**Berzerk 2D Game**

****

Session 2022\_2026

**Submitted by:**

Iqra Tariq 2022-CS-29

**Supervised by:**

**Maida shahid**

Department of computer science

**University of engineering and technology**

**Lahore Pakistan**

**Description**

Berzerk places the player in a series of top-down, maze like rooms containing armed robot.it is simply a shooting game in which character traverses a maze to shoot robots. In berzerk an unnamed human fighting his way though randomly generated maxes. It has a yellow color in first level but as level change color also change. he loves to kill the anime like robot in this game. He spends his whole day to kill the all robot there is some food pallet in this game when human eat this food pallet he gains some power.

He randomly generated mazes overrun with killer robots. the aim is to destroy all robots. When he destroys all robots he moves in next level with different color. the number of enemy increase with next level. Two control keys implemented to provide the player with more precise control, enemies with different behaviors were added to make the game challenging. Coming into contact with an enemy costs the player one life, but extra lives can be earned at certain point total.

The left arrow key control the movement of character, while the right key control the direction

**Game character Description**

**Player**

A human unnamed character is the main character which is known by his human like shape.

He is loves to kill the enemies. He is the hero of the game, admired for his bravery and determination in the face of danger.

**Enemies**

There are almost Three enemies in game. They all have multiple colors. They all want to kill the main player. One enemy move random at the top of the maze it can generate the bullets left and right both side. Second enemy move up and down it can generate it’s bullets left side.

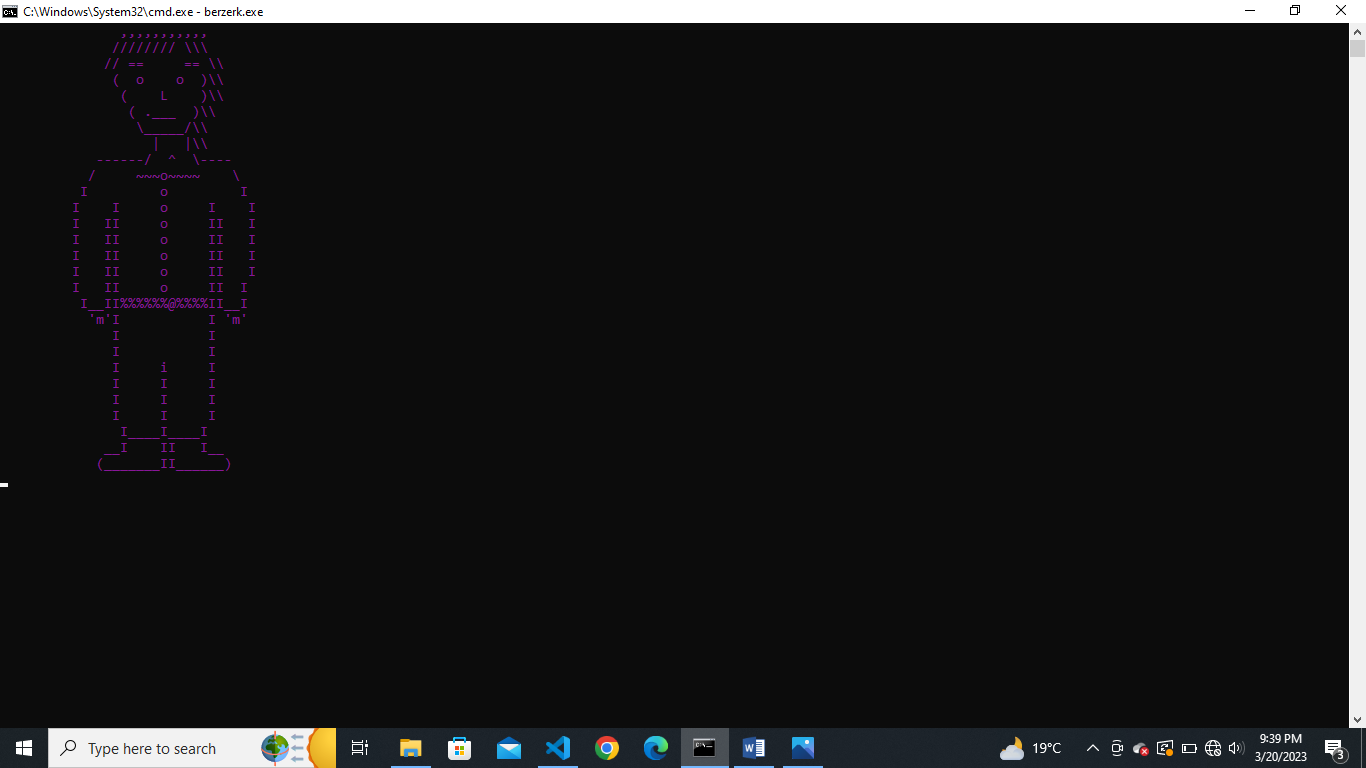
**Rules and instructions**

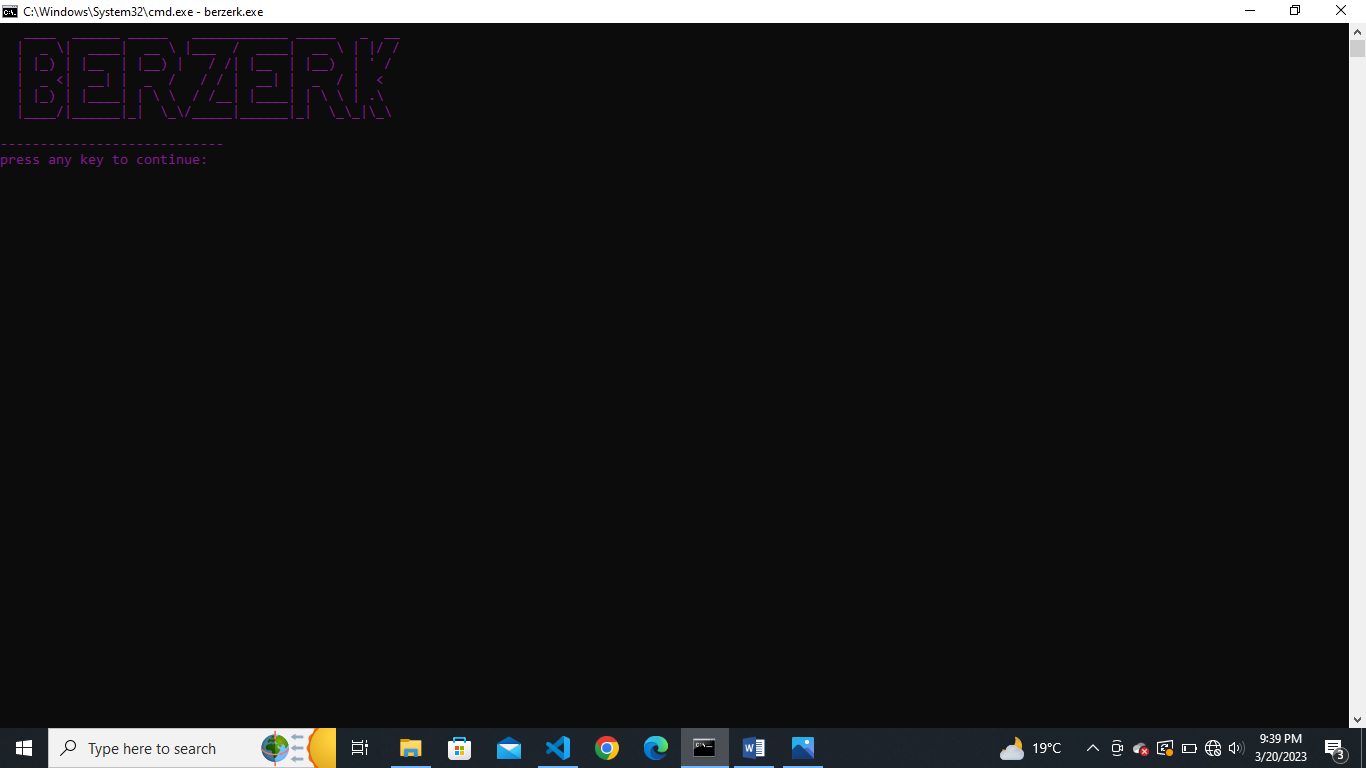
Player can generate bullets at right side. When bullets hit with enemies **Score** will be increase and the **specific enemy’ health** will be decrease. When the enemies’ bullets hit the Player the health of Player will decrease. When the Player health reaches at Zero you will lost, and when all enemies Health reaches at zero you will win the game.

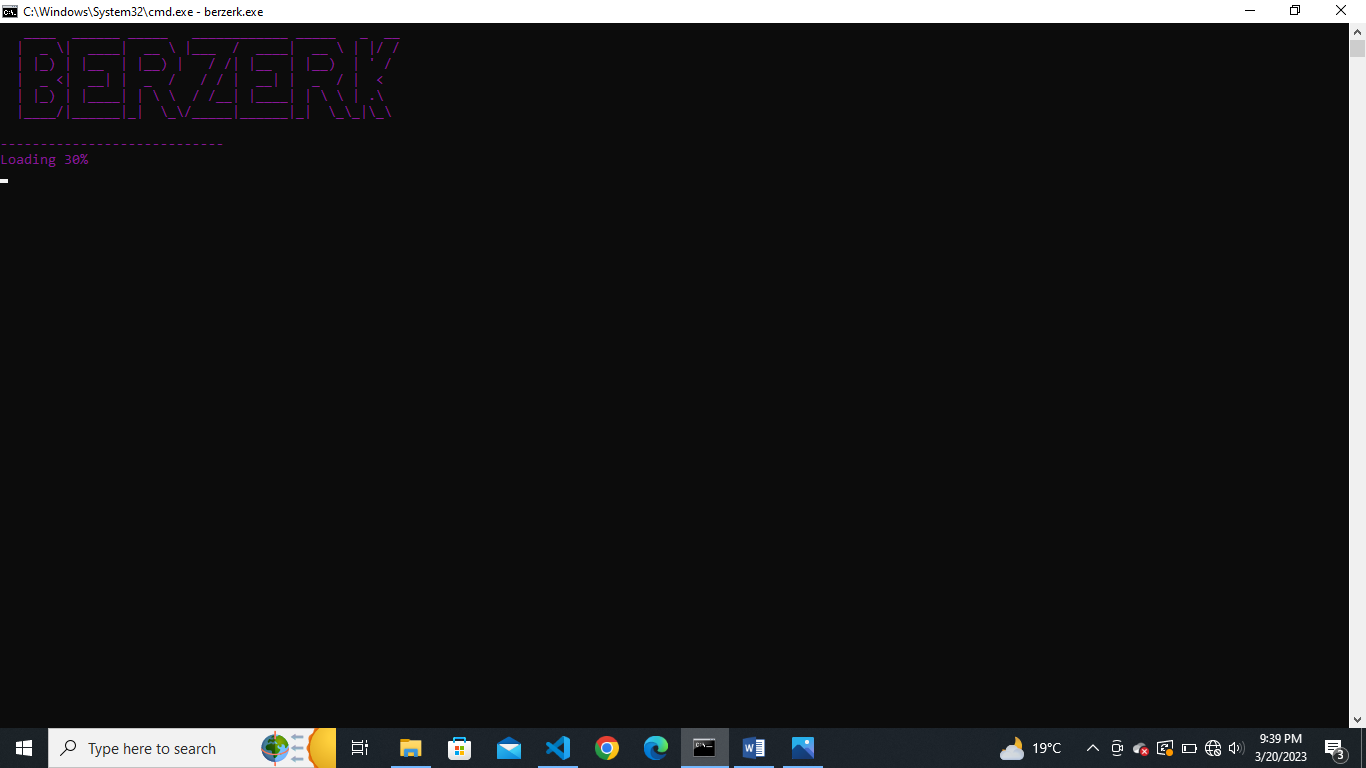
**Goal of Game**

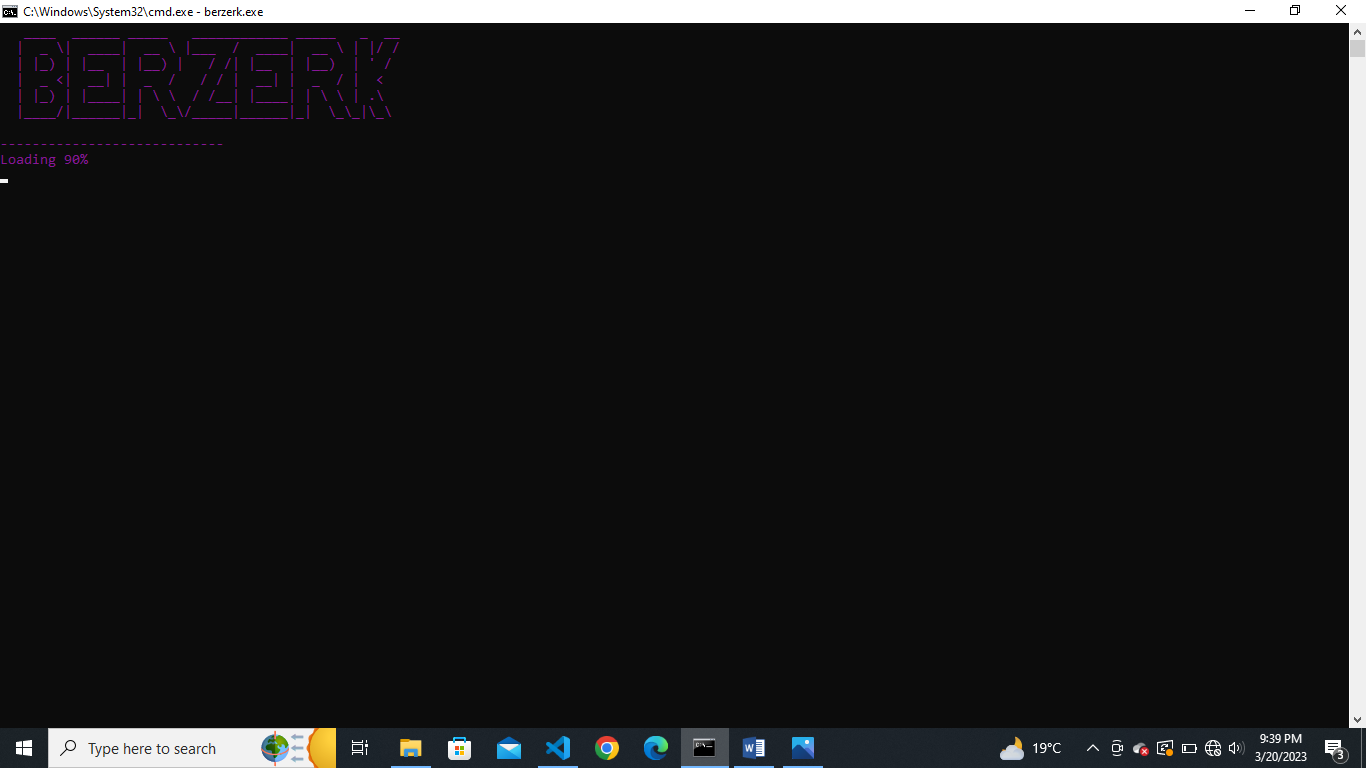
The goal of the game is to kill all robot (enemy) that have been across the maze.

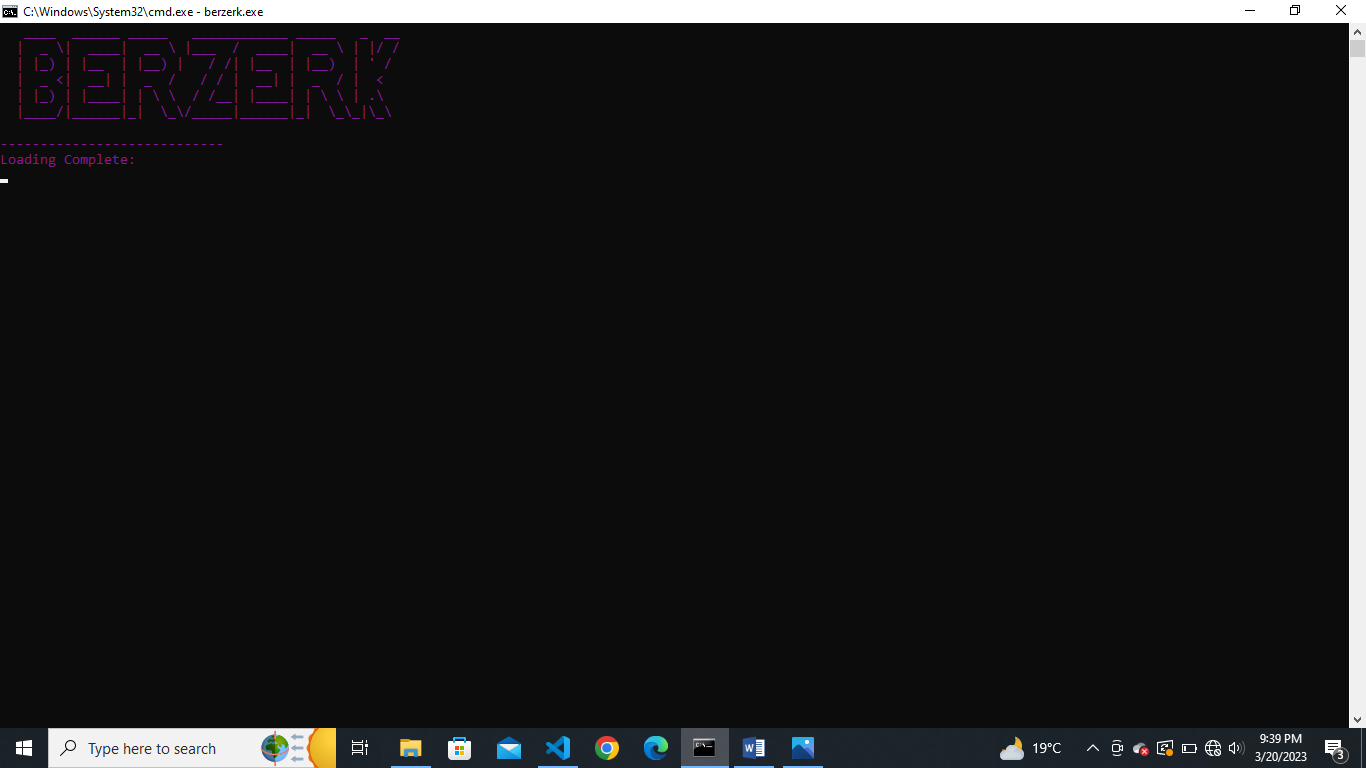
**Wireframes**

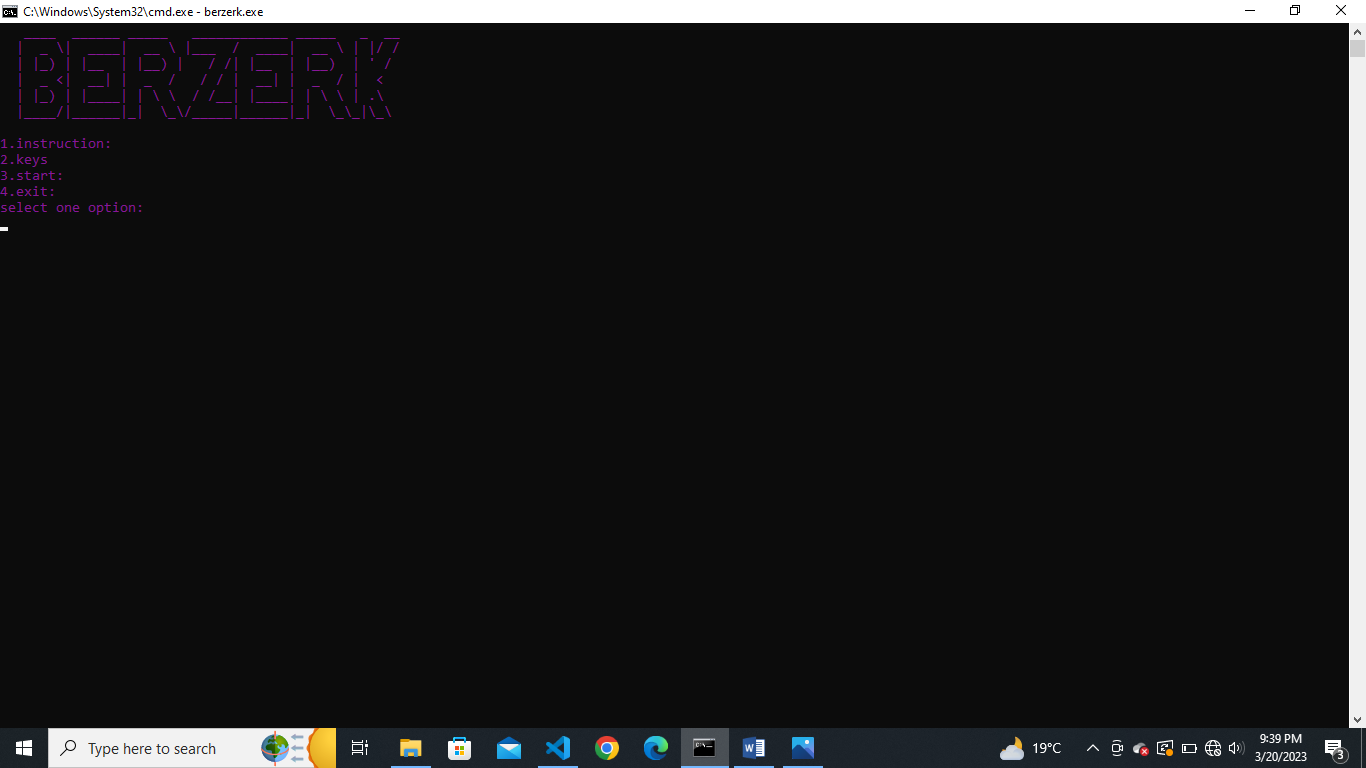
****

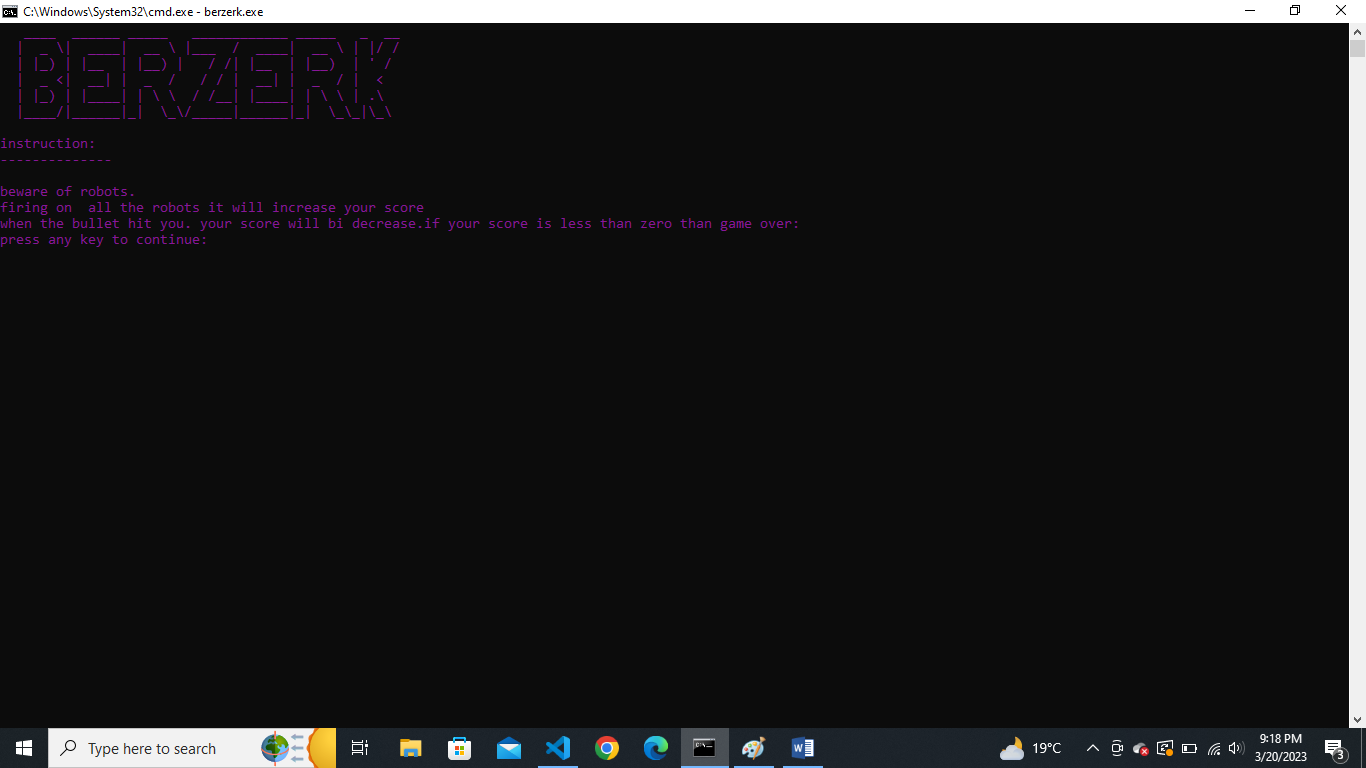
****

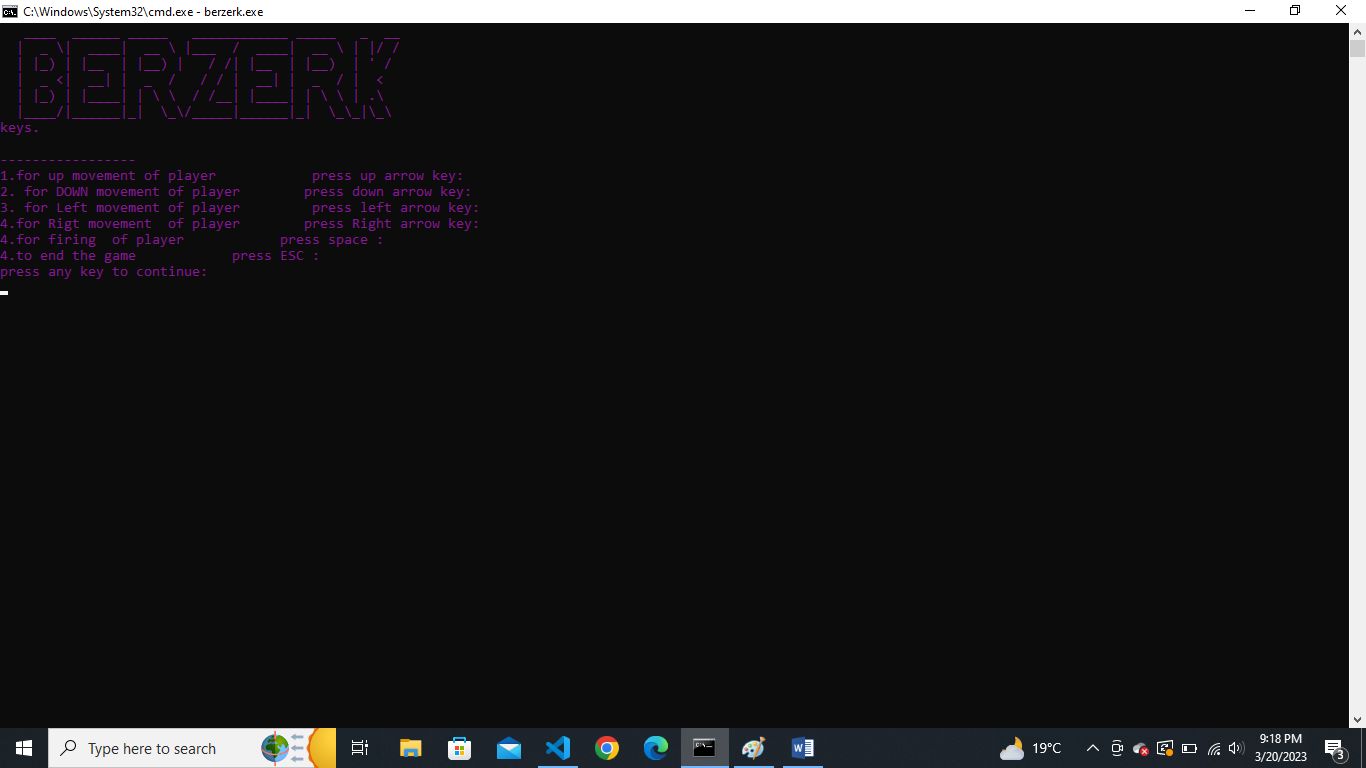
****

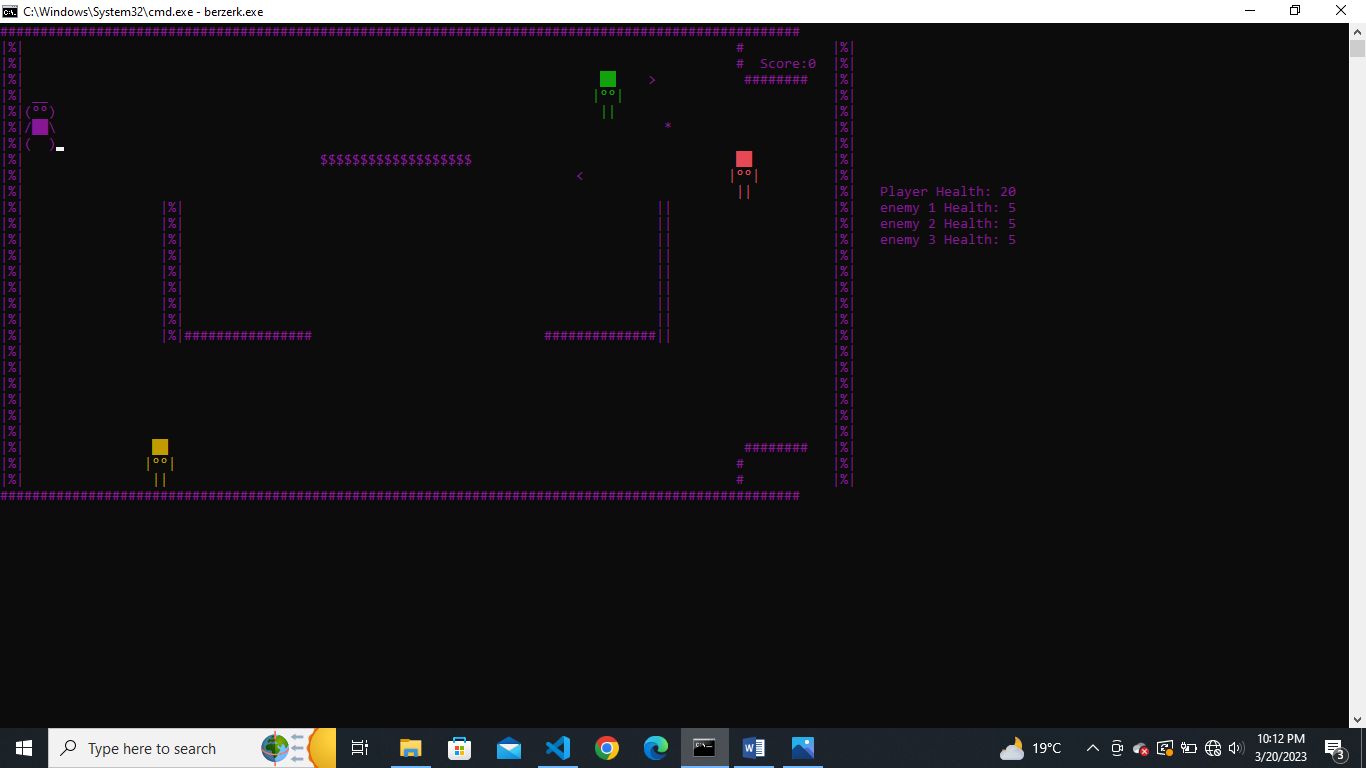
****

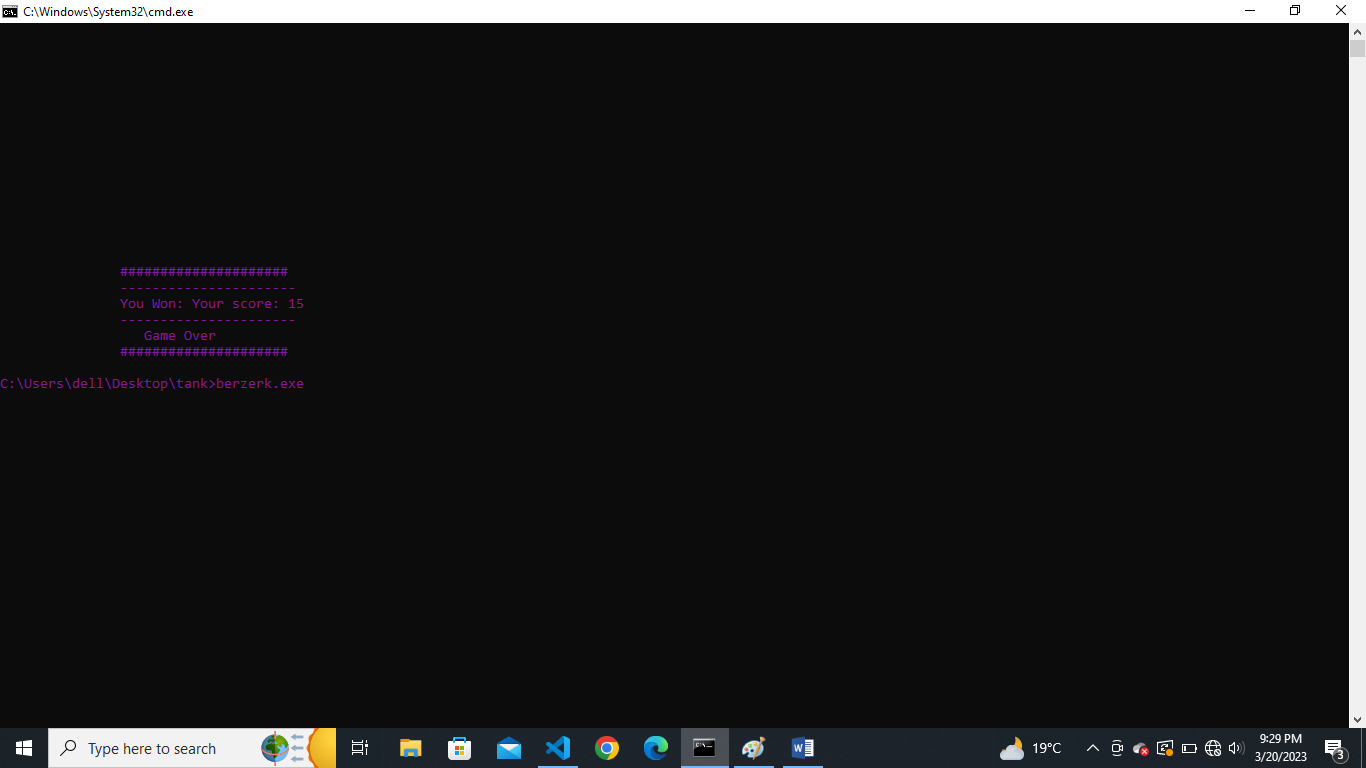
****

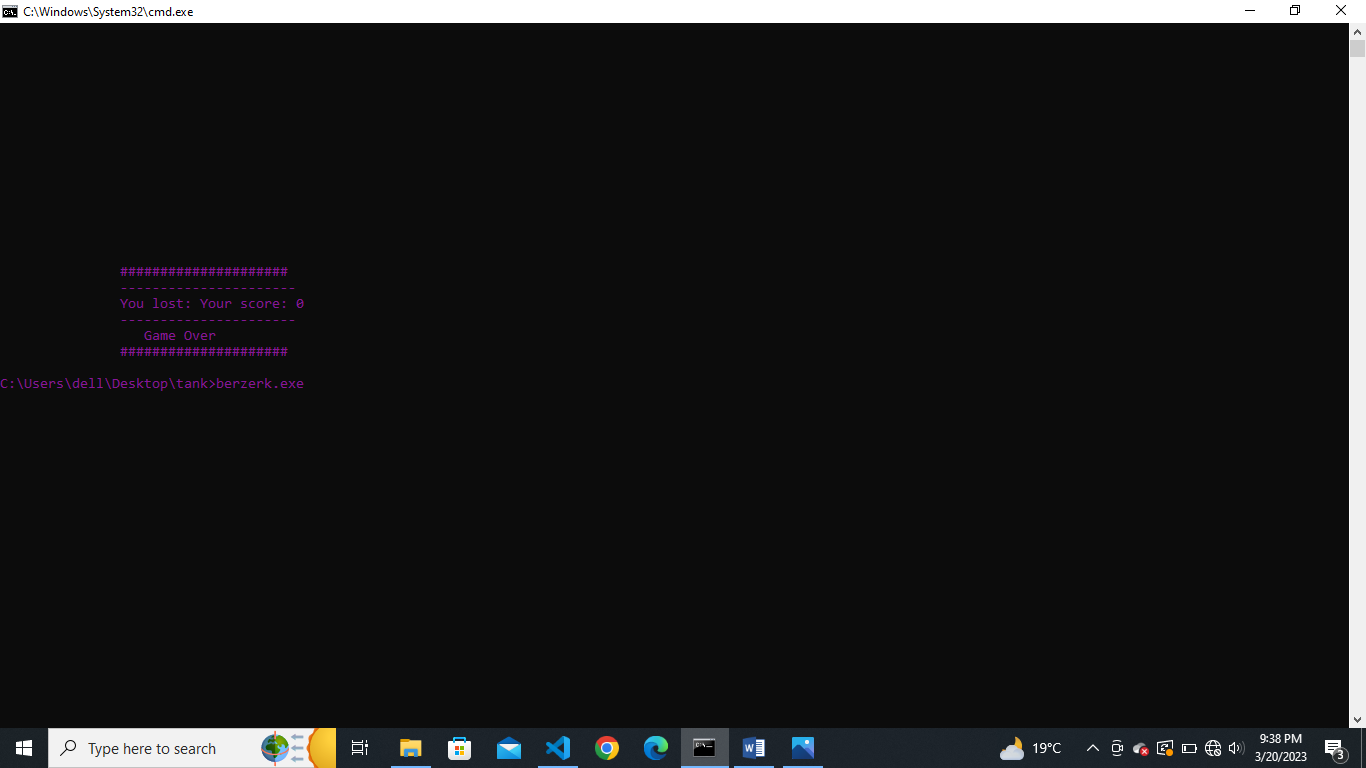
****

****

****

****

****



**Data Structure (2D Arrays)**

// player Dtat Structure

int tankx = 3;

int tanky = 3;

char box2 = 248, box3 = 219, box4 = 234, ear1 = 62, ear2 = 60; // player character

char tank1[4][4] = {{' ', '\_', '\_', ' '},

                    {'(', box2, box2, ')'},

                    {'/', box3, box3, '\\'},

                    {'(', ' ', ' ', ')'}};

// player bullets

int bulletX[1000];

int bulletY[1000];

bool isBulletActive[1000];

int bulletCount = 0;

void printHealthOfPlayer();

int playerHealth = 20;

// player functionalities end

// enemy1 at top functionalities srart

//  enemy 1 character

string enemyDirection = "Left";

int enemyX = 89;

int enemyY = 3;

//Enemy1 Data Structure

char enemybox3 = 248, enemybox = 219;

char enemy[3][4] = {{' ', enemybox, enemybox, ' '},

                    {'|', enemybox3, enemybox3, '|'},

                    {' ', '|', '|', ' '}};

//  enemy2 character

int enemy2X = 91;

int enemy2Y = 15;

char enemy2box3 = 248, enemy2box = 219;

char enemY[3][4] = {{' ', enemy2box, enemy2box, ' '},

                    {'|', enemy2box3, enemy2box3, '|'},

                    {' ', '|', '|', ' '}};

//  enemy 3 character

int enemy3X = 3;

int enemy3Y = 26;

char enemy3box3 = 248, enemy2box2 = 219;

char enemY3[3][4] = {{' ', enemy2box2, enemy2box2, ' '},

                     {'|', enemy3box3, enemy3box3, '|'},

                     {' ', '|', '|', ' '}};

**Function prototypes**

#include <iostream>

#include <conio.h>

#include <windows.h>

using namespace std;

void topHeader();

void mainmenu();

void instruction();

void start();

void submenu();

void key();

// supporting functions

char getCharAtxy(short int x, short int y);

void gotoxy(int x, int y);

void gameOver();

void addScore();

void printScore();

void printmaze();

void youWon();

void generateColors();//function for giving color to characters

// player 's functionalities start

void generateC();

void movetankleft();

void movetankright();

void movetankup();

void movetankdown();

void erase();

void generatebullet();

void movebullet();

void makebulletInActive(int z);

void eraseBullet(int x, int y);

void printBullet(int x, int y);

void collisionWithenemy3(); // player bullets collied with enemy3 at bottom

#include <iostream>

#include <windows.h>

using namespace std;

// supporting functions

char getCharAtxy(short int x, short int y);

void gotoxy(int x, int y);

void gameOver();

void addScore();

void printScore();

void printmaze();

void youWon();

int score = 0;

// player 's functionalities start

void generateC();

void movetankleft();

void movetankright();

void movetankup();

void movetankdown();

void erase();

void generatebullet();

void movebullet();

void makebulletInActive(int z);

void eraseBullet(int x, int y);

void printBullet(int x, int y);

void collisionWithenemy3(); // player bullets collied with enemy3 at bottom

void collisionWithenemy();  // player bullets collied with enemy2 at left side

void collisionWithenemy1(); // player bullets collied with enemy1 at top

// player Dtat Structure

int tankx = 3;

int tanky = 3;

char box2 = 248, box3 = 219, box4 = 234, ear1 = 62, ear2 = 60; // player character

char tank1[4][4] = {{' ', '\_', '\_', ' '},

                    {'(', box2, box2, ')'},

                    {'/', box3, box3, '\\'},

                    {'(', ' ', ' ', ')'}};

// player bullets

int bulletX[1000];

int bulletY[1000];

bool isBulletActive[1000];

int bulletCount = 0;

void printHealthOfPlayer();

int playerHealth = 20;

// player functionalities end

// enemy1 at top functionalities srart

//  enemy 1 character

string enemyDirection = "Left";

int enemyX = 89;

int enemyY = 3;

//Enemy1 Data Structure

char enemybox3 = 248, enemybox = 219;

char enemy[3][4] = {{' ', enemybox, enemybox, ' '},

                    {'|', enemybox3, enemybox3, '|'},

                    {' ', '|', '|', ' '}};

void printEnemy();

void eraseEnemy();

void moveEnemy();

void printEnemyBullet1(int x, int y); // enemy 1 bullets for right side

void generateEnemyBullet1();

void eraseEnemyBullet1(int x, int y);

void moveEnemyBullet1();

void makeEnemyBulletInActive1(int z);

void generateEnemy1Bullet();

void printEnemy1Bullet(int x, int y); // enemy 1 bullets for left side

void eraseEnemy1Bullet(int x, int y);

void moveEnemy1Bullet();

void makeEnemy1BulletInActive(int z);

void enemy1CollisionWithP(); // enemy 1  left bullets collied with player

void enemy1health();

void printEnemy1Health();

void deleteEnemy1(); //function for deleting the enemy 1

// enemy 1 bullets for left side

int enemy1BulletX[1000];

int enemy1BulletY[1000];

bool isenemy1BulletActive[1000];

int enemy1BulletCount = 0;

// enemy 1 bullets2 for right side

int enemyBullet1X[1000];

int enemyBullet1Y[1000];

bool isenemyBullet1Active[1000];

int enemyBullet1Count = 0;

int shotEnemy1 = 0;

int enemy1Health = 5;

bool enemy1Active = true; // boolian variable for delete the enemy1

// enemy2 functionalities

//  enemy2 character

int enemy2X = 91;

int enemy2Y = 15;

char enemy2box3 = 248, enemy2box = 219;

char enemY[3][4] = {{' ', enemy2box, enemy2box, ' '},

                    {'|', enemy2box3, enemy2box3, '|'},

                    {' ', '|', '|', ' '}};

// enemy 2 bullets

int enemyBulletX[1000];

int enemyBulletY[1000];

bool isenemyBulletActive[1000];

int enemyBulletCount = 0;

int shotEnemy2 = 0;

void moveEnemy2();

void printEnemy2();

void eraseEnemy2();

void printEnemyBullet(int x, int y);

void eraseEnemyBullet(int x, int y);

void moveEnemyBullet();

void generateEnemyBullet();

void makeEnemyBulletInActive(int z);

void enemyCollisionWithP();

void enemy2health();

void printEnemy2Health();

void deleteEnemy2(); //function for deleting the enemy2

string enemy2Direction = "Up"; // variable for up movement

int enemy2Health = 5;

bool enemy2Active = true; // boolian variable for delete the enemy2

// enemy 3 bottom side functionalities

//  enemy 3 character

int enemy3X = 3;

int enemy3Y = 26;

char enemy3box3 = 248, enemy2box2 = 219;

char enemY3[3][4] = {{' ', enemy2box2, enemy2box2, ' '},

                     {'|', enemy3box3, enemy3box3, '|'},

                     {' ', '|', '|', ' '}};

// enemy 3 bullets

int enemy3BulletX[1000];

int enemy3BulletY[1000];

bool isenemy3BulletActive[1000];

int enemy3BulletCount = 0;

void enemy3CollisionWithP();

// enemy 3 bullets left side

int enemy3Bullet1X[1000];

int enemy3Bullet1Y[1000];

bool isenemy3Bullet1Active[1000];

int enemy3Bullet1Count = 0;

string enemy3Direction = "right"; // variable for right movement

int enemy3Health = 5;

bool enemy3Active = true; // boolian variable for delete the enemy3

void moveEnemy3();

void printEnemy3();

void eraseEnemy3();

void generateEnemy3Bullet(); // bullets up movement

void printEnemy3Bullet(int x, int y);

void eraseEnemy3Bullet(int x, int y);

void moveEnemy3Bullet();

void makeEnemy3BulletInActive(int z);

void generateEnemy3Bullet1(); // enemy 3 bullets left movement

void printEnemy3Bullet1(int x, int y);

void eraseEnemy3Bullet1(int x, int y);

void moveEnemy3Bullet1();

void makeEnemy3Bullet1InActive(int z);

void enemy3health();

void printEnemy3Health();

void deleteEnemy3(); //function for deleting the enemy 3

bool gameExit=true;

**Complete Code**

#include <iostream>

#include <conio.h>

#include <windows.h>

using namespace std;

void topHeader();

void mainmenu();

void instruction();

void start();

void submenu();

void key();

// supporting functions

char getCharAtxy(short int x, short int y);

void gotoxy(int x, int y);

void gameOver();

void addScore();

void printScore();

void printmaze();

void youWon();

void generateColors();//function for giving color to characters

// player 's functionalities start

void generateC();

void movetankleft();

void movetankright();

void movetankup();

void movetankdown();

void erase();

void generatebullet();

void movebullet();

void makebulletInActive(int z);

void eraseBullet(int x, int y);

void printBullet(int x, int y);

void collisionWithenemy3(); // player bullets collied with enemy3 at bottom

#include <iostream>

#include <windows.h>

using namespace std;

// supporting functions

char getCharAtxy(short int x, short int y);

void gotoxy(int x, int y);

void gameOver();

void addScore();

void printScore();

void printmaze();

void youWon();

int score = 0;

// player 's functionalities start

void generateC();

void movetankleft();

void movetankright();

void movetankup();

void movetankdown();

void erase();

void generatebullet();

void movebullet();

void makebulletInActive(int z);

void eraseBullet(int x, int y);

void printBullet(int x, int y);

void collisionWithenemy3(); // player bullets collied with enemy3 at bottom

void collisionWithenemy();  // player bullets collied with enemy2 at left side

void collisionWithenemy1(); // player bullets collied with enemy1 at top

// player Dtat Structure

int tankx = 3;

int tanky = 3;

char box2 = 248, box3 = 219, box4 = 234, ear1 = 62, ear2 = 60; // player character

char tank1[4][4] = {{' ', '\_', '\_', ' '},

                    {'(', box2, box2, ')'},

                    {'/', box3, box3, '\\'},

                    {'(', ' ', ' ', ')'}};

// player bullets

int bulletX[1000];

int bulletY[1000];

bool isBulletActive[1000];

int bulletCount = 0;

void printHealthOfPlayer();

int playerHealth = 20;

// player functionalities end

// enemy1 at top functionalities srart

//  enemy 1 character

string enemyDirection = "Left";

int enemyX = 89;

int enemyY = 3;

//Enemy1 Data Structure

char enemybox3 = 248, enemybox = 219;

char enemy[3][4] = {{' ', enemybox, enemybox, ' '},

                    {'|', enemybox3, enemybox3, '|'},

                    {' ', '|', '|', ' '}};

void printEnemy();

void eraseEnemy();

void moveEnemy();

void printEnemyBullet1(int x, int y); // enemy 1 bullets for right side

void generateEnemyBullet1();

void eraseEnemyBullet1(int x, int y);

void moveEnemyBullet1();

void makeEnemyBulletInActive1(int z);

void generateEnemy1Bullet();

void printEnemy1Bullet(int x, int y); // enemy 1 bullets for left side

void eraseEnemy1Bullet(int x, int y);

void moveEnemy1Bullet();

void makeEnemy1BulletInActive(int z);

void enemy1CollisionWithP(); // enemy 1  left bullets collied with player

void enemy1health();

void printEnemy1Health();

void deleteEnemy1(); //function for deleting the enemy 1

// enemy 1 bullets for left side

int enemy1BulletX[1000];

int enemy1BulletY[1000];

bool isenemy1BulletActive[1000];

int enemy1BulletCount = 0;

// enemy 1 bullets2 for right side

int enemyBullet1X[1000];

int enemyBullet1Y[1000];

bool isenemyBullet1Active[1000];

int enemyBullet1Count = 0;

int shotEnemy1 = 0;

int enemy1Health = 5;

bool enemy1Active = true; // boolian variable for delete the enemy1

// enemy2 functionalities

//  enemy2 character

int enemy2X = 91;

int enemy2Y = 15;

char enemy2box3 = 248, enemy2box = 219;

char enemY[3][4] = {{' ', enemy2box, enemy2box, ' '},

                    {'|', enemy2box3, enemy2box3, '|'},

                    {' ', '|', '|', ' '}};

// enemy 2 bullets

int enemyBulletX[1000];

int enemyBulletY[1000];

bool isenemyBulletActive[1000];

int enemyBulletCount = 0;

int shotEnemy2 = 0;

void moveEnemy2();

void printEnemy2();

void eraseEnemy2();

void printEnemyBullet(int x, int y);

void eraseEnemyBullet(int x, int y);

void moveEnemyBullet();

void generateEnemyBullet();

void makeEnemyBulletInActive(int z);

void enemyCollisionWithP();

void enemy2health();

void printEnemy2Health();

void deleteEnemy2(); //function for deleting the enemy2

string enemy2Direction = "Up"; // variable for up movement

int enemy2Health = 5;

bool enemy2Active = true; // boolian variable for delete the enemy2

// enemy 3 bottom side functionalities

//  enemy 3 character

int enemy3X = 3;

int enemy3Y = 26;

char enemy3box3 = 248, enemy2box2 = 219;

char enemY3[3][4] = {{' ', enemy2box2, enemy2box2, ' '},

                     {'|', enemy3box3, enemy3box3, '|'},

                     {' ', '|', '|', ' '}};

// enemy 3 bullets

int enemy3BulletX[1000];

int enemy3BulletY[1000];

bool isenemy3BulletActive[1000];

int enemy3BulletCount = 0;

void enemy3CollisionWithP();

// enemy 3 bullets left side

int enemy3Bullet1X[1000];

int enemy3Bullet1Y[1000];

bool isenemy3Bullet1Active[1000];

int enemy3Bullet1Count = 0;

string enemy3Direction = "right"; // variable for right movement

int enemy3Health = 5;

bool enemy3Active = true; // boolian variable for delete the enemy3

void moveEnemy3();

void printEnemy3();

void eraseEnemy3();

void generateEnemy3Bullet(); // bullets up movement

void printEnemy3Bullet(int x, int y);

void eraseEnemy3Bullet(int x, int y);

void moveEnemy3Bullet();

void makeEnemy3BulletInActive(int z);

void generateEnemy3Bullet1(); // enemy 3 bullets left movement

void printEnemy3Bullet1(int x, int y);

void eraseEnemy3Bullet1(int x, int y);

void moveEnemy3Bullet1();

void makeEnemy3Bullet1InActive(int z);

void enemy3health();

void printEnemy3Health();

void deleteEnemy3(); //function for deleting the enemy 3

bool gameExit=true;

main()

{

    system("cls");

    topHeader();

    Sleep(2000);

    system("cls");

    submenu();

    cout<<"                "<<endl;

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

    cout << "press any key to continue:" << endl;

    getch();

        system("cls");

    submenu();

    cout<<"                "<<endl;

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

    Sleep(1000);

    cout<<"Loading 30%"<<endl;

     Sleep(1000);

             system("cls");

    submenu();

    cout<<"                "<<endl;

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

     cout<<"Loading 90%"<<endl;

     Sleep(1000);

             system("cls");

    submenu();

    cout<<"                "<<endl;

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

     cout<<"Loading Complete:"<<endl;

     Sleep(1000);

    mainmenu();

    // start();

}

void topHeader()

{

cout<<"               ,,,,,,,,,,,         "<<endl;

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

cout<<"             // ==     == \\\\         "<<endl;

cout<<"              (  o    o  )\\\\          "<<endl;

cout<<"               (    L    )\\\\            "<<endl;

cout<<"                ( .\_\_\_  )\\\\               "<<endl;

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

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

cout<<"            ------/  ^  \\----                "<<endl;

cout<<"           /     ~~~o~~~~    \\                  "<<endl;

cout<<"          I         o         I                  "<<endl;

cout<<"         I    I     o     I    I                   "<<endl;

cout<<"         I   II     o     II   I                      "<<endl;

cout<<"         I   II     o     II   I                     "<<endl;

cout<<"         I   II     o     II   I                          "<<endl;

cout<<"         I   II     o     II   I                          "<<endl;

cout<<"         I   II     o     II  I                                  "<<endl;

cout<<"          I\_\_II%%%%%%@%%%%II\_\_I                          "<<endl;

cout<<"           'm'I           I 'm'                              "<<endl;

cout<<"              I           I                  "<<endl;

cout<<"              I           I             "<<endl;

cout<<"              I     i     I                "<<endl;

cout<<"              I     I     I                  "<<endl;

cout<<"              I     I     I                   "<<endl;

cout<<"              I     I     I                      "<<endl;

cout<<"               I\_\_\_\_I\_\_\_\_I                             "<<endl;

cout<<"             \_\_I    II   I\_\_                       "<<endl;

cout<<"            (\_\_\_\_\_\_\_II\_\_\_\_\_\_)                           "<<endl;

}

void mainmenu()

{

    system("cls");

    submenu();

    int option;

    cout << endl;

    cout << "1.instruction:" << endl;

    cout << "2.keys" << endl;

    cout << "3.start:" << endl;

    cout << "4.exit:" << endl;

    cout << "select one option:" << endl;

    cin >> option;

    if (option == 1)

    {

        system("cls");

        submenu();

        instruction();

    }

    else if (option == 2)

    {

        system("cls");

        submenu();

        key();

    }

     if (option != 4)

    {

        system("cls");

        start();

    }

  else if (option == 4)

    {

     exit;

    }

}

void submenu()

{

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

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

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

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

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

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

}

void instruction()

{

    cout << endl;

    cout << "instruction:" << endl;

    cout << "--------------" << endl;

    cout << endl;

    cout << "beware of robots." << endl;

    cout << "firing on  all the robots it will increase your score" << endl;

    cout << "when the bullet hit you. your score will bi decrease.if your score is less than zero than game over:" << endl;

    cout << "press any key to continue:";

    getch();

    system("cls");

    mainmenu();

}

void key()

{

    cout << "keys." << endl;

    cout << endl;

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

    cout << "1.for up movement of player            press up arrow key:" << endl;

    cout << "2. for DOWN movement of player        press down arrow key:" << endl;

    cout << "3. for Left movement of player         press left arrow key:" << endl;

    cout << "4.for Rigt movement  of player        press Right arrow key:" << endl;

    cout << "4.for firing  of player            press space :" << endl;

    cout << "4.to end the game            press ESC :" << endl;

    cout << "press any key to continue:" << endl;

    getch();

    mainmenu();

}

void start()

{

    system("cls");

    printmaze();

    generateC();

    printEnemy2();

    printEnemy3();

    Sleep(200);

    while (true)

    {

          printHealthOfPlayer();

        printScore();

        printEnemy1Health();

        printEnemy2Health();

        printEnemy3Health();

           generateColors();

if(enemy1Health==0&&enemy2Health==0&&enemy3Health==0)

{

    system("cls");

    youWon();

    getch();

    break;

}

       else if (playerHealth <= 0)

        {

            system("cls");

            gameOver();

            getch();

            break;

        }

        movebullet(); // player bullet move

        if (GetAsyncKeyState(VK\_ESCAPE))

        {

     mainmenu();

        }

        if (GetAsyncKeyState(VK\_LEFT))

        {

            movetankleft();

        }

        if (GetAsyncKeyState(VK\_RIGHT))

        {

            movetankright();

        }

        if (GetAsyncKeyState(VK\_UP))

        {

            movetankup();

        }

        if (GetAsyncKeyState(VK\_DOWN))

        {

            movetankdown();

        }

        if (GetAsyncKeyState(VK\_SPACE))

        {

            generatebullet();

        }

                deleteEnemy3();          //function for deleting the enemy 3

        if (enemy3Active == true)

        {

                moveEnemy3();

        if (enemy3X == tankx)

        {

            generateEnemy3Bullet(); // enemy3 bullets generate up side

        }

      else  if (enemy3Y == tankx)

        {

            generateEnemy3Bullet1(); // enemy3 bullets generate right side

        }

        }

        else if(enemy3Active==false)

        {

            eraseEnemy3();

        }

        deleteEnemy1();

        if (enemy1Active == true) //function for deleting the enemy1

        {

             printEnemy();

            moveEnemy();

            generateEnemy1Bullet(); // enemy1 bullets generate

            generateEnemyBullet1(); // enemy1 bullets generate right side

        }

        else if(enemy1Active == false)

        {

            eraseEnemy();

        }

        moveEnemyBullet();     // enemy bullets move

        deleteEnemy2();          //function for deleting the enemy 2

        if (enemy2Active == true)

        {

     moveEnemy2();

 generateEnemyBullet(); // enemy2 bullets generate

        }

        collisionWithenemy3();

        collisionWithenemy();

        enemyCollisionWithP();

        enemy1CollisionWithP();

        enemy3CollisionWithP();

        collisionWithenemy1();

        moveEnemy1Bullet();  // enemy1 bullets move

        moveEnemyBullet1();  // enemy1 bullets right side

        moveEnemy3Bullet();  // enemy3 bullets up move

        moveEnemy3Bullet1(); // enemy3 bullets right move

        Sleep(80);

    }

}

void gotoxy(int x, int y)

{

    COORD coordinates;

    coordinates.X = x;

    coordinates.Y = y;

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

}

char getCharAtxy(short int x, short int y) // player movements end

{

    CHAR\_INFO ci;

    COORD xy = {0, 0};

    SMALL\_RECT rect = {x, y, x, y};

    COORD coordBufSize;

    coordBufSize.X = 1;

    coordBufSize.Y = 1;

    return ReadConsoleOutput(GetStdHandle(STD\_OUTPUT\_HANDLE), &ci, coordBufSize, xy, &rect) ? ci.Char.AsciiChar : ' ';

}

void gameOver()

{

    gotoxy(15, 15);

    cout << "#####################" << endl;

    gotoxy(15, 16);

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

    gotoxy(15, 17);

    cout << "You lost: Your score: " << score << endl;

    gotoxy(15, 18);

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

    gotoxy(15, 19);

    cout << "   Game Over     " << endl;

    gotoxy(15, 20);

    cout << "#####################" << endl;

    getch();

}

void youWon()

{

     gotoxy(15, 15);

    cout << "#####################" << endl;

    gotoxy(15, 16);

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

    gotoxy(15, 17);

    cout << "You Won: Your score: " << score << endl;

    gotoxy(15, 18);

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

    gotoxy(15, 19);

    cout << "   Game Over     " << endl;

    gotoxy(15, 20);

    cout << "#####################" << endl;

}

void addScore()

{

    score++;

}

void printScore()

{

    gotoxy(95, 2);

    cout << "Score:"

         << score;

}

void printHealthOfPlayer()

{

    gotoxy(110, 10);

    cout << "Player Health: " << playerHealth;

}

// player's functions implementation

void generateC() // player print

{

    for (int x = 0; x < 4; x++)

    {

        gotoxy(tankx, tanky + x);

        for (int y = 0; y < 4; y++)

        {

            cout << tank1[x][y];

        }

    }

}

void printmaze()

{

    string maze[30][100];

    for (int x = 0; x < 30; x++)

    {

        for (int y = 0; y < 100; y++)

        {

            if (x == 0 || x == 29)

            {

                maze[x][y] = "#";

            }

            else if (x >= 29 && y >= 99 && y <= 103)

            {

                maze[x][y] = "#";

            }

            else if (y >= 99 && x >= 0 && x <= 10)

            {

                maze[x][y] = "   |%|";

            }

            else if (y >= 99 && x >= 10 && x <= 19)

            {

                maze[x][y] = "|%|";

            }

            else if (y == 0 && x >= 0 && x <= 29)

            {

                maze[x][y] = "|%|";

            }

            else if (y >= 99 && x >= 19 && x <= 29)

            {

                maze[x][y] = "   |%|";

            }

            else if (y == 18 && x > 10 && x < 20)

            {

                maze[x][y] = "|%|";

            }

            else if (x == 19 && y > 18 && y < 35)

            {

                maze[x][y] = "#";

            }

            else if (y == 78 && x > 10 && x < 20)

            {

                maze[x][y] = "||";

            }

            else if (x == 19 && y > 63 && y < 78)

            {

                maze[x][y] = "#";

            }

            else if (x == 8 && y > 37 && y < 57)

            {

                maze[x][y] = "$";

            }

            else if (x == 3 && y > 90 && y < 99)

            {

                maze[x][y] = "#";

            }

            else if (y == 90 && x > 0 && x < 4)

            {

                cout << "#";

            }

            else if (x == 26 && y > 90 && y < 99)

            {

                maze[x][y] = "#";

            }

            else if (y == 90 && x > 26 && x < 29)

            {

                cout << "#";

            }

            else

            {

                cout << " ";

            }

            cout << maze[x][y];

        }

        cout << endl;

    }

}

void movetankleft() // player movements start

{

    char next = getCharAtxy(tankx - 1, tanky);

    char next2 = getCharAtxy(tankx - 1, tanky + 1);

    char next3 = getCharAtxy(tankx - 1, tanky + 2);

    char next4 = getCharAtxy(tankx - 1, tanky + 3);

    if (next == ' ' && next2 == ' ' && next3 == ' ' && next4 == ' ')

    {

        erase();

        tankx--;

        generateC();

    }

}

void movetankright()

{

    char next = getCharAtxy(tankx + 4, tanky);

    char next2 = getCharAtxy(tankx + 4, tanky + 1);

    char next3 = getCharAtxy(tankx + 4, tanky + 2);

    char next4 = getCharAtxy(tankx + 4, tanky + 3);

    if (next == ' ' && next2 == ' ' && next3 == ' ' && next4 == ' ')

    {

        erase();

        tankx++;

        generateC();

    }

}

void movetankdown()

{

    char next = getCharAtxy(tankx, tanky + 4);

    char next2 = getCharAtxy(tankx + 1, tanky + 4);

    char next3 = getCharAtxy(tankx + 2, tanky + 4);

    char next4 = getCharAtxy(tankx + 3, tanky + 4);

    if (next == ' ' && next2 == ' ' && next3 == ' ' && next4 == ' ')

    {

        erase();

        tanky++;

        generateC();

    }

}

void movetankup()

{

    char next = getCharAtxy(tankx, tanky - 1);

    char next2 = getCharAtxy(tankx + 1, tanky - 1);

    char next3 = getCharAtxy(tankx + 2, tanky - 1);

    char next4 = getCharAtxy(tankx + 3, tanky - 1);

    if (next == ' ' && next2 == ' ' && next3 == ' ' && next4 == ' ')

    {

        erase();

        tanky--;

        generateC();

    }

}

void erase() // player remove

{

    for (int x = 0; x < 4; x++)

    {

        gotoxy(tankx, tanky + x);

        for (int y = 0; y < 4; y++)

        {

            cout << " ";

        }

    }

}

void generatebullet() // player bullets

{

    char next = getCharAtxy(tankx + 4, tanky + 1);

    if (next == ' ')

    {

        bulletX[bulletCount] = tankx + 4;

        bulletY[bulletCount] = tanky + 1;

        isBulletActive[bulletCount] = true;

        gotoxy(tankx + 4, tanky + 1);

        cout << ".";

        bulletCount++;

    }

}

void movebullet()

{

    for (int index = 0; index < bulletCount; index++)

    {

        if (isBulletActive[index] == true)

        {

            char next = getCharAtxy(bulletX[index] + 1, bulletY[index]);

            if (next != ' ' && next != '.')

            {

                eraseBullet(bulletX[index], bulletY[index]);

                makebulletInActive(index);

            }

            else

            {

                eraseBullet(bulletX[index], bulletY[index]);

                bulletX[index] = bulletX[index] + 1;

                printBullet(bulletX[index], bulletY[index]);

            }

        }

    }

}

void makebulletInActive(int z)

{

    isBulletActive[z] = false;

}

void eraseBullet(int x, int y)

{

    gotoxy(x, y);

    cout << " ";

}

void printBullet(int x, int y) // player  bullets ends

{

    gotoxy(x, y);

    cout << ".";

}

void collisionWithenemy() // player bullets collision with enemy2

{

    for (int x = 0; x < bulletCount; x++)

    {

        if (isBulletActive[x] == true)

        {

            if (bulletX[x] == enemy2X - 1 && (bulletY[x] == enemy2Y || bulletY[x] == enemy2Y + 1 || bulletY[x] == enemy2Y + 2 || bulletY[x] == enemy2Y + 3 || bulletY[x] == enemy2Y + 3))

            {

                addScore();

                enemy2health();

            }

        }

    }

}

void collisionWithenemy1() // player bullets collision with enemy1

{

    for (int x = 0; x < bulletCount; x++)

    {

        if (isBulletActive[x] == true)

        {

            if (bulletX[x] == enemyX - 1 && (bulletY[x] == enemyY || bulletY[x] == enemyY + 1 || bulletY[x] == enemyY + 2 || bulletY[x] == enemyY + 3))

            {

                addScore();

                enemy1health();

            }

        }

    }

}

void collisionWithenemy3() // player bullets collision with enemy3

{

    for (int x = 0; x < bulletCount; x++)

    {

        if (isBulletActive[x] == true)

        {

            if (bulletX[x] == enemy3X - 1 && (bulletY[x] == enemy3Y || bulletY[x] == enemy3Y + 1 || bulletY[x] == enemy3Y + 2 || bulletY[x] == enemy3Y + 3 || bulletY[x] == enemy3Y + 4))

            {

                addScore();

                enemy3health();

            }

        }

    }

}

void generateColors()

{

    HANDLE hConsole=GetStdHandle(STD\_OUTPUT\_HANDLE);

        SetConsoleTextAttribute(hConsole,2);

        printEnemy();

                SetConsoleTextAttribute(hConsole,12);

        printEnemy2();

                        SetConsoleTextAttribute(hConsole,6);

        printEnemy3();

                        SetConsoleTextAttribute(hConsole,5);

        generateC();

        // SetConsoleTextAttribute(hConsole,15);

}

// enemy 1 implementation of functionalities

void printEnemy()

{

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

    {

        gotoxy(enemyX, enemyY + index);

        for (int x = 0; x < 4; x++)

        {

            cout << enemy[index][x];

        }

    }

}

void eraseEnemy()

{

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

    {

        gotoxy(enemyX, enemyY + index);

        for (int x = 0; x < 4; x++)

        {

            cout << " ";

        }

    }

}

void moveEnemy()

{

    Sleep(5);

    if (enemyDirection == "Left")

    {

        char next1 = getCharAtxy(enemyX - 3, enemyY);

        char next2 = getCharAtxy(enemyX - 3, enemyY + 1);

        char next3 = getCharAtxy(enemyX - 3, enemyY + 2);

        char next4 = getCharAtxy(enemyX - 3, enemyY + 3);

        if (next1 == ' ' || next1 == '\*'&& next2 == ' ' || next2 == '\*' && next3 == ' ' || next3 == '\*' && next4 == ' ' || next4 == '\*')

        {

            eraseEnemy();

            enemyX--;

            printEnemy();

        }

        if (next1 != ' ' || next1 == '\*' && next2 != ' ' || next2 == '\*' && next3 != ' ' || next3 == '\*' && next4 != ' ' || next4 == '\*')

        {

            enemyDirection = "Right";

        }

    }

    if (enemyDirection == "Right")

    {

        char next1 = getCharAtxy(enemyX + 5, enemyY);

        char next2 = getCharAtxy(enemyX + 5, enemyY + 1);

        char next3 = getCharAtxy(enemyX + 5, enemyY + 2);

        if (next1 == ' ' || next1 == '\*' || next1 == '>' && next2 == ' ' || next2 == '>' || next2 == '\*' && next3 == ' ' || next3 == '\*' || next3 == '>')

        {

            eraseEnemy();

            enemyX++;

            printEnemy();

        }

        else if (next1 != ' ' || next1 != '#' && next2 != ' ' || next2 != '#' && next3 != ' ' || next3 != '#')

        {

            enemyDirection = "Left";

        }

    }

}

void generateEnemy1Bullet() // enemy1(top side) bullets  generate for left side

{

    if (shotEnemy2 % 10 == 0)

    {

        char next = getCharAtxy(enemyX - 1, enemyY + 1);

        if (next == ' ')

        {

            enemy1BulletX[enemy1BulletCount] = enemyX - 1;

            enemy1BulletY[enemy1BulletCount] = enemyY + 1;

            isenemy1BulletActive[enemy1BulletCount] = true;

            gotoxy(enemyX - 1, enemyY + 1);

            cout << "<";

            enemy1BulletCount++;

        }

    }

    shotEnemy2++;

}

void moveEnemy1Bullet() // enemy1 bullet movement

{

    for (int y = 0; y < enemy1BulletCount; y++)

    {

        if (isenemy1BulletActive[y] == true)

        {

            char next = getCharAtxy(enemy1BulletX[y] - 1, enemy1BulletY[y] + 1);

            if (enemyY - 1 == tanky)

            {

                char next = getCharAtxy(enemy1BulletX[y] - 1, enemy1BulletY[y]);

            }

            if (next != ' ' && next != '\*')

            {

                eraseEnemy1Bullet(enemy1BulletX[y], enemy1BulletY[y]);

                makeEnemy1BulletInActive(y);

            }

            else

            {

                eraseEnemy1Bullet(enemy1BulletX[y], enemy1BulletY[y]);

                enemy1BulletX[y] = enemy1BulletX[y] - 1;

                if (enemyY - 1 != tanky)

                {

                    enemy1BulletY[y] = enemy1BulletY[y] + 1;

                }

                printEnemy1Bullet(enemy1BulletX[y], enemy1BulletY[y]);

            }

        }

    }

}

void eraseEnemy1Bullet(int x, int y)

{

    gotoxy(x, y);

    cout << " ";

}

void printEnemy1Bullet(int x, int y)

{

    gotoxy(x, y);

    cout << "<";

}

void makeEnemy1BulletInActive(int z)

{

    isenemy1BulletActive[z] = false;

}

// enemy1(top) bullet(left side) movement end

void generateEnemyBullet1() // enemy1(top) bullet(right side) movement start

{

    if (shotEnemy2 % 14 == 0)

    {

        enemyBullet1X[enemyBullet1Count] = enemyX + 4;

        enemyBullet1Y[enemyBullet1Count] = enemyY;

        isenemyBullet1Active[enemyBullet1Count] = true;

        gotoxy(enemyX + 4, enemyY);

        cout << ">";

        enemyBullet1Count++;

    }

}

void moveEnemyBullet1() // enemy1 bullet(right side) movement

{

    for (int y = 0; y < enemyBullet1Count; y++)

    {

        if (isenemyBullet1Active[y] == true)

        {

            char next = getCharAtxy(enemyBullet1X[y] + 1, enemyBullet1Y[y]);

            if (next != ' ' && next != '^')

            {

                eraseEnemyBullet1(enemyBullet1X[y], enemyBullet1Y[y]);

                makeEnemyBulletInActive1(y);

            }

            else

            {

                eraseEnemyBullet1(enemyBullet1X[y], enemyBullet1Y[y]);

                enemyBullet1X[y] = enemyBullet1X[y] + 1;

                printEnemyBullet1(enemyBullet1X[y], enemyBullet1Y[y]);

            }

        }

    }

}

void eraseEnemyBullet1(int x, int y)

{

    gotoxy(x, y);

    cout << " ";

}

void printEnemyBullet1(int x, int y)

{

    gotoxy(x, y);

    cout << ">";

}

void makeEnemyBulletInActive1(int z)

{

    isenemyBullet1Active[z] = false;

}

void enemy1CollisionWithP()

{

    for (int index = 0; index < enemy1BulletCount; index++)

    {

        if (isenemy1BulletActive[index] == true)

        {

            if (enemy1BulletX[index] == tankx + 4 && (enemy1BulletY[index] == tanky || enemy1BulletY[index] == tanky + 1 || enemy1BulletY[index] == tanky + 2 || enemy1BulletY[index] == tanky + 3))

            {

                playerHealth--;

            }

            else if (tanky - 1 == enemy1BulletY[index] && (tankx + 3 == enemy1BulletX[index] || tankx == enemy1BulletX[index] || tankx + 1 == enemy1BulletX[index] || tankx + 2 == enemy1BulletX[index]))

            {

                playerHealth--;

            }

        }

        if (isenemyBullet1Active[index] == true)

        {

            if (enemyBullet1X[index] == tankx - 1 && (enemyBullet1Y[index] == tanky || enemyBullet1Y[index] == tanky + 1 || enemyBullet1Y[index] == tanky + 2 || enemyBullet1Y[index] == tanky + 3))

            {

                playerHealth--;

            }

        }

    }

}

void enemy1health()

{

    enemy1Health--;

}

void printEnemy1Health()

{

    gotoxy(110, 11);

    cout << "enemy 1 Health: " << enemy1Health;

}

void deleteEnemy1() // deleting enemy1

{

    if (enemy1Health <= 0)

    {

        enemy1Active = false;

        enemy1Health = 0;

        // eraseEnemy();

        // enemyX = 150;

        // enemyY = 10;

        // gotoxy(enemyX,enemyY);

    }

}

// enemy2 left side functionalities start

void printEnemy2()

{

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

    {

        gotoxy(enemy2X, enemy2Y + index);

        for (int x = 0; x < 4; x++)

        {

            cout << enemY[index][x];

        }

    }

}

void eraseEnemy2()

{

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

    {

        gotoxy(enemy2X, enemy2Y + index);

        for (int x = 0; x < 4; x++)

        {

            cout << " ";

        }

    }

}

void moveEnemy2()

{

    Sleep(5);

    if (enemy2Direction == "Up")

    {

        char next1 = getCharAtxy(enemy2X, enemy2Y - 1);

        char next2 = getCharAtxy(enemy2X + 1, enemy2Y - 1);

        char next3 = getCharAtxy(enemy2X + 2, enemy2Y - 1);

        char next4 = getCharAtxy(enemy2X + 3, enemy2Y - 1);

        if (next1 == ' ' && next2 == ' ' && next3 == ' ' && next4 == ' ')

        {

            eraseEnemy2();

            enemy2Y--;

            printEnemy2();

        }

        if (next1 != ' ' || next2 != ' ' || next3 != ' ' || next4 != ' ')

        {

            enemy2Direction = "Down";

        }

    }

    if (enemy2Direction == "Down")

    {

        char next1 = getCharAtxy(enemy2X, enemy2Y + 3);

        char next2 = getCharAtxy(enemy2X + 1, enemy2Y + 3);

        char next3 = getCharAtxy(enemy2X + 2, enemy2Y + 3);

        char next4 = getCharAtxy(enemy2X + 3, enemy2Y + 3);

        if (next1 == ' ' && next2 == ' ' && next3 == ' ' && next4 == ' ')

        {

            eraseEnemy2();

            enemy2Y++;

            printEnemy2();

        }

        if (next1 != ' ' || next2 != ' ' || next3 != ' ' || next4 != ' ')

        {

            enemy2Direction = "Up";

        }

    }

}

// enemy2 bullets generate

void generateEnemyBullet()

{

    if (shotEnemy1 % 8 == 0)

    {

        char next = getCharAtxy(enemy2X - 2, enemy2Y);

        if (next == ' ')

        {

            enemyBulletX[enemyBulletCount] = enemy2X - 2;

            enemyBulletY[enemyBulletCount] = enemy2Y;

            isenemyBulletActive[enemyBulletCount] = true;

            gotoxy(enemy2X - 2, enemy2Y);

            cout << "\*";

            enemyBulletCount++;

        }

    }

    shotEnemy1++;

}

void moveEnemyBullet() // enemy2 bullet movement start

{

    for (int y = 0; y < enemyBulletCount; y++)

    {

        if (isenemyBulletActive[y] == true)

        {

            char next = getCharAtxy(enemyBulletX[y] - 1, enemyBulletY[y]);

            if (next != ' ')

            {

                eraseEnemyBullet(enemyBulletX[y], enemyBulletY[y]);

                makeEnemyBulletInActive(y);

            }

            else

            {

                eraseEnemyBullet(enemyBulletX[y], enemyBulletY[y]);

                enemyBulletX[y] = enemyBulletX[y] - 1;

                printEnemyBullet(enemyBulletX[y], enemyBulletY[y]);

            }

        }

    }

}

void eraseEnemyBullet(int x, int y)

{

    gotoxy(x, y);

    cout << " ";

}

void printEnemyBullet(int x, int y)

{

    gotoxy(x, y);

    cout << "\*";

}

void makeEnemyBulletInActive(int z) // enemy2 bullet movement end

{

    isenemyBulletActive[z] = false;

}

void enemyCollisionWithP() // enemy bullets collsion with player

{

    for (int index = 0; index < enemyBulletCount; index++)

    {

        if (isenemyBulletActive[index] == true)

        {

            if (enemyBulletX[index] == tankx + 4 && (enemyBulletY[index] == tanky || enemyBulletY[index] == tanky + 1 || enemyBulletY[index] == tanky + 2 || enemyBulletY[index] == tanky + 3))

            {

                playerHealth--;

            }

        }

    }

}

void enemy2health()

{

    enemy2Health--;

}

void printEnemy2Health()

{

    gotoxy(110, 12);

    cout << "enemy 2 Health: " << enemy2Health;

}

void deleteEnemy2() // deleting enemy2

{

    if (enemy2Health <= 0)

    {

        enemy2Health = 0;

        eraseEnemy2();

        enemy2X = 151;

        enemy2Y = 11;

        enemy2Active = false; //[New]

    }

}

// enemy 2 functionalities end

void printEnemy3() // enemy3  at bottom

{

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

    {

        gotoxy(enemy3X, enemy3Y + index);

        for (int x = 0; x < 4; x++)

        {

            cout << enemY3[index][x];

        }

    }

}

void eraseEnemy3()

{

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

    {

        gotoxy(enemy3X, enemy3Y + index);

        for (int x = 0; x < 4; x++)

        {

            cout << " ";

        }

    }

}

void moveEnemy3() // enemy3  at bottom movements's function start

{

    Sleep(5);

    if (enemy3Direction == "right")

    {

        char next1 = getCharAtxy(enemy3X + 5, enemy3Y);

        char next2 = getCharAtxy(enemy3X + 5, enemy3Y + 1);

        char next3 = getCharAtxy(enemy3X + 5, enemy3Y + 2);

        if (next1 == ' ' || next1 == '\*' || next1 == '+' && next2 == ' ' || next2 == '\*' || next2 == '+' && next3 == ' ' || next3 == '\*' || next3 == '+')

        {

            eraseEnemy3();

            enemy3X++;

            printEnemy3();

        }

        else if (next1 != ' ' || next1 == '#' || next2 != ' ' || next2 == '#' || next3 != ' ' || next3 == '#')

        {

            enemy3Direction = "left";

        }

    }

    if (enemy3Direction == "left")

    {

        char next1 = getCharAtxy(enemy3X - 1, enemy3Y);

        char next2 = getCharAtxy(enemy3X - 1, enemy3Y + 1);

        char next3 = getCharAtxy(enemy3X - 1, enemy3Y + 2);

        char next4 = getCharAtxy(enemy3X - 1, enemy3Y + 3);

        if (next1 == ' ' || next1 == '\*' && next2 == ' ' || next2 == '\*' && next3 == ' ' || next3 == '\*' && next4 == ' ' || next4 == '\*')

        {

            eraseEnemy3();

            enemy3X--;

            printEnemy3();

        }

        if (next1 != ' ' || next1 == '\*' && next2 != ' ' || next2 == '\*' && next3 != ' ' || next3 == '\*' && next4 != ' ' || next4 == '\*')

        {

            enemy3Direction = "right";

        }

    }

}

// enemy3  at bottom movents's function snd

void generateEnemy3Bullet() // enemy3 bullet movement

{

    enemy3BulletX[enemy3BulletCount] = enemy3X;

    enemy3BulletY[enemy3BulletCount] = enemy3Y - 1;

    isenemy3BulletActive[enemy3BulletCount] = true;

    gotoxy(enemy3X, enemy3Y - 1);

    cout << "+";

    enemy3BulletCount++;

}

void moveEnemy3Bullet() // enemy3 bullet movement  up side

{

    for (int y = 0; y < enemy3BulletCount; y++)

    {

        if (isenemy3BulletActive[y] == true)

        {

            char next = getCharAtxy(enemy3BulletX[y], enemy3BulletY[y] - 1);

            if (next != ' ' && next != '+')

            {

                eraseEnemy3Bullet(enemy3BulletX[y], enemy3BulletY[y]);

                makeEnemy3BulletInActive(y);

            }

            else

            {

                eraseEnemy3Bullet(enemy3BulletX[y], enemy3BulletY[y]);

                enemy3BulletY[y] = enemy3BulletY[y] - 1;

                printEnemy3Bullet(enemy3BulletX[y], enemy3BulletY[y]);

            }

        }

    }

}

void eraseEnemy3Bullet(int x, int y)

{

    gotoxy(x, y);

    cout << " ";

}

void printEnemy3Bullet(int x, int y)

{

    gotoxy(x, y);

    cout << "+";

}

void makeEnemy3BulletInActive(int z)

{

    isenemy3BulletActive[z] = false;

}

// enemy3 bullet movement  up side end

void generateEnemy3Bullet1() // enemy3 bullet left

{

    enemy3Bullet1X[enemy3Bullet1Count] = enemy3X - 1;

    enemy3Bullet1Y[enemy3Bullet1Count] = enemy3Y;

    isenemy3Bullet1Active[enemy3Bullet1Count] = true;

    gotoxy(enemy3X - 1, enemy3Y);

    cout << "+";

    enemy3Bullet1Count++;

}

void moveEnemy3Bullet1() // enemy3 bullet left movement

{

    for (int y = 0; y < enemy3Bullet1Count; y++)

    {

        if (isenemy3Bullet1Active[y] == true)

        {

            char next = getCharAtxy(enemy3Bullet1X[y] - 1, enemy3Bullet1Y[y]);

            if (next != ' ')

            {

                eraseEnemy3Bullet(enemy3Bullet1X[y], enemy3Bullet1Y[y]);

                makeEnemy3Bullet1InActive(y);

            }

            else

            {

                eraseEnemy3Bullet1(enemy3Bullet1X[y], enemy3Bullet1Y[y]);

                enemy3Bullet1X[y] = enemy3Bullet1X[y] - 1;

                printEnemy3Bullet1(enemy3Bullet1X[y], enemy3Bullet1Y[y]);

            }

        }

    }

}

void eraseEnemy3Bullet1(int x, int y)

{

    gotoxy(x, y);

    cout << " ";

}

void printEnemy3Bullet1(int x, int y)

{

    gotoxy(x, y);

    cout << "+";

}

void makeEnemy3Bullet1InActive(int z)

{

    isenemy3Bullet1Active[z] = false;

}

void enemy3CollisionWithP()

{

    for (int index = 0; index < enemyBulletCount; index++)

    {

        if (isenemy3BulletActive[index] == true)

        {

            if (enemy3BulletX[index] == tankx + 4 && (enemy3BulletY[index] == tanky || enemy3BulletY[index] == tanky + 1 || enemy3BulletY[index] == tanky + 2 || enemy3BulletY[index] == tanky + 3))

            {

                playerHealth--;

            }

            else if (tanky + 4 == enemy3BulletY[index] && (tankx + 3 == enemy3BulletX[index] || tankx == enemy3BulletX[index] || tankx + 1 == enemy3BulletX[index] || tankx + 2 == enemy3BulletX[index]))

            {

                playerHealth--;

            }

        }

    }

}

void enemy3health()

{

    enemy3Health--;

}

void printEnemy3Health()

{

    gotoxy(110, 13);

    cout << "enemy 3 Health: " << enemy3Health;

}

void deleteEnemy3() // deleting enemy3

{

    if (enemy3Health <= 0)

    {

        enemy3Health = 0;

        eraseEnemy3();

        enemy3X = 152;

        enemy3Y = 11;

        enemy3Active = false; //[New]

    }

}

// enemy3 bullet left movement end;  // player bullets collied with enemy2 at left side