**Attack of the Undead**



Session: 2021 – 2024

**Submitted by:**

Abdullah Nasir 2021-CS-113

**Supervised by:**

Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Table of Contents:**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Topics** | **Page No.** |
| 1 | Game Description | 3 |
| 2 | Game Storyline | 3 |
| 3 | Characters Description | 3 |
| 4 | Rules and Interactions | 4 |
| 5 | Goal of the Game | 4 |
| 6 | Function Prototypes | 4 |
| 7 | Data Structures | 5 |
| 8 | Game Screenshot | 6 |
| 9 | Code | 8 |

**Game Description:**

This game is programmed in C++ and can be played on command prompt. It is a game developed for entertainment purpose and contains violence. Player controls the movement of the hero by using keyboard. Player is also able to shoot the zombies. There are zombies running after the main character. Hero has limited health and it decreases every time he is hit by a zombie. Some of the zombies drop special items when they are killed. These special items are to be picked by the hero. If hero successfully picks up the item, he gets an upgrade. Last battle is with Evil Scientist who also tries to shoot the Hero in order to protect him.

**Game Storyline:**

There has been a deadly virus outbreak and the air has become contaminated. 85% of Human race have transformed into zombies. The hero of the game was lucky that he did not get infected. Now he is struggling to survive alone in this harsh environment. He plans to make things right and sets out to find the person behind all this ruination. He has to go through different stages to reach the final boss of the game. He defeats all the zombies, finds the person responsible for this devastation and uses the antidote to cure all the infected people.

**Characters Description:**

There are three characters around which the game revolves.

# Hero

The whole story revolves around this character as he is controlled by the person playing the game. Hero can run away from zombies or towards them. He has a gun which can be used to kill the zombies attacking him. He has a health bar which decreases when he is hit by an enemy. If his health is finished, the game ends and the world is destroyed.

# Zombies

These are creatures who were once normal human beings but after being infected by the virus, the have lost all their humanity and have turned into flesh eating monsters. They are able to deal damage to hero when the hit him. They can be killed by getting shot. Some of them drop items that can be used by hero for his advantage in the game.

# III. Evil scientist

The one responsible for the current state of the world is this evil scientist. He was not paid enough for his services by the US military so he planned to teach the military a lesson. He wanted to spread the virus in America but it was carried to other countries by the infected. He is the final boss of the game.

**Rules and Interactions:**

Following rules are followed in the game.

* Characters can move freely on screen but they cannot walk through walls.
* Hero’s health decreases when zombie hit him.
* Zombies die when they are hit by bullet.
* Defeating all zombies helps to climb the stage.
* If hero dies, he needs to restart the game.
* Zombies drop life upgrades which hero can use by touching them.

**Goal of the Game:**

The goal of this game is to defeat the Evil Scientist and use antidote to cure the infected humans. Hero defeats the zombies and climbs stages to reach the Evil Scientist and then fights the scientist to restore peace.

**Function Prototypes:**

void load();

void display();

void name();

void gotoxy(int x, int y);

void heroLeft();

void heroRight();

void heroUp();

void heroDown();

void printhealth();

void bossHealth();

void shootUp();

void shootDown();

void shootLeft();

void shootRight();

bool bossFight();

int bossDirection();

bool allZombies();

int zombie1Direction();

bool zombie1Movement();

int zombie2Direction();

bool zombie2Movement();

int zombie3Direction();

bool zombie3Movement();

int zombie4Direction();

bool zombie4Movement();

int zombie5Direction();

bool zombie5Movement();

void setColor (unsigned char color);

**Data Structures:**

char maze[30][100];

int heroX;

int heroY;

int zombie1X;

int zombie1Y;

int zombie2X;

int zombie2Y;

int zombie3X;

int zombie3Y;

int zombie4X;

int zombie4Y;

int zombie5X;

int zombie5Y;

int bossHeadX;

int bossHeadY;

int bossLeftX ;

int bossLeftY;

int bossRightX;

int bossRightY;

int bossTailX;

int bossTailY;

int bossMidX;

int bossMidY;

int horizontalBullet;

int verticalBullet;

int bulletUp;

int bulletDown;

int bulletLeft;

int bulletRight;

int z1;

int z2;

int z3;

int z4;

int z0;

int health;

int bHealth;

char previousItem;

**Screenshot of the Game:**

1. **Start of the Game**



1. **Stage 1**



1. **Stage 2**



**Code:**

#include <iostream>

#include <conio.h>

#include <fstream>

#include <windows.h>

#include <time.h>

#include <math.h>

using namespace std;

//\_\_\_\_\_\_Prototypes\_\_\_\_\_\_

void load();

void display();

void name();

void gotoxy(int x, int y);

void heroLeft();

void heroRight();

void heroUp();

void heroDown();

void printhealth();

void bossHealth();

void shootUp();

void shootDown();

void shootLeft();

void shootRight();

bool bossFight();

int bossDirection();

bool allZombies();

int zombie1Direction();

bool zombie1Movement();

int zombie2Direction();

bool zombie2Movement();

int zombie3Direction();

bool zombie3Movement();

int zombie4Direction();

bool zombie4Movement();

int zombie5Direction();

bool zombie5Movement();

//\_\_\_\_\_\_\_Maze\_\_\_\_\_\_\_

char maze[30][100];

//\_\_\_\_\_\_\_Hero\_\_\_\_\_\_

int heroX = 7;

int heroY = 5;

//\_\_\_\_\_\_Zombies\_\_\_\_\_

int zombie1X = 5;

int zombie1Y = 91;

int zombie2X = 26;

int zombie2Y = 13;

int zombie3X = 21;

int zombie3Y = 53;

int zombie4X = 25;

int zombie4Y = 91;

int zombie5X = 8;

int zombie5Y = 66;

//\_\_\_\_\_\_Boss\_\_\_\_\_\_\_

int bossHeadX = 14;

int bossHeadY = 59;

int bossLeftX = 15;

int bossLeftY = 58;

int bossRightX = 15;

int bossRightY = 60;

int bossTailX = 16;

int bossTailY = 59;

int bossMidX = 15;

int bossMidY = 59;

int horizontalBullet;

int verticalBullet;

//\_\_\_\_\_\_\_bullet\_\_\_\_

int bulletUp;

int bulletDown;

int bulletLeft;

int bulletRight;

//\_\_\_\_\_\_Others\_\_\_\_\_\_

int z1 = 1;

int z2 = 1;

int z3 = 1;

int z4 = 1;

int z0 = 1;

int health = 5;

int bHealth = 5;

char previousItem = ' ';

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_main

main()

{

bool gamerunning = true;

load();

system("cls");

name();

system("cls");

display();

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Stage 1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout << maze[heroX][heroY];

maze[zombie1X][zombie1Y] = '1';

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

maze[zombie2X][zombie2Y] = '2';

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

maze[zombie3X][zombie3Y] = '3';

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

maze[zombie4X][zombie4Y] = '4';

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

maze[zombie5X][zombie5Y] = '0';

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

while ( gamerunning )

{

Sleep(100);

gamerunning = allZombies();

if ( GetAsyncKeyState(VK\_LEFT) )

{

heroLeft();

}

if ( GetAsyncKeyState(VK\_RIGHT) )

{

heroRight();

}

if ( GetAsyncKeyState(VK\_UP) )

{

heroUp();

}

if ( GetAsyncKeyState(VK\_DOWN) )

{

heroDown();

}

if ( GetAsyncKeyState(VK\_ESCAPE) )

{

gamerunning = false;

}

if ( GetAsyncKeyState(0x57) )

{

shootUp();

}

if ( GetAsyncKeyState(0x53) )

{

shootDown();

}

if ( GetAsyncKeyState(0x41) )

{

shootLeft();

}

if ( GetAsyncKeyState(0x44) )

{

shootRight();

}

printhealth();

if ( health == 0 )

{

gamerunning = false;

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Stage 2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

gamerunning = true;

maze[bossHeadX][bossHeadY] = '^';

maze[bossMidX][bossMidY] = 'o';

maze[bossLeftX][bossLeftY] = '<';

maze[bossRightX][bossRightY] = '>';

maze[bossTailX][bossTailY] = 'v';

while ( gamerunning )

{

Sleep(100);

gamerunning = bossFight();

if ( GetAsyncKeyState(VK\_LEFT) )

{

heroLeft();

}

if ( GetAsyncKeyState(VK\_RIGHT) )

{

heroRight();

}

if ( GetAsyncKeyState(VK\_UP) )

{

heroUp();

}

if ( GetAsyncKeyState(VK\_DOWN) )

{

heroDown();

}

if ( GetAsyncKeyState(VK\_ESCAPE) )

{

gamerunning = false;

}

if ( GetAsyncKeyState(0x57) )

{

shootUp();

}

if ( GetAsyncKeyState(0x53) )

{

shootDown();

}

if ( GetAsyncKeyState(0x41) )

{

shootLeft();

}

if ( GetAsyncKeyState(0x44) )

{

shootRight();

}

printhealth();

if ( health == 0 )

{

gamerunning = false;

}

if ( bHealth == 0 )

{

gamerunning = false;

}

}

if (health == 0)

{

gotoxy(10,120);

cout <<"You lost";

}

else if (bHealth == 0)

{

gotoxy(10,120);

cout <<"You win";

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_file load

void load()

{

fstream file;

string line;

int r = 0;

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

while( getline( file, line ) )

{

for (int c = 0; c < 100; c++)

{

maze[r][c] = line[c];

}

r ++;

}

file.close();

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_display maze

void display()

{

for (int r = 0; r < 30; r++)

{

for (int c = 0; c < 100; c++)

{

cout <<maze[r][c];

}

cout <<endl;

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_hero health

void printhealth()

{

gotoxy(6,120);

cout <<"health = " <<health;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_gotoxy

void gotoxy(int x, int y)

{

COORD coordinate;

coordinate.X = y;

coordinate.Y = x;

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

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_hero movement

void heroLeft()

{

if ( maze[heroX][heroY - 1] == ' ' || maze[heroX][heroY - 1] == '\*')

{

maze[heroX][heroY] = ' ';

gotoxy(heroX, heroY);

cout <<maze[heroX][heroY];

heroY = heroY - 1;

if ( maze[heroX][heroY] == '\*' )

{

health = health + 1;

}

maze[heroX][heroY] = 'H';

gotoxy(heroX, heroY);

cout <<maze[heroX][heroY];

}

}

void heroRight()

{

if ( maze[heroX][heroY + 1] == ' ' || maze[heroX][heroY + 1] == '\*' )

{

maze[heroX][heroY] = ' ';

gotoxy(heroX, heroY);

cout <<maze[heroX][heroY];

heroY = heroY + 1;

if ( maze[heroX][heroY] == '\*' )

{

health = health + 1;

}

maze[heroX][heroY] = 'H';

gotoxy(heroX, heroY);

cout <<maze[heroX][heroY];

}

}

void heroUp()

{

if ( maze[heroX - 1][heroY] == ' ' || maze[heroX - 1][heroY] == '\*' )

{

maze[heroX][heroY] = ' ';

gotoxy(heroX, heroY);

cout <<maze[heroX][heroY];

heroX = heroX - 1;

if ( maze[heroX][heroY] == '\*' )

{

health = health + 1;

}

maze[heroX][heroY] = 'H';

gotoxy(heroX, heroY);

cout <<maze[heroX][heroY];

}

}

void heroDown()

{

if ( maze[heroX + 1][heroY] == ' ' || maze[heroX + 1][heroY] == '\*' )

{

maze[heroX][heroY] = ' ';

gotoxy(heroX, heroY);

cout <<maze[heroX][heroY];

heroX = heroX + 1;

if ( maze[heroX][heroY] == '\*' )

{

health = health + 1;

}

maze[heroX][heroY] = 'H';

gotoxy(heroX, heroY);

cout <<maze[heroX][heroY];

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ghosts directions

int zombie1Direction()

{

if(zombie1X > heroX)

{

if ( maze[zombie1X - 1][zombie1Y] != '#' )

{

return 3;

}

}

if(zombie1Y > heroY)

{

if ( maze[zombie1X][zombie1Y - 1] != '#' )

{

return 1;

}

}

if(zombie1X < heroX)

{

if ( maze[zombie1X + 1][zombie1Y] != '#' )

{

return 4;

}

}

if(zombie1Y < heroY)

{

if ( maze[zombie1X][zombie1Y + 1] != '#' )

{

return 2;

}

}

}

int zombie2Direction()

{

if(zombie2X > heroX)

{

if ( maze[zombie2X - 1][zombie2Y] != '#' )

{

return 3;

}

}

if(zombie2Y > heroY)

{

if ( maze[zombie2X][zombie2Y - 1] != '#' )

{

return 1;

}

}

if(zombie2X < heroX)

{

if ( maze[zombie2X + 1][zombie2Y] != '#' )

{

return 4;

}

}

if(zombie2Y < heroY)

{

if ( maze[zombie2X][zombie2Y + 1] != '#' )

{

return 2;

}

}

}

int zombie3Direction()

{

if(zombie3X > heroX)

{

if ( maze[zombie3X - 1][zombie3Y] != '#' )

{

return 3;

}

}

if(zombie3Y > heroY)

{

if ( maze[zombie3X][zombie3Y - 1] != '#' )

{

return 1;

}

}

if(zombie3X < heroX)

{

if ( maze[zombie3X + 1][zombie3Y] != '#' )

{

return 4;

}

}

if(zombie3Y < heroY)

{

if ( maze[zombie3X][zombie3Y + 1] != '#' )

{

return 2;

}

}

}

int zombie4Direction()

{

if(zombie4X > heroX)

{

if ( maze[zombie4X - 1][zombie4Y] != '#' )

{

return 3;

}

}

if(zombie4Y > heroY)

{

if ( maze[zombie4X][zombie4Y - 1] != '#' )

{

return 1;

}

}

if(zombie4X < heroX)

{

if ( maze[zombie4X + 1][zombie4Y] != '#' )

{

return 4;

}

}

if(zombie4Y < heroY)

{

if ( maze[zombie4X][zombie4Y + 1] != '#' )

{

return 2;

}

}

}

int zombie5Direction()

{

srand(time(0));

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

return result;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Zombies movements

bool allZombies()

{

if ( z1 == 0 && z2 == 0 && z3 == 0 && z4 == 0 && z0 == 0)

{

return 0;

}

if (zombie1Movement() && zombie2Movement() && zombie3Movement() && zombie4Movement() && zombie5Movement() )

{

return 1;

}

else

{

return 0;

}

}

bool zombie1Movement()

{

int value;

if (z1 == 0)

{

goto zombie1;

}

value = zombie1Direction();

if ( value == 1 ) //move left

{

if ( maze[zombie1X][zombie1Y - 1] == ' ' || maze[zombie1X][zombie1Y - 1] == 'H' || maze[zombie1X][zombie1Y - 1] == '\*' )

{

maze[zombie1X][zombie1Y] = previousItem;

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

zombie1Y = zombie1Y - 1;

previousItem = maze[zombie1X][zombie1Y];

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

maze[zombie1X][zombie1Y] = '1';

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 2) //move right

{

if ( maze[zombie1X][zombie1Y + 1] ==' ' || maze[zombie1X][zombie1Y + 1] == 'H' || maze[zombie1X][zombie1Y + 1] == '\*' )

{

maze[zombie1X][zombie1Y] = previousItem;

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

zombie1Y = zombie1Y + 1;

previousItem = maze[zombie1X][zombie1Y];

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

maze[zombie1X][zombie1Y] = '1';

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 3) //move up

{

if ( maze[zombie1X - 1][zombie1Y] ==' ' || maze[zombie1X - 1][zombie1Y] == 'H' || maze[zombie1X - 1][zombie1Y] == '\*' )

{

maze[zombie1X][zombie1Y] = previousItem;

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

zombie1X = zombie1X - 1;

previousItem = maze[zombie1X][zombie1Y];

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

maze[zombie1X][zombie1Y] = '1';

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 4) //move down

{

if ( maze[zombie1X + 1][zombie1Y] ==' ' || maze[zombie1X + 1][zombie1Y] == 'H' || maze[zombie1X + 1][zombie1Y] == '\*' )

{

maze[zombie1X][zombie1Y] = previousItem;

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

zombie1X = zombie1X + 1;

previousItem = maze[zombie1X][zombie1Y];

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

maze[zombie1X][zombie1Y] = '1';

gotoxy( zombie1X, zombie1Y);

cout <<maze[zombie1X][zombie1Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

zombie1 :

return 1;

}

bool zombie2Movement()

{

int value;

if (z2 == 0)

{

goto zombie2;

}

value = zombie2Direction();

if ( value == 1 ) //move left

{

if ( maze[zombie2X][zombie2Y - 1] == ' ' || maze[zombie2X][zombie2Y - 1] == 'H' || maze[zombie2X][zombie2Y - 1] == '\*' )

{

maze[zombie2X][zombie2Y] = previousItem;

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

zombie2Y = zombie2Y - 1;

previousItem = maze[zombie2X][zombie2Y];

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

maze[zombie2X][zombie2Y] = '2';

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 2) //move right

{

if ( maze[zombie2X][zombie2Y + 1] ==' ' || maze[zombie2X][zombie2Y + 1] == 'H' || maze[zombie2X][zombie2Y + 1] == '\*' )

{

maze[zombie2X][zombie2Y] = previousItem;

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

zombie2Y = zombie2Y + 1;

previousItem = maze[zombie2X][zombie2Y];

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

maze[zombie2X][zombie2Y] = '2';

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 3) //move up

{

if ( maze[zombie2X - 1][zombie2Y] ==' ' || maze[zombie2X - 1][zombie2Y] == 'H' || maze[zombie2X - 1][zombie2Y] == '\*' )

{

maze[zombie2X][zombie2Y] = previousItem;

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

zombie2X = zombie2X - 1;

previousItem = maze[zombie2X][zombie2Y];

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

maze[zombie2X][zombie2Y] = '2';

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 4) //move down

{

if ( maze[zombie2X + 1][zombie2Y] ==' ' || maze[zombie2X + 1][zombie2Y] == 'H' || maze[zombie2X + 1][zombie2Y] == '\*' )

{

maze[zombie2X][zombie2Y] = previousItem;

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

zombie2X = zombie2X + 1;

previousItem = maze[zombie2X][zombie2Y];

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

maze[zombie2X][zombie2Y] = '2';

gotoxy( zombie2X, zombie2Y);

cout <<maze[zombie2X][zombie2Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

zombie2 :

return 1;

}

bool zombie3Movement()

{

int value;

if (z3 == 0)

{

goto zombie3;

}

value = zombie3Direction();

if ( value == 1 ) //move left

{

if ( maze[zombie3X][zombie3Y - 1] == ' ' || maze[zombie3X][zombie3Y - 1] == 'H' || maze[zombie3X][zombie3Y - 1] == '\*' )

{

maze[zombie3X][zombie3Y] = previousItem;

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

zombie3Y = zombie3Y - 1;

previousItem = maze[zombie3X][zombie3Y];

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

maze[zombie3X][zombie3Y] = '3';

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 2) //move right

{

if ( maze[zombie3X][zombie3Y + 1] ==' ' || maze[zombie3X][zombie3Y + 1] == 'H' || maze[zombie3X][zombie3Y + 1] == '\*' )

{

maze[zombie3X][zombie3Y] = previousItem;

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

zombie3Y = zombie3Y + 1;

previousItem = maze[zombie3X][zombie3Y];

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

maze[zombie3X][zombie3Y] = '3';

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 3) //move up

{

if ( maze[zombie3X - 1][zombie3Y] ==' ' || maze[zombie3X - 1][zombie3Y] == 'H' || maze[zombie3X - 1][zombie3Y] == '\*' )

{

maze[zombie3X][zombie3Y] = previousItem;

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

zombie3X = zombie3X - 1;

previousItem = maze[zombie3X][zombie3Y];

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

maze[zombie3X][zombie3Y] = '3';

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 4) //move down

{

if ( maze[zombie3X + 1][zombie3Y] ==' ' || maze[zombie3X + 1][zombie3Y] == 'H' || maze[zombie3X + 1][zombie3Y] == '\*' )

{

maze[zombie3X][zombie3Y] = previousItem;

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

zombie3X = zombie3X + 1;

previousItem = maze[zombie3X][zombie3Y];

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

maze[zombie3X][zombie3Y] = '3';

gotoxy( zombie3X, zombie3Y);

cout <<maze[zombie3X][zombie3Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

zombie3 :

return 1;

}

bool zombie4Movement()

{

int value;

if (z4 == 0)

{

goto zombie4;

}

value = zombie4Direction();

if ( value == 1 ) //move left

{

if ( maze[zombie4X][zombie4Y - 1] == ' ' || maze[zombie4X][zombie4Y - 1] == 'H' || maze[zombie4X][zombie4Y - 1] == '\*' )

{

maze[zombie4X][zombie4Y] = previousItem;

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

zombie4Y = zombie4Y - 1;

previousItem = maze[zombie4X][zombie4Y];

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

maze[zombie4X][zombie4Y] = '4';

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 2) //move right

{

if ( maze[zombie4X][zombie4Y + 1] ==' ' || maze[zombie4X][zombie4Y + 1] == 'H' || maze[zombie4X][zombie4Y + 1] == '\*' )

{

maze[zombie4X][zombie4Y] = previousItem;

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

zombie4Y = zombie4Y + 1;

previousItem = maze[zombie4X][zombie4Y];

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

maze[zombie4X][zombie4Y] = '4';

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 3) //move up

{

if ( maze[zombie4X - 1][zombie4Y] ==' ' || maze[zombie4X - 1][zombie4Y] == 'H' || maze[zombie4X - 1][zombie4Y] == '\*' )

{

maze[zombie4X][zombie4Y] = previousItem;

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

zombie4X = zombie4X - 1;

previousItem = maze[zombie4X][zombie4Y];

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

maze[zombie4X][zombie4Y] = '4';

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 4) //move down

{

if ( maze[zombie4X + 1][zombie4Y] ==' ' || maze[zombie4X + 1][zombie4Y] == 'H' || maze[zombie4X + 1][zombie4Y] == '\*' )

{

maze[zombie4X][zombie4Y] = previousItem;

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

zombie4X = zombie4X + 1;

previousItem = maze[zombie4X][zombie4Y];

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

maze[zombie4X][zombie4Y] = '4';

gotoxy( zombie4X, zombie4Y);

cout <<maze[zombie4X][zombie4Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

zombie4 :

return 1;

}

bool zombie5Movement()

{

int value;

if (z0 == 0)

{

goto zombie5;

}

value = zombie5Direction();

if ( value == 1 ) //move left

{

if ( maze[zombie5X][zombie5Y - 1] == ' ' || maze[zombie5X][zombie5Y - 1] == 'H' || maze[zombie5X][zombie5Y - 1] == '\*' )

{

maze[zombie5X][zombie5Y] = previousItem;

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

zombie5Y = zombie5Y - 1;

previousItem = maze[zombie5X][zombie5Y];

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

maze[zombie5X][zombie5Y] = '0';

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 2) //move right

{

if ( maze[zombie5X][zombie5Y + 1] ==' ' || maze[zombie5X][zombie5Y + 1] == 'H' || maze[zombie5X][zombie5Y + 1] == '\*' )

{

maze[zombie5X][zombie5Y] = previousItem;

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

zombie5Y = zombie5Y + 1;

previousItem = maze[zombie5X][zombie5Y];

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

maze[zombie5X][zombie5Y] = '0';

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 3) //move up

{

if ( maze[zombie5X - 1][zombie5Y] ==' ' || maze[zombie5X - 1][zombie5Y] == 'H' || maze[zombie5X - 1][zombie5Y] == '\*' )

{

maze[zombie5X][zombie5Y] = previousItem;

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

zombie5X = zombie5X - 1;

previousItem = maze[zombie5X][zombie5Y];

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

maze[zombie5X][zombie5Y] = '0';

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

if ( value == 4) //move down

{

if ( maze[zombie5X + 1][zombie5Y] ==' ' || maze[zombie5X + 1][zombie5Y] == 'H' || maze[zombie5X + 1][zombie5Y] == '\*' )

{

maze[zombie5X][zombie5Y] = previousItem;

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

zombie5X = zombie5X + 1;

previousItem = maze[zombie5X][zombie5Y];

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

maze[zombie5X][zombie5Y] = '0';

gotoxy( zombie5X, zombie5Y);

cout <<maze[zombie5X][zombie5Y];

if (previousItem == 'H')

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

previousItem = ' ';

}

}

}

zombie5 :

return 1;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Shooting

void shootUp()

{

bulletUp = heroX - 1;

maze[bulletUp][heroY] = '|';

Sleep(300);

while ( maze[bulletUp - 1][heroY] == ' ' )

{

maze[bulletUp][heroY] = ' ';

gotoxy( bulletUp, heroY);

cout << maze[bulletUp][heroY];

bulletUp = bulletUp - 1;

maze[bulletUp][heroY] = '|';

gotoxy( bulletUp, heroY );

cout <<maze[bulletUp][heroY];

}

if ( maze[bulletUp - 1][heroY] == '#' || maze[bulletUp - 1][heroY] == '1' || maze[bulletUp - 1][heroY] == '2' || maze[bulletUp - 1][heroY] == '3' || maze[bulletUp - 1][heroY] == '4' || maze[bulletUp - 1][heroY] == '0' || maze[bulletUp - 1][heroY] == '\*')

{

maze[bulletUp][heroY] = ' ';

gotoxy(bulletUp, heroY);

cout <<maze[bulletUp][heroY];

if ( maze[bulletUp - 1][heroY] == '1' || maze[bulletUp - 1][heroY] == '2' || maze[bulletUp - 1][heroY] == '3' || maze[bulletUp - 1][heroY] == '4' )

{

if (maze[bulletUp - 1][heroY] == '1')

{

z1 = 0;

}

else if (maze[bulletUp - 1][heroY] == '2')

{

z2 = 0;

}

else if (maze[bulletUp - 1][heroY] == '3')

{

z3 = 0;

}

else if (maze[bulletUp - 1][heroY] == '4')

{

z4 = 0;

}

maze[bulletUp - 1][heroY] = ' ';

gotoxy( (bulletUp - 1), heroY);

cout <<maze[bulletUp - 1][heroY];

}

if ( maze[bulletUp - 1][heroY] == '0' )

{

z0 = 0;

maze[bulletUp - 1][heroY] = '\*';

gotoxy( (bulletUp - 1), heroY);

cout <<maze[bulletUp - 1][heroY];

}

}

if (maze[bulletUp - 1][heroY] == '<' || maze[bulletUp - 1][heroY] == '>' || maze[bulletUp - 1][heroY] == 'v' || maze[bulletUp - 1][heroY] == '^')

{

bHealth = bHealth - 1;

maze[bulletUp][heroY] = ' ';

gotoxy( bulletUp, heroY);

cout <<maze[bulletUp][heroY];

}

}

void shootDown()

{

bulletDown = heroX + 1;

maze[bulletDown][heroY] = '|';

Sleep(300);

while ( maze[bulletDown + 1][heroY] == ' ' )

{

maze[bulletDown][heroY] = ' ';

gotoxy( bulletDown, heroY);

cout << maze[bulletDown][heroY];

bulletDown = bulletDown + 1;

maze[bulletDown][heroY] = '|';

gotoxy( bulletDown, heroY );

cout <<maze[bulletDown][heroY];

}

if ( maze[bulletDown + 1][heroY] == '#' || maze[bulletDown + 1][heroY] == '1' || maze[bulletDown + 1][heroY] == '2' || maze[bulletDown + 1][heroY] == '3' || maze[bulletDown + 1][heroY] == '4' || maze[bulletDown + 1][heroY] == '0' || maze[bulletDown + 1][heroY] == '\*' )

{

maze[bulletDown][heroY] = ' ';

gotoxy(bulletDown, heroY);

cout <<maze[bulletDown][heroY];

if ( maze[bulletDown + 1][heroY] == '1' || maze[bulletDown + 1][heroY] == '2' || maze[bulletDown + 1][heroY] == '3' || maze[bulletDown + 1][heroY] == '4' )

{

if (maze[bulletDown + 1][heroY] == '1')

{

z1 = 0;

}

else if (maze[bulletDown + 1][heroY] == '2')

{

z2 = 0;

}

else if (maze[bulletDown + 1][heroY] == '3')

{

z3 = 0;

}

else if (maze[bulletDown + 1][heroY] == '4')

{

z4 = 0;

}

maze[bulletDown + 1][heroY] = ' ';

gotoxy( (bulletDown + 1), heroY);

cout <<maze[bulletDown + 1][heroY];

}

if ( maze[bulletDown + 1][heroY] == '0' )

{

z0 = 0;

maze[bulletDown + 1][heroY] = '\*';

gotoxy( (bulletDown + 1), heroY);

cout <<maze[bulletDown + 1][heroY];

}

}

if ( maze[bulletDown + 1][heroY] == '<' || maze[bulletDown + 1][heroY] == '>' || maze[bulletDown + 1][heroY] == 'v' || maze[bulletDown + 1][heroY] == '^' )

{

bHealth = bHealth - 1;

maze[bulletDown][heroY] = ' ';

gotoxy( bulletDown, heroY);

cout <<maze[bulletDown][heroY];

}

}

void shootLeft()

{

bulletLeft = heroY - 1;

maze[heroX][bulletLeft] = '-';

Sleep(300);

while ( maze[heroX][bulletLeft - 1] == ' ' )

{

maze[heroX][bulletLeft] = ' ';

gotoxy( heroX, bulletLeft);

cout << maze[heroX][bulletLeft];

bulletLeft = bulletLeft - 1;

maze[heroX][bulletLeft] = '-';

gotoxy( heroX, bulletLeft );

cout <<maze[heroX][bulletLeft];

}

if ( maze[heroX][bulletLeft - 1] == '#' || maze[heroX][bulletLeft - 1] == '1' || maze[heroX][bulletLeft - 1] == '2' || maze[heroX][bulletLeft - 1] == '3' || maze[heroX][bulletLeft - 1] == '4' || maze[heroX][bulletLeft - 1] == '0' || maze[heroX][bulletLeft - 1] == '\*' )

{

maze[heroX][bulletLeft] = ' ';

gotoxy(heroX, bulletLeft);

cout <<maze[heroX][bulletLeft];

if ( maze[heroX][bulletLeft - 1] == '1' || maze[heroX][bulletLeft - 1] == '2' || maze[heroX][bulletLeft - 1] == '3' || maze[heroX][bulletLeft - 1] == '4' )

{

if (maze[heroX][bulletLeft - 1] == '1')

{

z1 = 0;

}

else if (maze[heroX][bulletLeft - 1] == '2')

{

z2 = 0;

}

else if (maze[heroX][bulletLeft - 1] == '3')

{

z3 = 0;

}

else if (maze[heroX][bulletLeft - 1] == '4')

{

z4 = 0;

}

maze[heroX][bulletLeft - 1] = ' ';

gotoxy( heroX, (bulletLeft - 1) );

cout <<maze[heroX][bulletLeft - 1];

}

if ( maze[heroX][bulletLeft - 1] == '0' )

{

z0 = 0;

maze[heroX][bulletLeft - 1] = '\*';

gotoxy( heroX, (bulletLeft - 1) );

cout <<maze[heroX][bulletLeft - 1];

}

}

if ( maze[heroX][bulletLeft - 1] == '>' || maze[heroX][bulletLeft - 1] == '<' || maze[heroX][bulletLeft - 1] == '^' || maze[heroX][bulletLeft - 1] == 'v')

{

bHealth = bHealth - 1;

maze[heroX][bulletLeft] = ' ';

gotoxy( heroX, bulletLeft);

cout <<maze[heroX][bulletLeft];

}

}

void shootRight()

{

bulletRight = heroY + 1;

maze[heroX][bulletRight] = '-';

Sleep(300);

while ( maze[heroX][bulletRight + 1] == ' ' )

{

maze[heroX][bulletRight] = ' ';

gotoxy( heroX, bulletRight);

cout << maze[heroX][bulletRight];

bulletRight = bulletRight + 1;

maze[heroX][bulletRight] = '-';

gotoxy( heroX, bulletRight );

cout <<maze[heroX][bulletRight];

}

if ( maze[heroX][bulletRight + 1] == '#' || maze[heroX][bulletRight + 1] == '1' || maze[heroX][bulletRight + 1] == '2' || maze[heroX][bulletRight + 1] == '3' || maze[heroX][bulletRight + 1] == '4' || maze[heroX][bulletRight + 1] == '0' || maze[heroX][bulletRight + 1] == '\*' )

{

maze[heroX][bulletRight] = ' ';

gotoxy(heroX, bulletRight);

cout <<maze[heroX][bulletRight];

if ( maze[heroX][bulletRight + 1] == '1' || maze[heroX][bulletRight + 1] == '2' || maze[heroX][bulletRight + 1] == '3' || maze[heroX][bulletRight + 1] == '4' )

{

if (maze[heroX][bulletRight + 1] == '1')

{

z1 = 0;

}

else if (maze[heroX][bulletRight + 1] == '2')

{

z2 = 0;

}

else if (maze[heroX][bulletRight + 1] == '3')

{

z3 = 0;

}

else if (maze[heroX][bulletRight + 1] == '4')

{

z4 = 0;

}

maze[heroX][bulletRight + 1] = ' ';

gotoxy( heroX, (bulletRight + 1) );

cout <<maze[heroX][bulletRight + 1];

}

if ( maze[heroX][bulletRight + 1] == '0' )

{

z0 = 0;

maze[heroX][bulletRight + 1] = '\*';

gotoxy( heroX, (bulletRight + 1) );

cout <<maze[heroX][bulletRight + 1];

}

}

if ( maze[heroX][bulletRight + 1] == '<' || maze[heroX][bulletRight + 1] == '>' || maze[heroX][bulletRight + 1] == 'v' || maze[heroX][bulletRight + 1] == '^')

{

bHealth = bHealth - 1;

maze[heroX][bulletRight] = ' ';

gotoxy( heroX, bulletRight);

cout <<maze[heroX][bulletRight];

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Boss Condition

bool bossFight()

{

int value = bossDirection();

if ( value == 1 ) //move left

{

if ( maze[bossHeadX][bossHeadY - 1] != '#' && maze[bossLeftX][bossLeftY - 1] != '#' && maze[bossRightX][bossRightY - 1] != '#' && maze[bossTailX][bossTailY - 1] != '#' )

{

maze[bossLeftX][bossLeftY] = ' ';

gotoxy( bossLeftX, bossLeftY);

cout <<maze[bossLeftX][bossLeftY];

bossLeftY = bossLeftY - 1;

maze[bossLeftX][bossLeftY] = '<';

gotoxy( bossLeftX, bossLeftY);

cout <<maze[bossLeftX][bossLeftY];

maze[bossHeadX][bossHeadY] = ' ';

gotoxy( bossHeadX, bossHeadY);

cout <<maze[bossHeadX][bossHeadY];

bossHeadY = bossHeadY - 1;

maze[bossHeadX][bossHeadY] = '^';

gotoxy( bossHeadX, bossHeadY);

cout <<maze[bossHeadX][bossHeadY];

maze[bossTailX][bossTailY] = ' ';

gotoxy( bossTailX, bossTailY);

cout <<maze[bossTailX][bossTailY];

bossTailY = bossTailY - 1;

maze[bossTailX][bossTailY] = 'v';

gotoxy( bossTailX, bossTailY);

cout <<maze[bossTailX][bossTailY];

maze[bossMidX][bossMidY] = ' ';

gotoxy( bossMidX, bossMidY);

cout <<maze[bossMidX][bossMidY];

bossMidY = bossMidY - 1;

maze[bossMidX][bossMidY] = 'o';

gotoxy( bossMidX, bossMidY);

cout <<maze[bossMidX][bossMidY];

maze[bossRightX][bossRightY] = ' ';

gotoxy( bossRightX, bossRightY);

cout <<maze[bossRightX][bossRightY];

bossRightY = bossRightY - 1;

maze[bossRightX][bossRightY] = '>';

gotoxy( bossRightX, bossRightY);

cout <<maze[bossRightX][bossRightY];

}

}

if ( value == 2) //move right

{

if ( maze[bossHeadX][bossHeadY + 1] != '#' && maze[bossLeftX][bossLeftY + 1] != '#' && maze[bossRightX][bossRightY + 1] != '#' && maze[bossTailX][bossTailY + 1] != '#' )

{

maze[bossRightX][bossRightY] = ' ';

gotoxy( bossRightX, bossRightY);

cout <<maze[bossRightX][bossRightY];

bossRightY = bossRightY + 1;

maze[bossRightX][bossRightY] = '>';

gotoxy( bossRightX, bossRightY);

cout <<maze[bossRightX][bossRightY];

maze[bossHeadX][bossHeadY] = ' ';

gotoxy( bossHeadX, bossHeadY);

cout <<maze[bossHeadX][bossHeadY];

bossHeadY = bossHeadY + 1;

maze[bossHeadX][bossHeadY] = '^';

gotoxy( bossHeadX, bossHeadY);

cout <<maze[bossHeadX][bossHeadY];

maze[bossTailX][bossTailY] = ' ';

gotoxy( bossTailX, bossTailY);

cout <<maze[bossTailX][bossTailY];

bossTailY = bossTailY + 1;

maze[bossTailX][bossTailY] = 'v';

gotoxy( bossTailX, bossTailY);

cout <<maze[bossTailX][bossTailY];

maze[bossMidX][bossMidY] = ' ';

gotoxy( bossMidX, bossMidY);

cout <<maze[bossMidX][bossMidY];

bossMidY = bossMidY + 1;

maze[bossMidX][bossMidY] = 'o';

gotoxy( bossMidX, bossMidY);

cout <<maze[bossMidX][bossMidY];

maze[bossLeftX][bossLeftY] = ' ';

gotoxy( bossLeftX, bossLeftY);

cout <<maze[bossLeftX][bossLeftY];

bossLeftY = bossLeftY + 1;

maze[bossLeftX][bossLeftY] = '<';

gotoxy( bossLeftX, bossLeftY);

cout <<maze[bossLeftX][bossLeftY];

}

}

if ( value == 3) //move up

{

if ( maze[bossHeadX - 1][bossHeadY] != '#' && maze[bossLeftX - 1][bossLeftY] != '#' && maze[bossRightX - 1][bossRightY] != '#' && maze[bossTailX - 1][bossTailY] != '#' )

{

maze[bossHeadX][bossHeadY] = ' ';

gotoxy( bossHeadX, bossHeadY);

cout <<maze[bossHeadX][bossHeadY];

bossHeadX = bossHeadX - 1;

maze[bossHeadX][bossHeadY] = '^';

gotoxy( bossHeadX, bossHeadY);

cout <<maze[bossHeadX][bossHeadY];

maze[bossMidX][bossMidY] = ' ';

gotoxy( bossMidX, bossMidY);

cout <<maze[bossMidX][bossMidY];

bossMidX = bossMidX - 1;

maze[bossMidX][bossMidY] = 'o';

gotoxy( bossMidX, bossMidY);

cout <<maze[bossMidX][bossMidY];

maze[bossLeftX][bossLeftY] = ' ';

gotoxy( bossLeftX, bossLeftY);

cout <<maze[bossLeftX][bossLeftY];

bossLeftX = bossLeftX - 1;

maze[bossLeftX][bossLeftY] = '<';

gotoxy( bossLeftX, bossLeftY);

cout <<maze[bossLeftX][bossLeftY];

maze[bossRightX][bossRightY] = ' ';

gotoxy( bossRightX, bossRightY);

cout <<maze[bossRightX][bossRightY];

bossRightX = bossRightX - 1;

maze[bossRightX][bossRightY] = '>';

gotoxy( bossRightX, bossRightY);

cout <<maze[bossRightX][bossRightY];

maze[bossTailX][bossTailY] = ' ';

gotoxy( bossTailX, bossTailY);

cout <<maze[bossTailX][bossTailY];

bossTailX = bossTailX - 1;

maze[bossTailX][bossTailY] = 'v';

gotoxy( bossTailX, bossTailY);

cout <<maze[bossTailX][bossTailY];

}

}

if ( value == 4) //move down

{

if ( maze[bossHeadX + 1][bossHeadY] != '#' && maze[bossLeftX + 1][bossLeftY] != '#' && maze[bossRightX + 1][bossRightY] != '#' && maze[bossTailX + 1][bossTailY] != '#' )

{

maze[bossTailX][bossTailY] = ' ';

gotoxy( bossTailX, bossTailY);

cout <<maze[bossTailX][bossTailY];

bossTailX = bossTailX + 1;

maze[bossTailX][bossTailY] = 'v';

gotoxy( bossTailX, bossTailY);

cout <<maze[bossTailX][bossTailY];

maze[bossRightX][bossRightY] = ' ';

gotoxy( bossRightX, bossRightY);

cout <<maze[bossRightX][bossRightY];

bossRightX = bossRightX + 1;

maze[bossRightX][bossRightY] = '>';

gotoxy( bossRightX, bossRightY);

cout <<maze[bossRightX][bossRightY];

maze[bossLeftX][bossLeftY] = ' ';

gotoxy( bossLeftX, bossLeftY);

cout <<maze[bossLeftX][bossLeftY];

bossLeftX = bossLeftX + 1;

maze[bossLeftX][bossLeftY] = '<';

gotoxy( bossLeftX, bossLeftY);

cout <<maze[bossLeftX][bossLeftY];

maze[bossMidX][bossMidY] = ' ';

gotoxy( bossMidX, bossMidY);

cout <<maze[bossMidX][bossMidY];

bossMidX = bossMidX + 1;

maze[bossMidX][bossMidY] = 'o';

gotoxy( bossMidX, bossMidY);

cout <<maze[bossMidX][bossMidY];

maze[bossHeadX][bossHeadY] = ' ';

gotoxy( bossHeadX, bossHeadY);

cout <<maze[bossHeadX][bossHeadY];

bossHeadX = bossHeadX + 1;

maze[bossHeadX][bossHeadY] = '^';

gotoxy( bossHeadX, bossHeadY);

cout <<maze[bossHeadX][bossHeadY];

}

}

if (heroX == bossMidX)

{

if (heroY > bossMidY)

{

horizontalBullet = bossRightY + 1;

maze[bossMidX][horizontalBullet] = '-';

Sleep(300);

while ( maze[bossMidX][horizontalBullet + 1] == ' ')

{

maze[bossMidX][horizontalBullet] = ' ';

gotoxy( bossMidX, horizontalBullet);

cout <<maze[bossMidX][horizontalBullet];

horizontalBullet = horizontalBullet + 1;

maze[ bossMidX][horizontalBullet] = '-';

gotoxy( bossMidX, horizontalBullet );

cout <<maze[bossMidX][horizontalBullet];

}

if (maze[bossMidX][horizontalBullet + 1] == '#' || maze[bossMidX][horizontalBullet + 1] == 'H' )

{

maze[bossMidX][horizontalBullet] = ' ';

gotoxy(bossMidX, horizontalBullet);

cout <<maze[bossMidX][horizontalBullet];

if (maze[bossMidX][horizontalBullet + 1] == 'H' )

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

}

}

}

if (heroY < bossMidY)

{

horizontalBullet = bossLeftY - 1;

maze[bossMidX][horizontalBullet] = '-';

Sleep(300);

while ( maze[bossMidX][horizontalBullet - 1] == ' ')

{

maze[bossMidX][horizontalBullet] = ' ';

gotoxy( bossMidX, horizontalBullet);

cout <<maze[bossMidX][horizontalBullet];

horizontalBullet = horizontalBullet - 1;

maze[ bossMidX][horizontalBullet] = '-';

gotoxy( bossMidX, horizontalBullet );

cout <<maze[bossMidX][horizontalBullet];

}

if (maze[bossMidX][horizontalBullet - 1] == '#' || maze[bossMidX][horizontalBullet - 1] == 'H' )

{

maze[bossMidX][horizontalBullet] = ' ';

gotoxy(bossMidX, horizontalBullet);

cout <<maze[bossMidX][horizontalBullet];

if (maze[bossMidX][horizontalBullet - 1] == 'H' )

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

}

}

}

}

if (heroY == bossMidY)

{

if (heroX > bossMidX)

{

verticalBullet = bossTailX + 1;

maze[verticalBullet][bossMidY] = '|';

Sleep(300);

while ( maze[verticalBullet + 1][bossMidY] == ' ')

{

maze[verticalBullet][bossMidY] = ' ';

gotoxy( verticalBullet, bossMidY);

cout <<maze[verticalBullet][bossMidY];

verticalBullet = verticalBullet + 1;

maze[verticalBullet][bossMidY] = '|';

gotoxy(verticalBullet, bossMidY );

cout <<maze[verticalBullet][bossMidY];

}

if (maze[verticalBullet][bossMidY + 1] == '#' || maze[verticalBullet][bossMidY + 1] == 'H' )

{

maze[verticalBullet][bossMidY] = ' ';

gotoxy(verticalBullet, bossMidY);

cout <<maze[verticalBullet][bossMidY];

if (maze[verticalBullet][bossMidY + 1] == 'H' )

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

}

}

}

if (heroY < bossMidY)

{

verticalBullet = bossTailX - 1;

maze[verticalBullet][bossMidY] = '|';

Sleep(300);

while ( maze[verticalBullet - 1][bossMidY] == ' ')

{

maze[verticalBullet][bossMidY] = ' ';

gotoxy( verticalBullet, bossMidY);

cout <<maze[verticalBullet][bossMidY];

verticalBullet = verticalBullet - 1;

maze[verticalBullet][bossMidY] = '|';

gotoxy(verticalBullet, bossMidY );

cout <<maze[verticalBullet][bossMidY];

}

if (maze[verticalBullet][bossMidY - 1] == '#' || maze[verticalBullet][bossMidY - 1] == 'H' )

{

maze[verticalBullet][bossMidY] = ' ';

gotoxy(verticalBullet, bossMidY);

cout <<maze[verticalBullet][bossMidY];

if (maze[verticalBullet][bossMidY - 1] == 'H' )

{

health = health - 1;

maze[heroX][heroY] = ' ';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

heroX = 3;

heroY = 3;

maze[heroX][heroY] = 'H';

gotoxy( heroX, heroY);

cout <<maze[heroX][heroY];

}

}

}

}

return 1;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Boss Direction

int bossDirection()

{

bossHealth();

srand(time(0));

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

return result;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Boss Health

void bossHealth()

{

gotoxy(8,120);

cout <<"Boss health = " <<bHealth;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_name of the game

void name()

{

cout <<" \_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_ " <<endl;

cout <<"| | | | | | | | | / | | | | | | | " <<endl;

cout <<"| | | | | | | | / | | | | | | | " <<endl;

cout <<"| | | | | | | | / | | |\_\_\_\_\_ | | | | " <<endl;

cout <<"|\_\_\_\_\_\_| | | |\_\_\_\_\_\_| | |/ | | | | |\_\_\_\_\_\_| |\_\_\_\_\_ " <<endl;

cout <<"| | | | | | | |\\ | | | | | | | " <<endl;

cout <<"| | | | | | | | \\ | | | | | | | " <<endl;

cout <<"| | | | | | |\_\_\_\_\_\_| | \\ |\_\_\_\_\_\_| | | | | |\_\_\_\_\_\_ "<<endl <<endl <<endl;

cout <<" \_\_\_ \_\_\_\_\_ \_\_\_\_\_\_ \_\_\_ " <<endl;

cout <<"| | |\\ | | \\ | | | | \\ " <<endl;

cout <<"| | | \\ | | \\ | | | | \\ " <<endl;

cout <<"| | | \\ | | | | | | | | " <<endl;

cout <<"| | | \\ | | | |\_\_\_\_ |\_\_\_\_\_\_| | | " <<endl;

cout <<"| | | \\ | | | | | | | | " <<endl;

cout <<"| | | \\ | | / | | | | / " <<endl;

cout <<"|\_\_\_\_\_\_| | \\| |\_\_\_/ |\_\_\_\_\_ | | |\_\_\_/ "<<endl <<endl <<endl <<endl <<endl;

cout <<"Press any key to start.";

getch();

}