COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

COMPUTER GRAPHICS (UCS505)

PROJECT ON:

STAR WARS

Submitted to

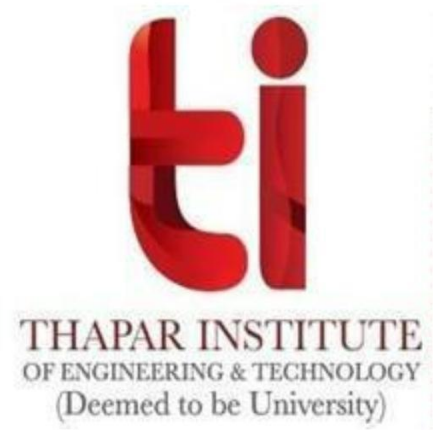
Ms. Namrata

Submitted by:

Harsh Kashyap (101917088)

Nikhil Singla (101917095)

Ritik Raj (101917101)



Department of Computer Science and Engineering

Thapar Institute of Engineering & Technology

TABLE OF CONTENTS

INTRODUCTION:

This project is designed and presented by Harsh Kashyap [101917088], Nikhil Singla [101917095], Ritik Raj [101917101] as a part of our curriculum in Computer Graphics course, Jan-Jun -2022, TIET.

Star Wars is a two-player game that is set in a space-like theme where each player 's goal is to eliminate the other player by shooting lasers directed towards the other player and reducing the latter's life from 100 to 0.

Working and Explanation:

The game begins with the two spaceships placed at opposite screen corners and centrally aligned. Both the player's lives are initially set to 100. The goal is to shoot laser beams and reduce the opponent's life to 0, where each successful hit reduces the other player's life by 5 points. Both players can move their spaceships in order to dodge from the incoming lasers and fire their lasers simultaneously.

Each player controls a spaceship in the space field, which he/she can move in all four directions, ie, up, down, left, right. Besides moving the spaceship, the players can also shoot lasers directed towards the opponent, and if they are successful in hitting their target, the other player's life gets reduced by 5 points. If the incoming laser beam misses the other player's spaceship, then his/her life won't be reduced.

The keys through which each player controls the his/her spaceship

are: For player 1(situated at the left of the screen):

Up movement: "W"

Down movement: "S"

Left movement: "A"

Right Movement: "D"

Fire Laser: "C"

For player 2(situated at the right of the screen):

Up movement: "I"

Down movement: "K"

Left movement: "J"

Right Movement: "L"

Fire Laser: “M”

Rules:

The game is simple with just one rule: The player whose life points reduces to 0 first loses.

CONCEPTS USED

2D and 3D transformations: -

- Translation - Movement of object on screen/in space without deformation. We translate a point by adding to x, y and z coordinates in accordance with keys pressed.
- Rotation - Process of changing the angle of object about any pivot point along any axis/plane.
- Scaling - Changing the size of an object. We scale an object by scaling the x and y coordinates of each vertex in the object.

Drawing: -

- Polygon: GL_POLYGON is used. Polygon is constructed by looping over different points in the space. Used to draw the buttons of the menu screen and specifications of aliens.
- Polygon outline: GL_LINE_LOOP is used to make the borders by specifying vertices.
- Line: GL_LINE_STRIP is used to make the body outlines and specific features of the aliens.
- Button click: GLUT_LEFT_BUTTON and GLUT_DOWN are used to check whether a button is pressed or not.

Polygon Filling: By using glcolor3f, we could fill colors inside any polygon by filling the RGB values.

Point Size: Before using the draw functions for points, point size is specified.

FUNCTIONS USED :-

- **void displayRasterText(float x, float y, float z, char* stringToDisplay):** Used to display the text at a particular position specified.
- **void introScreen():** Displays the intro screen.
- **void startScreenDisplay():** Displays the menu screen.
- **void backButton():** Used in the instruction screen to go back to the menu screen.
- **void instructionsScreenDisplay()**: Displays the instruction screen.
- **void DrawAlienBody(bool isPlayer1)**: Draws the aliens' bodies.
- **void DrawAlienCollar()**: Draws the aliens' collars using a line strip.
- **void DrawAlienFace(bool isPlayer1)**: Draws the aliens' faces with the specifications.
- **void DrawAlienBeak():** Draws the aliens' beaks using a line strip.
- **void DrawAlienEyes(bool isPlayer1):** Draws the aliens' eyes using translation, scaling and rotation.
- **void DrawAlien(bool isPlayer1):** Calling all the alien functions to display the aliens on the screen.
- **void DrawSpaceshipBody(bool isPlayer1)**: Draws the body of a spaceship.
- **void DrawSteeringWheel():** Draws the steering wheels of both spaceships.
- **void DrawSpaceshipDoom()**: Draws the dooms of spaceships.
- **void DrawLaser(int x, int y, bool dir[]) :** Draws the laser when pressed with a specific key for a particular spaceship.
- **void SpaceshipCreate(int x, int y, bool isPlayer1):** Calls all the predefined functions to create the spaceships.
- **void DisplayHealthBar1() :** Displays the health bar of player1 and updates the health accordingly.
- **void DisplayHealthBar2() :** Displays the health bar of player2 and updates the health accordingly.
- **void checkLaserContact(int x, int y, bool dir[], int xp, int yp, bool player1):** Checks whether the spaceship and laser contacted or not.
- **void gameScreenDisplay():** Calls all the functions to display the game screen.
- **void displayGameOverMessage():** Displays the game over message.
- **void keyOperations()** : Defines all the key controls for both the players.
- **void display()**: Displays the intro screen.
- **void mouseClick(int buttonPressed ,int state ,int x, int y) :** MouseClick functions defined here.
- **void keyPressed(unsigned char key, int x, int y):** Updates the keypressed array.
- **void refresh()** : Refreshes the display.
- **void keyReleased(unsigned char key, int x, int y)**: Updates the array that keys are released.

CODE

```
#ifdef _WIN32
#include <windows.h>
#endif
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <math.h>
#define GL_SILENCE_DEPRECATION

#define XMAX 1200
#define YMAX 700
#define SPACESHIP_SPEED 30
#define TOP 0
#define RIGHT 1
#define BOTTOM 2
#define LEFT 3

GLint m_viewport[4];
bool mButtonPressed = false;
float mouseX, mouseY;
enum view
{
    INTRO,
    MENU,
    INSTRUCTIONS,
    GAME,
    GAMEOVER
};
view viewPage = INTRO; // initial value
bool keyStates[256] = { false };
bool direction[4] = { false };
bool laser1Dir[2] = { false };
bool laser2Dir[2] = { false };

int alienLife1 = 100;
int alienLife2 = 100;
bool gameOver = false;
float xOne = 500, yOne = 0;
float xTwo = 500, yTwo = 0;
bool laser1 = false, laser2 = false;
```

```
GLint CI = 0;
GLfloat a[][2] = { 0, -50, 70, -50, 70, 70, -70, 70 };
GLfloat LightColor[][3] = { 1, 1, 0, 0, 1, 1, 0, 1, 0 };
GLfloat AlienBody[][2] = { {-4, 9}, {-6, 0}, {0, 0}, {0.5, 9}, {0.15,
12}, {-14, 18}, {-19, 10}, {-20, 0}, {-6, 0} };



GLfloat AlienCollar[][2] = { {-9, 10.5}, {-6, 11}, {-5, 12}, {6, 18},
{10, 20}, {13, 23}, {16, 30}, {19, 39}, {16, 38}, {10, 37}, {-13, 39},
{-18, 41}, {-20, 43}, {-20.5, 42}, {-21, 30}, {-19.5, 23}, {-19, 20},
{-14, 16}, {-15, 17}, {-13, 13}, {-9, 10.5} };
GLfloat ALienFace[][2] = { {-6, 11}, {-4.5, 18}, {0.5, 20}, {0., 20.5},
{0.1, 19.5}, {1.8, 19}, {5, 20}, {7, 23}, {9, 29}, {6, 29.5}, {5, 28},
{7, 30}, {10, 38}, {11, 38}, {11, 40}, {11.5, 48}, {10, 50.5}, {8.5,
51}, {6, 52}, {1, 51}, {-3, 50}, {-1, 51}, {-3, 52}, {-5, 52.5}, {-6,
52}, {-9, 51}, {-10.5, 50}, {-12, 49}, {-12.5, 47}, {-12, 43}, {-13,
40}, {-12, 38.5}, {-13.5, 33}, {-15, 38}, {-14.5, 32}, {-14, 28},
{-13.5, 33}, {-14, 28}, {-13.8, 24}, {-13, 20}, {-11, 19}, {-10.5, 12},
{-6, 11} };
GLfloat ALienBeak[][2] = { {-6, 21.5}, {-6.5, 22}, {-9, 21}, {-11,
20.5}, {-20, 20}, {-14, 23}, {-9.5, 28}, {-7, 27}, {-6, 26.5}, {-4.5,
23}, {-4, 21}, {-6, 19.5}, {-8.5, 19}, {-10, 19.5}, {-11, 20.5} };

void displayRasterText(float x, float y, float z, char*
stringToDisplay)
{
    glRasterPos3f(x, y, z);
    for (char* c = stringToDisplay; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *c);
    }
}

void init()
{
    glClearColor(0.0, 0.0, 0.0, 0);
    glColor3f(1.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    gluOrtho2D(-1200, 1200, -700, 700); //<-----CHANGE THIS TO GET
EXTRA SPACE
```

```
//gluOrtho2D(-200,200,-200,200);
    glMatrixMode(GL_MODELVIEW);
}

void introScreen()
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1.0, 1.0, 1.0);
    //displayRasterText(x,y);
    displayRasterText(-500, 490, 0.0, (char*)"THAPAR INSTITUTE OF
ENGINEERING AND TECHNOLOGY");
    glColor3f(1.0, 1.0, 1.0);
    displayRasterText(-500, 385, 0.0, (char*)"DEPARTMENT OF COMPUTER
SCIENCE AND ENGINEERING");
    glColor3f(0.0, 0.0, 1.0);
    displayRasterText(-225, 300, 0.0, (char*)"A MINI PROJECT ON ");
    glColor3f(1.0, 0.0, 1.0);


   displayRasterText(-125, 225, 0.0, (char*)"Star Wars");
   glColor3f(1.0, 0.0, 0.0);
   displayRasterText(-190, 100, 0.0, (char*)" TEAM MEMBERS (3CSE 4)");
   glColor3f(1.0, 1.0, 1.0);

    displayRasterText(-900, -50, 0.0, (char*)" Harsh Kashyap
[101917088]");
    displayRasterText(-300, -50, 0.0, (char*)" Nikhil Singla
[101917095]");
    displayRasterText(400, -50, 0.0, (char*)" Ritik Raj
[101917101]");
    glColor3f(1.0, 0.0, 0.0);
    displayRasterText(-400, -150, 0.0, (char*)"SUBMITTED TO:
 Ms Namrata Dimari");
    glColor3f(1.0, 1.0, 1.0);
    displayRasterText(500, -200, 0.0, (char*)"");
    glColor3f(1.0, 0.0, 0.0);
    displayRasterText(-250, -400, 0.0, (char*)"Academic Year
2021-2022");
    glColor3f(1.0, 1.0, 1.0);
    displayRasterText(-300, -550, 0.0, (char*)"Press ENTER to start the
game");
```

```
    glFlush();
    glutSwapBuffers();
}

void startScreenDisplay()
{
    glLineWidth(10);
    // SetDisplayMode(MENU_SCREEN);

    glColor3f(1, 0, 0);
    glBegin(GL_LINE_LOOP); // Border
    glVertex2f(-750, -500);
    glVertex2f(-750, 550);
    glVertex2f(750, 550);
    glVertex2f(750, -500);
    glEnd();

    glLineWidth(1);

    glColor3f(1, 1, 0);
    glBegin(GL_POLYGON); // START GAME POLYGON
    glVertex2f(-200, 300);
    glVertex2f(-200, 400);
    glVertex2f(200, 400);
    glVertex2f(200, 300);
    glEnd();

    glBegin(GL_POLYGON); // INSTRUCTIONS POLYGON
    glVertex2f(-200, 50);
    glVertex2f(-200, 150);
    glVertex2f(200, 150);
    glVertex2f(200, 50);
    glEnd();




    glBegin(GL_POLYGON); // QUIT POLYGON
    glVertex2f(-200, -200);
    glVertex2f(-200, -100);
    glVertex2f(200, -100);
    glVertex2f(200, -200);
    glEnd();
```

```
    if (mouseX >= -100 && mouseX <= 100 && mouseY >= 150 && mouseY <=
200)
    {
        glColor3f(0, 0, 1);
        if (mButtonPressed)
        {
            alienLife1 = alienLife2 = 100;
            viewPage = GAME;
            mButtonPressed = false;
        }
    }
    else
        glColor3f(0, 0, 0);

    displayRasterText(-100, 340, 0.4, (char*)"Start Game");

    if (mouseX >= -100 && mouseX <= 100 && mouseY >= 30 && mouseY <=
80)
    {
        glColor3f(0, 0, 1);
        if (mButtonPressed)
        {
            viewPage = INSTRUCTIONS;
            mButtonPressed = false;
        }
    }
    else
        glColor3f(0, 0, 0);
    displayRasterText(-120, 80, 0.4, (char*)"Instructions");

    if (mouseX >= -100 && mouseX <= 100 && mouseY >= -90 && mouseY <=
-40)
    {
        glColor3f(0, 0, 1);
        if (mButtonPressed)
        {
            mButtonPressed = false;
            exit(0);
        }
    }
    else
        glColor3f(0, 0, 0);
```

```
    displayRasterText(-100, -170, 0.4, (char*)"    Quit");
    glutPostRedisplay();
}

void backButton()
{
    if (mouseX <= -450 && mouseX >= -500 && mouseY >= -275 && mouseY <=
-250)
    {
        glColor3f(0, 0, 1);
        if (mButtonPressed)
        {
            viewPage = MENU;
            mButtonPressed = false;
            glutPostRedisplay();
        }
    }
    else
        glColor3f(1, 0, 0);
    displayRasterText(-1000, -550, 0, (char*)"Back");
}

void instructionsScreenDisplay()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glColor3f(1, 0, 0);
    displayRasterText(-900, 550, 0.4, (char*)"INSTRUCTIONS");
    glColor3f(1, 0, 0);
    displayRasterText(-1000, 400, 0.4, (char*)"PLAYER 1");
    displayRasterText(200, 400, 0.4, (char*)"PLAYER 2");
    glColor3f(1, 1, 1);
    displayRasterText(-1100, 300, 0.4, (char*)"Key 'w' to move up.");
    displayRasterText(-1100, 200, 0.4, (char*)"Key 's' to move down.");
    displayRasterText(-1100, 100, 0.4, (char*)"Key 'd' to move
right.");
    displayRasterText(-1100, 0, 0.4, (char*)"Key 'a' to move left.");
    displayRasterText(100, 300, 0.4, (char*)"Key 'i' to move up.");
    displayRasterText(100, 200, 0.4, (char*)"Key 'k' to move down.");
    displayRasterText(100, 100, 0.4, (char*)"Key 'j' to move right.");
    displayRasterText(100, 0, 0.4, (char*)"Key 'l' to move left.");
    displayRasterText(-1100, -100, 0.4, (char*)"Key 'c' to shoot, Use
'w' and 's' to change direction.");
    displayRasterText(100, -100, 0.4, (char*)"Key 'm' to shoot, Use 'i'
```

```
and 'k' to change direction.");
    displayRasterText(-1100, -300, 0.4, (char*)"The Objective is to
kill your opponent.");
    displayRasterText(-1100, -370, 0.4, (char*)"Each time a player gets
shot, LIFE decreases by 5 points.");
    backButton();
}

void DrawAlienBody(bool isPlayer1)
{
    if (isPlayer1)
        glColor3f(0, 1, 0);



    else
        glColor3f(1, 1, 0); // BODY color
    glBegin(GL_POLYGON);
    for (int i = 0; i <= 8; i++)
        glVertex2fv(AlienBody[i]);
    glEnd();

    glColor3f(0, 0, 0); // BODY Outline
    glLineWidth(1);
    glBegin(GL_LINE_STRIP);
    for (int i = 0; i <= 8; i++)
        glVertex2fv(AlienBody[i]);
    glEnd();

    glBegin(GL_LINES); // BODY effect
    glVertex2f(-13, 11);
    glVertex2f(-15, 9);
    glEnd();
}
void DrawAlienCollar()
{
    glColor3f(1, 0, 0); // COLLAR
    glBegin(GL_POLYGON);
    for (int i = 0; i <= 20; i++)
        glVertex2fv(AlienCollar[i]);
    glEnd();

    glColor3f(0, 0, 0); // COLLAR outline
```

```
    glBegin(GL_LINE_STRIP);
    for (int i = 0; i <= 20; i++)
        glVertex2fv(AlienCollar[i]);
    glEnd();
}
void DrawAlienFace(bool isPlayer1)
{
    glColor3f(0, 0, 1);
    glBegin(GL_POLYGON);
    for (int i = 0; i <= 42; i++)
        glVertex2fv(ALienFace[i]);
    glEnd();
    glColor3f(0, 0, 0); // FACE outline
    glBegin(GL_LINE_STRIP);
    for (int i = 0; i <= 42; i++)
        glVertex2fv(ALienFace[i]);
    glEnd();
    glBegin(GL_LINE_STRIP); // EAR effect
    glVertex2f(3.3, 22);
    glVertex2f(4.4, 23.5);



    glVertex2f(6.3, 26);
    glEnd();
}
void DrawAlienBeak()
{
    glColor3f(1, 1, 0); // BEAK color
    glBegin(GL_POLYGON);
    for (int i = 0; i <= 14; i++)
        glVertex2fv(ALienBeak[i]);
    glEnd();

    glColor3f(0, 0, 0); // BEAK outline
    glBegin(GL_LINE_STRIP);
    for (int i = 0; i <= 14; i++)
        glVertex2fv(ALienBeak[i]);
    glEnd();
}
void DrawAlienEyes(bool isPlayer1)
{
```

```
    glColor3f(0, 1, 1);
    glPushMatrix();
    glRotated(-10, 0, 0, 1);
    glTranslated(-6, 32.5, 0); // Left eye
    glScalef(2.5, 4, 0);
    glutSolidSphere(1, 20, 30);
    glPopMatrix();
    glPushMatrix();
    glRotated(-1, 0, 0, 1);
    glTranslated(-8, 36, 0); // Right eye
    glScalef(2.5, 4, 0);
    glutSolidSphere(1, 100, 100);
    glPopMatrix();
}
void DrawAlien(bool isPlayer1)
{
    DrawAlienBody(isPlayer1);
    DrawAlienCollar();
    DrawAlienFace(isPlayer1);
    DrawAlienBeak();
    DrawAlienEyes(isPlayer1);
}
void DrawSpaceshipBody(bool isPlayer1)
{
    if (isPlayer1)
        glColor3f(1, 0, 0); // BASE
    else
        glColor3f(0.5, 0, 0.5);



    glPushMatrix();
    glScalef(70, 20, 1);
    glutSolidSphere(1, 50, 50);
    glPopMatrix();

    glPushMatrix(); // LIGHTS
    glScalef(3, 3, 1);
    glTranslated(-20, 0, 0); // 1
    glColor3fv(LightColor[(CI + 0) % 3]);
    glutSolidSphere(1, 1000, 1000);
    glTranslated(5, 0, 0); // 2
    glColor3fv(LightColor[(CI + 1) % 3]);
```

```
    glutSolidSphere(1, 1000, 1000);
    glTranslated(5, 0, 0); // 3
    glColor3fv(LightColor[(CI + 2) % 3]);
    glutSolidSphere(1, 1000, 1000);
    glTranslated(5, 0, 0); // 4
    glColor3fv(LightColor[(CI + 0) % 3]);
    glutSolidSphere(1, 1000, 1000);
    glTranslated(5, 0, 0); // 5
    glColor3fv(LightColor[(CI + 1) % 3]);
    glutSolidSphere(1, 1000, 1000);
    glTranslated(5, 0, 0); // 6
    glColor3fv(LightColor[(CI + 2) % 3]);
    glutSolidSphere(1, 1000, 1000);
    glTranslated(5, 0, 0); // 7
    glColor3fv(LightColor[(CI + 0) % 3]);
    glutSolidSphere(1, 1000, 1000);
    glTranslated(5, 0, 0); // 8
    glColor3fv(LightColor[(CI + 1) % 3]);
    glutSolidSphere(1, 1000, 1000);
    glTranslated(5, 0, 0); // 9
    glColor3fv(LightColor[(CI + 2) % 3]);
    glutSolidSphere(1, 1000, 1000);
    glPopMatrix();
}
void DrawSteeringWheel()
{
    glPushMatrix();
    glLineWidth(3);
    glColor3f(0.20, 0., 0.20);
    glScalef(7, 4, 1);
    glTranslated(-1.9, 5.5, 0);
    glutWireSphere(1, 8, 8);
    glPopMatrix();
}
void DrawSpaceshipDoom()
{



    glColor4f(0.7, 1, 1, 0.0011);
    glPushMatrix();
    glTranslated(0, 30, 0);
    glScalef(35, 50, 1);
```

```
    glutSolidSphere(1, 50, 50);
    glPopMatrix();
}

void DrawLaser(int x, int y, bool dir[])
{
    int xend = -XMAX, yend = y;
    if (dir[0])
        yend = YMAX;
    else if (dir[1])
        yend = -YMAX;
    glLineWidth(5);
    glColor3f(1, 0, 0);
    glBegin(GL_LINES);
    glVertex2f(x, y);
    glVertex2f(xend, yend);
    glEnd();
}

void SpaceshipCreate(int x, int y, bool isPlayer1)
{
    glPushMatrix();
    glTranslated(x, y, 0);
    DrawSpaceshipDoom();
    glPushMatrix();
    glTranslated(4, 19, 0);
    DrawAlien(isPlayer1);
    glPopMatrix();
    DrawSteeringWheel();
    DrawSpaceshipBody(isPlayer1);
    glEnd();
    glPopMatrix();
}

void DisplayHealthBar1()
{
    char temp1[40];
    glColor3f(1, 1, 1);
    sprintf_s(temp1, "  LIFE = %d", alienLife1);
    displayRasterText(-1100, 600, 0.4, temp1);
    glColor3f(1, 0, 0);
}
```

```
void DisplayHealthBar2()



{
    char temp2[40];
    glColor3f(1, 1, 1);
    sprintf_s(temp2, "  LIFE = %d", alienLife2);
    displayRasterText(800, 600, 0.4, temp2);
    glColor3f(1, 0, 0);
}

void checkLaserContact(int x, int y, bool dir[], int xp, int yp, bool
player1)
{
    int xend = -XMAX, yend = y;
    xp += 8;
    yp += 8; // moving circle slightly up to fix laser issue
    if (dir[0])
        yend = YMAX;
    else if (dir[1])
        yend = -YMAX;

    // Here we find out if the laser(line) intersects with
spaceship(circle)
    // by solving the equations for the same and finding the
discriminant of the
    // quadratic equation obtained
    float m = (float)(yend - y) / (float)(xend - x);
    float k = y - m * x;
    int r = 50; // approx radius of the spaceship

    // calculating value of b, a, and c needed to find discriminant
    float b = 2 * xp - 2 * m * (k - yp);
    float a = 1 + m * m;
    float c = xp * xp + (k - yp) * (k - yp) - r * r;

    float d = (b * b - 4 * a * c); // discriminant for the equation
    printf("\nDisc: %f x: %d, y: %d, xp: %d, yp: %d", d, x, y, xp, yp);
    if (d >= 0)
    {
        if (player1)
            alienLife1 -= 5;
```

```
        else
            alienLife2 -= 5;

        printf("%d %d\n", alienLife1, alienLife2);
    }
}

void gameScreenDisplay()
{
    DisplayHealthBar1();
    DisplayHealthBar2();
    glScalef(2, 2, 0);



    if (alienLife1 > 0)
    {
        SpaceshipCreate(xOne, yOne, true);
        if (laser1)
        {
            DrawLaser(xOne, yOne, laser1Dir);
            checkLaserContact(xOne, yOne, laser1Dir, -xTwo, yTwo,
true);
        }
    }
    else
    {
        viewPage = GAMEOVER;
    }

    if (alienLife2 > 0)
    {
        glPushMatrix();
        glScalef(-1, 1, 1);
        SpaceshipCreate(xTwo, yTwo, false);
        if (laser2)
        {
            DrawLaser(xTwo, yTwo, laser2Dir);
            checkLaserContact(xTwo, yTwo, laser2Dir, -xOne, yOne,
false);
        }
        glPopMatrix();
```

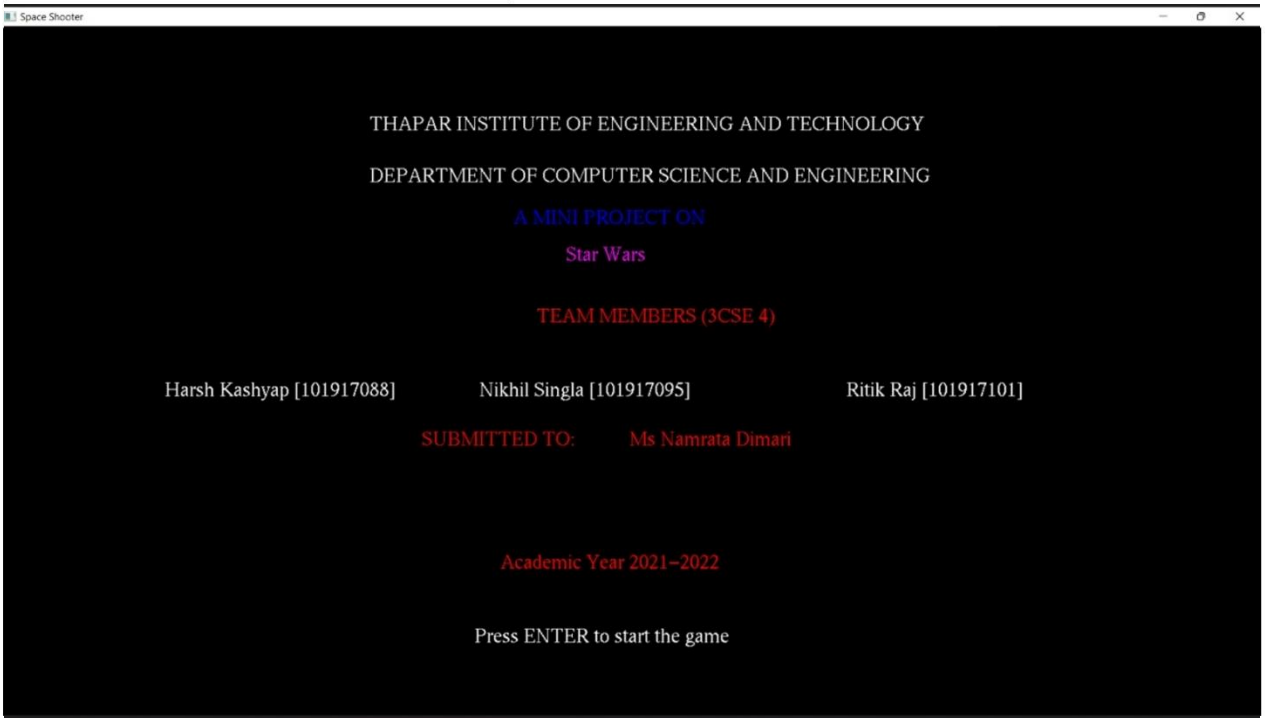
```
    }
    else
    {
        viewPage = GAMEOVER;
    }

    if (viewPage == GAMEOVER)
    {
        xOne = xTwo = 500;
        yOne = yTwo = 0;
    }
}

void displayGameOverMessage()
{
    glColor3f(1, 1, 0);
    char* message;
    if (alienLife1 > 0)
        message = (char*)"Game Over! Player 1 won the game";
    else
        message = (char*)"Game Over! Player 2 won the game";




    displayRasterText(-350, 600, 0.4, message);
}

void keyOperations()
{
    if (keyStates[13] == true && viewPage == INTRO)
    {
        viewPage = MENU;
        printf("view value changed to %d", viewPage);
        printf("enter key pressed\n");
    }
    if (viewPage == GAME)
    {
        laser1Dir[0] = laser1Dir[1] = false;
        laser2Dir[0] = laser2Dir[1] = false;
        if (keyStates['c'] == true)
        {
```

```
            laser2 = true;
            if (keyStates['w'] == true)
                laser2Dir[0] = true;
            if (keyStates['s'] == true)
                laser2Dir[1] = true;
        }
        else
        {
            laser2 = false;
            if (keyStates['d'] == true)
                xTwo -= SPACESHIP_SPEED;
            if (keyStates['a'] == true)
                xTwo += SPACESHIP_SPEED;
            if (keyStates['w'] == true)
                yTwo += SPACESHIP_SPEED;
            if (keyStates['s'] == true)
                yTwo -= SPACESHIP_SPEED;
        }

        if (keyStates['m'] == true)
        {
            laser1 = true;
            if (keyStates['i'] == true)
                laser1Dir[0] = true;
            if (keyStates['k'] == true)
                laser1Dir[1] = true;
        }
        else
        {
            laser1 = false;



            if (keyStates['l'] == true)
                xOne += SPACESHIP_SPEED;
            if (keyStates['j'] == true)
                xOne -= SPACESHIP_SPEED;
            if (keyStates['i'] == true)
                yOne += SPACESHIP_SPEED;
            if (keyStates['k'] == true)
                yOne -= SPACESHIP_SPEED;
        }
    }
```

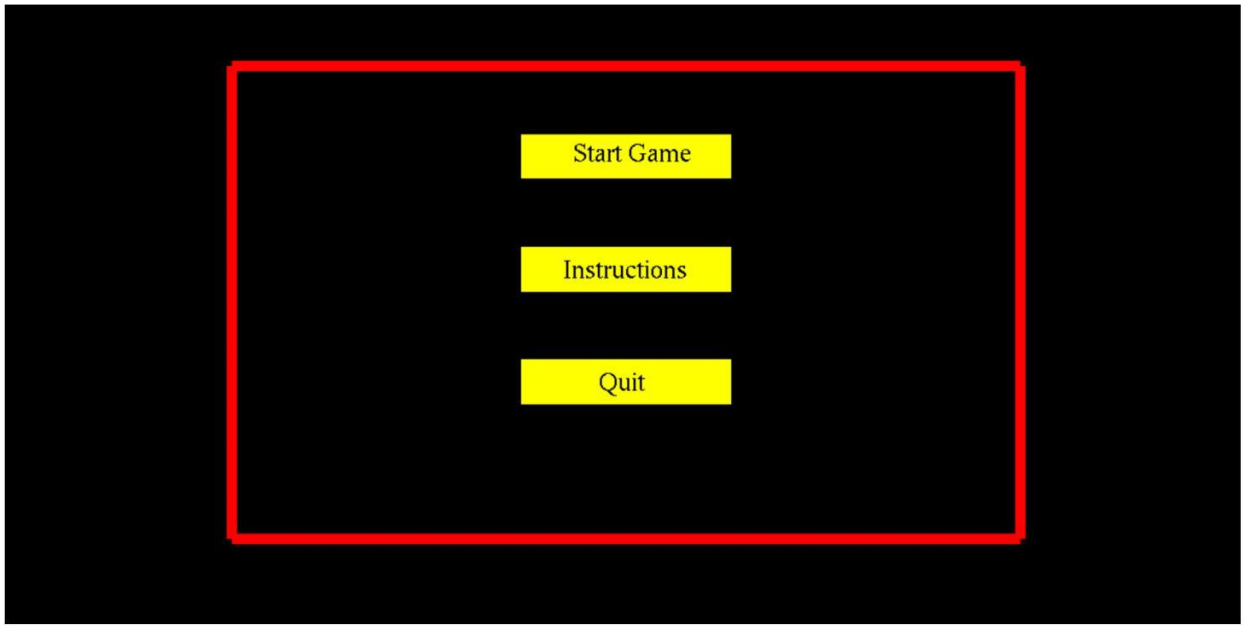
```
}
void display()
{

    keyOperations();
    glClear(GL_COLOR_BUFFER_BIT);
    switch (viewPage)
    {
    case INTRO:
        introScreen();
        break;
    case MENU:
        startScreenDisplay();
        break;
    case INSTRUCTIONS:
        instructionsScreenDisplay();
        break;
    case GAME:
        gameScreenDisplay();
        // reset scaling values
        glScalef(1 / 2, 1 / 2, 0);
        break;
    case GAMEOVER:
        displayGameOverMessage();
        startScreenDisplay();
        break;
    }

    glFlush();
    glLoadIdentity();
    glutSwapBuffers();
}
void passiveMotionFunc(int x, int y)
{
    // when mouse not clicked
    mouseX = float(x) / (m_viewport[2] / 1200.0) - 600.0; // converting
screen resolution to ortho 2d spec
    mouseY = -(float(y) / (m_viewport[3] / 700.0) - 350.0);



    // somethingMovedRecalculateLaserAngle();
    glutPostRedisplay();
```

```
}

void mouseClick(int buttonPressed, int state, int x, int y) {
    if (buttonPressed == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
mButtonPressed = true;
    else mButtonPressed = false;
    glutPostRedisplay();
}
void keyPressed(unsigned char key, int x, int y) {
    keyStates[key] = true;
    glutPostRedisplay();
}
void refresh() {
    glutPostRedisplay();
}
void keyReleased(unsigned char key, int x, int y) {
    keyStates[key] = false;
}
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(0, 0);
    glutInitWindowSize(1200, 600);
    glutCreateWindow("Space Shooter");
    init();
    glutIdleFunc(refresh);
    glutKeyboardFunc(keyPressed);
    glutKeyboardUpFunc(keyReleased);
    glutMouseFunc(mouseClick);
    glutPassiveMotionFunc(passiveMotionFunc);
    glGetIntegerv(GL_VIEWPORT, m_viewport);
    glutDisplayFunc(display);
    glutMainLoop();
}
```

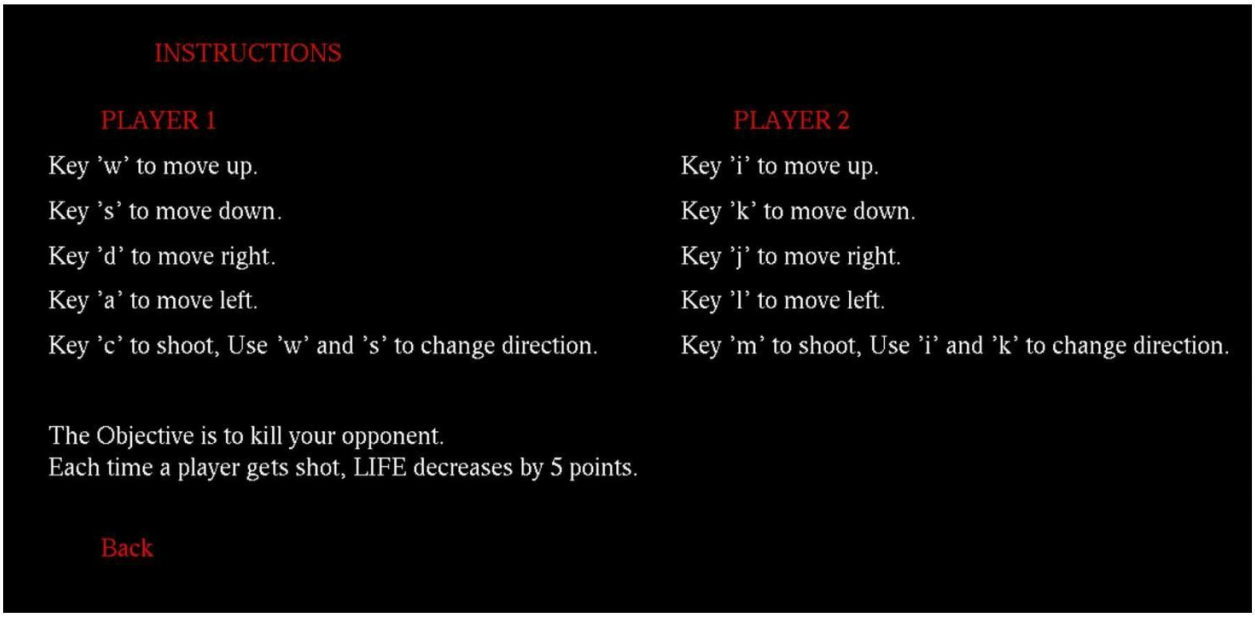
OUTPUT SCREENSHOTS



The intro screen



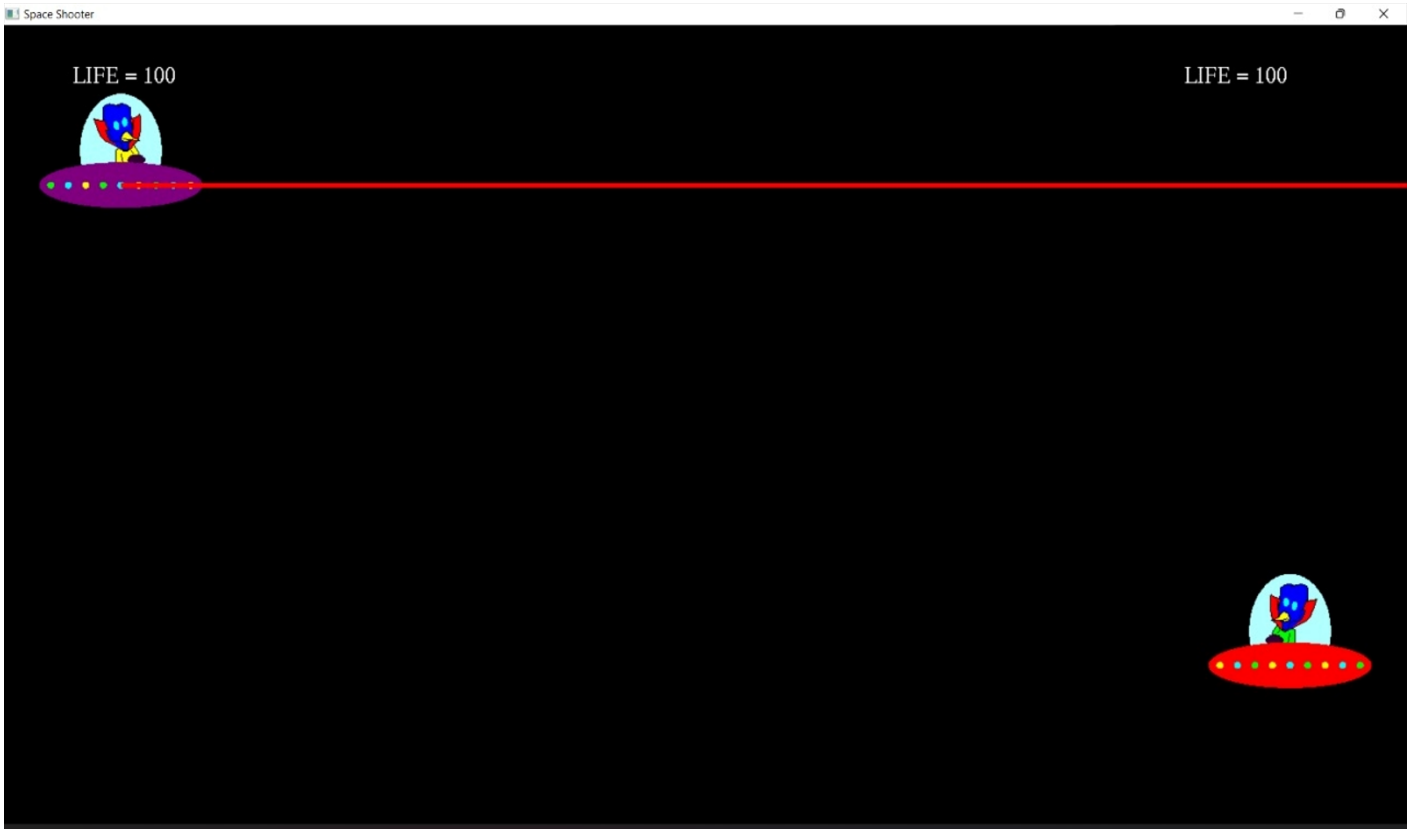
The menu screen

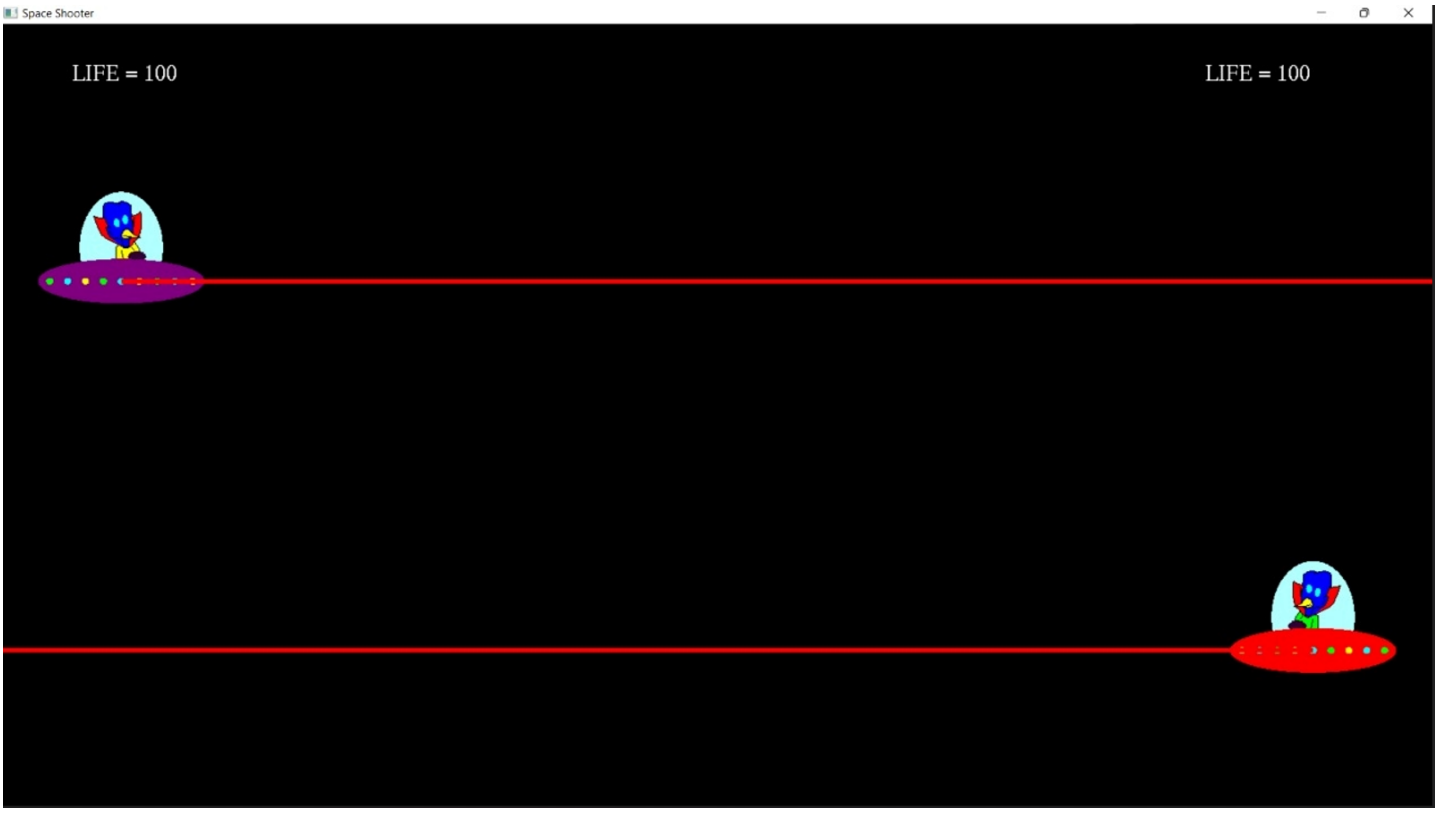


The instructions screen

The game screen

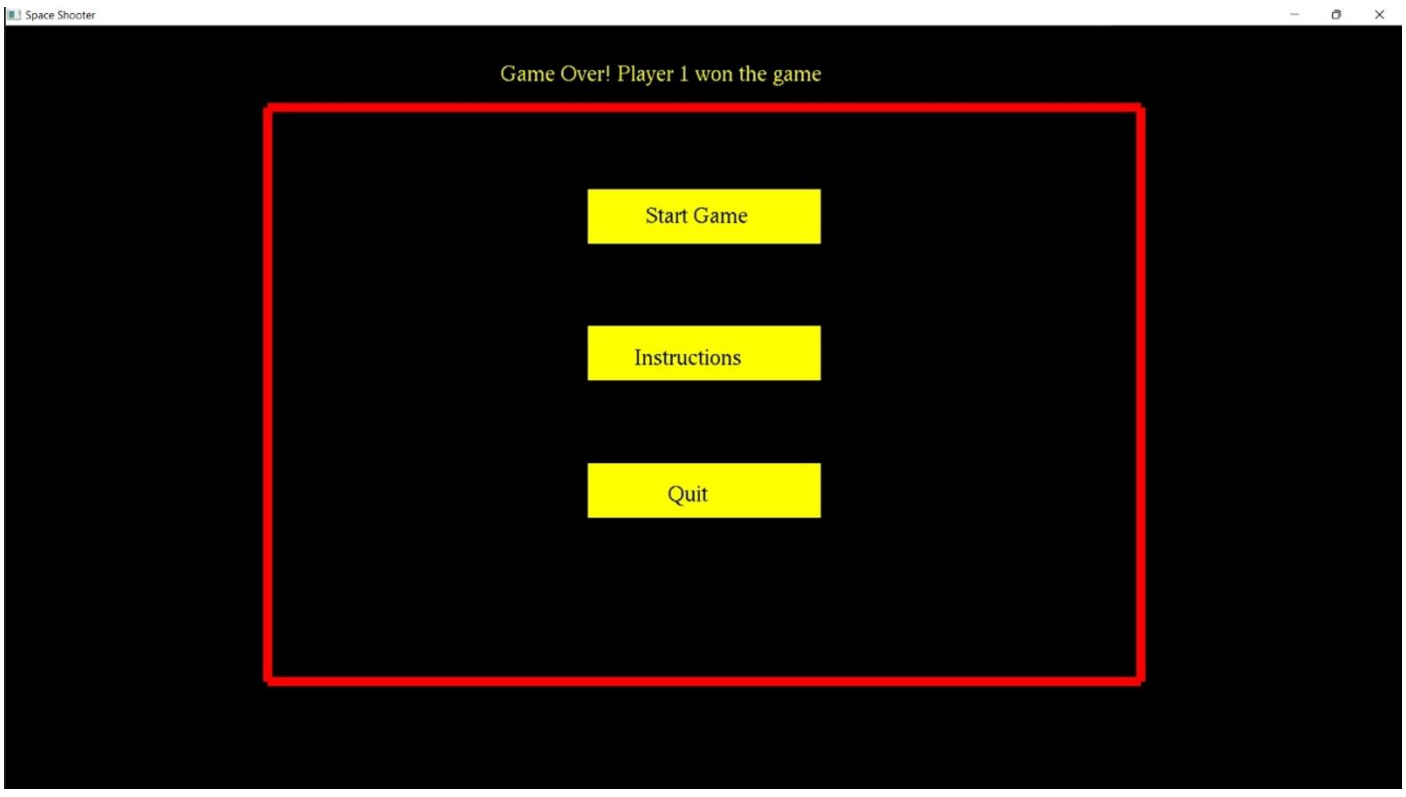

Space Shooter
LIFE = 100
LIFE = 100


Space Shooter
LIFE = 100
LIFE = 100


Space Shooter
LIFE = 100
LIFE = 100


Space Shooter
LIFE = 100
LIFE = 65


Space Shooter
LIFE = 100
LIFE = 60


Space Shooter
Game Over! Player 1 won the game
Start Game
Instructions
Quit