Computer Graphics (UCS505)

Project on

Windy Seascape

Submitted By

Aryaman Kalia 102003099

CO6

B.E. Third Year – COE

Submitted To:

Dr Harpreet Singh

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

Table of Contents

Introduction To Project

The scene consists of a landscape with three hills, a lake, and a cloudy sky. There are three windmills on the hills, and a ship is sailing in the lake. The ship and the windmills are animated, giving the scene a dynamic look. The windmills rotate around their axis, and the ship moves from left to right, simulating its journey on the water.

The project was developed using OpenGL, a widely used open-source graphics library that provides a set of functions for creating interactive 2D and 3D graphics applications. OpenGL supports various primitives like points, lines, and polygons, and provides functions for transformations like translation, rotation, and scaling. The library also supports animations through the use of timers.

The main objective of this project is to provide an insight into the basics of computer graphics and to demonstrate the use of OpenGL functions and features to create a simple scene. The project is suitable for beginners who want to learn computer graphics and OpenGL programming.

The following report presents the lab file for the project, which includes the description of the scene, the code used to create it, and an explanation of the functions and transformations used in the project.

Computer Graphics concepts used

This project is a simple animation implemented using OpenGL, a popular graphics API used for rendering 2D and 3D graphics. It uses several fundamental computer graphics concepts and techniques, including:

Translation: Translation refers to the movement of objects in a two or three-dimensional space. In computer graphics, translation is accomplished by manipulating the coordinates of the vertices of an object. Translation can be done along the x, y, or z-axis of the coordinate system, or a combination of these axes.

Rotation: Rotation, on the other hand, involves changing the orientation of an object around an axis. In computer graphics, objects are rotated by specifying the angle of rotation and the axis of rotation. Rotation can be done along the x, y, or z-axis of the coordinate system, or a combination of these axes.

Geometric primitives: OpenGL provides several primitives to draw basic shapes, such as points, lines, triangles, and polygons. The project uses several polygons to draw the background and the ship, and triangles to draw the windmill blades.

Coordinate system and transformations: OpenGL uses a 3D coordinate system to represent points in space. The project uses 2D transformations to position, scale, and rotate objects in the scene. The glTranslatef and glRotated functions are used to translate and rotate the windmill blades, while glPushMatrix and glPopMatrix are used to save and restore the state of the transformation stack.

Colors and shading: OpenGL allows setting the color of objects using the RGB color model, and provides several shading models to control how light interacts with objects. The project uses glColor3f to set the color of the objects, and the glClear function to set the background color.

Animation: OpenGL allows creating animations by updating the scene and redrawing it at a fixed frame rate. The project uses the glutTimerFunc function to update the frame number and trigger a redisplay every 30 milliseconds.

Some of the inbuilt OpenGL functions used in the project include:

glBegin and glEnd: These functions define a block of code that draws a set of primitives. The project uses these functions to draw polygons and triangles.

glVertex2f: This function specifies a vertex of a primitive using its 2D coordinates. The project uses this function to define the vertices of the polygons and triangles.

glColor3f: This function sets the current color to a given RGB value. The project uses this function to set the color of the ship, windmill, and background.

glClear: This function clears the color buffer, which holds the current color of each pixel in the screen. The project uses this function to set the background color.

glTranslatef and glRotated: These functions apply translations and rotations to the current transformation matrix. The project uses these functions to animate the windmill blades.

glPushMatrix and glPopMatrix: These functions save and restore the current transformation matrix to a stack. The project uses these functions to apply transformations to nested objects, such as the windmills.

In summary, this project demonstrates several core computer graphics concepts and techniques, and uses several inbuilt OpenGL functions to create a simple but visually appealing animation.

User Defined Functions

The **drawShip()** function draws a ship using OpenGL primitives, with a hull, cabin, and flag.

The **drawWindmill()** function draws a single windmill using OpenGL primitives. It consists of a rectangular base and three rotating blades.

The **display()** function is called each time the screen needs to be updated. It draws the background and calls the drawWindmill() and drawShip() functions to draw the windmills and ship.

The **doFrame()** function is called repeatedly to update the frame number and redraw the screen.

The **init()** function is called once at the beginning of the program to set up the OpenGL environment.

The **main()** function initializes GLUT and registers callback functions to handle events ,then starts the main loop.

Code:

```
#include <GL/gl.h>
#include <GL/glut.h>
#include <math.h>
#include<windows.h>
int frameNumber = 0;

void drawShip() {
    // Draw the hull of the ship
    glColor3f(0.2f, 0.2f, 1.0f);
    glBegin(GL_POLYGON);
    glVertex2f(-0.4f, -0.4f);
    glVertex2f(0.4f, -0.4f);
    glVertex2f(0.6f, 0.0f);
    glVertex2f(0.4f, 0.4f);
    glVertex2f(-0.4f, 0.4f);
    glEnd();

    // Draw the cabin of the ship
    glColor3f(0.9f, 0.9f, 0.9f);
    glBegin(GL_POLYGON);
    glVertex2f(-0.2f, 0.0f);
    glVertex2f(0.2f, 0.0f);
    glVertex2f(0.2f, 0.3f);
    glVertex2f(-0.2f, 0.3f);
    glEnd();

    // Draw the flag of the ship
    glColor3f(1.0f, 0.0f, 0.0f);
    glBegin(GL_POLYGON);
    glVertex2f(0.0f, 0.3f);
    glVertex2f(0.0f, 0.5f);
    glVertex2f(-0.1f, 0.45f);
    glVertex2f(0.0f, 0.4f);
    glEnd();
}

void drawWindmill()
{
    int i;
    glColor3f(0.9f, 0.9f, 0.9f);

    glBegin(GL_POLYGON);
    glVertex2f(-0.05f, 0);
    glVertex2f(0.05f, 0);
    glVertex2f(0.05f, 3);
    glVertex2f(-0.05f, 3);
    glEnd();
    glTranslatef(0, 3, 0);
    glRotated(frameNumber * (180.0/43), 0, 0, 1);

    glColor3f(1.0f, 1.0f, 1.0f);

    for (i = 0; i < 3; i++)
    {
        glRotated(120, 0, 0, 1);
        glBegin(GL_POLYGON);
        glVertex2f(0,0);
        glVertex2f(0.5f, 0.1f);
        glVertex2f(1.5f,0);
        glVertex2f(0.5f, -0.1f);
        glEnd();
    }
}
void display()
{

    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(0, 0.6f, 0.2f);
    glBegin(GL_POLYGON);
    glVertex2f(-3,-1);
    glVertex2f(1.5f,1.65f);
    glVertex2f(5,-1);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex2f(-3,-1);
    glVertex2f(3,2.1f);
    glVertex2f(7,-1);
    glEnd();

    glBegin(GL_POLYGON);
    glVertex2f(0,-1);
    glVertex2f(6,1.2f);
    glVertex2f(20,-1);
```

```
 glEnd();
 glPushMatrix();
 glTranslated(1.25,1,0);
 glScaled(0.6,0.6,1);
 drawWindmill();
 glPopMatrix();

 glPushMatrix();
 glTranslated(2.7,1.6,0);
 glScaled(0.4,0.4,1);
 drawWindmill();
 glPopMatrix();
 glPushMatrix();
 glPushMatrix();
 glTranslated(4.2,0.8,0);
 glScaled(0.7,0.7,1);
 drawWindmill();
 glPopMatrix();
 glPushMatrix();
 glBegin(GL_POLYGON);
 glColor3f(0.3, 0.25f, 0.7f);
 glVertex2f(0,-1);
 glVertex2f(7,-1);
 glVertex2f(7,0.1f);
 glVertex2f(0,0.1f);
 glEnd();
 glPopMatrix();
 glPushMatrix();
 glTranslatef(((GLfloat)frameNumber/500),0,0);
/* glColor3f(0, 0, 0);
 glBegin(GL_POLYGON);
 glVertex2f(0,0);
 glVertex2f(1,0);
 glVertex2f(1,1);
 glVertex2f(0,1);
 glEnd();*/
 drawShip();
 glPopMatrix();
 glutSwapBuffers();
```

```
}
void doFrame(int v)
{
    frameNumber++;
    glutPostRedisplay();
    glutTimerFunc(30,doFrame,0);
}
void init()
{
    glClearColor(0.5f, .8f, 1.0f,1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0, 7, -1, 4, -1, 1);
    glMatrixMode(GL_MODELVIEW);
}


int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE);
    glutInitWindowSize(800,500);
    glutInitWindowPosition(00,00);
    glutCreateWindow("Windmill Animation");
    init();
    glutDisplayFunc(display);
    glutTimerFunc(200,doFrame,0);
    glutMainLoop();
    return 0;
}
```

ScreenShots: