Arjun Kahlon

Final Project Reflection

The Final Project was my favorite Project thus far. I love the freedom we were given in this assignment. I made sure to follow the rubric in designing my game. Any weak points of my project will be addressed in this Reflection.

Task

This project had us create a Space class that would be derived into three subclasses. The Space class is an abstract class that contains virtual functions as well as 4 Space pointers: top, right, left, and bottom. We are then tasked with linking Space objects together in order to form the structure of our game.

Riddle Dungeon

I went through many different ideas when brainstorming this project. I initially considered implementing a battle game similar to the Project 3 and Project 4. The player would travel from space to space, battling different enemies until being tasked with defeating the final boss in the Final Space. However, as I began to write down the pseudocode, I found my implementation to be too similar to the previous projects. I wanted to try something new. So I went back to the drawing board.

I eventually came up with the idea of Riddle Dungeon. In this game, you progress through various rooms by answering riddles that unlock the passage to the next room. You are given a bag to carry Riddle Hints that you pick up in each room. These hints are quite necessary for answering the more difficult riddles in the game, unless you are naturally good at riddles. The player is given a maximum of 20 steps. Answering a riddle uses 1 step. Picking up a riddle hint uses 1 step. Travelling to the next/starting room uses 1 step. The games goes on until the user clears in the challenge in the Final Room or if he/she runs out of steps. The bag container is limited to 4 total riddle hints. When the capacity is exceeded, the oldest riddle hint is removed to make room for the newest riddle hint. I will now explain my classes and the structure of the game.

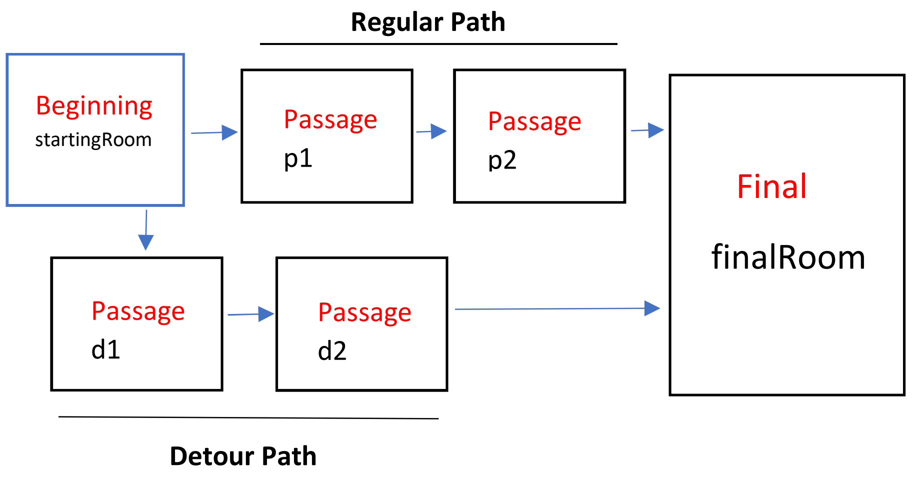
Riddle Dungeon Classes/Structure

In my game, Space is derived into 3 classes: Beginning, Passage, Final. All 3 contain virtual functions that are declared in the Space class (printOptions(), validateChoice(), doChoice(), reset()). They all hold Space pointers (up, down, left, right). Now let me explain each class.

Beginning: This class creates the starting room for Riddle Dungeon. This class contains no riddles. Just a riddle hint that will be used later in the game. This is the only class that uses the Space * for down. All other classes are linear and connect to the next space to the right. In this room, the user is presented with the option to pick up a riddle hint. Doing so will store the riddle hint (in the form of string) in the user's bag. The user is also presented with the option to travel to the right room (the normal passage of the dungeon) or travel down (and take the detour route to through the Dungeon). Both routes are of equal length. At any point in the game, the user can return to this starting room. Thus, if the user gets stuck during one path, he/she can return to the starting room and take the other path.

Passage: This class is implemented the most in the game structure. Both the regular path riddle rooms as well as the detour path room are created with this class. The riddle, riddle answer choices, correct answer, riddle hint, and correct answer are passed in the constructor. A riddle hint is not guaranteed to be passed in. If no riddle hint is passed in, an empty string is entered in the constructor. The user has the options of picking up the hint, answering the riddle, or returning to starting room. If the user answers the riddle correctly, they are allowed to move to the next room. If the user picks up a riddle hint and his bag is full, the oldest hint in the bag is removed to make room for the new hint. It is important to note that the available hint is a passage room may not correspond to the respective riddle in that room. In the beginning, you pick up the riddle hint that corresponds to the riddle in the next room. This was done intentionally to increase the ambiguity of the riddle hints and increase the challenge in the game.

Final: This class differs from the other two classes as it contains two riddles of increased difficulty. The hints available in this class correspond to the riddles in the class. I feel that the riddle hints are necessary to answer the riddles. Since there are two riddles, there are also two riddle hints that are available for pickup. This class also features user IO that is unique to this particular room. Once the two riddles are answered, the user wins the game. This is assuming that the user is under the total allotted steps for the game. Here is the layout of my game.



Now, I will need to address some areas of concerns you may have. The prompt said that the container must contain items that are part of the solution. While it is possible to make your way through the Dungeon without the riddle hints, I find these hints to be almost necessary for answering the more difficult riddles at the end. Instead of a time limit, the game uses a step limit. Another concern you may have is that my classes are too similar. While, they are indeed quite similar, I feel that similarity was necessary for the structure of the game. Given that is a riddle game, I was given much leeway to mess with the structure of each class. They are unique in how the user interacts with them. I wasn't sure how to change them while keeping the central focus I envisioned for this game. I feel that separating my classes to a beginning, passage, final was necessary to implementation this vision I had for the game.

Overall, I really enjoyed this Project as I got to see an idea I had in my head come to life. It was the most enjoyable Project for me this quarter. It is a simple text-based game but I feel it turned out to be a pretty fun game. I had my family members attempt the game and half of them failed on the first try. This project was an awesome experience and a great way to conclude this course.

What I did	**Location**	**Expected**	**Result**
Entered a string as an input on the p1 menu	main.cpp Passage.cpp	My input verification method would discard the string and prompt the user to renter an int value	My cin.ignore and cin.clear() didn't work and this caused the program to go into an endless loop. I had to change my verification method.
Answered Final Room Riddle 1 Incorrectly	Main.cpp Final.cpp	The program would output "Incorrect" once.	The program outputted "Incorrect" twice despite only once answer choice. Thus I had to adjust my conditional output statements for each Riddle.
Picked up every riddle hint in the game	Main.cpp Beginning.cpp Passage.cpp Final.cpp	My bag container would correctly replace the oldest riddle hint when capacity was reached	My bag container did correctly replace the old riddle hints to make way for the new ones.
Purposely made incorrect answer choices to test my step limit	Main.cpp	Program would output that I lost when I reached 0 steps	Program gave me one more chance despite reaching 0 steps left. Realized that I had to change steps <= totalSteps to steps < total Steps

Play Again after winning game	Main.cpp	Program would correctly take me back to the beginning Space	Took me immediately to the final room. Realized that I had to reset all the Boolean values in my classes. Thus I implemented a pure virtual function in Space for reset(). Then I reset all the Boolean values in all my classes at the end of the game loop. The game correctly restarted after.