribed to that item. It required the new function, but sometimes it also meant that the item's class also needed access to another item that it previously did not have. For example, the deathzone object needed access to the event manager itself so that it could create a new spawn event after the death was handled. So, I needed to add a pointer to the event manager in the class so it would have easy access to it and be able to create/raise new spawn events.

Overall though I was happy with the amount of code reuse I was able to get out of my engine during development, considering I did not know future requirements in advance. I only ever needed to make changes to implement the new functions that the requirements dictated I add, and I never needed to start over or scrap a whole design implementation I had. I simply had to add on to what I had built up until that point.

Having said all of that, I never quite got networking working properly and by homework 3 I dropped it entirely due to spending too much time trying to get it to work, only to fail and put me behind on other parts of the requirements. The same thing happened for the scripting. After spending hours on it with no success, I eventually scrapped it because submitting without it was better than no submission at all.

Since I was unsuccessful with getting scripting to work, when I made my second game, I was forced to change a portion of my main function. Despite this, I managed to make a second game, in a completely different style, in my engine. I needed to change about 16.76% of the code from my first game to complete it. This means that 83.24% of my code was the same for

both games. I accomplished this by just making changes inside the main function. If I had been able to get scripting working, I think I would have had to change very little of my actual code to get the second game to work.

Because I knew I would have to change actual code to make my second game, I tried to reuse as much functionality of my objects as I could. Although I have the in-game objects named things like Platform and FloatingPlatform, I reused those items for other purposes. I used the floating platforms as barriers on the top and bottom of the screen, since I already had collision in place for them, and I set them to not move. I then used the platform class to create white squares to be used as asteroids. Next, I created deathzones that were the same size and position of each asteroid I made. I already had a texture/image in use for the player character, so I changed the image to a pixelized spaceship that I made and then I set the background to a darker blue to resemble space better. I also already had shifting of objects in place, due to the side boundaries, so I used that same mechanic to move the asteroids across the screen.

Since I didn't make a proper "level complete" scenario for the first game, I chose to add something for the second game so the user knew it was over. Once the player makes it past all of the asteroids, the game pauses and the screen goes black so that the user knows the game is complete, rather than just flying through nothing forever until the user stops the program. This did add more changes in code than I needed though, perhaps, but I thought it was needed to alleviate any confusion of what was happening in the game.

Despite needing to change the actual code of my engine more than I would have if I had scripting working, I was happy with how I was able to reuse my objects for other things without making changes to them. I was also pleased with how I only needed to make those changes in the main function, without needing to refactor other classes or objects I had previously designed and implemented. Although the class is at an end, I still plan on implementing scripting on my own so that I can say that I did implement it. I am also curious how little of the code I would need to change once I have that implemented.

If I could do this all over again, with future requirements remaining unknown (as they were in this class), I don't know if I would have done anything differently. I would have liked to get networking and scripting working of course, but the overall design of my engine worked and allowed for a good amount of code reuse. If I was to do this all over again but knowing what the future requirements would be, as I now know what they all are, I would have definitely done some things differently. I would have been able to better plan out what was coming and how I could design for it without as much refactoring as I did, even though I was happy with how I did it at the time. I could have planned for things like passing a mutex into the platform, or other in game items that I would later use. I also could have probably better designed my object-centric game model even earlier. Although I already was working towards that model, I could have easily set up my objects back in Homework 1 to all inherit from a GameItem class right from the beginning, rather than needing to refactor to implement it later.

Since we were not told what the future requirements were going to be in advance though, I think that my strategy was mostly successful and I would likely do it in a very similar manner if I was starting the class all over again, not knowing what the requirements for future homework's' would be.