Ben Morledge-Hampton

Final Write-up

This project ended up being a lot more work than expected, but it was still a rewarding experience regardless. By far the most difficult part of the project was structuring the events that drove the game to completion. As mentioned in my proposal, I wanted to create classes that allowed for a high degree of variability in events, which was easier said than done. Before even that issue however, I ran into the problem of passing around resources from the stockpile as variables. This problem was solved by creating an enumerator for each of the resources and then using that to access details about the resources in EnumMaps. This way, events could use these enumerators to specify which resources to alter or retrieve! After solving this problem, the issue of structuring events was next. As part of the solution, two classes, Result and Conditional were created. The results handled the changes to the stockpiles and the conditionals determined whether the event succeeded or failed based on given resource requirements. Both of these classes utilized the aforementioned EnumMaps to specify Resources. Finally, to load in the events, a function to parse text into events was created and the events’ specifications were put into a text file to keep the java files relatively short!

That was the bulk of the data modeling. Besides the events, and resources, the buildings were the other source of modeling, but they were relatively easy. They were created directly in the code without the help of a text parser. Once they were done, it was on to the View and Controller! These were created though the children of the parent Menu class, which was a child of the JComponent class. This class contained some functions for displaying text or buttons at a given position. These functions were created to scale depending on the user’s resolution! The individual children simply served to create views and controllers for the different types of data. (Stockpile, Events, Buildings.) once these were completed, a Manager class which extended from JFram served to keep the Model interacting with the View and Controller. It all worked quite smoothly in the end!

If I could go back and do some things differently, I would consider breaking up my code into more classes. The ones that I had tended to grow pretty large. The event manager in particular could have used some splitting up. Perhaps Things could have been reworked to avoid the plethora of EnumMaps currently being used as well. With more time, I would have also liked to add more events to the game and had more time to test and balance the gameplay. Graphics would have been nice to implement as well. Also, it would have been neat to experiment with XML in creating my Events text file, but again, time simply did not permit it… As it stands, however, I am quite pleased with the game’s current state!

In the end, the list of key concepts used are these: Basic class structure and syntax were used throughout the project, obviously. The structure of code follows the basic Model View Controller philosophy. Exception handling is used when reading from the text file to generate events. That same code also utilized File I/O. Java Swing components were used in the Menu classes to create the UI’s. Multi-threading was never utilized as it felt unnecessary with the turn based structure of the game. Android programing was also left out because I do not like Android Studios.   
 That’s all!