**SPACE INVADERS**



Session: 2021 – 2024

**Submitted by:**

Shahzaib Rafi 2021-CS-2

**Supervised by:**

Dr. Awais

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**TABLE OF CONTENTS**

* Short Description and Story Writing of your Game
* Game Characters Description
* Rules & Interactions
* Goal of the Game
* Screenshot of the Game
* Data Structures Used in the Game
* Functions Prototype
* Full Code

**Short Description and Story Writing**

Space Invaders is a Japanese shooting video game released in 1978 by Taito. It  is considered one of the most influential video games of all time. It was the inspiration for numerous video games and game designers across different genres. Space Invaders is a fixed shooter(shoot ‘em up) .

In this game, player’s homeland is invaded by aliens from outer space. He defends and counterattacks in a spaceship. If he survives till the end and manages to destroy the enemy’s main warship, his homeland will be saved. Otherwise, his homeland will be DOOMED!!

**Game Characters**

There are 3 types of characters in this game

* Spaceship.
* Aliens.
* UFO

**Spaceship :**

Spaceship is a warship that we control using the arrow keys.

**Aliens :**

Invaders, some of them which move linearly, shoots lasers at us.

**UFO:**

A strange ship that appears randomly and does massive damage.

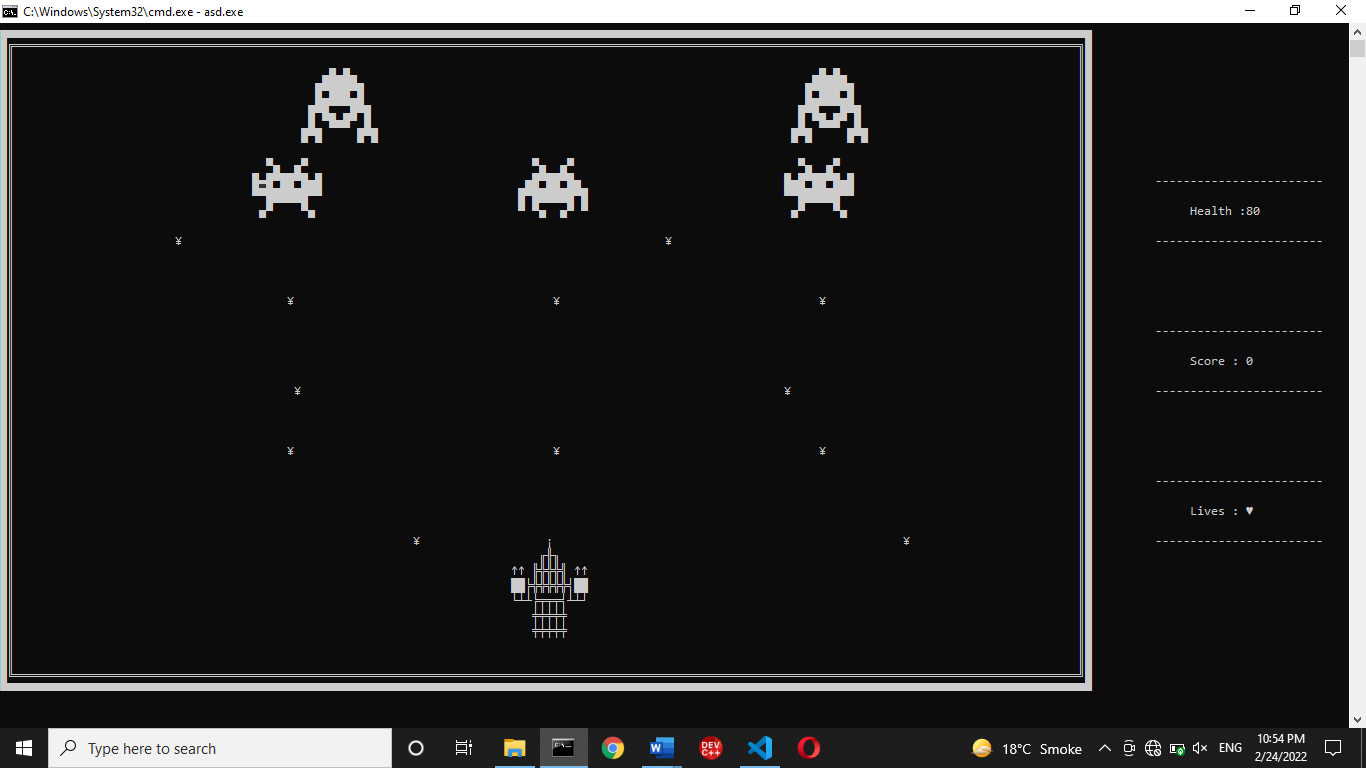
**Rules and Interactions**

* Spaceship shoots laser if hits the alien multiple times they die.
* If an alien laser hits spaceship, a life is loss.
* Successfully destroying an alien awards score.

**Goal**

Destroy all the invaders and don’t get destroyed yourself.

**Screenshot**



**Data Structures**

char map[45][156];

char spaceship[7][11];

char alien1[5][11];

char alien2[4][10];

char alien2A[4][10];

char UFO[4][16];

**Prototypes**

void loadData();

void storeData();

void printMap();

void gotoxy(int x, int y);

void Header();

int options();

void gameOver();

void DisplayHealth();

void DisplayScore();

void DisplayLives();

void DisplaySpaceship(int);

void spaceshipHealh();

void spaceshipMoveLeft();

void spaceshipMoveRight();

void spaceshipLaser();

void shootLaser(int n);

void moveLasers();

void alien1Horizontal1();

void alien2Horizontal1();

void Rip\_alien(int,int);

void alien1Lasers();

void alien2Lasers();

void aliensLasersMovement();

void UFOmovement();

void UFOlaser();

void UfoLaserMovement();

void eraseEnemy(int x, int y);

int RandomRow();

int RandomCol(int n);

**Source Code**

main()

{

    srand(time(0));

    Header();

    int option=options();

    loadData();

    system("cls");

    printMap();

    DisplaySpaceship(0);

    DisplayHealth();

    DisplayLives();

    DisplayScore();

    if(option==1)

    {

        while(!GetAsyncKeyState(VK\_ESCAPE) && lives>=0 && Running==true)

        {

            count+=10;

            speed++;

            n++;

            alien1Horizontal1();

            alien2Horizontal1();

            if(GetAsyncKeyState(VK\_LEFT))

            {

                spaceshipMoveLeft();

                count+=50;

                // speed=0;

            }

            else if(GetAsyncKeyState(VK\_RIGHT))

            {

                spaceshipMoveRight();

                count+=50;

                // speed=0;

            }

            else if(GetAsyncKeyState(VK\_SPACE) && speed>=50)

            {

                shootLaser(n);

                count+=50;

                speed=0;

            }

            else if(GetAsyncKeyState(VK\_END))

            {

                storeData();

                break;

            }

            if(n>50){

                alien1Lasers();

                alien2Lasers();

                n=0;

            }

            if(count>=100){

                count=0;

                moveLasers();

                aliensLasersMovement();

            }

            spaceshipHealh();

            Running = Alive\_1 || Alive\_2 || Alive\_3 || Alive\_4 || Alive\_5;

        }

    }

    else if(option==2)

    {

        while(!GetAsyncKeyState(VK\_ESCAPE) && lives>=0 && Running==true)

        {

            count+=10;

            speed++;

            n++;

            t++;

            if(t>100)

            {

                eraseEnemy(alienX1,alienY1);

                if(Alive\_3==true)

                {

                    eraseEnemy(alienX2,alienY2);

                }

            }

            UFOmovement();

            if(GetAsyncKeyState(VK\_LEFT))

            {

                spaceshipMoveLeft();

                count+=50;

                // speed=0;

            }

            else if(GetAsyncKeyState(VK\_RIGHT))

            {

                spaceshipMoveRight();

                count+=50;

                // speed=0;

            }

            else if(GetAsyncKeyState(VK\_SPACE) && speed>=50)

            {

                shootLaser(n);

                count+=50;

                speed=0;

            }

            if(n>50){

                UFOlaser();

                n=0;

            }

            if(count>=100){

                count=0;

                moveLasers();

                UfoLaserMovement();

            }

            spaceshipHealh();

            Running = Alive\_1 || Alive\_2 || Alive\_3;

        }

    }

    system("cls");

    gameOver();

}

void loadData()

{

    fstream file;

    int temp;

    if(ch==0)

    {

        file.open("D:\\Space invader\\Document\\map.txt",ios::in);

    }

    else

    {

        file.open("D:\\Space invader\\Document\\LastSaved.txt",ios::in);

        file >> sX;

        file >> sY;

        file >> alienX1;

        file >> alienY1;

        file >> alienX2;

        file >> alienY2;

        file >> hp\_1;

        file >> Alive\_1;

        file >> hp\_2;

        file >> Alive\_2;

        file >> hp\_3;

        file >> Alive\_3;

        file >> hp\_4;

        file >> Alive\_4;

        file >> hp\_5;

        file >> Alive\_5;

        file >> health;

        file >> Score;

        file >> lives;

    }

    for(int i=0;i<45;i++)

    {

        for(int j=0;j<156;j++)

        {

            file >> temp;

            map[i][j]=temp;

        }

    }

    file.close();

    file.open("D:\\Space invader\\Document\\Spaceship.txt",ios::in);

    for(int i=0;i<7;i++)

    {

        for(int j=0;j<11;j++)

        {

            file >> temp;

            spaceship[i][j]=temp;

        }

    }

    file.close();

    file.open("D:\\Space invader\\Document\\Alien1.txt",ios::in);

    for(int i=0;i<5;i++)

    {

        for(int j=0;j<11;j++)

        {

            file >> temp;

            alien1[i][j]=temp;

        }

    }

    file.close();

    file.open("D:\\Space invader\\Document\\Alien2.txt",ios::in);

    for(int i=0;i<4;i++)

    {

        for(int j=0;j<10;j++)

        {

            file >> temp;

            alien2[i][j]=temp;

        }

    }

    file.close();

    file.open("D:\\Space invader\\Document\\Alien2A.txt",ios::in);

    for(int i=0;i<4;i++)

    {

        for(int j=0;j<10;j++)

        {

            file >> temp;

            alien2A[i][j]=temp;

        }

    }

    file.close();

    file.open("D:\\Space invader\\Document\\UFO.txt",ios::in);

    for(int i=0;i<4;i++)

    {

        for(int j=0;j<16;j++)

        {

            file >> temp;

            UFO[i][j]=temp;

        }

    }

    file.close();

}

void storeData()

{

    fstream file;

    file.open("D:\\Space invader\\Document\\LastSaved.txt",ios::out);

    file << sX <<  ' ' << sY <<  ' ' << alienX1 <<  ' ' << alienY1 << ' ' << alienX2 <<  ' ' << alienY2 <<  ' ' << hp\_1 <<  ' ' << Alive\_1 <<  ' ' << hp\_2 <<  ' ' << Alive\_2 <<  ' ' << hp\_3 <<  ' ' << Alive\_3 <<  ' ' << hp\_4 <<  ' ' << Alive\_4 <<  ' ' << hp\_5 <<  ' ' << Alive\_5 << ' ' << health << ' ' << Score << ' ' << lives;

    for(int i=0;i<45;i++)

    {

        file << endl;

        for(int j=0;j<156;j++)

        {

            file << int(map[i][j]);

            if(j!=155){

                file << ' ';

            }

        }

    }

    file.close();

}

void Header()

{

    char d=15;

    system("cls");

    cout<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<endl;

    cout<<' '<<' '<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<endl;

    cout<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<" SPACE INVADER "<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<endl;

    cout<<' '<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<endl;

    cout<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<d<<endl;

}

int options()

{

    cout<<"\n\t1. Arena1.";

    cout<<"\n\t2. Arena2.";

    cout<<"\n\n\tSelect your option : ";

    char opt=getch();

    while((int(opt)-48)<1 || (int(opt)-48)>2){

        cout<<"Invalid! Input Again : ";

        opt=getch();

    }

    if(int(opt)-48==1){

        Header();

        cout<<"\n\tLoad Last Saved Game : ";

        char optt=getch();

        if(int(optt)-48==1){

            ch=1;

        }

    }

    return (int(opt)-48);

}

void printMap()

{

    for(int i=0;i<45;i++)

    {

        for(int j=0;j<156;j++)

        {

            cout<<map[i][j];

        }

        cout<<endl;

    }

}

void gotoxy(int x, int y)

{

COORD coordinates;

coordinates.X = x;

coordinates.Y = y;

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

}

void gameOver()

{

    if(Running==true)

    {

        gotoxy(70,20);

        cout<<"GAME OVER";

    }

    else

    {

        gotoxy(70,20);

        cout<<"YOU DID IT!";

    }

    getch();

    getch();

    getch();

    getch();

    getch();

    getch();

}

void DisplaySpaceship(int check)

{

    for(int i=0;i<7;i++)

    {

        for(int j=0;j<11;j++)

        {

            gotoxy(sY+j,sX+i);

            cout<<spaceship[i][j];

        }

        if(check==1){

            gotoxy(sY-1,sX+i);

        }

        cout<<' ';

        cout<<endl;

    }

}

void DisplayHealth()

{

    if(run==0)

    {

        gotoxy(165,10);

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

        gotoxy(170,12);

        cout<<"Health : ";

        gotoxy(165,14);

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

        gotoxy(165,20);

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

        gotoxy(170,22);

        cout<<"Score : ";

        gotoxy(165,24);

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

        gotoxy(165,30);

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

        gotoxy(170,32);

        cout<<"Lives : ";

        gotoxy(165,34);

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

        run=1;

    }

    gotoxy(178,12);

    cout<<health<<"  ";

}

void DisplayScore()

{

    gotoxy(178,22);

    cout<<Score;

}

void DisplayLives()

{

    gotoxy(178,32);

    if(lives==2){

        cout<<"\3 \3 \3";

    }

    else if(lives==1){

        cout<<"\3 \3   ";

    }

    else if(lives==0){

        cout<<"\3      ";

    }

}

void spaceshipMoveLeft()

{

    if(sY>3)

    {

        sY--;

        DisplaySpaceship(0);

    }

}

void spaceshipMoveRight()

{

    if(sY<142)

    {

        sY++;

        DisplaySpaceship(1);

    }

}

void spaceshipHealh()

{

    for(int i=0;i<7;i++)

    {

        for(int j=0;j<11;j++)

        {

            if(spaceship[i][j]!=' ' && map[sX+i][sY+j]==char(157))

            {

                health-=20;

                DisplayHealth();

                gotoxy(sY+j,sX+i);

                cout<<spaceship[i][j];

                map[sX+i][sY+j]=' ';

            }

        }

    }

    if(health<=0){

        lives--;

        if(lives>=0)

        {

            gotoxy(20,30);

            cout<<"Press any key to Continue : ";

            getch();

            getch();

            gotoxy(20,30);

            cout<<"                                      ";

        }

        health=100;

        DisplayHealth();

        DisplayLives();

    }

}

void alien1Horizontal1()

{

    if(Alive\_1 || Alive\_2)

    {

        alienY1+=a1way;

        for(int i=0;i<5;i++)

        {

            if(Alive\_1!=false)

            {

                gotoxy(alienY1-a1way,alienX1+i);

                if(a1way==-1){

                    gotoxy(alienY1+11,alienX1+i);

                }

                cout<<' ';

            }

        if(Alive\_2!=false)

            {

                gotoxy(alienY1-a1way+70,alienX1+i);

                if(a1way==-1){

                    gotoxy(alienY1+81,alienX1+i);

                }

                cout<<' ';

            }

            for(int j=0;j<11;j++)

            {

                if(Alive\_1!=false)

                {

                    gotoxy(alienY1+j,alienX1+i);

                    cout<<alien1[i][j];

                    if(alien1[i][j]!=' ' && map[alienX1+i][alienY1+j]==char(177)){

                        hp\_1-=20;

                        map[alienX1+i][alienY1+j]=' ';

                    }

                    if(hp\_1<=0){

                        Alive\_1=false;

                        Score+=200;

                        DisplayScore();

                        Rip\_alien(alienX1,alienY1);

                    }

                }

                if(Alive\_2!=false)

                {

                    gotoxy(alienY1+j+70,alienX1+i);

                    cout<<alien1[i][j];

                    if(alien1[i][j]!=' ' && map[alienX1+i][alienY1+j+70]==char(177)){

                        hp\_2-=20;

                        map[alienX1+i][alienY1+j+70]=' ';

                    }

                    if(hp\_2<=0){

                        Alive\_2=false;

                        Score+=200;

                        DisplayScore();

                        Rip\_alien(alienX1,alienY1+70);

                    }

                }

            }

        }

        gotoxy(0,1);

        if(alienY1<4 || alienY1>70)

        {

            a1way=a1way\*-1;

        }

    }

}

void alien2Horizontal1()

{

    if(Alive\_3 || Alive\_4 || Alive\_5)

    {

        for(int i=0;i<4;i++)

        {

            for(int j=0;j<10;j++)

            {

                if(Alive\_3!=false)

                {

                    gotoxy(alienY2+j,alienX2+i);

                    cout<<alien2[i][j];

                    if(alien2[i][j]!=' ' && map[alienX2+i][alienY2+j]==char(177)){

                        hp\_3-=34;

                        map[alienX2+i][alienY2+j]=' ';

                    }

                    if(hp\_3<=0){

                        Alive\_3=false;

                        Score+=100;

                        DisplayScore();

                        Rip\_alien(alienX2,alienY2);

                    }

                }

                if(Alive\_4!=false)

                {

                    gotoxy(alienY2+j+38,alienX2+i);

                    cout<<alien2A[i][j];

                    if(alien2[i][j]!=' ' && map[alienX2+i][alienY2+j+38]==char(177)){

                        hp\_4-=34;

                        map[alienX2+i][alienY2+j+38]=' ';

                    }

                    if(hp\_4<=0){

                        Alive\_4=false;

                        Score+=100;

                        DisplayScore();

                        Rip\_alien(alienX2,alienY2+38);

                    }

                }

                if(Alive\_5!=false)

                {

                    gotoxy(alienY2+j+76,alienX2+i);

                    cout<<alien2[i][j];

                    if(alien2[i][j]!=' ' && map[alienX2+i][alienY2+j+76]==char(177)){

                        hp\_5-=34;

                        map[alienX2+i][alienY2+j+76]=' ';

                    }

                    if(hp\_5<=0){

                        Alive\_5=false;

                        Score+=100;

                        DisplayScore();

                        Rip\_alien(alienX2,alienY2+76);

                    }

                }

            }

        }

    }

    gotoxy(0,1);

}

void Rip\_alien(int x,int y)

{

    for(int i=0;i<5;i++)

    {

        for(int j=-1;j<12;j++)

        {

            gotoxy(y+j,x+i);

            cout<<' ';

            map[x+i][y+j]=' ';

        }

    }

}

void spaceshipLaser()

{

    gotoxy(sY,sX);

    for(int i=0;sX-i>2;i++)

    {

        Sleep(100);

        gotoxy(sY,sX-i);

        cout<<char(177);

        gotoxy(sY,sX-i+1);

        cout<<' ';

    }

}

void shootLaser(int n)

{

    gotoxy(sY+5,sX-2);

    cout<<char(177);

    map[sX-2][sY+5]=char(177);

}

void moveLasers()

{

    char c=177,previousItem;

    int move;

    for(int i=5;i<37;i++)

    {

        for(int j=4;j<150;j++)

        {

            if(map[i][j]==c)

            {

                move=2;

                previousItem=map[i-move][j];

                map[i-move][j]=map[i][j];

                map[i][j]=previousItem;

                gotoxy(j,i);

                cout<<map[i][j];

                gotoxy(j,i-move);

                cout<<map[i-move][j];

            }

        }

    }

    for(int j=4;j<150;j++){

        if(map[4][j]==c){

            gotoxy(j,4);

            cout<<' ';

            map[4][j]=' ';

        }

    }

}

void alien1Lasers()

{

    if(hp\_1>0)

    {

        gotoxy(alienY1+5,alienX1+7);

        cout<<char(157);

        map[alienX1+7][alienY1+5]=char(157);

    }

    if(hp\_2>0)

    {

        gotoxy(alienY1+5+70,alienX1+7);

        cout<<char(157);

        map[alienX1+7][alienY1+5+70]=char(157);

    }

}

void alien2Lasers()

{

    if(hp\_3>0)

    {

        gotoxy(alienY2+5,alienX2+5);

        cout<<char(157);

        map[alienX2+5][alienY2+5]=char(157);

    }

    if(hp\_4>0)

    {

        gotoxy(alienY2+5+38,alienX2+5);

        cout<<char(157);

        map[alienX2+5][alienY2+5+38]=char(157);

    }

    if(hp\_5>0)

    {

        gotoxy(alienY2+5+76,alienX2+5);

        cout<<char(157);

        map[alienX2+5][alienY2+5+76]=char(157);

    }

}

void aliensLasersMovement()

{

    char d=157,previousItem;

    for(int i=37;i>5;i--)

    {

        for(int j=4;j<150;j++)

        {

            if(map[i][j]==d)

            {

                previousItem=map[i+2][j];

                map[i+2][j]=map[i][j];

                map[i][j]=previousItem;

                gotoxy(j,i);

                cout<<map[i][j];

                gotoxy(j,i+2);

                cout<<map[i+2][j];

            }

            // else if(i<34 && (map[i][j]==char(173) || map[i][j]==char(215) || map[i][j]==char(214) || map[i][j]==char(206) || map[i][j]==char(204)|| map[i][j]==char(185) || map[i][j]==char(219))){

            //     cout<<' ';

            //     map[i][j]=' ';

            // }

        }

    }

    for(int j=4;j<150;j++){

        if(map[38][j]==d)

        {

            gotoxy(j,38);

            cout<<' ';

            map[38][j]=' ';

        }

    }

}

void UFOmovement()

{

    if(Alive\_1==0)

    {

        if(t>100)

        {

            alienX1=RandomRow();

            alienY1=RandomCol(130);

            t=0;

        }

        if(t>50)

        {

            for(int i=0;i<4;i++)

            {

                for(int j=0;j<16;j++)

                {

                    if(UFO[i][j]!=' '&& map[alienX1+i][alienY1+j]==char(177))

                    hp\_1-=34;

                    gotoxy(alienY1+j,alienX1+i);

                    cout<<UFO[i][j];

                    map[alienX1+i][alienY1+j]=UFO[i][j];

                }

            }

        }

        if(hp\_1<=0)

        {

            Alive\_1=false;

            Score+=300;

            DisplayScore();

        }

    }

    else

    {

        if(Alive\_2==true)

        {

            if(t>100)

            {

                alienX1=RandomRow();

                alienY1=RandomCol(56);

            }

            if(t>50)

            {

                for(int i=0;i<4;i++)

                {

                    for(int j=0;j<16;j++)

                    {

                        if(UFO[i][j]!=' '&& map[alienX1+i][alienY1+j]==char(177))

                        hp\_1-=34;

                        gotoxy(alienY1+j,alienX1+i);

                        cout<<UFO[i][j];

                        map[alienX1+i][alienY1+j]=UFO[i][j];

                    }

                }

            }

            if(hp\_2<=0)

            {

                Alive\_2=false;

                Score+=300;

                DisplayScore();

            }

        }

        if(Alive\_3==true)

        {

            if(t>100)

            {

                alienX2=RandomRow();

                alienY2=75 + RandomCol(56);

            }

            if(t>50)

            {

                for(int i=0;i<4;i++)

                {

                    for(int j=0;j<16;j++)

                    {

                        if(UFO[i][j]!=' '&& map[alienX2+i][alienY2+j]==char(177))

                        hp\_3-=34;

                        gotoxy(alienY2+j,alienX2+i);

                        cout<<UFO[i][j];

                        map[alienX2+i][alienY2+j]=UFO[i][j];

                    }

                }

            }

            if(hp\_3<=0)

            {

                Alive\_3=false;

                Score+=300;

                DisplayScore();

            }

        }

        if(t>100){t=0;}

    }

}

void UFOlaser()

{

    if(Alive\_1==true)

    {

        gotoxy(alienY1+4,alienX1+5);

        cout<<char(157);

        gotoxy(alienY1+12,alienX1+5);

        cout<<char(157);

        map[alienX1+5][alienY1+4]=char(157);

        map[alienX1+5][alienY1+12]=char(157);

    }

    else

    {

        if(Alive\_2==true)

        {

            gotoxy(alienY1+4,alienX1+5);

            cout<<char(157);

            gotoxy(alienY1+12,alienX1+5);

            cout<<char(157);

            map[alienX1+5][alienY1+4]=char(157);

            map[alienX1+5][alienY1+12]=char(157);

        }

        if(Alive\_3==true)

        {

            gotoxy(alienY2+4,alienX2+5);

            cout<<char(157);

            gotoxy(alienY2+12,alienX2+5);

            cout<<char(157);

            map[alienX2+5][alienY2+4]=char(157);

            map[alienX2+5][alienY2+12]=char(157);

        }

    }

}

void UfoLaserMovement()

{

    char d=157,previousItem;

    for(int i=37;i>5;i--)

    {

        for(int j=4;j<150;j++)

        {

            if(map[i][j]==d)

            {

                previousItem=map[i+2][j];

                map[i+2][j]=map[i][j];

                map[i][j]=previousItem;

                gotoxy(j,i);

                cout<<map[i][j];

                gotoxy(j,i+2);

                cout<<map[i+2][j];

            }

        }

    }

    for(int j=4;j<150;j++){

        if(map[37][j]==d)

        {

            gotoxy(j,37);

            cout<<' ';

            map[37][j]=' ';

        }

        else if(map[38][j]==d)

        {

            gotoxy(j,38);

            cout<<' ';

            map[38][j]=' ';

        }

    }

}

void eraseEnemy(int x, int y)

{

    for(int i=0;i<4;i++)

    {

        for(int j=0;j<16;j++)

        {

            gotoxy(y+j,x+i);

            cout<<' ';

            map[x+i][y+j]=' ';

        }

    }

}

int RandomRow()

{

    // srand(time(0));

    int result = 3 + (rand() % 20);

    return result;

}

int RandomCol(int n)

{

    // srand(time(0));

    int result = 3 + (rand() % n);

    return result;

}

**Student Reg. No. :**   **Student Name.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Short Description and Story Writing of Game** - **Game Characters** Description - **Rules** & Interactions - **Goal** of the Game **- Screenshot** of the Game - **Data Structures** Used in the Game - **Functions** Prototype - **Full Code** | | | | |
| Project Complexity  **Grade:** | Project has at least 1 Player and 3 enemies. Proper use of gotoxy() function. Health system, Firing System and lives decreasing system. In case of board game (Pong, Ludo and 2048 is acceptable) | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Randomness  **Grade:** | Objects are produced randomly in the game. | meet more than 80% of the criteria given. | meet more than 50% of the criteria given. | Objects are appearing in the same pattern |
| Code Style  **Grade:** | All Code style criteria is followed | All code style criteria followed but some improvements required | lot of improvements required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation is synchronized. | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. |
| Idea Novelty and Creativity  **Grade:** | Idea is unique of the game | Idea is merged by combining other different games | Same idea as a previous game | Could not implement the existing game idea. |
| Data Structure (2D Arrays)  **Grade:** | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| File Handling  **Grade:** | Game maze is loaded and the updated maze is stored in the file | Game maze is loaded and partial data is stored in the file. | Game maze is just loaded but the updated game configuration is not stored in the maze. | Project do not contain file handling |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)- There is no global variable defined. Arrays and variables are passed as parameters to the functions. Functions exhibit single responsibility principle. | | | | |
| Screen flickering  **Grade:** | There is no Screen flickering. | Maze is not flickering but the characters are flickering at great speed | Flickering is done at lot of places | Screen is flickering at all places |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | Student has good understand but some place he does not know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. |

|  |  |
| --- | --- |
| **Checked by:** | Click or tap here to enter text. |

**Student Reg. No. :**   **Student Name.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Short Description and Story Writing of Game** - **Game Characters** Description - **Rules** & Interactions - **Goal** of the Game **- Screenshot** of the Game - **Data Structures** Used in the Game - **Functions** Prototype - **Full Code** | | | | |
| Project Complexity  **Grade:** | Project has at least 1 Player and 3 enemies. Proper use of gotoxy() function. Health system, Firing System and lives decreasing system. In case of board game (Pong, Ludo and 2048 is acceptable) | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Randomness  **Grade:** | Objects are produced randomly in the game. | meet more than 80% of the criteria given. | meet more than 50% of the criteria given. | Objects are appearing in the same pattern |
| Code Style  **Grade:** | All Code style criteria is followed | All code style criteria followed but some improvements required | lot of improvements required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation is synchronized. | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. |
| Idea Novelty and Creativity  **Grade:** | Idea is unique of the game | Idea is merged by combining other different games | Same idea as a previous game | Could not implement the existing game idea. |
| Data Structure (2D Arrays)  **Grade:** | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| File Handling  **Grade:** | Game maze is loaded and the updated maze is stored in the file | Game maze is loaded and partial data is stored in the file. | Game maze is just loaded but the updated game configuration is not stored in the maze. | Project do not contain file handling |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types)- There is no global variable defined. Arrays and variables are passed as parameters to the functions. Functions exhibit single responsibility principle. | | | | |
| Screen flickering  **Grade:** | There is no Screen flickering. | Maze is not flickering but the characters are flickering at great speed | Flickering is done at lot of places | Screen is flickering at all places |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | Student has good understand but some place he does not know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. |

|  |  |
| --- | --- |
| **Checked by:** | Click or tap here to enter text. |