Giselle Northy

6/10/2018

CS 162

Final Project

Endless Halls – World of Warcraft

My project is modeled after the Endless Halls in World of Warcraft. The Halls are the 5th step in a hidden quest chain of riddles and quests to earn the Lucid Nightmare mount. In World of Warcraft, the Halls are a maze in which the player can only see one room at a time; some rooms are blocked by rubble and some are shadowy doorways. Inside the maze, the majority of the rooms have nothing in them, five have different colored orbs and another five have colored altars. The goal is to pick up each of the colored orbs and place them on their same colored altar. Once the orb is picked up or placed, orbs a disappear from the room. Upon placing all the orbs, the next room the player enters will have an altar with the final clue in the quest chain.

World of Warcraft version of the maze: the board is an 8x8 maze in which rooms are linked together and randomly generated. The maze varies from player to player and even time of day, which made solving the maze for the Warcraft community incredibly difficult! There are also Non-Intersecting Cross rooms in which two rooms share the same space and cannot contain altars or orbs. Also, when rooms run off the side of the map, they are linked to another room across the map.

People had different tactics to solve the maze, from drawing out maps, leaving orbs in rooms so they could keep track of where they were or simply running in random doorways until they found all the orbs and altar rooms. There are also some confusing hidden tricks to it. Without being aware, when players enter the "Teleport Trap" room, they are randomly placed into a new room in the maze.

https://worldofwarcraft.com/en-us/news/21115833/riddle-me-this-decrypting-the-lucid-nightmare

My first version of the maze: I started with a small 3x3 maze of pointer linked rooms as part of Space* that is hard coded rather than randomly generated; random generation of rooms would be too difficult for the scope of this project. Each room is assigned a number and what type of room it is. All rooms have door ways denoted by green === or | and rubble (NULL) denoted by ####. In this map version, I had a one of each type of room in the map, including two rooms that linked to each other when run off the map (room 1 and 9.)

There are four types of rooms that inherit from Space*:

- **Normal rooms;** nothing special
- **Teleport rooms**; T on the diagram - when user enters this room, a function generates a random # between 1 and the max number of rooms then sends the user without warning to the random room assigned that #. If the random number happens to be the # of the teleport room, it will add +1 to that number and go to a different room.
- **Altar rooms;** denoted by colored -+- and a square on the diagram. The user places orbs on the altar in these rooms as part of the win condition

- **Orb rooms;** denoted by colored ~0~ and a circle on the diagram. The user has the option to pick up orbs in these rooms that are placed in their bag until the user puts them on an altar
- **Pointer linked spaces;** the rooms were connected to each other with pointers. Initially, all the connecting rooms are set to NULL, which is displayed as ### or "rubble."
- The connectSpace function reduced the amount of code needed to connect the rooms setting up the pointer from the previous room to the new room depending on whether they went left, right, up or down. For instance, when room 1 gets connect to room 2 going right, the function automatically sets room 2 going left back to room 1.

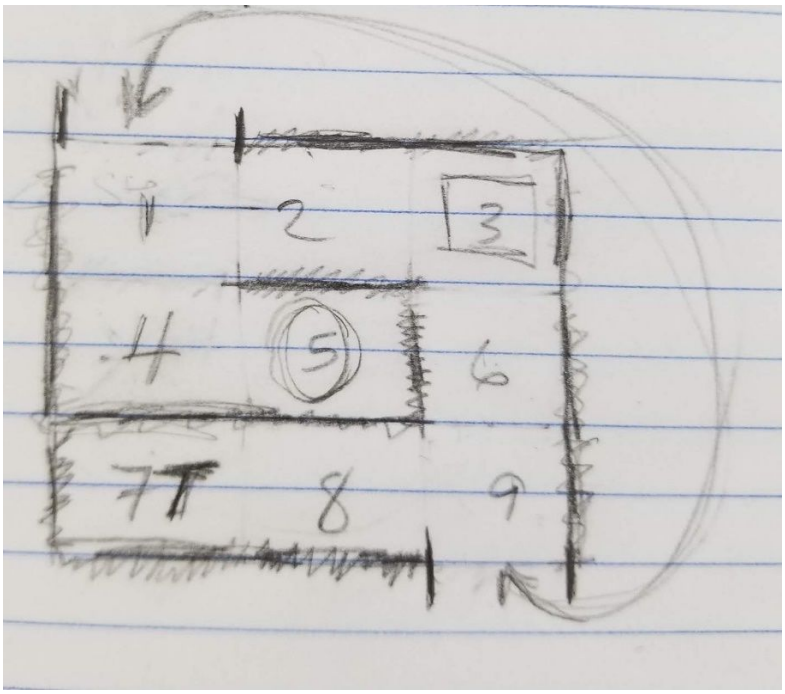

Diagram 1: First maze created for testing and to ensure all the room types worked; rubble is denoted by lines and no lines are doorways, e.g. pointers link those rooms together. Room 1 links to room 9; room 3 is an altar room; room 5 is an orb room; and room 7 is a teleport room.

Orb objects, Player bag and Altar rooms

- When the player enters a room with an orb, they are given the option to pick up the orb or not. If they pick it up, it is placed into their bag and is removed from the room space.
- The bag array holds five items; this number was chosen because that is the max number of colored orbs in the game.
- Orbs stay in the player's bag until they are placed onto the same colored altar.
- When a player enters a room and has the same colored orb in their bag as the altar in the room, the game will ask them if they want to place the orb. Incorrectly placing an orb is not an option.
- In order to win, players must place all orbs onto their altars before their Sanity runs out.
- Once the last orb is correctly placed, the player wins!

Sanity

In the Warcraft version, there is no timer or means to lose; players stay in the same maze until they either complete it or give up. The maze was so confusing that allegedly some players spent hours before completing it. Because of how maddening the maze is, I decided to add “Sanity” as the timer mechanic. Sanity is an integer variable that increments down each time the player enters a new room. Once the player runs out of sanity, they lose the game.

Game Play

At the start of the game, flavor text from the Warcraft version sets the stage for the maze. The player can pick between the Easy Maze, Hard Maze, or Quit.

Easy maze – mostly for testing of spaces and trying the game(Diagram 1)

- 3x3 map with 9 numbered rooms
- 1 red orb room, 1 red altar room and 1 random teleport room
- 2 rooms that run off the map that are linked together
- Sanity set to 20

Hard Maze – modeled after the WoW map version (Diagram 2)

- 222 numbered rooms
- 5 orbs; red, green, yellow, blue and purple and 5 matching colored altar rooms
- 14 rooms run off the map that are linked together
- 1 random teleport room
- Sanity set to 500

In both map versions, the player always starts in room 1 and moves via “w”, “a”, “s” and “d”; “w” = up, “s” = down, “a” = left and “d” = right. Each time the player enters a room, they can view their bag by pressing “b”. Input validation only allows the aforementioned chars in either lower or upper case to move or view the bag.

In similar vein to the Warcraft maze, the terminal is cleared before displaying a new room. This was added after debugging when it was clear the pointers were working.

Player finds orbs and the game gives them the choice to pick them up or leave them. If they pick them up, their bag is updated. If the player has the correct color orb when the player finds the same colored altar, then the game will ask if player wants to place it.

Once all orbs are placed (1 in easy, 5 in hard), the player wins! If Sanity runs out before then, player loses.

Testing Section

Main

- **Menu**
  - **Intro banner displays✓**
  - **Flavor text displays✓**
  - **User can pick Easy Maze, Hard and Quit✓**
  - **Play through game and play again is asked and works✓**

Space

- **Rooms**
  - **Normal room**: user passes through to new room✓
  - **Altar room:** display altar properly and in expected rooms; colors match✓
  - **Orb** room: display orb properly and in expected rooms; colors match✓
  - **Teleport room:** points player to random room✓

Player

- Bag array properly changes when orb is picked up✓
- Sanity increments down properly for each difficulty✓
- Play correctly keeps track of player pointer in the right room✓
- Bag displays properly✓

Board

- Rooms displayed properly✓
- NULL walls display as ####✓
- Linked rooms display as “doorways” === or |✓
- Rooms as expected, and link as shown on the map✓

Game

- Movement
  - Menu appears when Altar is in room and user has correct color orb✓
  - Orb can be placed on altar✓
  - Picked up orbs disappear from bag✓

Orbs

- Players orbs are removed from room upon pickup✓
- Orbs appear in bag when cout selected✓

Reflection

Early on when I read the requirements for the final project, I thought of the Endless halls maze. I played it in game and at the time I wasn't aware of the secret teleport room and how some linked rooms run off the map. When I checked out the developer notes, I was really impressed on how they made the game with an 8x8 board with randomly generated rooms that varied by player and by time. I thought about ways I could re-create it and thought it would be a fun challenge. Early on I realized random maze would be too difficult for the scope of the project, but decided to stick with the orbs, altar, teleport and "normal" rooms. In the future I would like to make the map randomly generated to make it more like the original. Overall, this was a fun project and was a great way for me to envision how instances are created in games!

*/** SPOILER - HARD MAP DIAGRAM **/*

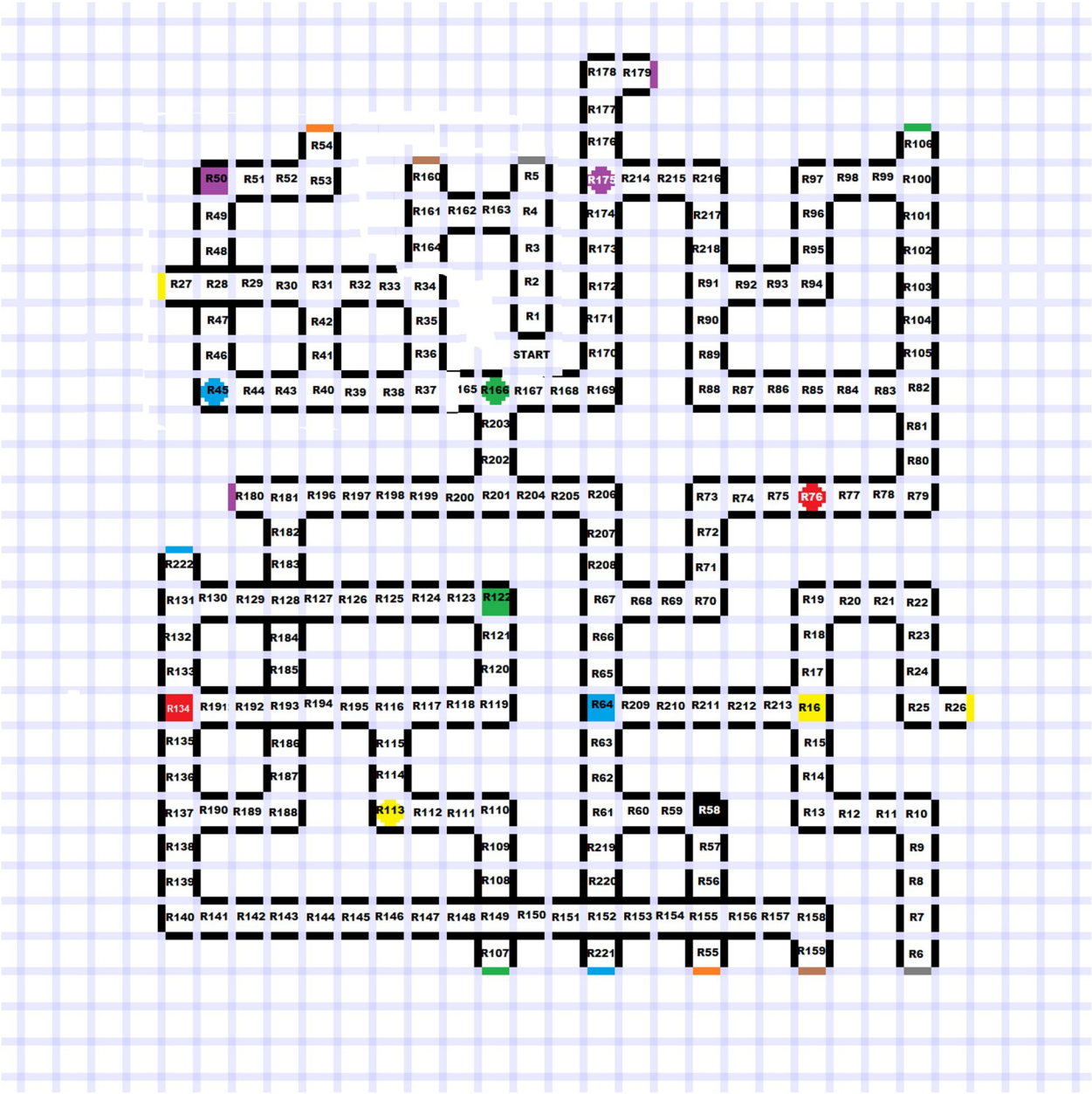

Diagram 2: Hard maze, modeled after a Warcraft player created map. I added numbers to each room. Square colored rooms are altar rooms, circle colored rooms are orb rooms, black room R58 is a teleport room and the colored rooms on the edges of the maps point to the same colored room on the other edge. When rooms intersect, for instance at R48 and R28 they go under each other rather than linking like in the Warcraft version.

Reference: https://docs.google.com/document/d/1MT2ueSYz0uSzGTiGvunexB2Z1cyA_d9aHSFAX-K8Eyk/mobilebasic.