Christian Fin

12/2/2018

CSCI 1300

CSCI Final Project Reflection

1. **How did you prepare for the project?**

Before beginning my code skeleton, I made sure to read through the project requirements a couple of times. I made sure I knew what elements had to be included in my code (classes, loops, file streams etc.) and also what my program had to be capable of (travelling, hunting, misfortunes etc.). I also played through the online Oregon Trail emulator a few times to get a feel for what the user experience during the game is like. Only after I knew that I had a firm grasp on the project parameters did I actually begin thinking about how I was going to approach writing the program.

2. **Did you write a code skeleton? Was it useful?**

I did write a fairly brief code skeleton before beginning the project in earnest. Even though I ended up altering a lot of what I started with in the skeleton, it was still helpful in that it forced me to think about my program's overall architecture before diving into the actual implementation of game mechanics. Creating useful classes and proper program structure makes writing, debugging and understanding the rest of the program much easier.

3. **Reflect on how you could have done better, or how you could have completed the project more efficiently.**

Overall, I think that I was efficient in creating this program. I never spent more than a couple minutes hung up on a problem before I found the solution and by constantly compiling and running my code, I was able to catch my errors before they became too hard to fix. One area where I think I could have done better was in input validation. Every time my program asked for user input, I wrote an if statement that checked for invalid inputs. I think a more efficient option would have been to create a devoted function or even a devoted class that could be called whenever input needed to be validated. But creating the if statements wasn't too much of a time sink either way.

4. **Did you have any false starts, or begin down a path only to have to turn back when figuring out the strategy/algorithm for your Final Project program?**

Fortunately, I did not really have this issue. After a few minutes of brainstorming, I was confident that the code structure I had set up was capable of handling all the required game functions. When I was first creating my code skeleton, I think I was overcomplicating things for myself. I was planning on having about 8 classes, all of which would be nested within one master class. Additionally, I was planning on making virtually no use of my driver file. However, I quickly realized that this approach was overly complicated and would ultimately probably not succeed. Eventually I realized that just about all of the game could be handled using the driver file and just two other classes. A game object which would handle all game functions related to milestones and travelling and then a store object which would handle all game functions related to money and supplies. My other two classes, player and misfortunes, were quite limited in capability and only helped with a few elements of game execution. I then used the driver function to allow the classes to interact and to handle game functions where capabilities from two or more objects are required.