Computer Graphics (UCS505)

Project on

Shooting Game

Submitted By

Rosy Dash	101903078
Nikita	101903081
Ananya Goel	101903073

Group No. 1

B.E. Third Year – COE

Submitted To:

Dr. Anupam Garg

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

Table of Contents

INTRODUCTION

Our beautiful galaxy is being attacked from enemies through stones of three types, control your spaceship from them and protect our home. Move your fighter left or right, up or down in the sky to avoid colliding with stones, Exercise controlling to know the regular about the bullet path. The last hope is in your hand.

In our project we will be placed in a space setting. The player can control a spaceship. The player can move up, down, left and right. There are three types of enemies (or stones) like ships, asteroids and boss from which the player has to protect the alien and the spaceship. In order to protect, there is a gun type placed on the center of the screen and we have to dodge the objects and shoot them. The gun can be moved in four directions (up, down, left and right) and shoot is done by mouse click.If the player successfully shoots the stones the score increases by 1. If the player scores 100 points, then the player gets proceeded to next level with a bonus of 50 points increment in score. But if the stone collides with the spaceship, the life of the player decreases by 10. Initial life given is 100 to the player. If the life becomes zero, the game ends. The main objective is to score the maximum points.

.

WORKING OF THE PROJECT

On starting the game,

1) A blue screen with three options will appear. The three options are:
   a) Start game: By clicking on this option, a new screen with instructions on it appears and finally the game starts
   b) Instructions: Using this option, the player can read all the instructions required to play the game successfully prior to start of the game.
   c) Quit: By clicking on this option, the player can leave the game.

2) The player has to choose any one of the three options. By clicking on the instructions, a screen appears which states that :
   i) Key 'w' to move up
   ii) Key 's' to move down
   iii) Key 'd' to move right
   iv) Key 'a' to move left
   v) Left mouse click to shoot laser
   vi) You get 1 point for shooting each object and 50 points for completing each level
   vii) Objective is to score maximum score points

3) After reading the instructions, the player has to click on start game in order to play the game. After clicking, the game finally starts. The alien is on a spaceship getting frequent attacks from the stones. There are three different types of stones from which the alien and spaceship needs to be protected. The spaceship moves using the keys

   - w(For moving up)
   - a(For moving left)
   - s(For moving down)
   - d(For moving right)

The alien sitting in a spaceship could shoot a laser beam on the three types of stones by clicking left mouse click anddestroy them, earning one point each for destroying a stone. After earning every score, the spaceship speed gets increased. On hitting a stone, life gets reduced by 10 and you could see howmuch life is present at the top right corner of screen and even a health bar is present at the top. The initial life score given is 100. Further on scoring scores like 10,20… some lines appear in the background. Moreover, scores could be seenat the top. On shooting every 100 stones, the level goes on increasing. The status of level is also displayed on the top of the game screen.

COMPUTER GRAPHICS CONCEPTS USED

In the shooting game project, the concepts used are:

Polygon Drawing: -

- **Lines: -** They are drawn using the library by mentioning starting and ending points (x, y coordinates) in the space.
- **Circle -** The polygon constructed about the centre with a particular radius using a loop over the angle from 0 to 360 degree.
- **Sphere -** The polygon using glutSolidSphere library.
- **Through Points -** Polygon is constructed by looping over different points in the space. Using LINE_STRIP and LINE_LOOP commands.

2D and 3D transformations: -

To rotate the stones, increase the decrease the size of stones and create various shapes the following 2D and 3D transformations are used in this project.

- **Translation -** Movement of object on screen/in space without deformation. We translate a point by adding to x, y and z coordinates.
- **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.

By using glcolor3f, we could fill colors inside any polygon by filling the RGB values to make the project visually appealing.

USER DEFINED FUNCTIONS

1. **void displayRasterText1(float x, float y, float z, const char* stringToDisplay):** To display text with GLUT_BITMAP_HELVETICA_12 style and size.

2. **void displayRasterText(float x, float y, float z, const char* stringToDisplay):** To display text with GLUT_BITMAP_TIMES_ROMAN_24 style and size.
3. **void circle(int x, int y):** To draw the circle with center x and y..
4. **void initializeStoneArray():** To initialize the random stone array with x and y coordinates and store it in the array xstone and ystone.
5. **void SetDisplayMode(int modeToDisplay):** To set the display background color for different displays.
6. **void DrawAlienBody():** To draw the alien body in the spaceship using the points stored in the array with x,y coordinates.
7. **void DrawAlienEyes():** To draw the eyes using the sphere and transformations.
8. **void DrawAlienFace():** To draw the face using the circle and line for the face.
9. **void DrawAlien():** To call all the above functions to draw the alien.
10. **void DrawSpaceshipDoom():** To draw the elliptical body of the spaceship.
11. **void DrawSpaceShipLaser():** To draw the red laser on the spaceship for the beginning of the beam to destroy stones.
12. **void DrawSpaceshipBody():** To draw the spaceship body with sphere and transformations.
13. **void DrawSteeringWheel():** To draw the steering wheel of the spaceship.
14. **void DrawLaserBeam():** Draw the beam from laser to point clicked by the left button of the mouse.
15. **void DrawStone(int StoneIndex):** Draw and choose the different stones designs by random function and draw them on the screen.
16. **void SpaceshipCreate():** Draw the spaceship on the coordinates by just pressing direction keys and decrease the health of spaceship if they collide by calling checkIfSpaceShipIsSafe().
17. **bool checkIfSpaceShipIsSafe():** Check whether the spaceship collides with the stone coving towards the spaceship by comparing coordinates.

18. **void DisplayHealthBar():** Display health bar on top and print score, life and level under it.
19. **void startScreenDisplay():** display the start screen with 3 colored buttons start, instructions and quit and examine which button is being pressed.
20. **void StoneGenerate():** if the player is alive, this function would generate stones heading towards the spaceship.
21. **void GameScreenDisplay():** if player is alive, get some distraction by displaying white polygons from upper to bottom and create spaceship and generate stones.
22. **void GameOverScreen():** If the player dies, this function would take him to a different screen showing options like startgame, quit.
23. **void backButton():** It is the button which will take the player from a certain display window to the previous display
24. **void somethingMovedRecalculateLaserAngle():** This will find the angle at which the laser needs to shoot.
25. **void keys(unsigned char key, int x, int y):** The keyboard keys that are used to move the spaceship.
26. **void mouseClick(int buttonPressed, int state, int x, int y) :** This function will find if the mouse is clicked and throw the laser beam to the specified direction.
27. **void passiveMotionFunc(int x, int y):** It will find the x and y coordinates of the mouse at a given point.
28. **void idleCallBack():** If no mouse or key is detected, even then the display function will be called using this function.
29. **void InstructionsScreenDisplay():** This function contains all the instructions to play the game.
30. **void display():** It contains the overall display parts used during playing of the

CODE

```
#include <stdio.h>     // standard input output ,deals input output related functions

#include <string.h>    //helps us to work with strings through different functions

#include <stdlib.h>    //standard library works with functions such as memory allocation,
conversion, process control etc.

#include <gl/glut.h>   //library of utilities for open-gl programs

#include <math.h>      //header file designed for basic mathematical operation.

#include <fstream>     //deals with file operation

#define PI 3.14159     //assign PI value as 3.14159

#define GAME_SCREEN 0  //Constant to identify background color

#define MENU_SCREEN 4  //constant to identify menu screen color

#define MAX_STONES  100  //constant to identify maximum no of stones

#define MAX_STONE_TYPES 3  //constant to identify type of stones

#define stoneRotationSpeed 4   //constant to rotation speed of stone

#define SPACESHIP_SPEED 20       //constant to identify speed of spaceship.

int stoneTranslationSpeed = 5;    //says stone translation speed

GLint m_viewport[4];

GLint CI = 0;    // color indexes for spaceship (three colors in total)

int x, y;

int i;

int randomStoneIndices[100]; // No. of stones

int index;

int Score = 0;
```

```
int alienLife = 100;

int GameLvl = 1;

float mouseX, mouseY;                                   //Cursor coordinates;

float LaserAngle = 0, stoneAngle = 0, lineWidth = 1;

float xOne = 0, yOne = 0;                               //Spaceship coordinates

float xStone[MAX_STONES], yStone[MAX_STONES];//coordinates of stones

float xHealthBarStart = 1200;                           //Health bar starting coodinate

GLint stoneAlive[MAX_STONES];           //check to see if stone is killed


bool mButtonPressed = false, startGame = false, gameOver = false;           //boolean values to
check state of the game

bool startScreen = true, nextScreen = false, previousScreen = false;

bool gameQuit = false, instructionsGame = false, optionsGame = false;


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] = { {0,0}, {1,9}, {0,12}, {-15,10}, {-16,0},{0,0} };

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}
};
char highScore[100], ch;
void display();
void StoneGenerate();
void displayRasterText(float x, float y, float z, const char* stringToDisplay) {
        int length;
        glRasterPos3f(x, y, z);
        length = strlen(stringToDisplay);

        for (int i = 0; i < length; i++) {
                glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,
stringToDisplay[i]);
        }
}
void displayRasterText1(float x, float y, float z, const char* stringToDisplay) {
        int length;
        glRasterPos3f(x, y, z);
        length = strlen(stringToDisplay);

        for (int i = 0; i < length; i++) {
```

```
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, stringToDisplay[i]);
        }
}
void SetDisplayMode(int modeToDisplay) {
        switch (modeToDisplay) {
        case GAME_SCREEN: glClearColor(0, 0, 0, 1); break; //background
        case MENU_SCREEN: glClearColor(0, 1, 1, 1); break;

        }
}
void initializeStoneArray() {
        //random stones index

        for (int i = 0; i < MAX_STONES; i++) {
                randomStoneIndices[i] = rand() % MAX_STONE_TYPES;
                stoneAlive[i] = true;
        }

        xStone[0] = -(100 * MAX_STONES) - 600;          //START LINE for stone appearance
//y axis should not cross boundary // at a gap of 200 units from x // stones gap is shown with the
indices
```

```
for (int i = 0; i < MAX_STONES; i++) {                    //random appearance yIndex
for each stone
            yStone[i] = rand() % 600;
            if (int(yStone[i]) % 2)
                  yStone[i] *= -1;
            xStone[i + 1] = xStone[i] + 100;                    //xIndex of stone
aligned with 100 units gap
      }
}
void circle(int x, int y)          //circle generation code
{
      float th;
      glColor3f(0, 0, 1);
      glBegin(GL_POLYGON);
      for (int i = 0; i < 360; i++)
      {
            th = i * (3.1416 / 180);
            glVertex2f(x + 13 * cos(th), y + 13 * sin(th));
      }

      glEnd();
}
```

```
void DrawAlienBody()
{
	//glColor3f(0, 1, 0);				//BODY color
	//glBegin(GL_POLYGON);
	/*for (i = 0; i <= 5; i++)
		glVertex2fv(AlienBody[i]);*/
	//glEnd();
	glColor3f(0, 0, 0);			//BODY Outline
	glLineWidth(1);
	glBegin(GL_LINE_STRIP);
	for (i = 0; i <= 5; i++)
		glVertex2fv(AlienBody[i]);
	glEnd();
}
void DrawAlienFace()
{
	circle(-8, 24);		//aliens face outline
	glColor3f(0, 0, 0);
	glBegin(GL_LINES);
	glLineWidth(5);			//aliens mouth
	glVertex2f(-14, 19);
	glVertex2f(-2, 19);
	glEnd();
```

```
}

void DrawAlienEyes()
{

    glColor3f(0, 1, 1);
    glPushMatrix();          //push the current matrix into the stack, takes alien face circled outline and applies translation for eyes
    glRotated(-5, 0, 0, 1);
    glTranslated(-7, 27.5, 0);    //Left eye
    glScalef(2.5, 4, 0);
    glutSolidSphere(1, 20, 30);
    glPopMatrix();           //pops the current matrix from the stack
    glPushMatrix();
    glRotated(-1, 0, 0, 1);
    glTranslated(-13, 27.5, 0);                                    //Right eye
    glScalef(2.5, 4, 0);
    glutSolidSphere(1, 100, 100);  //renders a solid sphere
    glPopMatrix();
}
void DrawAlien()          //integrates alien body, face and eyes
{
```

```
	DrawAlienBody();
	DrawAlienFace();
	DrawAlienEyes();
}
void DrawSpaceshipBody()
{
	glColor3f(1, 0, 0);				//BASE
	glPushMatrix();		//pushes the current matrix
	glScalef(70, 20, 1);		//makes it larger
	glutSolidSphere(1, 50, 50);		//renders sphere
	glPopMatrix();

}

void DrawSpaceshipDoom()
{
	glColor4f(0.7, 1, 1, 0.0011);	//draws the blue doom
	glPushMatrix();
	glTranslated(0, 30, 0);
	glScalef(35, 50, 1);
	glutSolidSphere(1, 50, 50);
	glPopMatrix();
}
```

```
void DrawSpaceShipLaser() {

        glColor3f(1, 0,0);
        glPushMatrix();
        glBegin(GL_POLYGON);          //Lazer stem
        glVertex2f(-55, 10);
        glVertex2f(-55, 30);
        glVertex2f(-50, 30);
        glVertex2f(-50, 10);
        glEnd();

        //find mid point of top of lazer stem
        float midPoint = -(55 + 50) / 2.0;

        glBegin(GL_POLYGON);          //Lazer horizontal stem
        glVertex2f(midPoint + 10, 25);
        glVertex2f(midPoint + 10, 35);
        glVertex2f(midPoint - 10, 35);
        glVertex2f(midPoint - 10, 25);
        glEnd();


        glPopMatrix();
```

```
}
void DrawLaserBeam() {

        float xMid = -(55 + 50) / 2.0;
        float yMid = (25 + 35) / 2.0;

        float mouseXEnd = mouseX;
        float mouseYEnd = mouseY;
        glLineWidth(5);   //----Laser beam width

        glColor3f(1, 0, 0);
        glBegin(GL_LINES);                          //laser beam drawn
        glVertex2f(xMid, yMid);
        glVertex2f(mouseXEnd, mouseYEnd);
        glEnd();
        glLineWidth(1);
}
void DrawStone(int StoneIndex)
{
        glPushMatrix();                     //draw different types of stone and rotate them
        glLoadIdentity();
        switch (StoneIndex)                  //CHANGE INDEX VALUE FOR DIFFERENT
STONE VARIETY;
```

```
{
case 0:

        glTranslated(xStone[index], yStone[index], 0);
        glRotatef(stoneAngle, 0, 0, 1);

        glColor3f(0.4f, 0.0f, 0.0f);
        glScalef(35, 35, 1);
        glutSolidSphere(1, 9, 50);
        break;

case 1:
        glColor3f(1.0f, 0.8f, 0.8f);
        glTranslated(xStone[index], yStone[index], 0);
        glRotatef(stoneAngle, 0, 0, 1);

        glScalef(40, 30, 1);
        glutSolidSphere(1, 9, 50);
        break;

case 2:
        glColor3f(0.2f, 0.2f, 0.0f);
        glTranslated(xStone[index], yStone[index], 0);
```

```
            glRotatef(stoneAngle, 0, 0, 1);


            glScalef(60, 25, 1);
            glutSolidSphere(1, 9, 50);
            break;
        }
        glPopMatrix();
}
bool checkIfSpaceShipIsSafe()
{
        for (int i = 0; i < MAX_STONES; i++) {
            if (stoneAlive[i] & ((xOne >= (xStone[i] / 2 - 70) && xOne <= (xStone[i] / 2 +
70) && yOne >= (yStone[i] / 2 - 18) && yOne <= (yStone[i] / 2 + 53)) || (yOne <= (yStone[i] / 2
- 20) && yOne >= (yStone[i] / 2 - 90) && xOne >= (xStone[i] / 2 - 40) && xOne <= (xStone[i]
/ 2 + 40))))
            {
                stoneAlive[i] = 0;
                return false;
            }
        }
        return true;
}
void SpaceshipCreate() {
```

```
        glPushMatrix();
        glTranslated(xOne, yOne, 0);
        if (!checkIfSpaceShipIsSafe() && alienLife) {
                alienLife -= 10;
                xHealthBarStart -= 240; //changes
        }
        DrawSpaceshipDoom();
        glPushMatrix();
        glTranslated(4, 19, 0);
        DrawAlien();
        glPopMatrix();
        DrawSpaceshipBody();
        DrawSpaceShipLaser();
        if (mButtonPressed) {
                DrawLaserBeam();
        }
        glEnd();
        glPopMatrix();
}
void DisplayHealthBar() {

        glColor3f(1, 0, 0);
        glBegin(GL_POLYGON);
```

```
        glVertex2f(-xHealthBarStart, 700);
        glVertex2f(1200, 700);
        glVertex2f(1200, 670);
        glVertex2f(-xHealthBarStart, 670);
        glEnd();
        char temp[40];
        glColor3f(0, 0, 1);
        sprintf_s(temp, "SCORE = %d", Score);
        displayRasterText(-1100, 600, 0.4, temp);//<---display variable score ?
        sprintf_s(temp, "  LIFE = %d", alienLife);
        displayRasterText(800, 600, 0.4, temp);
        sprintf_s(temp, "  LEVEL : %d", GameLvl);
        displayRasterText(-100, 600, 0.4, temp);
        glColor3f(1, 0, 0);
}
void startScreenDisplay()
{
        glLineWidth(50);
        SetDisplayMode(MENU_SCREEN);

        glColor3f(0, 0, 0);
        glBegin(GL_LINE_LOOP);              //Border
        glVertex3f(-750, -500, 0.5);
```

```
glVertex3f(-750, 550, 0.5);
glVertex3f(750, 550, 0.5);
glVertex3f(750, -500, 0.5);
glEnd();

glLineWidth(1);

glColor3f(1, 1, 0);
glBegin(GL_POLYGON);                    //START GAME PLOYGON
glVertex3f(-200, 300, 0.5);
glVertex3f(-200, 400, 0.5);
glVertex3f(200, 400, 0.5);
glVertex3f(200, 300, 0.5);
glEnd();

glBegin(GL_POLYGON);                    //INSTRUCTIONS POLYGON
glVertex3f(-200, 50, 0.5);
glVertex3f(-200, 150, 0.5);
glVertex3f(200, 150, 0.5);
glVertex3f(200, 50, 0.5);
glEnd();

glBegin(GL_POLYGON);                    //QUIT POLYGON
```

```
        glVertex3f(-200, -200, 0.5);
        glVertex3f(-200, -100, 0.5);
        glVertex3f(200, -100, 0.5);
        glVertex3f(200, -200, 0.5);
        glEnd();
        glColor3f(0, 0, 0);
        displayRasterText1(-120, -380, 0, "Developed By: Rosy Dash     Nikita     Ananya Goel");
        if (mouseX >= -100 && mouseX <= 100 && mouseY >= 150 && mouseY <= 200) {
                glColor3f(0, 0, 1);
                if (mButtonPressed) {
                        startGame = true;          //changes mouse pointer to blue when mouse hovers there
                        gameOver = false;
                        mButtonPressed = false;
                }
        }
        else
                glColor3f(0, 0, 0);

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

        if (mouseX >= -100 && mouseX <= 100 && mouseY >= 30 && mouseY <= 80) {
```

```
            glColor3f(0, 0, 1);
            if (mButtonPressed) {
                instructionsGame = true;
                mButtonPressed = false;
            }
        }
        else
            glColor3f(0, 0, 0);
        displayRasterText(-120, 80, 0.4, "Instructions");

        if (mouseX >= -100 && mouseX <= 100 && mouseY >= -90 && mouseY <= -40) {
            glColor3f(0, 0, 1);
            if (mButtonPressed) {
                gameQuit = true;
                mButtonPressed = false;
            }
        }
        else
            glColor3f(0, 0, 0);
        displayRasterText(-100, -170, 0.4, "    Quit");

}
void GameScreenDisplay()
```

```
{
	SetDisplayMode(GAME_SCREEN);
	DisplayHealthBar();
	glScalef(2, 2, 0);
	if (alienLife) {
		SpaceshipCreate();
	}
	else {
		gameOver = true;
		instructionsGame = false;
		startScreen = false;
	}
	StoneGenerate();

}

void GameOverScreen()
{
	SetDisplayMode(MENU_SCREEN);
	glColor3f(0, 0, 0);
	glLineWidth(50);
	glBegin(GL_LINE_LOOP);		//Border
	glVertex3f(-650, -500, 0.5);
```

```
glVertex3f(-650, 520, 0.5);
glVertex3f(650, 520, 0.5);
glVertex3f(650, -500, 0.5);
glEnd();

glLineWidth(1);
stoneTranslationSpeed = 5;
glColor3f(1, 0, 0);
glBegin(GL_POLYGON);                    //GAME OVER
glVertex3f(-550, 810, 0.5);
glVertex3f(-550, 610, 0.5);
glVertex3f(550, 610, 0.5);
glVertex3f(550, 810, 0.5);
glEnd();

glColor3f(1, 1, 0);
glBegin(GL_POLYGON);                    //RESTART POLYGON
glVertex3f(-200, 50, 0.5);
glVertex3f(-200, 150, 0.5);
glVertex3f(200, 150, 0.5);
glVertex3f(200, 50, 0.5);
glEnd();
```

```
        glBegin(GL_POLYGON);                                //QUIT POLYGON
        glVertex3f(-200, -200, 0.5);
        glVertex3f(-200, -100, 0.5);
        glVertex3f(200, -100, 0.5);
        glVertex3f(200, -200, 0.5);
        glEnd();

        glColor3f(0, 0, 0);
        displayRasterText(-300, 640, 0.4, "    G A M E    O V E R ! ! !");
        glColor3f(0, 0, 0);
        char temp[40];
        sprintf_s(temp, "Score : %d", Score);
        displayRasterText(-100, 340, 0.4, temp);

        if (mouseX >= -100 && mouseX <= 100 && mouseY >= 25 && mouseY <= 75) {
                glColor3f(0, 0, 1);
                if (mButtonPressed) {                                   //Reset game default
values
                        startGame = true;
                        gameOver = false;
                        mButtonPressed = false;
                        initializeStoneArray();
                        alienLife = 100;
```

```
                xHealthBarStart = 1200;
                Score = 0;
                GameLvl = 1;
                GameScreenDisplay();
            }
        }
        else
            glColor3f(0, 0, 0);
        displayRasterText(-70, 80, 0.4, "Restart");

        if (mouseX >= -100 && mouseX <= 100 && mouseY >= -100 && mouseY <= -50) {
            glColor3f(0, 0, 1);
            if (mButtonPressed) {
                exit(0);
                mButtonPressed = false;
            }
        }
        else
            glColor3f(0, 0, 0);
        displayRasterText(-100, -170, 0.4, "    Quit");

}
void StoneGenerate() {
```

```
// if (xStone[0] >= 1200)


        if (Score == 100) {      //If the last screen hits the end of screen then go to Nxt lvl
        char temp[40];
                GameLvl++;
                stoneTranslationSpeed += 3;
                Score += 50;
                sprintf_s(temp, "  LEVEL : %d", GameLvl);
           displayRasterText(-100, 600, 0.4, temp);
        }


        for (int i = 0; i < MAX_STONES; i++) {
                index = i;
//checks if stone is not hit or hit by laser beam
                if (mouseX <= (xStone[i] / 2 + 30) && mouseX >= (xStone[i] / 2 - 30) &&
mouseY >= (yStone[i] / 2 - 30) && mouseY <= (yStone[i] / 2 + 30) && mButtonPressed) {
                        if (stoneAlive[i]) {   // IF ALIVE KILL STONE
                                stoneAlive[i] = 0;
                                Score++;
                                if (Score % 1 == 0) {         //speed increase with each score
                                        stoneTranslationSpeed += 1;                    //<----------
Rate of increase of game speed
                                }
```

```
                }
            }
            xStone[i] += stoneTranslationSpeed;
            if (stoneAlive[i])          //stone alive
                DrawStone(randomStoneIndices[i]);
        }
        stoneAngle += stoneRotationSpeed;           //maintained for stone rotation
        if (stoneAngle > 360) stoneAngle = 0;
}
void backButton() {
        glColor3f(1, 1, 0);
        glBegin(GL_POLYGON);                        //START GAME PLOYGON
        glVertex3f(-1010, -500, 0.5);
        glVertex3f(-1010, -570, 0.5);
        glVertex3f(-890, -570, 0.5);
        glVertex3f(-890, -500, 0.5);
        glEnd();
        if (mouseX <= -450 && mouseX >= -500 && mouseY >= -275 && mouseY <= -250) {
//if mouse clicks on back button in instruction screen the it goes to start screen
            glColor3f(0, 0, 1);
            if (mButtonPressed) {
                mButtonPressed = false;
                instructionsGame = false;
```

```
                    startScreenDisplay();
                }
        }
        else glColor3f(0, 0, 0);

        displayRasterText(-1000, -550, 0, "Back");

}
void InstructionsScreenDisplay()
{

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        SetDisplayMode(MENU_SCREEN);
        glColor3f(0, 0, 0);
        displayRasterText(-900, 400, 0.4, "-> Press Key 'w' to move up.");
        displayRasterText(-900, 300, 0.4, "-> Press Key 's' to move down.");
        displayRasterText(-900, 200, 0.4, "-> Press Key 'd' to move right.");
        displayRasterText(-900, 100, 0.4, "-> Press Key 'a' to move left.");
        displayRasterText(-900, 0.0, 0.4, "-> Left mouse click to shoot laser");
        displayRasterText(-900, -200, 0.4, "-> You Get 1 point for shooting each object and 50 points for completing each level ");
        displayRasterText(-900, -270, 0.4, "-> The Objective is to score maximum points");
        backButton();
```

```
	if (previousScreen)
		nextScreen = false, previousScreen = false; //as set by backButton()
}
void display() {

	glClear(GL_COLOR_BUFFER_BIT);	//buffer selection, Indicates the buffers currently enabled for color writing.
	glViewport(0,0,1200, 700);		//0,0 is the bottom left coordinate of our window
//width,height

	if (startGame && !gameOver)		//if the game starts then the start screen appears
		GameScreenDisplay();

	else if (instructionsGame)		//instruction screen appears
		InstructionsScreenDisplay();

	else if (gameOver)			//if the game is over game over screen appears
		GameOverScreen();

	//Make spaceship bigger
	else if (startScreen) {

		startScreenDisplay();
```

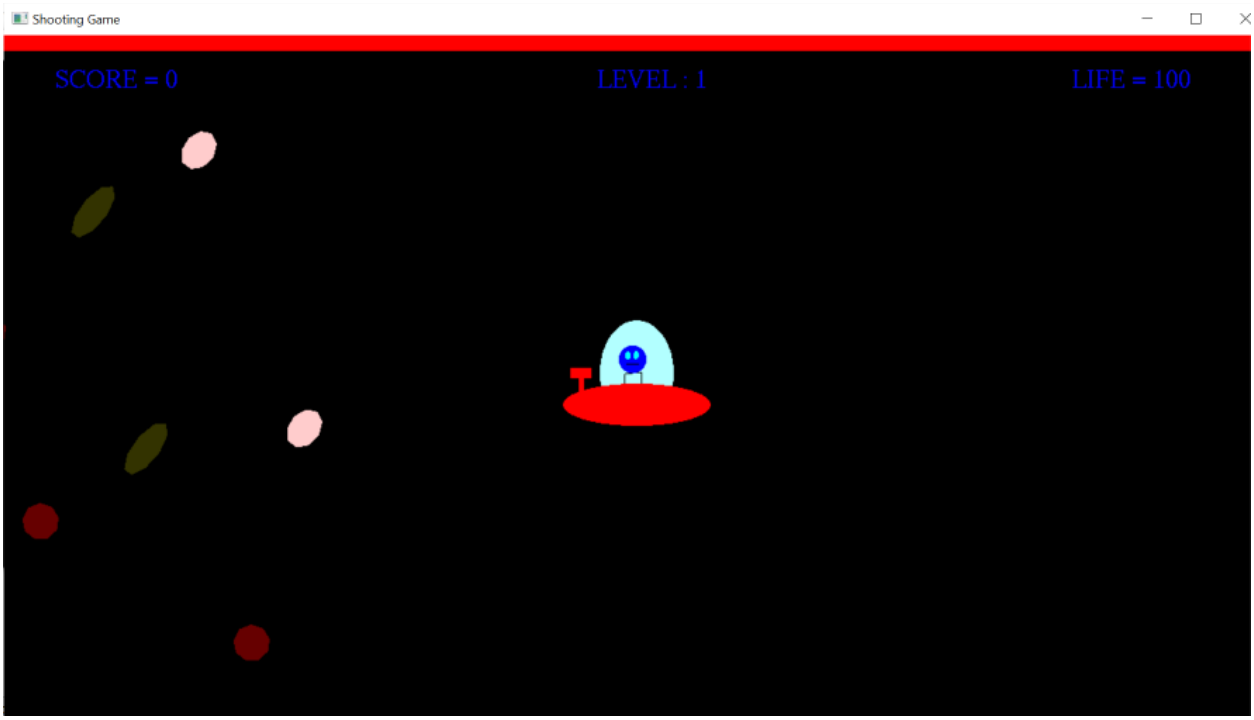
```
        if (gameQuit || startGame  || instructionsGame) {
                //startScreen = false;

                if (startGame) {
                        SetDisplayMode(GAME_SCREEN);
                        startScreen = false;

                }
                else if (gameQuit)
                        exit(0);

        }
        else if (instructionsGame) {
                SetDisplayMode(GAME_SCREEN);
                InstructionsScreenDisplay();
        }
}

//Reset Scaling values
glScalef(1 / 2, 1 / 2, 0);
glFlush();
glLoadIdentity();
```

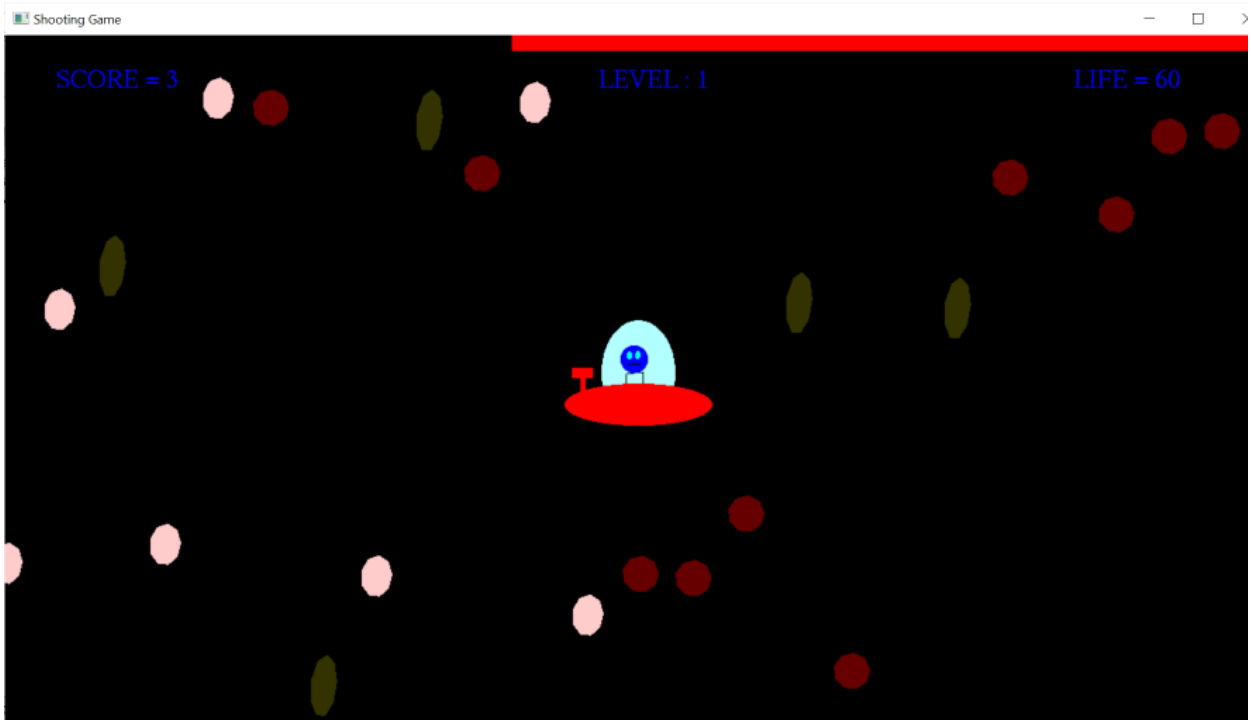
```
	glutSwapBuffers();		//for multiple screen display to swap from one screen to another without any difficulty
}

void keys(unsigned char key, int x, int y)
{
	if (key == 'd') xOne += SPACESHIP_SPEED;
	if (key == 'a') xOne -= SPACESHIP_SPEED;
	if (key == 'w') { yOne += SPACESHIP_SPEED; }
	if (key == 's') { yOne -= SPACESHIP_SPEED; }

	display();

}
void myinit()
{
	glClearColor(0, 0, 0, 0);		//set the background color
	glMatrixMode(GL_PROJECTION);	//choose the current matrix and projection for viewing the projection
	glLoadIdentity();			//replaces the current matrix with identity matrix , eacj time we load a screen the matrix is reset to identity.

	gluOrtho2D(-1200, 1200, -700, 700);			//<-----CHANGE THIS TO GET EXTRA SPACE//sets a 2d viewing region (l,r,t,b)
```

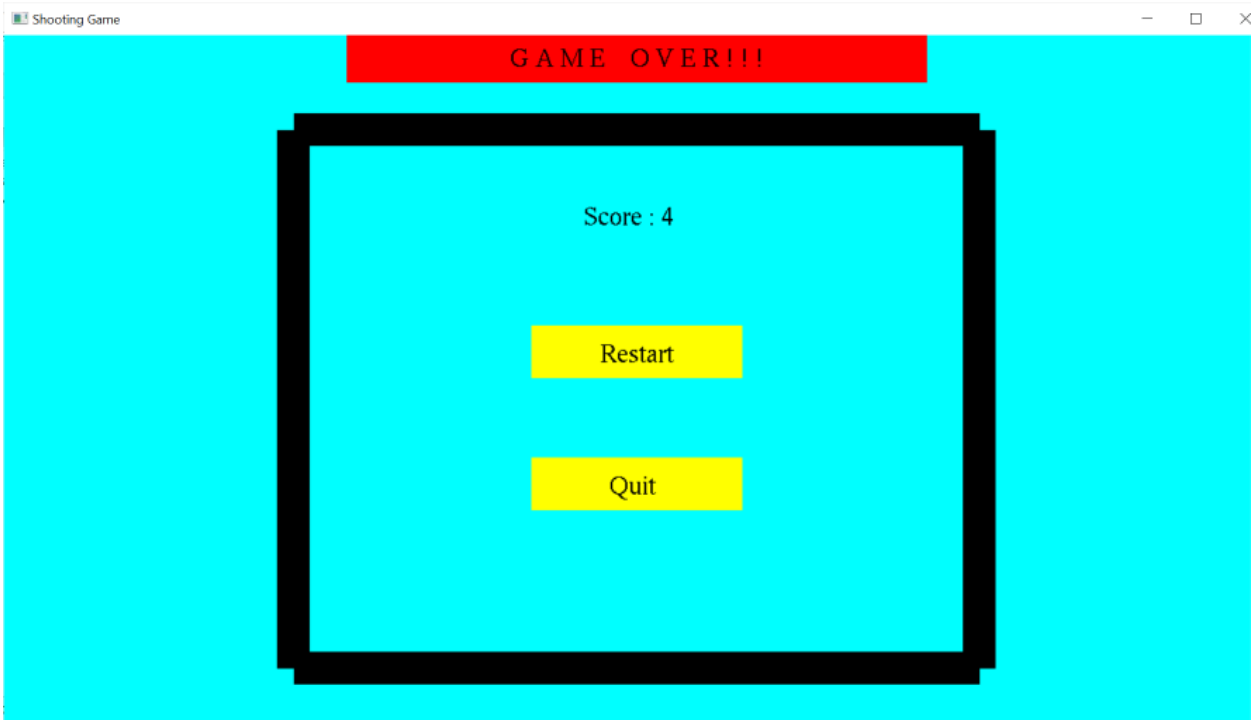
```
	glMatrixMode(GL_MODELVIEW);		//used for model setup
}
void passiveMotionFunc(int x, int y) {

	//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);
	display();
}
void mouseClick(int buttonPressed, int state, int x, int y)
{

	if (buttonPressed == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
		mButtonPressed = true;
	else
		mButtonPressed = false;
	display();
}


int main(int argc, char** argv) {
```

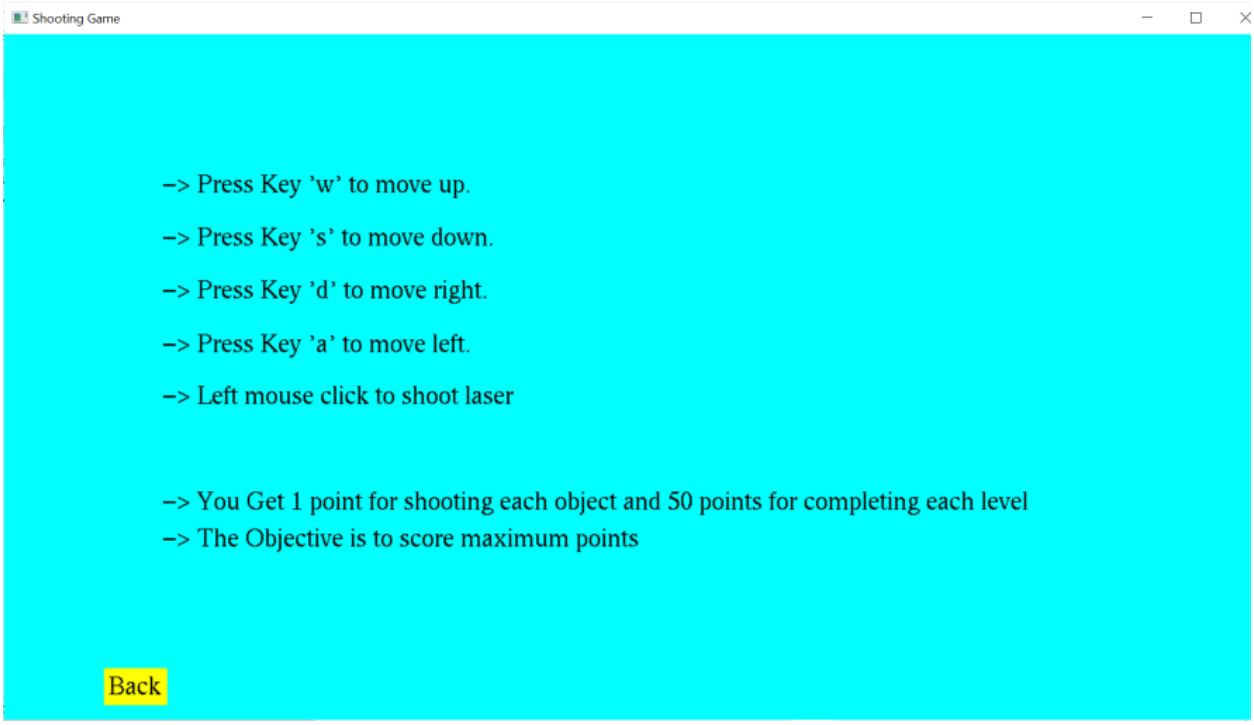
```
    glutInit(&argc, argv);            //initialize glut library
    glutInitWindowSize(1200, 700);    //window size
    glutInitWindowPosition(90, 0);    //window position
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);      //sets initial display mode ,
allows for display on double buffer window
    glutCreateWindow("Shooting Game");        //name of window
    glutDisplayFunc(display);          //execution of functions
    glutKeyboardFunc(keys);            //sets the keyboard callback for the current window
    glutPassiveMotionFunc(passiveMotionFunc);        //to track the mouse
    glutMouseFunc(mouseClick);          //on mouse click
    glGetIntegerv(GL_VIEWPORT, m_viewport);         //returns coordinates of viewport
    myinit();
    SetDisplayMode(GAME_SCREEN);       //displays the game screen
    initializeStoneArray();            //displays the stone
    glutMainLoop();
}
```

OUTPUT SCREENSHOTS




Shooting Game
SCORE = 3
LEVEL : 1
LIFE = 60


Shooting Game
GAME OVER!!!
Score : 4
Restart
Quit


Shooting Game
–> Press Key 'w' to move up.
–> Press Key 's' to move down.
–> Press Key 'd' to move right.
–> Press Key 'a' to move left.
–> Left mouse click to shoot laser
–> You Get 1 point for shooting each object and 50 points for completing each level
–> The Objective is to score maximum points
Back