Jeremy Barzas
10/7/2015
ADGP101

A. Requirements Documentation

1. Name: Wumpus World

Problem Statement:
The player must navigate their robot character around the 4x4 grid and get the gold then return the the starting point to win.

Problem Specification:
If the player’s robot character moves to a cell that has a wumpus or a pit you lose. if you are on a cell that is next to a pit, gold , or wumpus a warning message of breeze, shimmer, or stench displays to let you know that one or more dangers are near.

B. Input Methods

1. Description: cin is the method i used to get input from the user to move the character.

2. Type: I used the “char” variable type to accept the input of ‘w’,’a’,’s’,’d’, as the arrow keys to move.

C. Output Items:

Description: cout is the output method I used to display the games information to the screen. In the game the user will first see "You enter a dark cave which direction do you go?" and be prompted to give and input of ‘w’,’a’,’s’,’d’.

Type: char, int, and strings will be printed to the screen.

D. User Interface Information

1. Description: The player is show the message "You enter a dark cave in search for gold...". Then they will be prompted to input a direction with ‘w’,’a’,’s’,’d’. The player will not be able to see the grid of cells at all and have to navigate based on a set of “X” and “Y” coordinates.

Objective:
The player must get to the cell with the gold without moving to a cell with a pit or wumpus and then return to the start location to win.

Danger:
Wumpus - Creature that will destroy your character if you move on a cell that contains it.
Pit - Hole in the ground that will destroy your character if you move to a cell that contains it.
Ledge - If you walk off the grid you will die.

E. Design Documentation

1. System Architecture Description

The objects used in the game are a cell class, a player class, a gold class, a pit class, a wumpus class, and a position struct. These are located in the header file.

2. Information about the Objects

I. Name: position

Type: Struct

Description: this struct gives an x and y variable to be used as points on a grid for other objects to have location.

Attributes: two integers x and y

II. Name: Player

Type: Class

Description: A class to make the player's character.

Attributes: Two bools , a position struct, and a Player constructor function.

III. Name: Gold

Type: Class

Description: A class to make the Gold item.

Attributes: A position struct, and a Gold constructor function.

IV. Name: Wumpus

Type: Class

Description: A class to make the Wumpus creature.

Attributes: A position struct, and a Wumpus constructor function.

V. Name: Pit

Type: Class

Description: A class to make the Pit hazzard.

Attributes: A position struct, and a Pit constructor function.

F. Implementation Documentation

1. Source Code

File Type: Header

Name: Classes.h

```
#ifndef CLASSES_H
#define CLASSES_H

#include <iostream>

using namespace std;

// creates a struct that contains two intergers for other classes to use as a position for location.
struct position
{
	int x;
	int y;
};

// creates a class that is a grid cell that contains a position struct for location,
// contains a fucntion to create cells to build the grid with.
class Cell
{
public:
	position location;
	Cell();
};
```

```
// creates a class for the players character that contains a position struct for location,
// contains a function to create a player character,
// contains two bool variables to hold the information on wether the player is alive or has gold.
class Player
{
public:
        position location;
        bool alive;
        bool gold;
        Player();
};

// creates a class for the Gold that contains a position struct for location.
// contains a function to create the gold.
class Gold
{
public:
        position location;
        Gold();
};

// creates a class for the Wumpus that contains a position struct for location.
// contains a function to create the Wumpus.
class Wumpus
{
public:
        position location;
        Wumpus();
};

// creates a class for the Wumpus that contains a position struct for location.
// contains a fucntion to create the Wumpus and give it position.
class Pit
{
public:
        position location;
        Pit(position);
};

#endif
```

File Type: CPP

Name: Functions.cpp

```
#include "Classes.h"

// constructor function for the Cell class.
Cell::Cell()
{
}

// constructor function for the Player class.
// defines the Players position on creation as being at (0, 0) to give it a fixed start location.
// defines the Player on creation as alive.
// defines the Player on creation as not having the gold.
Player::Player()
{
	location = { 0,0 };
	alive = true;
	gold = false;
}

// constructor function for the Gold class.
// defines the Golds position on creation as being at (3, 1) to give a fixed start location.
Gold::Gold()
{
	location = { 3,1 };
}

// constructor function for the Wumpus class.
// defines the Wumpus position on creation as being at (2, 1) to give a fixed start location.
Wumpus::Wumpus()
{
	location = { 2,1 };
}

// constructor function for the Pit class.
// function takes in the argument of a position struct to be bale to make multiple pits at different locations.
Pit::Pit(position p)
{
	location = p;
}
```

File Type: CPP

Name: Game.cpp

```
#include "Classes.h"

// creates a function that creates a grid and returns void.
// takes in the arguments of two intergers and an array of instances of the Cell class.
// loops the amount of times that is placed in the first interger argument.
// loops the amount of times that is placed in the second interger argument.
// defines the "x" location of the "i" position in the array of Cell instances as "i".
// defines the "y" location of the "j" position in the array of Cell instances as "j".
void createGrid(int rows, int cols, Cell g[])
{
	for (int i = 0; i < rows; i++)
	{
		for (int j = 0; j < cols; j++)
		{
			g[i].location.x = i;
			g[j].location.y = j;
		}
	}
}

int main()
{
	// makes a bool variable used to define the default win condition as false until changed
when certain criteria is met.
	bool winCondition = false;

	// makes a char variable used to store user input to be used in the movement switch
case statements.
	char input;

	// creates a instance of the Player class named player defined as the constructor
fucntion.
	Player player = Player();

	// creates a instance of the Gold class named gold defined as the constructor fucntion.
	Gold gold = Gold();

	// creates a instance of the Wumpus class named wumpus defined as the constructor
fucntion.
	Wumpus wumpus = Wumpus();
```

```
    // creates four different instances of the Pit class.
    // each instance of pit is defined its own fixed location.
    Pit pit1 = Pit({ 1,1 });
    Pit pit2 = Pit({ 1,3 });
    Pit pit3 = Pit({ 3,0 });
    Pit pit4 = Pit({ 3,3 });

    // creates an array named grid of size 16 of instances of the Cell class
    Cell grid[16];

    // calls the function that creates the grid to be used as the game map defined as a 4 x 4
grid stored inside the grid array.
    createGrid(4, 4, grid);

    // creates a condition to execute the the following code only if the win conditon isnt met
yet.
    if (winCondition == false)
    {
        // displays instructions for the user to the console.
        cout << "Use 'w','a','s','d' as the arrow keys to move." << endl;
        cout << "" << endl;
        cout << "You enter the cave of the Wumpus in search of gold..." << endl;
        cout << "" << endl;

        // creates the game loop.
        do
        {
            // asks the user to  eneter a direction.
            cout << "Which direction do you want to go?" << endl;
            cin >> input;
            cout << "" << endl;

            // tells the user that the wrong input was entered and lists the correct
inputs if the wrong input was entered.
            if (!(input == 'w' || input == 'a' || input == 's' || input == 'd'))
            {
                cout << "That is not a valid input..." << endl;
                cout << "Use 'w','a','s','d' as the arrow keys to move." << endl;
                cout << "" << endl;
            }

            // creates a switch statment thats accepts the input character variable.
            switch (input)
```

```
			{
				// adds 1 to the players "y" cooridinate if 'w' was the input.
			case 'w':
				player.location.x, player.location.y += 1;
				cout << "Your X, Y position is: " << player.location.x << ", "<<
player.location.y << endl;
				break;

				// adds 1 to the players "y" cooridinate if 'w' was the input.
			case 's':
				player.location.x, player.location.y -= 1;
				cout << "Your X, Y position is: " << player.location.x << ", "
<<player.location.y << endl;
				break;

				// adds 1 to the players "y" cooridinate if 'w' was the input.
			case 'a':
				player.location.x -= 1, player.location.y;
				cout << "Your X, Y position is: " << player.location.x << ", " <<
player.location.y << endl;
				break;

				// adds 1 to the players "y" cooridinate if 'w' was the input.
			case 'd':
				player.location.x += 1, player.location.y;
				cout << "Your X, Y position is: " << player.location.x << ", " <<
player.location.y << endl;
				break;

				// if no case statement is met then it breaks out of the switch.
			default:
				break;
			}

			// if the players "x" or "y" coordinate is not on the game map grid it sets
the alive bool of the player to false.
			// displays a message to the console that you have died.
			if ((player.location.x == -1) ||
				(player.location.x == 4) ||
				(player.location.y == -1) ||
				(player.location.y == 4))
			{
				player.alive = false;
```

```
					cout << "You fall of a ledge to your death..." << endl;
					cout << "" << endl;
				}

				// if the players "x" and "y" coordinates are equal to that of a Pit it sets the
alive bool of the player to false.
				// displays a message to the console that you have died.
				if ((player.location.x == pit1.location.x) &&
					(player.location.y == pit1.location.y) ||
					(player.location.x == pit2.location.x) &&
					(player.location.y == pit2.location.y) ||
					(player.location.x == pit3.location.x) &&
					(player.location.y == pit3.location.y) ||
					(player.location.x == pit4.location.x) &&
					(player.location.y == pit4.location.y))
				{
					player.alive = false;
					cout << "You fall in a giant pit and are trapped forever..." << endl;
					cout << "" << endl;
				}

				// if the players "x" and "y" coordinates are equal to that of the Wumpus it
sets the alive bool of the player to false.
				// displays a message to the console that you have died.
				if ((player.location.x == wumpus.location.x) && (player.location.y ==
wumpus.location.y))
				{
					player.alive = false;
					cout << "The Wumpus monster bites your head off.." << endl;
					cout << "" << endl;
				}

				// if the players "x" and "y" coordinates are equal to that of the Gold is sets
the gold bool of the player to true.
				// dislplays a message to the colse to tell the play that they have picked
up the gold and must leave the cave.
				if ((player.location.x == gold.location.x) && (player.location.y ==
gold.location.y))
				{
					player.gold = true;
					cout << "You have found the gold, now you must escape the
Wumpus cave" << endl;
					cout << "" << endl;
				}

				// if the players "x" and "y" coordinates are equal to the start position (0,0)
and the gold bool is true it sets the winCondition to true.
				if ((player.gold == true) && (player.location.x == 0) && (player.location.y
== 0))
				{
					winCondition = true;
				}

				// if the players alive bool is false it sets the winCondition bool to false.
				if (player.alive == false)
				{
					winCondition = false;
				}

			// while the winCondition bool is false and the players alive bool is true continue
to execute this loop.
			} while ((winCondition == false) && (player.alive == true));
		}

		//if the winCondtion bool is true and the players alive bool is true display to the console
that they have won the game.
		if ((winCondition == true) && (player.alive == true))
		{
			cout << "" << endl;
			cout << "Congradulations you escaped the cave with the gold!" << endl;
			cout << "" << endl;
		}

		system("pause");
}
```