Final Design Implementation - Retrospective

1 Team Members:

Kevin Masson, Alex Hughes, Cale Koi, Musaddique Khondoker Mohammed

2 What Went Well?

- The abstraction of the items made it very useful to make containers of items without needing to make containers of specific items, so we were able to specify the use of every item.
- The use of branches was extremely useful for not wasting time and preventing merge conflicts. We had an item idea we tried to implement and just couldn't get to work, but since it was on a branch it never affected our build so we just started from the working build and tried again with a different approach.
- Our textstream function of streaming dialogue from files was very useful as it cleaned up our dialogue file, making it much more readable, and contained all our dialogue in one .txt file that relied on keywords to know what to output
- The help function I feel turned out really well, the way that it lists inventory, items, and the available locations was very easy to code due to the way we laid out our functions
- The use of maps for the locations and usable items was very helpful and easy to use for interface, as it was an easy translation of an inputted string to getting a pointer to a needed class. It did cause a little bit of repetition for creating objects, but it became much easier to take user inputs and have a stable container of pointers to pull from.
- A map was created to visualize the layout of the game which helped to keep everything in order and it allowed for faster testing because we could play through the game at a faster rate to a specific area to see if things were working.

3 What Did Not Go Well?

- Our testing for the basic implementation relied on us using a very specific method of using items and players, and as we refined the build the tests kept breaking. Not due to the functionality not working but the tests expecting a different method. Doing more thought of how we are going to get all the info needed for classes and what we want to specifically return would have made a lot less headache for testing.
- Our items relied on a lot of the game world knowledge to function, and since we did not want to have every item contain a lot of the games info, and not require a lot of parameters we ended up making the player a bit of a superclass unintentionally that contained a lot of info needed but never itself really needed. If we were to refactor our design this is something I would have liked to have thought through more, but it was too easy of a solution to our problem to our items for us to be willing to rework at a late stage.
- The way we did our characters and dialogue there wasn't really any way for us to store info on the state of the characters, so to make our characters and locations change dialogue was something I wanted to implement but ran out of time for unfortunately, so it seems a bit lazier than I wanted it to be, since the game world updates but the world as far as the player can see does not aside from single use dialogues and the help function. More characters would have been nice, but our design was already getting a bit bloated and we would have needed to rethink a lot of puzzles.
- Due to the workload of all the members across all classes, the final weeks of the project did not have too much collaborative work or meetings other than discord updates, which may have lead to a better design of things with more feedback and better group understandings of the interfaces for creating location and items (and thus quicker work time).