Alex Swider
June 12, 2018
Final Project

Reflection Document

In tackling this final project, I read the description and took a step back for a while and just gave myself time to think about planning and executing this subject. This time, I did not use pen and paper really to plan it out. Rather, I looked at different portions of the assignment description and tackled coding each of them one at a time.

First, I had to figure out what game design and story I wanted to go with. I decided upon the Christmas gift search as it resonated with my childhood and I thought it could be an interesting game to design for this assignment. With this theme, I was able to implement a number of different interactions throughout the short game. With this theme in mind, I began my coding directly.

The first part of the code I wanted to tackle was designing all the classes and subclasses for the spaces. I did not even begin the main function before doing these, and just jotted down functions I thought I would need throughout the game (being able to add more later if needed). Most importantly, I added the “viewEvent” and get name functions. I knew these would be the most important for the purposes of the program. With each different space subclass, I wanted to think of some interaction that could make the room unique in the program, resulting in 6 unique rooms with a variety of interactions. After designing these, I moved on to the next portion.

Next, I began tackling the main function. However, instead of focusing on the main function itself, I began focusing on organizing the spaces for the rooms and initializing them. I used code similar to what I did with project 4 and was able to execute this quickly. However, I would later find out this strategy was more difficult for finding how to deallocate memory (as it was for project 4).

Finally, with all these different pieces in place, I tackled designing the game itself and putting it in a form that was able to be compiled .Once I had it in this state, I was able to test the different parts of my program to ensure they work. With each component that worked, I added another component, paying special attention to the ones that were listed in the project description. Before I knew it, I was able to implement all the different aspects into an interesting short game and have it work successfully.

The final part of the project was figuring out how to free all the memory successfully. Unfortunately, I was unable to figure out how to get rid of the memory leaks present in the program (I had the same problem in project 4). If you have any ideas as to how I can rid my program of leaks it would be greatly appreciated.

Test Case	Input Value	Driver Functions	Expected Outcome	Observed Outcome
Inputs too low	Inputs < 0	main()	Prompt user input is invalid and to redo the input	User was prompted to enter values that were in proper range.
Inputs in proper range	Input is a menu option	main()	Functions take in the values and run the program accordingly	Proper inputs accepted and program functioned accordingly.
Inputs too high	Inputs > menu options	main()	Functions see values too high and prompt for input again	Inputs that were above the range were discarded and reentry was prompted.
Input is non-integer value (character)	Inputs are letters	main()	Function cancels this input and prompts for re-entry	Characters did not crash my program through input validation techniques.

With these basic tests in place (primarily focused on input and output regulation), I was able to test the different aspects of my program itself, which focused on the events in each of the spaces (and ensuring events did not repeat if already completed i.e.: killing the spider).

Event	Does it work?	Does it repeat?
YourRoom - lay down	Yes	Yes (intended)
ParentsRoom - turn on light, find bat	Yes, yes	Yes (intended), no (intended)
LivingRoom - unplug phone, smack babysitter	Yes, Yes	No, no
Basement - kill spider (with bat), pick up key	Yes, Yes	No, no
Kitchen - eat cookies	yes	no
Shed - open shed	yes	No (game ends)

After testing these various functions and ensuring they were working, I was satisfied with my program's performance. At this point, I strictly focused on the aesthetic layout of my code, labeling and noting where necessary, as well as focusing my efforts on finding the memory leaks in my program. I know where the leaks are (when I am dynamically allocating my subclasses), but I am unfortunately unable to figure out how to successfully eliminate these leaks when the program is running.