Concept: Player needs to search for items through fog of war, with sections of the board becoming inaccessible as time goes on.

**Game flow**

while(!playerDeath && !playerWin)

* + Print board
  + Player interact/move
    - choose direction
    - player interaction with terrain
  + if player has enough parts and is on rocket, player wins
  + if player touches lava, dies
    - player move
  + destination item pointer points to player
  + source item pointer is null
  + update FOW
  + decrement time remaining until lava
    - if 0, lava, and reset timer

**Classes**

Terrain (space) - the board consists of terrain. Different terrain have different interactions upon player move and take different amounts of time to traverse.

protected:

Terrain \* up down left right

Item \* contents

int row, col

bool fogOfWar

int travelTime

public:

getUp/Down/Left/Right

setUp/Down/Left/Right

get Item

bool interact

void print

void setFOW

Terrain types:

Lava – kills player on touch. Interaction: confirm if player really wants to lava himself

Mountain – allows further vision, but takes 3 time to pass. Interaction: confirm if player wants to spend time climbing

Hot spring: Heals all player damage, but takes 3 time. Interaction: confirm if player wants to spend time soaking

Spikes: Does HP damage to cross. Interaction: confirm whether player wants to cross

Item – things that go on terrain. The player, rocket, and rocket parts are all items. I was going to add health packs and roving aliens as items but ran out of time

protected:

string name

Terrain \* location

private:

get name

void addToInventory

int countRocketParts

void setHP (heal)

void damageHP

int getHP

Item types:

Player – player has an inventory and HP. If player moves to a space that hold a rocket part, the part is added to the inventory.

Rocket – the ultimate goal of the game. If player moves to the rocket, it checks if the player has enough parts, and if so the player wins

Rocket part – player needs to collect these to win the game

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case** | **Input Values** | **Affected functions** | **Expected outcomes** | **Observed outcomes** |
| Input validation | Ints are not ints, etc. | User input functions | Reprompt for correct input | Reprompt for correct input |
| Board instantiates correctly | - | Game() | Board is filled with “normal” terrain initially, which is replaced with other terrain | Board is filled with “normal” terrain initially, which is replaced with other terrain |
| Each terrain points to neighboring terrain | - | Game setup functions | Each terrain points to neighboring terrain | Terrain pointers are messed up when replacing “normal” terrain with new terrain. Made updatePointers() in Terrain to fix pointers, which updates each terrains’ pointers individually. It’s not elegant but it works. |
| Mountains are instantiated, at least 1 space from edge | - | Game setup functions | Mountains are instantiated, at least 1 space from edge | Mountains are instantiated, at least 1 space from edge |
| Mountains increase player’s view range |  | Interact() | Mountains increase player’s view range | Mountains increase player’s view range |
| Mountains take 3 time to traverse |  | Interact() | Mountains take 3 time to traverse | Mountains take 3 time to traverse |
| Hot springs are instantiated, overwriting “normal” terrain | - | Game setup functions | Hot springs are instantiated, overwriting “normal” terrain | Hot springs are instantiated, overwriting “normal” terrain |
| Hot springs heal the player to full HP |  | Interact() | Hot springs heal the player to full HP | Hot springs heal the player to full HP |
| Hot springs take 3 time to traverse |  | Interact() | Hot springs take 3 time to traverse | Hot springs take 3 time to traverse |
| Spikes are instantiated, overwriting “normal” terrain | - | Game setup functions | Spikes are instantiated, overwriting “normal” terrain | Spikes are instantiated, overwriting “normal” terrain |
| Spikes damage player HP | - | Interact() | Spikes damage player HP | Spikes damage player HP |
| Lava timer increments each move | - | runGame() | Lava timer increments each move | Lava timer increments each move |
| Lava timer fills the bottom row with lava when <= 0 | - | runGame () | Lava timer fills the bottom row with lava when <= 0 | Rows get completely messed up, which random spaces getting deleted. Created addLava() function to safely add lava. |
| Player dies when standing on lava |  | Interact(), lavaCalc() | Player dies when standing on lava | Player dies when standing on lava |
| Player dies when HP is 0 |  | Interact(), lavaCalc() | Player dies when HP is 0 | Player dies when HP is 0 |
| Game ends when player dies |  | Game() | Game ends when player dies | Game ends when player dies |
| Rocket parts are stored in normal terrain squares |  | Game setup functions | Rocket parts are stored in normal terrain squares | Rocket parts are stored in normal terrain squares |
| Player adds rocket parts to inventory | - | Move() | Player adds rocket parts to inventory when stepping on them | Player adds rocket parts to inventory when stepping on them |
| Player can’t interact with rocket with <5 parts |  | Interact(), | Player can’t interact with rocket with <5 parts | Player can’t interact with rocket with <5 parts |
| Player wins when interacting with rocket with 5 parts |  | Interact(), | Player wins when interacting with rocket with 5 parts | Player wins when interacting with rocket with 5 parts |

**Reflection**

I quite enjoyed this program. I got the idea for a fog of war game when I stumbled upon the stack overflow post about outputting color text to the console. Getting the Color namespace to work took a little bit of finagling, but once that was set the rest of the project was pretty straightforward.

* I decided to use a board similar to the one from the group project, and was able to repurpose a lot of its code, specifically instantiating and printing the board. Since each type of terrain has a unique appearance and can also be hidden, I made printBoard() call each terrains’ print() function, which prints a different output based on its type and visibility.
* I initially thought to create the player as a type of terrain, but decided that 1. the idea of a person as a type of terrain didn’t make much sense, and 2. it would require deleting terrain and messing with pointers each time the player moved.
* I then came up with the idea for “item”, which is basically any sort of thing that could be placed on terrain. From there adding the player, rocket, and parts was fairly easy.
* Once I had the board, fog of war, and items created, I started on the lava timer. The timer itself was simple, but replacing a row with lava didn’t work with the addTerrain() function I had been using, so I rewrote it into an addLava() function that was much more careful of pointers, and everything started to work.
* Adding the first interaction for terrain was a little tricky, but once I split the player’s movement into two parts (interact, then move) everything clicked into place. If an interaction fails for whatever reason (inventory full, player chooses not to climb, etc.), the interaction returns false so the move part doesn’t happen, and it kicks back to asking the player for input.
* I had originally intended to add roving aliens to damage player HP and portable health packs that could be used on demand, but ran out of time. I’d like to go back and add them later if possible.

I like to think I’ve come pretty far from when I started this class. I still have much, much more room to improve and many best practices to learn, but mechanically I’m a lot more comfortable with pointers, classes, inheritance and polymorphism.