Final Project

Banks Cargill

Due Date: 8.13.19

Design:

The concept behind my game is a simple dungeon crawler based on someone dreaming. This isn't made aware to the user until the game is won.

The player starts the game in a room at the end of a hallway, confused as to how they got there. A booming voice gives them directions on what they have to do to return home.

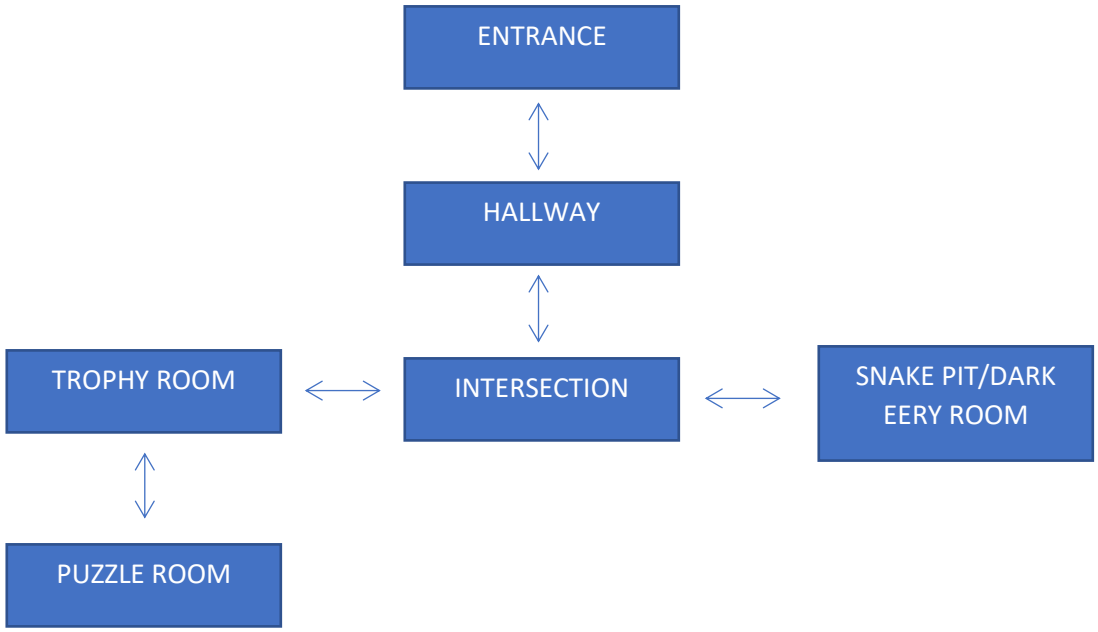
The goal of the game is to find a "key"(a gem) and answer the voices riddles (2 of them).

The player has 8 turns ("hours") to accomplish the goal.

The player will have 5 choices for each round:

1. Look around (interact with space)
2. Move to an adjacent space (pointer manipulation)
3. Print list of items (use doubly linked list)
4. Print remaining steps (each move to an adjacent space counts as a step)
5. Quit the game

Here is a simple diagram to show the linked spaces



Required items:

1. Need torch from hallway to get across snakepit
2. Need gem (key) from snakepit to access the puzzle room

Puzzles/Riddles:

1. In a certain country ½ of 5 = 3. If the same proportion holds, what is the value of 1/3 of 10 ?
   Answer: 4 (5/2 = 3 | 5=6 | 5/3 = 2 →10/3 = 4)
2. What digit is the least frequent between the numbers 1 and 1,000 (0-9)
   Answer: 0

Design:

Space class:

Use class to create a game with the structure of linked space…

Protected:
Space objects that point North, west, east south String name Integer ID
Public: Constructor and destructor Return funtions for adjacent space in each direction Return function for name and ID Pure virtual functions: Interaction

Classes derived from space: Entrance, Hallway, intersection, snakepit, trophy room, puzzle room

Protected:
Public: Constructor and destructor Declare name and ID Virtual Interaction function Determines what happens if you choose to interact in the room Interactions based on items in each room

Backpack class:

Private:
Structure for double linked list (can contain 3 items) Integer numItems
Public: Constructor and destructor addItem printItems searchItems

Player class: Game runs through this class

Private: Pointers to space objects to create linked game structure Pointer of type backpack Integer for steps remaining Functions for printing remaining steps and moving spaces

Public:
Constructor and destructor
Create new space objects, point each correctly to set up the linked game structure, set current space
playGame function
menuIntro()
While (exitGame = false, stepsRemaining>0 , objects not in bag)
displayGameMenu
Switch statement for choice
If (objects in bag) {do this}
Else if (steps remaining<=0){do this}
Else (user chose to exit game){print exiting game}

Game menu

Function to display the menu choices at each step

Testing:

Test Scope	**Description**	**Expected Results**	**Actual Results**
Player.cpp Constructor/destructor Playgame() Entrance.cpp	Created main and tested my player.cpp functions (excluding move) with just the one space (entrance)	Prints intro, menu choices, interaction, and quit	Works as intended. Just needed to do some aesthetic changes
Hallway/player.cpp move function	Testing to see if hallway is implemented correctly and if the move function works	Able to move to hallway, able to add torch as item	Had some trouble with the move function. Had created a space pointer that I didn't need and it was causing a segmentation fault
Backpack (in above test)	Testing for print functionality of items once the above test functions as expected	Prints 1. Torch	Works as expected I am getting a warning that a pointer is used uninitialized that I need

			to look into
Intersection.cpp	Testing interaction in intersection	Nothing to interact with	Works as intended
Snakepit.cpp	Testing interaction in snakepit With torch Without torch	Able to cross pit and acquire gem Death…	Works as intended. Had to rethink the way I was doing this. Ended up adding item called snakebite and exiting while loop in player to deal with this death
Trophy room	Testing interaction in trophy room	Nothing to interact with	Works as intended
Puzzle room	Testing interaction in puzzle room With gem: Without Gem:	Able to access the final riddles. Nothing to interact with	Works as intended Works as intended
Puzzleroom riddles	Testing to be sure that entering the correct answers allows you to win the game. Testing to be sure that entering the incorrect answer to either riddle makes you lose the game	WIN! Lose…	Works as intended Works as intended
Death by stepcount	Intentionally run out of steps	Death	Works as intended
Valgrind	Run through the game a few times with different interactions chosen to test my destructors and for leaks/unfreed memory	Hopefully at this point I have gotten good enough with pointers and deleting new that I won't have issues	Works as intended!

Reflection:

This project started out as a massive undertaking. I had grand plans for a game that was going to have far more complexity than I ended up executing. I think I could have gone bigger but lab 7 took a lot more time than I thought it would and taking another course (final project also due this past Sunday) in addition to work, I just didn't end up with the time necessary to push the boundaries on this game. As much as I would have liked to create something a bit more original, I'm still happy with this game and it shows how far I've come in understanding some of the basic structures in c++. In particular I'm so happy to finally say that I feel comfortable with pointers -that's not to say I won't trip up from time to time but that I can deal with those trips now.

This project could have been one of the most fun for me if we'd had a bit more time. Maybe giving 3 weeks of time for it or removing the lab 7 and implementing some of that lab in a previous week (recursive week perhaps?). As much as I am happy this course is over, I wish I had taken it during a 12 week period of time. I think I would have learned more, covered the topics in more depth and walked away with a more concrete grasp on C++.

The main design choices I had to change were how to deal with player death/victory. I mentioned it in the testing plan but I ended up adding an item "snakebite" that would cause the loop to exit. This was obviously used in the snakepit but proved an easy way of exiting the loop. I decided to implement this to exit the game too as "drool" since the character was sleeping. It ended up being a good workaround that actually added to the theme.

I also hoped to be able to implement the key as a way to get into the puzzle room but decided after having coded the rest of the spaces that it would take more work than it was worth, yet another reminder to work out the design more thoroughly before coding.