Andrew Weller
109128949
CSCI 1300

Monster Eyes - Write Up

How did you prepare for the Project?

I prepared for this project mostly by brainstorming ideas about what kind of game I wanted to make. Once I decided that I wanted to make an rpg I then figured out how to make a repeatable structure for my game. I thought it would be cool to have a game that is structurally the same every time you play it, but the names of people and items, the stats of items and damage are all randomized. Sort of like a big game of mad-libs but I'm putting random elements into something more complex like a game. Adding the random names was the easiest part of the game for me. I created a class called NameGenerator which contains a vector of strings, which is filled from a text file. Then when I am loading the game I assign random names to characters and buildings and enemies and many other things. For the actual game I tried thinking of many different ways to structure my game, I actually had to meet twice with my TA before we both felt like I had a solid concept in my mind. Eventually I decided on a general structure for the game. The basic rundown is this:

I have a **Game** class which contains:
- A **player** object - which inherits most of its structure from a Character class.
- An array of **Zone** objects
- Variables to keep track of:
  - The current **Zone** the player is in.
  - The current **Room** of the **Zone** the **player** is in.
  - The current **Dungeon** the **player** is in.

Zones serve as a container class to hold pointers to **Room** and **Dungeon** objects.
Rooms:
- Contain a pointer to a **character** (sometimes the character needs to be modified) -
- Either an npc for the **player** to talk to, or an enemy for the **player** to fight

Dungeons:
- Contain an array of **Room** pointers.
- These rooms contain a pointer to a **Character**

The game loop gets information from the Zones rooms and dungeons in order for the player to interact with them. (i.e. hurting an enemy or giving them gold.
The Rooms and Dungeons from the Zones send back information to the game loop to interact with the player (i.e. damaging the player or giving them an item, giving them gold, etc).

Did you write a Code Skeleton? Was it useful? How?

I did write a code skeleton. I didn't really fully flesh it out in c++ but I had a structure written out in paper for the program. It was very useful. It wasn't very fun, I think I would have had more fun with the project if I could just start coding and not worry about that stuff. Even though it wasn't fun it made my project a lot more sensible and it made me think about problems that I would have in the code later down the line. I have made projects before where I started without a skeleton and it was very frustrating during the finishing parts of the project to add things to the code or change it. With this project, the skeleton forced me to plan ahead and to think about how I will use classes and function way into the future. With the structure I have made for this project, I feel like I could expand it a lot more and add many things to the game without it getting super complicated. It really made my project much more scalable.

Reflect on how you could have done better, or how you could have completed the project faster or more efficiently

Even though the skeleton made the project easier to do once I got running, I still ran into many problems. My project is very scalable, but there is a problem with the scalability. If I want to create a different item for the game, I need to create a new class for it and have it inherit the general . So if my game were to have 100s of different kinds of items in it. I would have a huge amount of class files. I think there might be a way for me to only have one item class which can contain a reference to a function to perform on a player which would be called when the item is interacted with. This way I would only need to have one class for all items. I'm not sure though because I don't know that much about things like that. I already had to learn a lot of new stuff on my own in order to complete this project the way I wanted to.

On the subject of Items, I had some very superfluous classes that served as an intermediary between the Item class and the sub class. For instance, I have an armor class which inherits from an abstract class Equippable, which inherits Item. I think I could have made the game without the equippable class and just had armor and weapons as their own classes. The main reason for doing this was because I wanted see if I could store armor and weapons as equippable pointers and have them equipped in the same fashion even though they do different things. It worked and it was cool, but I don't know if it was necessary.

My main game class is like 1300 lines of code, I think I could refactor some of that and condense some functions into a different utility class. But it works.

Other than that the structure of Zones -> Rooms -> Dungeons worked extremely well because I planned that out very intently at the start. I think I should have spent more time with things like Items and also the game logic.

Paragraph

I had many false starts and wrong turns during the construction of this project.

One of the bigges 'false starts' I had was in the overall structure for my game. Originally I wanted to have the Zone object contain the player, and it to interact with the player directly. I realized it was much easier to have a Game class that interacted with both the player and the zone at the same time. Even though it makes more sense logically to have a player reside in the zone they are playing in, the other way is much easier to program. It still creates the illusion to the player that they are "inside" a zone, but without that hassle.

I also had many hurdles to get over which resulted in plenty of learning opportunities.

The first hurdle to get over was I wanted to user pointers but I did not know it yet. This started out was wanting to create an array of equippable objects for characters which could contain both armor and weapons, which are equipables of sub-classes. The first language that I learned about object oriented programming in was Java. In Java all objects are passed as reference to functions, but in c++ they are passed as values by default. So I need to learn how to use pointers in order to make this work. Eventually I learned how to use pointers and virtual functions to make this work. I had an array of equippable objects which could be equipped and unequipped from the player without any kind of casting required.

I then got stuck because I didn't know how the stack and heap worked. Most of my game structure relied on pointers and storing addresses of objects in vectors and arrays. So I ran into a problem where when I was loading the game from a text file, I was creating items to be placed in the players inventory. The players inventory actually just contains a list of Item pointers so that I could store multiple types of items in it though. So I would instantiate an Item and push the reference for that object into the player's inventory. The problem was that, I was creating the object on the stack, so I would create this object on the stack, push its address to the players inventory and then go back to the top of a loop. This would destroy the object on the stack because it went out of scope. So I was creating an object in the same place on the stack and then pushing that address multiple times to the player's inventory. This resulted in a bunch of Item pointers pointing to the same address which was null.

Those are just some specific examples of things I had to learn myself and overcome during this project. I learned a lot about how computers store data and how c++ and other programming languages work as well. This project was a fantastic learning experience for me.