Victoria Dmyterko
CS 162: Intro to CS II
6/6/2019
Final Project Design

Design

Theme: Alice in Wonderland

1. Space 1: Alice spots a rabbit and must follow him into a rabbit hole
2. Space 2: Alice must drink the potion to shrink small enough to get through the door.
3. Space 3: Alice meets the Caterpillar and must eat a piece of mushroom to return to regular size.
4. Space 4: Alice can have tea with the Madhatter, if she does then she gets a small random key.
5. Space 5: The queen will ask Alice to play croquet. If she refuses then the queen will get mad and yell "Off with her head!". If she agrees then the queen will play croquet with her, then get mad at her for winning and still yells "Off with her head!" The player must either run to the next space or go back to the previous space. (if the player returns, the queen will be gone and will have left a queen of hearts card.)
6. Space 6: Run into some card guards playing cards. Can get a bonus item.
7. Space 7: A hut with a small door, this door leads to home if Alice has the key.

Number of steps: 30 (should be plenty to complete the game)

Classes:

1. Menu Class
2. Player Class
3. Space Class
    a. Derived classes (at least 3)

The Menu class will run most of the game by calling on the correct classes. It will use a pointer to hold the different classes it will use.

<u>Menu</u>

Private:

Space pointers for each space
Space pointer for current location

Player pointer to player class object

Int to keep track of number of steps

Public:

Constructor/Destructor

```
          Intro()

          Play Game
```

Intro Function in Menu Class

```
Introduce the game
Create all of the class objects
Hard code the left and right direction pointers for the classes
Call the play game function
```

Play Game function in Menu class

```
Set number of steps to 0
Hard code location pointer to first space (rabbit hole)
Use a while loop to run the game while the number of steps is less than 30 and the game is not
yet won
     Call action function for current location
     Get the item for the current location
     Set location's visited bool member variable to true
     Add item to player container
     Get returned character from location, if R then call R pointer, if L then call L pointer,
     finally, if X call X pointer (which is the win condition)
     Increase step counter
If win game bool is still false
     Tell player they lost the game
```

Space class

```
Private:
     Declare four pointers for up, down, left, and right

     Bool for whether the space is visited or not
     Bool for whether the item was retrieved or not

     Declare player pointer to represent Alice

Public:
     Constructor/Destructor
     Get and set methods for member variables

     Abstract functions : action and getItem to be defined in derived classes
```

Action function for derived class

Set up the scene for the user

Give user a variety of options
At the end return ‘L’, ‘R’, or ‘X’ to the calling program

getItem function for derived class
Return a string with the item that the user collected in the game

<u>Player class</u>
Private:
- Int number of items
- Vector to represent inventory

Public:
- Constructor/Destructor
- Function to add item to inventory
- Function to find item in inventory
- Function to clear the Inventory
- Function to check if inventory is full

Test Table

Test Case	Input Values	Expected Outcomes	Observed Outcomes
User enters invalid input to start menu	0, 3, a	Program asks user to re-enter their input until it is valid	Program asks user to re-enter their input until it is valid
User enters 2 on the start menu	2	Program exits	Program exits
User enters 1 on the start menu	1	Game starts, asks user to create their teams	Game starts, asks user to create their teams
User chooses a number outside of the choices for riddle questions	6, 10, 0	Program asks user to re-enter their input until it is valid	Program asks user to re-enter their input until it is valid
Player chooses correct answers within step limits	Correct answers, numSteps < step limit	Win game	Win game
Player takes steps > step limit	Takes too many steps	Lose game	Lose game
Test valgrind on program	N/A	All heap blocks freed	All heap blocks freed

Reflection

This project took me a long time to get an idea for. Although I've never actually read or seen Alice in Wonderland I thought it would make a good theme because it had a variety of characters and scenes which would make the creation of derived spaces easy. For the most part creating the outline for this project wasn't too bad. The most difficult part was deciding how to divide the project into different classes. I knew I needed at least a Space class and a Menu. I decided to add a Player class to keep track of the player's inventory. In the menu class I created intro and playGame functions. The intro introduces the game. The playGame function runs the main body of the game. The only thing I really changed about the Menu function was to add a bool to keep track of whether the user made it to the winning screen or not. This was helpful in creating the flow of the menu.

The Space class wasn't changed too much. The one thing that I changed was to place the Player pointer, Alice, under Protected instead of Private where it was originally. This is so that derived classes can call the Player functions relatively easily. I did have to add bools such as visited and item retrieved to the Space class somewhere along the way. This helps with conditional statements throughout the program and allowing me to create a good flow. The Player class was not changed really, since it was such a simple class.

One final change was the overall design of the project. Originally I had planned to make 7 spaces, however, after coding 5 different derived spaces I was getting a bit burnt out typing so much dialogue and description. I decided to toss the original 'Space 6' space with the card guards playing cards because it wasn't necessary. The story of the game isn't quite the same as the original movie or book for Alice in Wonderland, however, I thought it was still whimsical enough and had enough of the elements to work. If I were to expand this project I would continue to add unique spaces that more accurately represent the story of Alice in Wonderland. For this project, the story is too long to properly fit into 6 – 7 spaces. Additionally I would probably use the up and down pointers. However, I decided to keep it simple this time around so that I could complete the project without rushing or turning in something incomplete.

There weren't really any issues with this project. Since all of the directional pointers are hard-coded it is pretty easy to keep track of the dynamically allocated memory. The issues that popped up were largely small syntax errors such as = vs == because I was writing a lot of if/else if/else code so it was easy to miss them when typing quickly. The other issue that came up was the problem of accessing the Player object, Alice through the derived classes. This was easily fixed by moving that member variable to Protected instead of Private. Now the derived classes could access it. One thing I did notice was that I had a clearInventory function in the Player class. However, I don't think I ever used it in the game, because it wasn't really necessary. Also, there are only 3 objects to collect in the game despite having an inventory size of 10, so the player shouldn't ever overfill the capacity of their inventory.

I had a lot of fun writing the dialogue and descriptions! If I come back to this project I'd definitely flesh it out more.