**Space Shooter**



Session: 2021 – 2024

**Submitted by:**

Bisma Muhammad Ali 2021-CS-170

**Supervised by:**

Mam Maida Shahid

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Table Of Contents**

1. Description and Story Writing…………………………………….…Pg1
2. Game Characters……………………………….…………………….Pg1
3. Rules & Interactions.…………………………………………………Pg1
4. Goal…………………………….…………………………………….Pg1
5. Screenshots…………………………………………………………..Pg2-4
6. Data Structures………………………………………………………Pg5
7. Function Prototypes…………………………………………………Pg6
8. Full Code………………………………………………………….Pg7-30

**Description:**

This game is like the original space shooter game. In which a ship has to destroy all hurdle ships with its fire.

**Game Characters Description:**

There is a ship that will be controlled by the user. And there are hurdle ships.

**Rules & Interactions:**

The user will have to fire from its ship by moving left, right, up and down and also avoid fire shoot by enemy ships. The user only has 3 lives. If the enemy fire hits the ship or if our ship touch with enemy ship its life will be decreased by one, when all lives are gone the game will be over.

* Press left arrow key to move towards left
* Press right arrow key to move towards the right
* Press down arrow key to move towards the down
* Press up arrow key to move towards the up
* Press space bar to shoot fire

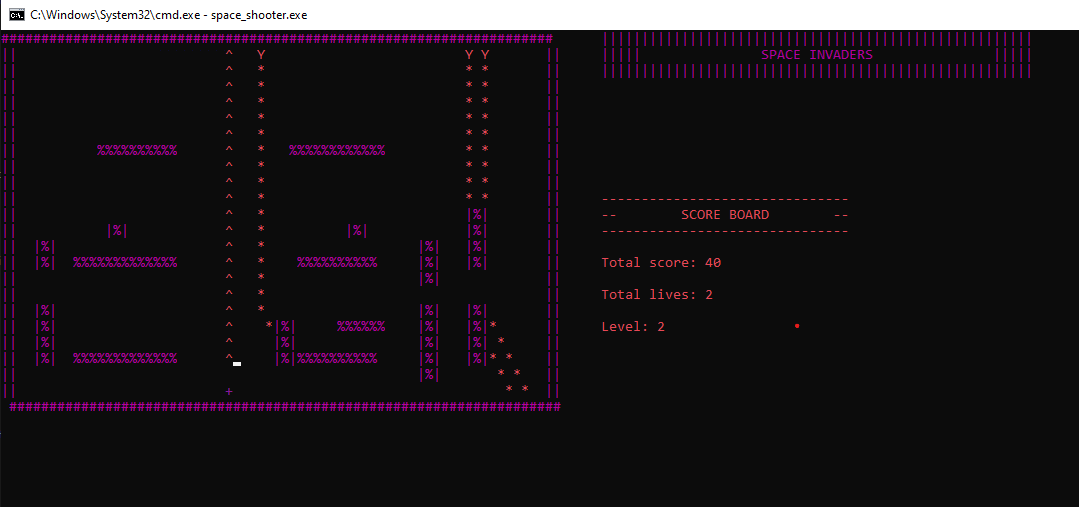
**Goal:**

Users have to destroy all of the enemy ships by shooting fire and avoid the fires coming from enemy ships.

**Screen Shots:**

Level 1



Level 2

Level 3



**Data Structure:**

char space [24][71];

int spaceX = 22;

int spaceY = 31;

char p\_item = ' ';

char h\_item = ' ';

char h2\_item = ' ';

char h3\_item = ' ';

char h4\_item = ' ';

char f\_item = ' ';

int hurdleX1 = 19;

int hurdleY1 = 25;

int hurdleX2 = 10;

int hurdleX3 = 14;

int hurdleY2 = 60;

int hurdleY3 = 22;

int hurdleX4 = 16;

int hurdleY4 = 50;

int levelup = 0;

int level = 1;

int lives = 3;

bool gameRunning1 = true;

bool gameRunning2 = true;

bool gameRunning3 = true;

bool gameRunning4 = true;

bool H1Live = true;

bool H2Live = true;

bool H3Live = true;

bool H4Live = true;

int shootCount1 = 0;

int shootCount2 = 0;

int shootCount3 = 0;

int shootCount4 = 0;

int score = 0;

int fireStart = 0;

**Function Prototypes:**

void display\_Space();

void movespaceshipLeft();

void movespaceshipRight();

void movespaceshipUp();

void movespaceshipDown();

void gotoxy(int x, int y);

int hurdel\_direction();

bool hurdle\_movement();

bool hurdle2\_movement();

bool hurdle3\_movement();

bool hurdle4\_movement();

void split\_fire();

void MoveFire();

void health();

void total\_lives();

void calculate\_score();

void print\_result();

void header();

void split\_hurdle\_fire1();

void split\_hurdle\_fire2();

void split\_hurdle\_fire3();

void split\_hurdle\_fire4();

void hurdle\_fire();

void gameOpener();

void storeSpaceIntoFile();

void loadSpaceIntoArray();

**Code:**

#include <iostream>

#include <fstream>

#include <conio.h>

#include <time.h>

#include <windows.h>

using namespace std;

char space[24][71];

////// global variables////////

int spaceX = 22;

int spaceY = 31;

char p\_item = ' ';

char h\_item = ' ';

char h2\_item = ' ';

char h3\_item = ' ';

char h4\_item = ' ';

char f\_item = ' ';

int hurdleX1 = 19;

int hurdleY1 = 25;

int hurdleX2 = 10;

int hurdleX3 = 14;

int hurdleY2 = 60;

int hurdleY3 = 22;

int hurdleX4 = 16;

int hurdleY4 = 50;

int levelup = 0;

int level = 1;

int lives = 3;

bool gameRunning1 = true;

bool gameRunning2 = true;

bool gameRunning3 = true;

bool gameRunning4 = true;

bool H1Live = true;

bool H2Live = true;

bool H3Live = true;

bool H4Live = true;

int shootCount1 = 0;

int shootCount2 = 0;

int shootCount3 = 0;

int shootCount4 = 0;

int score = 0;

int fireStart = 0;

/////   /       prototypes/////////

void display\_Space();

void movespaceshipLeft();

void movespaceshipRight();

void movespaceshipUp();

void movespaceshipDown();

void gotoxy(int x, int y);

int hurdel\_direction();

bool hurdle\_movement();

bool hurdle2\_movement();

bool hurdle3\_movement();

bool hurdle4\_movement();

void split\_fire();

void MoveFire();

void health();

void total\_lives();

void calculate\_score();

void print\_result();

void header();

void split\_hurdle\_fire1();

void split\_hurdle\_fire2();

void split\_hurdle\_fire3();

void split\_hurdle\_fire4();

void hurdle\_fire();

void gameOpener();

void storeSpaceIntoFile();

void loadSpaceIntoArray();

main()

{

    loadSpaceIntoArray();

    HANDLE console\_color;

    console\_color = GetStdHandle(STD\_OUTPUT\_HANDLE);

    gameOpener();

    if (levelup == 0)

    {

        system("CLS");

        SetConsoleTextAttribute(console\_color, 5);

        display\_Space();

        header();

        SetConsoleTextAttribute(console\_color, 10);

        gotoxy(spaceY, spaceX);

        cout << "+";

        while (gameRunning1 && gameRunning2)

        {

            Sleep(50);

            total\_lives();

            print\_result();

            if (H1Live)

            {

                gameRunning1 = hurdle\_movement();

            }

            if (H2Live)

            {

                gameRunning2 = hurdle2\_movement();

            }

            hurdle\_fire();

            if (fireStart >= 30)

            {

                if (H1Live)

                {

                    split\_hurdle\_fire1();

                }

                if (H2Live)

                {

                    split\_hurdle\_fire2();

                }

            }

            if (fireStart == 60)

            {

                fireStart = 0;

            }

            if (GetAsyncKeyState(VK\_LEFT))

            {

                movespaceshipLeft();

            }

            if (GetAsyncKeyState(VK\_RIGHT))

            {

                movespaceshipRight();

            }

            if (GetAsyncKeyState(VK\_UP))

            {

                movespaceshipUp();

            }

            if (GetAsyncKeyState(VK\_DOWN))

            {

                movespaceshipDown();

            }

            if (GetAsyncKeyState(VK\_SPACE))

            {

                split\_fire();

            }

            if (GetAsyncKeyState(VK\_ESCAPE))

            {

                gameRunning1 = false; // stop the game

            }

            MoveFire();

            health();

            fireStart++;

            if (lives <= 0)

            {

                break;

            }

            if (levelup == 2)

            {

                break;

            }

        }

    }

    H1Live = true;

    H2Live = true;

    H3Live = true;

    H4Live = true;

    if (levelup == 2)

    {

        system("CLS");

        level = 2;

        SetConsoleTextAttribute(console\_color,13);

        display\_Space();

        header();

        gotoxy(spaceY, spaceX);

        cout << "+";

        while (gameRunning1 && gameRunning2 && gameRunning3)

        {

            Sleep(50);

            total\_lives();

            print\_result();

            if (H1Live)

            {

                gameRunning1 = hurdle\_movement();

            }

            if (H2Live)

            {

                gameRunning2 = hurdle2\_movement();

            }

            if (H3Live)

            {

                gameRunning3 = hurdle3\_movement();

            }

            hurdle\_fire();

            if (fireStart >= 30)

            {

                if (H1Live)

                {

                    split\_hurdle\_fire1();

                }

                if (H2Live)

                {

                    split\_hurdle\_fire2();

                }

                if (H3Live)

                {

                    split\_hurdle\_fire3();

                }

            }

            if (fireStart == 60)

            {

                fireStart = 0;

            }

            if (GetAsyncKeyState(VK\_LEFT))

            {

                movespaceshipLeft();

            }

            if (GetAsyncKeyState(VK\_RIGHT))

            {

                movespaceshipRight();

            }

            if (GetAsyncKeyState(VK\_UP))

            {

                movespaceshipUp();

            }

            if (GetAsyncKeyState(VK\_DOWN))

            {

                movespaceshipDown();

            }

            if (GetAsyncKeyState(VK\_SPACE))

            {

                split\_fire();

            }

            if (GetAsyncKeyState(VK\_ESCAPE))

            {

                gameRunning1 = false; // stop the game

            }

            MoveFire();

            health();

            fireStart++;

            if (lives <= 0)

            {

                break;

            }

            if (levelup == 5)

            {

                break;

            }

        }

    }

    H1Live = true;

    H2Live = true;

    H3Live = true;

    H4Live = true;

    if (levelup == 5)

    {

        system("CLS");

        level = 3;

      SetConsoleTextAttribute(console\_color,11);

        display\_Space();

        header();

        gotoxy(spaceY, spaceX);

        cout << "+";

        while (gameRunning1 && gameRunning2 && gameRunning3 && gameRunning4)

        {

            Sleep(50);

            total\_lives();

            print\_result();

            if (H1Live)

            {

                gameRunning1 = hurdle\_movement();

            }

            if (H2Live)

            {

                gameRunning2 = hurdle2\_movement();

            }

            if (H3Live)

            {

                gameRunning3 = hurdle3\_movement();

            }

            if (H4Live)

            {

                gameRunning4 = hurdle4\_movement();

            }

            hurdle\_fire();

            if (fireStart >= 30)

            {

                if (H1Live)

                {

                    split\_hurdle\_fire1();

                }

                if (H2Live)

                {

                    split\_hurdle\_fire2();

                }

                if (H3Live)

                {

                    split\_hurdle\_fire3();

                }

                if (H4Live)

                {

                    split\_hurdle\_fire4();

                }

            }

            if (fireStart == 60)

            {

                fireStart = 0;

            }

            if (GetAsyncKeyState(VK\_LEFT))

            {

                movespaceshipLeft();

            }

            if (GetAsyncKeyState(VK\_RIGHT))

            {

                movespaceshipRight();

            }

            if (GetAsyncKeyState(VK\_UP))

            {

                movespaceshipUp();

            }

            if (GetAsyncKeyState(VK\_DOWN))

            {

                movespaceshipDown();

            }

            if (GetAsyncKeyState(VK\_SPACE))

            {

                split\_fire();

            }

            if (GetAsyncKeyState(VK\_ESCAPE))

            {

                gameRunning1 = false; // stop the game

            }

            MoveFire();

            health();

            fireStart++;

            if (lives <= 0)

            {

                break;

            }

            if (levelup == 9)

            {

                break;

            }

        }

    }

    if (levelup == 9)

    {

        system("CLS");

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

        cout << "//    Hurrayyyy! You won the game    //" << endl;

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

    }

 //  storeSpaceIntoFile();

}

void display\_Space()

{

    for (int r = 0; r < 24; r++)

    {

        for (int c = 0; c < 71; c++)

        {

            cout << space[r][c];

        }

        cout << endl;

    }

}

void movespaceshipLeft()

{

    if ((space[spaceX][spaceY - 1] == ' ') || (space[spaceX][spaceY - 1] == 'o') || (space[spaceX][spaceY - 1] == '\*') || (space[spaceX][spaceY - 1] == 'Y'))

    {

        space[spaceX][spaceY] = p\_item;

        gotoxy(spaceY, spaceX);

        cout << " ";

        spaceY = spaceY - 1;

        gotoxy(spaceY, spaceX);

        cout << "+";

        if (space[spaceX][spaceY] == 'o')

        {

            lives = lives + 1;

        }

    }

}

void movespaceshipRight()

{

    if ((space[spaceX][spaceY + 1] == ' ') || (space[spaceX][spaceY + 1] == 'o') || (space[spaceX][spaceY + 1] == '\*') || (space[spaceX][spaceY + 1] == 'Y'))

    {

        space[spaceX][spaceY] = p\_item;

        gotoxy(spaceY, spaceX);

        cout << " ";

        spaceY = spaceY + 1;

        gotoxy(spaceY, spaceX);

        cout << "+";

        if (space[spaceX][spaceY] == 'o')

        {

            lives = lives + 1;

        }

    }

}

void movespaceshipUp()

{

    if ((space[spaceX - 1][spaceY] == ' ') || (space[spaceX - 1][spaceY] == 'o') || (space[spaceX - 1][spaceY] == '\*') || (space[spaceX - 1][spaceY] == 'Y'))

    {

        space[spaceX][spaceY] = p\_item;

        gotoxy(spaceY, spaceX);

        cout << " ";

        spaceX = spaceX - 1;

        gotoxy(spaceY, spaceX);

        cout << "+";

        if (space[spaceX][spaceY] == 'o')

        {

            lives = lives + 1;

        }

    }

}

void movespaceshipDown()

{

    if ((space[spaceX + 1][spaceY] == ' ') || (space[spaceX + 1][spaceY] == 'o') || (space[spaceX + 1][spaceY] == '\*') || (space[spaceX + 1][spaceY] == 'Y'))

    {

        space[spaceX][spaceY] = p\_item;

        gotoxy(spaceY, spaceX);

        cout << " ";

        spaceX = spaceX + 1;

        gotoxy(spaceY, spaceX);

        cout << "+";

        if (space[spaceX][spaceY] == 'o')

        {

            lives = lives + 1;

        }

    }

}

void gotoxy(int x, int y)

{

    COORD coordinates;

    coordinates.X = x;

    coordinates.Y = y;

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

}

int hurdle\_direction()

{

    srand(time(0));

    int result = 1 + (rand() % 4);

    return result;

}

bool hurdle\_movement()

{

    HANDLE console\_color;

    console\_color = GetStdHandle(STD\_OUTPUT\_HANDLE);

    int move = hurdle\_direction();

    if (move == 1)

    {

        if ((space[hurdleX1][hurdleY1 - 1] == ' ') || (space[hurdleX1][hurdleY1 - 1] == '+'))

        {

            space[hurdleX1][hurdleY1] = h\_item;

            gotoxy(hurdleY1, hurdleX1);

            cout << h\_item;

            gotoxy(hurdleY1, hurdleX1);

            cout << " ";

            hurdleY1 = hurdleY1 - 1;

            h\_item = space[hurdleX1][hurdleY1];

            if (space[hurdleX1][hurdleY1] == 'Y')

            {

                hurdle\_movement();

            }

            else

            {

                gotoxy(hurdleY1, hurdleX1);

                SetConsoleTextAttribute(console\_color, 12);

                cout << "Y";

                space[hurdleX1][hurdleY1] = 'Y';

            }

        }

    } // move left

    if (move == 2)

    {

        if ((space[hurdleX1][hurdleY1 + 1] == ' ') || (space[hurdleX1][hurdleY1 + 1] == '+'))

        {

            space[hurdleX1][hurdleY1] = h\_item;

            gotoxy(hurdleY1, hurdleX1);

            cout << h\_item;

            gotoxy(hurdleY1, hurdleX1);

            cout << " ";

            hurdleY1 = hurdleY1 + 1;

            h\_item = space[hurdleX1][hurdleY1];

            if (space[hurdleX1][hurdleY1] == 'Y')

            {

                hurdle\_movement();

            }

            else

            {

                SetConsoleTextAttribute(console\_color, 5);

                gotoxy(hurdleY1, hurdleX1);

                cout << "Y";

                space[hurdleX1][hurdleY1] = 'Y';

            }

        } // move right

    }

    if (move == 3)

    {

        if ((space[hurdleX1 - 1][hurdleY1] == ' ') || (space[hurdleX1 - 1][hurdleY1] == '+'))

        {

            space[hurdleX1][hurdleY1] = h\_item;

            gotoxy(hurdleY1, hurdleX1);

            cout << h\_item;

            gotoxy(hurdleY1, hurdleX1);

            cout << " ";

            hurdleX1 = hurdleX1 - 1;

            h\_item = space[hurdleX1][hurdleY1];

            if (space[hurdleX1][hurdleY1] == 'Y')

            {

                hurdle\_movement();

            }

            else

            {

                SetConsoleTextAttribute(console\_color, 14);

                gotoxy(hurdleY1, hurdleX1);

                cout << "Y";

                space[hurdleX1][hurdleY1] = 'Y';

            }

        } // move up

    }

    if (move == 4)

    {

        if ((space[hurdleX1 + 1][hurdleY1] == ' ') || (space[hurdleX1 + 1][hurdleY1] == '+'))

        {

            space[hurdleX1][hurdleY1] = h\_item;

            gotoxy(hurdleY1, hurdleX1);

            cout << h\_item;

            gotoxy(hurdleY1, hurdleX1);

            cout << " ";

            hurdleX1 = hurdleX1 + 1;

            h\_item = space[hurdleX1][hurdleY1];

            if (space[hurdleX1][hurdleY1] == 'Y')

            {

                hurdle\_movement();

            }

            else

            {

                SetConsoleTextAttribute(console\_color, 3);

                gotoxy(hurdleY1, hurdleX1);

                cout << "Y";

                space[hurdleX1][hurdleY1] = 'Y';

            }

        }

    } // move down

    if (spaceX == hurdleX1 && spaceY == hurdleY1)

    {

        gotoxy(hurdleY1, hurdleX1);

        cout << "Y";

        return 0;

    }

    return 1;

}

/////////////////  2nd hurdle ////////////

bool hurdle2\_movement()

{

    int move = hurdle\_direction();

    if (move == 1)

    {

        if ((space[hurdleX2][hurdleY2 - 1] == ' ') || (space[hurdleX2][hurdleY2 - 1] == '+'))

        {

            space[hurdleX2][hurdleY2] = h2\_item;

            gotoxy(hurdleY2, hurdleX2);

            cout << h2\_item;

            gotoxy(hurdleY2, hurdleX2);

            cout << " ";

            hurdleY2 = hurdleY2 - 1;

            h2\_item = space[hurdleX2][hurdleY2];

            if (space[hurdleX2][hurdleY2] == 'Y')

            {

                hurdle2\_movement();

            }

            else

            {

                gotoxy(hurdleY2, hurdleX2);

                cout << "Y";

                space[hurdleX2][hurdleY2] = 'Y';

            }

        }

    } // move left

    if (move == 2)

    {

        if ((space[hurdleX2][hurdleY2 + 1] == ' ') || (space[hurdleX2][hurdleY2 + 1] == '+'))

        {

            space[hurdleX2][hurdleY2] = h2\_item;

            gotoxy(hurdleY2, hurdleX2);

            cout << h2\_item;

            gotoxy(hurdleY2, hurdleX2);

            cout << " ";

            hurdleY2 = hurdleY2 + 1;

            h2\_item = space[hurdleX2][hurdleY2];

            if (space[hurdleX2][hurdleY2] == 'Y')

            {

                hurdle2\_movement();

            }

            else

            {

                gotoxy(hurdleY2, hurdleX2);

                cout << "Y";

                space[hurdleX2][hurdleY2] = 'Y';

            }

        } // move right

    }

    if (move == 3)

    {

        if ((space[hurdleX2 - 1][hurdleY2] == ' ') || (space[hurdleX2 - 1][hurdleY2] == '+'))

        {

            space[hurdleX2][hurdleY2] = h2\_item;

            gotoxy(hurdleY2, hurdleX2);

            cout << h2\_item;

            gotoxy(hurdleY2, hurdleX2);

            cout << " ";

            hurdleX2 = hurdleX2 - 1;

            h2\_item = space[hurdleX2][hurdleY2];

            if (space[hurdleX2][hurdleY2] == 'Y')

            {

                hurdle2\_movement();

            }

            else

            {

                gotoxy(hurdleY2, hurdleX2);

                cout << "Y";

                space[hurdleX2][hurdleY2] = 'Y';

            }

        } // move up

    }

    if (move == 4)

    {

        if ((space[hurdleX2 + 1][hurdleY2] == ' ') || (space[hurdleX2 + 1][hurdleY2] == '+'))

        {

            space[hurdleX2][hurdleY2] = h2\_item;

            gotoxy(hurdleY2, hurdleX2);

            cout << h2\_item;

            gotoxy(hurdleY2, hurdleX2);

            cout << " ";

            hurdleX2 = hurdleX2 + 1;

            h2\_item = space[hurdleX2][hurdleY2];

            if (space[hurdleX2][hurdleY2] == 'Y')

            {

                hurdle2\_movement();

            }

            else

            {

                gotoxy(hurdleY2, hurdleX2);

                cout << "Y";

                space[hurdleX2][hurdleY2] = 'Y';

            }

        }

    } // move down

    if (spaceX == hurdleX2 && spaceY == hurdleY2)

    {

        gotoxy(hurdleY2, hurdleX2);

        cout << "Y";

        return 0;

    }

    return 1;

}

////////////  3rd hurdel /////////////

bool hurdle3\_movement()

{

    int move = hurdle\_direction();

    if (move == 1)

    {

        if ((space[hurdleX3][hurdleY3 - 1] == ' ') || (space[hurdleX3][hurdleY3 - 1] == '+'))

        {

            space[hurdleX3][hurdleY3] = h3\_item;

            gotoxy(hurdleY3, hurdleX3);

            cout << h3\_item;

            gotoxy(hurdleY3, hurdleX3);

            cout << " ";

            hurdleY3 = hurdleY3 - 1;

            h3\_item = space[hurdleX3][hurdleY3];

            if (space[hurdleX3][hurdleY3] == 'Y')

            {

                hurdle3\_movement();

            }

            else

            {

                gotoxy(hurdleY3, hurdleX3);

                cout << "Y";

                space[hurdleX3][hurdleY3] = 'Y';

            }

        }

    } // move left

    if (move == 2)

    {

        if ((space[hurdleX3][hurdleY3 + 1] == ' ') || (space[hurdleX3][hurdleY3 + 1] == '+'))

        {

            space[hurdleX3][hurdleY3] = h3\_item;

            gotoxy(hurdleY3, hurdleX3);

            cout << h3\_item;

            gotoxy(hurdleY3, hurdleX3);

            cout << " ";

            hurdleY3 = hurdleY3 + 1;

            h3\_item = space[hurdleX3][hurdleY3];

            if (space[hurdleX3][hurdleY3] == 'Y')

            {

                hurdle3\_movement();

            }

            else

            {

                gotoxy(hurdleY3, hurdleX3);

                cout << "Y";

                space[hurdleX3][hurdleY3] = 'Y';

            }

        } // move right

    }

    if (move == 3)

    {

        if ((space[hurdleX3 - 1][hurdleY3] == ' ') || (space[hurdleX3 - 1][hurdleY3] == '+'))

        {

            space[hurdleX3][hurdleY3] = h3\_item;

            gotoxy(hurdleY3, hurdleX3);

            cout << h3\_item;

            gotoxy(hurdleY3, hurdleX3);

            cout << " ";

            hurdleX3 = hurdleX3 - 1;

            h3\_item = space[hurdleX3][hurdleY3];

            if (space[hurdleX3][hurdleY3] == 'Y')

            {

                hurdle3\_movement();

            }

            else

            {

                gotoxy(hurdleY3, hurdleX3);

                cout << "Y";

                space[hurdleX3][hurdleY3] = 'Y';

            }

        } // move up

    }

    if (move == 4)

    {

        if ((space[hurdleX3 + 1][hurdleY3] == ' ') || (space[hurdleX3 + 1][hurdleY3] == '+'))

        {

            space[hurdleX3][hurdleY3] = h3\_item;

            gotoxy(hurdleY3, hurdleX3);

            cout << h3\_item;

            gotoxy(hurdleY3, hurdleX3);

            cout << " ";

            hurdleX3 = hurdleX3 + 1;

            h3\_item = space[hurdleX3][hurdleY3];

            if (space[hurdleX3][hurdleY3] == 'Y')

            {

                hurdle3\_movement();

            }

            else

            {

                gotoxy(hurdleY3, hurdleX3);

                cout << "Y";

                space[hurdleX3][hurdleY3] = 'Y';

            }

        }

    } // move down

    if (spaceX == hurdleX3 && spaceY == hurdleY3)

    {

        gotoxy(hurdleY3, hurdleX3);

        cout << "Y";

        return 0;

    }

    return 1;

}

///////  4th hurdle ////////////

bool hurdle4\_movement()

{

    int move = hurdle\_direction();

    if (move == 1)

    {

        if ((space[hurdleX4][hurdleY4 - 1] == ' ') || (space[hurdleX4][hurdleY4 - 1] == '+'))

        {

            space[hurdleX4][hurdleY4] = h4\_item;

            gotoxy(hurdleY4, hurdleX4);

            cout << h4\_item;

            gotoxy(hurdleY4, hurdleX4);

            cout << " ";

            hurdleY4 = hurdleY4 - 1;

            h4\_item = space[hurdleX4][hurdleY4 - 1];

            if (space[hurdleX4][hurdleY4] == 'Y')

            {

                hurdle4\_movement();

            }

            else

            {

                gotoxy(hurdleY4, hurdleX4);

                cout << "Y";

                space[hurdleX4][hurdleY4] = 'Y';

            }

        }

    } // move left

    if (move == 2)

    {

        if ((space[hurdleX4][hurdleY4 + 1] == ' ') || (space[hurdleX4][hurdleY4 + 1] == '+'))

        {

            space[hurdleX4][hurdleY4] = h4\_item;

            gotoxy(hurdleY4, hurdleX4);

            cout << h4\_item;

            gotoxy(hurdleY4, hurdleX4);

            cout << " ";

            hurdleY4 = hurdleY4 + 1;

            h4\_item = space[hurdleX4][hurdleY4];

            if (space[hurdleX4][hurdleY4] == 'Y')

            {

                hurdle4\_movement();

            }

            else

            {

                gotoxy(hurdleY4, hurdleX4);

                cout << "Y";

                space[hurdleX4][hurdleY4] = 'Y';

            }

        } // move right

    }

    if (move == 3)

    {

        if ((space[hurdleX4 - 1][hurdleY4] == ' ') || (space[hurdleX4 - 1][hurdleY4] == '+'))

        {

            space[hurdleX4][hurdleY4] = h4\_item;

            gotoxy(hurdleY4, hurdleX4);

            cout << h4\_item;

            gotoxy(hurdleY4, hurdleX4);

            cout << " ";

            hurdleX4 = hurdleX4 - 1;

            h4\_item = space[hurdleX4][hurdleY4];

            if (space[hurdleX4][hurdleY4] == 'Y')

            {

                hurdle4\_movement();

            }

            else

            {

                gotoxy(hurdleY4, hurdleX4);

                cout << "Y";

                space[hurdleX4][hurdleY4] = 'Y';

            }

        } // move up

    }

    if (move == 4)

    {

        if ((space[hurdleX4 + 1][hurdleY4] == ' ') || (space[hurdleX4 + 1][hurdleY4] == '+'))

        {

            space[hurdleX4][hurdleY4] = h4\_item;

            gotoxy(hurdleY4, hurdleX4);

            cout << h4\_item;

            gotoxy(hurdleY4, hurdleX4);

            cout << " ";

            gotoxy(hurdleY4, hurdleX4);

            cout << " ";

            hurdleX4 = hurdleX4 + 1;

            h4\_item = space[hurdleX4][hurdleY4];

            if (space[hurdleX4][hurdleY4] == 'Y')

            {

                hurdle4\_movement();

            }

            else

            {

                gotoxy(hurdleY4, hurdleX4);

                cout << "Y";

                space[hurdleX4][hurdleY4] = 'Y';

            }

        }

    } // move down

    if (spaceX == hurdleX4 && spaceY == hurdleY4)

    {

        gotoxy(hurdleY4, hurdleX4);

        cout << "Y";

        return 0;

    }

    return 1;

}

void split\_fire()

{

    if (space[spaceX - 1][spaceY] != '|' && space[spaceX - 1][spaceY] != '#' && space[spaceX - 1][spaceY] != '%')

    {

        gotoxy(spaceY, spaceX - 1);

        cout << "^";

        space[spaceX - 1][spaceY] = '^';

    }

}

void MoveFire()

{

    for (int r = 0; r < 24; r++)

    {

        for (int c = 0; c < 71; c++)

        {

            if (space[r][c] == '^')

            {

                gotoxy(c, r);

                cout << " ";

                space[r][c] = ' ';

                if (space[r - 1][c] != '|' && space[r - 1][c] != '#' && space[r - 1][c] != '%' && space[r - 1][c] != 'o')

                {

                    gotoxy(c, r - 1);

                    cout << "^";

                    if (space[r - 1][c] == 'Y')

                    {

                        score = score + 5;

                        gotoxy(c, r - 1);

                        cout << "o ";

                        space[r - 1][c] = 'o';

                        if (hurdleX1 == r - 1 && hurdleY1 == c)

                        {

                            shootCount1++;

                            if (shootCount1 == 3)

                            {

                                H1Live = false;

                                levelup++;

                                shootCount1 = 0;

                            }

                        }

                        if (hurdleX2 == r - 1 && hurdleY2 == c)

                        {

                            shootCount2++;

                            if (shootCount2 == 3)

                            {

                                H2Live = false;

                                levelup++;

                                shootCount2 = 0;

                            }

                        }

                        if (hurdleX3 == r - 1 && hurdleY3 == c)

                        {

                            shootCount3++;

                            if (shootCount3 == 3)

                            {

                                H3Live = false;

                                levelup++;

                                shootCount3 = 0;

                            }

                        }

                        if (hurdleX4 == r - 1 && hurdleY4 == c)

                        {

                            shootCount4++;

                            if (shootCount4 == 3)

                            {

                                H4Live = false;

                                levelup++;

                                shootCount4 = 0;

                            }

                        }

                    }

                    else if (space[r - 1][c] == '\*')

                    {

                        gotoxy(c, r - 1);

                        cout << "^";

                        space[r - 1][c] = '^';

                    }

                    else

                    {

                        space[r - 1][c] = '^';

                    }

                }

            }

        }

    }

}

void health()

{

    bool total\_lives = gameRunning1 && gameRunning2 && gameRunning3 && gameRunning4;

    if (!total\_lives)

    {

        if (lives != 0)

        {

            gameRunning1 = true;

            gameRunning2 = true;

            gameRunning3 = true;

            gameRunning4 = true;

            gotoxy(spaceY, spaceX);

            cout << " ";

            spaceX = 22;

            spaceY = 31;

            gotoxy(spaceY, spaceX);

            cout << "+";

            lives--;

        }

    }

    if (lives == 0)

    {

        system("CLS");

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

        cout << "//             GAME OVER             //" << endl;

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

    }

}

void total\_lives()

{

    gotoxy(75, 16);

    cout << "Total lives: " << lives;

}

void calculate\_score()

{

    score = score + 1;

}

void print\_result()

{

    gotoxy(75, 10);

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

    gotoxy(75, 11);

    cout << "--        SCORE BOARD        --";

    gotoxy(75, 12);

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

    gotoxy(75, 14);

    cout << "Total score: " << score;

    gotoxy(75, 18);

    cout << "Level: " << level;

}

void header()

{

    gotoxy(75, 0);

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

    gotoxy(75, 1);

    cout << "|||||               SPACE INVADERS               |||||" << endl;

    gotoxy(75, 2);

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

}

void hurdle\_fire()

{

    for (int r = 23; r > 0; r--)

    {

        for (int c = 0; c < 71; c++)

        {

            if (space[r][c] == '\*')

            {

                gotoxy(c, r);

                cout << " ";

                space[r][c] = ' ';

                if (space[r + 1][c] != '|' && space[r + 1][c] != '#' && space[r + 1][c] != '%')

                {

                    if (spaceX == r + 1 && spaceY == c)

                    {

                        gotoxy(c, r + 1);

                        cout << " ";

                        space[r + 1][c] = ' ';

                        lives--;

                        spaceX = 22;

                        spaceY = 31;

                        gotoxy(spaceY, spaceX);

                        cout << "+";

                    }

                    else

                    {

                        gotoxy(c, r + 1);

                        cout << "\*";

                        space[r + 1][c] = '\*';

                    }

                }

            }

        }

    }

}

void split\_hurdle\_fire1()

{

    if (space[hurdleX1 + 1][hurdleY1] != '|' && space[hurdleX1 + 1][hurdleY1] != '#' && space[hurdleX1 + 1][hurdleY1] != '%')

    {

        gotoxy(hurdleY1, hurdleX1 + 1);

        cout << "\*";

        space[hurdleX1 + 1][hurdleY1] = '\*';

    }

}

void split\_hurdle\_fire2()

{

    if (space[hurdleX2 + 1][hurdleY2] != '|' && space[hurdleX2 + 1][hurdleY2] != '#' && space[hurdleX2 + 1][hurdleY2] != '%')

    {

        gotoxy(hurdleY2, hurdleX2 + 1);

        cout << "\*";

        space[hurdleX2 + 1][hurdleY2] = '\*';

    }

}

void split\_hurdle\_fire3()

{

    if (space[hurdleX3 + 1][hurdleY3] != '|' && space[hurdleX3 + 1][hurdleY3] != '#' && space[hurdleX3 + 1][hurdleY3] != '%')

    {

        gotoxy(hurdleY3, hurdleX3 + 1);

        cout << "\*";

        space[hurdleX3 + 1][hurdleY3] = '\*';

    }

}

void split\_hurdle\_fire4()

{

    if (space[hurdleX4 + 1][hurdleY4] != '|' && space[hurdleX4 + 1][hurdleY4] != '#' && space[hurdleX4 + 1][hurdleY4] != '%')

    {

        gotoxy(hurdleY4, hurdleX4 + 1);

        cout << "\*";

        space[hurdleX4 + 1][hurdleY4] = '\*';

    }

}

void gameOpener()

{

    Sleep(200);

    gotoxy(35, 5);

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

    Sleep(200);

    gotoxy(35, 6);

    cout << "|||||               SPACE SHOOTER                |||||" << endl;

    Sleep(200);

    gotoxy(35, 7);

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

}

void storeSpaceIntoFile()

{

    fstream file;

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

    for (int r = 0; r < 24; r++)

    {

        if (r > 0)

        {

            file << endl;

        }

        for (int c = 0; c < 71; c++)

        {

            file << space[r][c];

        }

    }

    file.close();

}

void loadSpaceIntoArray()

{

    string line;

    fstream file;

    file.open("space.txt", ios::in);

    for (int r = 0; r < 24; r++)

    {

        getline(file, line);

        for (int c = 0; c < 71; c++)

        {

            space[r][c] = line[c];

         }

    }

    file.close();

}

**Student Reg. No. :**  2021-CS-170  **Student Name.**  Bisma Muhamad Ali

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 are 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 consistent and according to given **guidelines**. The project **Poster** is professionally designed and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meets more than 80% of the criteria given. | Documentation meets more than 50% of the criteria. | When the documentation meets 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:** | The project has at least 1 Player and 3 enemies. Proper use of gotoxy() function. Health system, Firing System and lives decreasing system. In the case of the 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 are followed | All code style criteria followed but some improvements required | A lot of improvements are 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 are synchronized. | Code and documentation does not synchronize at **some** places | Code and documentation do not synchronize in **many** places | Code and documentation **do not** synchronize. |
| Idea Novelty and Creativity  **Grade:** | The idea is unique of the game | The 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:** | The data structure is sufficient for the project requirements | Data Structure is sufficient but requires improvement to meet project requirements. | The data structure is not sufficient and needs a lot of improvement | Data Structure is not properly identified and declared. |
| File Handling  **Grade:** | The game maze is loaded and the updated maze is stored in the file | The game maze is loaded and partial data is stored in the file. | The game maze is loaded, but the updated game configuration is not stored. | The project does 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 a 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 a lot of places | The 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 | The presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | The student has a complete understanding of how the code is working and knows the concept. | The student has good understanding but in someplace he does not know the concepts | The student has very little understanding and lacks the major concepts. | The student does not have any level of understanding of the code. |

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