**Project 2**

Title

**Random RPG Game**

Course

**CSC-17A**

Section

**48130**

Due Date

**08 December 2014**

Author

**Najera, Enrique**

**Introduction**

Title: Random RPG Game

The Random RPG game is a game in which a player moves around a map using the w a s d keys and unexpectedly receives visits from entities trying to kill the player.

The player starts at the center of a map generated by the player and must walk around to get health, bullets, and points.

The player receives points after killing an entity. The more points the player gets, the better.

The player can quit at any time by pressing ‘p’ or going to the pause menu ‘b’ and choosing to exit.

The game is over once the player runs out of health or simply exits the game.

**Summary**

Project size: 1686 lines

Variables: 17 (in main)

Constructs utilized: 23 (From chapters 9 to 16)

My project meets the criteria for a first project because it uses all the constructs we have learned thus far .My project was a bit challenging since I am used to doing these kinds of projects using dynamic-link libraries and headers from other sources. I also had difficulty using a two-dimensional array (the map) in a single class; the program would always crash upon building the map array. The final problem I had was utilizing the ‘save’ option. I wanted to decrypt the file’s contents and decrypt the file and send the information to the appropriate members and have the members rewritten based on the new information. I could not properly decrypt the file’s name so I could not open the file; thus leading into abandoning the ‘save’ and ‘load’ feature. This project took me about three days to fully complete, excluding the write up and flowchart.

**Description**

To program the solution to the problem I first made a two-dimensional array and added a character to the center of the screen. Then I incremented the player’s position to make the character appear as if it is walking along the array. After, I added random entities and pickups the player can interact with.

Note: Anything red on the screenshots below are emphasis and not actual in/outputs.

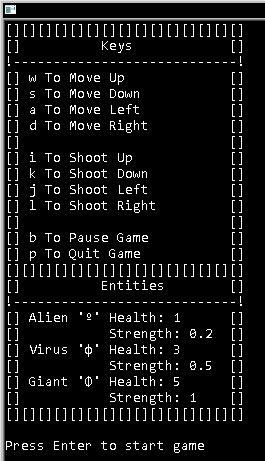
Input menu option 1 Output menu option 1

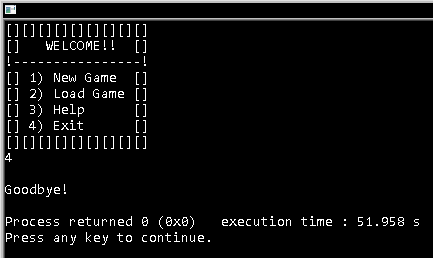
Input menu option 2 Output menu option 2

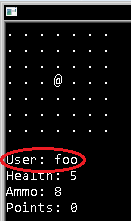
Input menu option 3 Output menu option 3

Input menu option 4 Output menu option 4

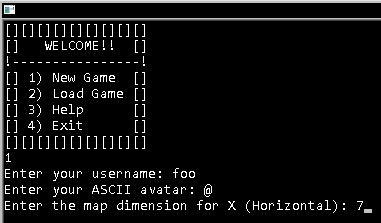
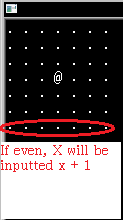
Input Username Output Username

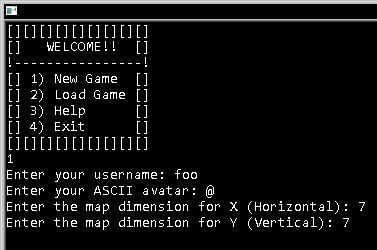
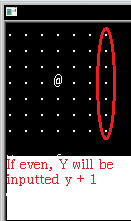
Input Avatar Output Avatar

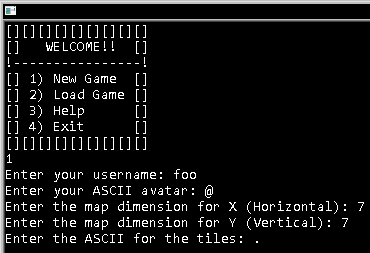
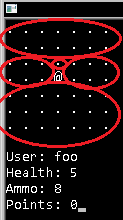
Input map dimension for X Output map dimension for X

Input map dimension for Y Output map dimension for Y

Input ASCII for tiles Output ASCII for tiles

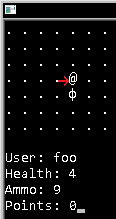
 

Note: These screenshots may vary since things are generated at random.

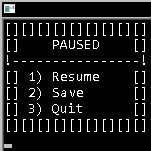
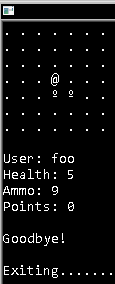
Input ‘w’ during gameplay Input ‘s’ during gameplay

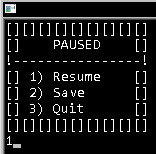
Input ‘a’ during gameplay Input ‘d’ during gameplay

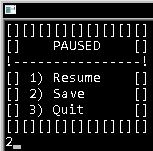
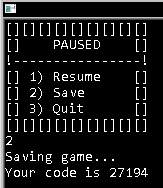
Input ‘b’ during gameplay Input ‘p’ during gameplay

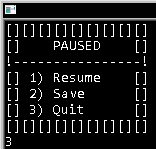
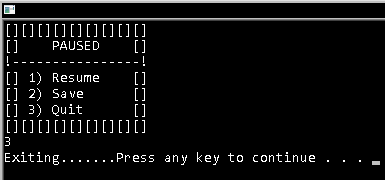
Input pause menu option 1 Output pause menu option 1

Input pause menu option 2 Output pause menu option 2

Input pause menu option 3 Output pause menu option 3

Input load code Output if load code was correct (VARIES)

Output if load code incorrect



Basic Flowchart

3

B

A

p.health > 0

isGameRunning

Input code

End main

True

Output player name, health, ammo, points and arrMap1

Display help menu

2

1

Input pName, avatar, mapX, mapY, tile

B

menuIn

Prompt for menu option

Declare Variables

wasd, \*loc, \*\*arrMap1, locX, locY, turn, menuIn, debug[0], isGameRunning, action

Start method main

A

Function Prototypes

randnumX,

randnumY

System Libraries

iostream,

cstdlib,

ctime,

conio.h,

new

Published Libraries

Map,

Player,

Entity,

MainMenu,

Speechbox

Global Comments

Name

Date

Purpose

False

True

False

Pseudo Code

*Prompt for menu option*

*If menu option == 1*

*If getIsCodeGood == false*

*Prompt for player name, avatar, map x, map y*

*Try to allocate memory, catch insufficient memory*

*Fill the map for y = 0; y < map y; y++*

*Fill the map for x = 0; x < map x; x++*

*While game is running*

*Place entities and pickups on map*

*Wait for player input*

*If input == p*

*Exit program*

*If input == b*

*Display menu*

*Prompt for menu option*

*If option == 1*

*Resume game loop*

*If option == 2*

*Save game*

*If option == 3*

*Exit program*

*If input == w*

*Move up*

*Check for collision*

*If collision == entity*

*Remove health*

*If collision == health*

*Add health*

*If collision == bullet*

*Add ammo*

*If input == s*

*Move down*

*Check for collision*

*If collision == entity*

*Remove health*

*If collision == health*

*Add health*

*If collision == bullet*

*Add ammo*

*If input == a*

*Move left*

*Check for collision*

*If collision == entity*

*Remove health*

*If collision == health*

*Add health*

*If collision == bullet*

*Add ammo*

*If input == d*

*Move left*

*Check for collision*

*If collision == entity*

*Remove health*

*If collision == health*

*Add health*

*If collision == bullet*

*Add ammo*

*If input == i*

*If ammo == 0*

*Output no more ammo*

*Else if ammo > 0*

*Remove ammo*

*Check for entity*

*If entity on top*

*Remove entity health*

*If entity dead*

*Add points*

*If input == k*

*If ammo == 0*

*Output no more ammo*

*Else if ammo > 0*

*Remove ammo*

*Check for entity*

*If entity on bottom*

*Remove entity health*

*If entity dead*

*Add points*

*If input == j*

*If ammo == 0*

*Output no more ammo*

*Else if ammo > 0*

*Remove ammo*

*Check for entity*

*If entity on left*

*Remove entity health*

*If entity dead*

*Add points*

*If input == l*

*If ammo == 0*

*Output no more ammo*

*Else if ammo > 0*

*Remove ammo*

*Check for entity*

*If entity on right*

*Remove entity health*

*If entity dead*

*Add points*

*If menu option == 2*

*Prompt for load code*

*If code is good*

*Set map and player properties according to save file*

*Start game loop*

*If menu option == 3*

*Display help menu*

*Start game loop*

*If menu option == 4*

*Exit program*

Major Variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Name** | **Description** | **File** | **Line(s)** |
| char | wasd | Holds player in game input | main.cpp | 34, 168, 180, 192, 215, 258, 301, 344, 390, 476, 562, 648 |
|  | \*loc | Holds plyer’s location | main.cpp | 35, 177, 249, 292, 335, 378 |
|  | \*\*arrMap1 | Game map | main.cpp | 36, 91, 105, 128, 747, 752, 764, 766 |
|  | avatar | Avatar of entities | Entity.hpp | 48, 53, 66, 83, 88, 101, 118, 123, 136 |
|  |  | Avatar of player | Player.hpp | 27, 35, 69 |
|  |  |  | Player.cpp | 50, 54 |
|  | tile | Stores the map tile | Map.hpp | 23, 34, 40, 48 |
|  |  |  | Map.cpp | 54, 74 |
| int | locX | Holds entity’s location on ‘x’ | Entitiy.hpp | 16, 25, 30, 35, 53, 58, 68, 88, 93, 103, 123, 128, 138 |
|  |  | Holds player’s location on ‘x’ | main.cpp | 37, 116, 139, 115, 157, 159, 162, 164, 177, 248, 249, 291, 292, 304, 334, 335, 347 |
|  |  | Holds player’s location on ‘x’ | Player.hpp | 22, 42, 57 |
|  | locY | Holds entity’s location on ‘y’ | Entity.hpp | 17, 25, 30, 37, 53, 60, 70, 88, 95, 105, 123, 130, 140 |
|  |  | Holds player’s location on ‘y’ | main.cpp | 38, 117, 140, 155, 177, 218, 248, 249, 291, 292, 294, 337 |
|  |  | Holds player’s location on ‘y’ | Player.hpp | 23, 44, 59 |
|  | turn | Calculates turn taken | main.cpp | 39, 157, 159, 162, 164, 170, 171 |
|  | menuIn | Holds menu options | main.cpp | 40, 60, 63, 72, 75, 76, 200, 202, 205, 206, 208 |
|  | isGameRunning | Checks if game is running or not | main.cpp | 42, 183, 757 |
|  | action | Checks if player is attacking or not | main.cpp | 43, 152, 182, 220, 263, 306, 349, 392, 478, 564, 650 |
|  | points | Stores an entity’s point worth | Entity.hpp | 19, 27, 54, 74, 89, 109, 124, 44 |
|  | strength | Stores an entity’s point worth | Entity.hpp | 20, 26, 31, 41, 54, 76, 89, 111, 124, 146 |
|  | mapX | Stores location on ‘x’ | Map.hpp | 21, 34, 44 |
|  |  |  | Map.cpp | 25, 46, 59 |
|  | mapY | Stores location on ‘y’ | Map.hpp | 22, 34, 46 |
|  |  |  | Map.cpp | 37, 50, 60 |
|  | ammo | Stores player ammo amount | Player.hpp | 24, 36, 61 |
|  |  |  | Player.cpp | 37, 38 |
|  | pts | Stores total player points | Player.hpp | 25, 37, 53, 63 |
|  | health | Stores entity health amount | Entity.hpp | 18, 26, 31, 39, 54, 62, 72, 89, 97, 107, 124, 132, 142 |
|  |  | Stores player health amount | Player.hpp | 26, 35, 50, 65 |
| bool | isOpenSuccess | Checks if file opened successfully | FileStream.hpp | 17, 21, 25 |
| vector | rndm | Stores random numbers | Random.hpp | 24, 40, 48, 56 |

Concepts, syntax, keywords… (chapters 9 - 16)

|  |  |  |
| --- | --- | --- |
| **Type** | **File** | **Line** |
| Pointer ‘\*’ | FileStream.hpp | 29, 30 |
|  | FileStream.cpp | 22, 74, 102 |
|  | main.cpp | 25, 26, 35, 36, 51, 52 , 53, 91, 778, 786, 789, 796, 802, 805 |
|  | MainMenu.hpp | 29 |
| Pointer ‘&’ | main.cpp | 142, 143, 145, 146, 148, 149, 155, 157, 159, 162, 164, 177, 249, 292, 335, 378, 752 |
|  | Map.cpp | 42 |
|  | Map.hpp | 51 |
|  | Player.cpp | 42, 60 |
|  | Player.hpp | 76, 77 |
| Dynamic Memory Allocation | main.cpp | 51, 52, 53, 91 - 93 |
| ‘delete’ | main.cpp | 764, 766 - 769 |
| ‘string’ | Player.hpp | 30, 47, 72 |
| Aggregation ‘.’ | main.cpp | 66, 75, 76, 79, 91 – 93, 102, 104, 105, 109, 112, 113, 116, 117, 126, 128, 139 – 149, 155 – 164, 174, 187, 198, 205, 206, 208, 222, 223, 225, 226, 228, 235, 240, 242, 248, 251, 261, 264, 265, 268, 269, 271, 276, 278, 283, 285, 291, 294, 308, 309, 311, 312, 314, 319, 321, 326, 328, 334, 337, 347, 351, 352, 354, 355, 357, 362, 364, 369, 371, 377, 380, 395, 397, 400, 404, 407, 409, 413, 416, 419, 422, 425, 427, 429, 432, 437, 440, 443… |
|  | Map.cpp | 46, 50, 54, 59, 60, 74 |
|  | Player.cpp | 46, 50, 53, 54, 63 - 66 |
| ‘::’ | FileStream.cpp | 22 |
|  | Map.cpp | 20, 30, 63, 68 |
|  | Player.cpp | 20, 34, 37, 45, 53 |
| Arrow pointer ‘->’ | main.cpp | 223, 225, 230, 237, 244, 266, 268, 273, 280, 287, 309, 311, 316, 323, 330, 352, 354, 359, 366, 373, 407, 425, 450, 493, 511, 536, 579, 597, 622, 665, 683, 708 |
|  | Player.cpp | 28, 29 |
| ‘this’ | Player.cpp | 28, 29 |
| Enumerated data type | Speechbox.hpp | 24 |
| Opening file for I/O | FileStream.cpp | 34, 66, 86, 105, 114 |
| File output (read from) | FileStream.cpp | 37 – 48, 66 - 69 |
|  | main.cpp | 58, 66, 198 |
|  | MainMenu.hpp | 82 |
| File input (write to) | FileStream.cpp | 80 – 91, 105 – 107, 114 - 116 |
|  | main.cpp | 187 |
|  | MainMenu.hpp | 104 |
| Binary files | FileStream.cpp | 66 – 69, 105 – 107, 114 - 116 |
|  | MainMenu.cpp | 82, 104 |
| ‘class’ | Entity.hpp | 13, 45, 80, 115, |
|  | FileStream.hpp | 14 |
|  | MainMenu.hpp | 25 |
|  | Map.hpp | 18 |
|  | Menu.hpp | 21 |
|  | Player.hpp | 19 |
|  | Random.hpp | 21 |
|  | Speechbox.hpp | 21, 39, 54, |
| Constructor | Entity.hpp | 24, |
|  | MainMenu.hpp | 37 |
|  | Map.hpp | 33 |
|  | Menu.hpp | 28 |
|  | Player.hpp | 34 |
| Destructor | N/A | N/A |
| Overloaded constructor | Entity.hpp | 29, 52, 87, 122 |
|  | MainMenu.hpp | 40 |
| ‘private’ |  |  |
| ‘public’ | Entity.hpp | 21, 49, 84, 119 |
|  | FileStream.hpp | 18 |
|  | MainMenu.hpp | 34 |
|  | Map.hpp | 24 |
|  | Menu.hpp | 26 |
|  | Player.hpp | 31 |
|  | Random.hpp | 25 |
|  | Speechbox.hpp | 25, 41, 56, |
| Instance variables | main.cpp | 46 – 48, 51 - 53 |
| ‘friend’ | Map.hpp | 51 |
|  | Player.hpp | 76, 77 |
| Overload operators | Map.hpp | 51 |
|  | Map.cpp | 42 - 78 |
|  | main.cpp | 82, 85, 133, 755 |
|  | Player.hpp | 76, 77 |
|  | Player.cpp | 42 – 57, 60 - 68 |
| Base class | Entity.hpp | 9 - 42 |
|  | FileStream.hpp | (Whole file) |
|  | Map.hpp | (Whole file) |
|  | Speechbox.hpp | 21 - 36 |
| Inheritance (class) | Entity.hpp | 45 – 77, 80 – 112, 115 - 147 |
|  | MainMenu.hpp | (Whole file) |
|  | Menu.hpp | (Whole file) |
|  | Player.hpp | (Whole file) |
|  | Speechbox.hpp | 54 - 64 |
| Multiple Inheritance | Speechbox.hpp | 39 - 51 |
| ‘protected’ | Entity.hpp | 15, 47, 82, 117 |
|  | FleStream.hpp | 16 |
|  | Map.hpp | 20 |
|  | Menu.hpp | 23 |
| Abstract class | Speechbox.hpp | 21 – 36, 39 - 51 |
| ‘virtual’ | Speechbox.hpp | 28, 43 |
| Polymorphism | main.cpp | 230, 237, 244, 273, 280, 287, 316, 323, 330, 359, 366, 373, 407, 425, 450, 493, 511, 536, 579, 597, 622, 665, 683, 708 |
| Function Templates | Random.hpp | 37 – 41, 44 – 49, 52 - 57 |
| Class Templates | Random.hpp | 21 – 33 |
| Exceptions | Map.hpp | 26, 28 |
|  | Map.cpp | 57 – 72, |
|  | main.cpp | 89 - 99 |
| STL | main.cpp | 15, 96 |
|  | Random.hpp | 14, 24, 48, 56 |

References

I mostly used the book for syntax and example code and used the cplusplus.com for understanding and correctly implementing keywords. If I got an error I would search the cplusplus forum and it usually turned out to be syntax errors. Finally, I used similar methods of RPG game format from game libraries in other languages I program in.

Main program

/\*

\* File: main.cpp

\* Author: Najera Enrique

\* Purpose: CSC-17-A Project 2

\* Some random RPG

\*

\* 08 December 2014

\*/

//System Libraries

#include <iostream>

#include <cstdlib> // rand, srand

#include <ctime> // time

#include <conio.h> // \_getch()

#include <new> // Catch memory errors

//Published Libraries

#include "Map.hpp"

#include "Player.hpp"

#include "Entity.hpp"

#include "MainMenu.hpp"

#include "Speechbox.hpp"

//Function Prototypes

int randnumX(int \*, int \*);

int randnumY(int \*, int \*);

//Namespaces

using namespace std;

int main()

{

//Declare Variables

char wasd = ' '; // Holds player's movement

char \*loc = 0; // Holds player location

char \*\*arrMap1; // For allocating memory

int locX = 0; // Holds X location

int locY = 0; // Holds Y location

int turn = 0; // Holds turns taken

int menuIn = 0; // Holds menu input

int debug[0]; // Converts char to int for debug file

bool isGameRunning = true; // Checks if game is still running

bool action = false; // Checks if player is fighting

//Define objects

Player p;

Map map1;

MainMenu menu;

//Allocate instances

Speechbox \*sB = new Playerbox();

Playerbox \*pB = new Playerbox();

Pickupbox \*pU = new Pickupbox();

//Build FileStream object

FileStream f;

//Display menu and prompt for choice

f.readFile("Main.mnu", 0, 0);

cin >> menuIn;

//If 3, display help screen

if(menuIn == 3)

{

system("cls");

f.readFile("help.mnu", 0, 0);

cout << "\nPress Enter to start game\n";

\_getch();

system("cls");

}

//If 4, exit and don,t bother set or output

else if(menuIn == 4){ cout << "\nGoodbye!\n"; exit(EXIT\_SUCCESS); }

//If not 3, set and output menu

menu.setInN(menuIn);

menu.outMenu(menuIn);

//If load game fail, prompt

if(menu.getIsCodeGood() == false)

{

//Prompt for user info

cin >> p;

//Prompt for map properties

cin >> map1;

}

//Allocate memory for map

try

{

arrMap1 = new char\*[map1.getMapY()];

for (int i = 0; i < map1.getMapY(); ++i)

arrMap1[i] = new char[map1.getMapX()];

}

//Catch error

catch (bad\_alloc)

{

cout << "Insufficient memory!\n";

}

//Fill the map with tiles

for(int y = 0; y < map1.getMapY(); y++)

{

for(int x = 0; x < map1.getMapX(); x++)

arrMap1[y][x] = map1.getTile();

}

//Place player in center of map

arrMap1[map1.getMapY() / 2][map1.getMapX() / 2] = p.getAvatar();

//Get player's location for logging

p.setLocX(map1.getMapX());

p.setLocY(map1.getMapY());

//Get player's location for movement

locX = p.getLocX() / 2;

locY = p.getLocY() / 2;

//Build entities

//location x, location y, health, strength

Alien alien(0, 0, 1, -0.2f);

Virus virus(0, 0, 3, -0.5f);

Giant giant(0, 0, 5, -1.0f);

//Output map

for (int y = 0; y < map1.getMapY(); y++)

{

for (int x = 0; x < map1.getMapX(); x++)cout << arrMap1[y][x]<<" ";

cout<<endl;

}

//Display user info

cout << p;

//Game loop starts here

do{

//Update locations

//Player

p.setLocX(locX);

p.setLocY(locY);

//Alien

alien.setLocY\_Alien(randnumY(&locY, &locX));

alien.setLocX\_Alien(randnumX(&locY, &locX));

//Virus

virus.setLocY\_Virus(randnumY(&locY, &locX));

virus.setLocX\_Virus(randnumY(&locY, &locX));

//Giant

giant.setLocY\_Giant(randnumY(&locY, &locX));

giant.setLocX\_Giant(randnumY(&locY, &locX));

//Place objects on map if not action

if(action == false)

{

//Entities

arrMap1[randnumY(&locY, &locX)][randnumX(&locY, &locX)] = alien.getAvatar\_Alien();

//Every 2 turns add a virus

if(turn % 2 == 0)arrMap1[randnumY(&locY, &locX)][randnumX(&locY, &locX)] = virus.getAvatar\_Virus();

//Every 4 turns add a giant

if(turn % 4 == 0)arrMap1[randnumY(&locY, &locX)][randnumX(&locY, &locX)] = giant.getAvatar\_Giant();

//Pickups

//Every 5 turns add a bullet

if(turn % 5 == 0)arrMap1[randnumY(&locY, &locX)][randnumX(&locY, &locX)] = p.getBullet();

//every 10 turns add health

if(turn % 10 == 0)arrMap1[randnumY(&locY, &locX)][randnumX(&locY, &locX)] = '+';

}//End placing objects

//Get player movement

wasd = \_getch();

//Increment turn

turn++;

//For keylogging

p.setMovement(wasd);

//Keep track of player location

loc = &arrMap1[locY][locX];

//If player quits, break from game loop

if (wasd == 'p')

{

action = false; // No action raking place

isGameRunning = false; // Game will close

debug[0] = 112; // Debug value is quit

cout << "\n\nGoodbye!\n" << endl;

//Write to file for debugging purposes

f.writeFile("debug.info", debug, 0);

break;

}//End player quits

//Pause game

else if(wasd == 'b')

{

//INTIT var menuIn

menuIn = 0;

system("cls");

//Read pause menu file

f.readFile("Pause.mnu", 0, 0);

//Prompt for decision

cin >> menuIn;

//If 3, exit and clean

if(menuIn == 3)break;

//If not 3, set and output menu

menu.setInN(menuIn);

menu.outPMenu(menuIn);

//If game saved, output load game code

if(menuIn == 2)cout << "Your code is " << menu.getPassword() << endl;

\_getch();

}//End pause game

/\* PLAYER MOVEMENT \*/

//Move up

else if (wasd == 'w')

{

//If inside boundary

if(locY > 0)

{

action = false;

//Detect if player touched something

if(arrMap1[locY][locX] == p.getBullet()) // Bullet add bullet

{ p.setAmmo(1); pU->picksup(0); \_getch(); }

else if(arrMap1[locY][locX] == '+') // Health Box add health

{ p.setHealth(1); pU->picksup(1); \_getch(); }

else if(arrMap1[locY][locX] == alien.getAvatar\_Alien()) // Alien take health

{

p.setHealth(alien.getStrength\_Alien());

//Output entity attacks

sB->attacks(0);

\_getch();

}

else if(arrMap1[locY][locX] == virus.getAvatar\_Virus()) // Virus take health

{

p.setHealth(virus.getStrength\_Virus());

//Output entity attacks

sB->attacks(1);

\_getch();

}

else if(arrMap1[locY][locX] == giant.getAvatar\_Giant()) // Giant take health

{

p.setHealth(giant.getStrength\_Giant());

//Output entity attacks

sB->attacks(2);

\_getch();

}

arrMap1[locY][locX] = map1.getTile(); // Redraw tile for player

loc = &arrMap1[locY--][locX]; // Get new location

arrMap1[locY][locX] = p.getAvatar(); // Draw avatar on new location

system("cls"); // Clear the screen

}

else system("cls");

}//End move up

//Move down

else if (wasd == 's')

{

//If inside boundary

if(locY < ((map1.getMapY()) - 1))

{

action = false;

//Detect if player touched something

if(arrMap1[locY][locX] == p.getBullet()) // Bullet add bullet

{ p.setAmmo(1); pU->picksup(0); \_getch(); }

else if(arrMap1[locY][locX] == '+') // Health Box add health

{ p.setHealth(1); pU->picksup(1); \_getch(); }

else if(arrMap1[locY][locX] == alien.getAvatar\_Alien()) // Alien take health

{

p.setHealth(alien.getStrength\_Alien());

//Output entity attacks

sB->attacks(0);

\_getch();

}

else if(arrMap1[locY][locX] == virus.getAvatar\_Virus()) // Virus take health

{

p.setHealth(virus.getStrength\_Virus());

//Output entity attacks

sB->attacks(1);

\_getch();

}

else if(arrMap1[locY][locX] == giant.getAvatar\_Giant()) // Giant take health

{

p.setHealth(giant.getStrength\_Giant());

//Output entity attacks

sB->attacks(2);

\_getch();

}

arrMap1[locY][locX] = map1.getTile(); // Redraw tile

loc = &arrMap1[locY++][locX]; // Get new location

arrMap1[locY][locX] = p.getAvatar(); // Draw avatar on new location

system("cls"); // Clear the screen

}

else system("cls");

}//End move down

//Move left

else if (wasd == 'a')

{

//If inside boundary

if(locX > 0)

{

action = false;

//Detect if player touched something

if(arrMap1[locY][locX] == p.getBullet()) // Bullet add bullet

{ p.setAmmo(1); pU->picksup(0); \_getch(); }

else if(arrMap1[locY][locX] == '+') // Health Box add health

{ p.setHealth(1); pU->picksup(1); \_getch(); }

else if(arrMap1[locY][locX] == alien.getAvatar\_Alien()) // Alien take health

{

p.setHealth(alien.getStrength\_Alien());

//Output entity attacks

sB->attacks(0);

\_getch();

}

else if(arrMap1[locY][locX] == virus.getAvatar\_Virus()) // Virus take health

{

p.setHealth(virus.getStrength\_Virus());

//Output entity attacks

sB->attacks(1);

\_getch();

}

else if(arrMap1[locY][locX] == giant.getAvatar\_Giant()) // Giant take health

{

p.setHealth(giant.getStrength\_Giant());

//Output entity attacks

sB->attacks(2);

\_getch();

}

arrMap1[locY][locX] = map1.getTile(); // Redraw tile

loc = &arrMap1[locY][locX--]; // Get new location

arrMap1[locY][locX] = p.getAvatar(); // Draw avatar on new location

system("cls"); // Clear the screen

}

else system("cls");

}//End move left

//Move right

else if (wasd == 'd')

{

//If inside boundary

if(locX < (map1.getMapX() - 1))

{

action = false;

//Detect if player touched something

if(arrMap1[locY][locX] == p.getBullet()) // Bullet add bullet

{ p.setAmmo(1); pU->picksup(0); \_getch(); }

else if(arrMap1[locY][locX] == '+') // Health Box add health

{ p.setHealth(1); pU->picksup(1); \_getch(); }

else if(arrMap1[locY][locX] == alien.getAvatar\_Alien()) // Alien take health

{

p.setHealth(alien.getStrength\_Alien());

//Output entity attacks

sB->attacks(0);

\_getch();

}

else if(arrMap1[locY][locX] == virus.getAvatar\_Virus()) // Virus take health

{

p.setHealth(virus.getStrength\_Virus());

//Output entity attacks

sB->attacks(1);

\_getch();

}

else if(arrMap1[locY][locX] == giant.getAvatar\_Giant()) // Giant take health

{

p.setHealth(giant.getStrength\_Giant());

//Output entity attacks

sB->attacks(2);

\_getch();

}

arrMap1[locY][locX] = map1.getTile(); // Redraw tile

loc = &arrMap1[locY][locX++]; // Get new location

arrMap1[locY][locX] = p.getAvatar(); // Draw avatar on new location

system("cls"); // Clear the screen

}

else system("cls");

}//End move right

/\* PLAYER ACTIONS \*/

//Shooting

//shoot up

if(wasd == 'i')

{

action = true;

//Check if still has ammo

if(p.getAmmo() <= 0)

cout << "\nNo more ammo!\n";

else if(p.getAmmo() > 0)

{

//Remove bullet

p.setAmmo(-1);

//Shoot entity if present

//Alien

if(arrMap1[locY - 1][locX] == alien.getAvatar\_Alien())

{

//Tell player attacking alien

pB->attacks(alien.getAvatar\_Alien());

//Take away 1 health point from alien

alien.setHealth\_Alien(-1);

cout << "\nAlien killed!\n\n";

//Give points

p.setPts(alien.getPoints\_Alien());

//Remove alien from map

arrMap1[locY - 1][locX] == map1.getTile();

//Reset alien health for other aliens

alien.setHealth\_Alien(1);

}//End murdering Alien

//Virus

else if(arrMap1[locY - 1][locX] == virus.getAvatar\_Virus())

{

//Tell player attacking virus

pB->attacks(virus.getAvatar\_Virus());

//Show and remove virus health

cout << "\nVirus Health: " << virus.getHealth\_Virus()

<< endl << endl;

virus.setHealth\_Virus(-1);

//If virus killed

if(virus.getHealth\_Virus() == 0)

{

cout << "\nVirus Killed!\n\n";

//Give points

p.setPts(virus.getPoints\_Virus());

//Remove virus from map

arrMap1[locY - 1][locX] == map1.getTile();

//Reset virus health for other viruses

virus.setHealth\_Virus(3);

}

}//End murdering Virus

//Giant

else if(arrMap1[locY - 1][locX] == giant.getAvatar\_Giant())

{

//Tell player attacking giant

pB->attacks(giant.getAvatar\_Giant());

//Show and remove giant health

cout << "\nGiant Health: " << giant.getHealth\_Giant()

<< endl << endl;

giant.setHealth\_Giant(-1);

//If giant killed

if(giant.getHealth\_Giant() == 0)

{

cout << "\nGiant Killed!\n\n";

//Give points

p.setPts(giant.getPoints\_Giant());

//Remove Giant from map

arrMap1[locY - 1][locX] == map1.getTile();

//Reset Giant health for other giants

giant.setHealth\_Giant(5);

}

}//End murdering Giant

else cout << endl;

}//End if p.getAmmo() > 0

}//End shoot up

//shoot down

else if(wasd == 'k')

{

action = true;

//Check if still has ammo

if(p.getAmmo() <= 0)

cout << "\nNo more ammo!\n";

else if(p.getAmmo() > 0)

{

//Remove bullet

p.setAmmo(-1);

//Shoot entity if present

//Alien

if(arrMap1[locY + 1][locX] == alien.getAvatar\_Alien())

{

//Tell player attacking alien

pB->attacks(alien.getAvatar\_Alien());

//Take away 1 health point from alien

alien.setHealth\_Alien(-1);

cout << "\nAlien killed!\n\n";

//Give points

p.setPts(alien.getPoints\_Alien());

//Remove alien from map

arrMap1[locY + 1][locX] == map1.getTile();

//Reset alien health for other aliens

alien.setHealth\_Alien(1);

}//End murdering Alien

//Virus

else if(arrMap1[locY + 1][locX] == virus.getAvatar\_Virus())

{

//Tell player attacking virus

pB->attacks(virus.getAvatar\_Virus());

//Show and remove virus health

cout << "\nVirus Health: " << virus.getHealth\_Virus()

<< endl << endl;

virus.setHealth\_Virus(-1);

//If virus killed

if(virus.getHealth\_Virus() == 0)

{

cout << "\nVirus Killed!\n\n";

//Give points

p.setPts(virus.getPoints\_Virus());

//Remove virus from map

arrMap1[locY + 1][locX] == map1.getTile();

//Reset virus health for other viruses

virus.setHealth\_Virus(3);

}

}//End murdering Virus

//Giant

else if(arrMap1[locY + 1][locX] == giant.getAvatar\_Giant())

{

//Tell player attacking giant

pB->attacks(giant.getAvatar\_Giant());

//Show and remove giant health

cout << "\nGiant Health: " << giant.getHealth\_Giant()

<< endl << endl;

giant.setHealth\_Giant(-1);

//If giant killed

if(giant.getHealth\_Giant() == 0)

{

cout << "\nGiant Killed!\n\n";

//Give points

p.setPts(giant.getPoints\_Giant());

//Remove Giant from map

arrMap1[locY + 1][locX] == map1.getTile();

//Reset Giant health for other giants

giant.setHealth\_Giant(5);

}

}//End murdering Giant

else cout << endl;

}//End if p.getAmmo() > 0

}//End shoot down

//shoot left

else if(wasd == 'j')

{

action = true;

//Check if still has ammo

if(p.getAmmo() <= 0)

cout << "\nNo more ammo!\n";

else if(p.getAmmo() > 0)

{

//Remove bullet

p.setAmmo(-1);

//Shoot entity if present

//Alien

if(arrMap1[locY][locX - 1] == alien.getAvatar\_Alien())

{

//Tell player attacking alien

pB->attacks(alien.getAvatar\_Alien());

//Take away 1 health point from alien

alien.setHealth\_Alien(-1);

cout << "\nAlien killed!\n\n";

//Give points

p.setPts(alien.getPoints\_Alien());

//Remove alien from map

arrMap1[locY][locX] == map1.getTile();

//Reset alien health for other aliens

alien.setHealth\_Alien(1);

}//End murdering Alien

//Virus

else if(arrMap1[locY][locX - 1] == virus.getAvatar\_Virus())

{

//Tell player attacking virus

pB->attacks(virus.getAvatar\_Virus());

//Show and remove virus health

cout << "\nVirus Health: " << virus.getHealth\_Virus()

<< endl << endl;

virus.setHealth\_Virus(-1);

//If virus killed

if(virus.getHealth\_Virus() == 0)

{

cout << "\nVirus Killed!\n\n";

//Give points

p.setPts(virus.getPoints\_Virus());

//Remove virus from map

arrMap1[locY][locX - 1] == map1.getTile();

//Reset virus health for other viruses

virus.setHealth\_Virus(3);

}

}//End murdering Virus

//Giant

else if(arrMap1[locY][locX - 1] == giant.getAvatar\_Giant())

{

//Tell player attacking giant

pB->attacks(giant.getAvatar\_Giant());

//Show and remove giant health

cout << "\nGiant Health: " << giant.getHealth\_Giant()

<< endl << endl;

giant.setHealth\_Giant(-1);

//If giant killed

if(giant.getHealth\_Giant() == 0)

{

cout << "\nGiant Killed!\n\n";

//Give points

p.setPts(giant.getPoints\_Giant());

//Remove Giant from map

arrMap1[locY][locX - 1] == map1.getTile();

//Reset Giant health for other giants

giant.setHealth\_Giant(5);

}

}//End murdering Giant

else cout << endl;

}//End if p.getAmmo() > 0

}//End shoot left

//shoot right

else if(wasd == 'l')

{

action = true;

//Check if still has ammo

if(p.getAmmo() <= 0)

cout << "\nNo more ammo!\n";

else if(p.getAmmo() > 0)

{

//Remove bullet

p.setAmmo(-1);

//Shoot entity if present

//Alien

if(arrMap1[locY][locX + 1] == alien.getAvatar\_Alien())

{

//Tell player attacking alien

pB->attacks(alien.getAvatar\_Alien());

//Take away 1 health point from alien

alien.setHealth\_Alien(-1);

cout << "\nAlien killed!\n\n";

//Give points

p.setPts(alien.getPoints\_Alien());

//Remove alien from map

arrMap1[locY][locX + 1] == map1.getTile();

//Reset alien health for other aliens

alien.setHealth\_Alien(1);

}//End murdering Alien

//Virus

else if(arrMap1[locY][locX + 1] == virus.getAvatar\_Virus())

{

//Tell player attacking virus

pB->attacks(virus.getAvatar\_Virus());

//Show and remove virus health

cout << "\nVirus Health: " << virus.getHealth\_Virus()

<< endl << endl;

virus.setHealth\_Virus(-1);

//If virus killed

if(virus.getHealth\_Virus() == 0)

{

cout << "\nVirus Killed!\n\n";

//Give points

p.setPts(virus.getPoints\_Virus());

//Remove virus from map

arrMap1[locY][locX + 1] == map1.getTile();

//Reset virus health for other viruses

virus.setHealth\_Virus(3);

}

}//End murdering Virus

//Giant

else if(arrMap1[locY][locX + 1] == giant.getAvatar\_Giant())

{

//Tell player attacking giant

pB->attacks(giant.getAvatar\_Giant());

//Show and remove giant health

cout << "\nGiant Health: " << giant.getHealth\_Giant()

<< endl << endl;

giant.setHealth\_Giant(-1);

//If giant killed

if(giant.getHealth\_Giant() == 0)

{

cout << "\nGiant Killed!\n\n";

//Give points

p.setPts(giant.getPoints\_Giant());

//Remove Giant from map

arrMap1[locY][locX + 1] == map1.getTile();

//Reset Giant health for other giants

giant.setHealth\_Giant(5);

}

}//End murdering Giant

else cout << endl;

}//End if p.getAmmo() > 0

}//End shoot right

//If player us dead

else if(p.getHealth() == 0)

{

cout << "\nYou Have Died!!\n\n"

<< "Final Score: " << p.getPts() << endl;

break;

}

//If wrong character, cls to avoid duplicating screen

else system("cls");

//Display map

for (int y = 0; y < map1.getMapY(); y++)

{

for (int x = 0; x < map1.getMapX(); x++)cout << arrMap1[y][x]<<" ";

cout<<endl;

}

//Clear some entities off the map

arrMap1[randnumY(&locY, &locX)][randnumX(&locY, &locX)] = map1.getTile();

//Display player info

cout << p;

}while(isGameRunning); //End game loop

//Free allocated memory

cout << "Exiting";

for(int i = 0; i < map1.getMapY(); ++i)

{

cout << ".";

delete [] arrMap1;

}

delete [] arrMap1;

delete sB;

delete pB;

delete pU;

//Exit program

system("PAUSE");

return 0;

}//End method main

//Function Prototypes

//Start method randnumX

int randnumX(int \*y, int \*x)

{

//Declare function variables

int randN; // Holds random number

//Set random number seed

srand(time(0));

//If denominator will be 0, return var y

if(((\*x + 1) - \*y) <= 0)return \*y;

else

//Keeps number between locX and locY

randN = \*y + rand() % ((\*x + 1) - \*y);

//Return the random number

return randN;

}//End method randnumX

//Start method randnumY

int randnumY(int \*y, int \*x)

{

//Declare function variables

int randN; // Holds random number

//If denominator will be 0, return var y

if(((\*x + 1) - \*y) <= 0)return \*y;

else

//Keeps number between locX and locY

randN = \*y + rand() % ((\*x + 1) - \*y);

//Return the random number

return randN;

}//End method randnumY