**Super Mario Game**

****

Session: 2021 – 2025

**Submitted by:**

**Hammad Ejaz 2021-CS-102**

**Supervised by:**

**Dr. Awais Hassan**

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**CONTENT**

* **Story of game**
* **Game character description**
* **Rules and interactions**
* **Goal of the game**

**Story of game:**

Super Mario Bros is a platform game. In the game, Mario must race through the Mushroom kingdom. Mario jumps runs and walk across each level. The world are full of enemies and platforms, and open holes.

**Game character description:**

Super Mario is a 2D game.

There are a total 17 characters in the super Mario Game.

1 Mario.

16 Enemies

* The main character of the game is Mario which is controlled by the player. It can jump and walk across the levels by using arrow keys.
* The other character of game are enemies. Each of the four enemies is controlled by the computer. Which stops you to complete your game.

**Rules and Interactions:**

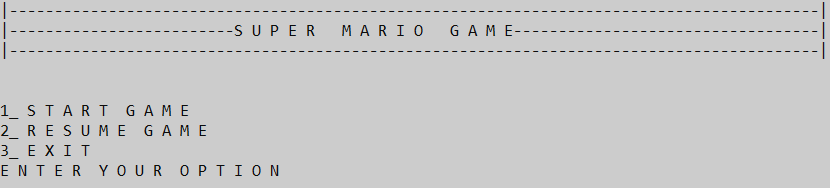
1. You have to collect coins it will increased your score and helps you to complete the level.
2. One (1) Score increases when the Mario collect one coin.
3. The game contain enemies if he collides with any of the enemies, if he fall in holes and if he strike with spikes Mario loses one life.
4. Mario can kill the enemies except “G”.
5. You have three life’s to complete your level, and to win the game.

**Goal of the game:**

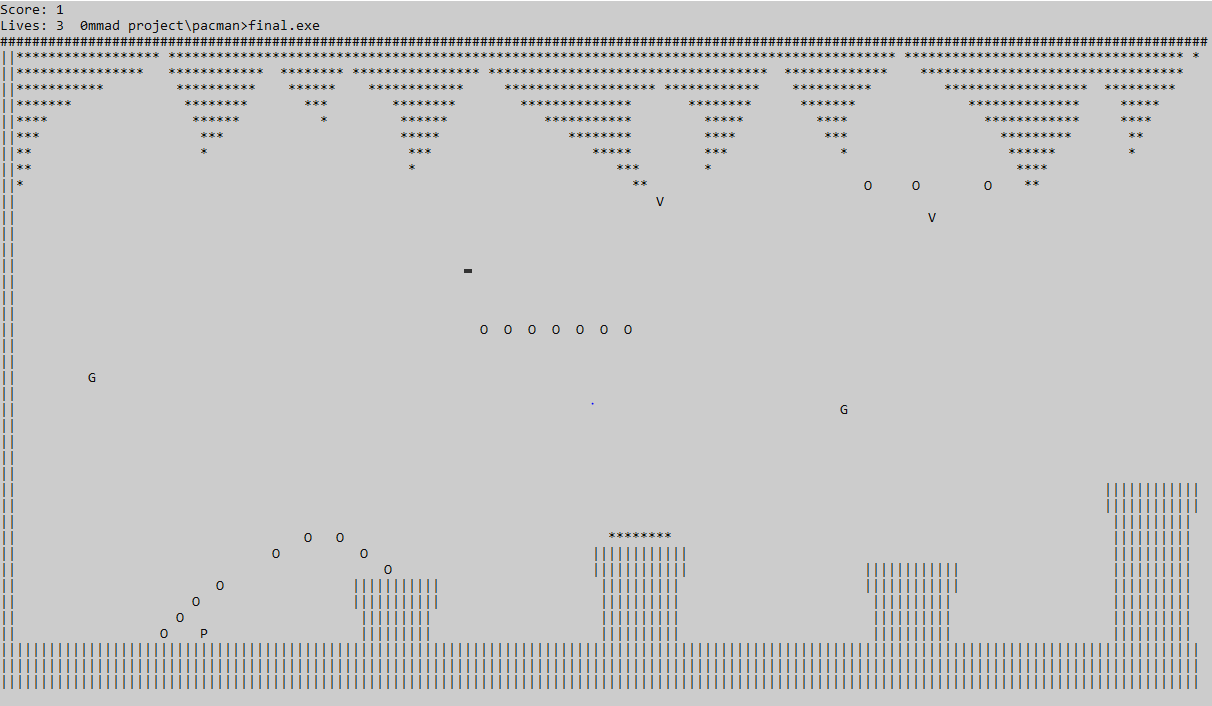
The goal of the game is to collect all of the coins until he reached to the given score that have been put across the world while avoiding the enemies.

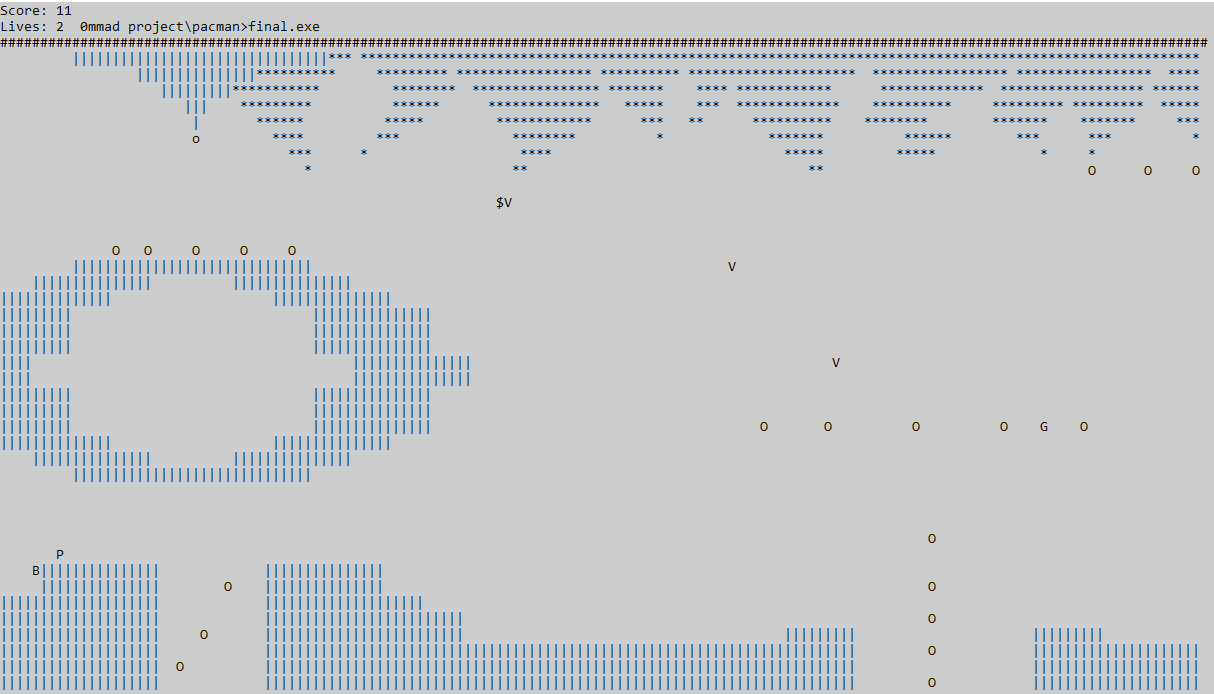
**Wire frame**s:

**Main menu:**



**Game Map;**





**Note: we will use different Character as**

1. **|| as wall**
2. **Q as Mario**
3. **V , G , B as enemy**

**Data Structures:**

bool gameRun = true;

int lives = 3;

int en = 0;

int maxScore = 5;

int dis[4];

int dis2[4];

string direction1 = "left";

string direction2 = "right";

string direction3 = "left";

string direction4 = "left";

string direction5 = "right";

string direction6 = "right";

string direction7 = "left";

string direction8 = "left";

string direction9 = "right";

string direction10 = "up";

char previousitem1 = ' ';

char previousitem2 = ' ';

char previousitem3 = ' ';

char previousitem4 = ' ';

char previousitem5 = ' ';

int old1x, old1y;

int bulletX[100], bulletY[100], pX, pY, enemy1X, enemy1Y, enemy2X, enemy2Y, enemy3X, enemy3Y, enemy4X, enemy4Y, enemy5X, enemy5Y, enemy6X, enemy6Y, enemy7X, enemy7Y, enemy8X, enemy8Y, enemy9X, enemy9Y, enemy10X, enemy10Y, Renemy1X, Renemy1Y, Renemy2X, Renemy2Y, Renemy3X, Renemy3Y, Renemy4X, Renemy4Y, Renemy5X, Renemy5Y, IenemyX1, IenemyY1, IenemyX2 , IenemyY2 ,score = 0, m = 0, c = 0;

char maze[42][852];

**Function Prototype:**

bool gameRun = true;

int lives = 3;

int en = 0;

int maxScore = 5;

int dis[4];

int dis2[4];

string direction1 = "left";

string direction2 = "right";

string direction3 = "left";

string direction4 = "left";

string direction5 = "right";

string direction6 = "right";

string direction7 = "left";

string direction8 = "left";

string direction9 = "right";

string direction10 = "up";

char previousitem1 = ' ';

char previousitem2 = ' ';

char previousitem3 = ' ';

char previousitem4 = ' ';

char previousitem5 = ' ';

int old1x, old1y;

int bulletX[100], bulletY[100], pX, pY, enemy1X, enemy1Y, enemy2X, enemy2Y, enemy3X, enemy3Y, enemy4X, enemy4Y, enemy5X, enemy5Y, enemy6X, enemy6Y, enemy7X, enemy7Y, enemy8X, enemy8Y, enemy9X, enemy9Y, enemy10X, enemy10Y, Renemy1X, Renemy1Y, Renemy2X, Renemy2Y, Renemy3X, Renemy3Y, Renemy4X, Renemy4Y, Renemy5X, Renemy5Y, IenemyX1, IenemyY1, IenemyX2 , IenemyY2 ,score = 0, m = 0, c = 0;

char maze[42][852];

**Code :**

#include <iostream>

#include <string>

#include <fstream>

#include <windows.h>

#include <conio.h>

using namespace std;

bool gameRun = true;

int lives = 3;

int en = 0;

int maxScore = 5;

int dis[4];

int dis2[4];

string direction1 = "left";

string direction2 = "right";

string direction3 = "left";

string direction4 = "left";

string direction5 = "right";

string direction6 = "right";

string direction7 = "left";

string direction8 = "left";

string direction9 = "right";

string direction10 = "up";

char previousitem1 = ' ';

char previousitem2 = ' ';

char previousitem3 = ' ';

char previousitem4 = ' ';

char previousitem5 = ' ';

int old1x, old1y;

int bulletX[100], bulletY[100], pX, pY, enemy1X, enemy1Y, enemy2X, enemy2Y, enemy3X, enemy3Y, enemy4X, enemy4Y, enemy5X, enemy5Y, enemy6X, enemy6Y, enemy7X, enemy7Y, enemy8X, enemy8Y, enemy9X, enemy9Y, enemy10X, enemy10Y, Renemy1X, Renemy1Y, Renemy2X, Renemy2Y, Renemy3X, Renemy3Y, Renemy4X, Renemy4Y, Renemy5X, Renemy5Y, IenemyX1, IenemyY1, IenemyX2 , IenemyY2 ,score = 0, m = 0, c = 0;

char maze[42][852] ;

int mainmenu();

void gotoxy(int x, int y);

void calculateMaxScore();

void displayMaze();

void findPacPos();

void fire();

void movebullet(int a);

void findGPos();

void moveUp();

void moveDown();

void moveLeft();

void moveRight();

void move\_enemy1\_Left();

void move\_enemy1\_Right();

void move\_enemy1();

void move\_enemy2\_Left();

void move\_enemy2\_Right();

void move\_enemy2();

void move\_enemy3\_Left();

void move\_enemy3\_Right();

void move\_enemy3();

void move\_enemy4\_Left();

void move\_enemy4\_Right();

void move\_enemy4();

void move\_enemy5\_Left();

void move\_enemy5\_Right();

void move\_enemy5();

void move\_enemy6\_Left();

void move\_enemy6\_Right();

void move\_enemy6();

void move\_enemy7\_Left();

void move\_enemy7\_Right();

void move\_enemy7();

void move\_enemy8\_Left();

void move\_enemy8\_Right();

void move\_enemy8();

void move\_enemy9\_Left();

void move\_enemy9\_Right();

void move\_enemy9();

void move\_enemy10\_Up();

void move\_enemy10\_Down();

void move\_enemy10();

int getrand1();

void move\_Renemy1();

int getrand2();

void move\_Renemy2();

int getrand3();

void move\_Renemy3();

int getrand4();

void move\_Renemy4();

int Distance1(int x1, int y1, int x2, int y2);

int findMinimum1();

void move\_Ienemy1\_Up();

void move\_Ienemy1\_Down();

void move\_Ienemy1\_Left();

void move\_Ienemy1\_Right();

void move\_Ienemy1();

int Distance2(int x1, int y1, int x2, int y2);

int findMinimum2();

void move\_Ienemy2\_Up();

void move\_Ienemy2\_Down();

void move\_Ienemy2\_Left();

void move\_Ienemy2\_Right();

void move\_Ienemy2();

void loadMaze();

void savemaze();

main()

{ int op = 0 ;

findPacPos();

findGPos();

calculateMaxScore();

int a;

op = mainmenu();

if (op == 2)

{

loadMaze();

displayMaze();

getch();

}

if (op == 3)

{

gameRun = false;

}

while (gameRun)

{

a++;

int oldLives = lives;

gotoxy(0, 0);

cout << "Score: " << score;

gotoxy(0, 1);

cout << "Lives: " << lives << " " << en;

gotoxy(0, 2);

displayMaze();

if (GetAsyncKeyState(VK\_UP))

{

for (int i = 0; i < 3; i++)

{

moveUp();

}

}

moveDown();

if (GetAsyncKeyState(VK\_DOWN))

{

moveDown();

}

if (GetAsyncKeyState(VK\_RIGHT))

{

moveRight();

if (pX + 1 != '|')

{

moveDown();

}

}

if (GetAsyncKeyState(VK\_LEFT))

{

moveLeft();

moveDown();

}

if (GetAsyncKeyState(VK\_SPACE))

{

fire();

}

movebullet(a);

if (maze[enemy1X][enemy1Y] == 'V')

{

move\_enemy1();

}

if (maze[enemy2X][enemy2Y] == 'V')

{

move\_enemy2();

}

if (maze[enemy3X][enemy3Y] == 'V')

{

move\_enemy3();

}

if (maze[enemy4X][enemy4Y] == 'V')

{

move\_enemy4();

}

if (maze[enemy5X][enemy5Y] == 'V')

{

move\_enemy5();

}

if (maze[enemy6X][enemy6Y] == 'V')

{

move\_enemy6();

}

if (maze[enemy7X][enemy7Y] == 'V')

{

move\_enemy7();

}

if (maze[enemy8X][enemy8Y] == 'V')

{

move\_enemy8();

}

if (maze[enemy9X][enemy9Y] == 'V')

{

move\_enemy9();

}

if (maze[enemy10X][enemy10Y] == 'V')

{

move\_enemy10();

}

move\_Renemy1();

move\_Renemy2();

move\_Renemy3();

move\_Renemy4();

if (maze[IenemyX1][IenemyY1] == 'B')

{

move\_Ienemy1();

}

if (maze[IenemyX2][IenemyY2] == 'B')

{

move\_Ienemy2();

}

if (oldLives != lives)

{

maze[pX][pY] = ' ';

pX = 20;

pY = 4;

maze[pX][pY] = 'P';

c = 0 ;

}

if (score == maxScore && maze[pX][pY + 1] == '+' || maze[pX + 1][pY] == '+')

{

system("cls");

gotoxy(0, 0);

cout << "score: " << score << endl;

cout << "YOU WONF" << endl;

break;

}

if (lives <= 0)

{

system("cls");

gotoxy(0, 0);

cout << "score: " << score << endl;

;

cout << "YOU LOST" << endl;

break;

}

if (GetAsyncKeyState(VK\_ESCAPE))

{

savemaze();

break;

}

}

system("pause");

}

int mainmenu()

{

int op;

cout << " -----------------------------------" << endl;

cout << "| s u p e r m a r i o |" << endl;

cout << " ----------------------------------- " << endl;

cout << "1\_Start new game " << endl;

cout << "2\_Resume game "<<endl;

cout << "3\_Quit game "<< endl;

cin >> op;

system("cls");

return op;

}

void calculateMaxScore()

{

for (int i = 0; i < 24; i++)

{

for (int j = 0; j < 71; j++)

{

if (maze[i][j] == 'O')

{

maxScore++;

}

}

}

}

void displayMaze()

{

c = c + m;

for (int i = 0; i < 42; i++)

{

for (int j = 0 + c; j < 71 + c; j++)

{

if (j + c < 0)

{

c = 0;

}

cout << maze[i][j];

}

cout << "\n";

}

m = 0;

}

void findPacPos()

{

int count = 0;

for (int i = 0; i < 24; i++)

{

for (int j = 0; j < 71; j++)

{

if (maze[i][j] == 'P')

{

pX = i;

pY = j;

break;

}

}

}

}

void fire()

{

if (maze[pX][pY + 1] == ' ' || maze[pX][pY + 1] == 'V' || maze[pX][pY + 1] == 'G' || maze[pX][pY + 1] == 'B')

maze[pX][pY + 1] = '=';

}

void movebullet(int a)

{

int stop = 0 ;

int count = 0;

for (int i = 0; i < 42; i++)

{

for (int j = 0; j < 852; j++)

{

if (maze[i][j] == '=')

{

bulletX[count] = i;

bulletY[count] = j;

count++;

}

}

}

for (int k = 0; k < count; k++)

{

if (pY+ 20 <= bulletY[k])

{

maze[bulletX[k]][bulletY[k]] == ' ';

}

else if (maze[bulletX[k]][bulletY[k] + 1] == ' ')

{

maze[bulletX[k]][bulletY[k] + 1] = '=';

}

else if (maze[bulletX[k]][bulletY[k] + 1] == 'V')

{

maze[bulletX[k]][bulletY[k] + 1] = '=';

}

else if (maze[bulletX[k]][bulletY[k] + 1] == 'B')

{

maze[bulletX[k]][bulletY[k] + 1] = '=';

}

else if (maze[bulletX[k]][bulletY[k] + 1] == 'O')

{

maze[bulletX[k]][bulletY[k] + 2] = '=';

}

maze[bulletX[k]][bulletY[k]] = ' ';

}

}

void findGPos()

{

int count = 0;

int countG = 0;

int countB = 0;

for (int i = 0; i < 42; i++)

{

for (int j = 0; j < 852; j++)

{

if (maze[i][j] == 'V')

{

count++;

if (count == 1)

{

enemy1X = i;

enemy1Y = j;

}

if (count == 2)

{

enemy2X = i;

enemy2Y = j;

}

if (count == 3)

{

enemy3X = i;

enemy3Y = j;

}

if (count == 4)

{

enemy4X = i;

enemy4Y = j;

}

if (count == 5)

{

enemy5X = i;

enemy5Y = j;

}

if (count == 6)

{

enemy6X = i;

enemy6Y = j;

}

if (count == 7)

{

enemy7X = i;

enemy7Y = j;

}

if (count == 8)

{

enemy8X = i;

enemy8Y = j;

}

if (count == 9)

{

enemy9X = i;

enemy9Y = j;

}

if (count == 10)

{

enemy10X = i;

enemy10Y = j;

}

}

}

}

for (int k = 0; k < 42; k++)

{

for (int l = 0; l < 852; l++)

{

if (maze[k][l] == 'G')

{

countG++;

if (countG == 1)

{

Renemy1X = k;

Renemy1Y = l;

}

if (countG == 2)

{

Renemy2X = k;

Renemy2Y = l;

}

if (countG == 3)

{

Renemy3X = k;

Renemy3Y = l;

}

if (countG == 4)

{

Renemy4X = k;

Renemy4Y = l;

}

if (countG == 5)

{

Renemy5X = k;

Renemy5Y = l;

}

}

}

}

for (int m = 0; m < 42; m++)

{

for (int n = 0; n < 852; n++)

{

if (maze[m][n] == 'B')

{

countB++;

if (countB == 1)

{

IenemyX1 = m;

IenemyY1 = n;

}

if (countB == 2)

{

IenemyX2 = m;

IenemyY2 = n;

}

}

}

}

}

void moveUp()

{

if (maze[pX - 1][pY] == ' ')

{

maze[pX][pY] = ' ';

maze[pX - 1][pY] = 'P';

pX--;

}

if (maze[pX - 1][pY] == 'O')

{

maze[pX][pY] = ' ';

maze[pX - 1][pY] = 'P';

pX--;

score++;

}

if (maze[pX - 1][pY] == '\*')

{

lives--;

}

}

void moveDown()

{

if (maze[pX + 1][pY] == ' ')

{

maze[pX][pY] = ' ';

maze[pX + 1][pY] = 'P';

pX++;

}

if (maze[pX + 1][pY] == 'O')

{

score++;

maze[pX][pY] = ' ';

maze[pX + 1][pY] = 'P';

pX++;

}

if (maze[pX + 1][pY] == '\*' || pX == 40)

{

lives--;

}

}

void moveLeft()

{

if (maze[pX][pY - 1] == ' ')

{

m = -1;

maze[pX][pY] = ' ';

maze[pX][pY - 1] = 'P';

pY--;

}

if (maze[pX][pY - 1] == 'O')

{

score++;

maze[pX][pY] = ' ';

maze[pX][pY - 1] = 'P';

pY--;

}

if (maze[pX][pY - 1] == '\*')

{

lives--;

}

}

void moveRight()

{

if (maze[pX][pY + 1] == ' ')

{

m = 1;

maze[pX][pY] = ' ';

maze[pX][pY + 1] = 'P';

pY++;

}

if (maze[pX][pY + 1] == 'O')

{

score++;

maze[pX][pY] = ' ';

maze[pX][pY + 1] = 'P';

pY++;

}

if (maze[pX][pY + 1] == '\*')

{

lives--;

}

}

// for enemy 1 -------------- right to left movement

void move\_enemy1\_Left()

{

if (maze[enemy1X][enemy1Y - 1] == ' ')

{

maze[enemy1X][enemy1Y] = ' ';

maze[enemy1X][enemy1Y - 1] = 'V';

enemy1Y--;

}

if (maze[enemy1X][enemy1Y - 1] == 'O')

{

maze[enemy1X][enemy1Y] = 'O';

maze[enemy1X][enemy1Y - 1] = 'V';

enemy1Y--;

}

if (maze[enemy1X][enemy1Y - 1] == '=')

{

maze[enemy1X][enemy1Y] = ' ';

}

if (maze[enemy1X][enemy1Y - 1] == 'P')

{

lives--;

}

}

void move\_enemy1\_Right()

{

if (maze[enemy1X][enemy1Y + 1] == ' ')

{

maze[enemy1X][enemy1Y] = ' ';

maze[enemy1X][enemy1Y + 1] = 'V';

enemy1Y++;

}

if (maze[enemy1X][enemy1Y + 1] == 'O')

{

maze[enemy1X][enemy1Y] = 'O';

maze[enemy1X][enemy1Y + 1] = 'V';

enemy1Y++;

}

if (maze[enemy1X][enemy1Y + 1] == '=')

{

maze[enemy1X][enemy1Y] = ' ';

}

if (maze[enemy1X][enemy1Y + 1] == 'P')

{

lives--;

}

}

void move\_enemy1()

{

if (maze[enemy1X][enemy1Y + 1] == '|')

{

direction1 = "left";

}

if (maze[enemy1X][enemy1Y - 1] == ']')

{

direction1 = "right";

}

if (direction1 == "left")

{

move\_enemy1\_Left();

}

if (direction1 == "right")

{

move\_enemy1\_Right();

}

}

// enemy 2 ------------------------------- left to right movement

void move\_enemy2\_Left()

{

if (maze[enemy2X][enemy2Y - 1] == ' ')

{

maze[enemy2X][enemy2Y] = ' ';

maze[enemy2X][enemy2Y - 1] = 'V';

enemy2Y--;

}

if (maze[enemy2X][enemy2Y - 1] == 'O')

{

maze[enemy2X][enemy2Y] = 'O';

maze[enemy2X][enemy2Y - 1] = 'V';

enemy2Y--;

}

if (maze[enemy2X][enemy2Y - 1] == '=')

{

maze[enemy2X][enemy2Y] = ' ';

}

if (maze[enemy2X][enemy2Y - 1] == 'P')

{

lives--;

}

}

void move\_enemy2\_Right()

{

if (maze[enemy2X][enemy2Y + 1] == ' ')

{

maze[enemy2X][enemy2Y] = ' ';

maze[enemy2X][enemy2Y + 1] = 'V';

enemy2Y++;

}

if (maze[enemy2X][enemy2Y + 1] == 'O')

{

maze[enemy2X][enemy2Y] = 'O';

maze[enemy2X][enemy2Y + 1] = 'V';

enemy2Y++;

}

if (maze[enemy2X][enemy2Y + 1] == '=')

{

maze[enemy2X][enemy2Y] = ' ';

}

if (maze[enemy2X][enemy2Y + 1] == 'P')

{

lives--;

}

}

void move\_enemy2()

{

if (maze[enemy2X][enemy2Y + 1] == '|')

{

direction2 = "left";

}

if (maze[enemy2X][enemy2Y - 1] == '|')

{

direction2 = "right";

}

if (direction2 == "left")

{

move\_enemy2\_Left();

}

if (direction2 == "right")

{

move\_enemy2\_Right();

}

}

// enemy 3 -------------------------- right to left movement

void move\_enemy3\_Left()

{

if (maze[enemy3X][enemy3Y - 1] == ' ')

{

maze[enemy3X][enemy3Y] = ' ';

maze[enemy3X][enemy3Y - 1] = 'V';

enemy3Y--;

}

if (maze[enemy3X][enemy3Y - 1] == 'O')

{

maze[enemy3X][enemy3Y] = 'O';

maze[enemy3X][enemy3Y - 1] = 'V';

enemy3Y--;

}

if (maze[enemy3X][enemy3Y - 1] == '=')

{

maze[enemy3X][enemy3Y] = ' ';

}

if (maze[enemy3X][enemy3Y - 1] == 'P')

{

lives--;

}

}

void move\_enemy3\_Right()

{

if (maze[enemy3X][enemy3Y + 1] == ' ')

{

maze[enemy3X][enemy3Y] = ' ';

maze[enemy3X][enemy3Y + 1] = 'V';

enemy3Y++;

}

if (maze[enemy3X][enemy3Y + 1] == 'O')

{

maze[enemy3X][enemy3Y] = 'O';

maze[enemy3X][enemy3Y + 1] = 'V';

enemy3Y++;

}

if (maze[enemy3X][enemy3Y + 1] == '=')

{

maze[enemy3X][enemy3Y] = ' ';

}

if (maze[enemy3X][enemy3Y + 1] == 'P')

{

lives--;

}

}

void move\_enemy3()

{

if (maze[enemy3X][enemy3Y + 1] == '|')

{

direction3 = "left";

}

if (maze[enemy3X][enemy3Y - 1] == '|')

{

direction3 = "right";

}

if (direction3 == "left")

{

move\_enemy3\_Left();

}

if (direction3 == "right")

{

move\_enemy3\_Right();

}

}

// enemy 4 ------------------------right to left movement

void move\_enemy4\_Left()

{

if (maze[enemy4X][enemy4Y - 1] == ' ')

{

maze[enemy4X][enemy4Y] = ' ';

maze[enemy4X][enemy4Y - 1] = 'V';

enemy4Y--;

}

if (maze[enemy4X][enemy4Y - 1] == 'O')

{

maze[enemy4X][enemy4Y] = 'O';

maze[enemy4X][enemy4Y - 1] = 'V';

enemy4Y--;

}

if (maze[enemy4X][enemy4Y - 1] == '=')

{

maze[enemy4X][enemy4Y] = ' ';

}

if (maze[enemy4X][enemy4Y - 1] == 'P')

{

lives--;

}

}

void move\_enemy4\_Right()

{

if (maze[enemy4X][enemy4Y + 1] == ' ')

{

maze[enemy4X][enemy4Y] = ' ';

maze[enemy4X][enemy4Y + 1] = 'V';

enemy4Y++;

}

if (maze[enemy4X][enemy4Y + 1] == 'O')

{

maze[enemy4X][enemy4Y] = 'O';

maze[enemy4X][enemy4Y + 1] = 'V';

enemy4Y++;

}

if (maze[enemy4X][enemy4Y + 1] == '=')

{

maze[enemy4X][enemy4Y] = ' ';

}

if (maze[enemy4X][enemy4Y + 1] == 'P')

{

lives--;

}

}

void move\_enemy4()

{

if (maze[enemy4X][enemy4Y + 1] == '[')

{

direction4 = "left";

}

if (maze[enemy4X][enemy4Y - 1] == '|')

{

direction4 = "right";

}

if (direction4 == "left")

{

move\_enemy4\_Left();

}

if (direction4 == "right")

{

move\_enemy4\_Right();

}

}

// enemy 5 ------------------------left to right

void move\_enemy5\_Left()

{

if (maze[enemy5X][enemy5Y - 1] == ' ')

{

maze[enemy5X][enemy5Y] = ' ';

maze[enemy5X][enemy5Y - 1] = 'V';

enemy5Y--;

}

if (maze[enemy5X][enemy5Y - 1] == 'O')

{

maze[enemy5X][enemy5Y] = 'O';

maze[enemy5X][enemy5Y - 1] = 'V';

enemy5Y--;

}

if (maze[enemy5X][enemy5Y - 1] == '=')

{

maze[enemy5X][enemy5Y] = ' ';

}

if (maze[enemy5X][enemy5Y - 1] == 'P')

{

lives--;

}

}

void move\_enemy5\_Right()

{

if (maze[enemy5X][enemy5Y + 1] == ' ')

{

maze[enemy5X][enemy5Y] = ' ';

maze[enemy5X][enemy5Y + 1] = 'V';

enemy5Y++;

}

if (maze[enemy5X][enemy5Y + 1] == 'O')

{

maze[enemy5X][enemy5Y] = 'O';

maze[enemy5X][enemy5Y + 1] = 'V';

enemy5Y++;

}

if (maze[enemy5X][enemy5Y + 1] == '=')

{

maze[enemy5X][enemy5Y] = ' ';

}

if (maze[enemy5X][enemy5Y + 1] == 'P')

{

lives--;

}

}

void move\_enemy5()

{

if (maze[enemy5X][enemy5Y + 1] == '|')

{

direction5 = "left";

}

if (maze[enemy5X][enemy5Y - 1] == '|')

{

direction5 = "right";

}

if (direction5 == "left")

{

move\_enemy5\_Left();

}

if (direction5 == "right")

{

move\_enemy5\_Right();

}

}

// enemy 6 ------------------------left to right movement

void move\_enemy6\_Left()

{

if (maze[enemy6X][enemy6Y - 1] == ' ')

{

maze[enemy6X][enemy6Y] = ' ';

maze[enemy6X][enemy6Y - 1] = 'V';

enemy6Y--;

}

if (maze[enemy6X][enemy6Y - 1] == 'O')

{

maze[enemy6X][enemy6Y] = 'O';

maze[enemy6X][enemy6Y - 1] = 'V';

enemy6Y--;

}

if (maze[enemy6X][enemy6Y - 1] == '=')

{

maze[enemy6X][enemy6Y] = ' ';

}

if (maze[enemy6X][enemy6Y - 1] == 'P')

{

lives--;

}

}

void move\_enemy6\_Right()

{

if (maze[enemy6X][enemy6Y + 1] == ' ')

{

maze[enemy6X][enemy6Y] = ' ';

maze[enemy6X][enemy6Y + 1] = 'V';

enemy6Y++;

}

if (maze[enemy6X][enemy6Y + 1] == 'O')

{

maze[enemy6X][enemy6Y] = 'O';

maze[enemy6X][enemy6Y + 1] = 'V';

enemy6Y++;

}

if (maze[enemy6X][enemy6Y + 1] == '=')

{

maze[enemy6X][enemy6Y] = ' ';

}

if (maze[enemy6X][enemy6Y + 1] == 'P')

{

lives--;

}

}

void move\_enemy6()

{

if (maze[enemy6X][enemy6Y + 1] == '|')

{

direction6 = "left";

}

if (maze[enemy6X][enemy6Y - 1] == '|')

{

direction6 = "right";

}

if (direction6 == "left")

{

move\_enemy6\_Left();

}

if (direction6 == "right")

{

move\_enemy6\_Right();

}

}

// enemy 7 ----------------------------right to left

void move\_enemy7\_Left()

{

if (maze[enemy7X][enemy7Y - 1] == ' ')

{

maze[enemy7X][enemy7Y] = ' ';

maze[enemy7X][enemy7Y - 1] = 'V';

enemy7Y--;

}

if (maze[enemy7X][enemy7Y - 1] == 'O')

{

maze[enemy7X][enemy7Y] = 'O';

maze[enemy7X][enemy7Y - 1] = 'V';

enemy7Y--;

}

if (maze[enemy7X][enemy7Y - 1] == '=')

{

maze[enemy7X][enemy7Y] = ' ';

}

if (maze[enemy7X][enemy7Y - 1] == 'P')

{

lives--;

}

}

void move\_enemy7\_Right()

{

if (maze[enemy7X][enemy7Y + 1] == ' ')

{

maze[enemy7X][enemy7Y] = ' ';

maze[enemy7X][enemy7Y + 1] = 'V';

enemy7Y++;

}

if (maze[enemy7X][enemy7Y + 1] == 'O')

{

maze[enemy7X][enemy7Y] = 'O';

maze[enemy7X][enemy7Y + 1] = 'V';

enemy7Y++;

}

if (maze[enemy7X][enemy7Y + 1] == '=')

{

maze[enemy7X][enemy7Y] = ' ';

}

if (maze[enemy7X][enemy7Y + 1] == 'P')

{

lives--;

}

}

void move\_enemy7()

{

if (maze[enemy7X][enemy7Y + 1] == '|')

{

direction7 = "left";

}

if (maze[enemy7X][enemy7Y - 1] == '|')

{

direction7 = "right";

}

if (direction7 == "left")

{

move\_enemy7\_Left();

}

if (direction7 == "right")

{

move\_enemy7\_Right();

}

}

// enemy 8 ----------------------------right to left

void move\_enemy8\_Left()

{

if (maze[enemy8X][enemy8Y - 1] == ' ')

{

maze[enemy8X][enemy8Y] = ' ';

maze[enemy8X][enemy8Y - 1] = 'V';

enemy8Y--;

}

if (maze[enemy8X][enemy8Y - 1] == 'O')

{

maze[enemy8X][enemy8Y] = 'O';

maze[enemy8X][enemy8Y - 1] = 'V';

enemy8Y--;

}

if (maze[enemy8X][enemy8Y - 1] == '=')

{

maze[enemy8X][enemy8Y] = ' ';

}

if (maze[enemy8X][enemy8Y - 1] == 'P')

{

lives--;

}

}

void move\_enemy8\_Right()

{

if (maze[enemy8X][enemy8Y + 1] == ' ')

{

maze[enemy8X][enemy8Y] = ' ';

maze[enemy8X][enemy8Y + 1] = 'V';

enemy8Y++;

}

if (maze[enemy8X][enemy8Y + 1] == 'O')

{

maze[enemy8X][enemy8Y] = 'O';

maze[enemy8X][enemy8Y + 1] = 'V';

enemy8Y++;

}

if (maze[enemy8X][enemy8Y + 1] == '=')

{

maze[enemy8X][enemy8Y] = ' ';

}

if (maze[enemy8X][enemy8Y + 1] == 'P')

{

lives--;

}

}

void move\_enemy8()

{

if (maze[enemy8X][enemy8Y + 1] == '|')

{

direction8 = "left";

}

if (maze[enemy8X][enemy8Y - 1] == '|')

{

direction8 = "right";

}

if (direction8 == "left")

{

move\_enemy8\_Left();

}

if (direction8 == "right")

{

move\_enemy8\_Right();

}

}

// enemy 9 ----------------------------right to left

void move\_enemy9\_Left()

{

if (maze[enemy9X][enemy9Y - 1] == ' ')

{

maze[enemy9X][enemy9Y] = ' ';

maze[enemy9X][enemy9Y - 1] = 'V';

enemy9Y--;

}

if (maze[enemy9X][enemy9Y - 1] == 'O')

{

maze[enemy9X][enemy9Y] = 'O';

maze[enemy9X][enemy9Y - 1] = 'V';

enemy9Y--;

}

if (maze[enemy9X][enemy9Y - 1] == '=')

{

maze[enemy9X][enemy9Y] = ' ';

}

if (maze[enemy9X][enemy9Y - 1] == 'P')

{

lives--;

}

}

void move\_enemy9\_Right()

{

if (maze[enemy9X][enemy9Y + 1] == ' ')

{

maze[enemy9X][enemy9Y] = ' ';

maze[enemy9X][enemy9Y + 1] = 'V';

enemy9Y++;

}

if (maze[enemy9X][enemy9Y + 1] == 'O')

{

maze[enemy9X][enemy9Y] = 'O';

maze[enemy9X][enemy9Y + 1] = 'V';

enemy9Y++;

}

if (maze[enemy9X][enemy9Y + 1] == '=')

{

maze[enemy9X][enemy9Y] = ' ';

}

if (maze[enemy9X][enemy9Y + 1] == 'P')

{

lives--;

}

}

void move\_enemy9()

{

if (maze[enemy9X][enemy9Y + 1] == '|')

{

direction9 = "left";

}

if (maze[enemy9X][enemy9Y - 1] == '|')

{

direction9 = "right";

}

if (direction9 == "left")

{

move\_enemy9\_Left();

}

if (direction9 == "right")

{

move\_enemy9\_Right();

}

}

// enemy 10 ---------------------------------up movement

void move\_enemy10\_Up()

{

if (maze[enemy10X - 1][enemy10Y] == ' ')

{

maze[enemy10X][enemy10Y] = ' ';

maze[enemy10X - 1][enemy10Y] = 'V';

enemy10X--;

}

if (maze[enemy10X - 1][enemy10Y] == 'O')

{

maze[enemy10X][enemy10Y] = 'O';

maze[enemy10X - 1][enemy10Y] = 'V';

enemy10X--;

}

if (maze[enemy10X - 1][enemy10Y] == '=')

{

maze[enemy10X][enemy10Y] = ' ';

}

if (maze[enemy10X - 1][enemy10Y] == 'P')

{

lives--;

}

}

void move\_enemy10\_Down()

{

if (maze[enemy10X + 1][enemy10Y] == ' ')

{

maze[enemy10X][enemy10Y] = ' ';

maze[enemy10X + 1][enemy10Y] = 'V';

enemy10X++;

}

if (maze[enemy10X + 1][enemy10Y] == 'O')

{

maze[enemy10X][enemy10Y] = 'O';

maze[enemy10X + 1][enemy10Y] = 'V';

enemy10X++;

}

if (maze[enemy10X + 1][enemy10Y] == '=')

{

maze[enemy10X][enemy10Y] = ' ';

}

if (maze[enemy10X - 1][enemy10Y] == 'P')

{

lives--;

}

}

void move\_enemy10()

{

if (maze[enemy10X - 1][enemy10Y] == '#')

{

direction10 = "down";

}

if (maze[enemy10X + 1][enemy10Y] == '|')

{

direction10 = "up";

}

if (direction10 == "up")

{

move\_enemy10\_Up();

}

if (direction10 == "down")

{

move\_enemy10\_Down();

}

}

// random enemy 1 ---------------------- movement

int getrand1()

{

srand(time(0));

int num = (1 + (rand() % 4));

return num;

}

void move\_Renemy1()

{

int move = getrand1();

if (move == 1)

{

if (maze[Renemy1X][Renemy1Y - 1] == ' ' || maze[Renemy1X][Renemy1Y - 1] == 'O' || maze[Renemy1X][Renemy1Y - 1] == 'P')

{

maze[Renemy1X][Renemy1Y] = previousitem1;

Renemy1Y = Renemy1Y - 1;

previousitem1 = maze[Renemy1X][Renemy1Y];

if (previousitem1 == 'P')

{

previousitem1 = ' ';

lives--;

}

maze[Renemy1X][Renemy1Y] = 'G';

}

}

if (move == 2)

{

if (maze[Renemy1X][Renemy1Y + 1] == ' ' || maze[Renemy1X][Renemy1Y + 1] == 'O' || maze[Renemy1X][Renemy1Y + 1] == 'P')

{

maze[Renemy1X][Renemy1Y] = previousitem1;

Renemy1Y = Renemy1Y + 1;

previousitem1 = maze[Renemy1X][Renemy1Y];

if (previousitem1 == 'P')

{

previousitem1 = ' ';

lives--;

}

maze[Renemy1X][Renemy1Y] = 'G';

}

}

if (move == 3)

{

if (maze[Renemy1X - 1][Renemy1Y] == ' ' || maze[Renemy1X - 1][Renemy1Y] == 'O' || maze[Renemy1X - 1][Renemy1Y] == 'P')

{

maze[Renemy1X][Renemy1Y] = previousitem1;

Renemy1X = Renemy1X - 1;

previousitem1 = maze[Renemy1X][Renemy1Y];

if (previousitem1 == 'P')

{

previousitem1 = ' ';

lives--;

}

maze[Renemy1X][Renemy1Y] = 'G';

}

}

if (move == 4)

{

if (maze[Renemy1X + 1][Renemy1Y] == ' ' || maze[Renemy1X + 1][Renemy1Y] == 'O' || maze[Renemy1X + 1][Renemy1Y] == 'P')

{

maze[Renemy1X][Renemy1Y] = previousitem1;

Renemy1X = Renemy1X + 1;

previousitem1 = maze[Renemy1X][Renemy1Y];

if (previousitem1 == 'P')

{

previousitem1 = ' ';

lives--;

}

maze[Renemy1X][Renemy1Y] = 'G';

}

}

}

// random enemy 2 ---------------------- movement

int getrand2()

{

srand(time(0));

int num = (1 + (rand() % 4));

return num;

}

void move\_Renemy2()

{

int move = getrand2();

if (move == 1)

{

if (maze[Renemy2X][Renemy2Y - 1] == ' ' || maze[Renemy2X][Renemy2Y - 1] == 'O' || maze[Renemy2X][Renemy2Y - 1] == 'P')

{

maze[Renemy2X][Renemy2Y] = previousitem2;

Renemy2Y = Renemy2Y - 1;

previousitem2 = maze[Renemy2X][Renemy2Y];

if (previousitem2 == 'P')

{

previousitem2 = ' ';

lives--;

}

maze[Renemy2X][Renemy2Y] = 'G';

}

}

if (move == 2)

{

if (maze[Renemy2X][Renemy2Y + 1] == ' ' || maze[Renemy2X][Renemy2Y + 1] == 'O' || maze[Renemy2X][Renemy2Y + 1] == 'P')

{

maze[Renemy2X][Renemy2Y] = previousitem2;

Renemy2Y = Renemy2Y + 1;

previousitem2 = maze[Renemy2X][Renemy2Y];

if (previousitem2 == 'P')

{

previousitem2 = ' ';

lives--;

}

maze[Renemy2X][Renemy2Y] = 'G';

}

}

if (move == 3)

{

if (maze[Renemy2X - 1][Renemy2Y] == ' ' || maze[Renemy2X - 1][Renemy2Y] == 'O' || maze[Renemy2X - 1][Renemy2Y] == 'P')

{

maze[Renemy2X][Renemy2Y] = previousitem2;

Renemy2X = Renemy2X - 1;

previousitem2 = maze[Renemy2X][Renemy2Y];

if (previousitem2 == 'P')

{

previousitem2 = ' ';

lives--;

}

maze[Renemy2X][Renemy2Y] = 'G';

}

}

if (move == 4)

{

if (maze[Renemy2X + 1][Renemy2Y] == ' ' || maze[Renemy2X + 1][Renemy2Y] == 'O' || maze[Renemy2X + 1][Renemy2Y] == 'P')

{

maze[Renemy2X][Renemy2Y] = previousitem2;

Renemy2X = Renemy2X + 1;

previousitem2 = maze[Renemy2X][Renemy2Y];

if (previousitem2 == 'P')

{

previousitem2 = ' ';

lives--;

}

maze[Renemy2X][Renemy2Y] = 'G';

}

}

}

// random enemy 3 ------------------------------- movement

int getrand3()

{

srand(time(0));

int num = (1 + (rand() % 4));

return num;

}

void move\_Renemy3()

{

int move = getrand3();

if (move == 1)

{

if (maze[Renemy3X][Renemy3Y - 1] == ' ' || maze[Renemy3X][Renemy3Y - 1] == 'O' || maze[Renemy3X][Renemy3Y - 1] == 'P')

{

maze[Renemy3X][Renemy3Y] = previousitem3;

Renemy3Y = Renemy3Y - 1;

previousitem3 = maze[Renemy3X][Renemy3Y];

if (previousitem3 == 'P')

{

previousitem3 = ' ';

lives--;

}

maze[Renemy3X][Renemy3Y] = 'G';

}

}

if (move == 2)

{

if (maze[Renemy3X][Renemy3Y + 1] == ' ' || maze[Renemy3X][Renemy3Y + 1] == 'O' || maze[Renemy3X][Renemy3Y + 1] == 'P')

{

maze[Renemy3X][Renemy3Y] = previousitem3;

Renemy3Y = Renemy3Y + 1;

previousitem3 = maze[Renemy3X][Renemy3Y];

if (previousitem3 == 'P')

{

previousitem3 = ' ';

lives--;

}

maze[Renemy3X][Renemy3Y] = 'G';

}

}

if (move == 3)

{

if (maze[Renemy3X - 1][Renemy3Y] == ' ' || maze[Renemy3X - 1][Renemy3Y] == 'O' || maze[Renemy3X - 1][Renemy3Y] == 'P')

{

maze[Renemy3X][Renemy3Y] = previousitem3;

Renemy3X = Renemy3X - 1;

previousitem3 = maze[Renemy3X][Renemy3Y];

if (previousitem3 == 'P')

{

previousitem3 = ' ';

lives--;

}

maze[Renemy3X][Renemy3Y] = 'G';

}

}

if (move == 4)

{

if (maze[Renemy3X + 1][Renemy3Y] == ' ' || maze[Renemy3X + 1][Renemy3Y] == 'O' || maze[Renemy3X + 1][Renemy3Y] == 'P')

{

maze[Renemy3X][Renemy3Y] = previousitem3;

Renemy3X = Renemy3X + 1;

previousitem3 = maze[Renemy3X][Renemy3Y];

if (previousitem3 == 'P')

{

previousitem3 = ' ';

lives--;

}

maze[Renemy3X][Renemy3Y] = 'G';

}

}

}

// random enemy 4 -------------------- movement

int getrand4()

{

srand(time(0));

int num = (1 + (rand() % 4));

return num;

}

void move\_Renemy4()

{

int move = getrand4();

if (move == 1)

{

if (maze[Renemy4X][Renemy4Y - 1] == ' ' || maze[Renemy4X][Renemy4Y - 1] == 'O' || maze[Renemy4X][Renemy4Y - 1] == 'P')

{

maze[Renemy4X][Renemy4Y] = previousitem4;

Renemy4Y = Renemy4Y - 1;

previousitem4 = maze[Renemy4X][Renemy4Y];

if (previousitem4 == 'P')

{

previousitem4 = ' ';

lives--;

}

maze[Renemy4X][Renemy4Y] = 'G';

}

}

if (move == 2)

{

if (maze[Renemy4X][Renemy4Y + 1] == ' ' || maze[Renemy4X][Renemy4Y + 1] == 'O' || maze[Renemy4X][Renemy4Y + 1] == 'P')

{

maze[Renemy4X][Renemy4Y] = previousitem4;

Renemy4Y = Renemy4Y + 1;

previousitem4 = maze[Renemy4X][Renemy4Y];

if (previousitem4 == 'P')

{

previousitem4 = ' ';

lives--;

}

maze[Renemy4X][Renemy4Y] = 'G';

}

}

if (move == 3)

{

if (maze[Renemy4X - 1][Renemy4Y] == ' ' || maze[Renemy4X - 1][Renemy4Y] == 'O' || maze[Renemy4X - 1][Renemy4Y] == 'P')

{

maze[Renemy4X][Renemy4Y] = previousitem4;

Renemy4X = Renemy4X - 1;

previousitem4 = maze[Renemy4X][Renemy4Y];

if (previousitem4 == 'P')

{

previousitem4 = ' ';

lives--;

}

maze[Renemy4X][Renemy4Y] = 'G';

}

}

if (move == 4)

{

if (maze[Renemy4X + 1][Renemy4Y] == ' ' || maze[Renemy4X + 1][Renemy4Y] == 'O' || maze[Renemy4X + 1][Renemy4Y] == 'P')

{

maze[Renemy4X][Renemy4Y] = previousitem4;

Renemy4X = Renemy4X + 1;

previousitem4 = maze[Renemy4X][Renemy4Y];

if (previousitem4 == 'P')

{

previousitem4 = ' ';

lives--;

}

maze[Renemy4X][Renemy4Y] = 'G';

}

}

}

// inteligent enemy 1 ------------------------ movements

int Distance1(int x1, int y1, int x2, int y2)

{

if (maze[IenemyX1][IenemyY1] == '#' || maze[IenemyX1][IenemyY1] == '|' || maze[IenemyX1][IenemyY1] == '[' || maze[IenemyX1][IenemyY1] == ']' || maze[IenemyX1][IenemyY1] == '\*')

{

return 1000;

}

else

return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2) \* 1.0);

}

int findMinimum1()

{

int minima = 99999;

int minIdx = -1;

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

{

if (minima > dis[i])

{

minima = dis[i];

minIdx = i;

}

}

return minIdx;

}

void move\_Ienemy1\_Up()

{

if (maze[IenemyX1 - 1][IenemyY1] == ' ')

{

maze[IenemyX1][IenemyY1] = ' ';

maze[IenemyX1 - 1][IenemyY1] = 'B';

IenemyX1--;

}

if (maze[IenemyX1 - 1][IenemyY1] == 'O')

{

maze[IenemyX1][IenemyY1] = 'O';

maze[IenemyX1 - 1][IenemyY1] = 'B';

IenemyX1--;

}

if (maze[IenemyX1 - 1][IenemyY1] == '=')

{

maze[IenemyX1][IenemyY1] = ' ';

}

if (maze[IenemyX1 - 1][IenemyY1] == 'P')

{

lives--;

}

}

void move\_Ienemy1\_Down()

{

if (maze[IenemyX1 + 1][IenemyY1] == ' ')

{

maze[IenemyX1][IenemyY1] = ' ';

maze[IenemyX1 + 1][IenemyY1] = 'B';

IenemyX1++;

}

if (maze[IenemyX1 + 1][IenemyY1] == 'O')

{

maze[IenemyX1][IenemyY1] = 'O';

maze[IenemyX1 + 1][IenemyY1] = 'B';

IenemyX1++;

}

if (maze[IenemyX1 + 1][IenemyY1] == '=')

{

maze[IenemyX1][IenemyY1] = ' ';

}

if (maze[IenemyX1 + 1][IenemyY1] == 'P')

{

lives--;

}

}

void move\_Ienemy1\_Left()

{

if (maze[IenemyX1][IenemyY1 - 1] == ' ')

{

maze[IenemyX1][IenemyY1] = ' ';

maze[IenemyX1][IenemyY1 - 1] = 'B';

IenemyY1--;

}

if (maze[IenemyX1][IenemyY1 - 1] == 'O')

{

maze[IenemyX1][IenemyY1] = 'O';

maze[IenemyX1][IenemyY1 - 1] = 'B';

IenemyY1--;

}

if (maze[IenemyX1][IenemyY1 - 1] == '=')

{

maze[IenemyX1][IenemyY1] = ' ';

}

if (maze[IenemyX1][IenemyY1 - 1] == 'P')

{

lives--;

}

}

void move\_Ienemy1\_Right()

{

if (maze[IenemyX1][IenemyY1 + 1] == ' ')

{

maze[IenemyX1][IenemyY1] = ' ';

maze[IenemyX1][IenemyY1 + 1] = 'B';

IenemyY1++;

}

if (maze[IenemyX1][IenemyY1 + 1] == 'O')

{

maze[IenemyX1][IenemyY1] = 'O';

maze[IenemyX1][IenemyY1 + 1] = 'B';

IenemyY1++;

}

if (maze[IenemyX1][IenemyY1 + 1] == '=')

{

maze[IenemyX1][IenemyY1] = ' ';

}

if (maze[IenemyX1][IenemyY1 + 1] == 'P')

{

lives--;

}

}

void move\_Ienemy1()

{

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

{

dis[i] = 0;

}

dis[0] = Distance1(IenemyX1 - 1, IenemyY1, pX, pY);

dis[1] = Distance1(IenemyX1 + 1, IenemyY1, pX, pY);

dis[2] = Distance1(IenemyX1, IenemyY1 - 1, pX, pY);

dis[3] = Distance1(IenemyX1, IenemyY1 + 1, pX, pY);

int x = findMinimum1();

if (x == 0)

{

for (int i = 0; i < 3; i++)

{

move\_Ienemy1\_Up();

}

move\_Ienemy1\_Down();

}

if (x == 1)

{

move\_Ienemy1\_Down();

}

if (x == 2)

{

move\_Ienemy1\_Left();

}

if (x == 3)

{

move\_Ienemy1\_Right();

}

}

// inteligent enemy 2 .................. movement

int Distance2(int x1, int y1, int x2, int y2)

{

if (maze[IenemyX2][IenemyY2] == '#' || maze[IenemyX2][IenemyY2] == '|' || maze[IenemyX2][IenemyY2] == '[' || maze[IenemyX2][IenemyY2] == ']' || maze[IenemyX2][IenemyY2] == '\*')

{

return 1000;

}

else

return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2) \* 1.0);

}

int findMinimum2()

{

int minima = 99999;

int minIdx = -1;

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

{

if (minima > dis[i])

{

minima = dis[i];

minIdx = i;

}

}

return minIdx;

}

void move\_Ienemy2\_Up()

{

if (maze[IenemyX2 - 1][IenemyY2] == ' ')

{

maze[IenemyX2][IenemyY2] = ' ';

maze[IenemyX2 - 1][IenemyY2] = 'B';

IenemyX2--;

}

if (maze[IenemyX2 - 1][IenemyY2] == 'O')

{

maze[IenemyX2][IenemyY2] = 'O';

maze[IenemyX2 - 1][IenemyY2] = 'B';

IenemyX2--;

}

if (maze[IenemyX2 - 1][IenemyY2] == '=')

{

maze[IenemyX2][IenemyY2] = ' ';

}

if (maze[IenemyX2 - 1][IenemyY2] == 'P')

{

lives--;

}

}

void move\_Ienemy2\_Down()

{

if (maze[IenemyX2 + 1][IenemyY2] == ' ')

{

maze[IenemyX2][IenemyY2] = ' ';

maze[IenemyX2 + 1][IenemyY2] = 'B';

IenemyX2++;

}

if (maze[IenemyX2 + 1][IenemyY2] == 'O')

{

maze[IenemyX2][IenemyY2] = 'O';

maze[IenemyX2 + 1][IenemyY2] = 'B';

IenemyX2++;

}

if (maze[IenemyX2 + 1][IenemyY2] == '=')

{

maze[IenemyX2][IenemyY2] = ' ';

}

if (maze[IenemyX2 + 1][IenemyY2] == 'P')

{

lives--;

}

}

void move\_Ienemy2\_Left()

{

if (maze[IenemyX2][IenemyY2 - 1] == ' ')

{

maze[IenemyX2][IenemyY2] = ' ';

maze[IenemyX2][IenemyY2 - 1] = 'B';

IenemyY2--;

}

if (maze[IenemyX2][IenemyY2 - 1] == 'O')

{

maze[IenemyX2][IenemyY2] = 'O';

maze[IenemyX2][IenemyY2 - 1] = 'B';

IenemyY2--;

}

if (maze[IenemyX2][IenemyY2 - 1] == '=')

{

maze[IenemyX2][IenemyY2] = ' ';

}

if (maze[IenemyX2][IenemyY2 - 1] == 'P')

{

lives--;

}

}

void move\_Ienemy2\_Right()

{

if (maze[IenemyX2][IenemyY2 + 1] == ' ')

{

maze[IenemyX2][IenemyY2] = ' ';

maze[IenemyX2][IenemyY2 + 1] = 'B';

IenemyY2++;

}

if (maze[IenemyX2][IenemyY2 + 1] == 'O')

{

maze[IenemyX2][IenemyY2] = 'O';

maze[IenemyX2][IenemyY2 + 1] = 'B';

IenemyY2++;

}

if (maze[IenemyX2][IenemyY2 + 1] == '=')

{

maze[IenemyX2][IenemyY2] = ' ';

}

if (maze[IenemyX2][IenemyY2 + 1] == 'P')

{

lives--;

}

}

void move\_Ienemy2()

{

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

{

dis2[i] = 0;

}

dis2[0] = Distance2(IenemyX2 - 1, IenemyY2, pX, pY);

dis2[1] = Distance2(IenemyX2 + 1, IenemyY2, pX, pY);

dis2[2] = Distance2(IenemyX2, IenemyY2 - 1, pX, pY);

dis2[3] = Distance2(IenemyX2, IenemyY2 + 1, pX, pY);

int x = findMinimum2();

if (x == 0)

{

for (int i = 0; i < 3; i++)

{

move\_Ienemy2\_Up();

}

move\_Ienemy2\_Down();

}

if (x == 1)

{

move\_Ienemy2\_Down();

}

if (x == 2)

{

move\_Ienemy2\_Left();

}

if (x == 3)

{

move\_Ienemy2\_Right();

}

}

void loadMaze()

{

fstream file;

string line;

file.open("mario.txt" , ios::in);

getline(file,line);

score = stoi(line);

getline(file,line);

lives = stoi(line);

while(file.eof())

{

getline(file,line);

for (int row = 0; row < 42; row = row + 1)

{

for (int col = 0; col < line.length(); col = col + 1)

{

maze[row][col] = line[col];

}

}

}

file.close();

}

void savemaze()

{

fstream file;

file.open("mario.txt" , ios::out);

file << score << endl;

file << lives << endl;

for (int row = 0; row < 42 ; row++ )

{

for (int col = 0; col < 852 ; col++)

{

file << maze[row][col];

}

file << endl;

}

file.close();

}

void gotoxy(int x, int y)

{

COORD c;

c.X = x;

c.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), c);

}