Name: Khoa Minh Nguyen
ONID: nguyekho
CS 162 – Winter 2017

Final Project – Reflection

OVERALL

This project is definitely the largest program I have ever created and I have enjoyed programming it from start to end. It has challenged me on my understanding of multiple concepts I have learned so far, including: inheritance, polymorphism, encapsulation, STL library, memory management, design, test plan, etc. The entire process took me around 60 hours and I am finally happy to see my efforts come together. Although there are some areas I could improve in my codes, I chose to trim it down in favor of time.

DESIGN

As the requirement for this final project was very open with only a few specifications to meet, I initially struggle quite a bit to come up with a design. I looked at the example files and found that the text-based adventure game from Jason Long was a great one. It took me a few days thinking about the theme and finally I chose a Doctor Strange Silly Adventure that intends to mimic what happened in the blockbuster movie but adds some "hopefully" hilarious twists to the storyline.

The game will have an abstract base class named Space that will be derived to City, Building, and Room class. Most of the functionalities will be implemented in the base class as my design didn't need much polymorphic actions for this. The only polymorphic action is when Strange needs to know where he is in the game, which will be implemented in the whereIsThis() function.

The building blocks that make up a Space will be a Square. They are individual block within the area of a Space that will be located by its row and column in the area of the Space. The Square class is also an abstract class which will be derived to Floor, Wall, Fence, Tree, DarkMatter, and Portal class. Mostly, the polymorphic difference of these derived classes are in the interact() function which will be called when Strange interacts with them. The Portal class is the most different derived class which holds a few more private data member to assist in the process of moving Strange from a Space to another. On each of these classes, an Item or a Character can be placed on.

The Item class is to create different items that are available in the game, some of which will be key to solving the game logic, while others are just there to add more fun and complexity.

The Character class is to create different characters other than Strange in the game. The class is not abstract as it makes sense to create other characters from it. These characters can hold items in their inventory and have ability to take and give items from and to Strange. Each character will have a required item it has to receive from Strange to solve a game logic, and another item which it will give to Strange to help Strange continue solving the game logic.

The DrStrange class is derived from the Character class. It overrides some of the member functions of the Character class so that things make more sense when the game run. Also it has a private boolean data member to indicate whether Strange is flying or not, which is key to solving the second to last logic of the game.

Last but not least, the Engine class is the driver class of all other classes. It contains 6 Space pointers to the different maps of the game, a currentSpace pointer to keep track of where Strange is on, and a It will initialize all maps and have the driver function play() call other support functions to run the game based on user input. It will keep track of the time and if Strange has got his Sorcerer Supreme Certification.

KEY TO SOLVING GAME LOGICS

Objective: gain the 'Sorcerer Supreme Certification' before Earth is devoured (time runs out).

Kathmandu: Strange needs to pick up the lost Rolex, pick up the Ladder and put in to the Fence to go to the next room and give the Rolex to Mordo, who will then give Strange a Sequence to enlarge the portal to KamarTaj.

KamarTaj: Strange needs to give the character on top a ladder so the character can reach to take the Forgetting Powder for him. Then he needs to take the powder back to Mordo (right), so he can make the Forgetting Tea and give Strange the Certification of Humility, which he then can bring to the Ancient One (bottom) to receive the Sequence to the Mystic Art Library. The person on the left is just there to confuse the player while also giving some hint.

Library: Strange needs to pick up the Lady Gaga CD and give to Wong so Wong can give him the Direction to Secret Room. Then Strange need to locate the picture of the HongKong sanctum, pick it up so that he can go through the hidden portal right behind the picture. Other items in the room are irrelevant.

SecretRoom: Strange needs to go to the NewYorkSanctum thru the top portal to get the Hong Kong Mystical Sequence. Once he gets that he can unlock the left portal to go to Hong Kong. The bottom portal leads back to Kathmandu City just to throw the player off and make the player go all the way back. In the middle of the room is the Eye of Agamotto which is another throw-off item that requires the user to make sure to pick up a ladder from previous maps to get. Once they get it, it doesn't really do anything except displaying that Strange can control time which doesn't have anything to do with subsequent game logics.

NewYorkSanctum: Strange needs to pick up the Crimson Bands of Cyttorak and use it on Kaecilius so that Strange can take the Cloak of Levitation from him. Then he needs to acquire the Hong Kong Mystical Sequence to go back to the SecretRoom to unlock the portal to Hong Kong. The vase has nothing to do with the game logic.

HongKong: Apart from the hopefully apparent word that is visually displayed on the map when Strange goes through the portal, those items are irrelevant to the game logic. Strange needs to pick up the beer, use the Cloak of Levitation to fly through the Dark Matter, and give Dormammu the beer so the monster can feel more HAPPY than devouring our little planet. Once

he does that, Dormmamu is the ironic character that will give Strange the Sorcerer Supreme Certification and the player has won the game.

ALGORITHM

Do

    Print Introduction
    Create an Engine (set up the maps, set time to 500, and put Strange at Kathmandu)
    Play the game until the user hits exit, or game is won, or time has run out
    Deallocate memory

Repeat while the user chooses to play another game.

Note: When validating user input, if the input is invalid, the program will display an error message and loop back to the prompt. The validations will utilize my getInt() utility function that I have created in Project 1.

TEST PLAN

-User inputs are validated correctly
-Characters are displayed correctly based on direction
-Squares are displayed correctly based on its derived class.
-Squares are displayed correctly based on whether there is an Item or Character on it or not.
-Strange moves correctly based on user input ‘w’/ ‘s’/ ‘a’/ ‘d’
-Show location of Strange when user hits ‘l’
-Strange interacts with what is in front when user hits ‘c’
-Display Strange inventory correctly when user hits ‘i’
-Strange picks up the item in front if available when user hits ‘p’
-Strange uses item on himself when user hits ‘u’
-Strange gives the chosen item to the Character in front, or put the item to the front Square, or apply it (sequence to portal) when user hits ‘g’
-Character/Portal will not take anything from Strange if they are not the key to solving the character’s game logic.
-Confirm and exit the game when user hits ‘x’
-Ask if user want to play another game before exit
-Strange can climb a fence with a ladder
-Strange can fly through DarkMatter when used Cloak of Levitation
-Strange can pick items and inventory is stored and displayed correctly
-Strange cannot move out of bound even with a ladder or the Cloak of Levitation
-Characters give key item to Strange when they received their key item from Strange
-Conversations of Characters change according to whether their game logic has been solved
-The updated Spaces, conversation, time, and prompts are displayed sequentially correctly after the user chose an action.
-Game ends when Strange receives the Sorcerer Supreme Certification or time runs out, or user chooses to exit
-Game runs on flip, no segmentation fault and all heap memory deallocated.

RESULT: All test results are as expected to my understanding.

PROBLEMS

As I mentioned in the beginning, I struggled quite a bit with the game design to both satisfy the project requirements and have a decent game design. I found that constantly thinking about the design and give pros and cons to each design in my planning phase helped to eliminate overly-complicated designs.

I struggled the most with making sure the updated Spaces, conversation, time, and prompts are displayed sequentially correctly after the user had chosen an action. I had to reorganize my codes quite a bit to make sure that things are displayed correctly and that would make a user-friendly interface (I hope).

There was a problem I encountered when implement polymorphism for the DrStrange class. At the time, I thought for some odd reason I had to implement a virtual useItem() in the base Character class for the useItem() function for the DrStrange class to work. I checked the constructor and I saw that I used the correct DrStrange class to create the Strange. After digging around quite a bit late after midnight, I found that I has used a Character pointer in the Engine class and thus I needed to implement a blanked useItem() function in the Character class just to have it overridden in the DrStrange class. I changed the pointer to the DrStrange class and problem solved!

There were a few memory leaks at the beginning when I first completed a working program. I find that making sure I delete objects as necessary while writing the codes helps a lot later when I have completed the entire project and not have to check everywhere to find the leaks. I learned to set all pointers to NULL once I have deleted the underlying objects. It was important in this project as I needed to pass pointers around Squares and Characters and sometimes it left a dangling pointer that caused a segmentation fault when the destructor tries to delete the underlying object again. Also, I initially thought I needed to delete Strange in the Engine destructor but that was causing over freeing (number of frees greater than number of allocations). After putting some breakpoints to see how the destructor worked, I realized that Strange was already deallocated when the destructor delete HongKong since Strange was on the Space at the end of the game. I commented out the delete Strange line and the magical "All heap blocks were freed -- no leaks are possible" appeared and I was happy.

There were also a lot of testing to make sure that each unit of my program is working as intended. I hope I have not left any logic or error unchecked but I have tried to the best of my ability and within my time constraint. I hope that the program works as I intended and you find it fun to play. Thanks for the fantastic, even though super-busy, quarter and I am happy to report that I have learned a lot!