gg1126
Guy Goudeau
October 13, 2015

ADPG-105 Assessment

I.1. Description of the Problem

a.) Name: No Splash Screen or High Scores
Problem Statement: I don't have a splash screen or high scores coded.
Problem Specification: I don't need to load anything nor do I have high scores involved due to the nature of my game, so I didn't code them in.

b.) Name: No Dynamic Memory
Problem Statement: I couldn't figure out a way to efficiently use dynamic memory.
Problem Specification: I finished my program without finding a way to use dynamic memory and am not sure where it could help me out more than not having it.

I.2.1. Input Streams

a.) Name: file2
Description: Used to open MyLog.txt file to write to. Purpose is for the player to write in it should they need help later.
Format: Display every line of the file one after another
Size: Any number

I.2.2. Input Items

a.) Description: character variable named dirInput to determine which way the robot moves
Type: char
Range of Acceptable Values: w, a, s, d, e, q, 1, 2, 3, 4, j, k, l, p

b.) Description: character variable named input with size 100 so the player can write to MyLog.txt file
Type: char
Range of Acceptable Values: Any enterable character

c.) Description: z1Alive boolean to determine if Zambo1 is alive or not
Type: bool
Range of Acceptable Values: true, false

d.) Description: z2Alive boolean to determine if Zambo2 is alive or not
Type: bool
Range of Acceptable Values: true, false

e.) Description: z3Alive boolean to determine if Zambo3 is alive or not
Type: bool
Range of Acceptable Values: true, false

I.3.1 Output Streams

a. Name: file

Description: Used to open a text file to look at what was recorded in it. Purpose is for the player to look at what they wrote in it if they need help.

Format: Whatever is inputted gets appended to the end of the MyLog.txt file

Size: Any number

I.3.2. Output Items

a.) Description: move() function to output where the player is located

Type: int function

Range of Acceptable Values: n/a

b.) Description: character variable named c with size 100 to store and output whatever was written to the MyLog.txt file

Type: char

Range of Acceptable Values: Any enterable character

1.4.1 Description

The user is shown the following: “Welcome to Wumpus World. There are no Wumpi. Kill all the zombies to win. You have (# of bullets) bullets, use them wisely. Don't go out of bounds or you die!” They also see a 4x4 grid with x and y coordinates. It also says “Use W, A, S, D to move, press E to shoot, or press Q to quit. Enter 1 to face North, 2 to face South, 3 to face East or 4 to face West. Press J to look at your log, K to write to it or L to clear it. Press P to check your current position.” They are then able to enter a character that would perform one of the actions they were told of.

II.2 Information About the Objects

a.) Name: Hunter

Description: a class to deal with the player’s ammo supply

Class Attributes: Hunter(int), int ammo=5

Type: class

Range of Acceptable Values: all positive integers

b.) Name: CellLocation

Description: a structure including two integers to determine player’s position

Attributes: int x, int y

Type: struct

Range of Acceptable Values: all integers

c.) Name: Player

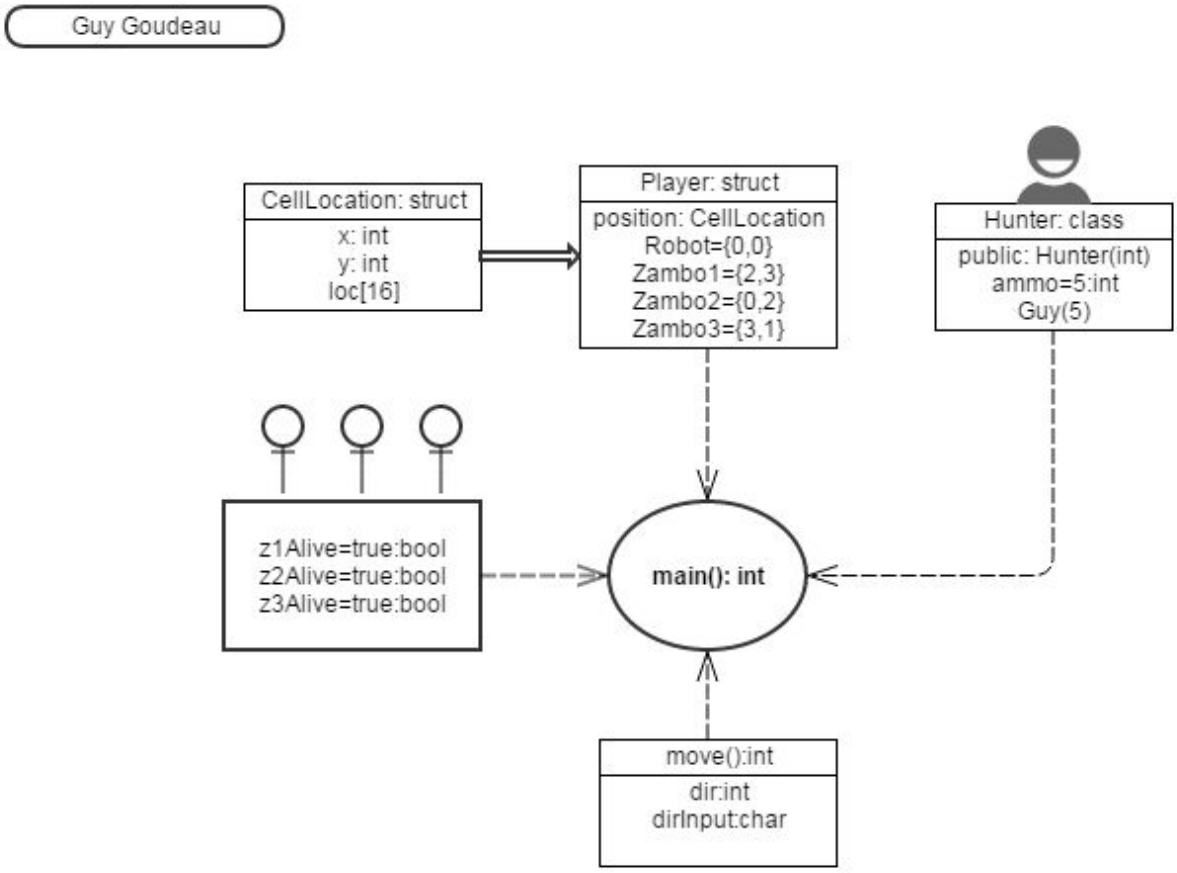
Description: contains position variable of struct CellLocation

Attributes: CellLocation position

Type: struct

Range of Acceptable Values: all integers

II.4 Design Diagrams



III.1 Source Code

Header:

```
struct CellLocation          // create new struct to determine location of cells
{
        int x;               // create variable called x
        int y;               // create variable called y
};

struct Player                // create new struct for the player
{
        CellLocation position;              // call earlier struct to determine player's position
};
```

```
class Hunter      // create new class called Hunter
{
public:   // public data as follows
          Hunter(int);       // Give Hunter 1 parameter for ammo
          int ammo = 5;    // int ammo given value of 5 meaning Hunter has 5 bullets

private:            // no private data
};

Hunter::Hunter(int a)      // hunter constructor
{
          ammo = a;        // ammo = a;
}
```

Source:

```
#include <iostream>      // calling a library
#include <cstdlib>       // calling a library
#include <fstream>       // calling a library
#include "Header.h"      // calling a library
using namespace std;     // using a namespace so you don't have to type std::

bool z1Alive = true;       // Zambo1 is set to be alive by default
bool z2Alive = true;       // Zambo2 is set to be alive by default
bool z3Alive = true;       // Zambo3 is set to be alive by default
int dir;             // integer for direction. 1 is North, 2 is South, 3 is East, 4 is West
Player Robot = { { 0, 0 } };          // make the player using player struct to start at postion 0,0
Player Zambo1 = { { 2, 3 } };       // initialize a zombie to position 2,3
Player Zambo2 = { { 0, 2 } };       // initialize a zombie to position 0,2
Player Zambo3 = { { 3, 1 } };       // initialize a zombie to position 3,1
CellLocation loc[16];      // create array of 16 cells to use in grid generation
Hunter Guy(5);             // Make a Hunter class called Guy with 5 bullets
ifstream file;             // initialize an in stream called file
ofstream file2;            // initialize an out stream called file2
char c[100];               // character array c of size 100 for storing in file
char input[100];           // used to store input into file

void generateGrid(int rows, int cols, CellLocation c[])        // create new function to actually create the grid
{
          for (int i = 0; i < rows; i++)          // for loop for rows
          {
                    for (int j = 0; j < cols; j++)// for loop for columns
                    {
                              c[i].x = i;          // Struct CellLocation or c of x is the same as the for loop's i
                              c[j].y = j;          // Struct CellLocation or c of y is the same as the for loop's j
                              cout << c[i].x << c[j].y << " ";       // output the 4x4 grid
                    }
```

```
                cout << endl;      // end line
        }
}

int move()          // create new function to use for moving the player
{
        cout << "Welcome to Wumpus World. There are no Wumpi. Kill all the zombies to win. " << endl;
// Welcome statement
        cout << "You have "<< Guy.ammo <<" bullets, use them wisely. " << endl;  // Info statement
        cout << "Don't go out of bounds or you die! " << endl << endl;         // Warning statement
        generateGrid(4, 4, loc);   // Arguments of 4 rows, 4 columns, using struct array of 16
        cout << endl;      // end the line
        cout << "Use W, A, S, D to move, press E to shoot, or press Q to quit. " << endl;      // choose initial
move
        cout << "Enter 1 to face North, 2 to face South, 3 to face East or 4 to face West" << endl;
// inform about direction changing
        cout << "Press J to look at your log, K to write to it or L to clear it." << endl; // inform about log
        cout << "Press P to check your current position." << endl << endl; // inform about game mechanic
        char dirInput;              // create a new character variable for input to move
        cin >> dirInput;            // enter the variable
        dir = 2;                    // starting direction is South


        if (dirInput == 'w')        // if you entered the w key...
        {
                Robot.position.x--;         // x value decrements
                cout << "Your position is " << Robot.position.x << Robot.position.y << ". " << "You're facing
North. "; // tells x and y, you face north
                dir = 1;            // int dir = 1, meaning north
        }
        if (dirInput == 's')        // if you entered the s key...
        {
                Robot.position.x++;         // x value increments
                cout << "Your position is " << Robot.position.x << Robot.position.y << ". " << "You're facing
South. "; //tells x and y, you face south
                dir = 2;            // int dir = 2, meaning south
        }
        if (dirInput == 'd')        // if you entered the d key...
        {
                Robot.position.y++;         // y value increments
                cout << "Your position is " << Robot.position.x << Robot.position.y << ". " << "You're facing
East. ";//tells x and y, you face east
                dir = 3;            // int dir = 3, meaning east
        }
        if (dirInput == 'a')        // if you entered the a key...
        {
                Robot.position.y--;         // y value decrements
```

```
                cout << "Your position is " << Robot.position.x << Robot.position.y << ". " << "You're facing
West. ";                        //tells x and y, you face west
                dir = 4;                // int dir = 4, meaning west
        }


        if (dirInput == '1')            // if you entered the 1 key...
        {
                cout << "You face North. ";          // output direction change
                dir = 1;                // int dir = 1, meaning north
        }
        if (dirInput == '2')            // if you entered the 2 key...
        {
                cout << "You face South. ";          // output direction change
                dir = 2;                // int dir = 2, meaning south
        }
        if (dirInput == '3')            // if you entered the 3 key...
        {
                cout << "You face East. ";           // output direction change
                dir = 3;                // int dir = 3, meaning east
        }
        if (dirInput == '4')            // if you entered the 4 key...
        {
                cout << "You face West. ";           // output direction change
                dir = 4;                // int dir = 4, meaning west
        }


        if (dirInput == 'q')            // if you entered the q key...
        {
                cout << "Quitting..." << endl;       // output quitting
                return 0;               // end program
        }
        if (dirInput == 'e')            // if you entered the e key...
        {
                if (Guy.ammo > 0)           // if you have more than 0 bullets, continue
                {
                        Guy.ammo--;     // decrement ammo by 1
                        cout << "You shoot your gun. You have " << Guy.ammo << " bullets remaining. "; //
tells you amount remaining bullets
                        if ( z1Alive == true && ((Robot.position.x == 1 && Robot.position.y == 3 && dir ==
2) || (Robot.position.x == 2 && Robot.position.y == 2 && dir == 3) || (Robot.position.x == 3 &&
Robot.position.y == 3 && dir == 1))) // if you're facing an alive zombie
                        {
                                z1Alive = false;  // zombie being alive is set to false
                                cout << endl;     // output an endline
                                cout << "You killed a zombie! ";    // output kill statement
                        }
```

```
                        else if ( z2Alive == true && ((Robot.position.x == 0 && Robot.position.y == 1 && dir
== 3) || (Robot.position.x == 1 && Robot.position.y == 2 && dir == 1) || (Robot.position.x == 0 &&
Robot.position.y == 3 && dir == 4))) // if youre facing another alive zombie
                        {
                                z2Alive = false;  // zombie being alive is set to false
                                cout << endl;     // output an endline
                                cout << "You killed a zombie! ";   // output kill statement
                        }
                        else if ( z3Alive == true && ((Robot.position.x == 3 && Robot.position.y == 0 && dir
== 3) || (Robot.position.x == 2 && Robot.position.y == 1 && dir == 2) || (Robot.position.x == 3 &&
Robot.position.y == 2 && dir == 4))) // if youre facing a third alive zombie
                        {
                                z3Alive = false;  // zombie being alive is set to false
                                cout << endl;     // output an endline
                                cout << "You killed a zombie! ";   // output kill statement
                        }
                        else      // if you weren't facing a zombie
                        {
                                cout << endl;     // output an endline
                                cout << "You didn't seem to hit anything. ";          // output miss statement
                        }
                }
                else      // if you have no bullets
                {
                        cout << "You have no bullets remaining. All seems hopeless... ";    // output no
bullets
                }
        }


        if (dirInput == 'j')            // if you entered the j key...
        {
                cout << "Your log reads: ";          // output statement
                //display file
                file.open("MyLog.txt", ios_base::in | ios_base::_Nocreate); // open MyLog.txt for display,
don't create
                file.getline(c, 100);         // get whatever was written in the file
                cout << c << endl;          // output that shit
                file.close();         // close the file
        }
        if (dirInput == 'k')            // if you entered the k key...
        {
                //write to file
                file2.open("MyLog.txt", ios_base::app);     // open MyLog.txt to append
                cout << "You write the following in your log: ";       // output statement
                cin.clear();          // clear
                cin.getline(input, 100);    // get an inputted line
                cin.getline(input, 100);    // get an inputted line
```

```
                file2 << input;     // store input in file
                cout << "You finish writing and close your log." << endl;     // output statement
                file2.close();      // close file
        }
        if (dirInput == 'l')            // if you entered the l key...
        {
                //clear file
                file.open("MyLog.txt", ios_base::out | ofstream::trunc);       // open MyLog.txt to truncate
                cout << "You erase everything in your log." << endl;          // output statement
                file.close();       // close file
        }

        if (dirInput == 'p')            // if you entered the p key...
        {
                cout << "Your current position is " << Robot.position.x << Robot.position.y << ". "; // state
current position
                if (dir == 1)       // if int dir is equal to 1
                {
                        cout << "You're facing North. ";    // you're facing north
                }
                if (dir == 2)       // if int dir is equal to 2
                {
                        cout << "You're facing South. ";    // you're facing south
                }
                if (dir == 3)       // if int dir is equal to 3
                {
                        cout << "You're facing East. ";     // you're facing east
                }
                if (dir == 4)       // if int dir is equal to 4
                {
                        cout << "You're facing West. ";     // you're facing west
                }
        }


        if (z1Alive == true && ((Robot.position.x == 1 && Robot.position.y == 3) || (Robot.position.x == 2 &&
Robot.position.y == 2) || (Robot.position.x == 3 && Robot.position.y == 3))) // if you go near a living zombie
        {
                cout << endl;                           // output an endline
                cout << "You smell rotten flesh...";  // output warning statement
        }
        if (z2Alive == true && ((Robot.position.x == 0 && Robot.position.y == 1) || (Robot.position.x == 1 &&
Robot.position.y == 2) || (Robot.position.x == 0 && Robot.position.y == 3))) // if you go near a living zombie
        {
                cout << endl;                           // output an endline
                cout << "You smell rotten flesh..."; // output warning statement
        }
```

```
	if (z3Alive == true && ((Robot.position.x == 3 && Robot.position.y == 0) || (Robot.position.x == 2 &&
Robot.position.y == 1) || (Robot.position.x == 3 && Robot.position.y == 2))) // if you go near a living zombie
	{
		cout << endl;				// output an endline
		cout << "You smell rotten flesh..."; // output warning statement
	}

	if (Robot.position.x == -1 || Robot.position.y == -1 || Robot.position.x == 4 || Robot.position.y == 4)
					// check to see if player went out of bounds
	{
		cout << endl;				// output an endline
		cout << "You went out of bounds, fell in a magma pit and died horribly. GAME OVER " <<
endl;	// output death statement
		return 0;				// end program
	}
	if ((z1Alive == true && Robot.position.x == 2 && Robot.position.y == 3) || (z2Alive == true &&
Robot.position.x == 0 && Robot.position.y == 2) || (z3Alive == true && Robot.position.x == 3 &&
Robot.position.y == 1))				 // if you go to a space with a living zombie in it
	{
		cout << endl;				// output an endline
		cout << "You ran into a zombie and got eaten. You're one of them now. GAME OVER" <<
endl;					// output death statement
		return 0;				// end program
	}
	if ((z1Alive == false) && (z2Alive == false) && (z3Alive == false))	// check to see if every zombie
has been killed
	{
		cout << endl;				// output an endline
		cout << "Congratulations, you killed all the zombies! YOU WIN!" << endl; // output congrats
statement
		return 0;				// end program
	}

	system("pause");			// pauses the screen
	system("cls");				// clears the screen
	move();					// repeat move function until terminated
}


int main()					// call your main function
{

	move();					// call move function

	system("pause");			// pause system
	return 0;				// return 0 and end program
}
```

IV.3 Operating Directions

1. Source.cpp located at C:\Users\Guy.Goudeau\Desktop\Homework\Wumpus 2.0\Source.cpp
   Header.h located at C:\Users\Guy.Goudeau\Desktop\Homework\Wumpus 2.0\Header.h

2. Open C:\Users\Guy.Goudeau\Desktop\Homework\Wumpus 2.0\Source.cpp in Visual Studios and build program program

3. Open C:\Users\Guy.Goudeau\Desktop\Homework\Wumpus 2.0\Debug and run the Wumpus 2.0 application/executable