Name: Rory Vaughn

Wumpus world Documentation

Teacher: Matthew Williamson

9/19/2015

Title: Wumpus world

Problem Statement: Create a text-based game that allows the user to traverse a 4x4 grid in order to find gold and return to the starting square by using senses to dodge traps and the Wumpus.

Problem Specifications: The game must allow the user to sense the Wumpus, pits, and the gold in the grid. In order to win, the user must make it to the gold and back to the starting square without stepping on a square with the live Wumpus or the pits to win the game. The player will have an arrow that can kill the Wumpus if the player is facing the Wumpus and in a square directly adjacent

Input items description: The input used for this game is an English keyboard.

The logical data type is integers.

Range of acceptable values: 0, 2, 4, 6, 8

Output stream: Main game

Description: This screen houses all of the games possible actions, from beginning to end.

Format: The screen begins with text that has instructions and the objective of the game, as well as the starting state of the game.

Size: The minimum lines possible is 9 and the total is variable.

Class information:

Name: Robot

Description: A class that stores info on the player.

Name: Pit

Description: contains game mechanics to allowing the player to sense the pit and be killed by it.

Name: Pit2.

Description: contains game mechanics to allowing the player to sense the second pit and be killed by it.

Name: Wumpus

Description: Contains game mechanics allowing the player to sense, kill, or be killed by the Wumpus.

Name: Gold

Description: Contains game mechanics allowing the player to sense and receive the gold.

Class Attributes

Player Robot;

Robot.Alive = 1;

Robot.Pos\_X = 0;

Robot.Pos\_Y = 0;

Robot.Direction = 8;

Object Pit;

Pit.Exist = 1;

Pit.E\_Pos\_X = 1;

Pit.E\_Pos\_Y = 3;

Object Pit2;

Pit2.Exist = 1;

Pit2.E\_Pos\_X = 3;

Pit2.E\_Pos\_Y = 1;

Object Wumpus;

Wumpus.Exist = 1;

Wumpus.E\_Pos\_X = 2;

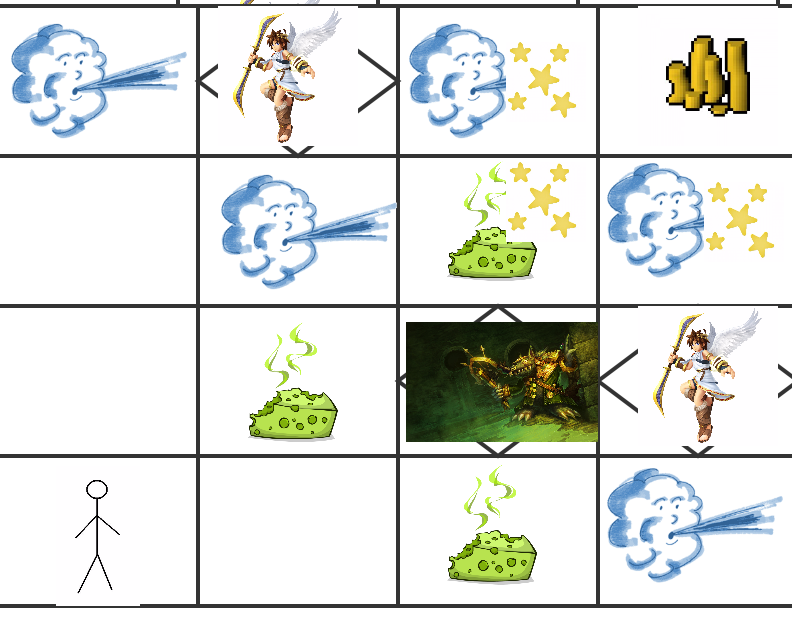
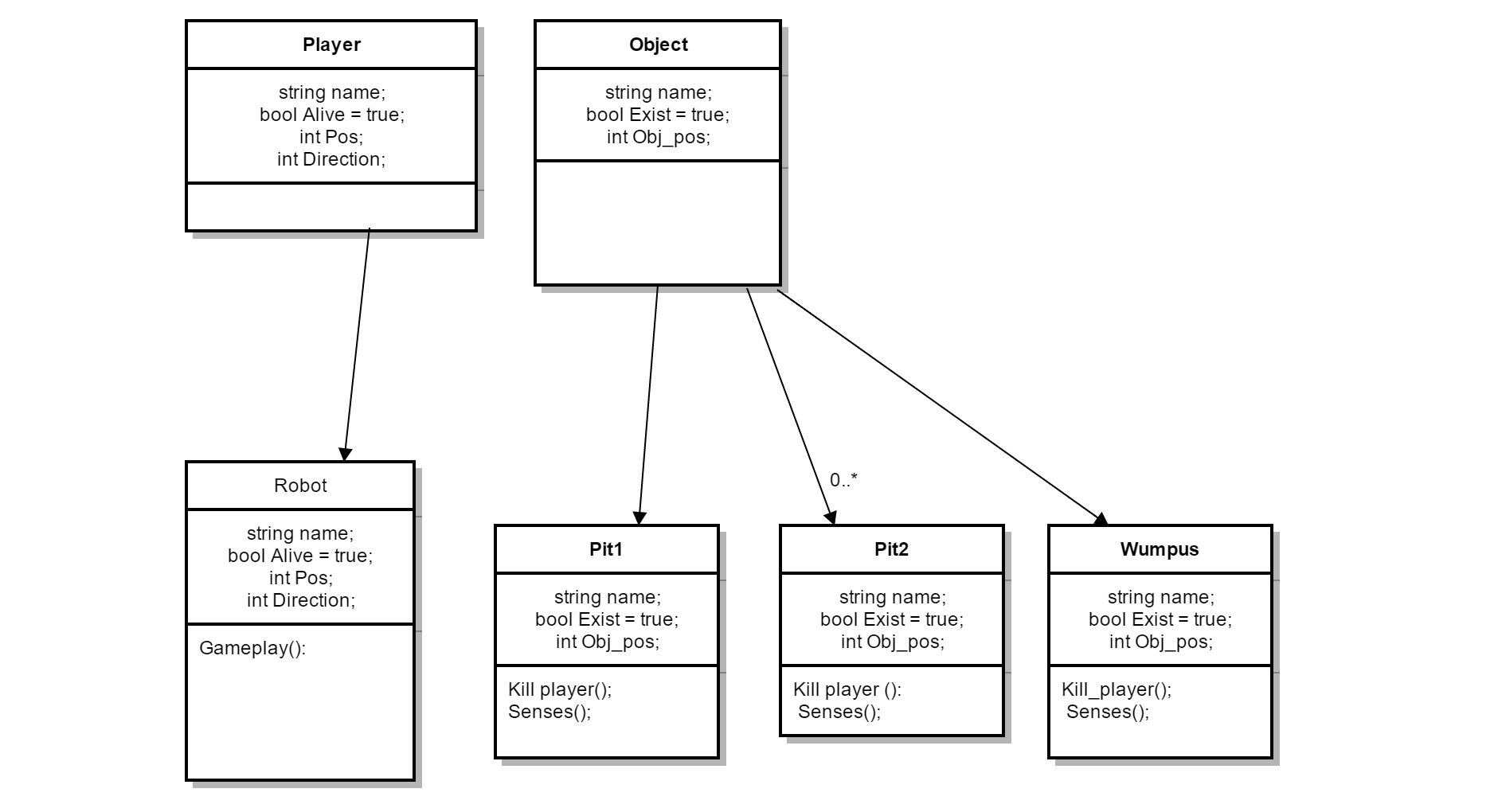
Wumpus.E\_Pos\_Y = 1;

Object Gold;

Gold.Exist = 1;

Gold.E\_Pos\_X = 3;

Gold.E\_Pos\_Y = 3;



Implementation Documentation:

The following code was implemented and tested within Visual Studio 2015.

Header:

#ifndef WUMPUS\_H

#define WUMPUS\_H

#include <iostream>

#include <string>

using namespace std;

struct Player

{

string name;

bool Alive;

int Pos\_X = 0;

int Pos\_Y = 0;

int Direction;

};

struct Object

{

string name;

bool Exist;

int E\_Pos\_X;

int E\_Pos\_Y;

};

class Robot

{

public:

Robot(int, int, int);

void Gameplay();

private:

int Pos\_X;

int Pos\_Y;

int Direction;

};

Robot::Robot(int x, int y, int d)

{

int Pos\_X = x;

int Pos\_Y = y;

int Direction = d;

}

void Robot::Gameplay()

{

Player Robot;

Robot.Alive = 1;

Robot.Pos\_X = 0;

Robot.Pos\_Y = 0;

Robot.Direction = 8;

Object Pit;

Pit.Exist = 1;

Pit.E\_Pos\_X = 1;

Pit.E\_Pos\_Y = 3;

Object Pit2;

Pit2.Exist = 1;

Pit2.E\_Pos\_X = 3;

Pit2.E\_Pos\_Y = 1;

Object Wumpus;

Wumpus.Exist = 1;

Wumpus.E\_Pos\_X = 2;

Wumpus.E\_Pos\_Y = 1;

Object Gold;

Gold.Exist = 1;

Gold.E\_Pos\_X = 3;

Gold.E\_Pos\_Y = 3;

int arrow = 1;

while (Robot.Alive == true)

{

int input;

cin >> input;

switch (input)

{

case 8:

{

cout << "You face North." << endl;

Robot.Direction = 8;

break;

}

case 2:

{

cout << "You face South." << endl;

Robot.Direction = 2;

break;

}

case 4:

{

cout << "You face West." << endl;

Robot.Direction = 4;

break;

}

case 6:

{

cout << "You face East." << endl;

Robot.Direction = 6;

break;

}

case 5:

{

if (arrow == 1)

{

cout << "You shoot the arrow." << endl;

arrow = arrow - 1;

if (Robot.Pos\_X == 1)

{

if (Robot.Pos\_Y == 1)

{

if (Robot.Direction = 6)

{

cout << "You killed the Wumpus" << endl;

Wumpus.Exist = Wumpus.Exist - 1;

}

}

}

else if (Robot.Pos\_X == 2)

{

if (Robot.Pos\_Y == 0)

{

if (Robot.Direction = 8)

{

cout << "You killed the Wumpus" << endl;

Wumpus.Exist = Wumpus.Exist - 1;

}

}

}

else if (Robot.Pos\_X == 2)

{

if (Robot.Pos\_Y == 2)

{

if (Robot.Direction = 2)

{

cout << "You killed the Wumpus" << endl;

Wumpus.Exist = Wumpus.Exist - 1;

}

}

}

else

{

cout << "Your arrow misses." << endl;

}

}

else

{

cout << "You have no more arrows." << endl;

}

break;

}

case 0:

{

switch (Robot.Direction)

{

case 2:

{

Robot.Pos\_Y = Robot.Pos\_Y--;

cout << "you new position is (" << Robot.Pos\_X << ", " << Robot.Pos\_Y << ")" << endl;

break;

}

case 4:

{

Robot.Pos\_X = Robot.Pos\_X--;

cout << "you new position is (" << Robot.Pos\_X << ", " << Robot.Pos\_Y << ")" << endl;

break;

}

case 6:

{

Robot.Pos\_X = Robot.Pos\_X++;

cout << "you new position is (" << Robot.Pos\_X << ", " << Robot.Pos\_Y << ")" << endl;

break;

}

case 8:

{

Robot.Pos\_Y = Robot.Pos\_Y++;

cout << "you new position is (" << Robot.Pos\_X << ", " << Robot.Pos\_Y << ")" << endl;

break;

}

}

break;

}

default:

{

cout << "Incorrect input \nPlease enter a number listed in the controls." << endl;

}

}

if (Gold.Exist == true)

{

if (Robot.Pos\_X == 2)

{

if (Robot.Pos\_Y == 3)

{

cout << "You see some glitter" << endl;

}

}

if (Robot.Pos\_X == 3)

{

if (Robot.Pos\_Y == 2)

{

cout << "You see some glitter" << endl;

}

}

if (Robot.Pos\_X == 1)

{

if (Robot.Pos\_Y == 2)

{

cout << "You feel a slight breeze" << endl;

}

}

if (Robot.Pos\_X == 0)

{

if (Robot.Pos\_Y == 3)

{

cout << "You feel a slight breeze" << endl;

}

}

if (Robot.Pos\_X == 2)

{

if (Robot.Pos\_Y == 3)

{

cout << "You feel a slight breeze" << endl;

}

}

if (Robot.Pos\_X == 2)

{

if (Robot.Pos\_Y == 1)

{

cout << "You feel a slight breeze" << endl;

}

}

if (Robot.Pos\_X == 3)

{

if (Robot.Pos\_Y == 0)

{

cout << "You feel a slight breeze" << endl;

}

if (Robot.Pos\_Y == 2)

{

cout << "You feel a slight breeze" << endl;

}

}

if (Robot.Pos\_X == Gold.E\_Pos\_X)

{

if (Robot.Pos\_Y == Gold.E\_Pos\_Y)

{

cout << "You Found the Gold! \nGo back to the starting square to win!\nBe careful. You will no longer be able to sense the traps." << endl;

Gold.Exist = 0;

}

}

}

if (Gold.Exist == 0)

{

if (Robot.Pos\_X == 0)

{

if (Robot.Pos\_Y == 0)

{

cout << "You won!\n" << endl;

break;

}

}

}

if (Robot.Pos\_X >= 4 || Robot.Pos\_X < 0)

{

cout << "You went out of the grid and died!" << endl;

Robot.Alive = false;

break;

}

if (Robot.Pos\_Y >= 4 || Robot.Pos\_Y < 0)

{

cout << "You went out of the grid and died!" << endl;

Robot.Alive = false;

break;

}

if (Robot.Pos\_X == 1)

{

if (Robot.Pos\_Y == 3)

{

cout << "You fell into the Wumpus' trap and died!!" << endl;

Robot.Alive = false;

break;

}

}

if (Wumpus.Exist == 1)

{

if (Robot.Pos\_X == 1)

{

if (Robot.Pos\_Y == 1)

{

cout << "You smell a terrible smell." << endl;

}

}

if (Robot.Pos\_X == 2)

{

if (Robot.Pos\_Y == 2)

{

cout << "You smell a terrible smell." << endl;

}

if (Robot.Pos\_Y == 1)

{

cout << "You were slain by the Wumpus!" << endl;

Robot.Alive = false;

}

if (Robot.Pos\_Y == 0)

{

cout << "You smell a terrible smell." << endl;

}

}

}

if (Robot.Pos\_X == 3)

{

if (Robot.Pos\_Y == 1)

{

cout << "You fell into the Wumpus' trap and died!" << endl;

Robot.Alive = false;

break;

}

}

}

}

#endif WUMPUS\_H

Sourse code:

#include "Wumpus.h"

Void MakeGrid()

{

const int i = 4;

const int h = 4;

int grid[i][h] = {};

for (int i = 0; i < 4; i++)

{

for (int h = 0; h < 4; h++)

{

}

}

}

int main()

{

cout << "Welcome to Wumpus World! \nYou can travel in a 4x4 grid of squares.\nWatch out for the Wumpus and the Traps he set up!\n";

cout << "Controls : \n8: Face North, 2: Face South, 4: Face West, \n";

cout << "6: Face East, 5: Shoot the arrow, \n0: Move in the direction the character is facing. \n \n" << endl;

cout << "Your journey for gold begins in the square (0,0), facing North.\n " << endl;

MakeGrid(); //the fuction uses nested loops and an array to create a 2D grid.

int start = 1;

switch (start)

{

case 1:

{Robot Robot(0, 0, 8);

Robot.Gameplay();

break;

}

}

int retry;

cout << "enter 1 to retry or anything else to quit." << endl;

cin >> retry;

switch (retry)

{

case 1:

{

{system("cls");

return main(); }

}

default:

{return 0; }

}

system("pause"); //this line of code allows the console to stay open after the game is over.

return 0;

}

Test plan:

I will check to see if:

The movement in every square functions properly

The correct senses display in each square.

The arrow functions properly.

The senses and player death mechanic go away after the Wumpus is killed.

The gold and starting square function together to allow the player to win the game.

The pits and the Wumpus kill the player if stepped on.

Test results:

All Tests were completed and no errors were found during the testing of all of these features.

Operating directions:

1: Open the “Wumpus Assessment” folder

2: Open the “Wumpus Assessment folder” (This is not an unintended repeat.)

3: Open the “x64” folder.

4: Open the “Debug” folder.

5: Open the “Wumpus Assessment” Application.