Lab Report 9

Name: **Arnob Das Shacha**

ID: **221-15-5277**

Section: 61_D1

Course Code: CSE 422

Experiment Name: Drawing different objects using different primitives, circle and 2D Transformation.

Submitted To:

Sayeda Tasnim Alvi

Senior Lecturer

Dept of CSE

Daffodil International University

Title

Drawing different objects using different primitives, circle and 2D Transformation.

Introduction

In this project, complete outdoor scenery has been created using various **OpenGL primitive shapes** such as rectangles, triangles, circles, and lines. The program also demonstrates the application of **2D Transformations** including translation and scaling to position and size different objects such as trees, the sun, flowers, and house components. Multiple helper functions were written to draw reusable objects like trees, flowers, circles, and mountains. This project shows how simple primitive shapes can be combined to form complex graphical scenes.

Contents

The following functions and shapes have been used in this project:

1. drawRectangle(x, y, w, h)

- Draws a quadrilateral using GL_POLYGON.
- Used for sky background, house body, windows, door, flower stem, and tree trunks.

2. drawTriangle(x1, y1, x2, y2, x3, y3)

- Uses GL_TRIANGLES to draw triangular shapes.
- Applied for mountains, roof of the house, and the leaves of the trees.

3. drawCircle(cx, cy, r)

- Approximates a circle using 360 small segments.
- Used for the sun and flower petals.

4. drawSun()

- Combines a circle with rotated line segments to create sun rays.
- Uses glPushMatrix, glRotatef, and glTranslatef for positioning and rotation.

5. drawTree(x, y) & drawBigTree(x, y, scale)

- Uses translation and scaling transformations.
- Each tree consists of a rectangle (trunk) and a triangle (leaves).

6. drawFlower(x, y)

- A flower composed of multiple circles arranged using translation.

7. 2D Transformations Used

- Translation: Positioning trees, flowers, and sun.
- Scaling: Creating larger trees using glScalef().
- Rotation: Used for rotating rays around the sun.

Code

```
#include <GL/glut.h>
#include <math.h>
void drawRectangle(float x, float y, float w, float h)
{
   glBegin(GL_POLYGON);
   glVertex2f(x,     y);
   glVertex2f(x + w, y);
   glVertex2f(x + w, y + h);
   glVertex2f(x,     y + h);
   glEnd();
}
void drawTriangle(float x1, float y1, float x2, float y2, float x3, float y3)
{
   glBegin(GL_TRIANGLES);
   glVertex2f(x1, y1);
   glVertex2f(x2, y2);
   glVertex2f(x3, y3);
   glEnd();
}
void drawCircle(float cx, float cy, float r)
{
   glBegin(GL_POLYGON);
```

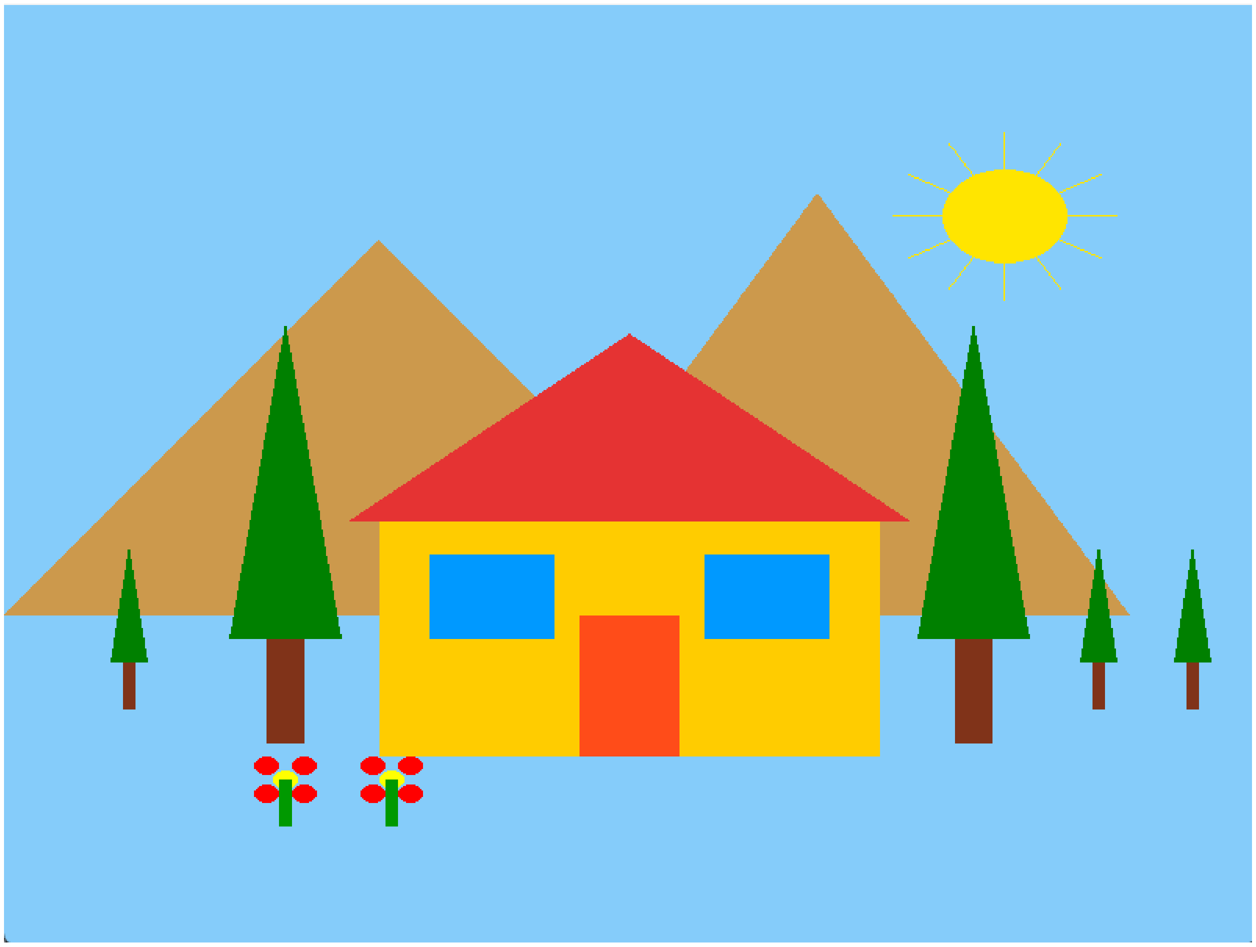
```
    for (int i = 0; i < 360; i++)
    {
        float theta = i * 3.1416 / 180;
        glVertex2f(cx + r * cos(theta), cy + r * sin(theta));
    }
    glEnd();
}

void drawSun()
{
    glColor3f(1, 1, 0);

    // Rays
    glColor3f(1, 0.9, 0);
    for (int i = 0; i < 12; i++)
    {
        glPushMatrix();
        glTranslatef(0.6, 0.55, 0);
        glRotatef(i * 30, 0, 0, 1);
        glBegin(GL_LINES);
        glVertex2f(0, 0);
        glVertex2f(0.18, 0);
        glEnd();
        glPopMatrix();
    }
    drawCircle(0.6, 0.55, 0.10);
}
```

```
void drawTree(float x, float y)
{
    glPushMatrix();
    glTranslatef(x, y, 0);

    // Leaves
    glColor3f(0.0, 0.5, 0.0);
    drawTriangle(-0.03, 0, 0.03, 0, 0, 0.25);

    // Trunk
    glColor3f(0.5, 0.2, 0.1);
    drawRectangle(-0.01, -0.1, 0.02, 0.1);

    glPopMatrix();
}

void drawFlower(float x, float y)
{
    glPushMatrix();
    glTranslatef(x, y, 0);

    glColor3f(1, 0, 0);
    drawCircle(0.03, 0.03, 0.02);
    drawCircle(-0.03, 0.03, 0.02);
    drawCircle(0.03, -0.03, 0.02);
    drawCircle(-0.03, -0.03, 0.02);

    glColor3f(1, 1, 0);
    drawCircle(0, 0, 0.02);

    glColor3f(0, 0.6, 0);
    drawRectangle(-0.01, -0.1, 0.02, 0.1);

    glPopMatrix();
}
void drawBigTree(float x, float y, float scale)
{
    glPushMatrix();
    glTranslatef(x, y, 0);
    glScalef(scale, scale, 1);

    // Leaves (bigger)
    glColor3f(0.0, 0.5, 0.0);
    drawTriangle(-0.06, 0, 0.06, 0, 0, 0.45);

    // Trunk
    glColor3f(0.5, 0.2, 0.1);
    drawRectangle(-0.02, -0.15, 0.04, 0.15);

    glPopMatrix();
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
```

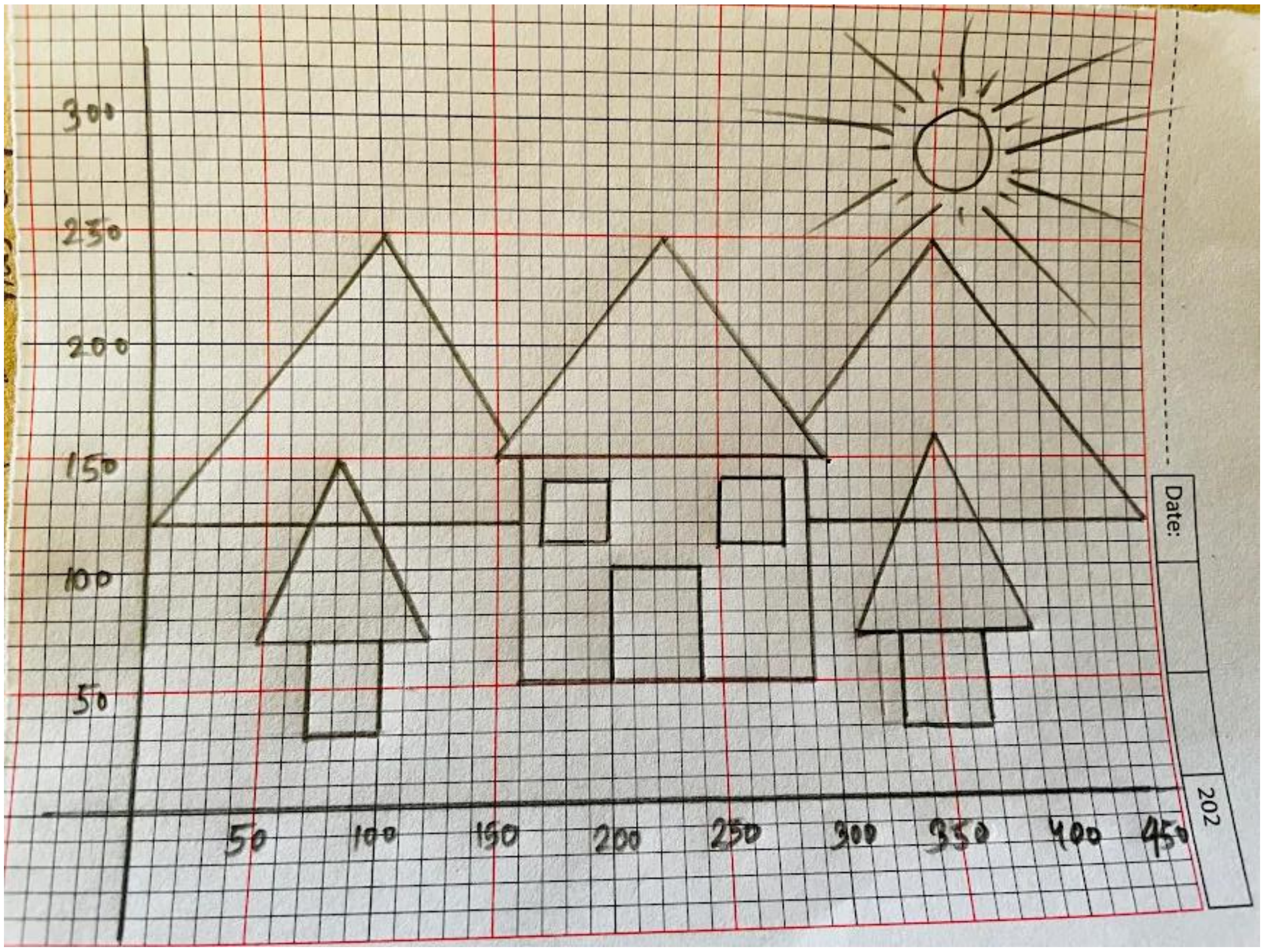
```
// Background sky
glColor3f(0.52, 0.80, 0.98);
drawRectangle(-1, -1, 2, 2);

// Mountains
glColor3f(0.8, 0.6, 0.3);
drawTriangle(-1, -0.3, -0.4, 0.5, 0.2, -0.3);
drawTriangle(-0.2, -0.3, 0.3, 0.6, 0.8, -0.3);

// Sun
drawSun();

// House body
glColor3f(1, 0.8, 0);
drawRectangle(-0.4, -0.6, 0.8, 0.5);

// Roof
glColor3f(0.9, 0.2, 0.2);
drawTriangle(-0.45, -0.1, 0.45, -0.1, 0, 0.3);

// Door
glColor3f(1, 0.3, 0.1);
drawRectangle(-0.08, -0.6, 0.16, 0.3);

// Windows
glColor3f(0, 0.6, 1);
drawRectangle(-0.32, -0.35, 0.20, 0.18);
drawRectangle(0.12, -0.35, 0.20, 0.18);
```

```
    //  Big Trees
    drawBigTree(-0.55, -0.35, 1.5);   // Left big tree
    drawBigTree(0.55, -0.35, 1.5);    // Right big tree

    // Small Trees on right
    drawTree(0.75, -0.4);
    drawTree(0.90, -0.4);
    drawTree(1.05, -0.4);

    // Small Tree on left
    drawTree(-0.8, -0.4);

    // Flowers
    drawFlower(-0.55, -0.65);
    drawFlower(-0.38, -0.65);

    glFlush();
}

void init()
{
    glClearColor(1, 1, 1, 1); // White background
    gluOrtho2D(-1, 1, -1, 1);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Scenery using Primitives & 2D Transformations");

    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Output



Graph



Discussion

This task required drawing entire scenery using only primitive shapes and applying 2D transformations to position and scale each object correctly. To achieve this:

- The **background**, **mountains**, and **house** were drawn using rectangles and triangles.
- The **sun** was created using a circle combined with rotated rays using rotation transformations.
- Multiple **trees** were placed across the scene by applying translation, and large trees were created by applying scaling.
- **Flowers** were drawn with multiple circles arranged in a symmetric pattern.

- Each object was placed in the correct position using glTranslatef(), scaled using glScalef(), and grouped using glPushMatrix() and glPopMatrix().

By modularizing the drawing functions (tree, flower, sun, mountain), the entire scenery becomes structured, easier to understand, and reusable. The use of 2D transformations helps demonstrate how complex compositions can be formed from simple shapes.

Conclusion

This project successfully demonstrates the use of **OpenGL primitives** along with **2D Transformation techniques** to draw detailed scenery. By combining simple shapes such as triangles, rectangles, and circles, complex objects like trees, flowers, mountains, and a house were created. Translation, scaling, and rotation transformations allowed proper placement, orientation, and resizing of the objects. Overall, the project provides a strong understanding of how fundamental graphics concepts are used to construct meaningful scenes in computer graphics.