Tucker Lavell
CS162 Fall 2017
Final Project: Text Based Game
Documentation

Design

Backpack

```
class Backpack {
private:
	int capacity = 3;
	deque<string> inventory;
	bool invFull();
public:
	Backpack();
	~Backpack();
	void addToInv(string);
	void removeFromInv();
	void printInv();
};
```

Player

```
class Player {
private:
	int moonStones;
	int y;
	int x;
public:
	Player();
	~Player();
	Backpack *bp;
	void setY(int);
	int getY();
	void setX(int);
	int getX();
};
```

Board

```
const int ROWS = 5;
const int COLS = 5;
```

```
enum SlotState {
	NORTH = '^',
	SOUTH = 'v',
	EAST = '>',
	WEST = '<',
	CLEFAIRY = 'C',
	ROCKET = 'R',
	CURRENT = '@',
	UNVISITED = ' '
};
class Board {
private:
	SlotState gameBoard[ROWS][COLS];
public:
	Board();
	~Board();
	void setHome();
	void setRocket();
	int move(int, int, SlotState);
	void printBoard();
};
```

Space

```
class Space {
protected:
	int roomNum;
	string roomName;
	string roomType;
	Space *north; // up
	Space *south; // down
	Space *east; // right
	Space *west; // left
public:
	Space();
	~Space();
	void setRoomNum(int);
	virtual int getRoomNum();
	virtual void setRoom() = 0;
	virtual void printEvent() = 0;
```

```
};

HomeRoom
class HomeRoom : public Space {

};

EmptyRoom
class EmptyRoom : public Space {

};

ItemRoom
class ItemRoom : public Space {

};

RocketRoom
class RocketRoom : public Space {

};

Quest
class Quest {
private:
        void moveResult(int);
public:
        Quest();
        ~Quest();
        void mainMenu();
        int questMenu();
        void turn();
        void moveNorth();
        void moveSouth();
        void moveEast();
        void moveWest();
        void action(int);
        int questLength;
        string north;
        string south;
        string east;
        string west;
        string inventory;
        Board *board;
        Player *ash;
};
```

Test Table

Test Case	Input Values	Expected Outcome	Observed Outcome
Print gameboard	Fill table with empty slots	A nice looking grid	Not exact, took a little fine tuning
Print gameboard with Start and Finish	Fill table with empty slots. Set the center to Home Random set Finish	Grid with a C in the center, and an R somewhere random	Worked as expected, forgot to seed random first time
Add an item to inventory	Add "pants"	Inventory with one item (pants)	Inventory with one item (pants)
Add an item beyond capacity (3)	1. "Pants" 2. "Grass" 3. "Glass" Add "Helium" Remove 2	1. "Pants" 2. "Glass" 3. "Helium"	1. "Pants" 2. "Glass" 3. "Helium"
Add an item to a Players backpack	Add "pants"	1. "pants"	1. "pants"
Have a player move on the board	1. North	Board shows @ for players loc, and an arrow showing movement direction from last room	Board showed @ for players loc, and an arrow showing movement direction from last room
Don't change state of HomeRoom	1. North 2. South	HomeRoom shows C even if we leave it and come back	HomeRoom showed C

Reflection

I had a lot of fun with this Project. It was definitely the most challenging one yet. I wish we had somehow gotten this assignment earlier, because I was out of town until Tuesday night after Thanksgiving, so I only had a week to work on this. I also didn't end up having time to do Lab 10 either. It took me a long time to wrap my head around this, because of the linked structure. My idea for a game was to use a grid and have a user navigate in all 4 directions, with the final room being locked behind the requirement to gather 3 items. I decided my theme would be Pokemon.

I started off small, doing like P3, gradually increasing the scope. First thing I did was look at my TicTacToe game from 161, and used that to make a board. Then I made the board print a start location (HomeRoom, C) and a final destination (RocketRoom, R). Next I created a backpack, with a capacity, that could be filled and have items removed. Then I created a player with a backpack.

Next I started on Quest. Quest was fun to write, I didn't realize how much I had learned about directional movement from Langton's Ant. I was able to make these movement functions much more readable. The biggest mistake I made with directional movement was getting north and south on the board mixed up, because i was thinking north is up so it adds and south is down so it subtracts. I was confident that those were working properly, but movement was still not being printed correctly. I decided to just come back to it the next day.

I spent the next several days trying to change my board to a linked structure, to very little success. I will include a separate zip file with that code, if I could maybe get some suggestions or help, that would be really helpful and appreciated, so I can do it in the future. I had planned to move the head(location) around the board, but I couldn't even get the structure. I didn't want to change my idea for my game, because I thought a linear game would be even more boring to play. I settled on continuing to use the array structure that I had for my board, but I would turn it into an array of Spaces at least. For this, Space is basically a pure abstract class.

After a little tweaking of my board code, I was able to print a board full of Spaces(EmptyRoom, ' ') successfully. Then I used empty classes that returned different things to fill my board with the home room, the rocket room, and the moonstone rooms(ItemRoom, X). Then I went back to working on movement. After not looking at this version of the code for a couple days, I noticed that I wasn't using the players coordinates consistently. Some places I passed it with math, other places I passed just the coordinate then did math. After some tweaking, I made it consistent where math was being done and where math was being passed.

Once movement was working I used my abstract function to have an event occur in each room, on its first visit. This came together so nicely, suddenly my game was completely working, it felt so good. I had a bit of an error, that I didn't know the cause of, but once in awhile the board would print more than 3 MoonStones. I also found out Play Again wasn't working, it did not print the MoonStones after the first time, and even if you made a move, the game would go back to play again after the 1 move. It had to do with my static int counter in my MoonStones assignment. There was also another problem with my static int turn counter too. Once I figured this out, it took me a while to figure out how to fix the statics problem. I ended up just deciding to use a do while loop to reinitiate the Quest instead of using the do while to keep the game going. This ended up fixing the problems with the MoonStones and Play Again.

Even though I am disappointed that I couldn’t get this to work with a linked structure, I am still proud of it. I think this code is structured well, and is mostly self documenting. I have learned a lot from this course, and I feel like this Project reflects that. I think I know linked lists well after my investment on the previous assignments, but a 2D doubly linked list was really hard to wrap my head around.

Printed a board
Printed with a Home and an end
Created backpack
Filled and removed backpack
Created player with a backpack
Started on “quest”
Worked for a while on quest, making minor changes to current stuff along the way.
First time i ran quest it would only allow me to move south, logic for printing questMenu was broken
North and south mixed up
Priting movement problems
Tried to completely change it (had board as an array, tried converting to linked list, failed)
Filled board with spaces
Movement successful, funky on rocket
Forgot to check for rocket in previous properly
MoonStones on the board werent properly reset originally if I did play again.
Also play again didnt work, the program would ask play again after 1 input turn
Fixing that also fixed occassionally having more than 3 moonstones print on the board