Lab Report

Course Code: CSE 422

Course Title: Computer Graphics Lab

Report No: 09

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

Submitted To:

Syada Tasmia Alvi

Senior Lecturer

Department of Computer Science and Engineering (C.S.E)

Submitted by:

Name: Mahmudul Hasan Piash

ID: 221-15-5606

Section: 61_D1

Department of Computer Science and Engineering (C.S.E)

Date of submission: 15th December, 2025

Title

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

Introduction

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

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, locomotive body, train car body, wheels axles, windows, chimney, and tree components.

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

- Uses GL_TRIANGLES to draw triangular shapes.
- Applied for tree leaves, palm tree leaves, and smoke effects.

3. drawCircle(cx, cy, r)

- Approximates a circle using 360 small segments.
- Used for wheels, clouds, smoke, sun, and fruits on trees.

4. drawSky()

- Creates the blue sky background and green grass ground respectively.
- Fundamental components for setting the scene environment.

5. drawCloud(x,y)

- Composed of 3 overlapping circles to create a cloud shape.
- Positioned using translation transformations.

6. drawSunRays(x, y)

- Creates 3 diagonal yellow rays in the top-left corner.
- Uses drawLine() with varied positions for aesthetic effect.

7. drawSky() & drawGround()

- Creates the blue sky background and green grass ground respectively.
- Fundamental components for setting the scene environment

8. drawTree(x, y) & drawFruitTree(x, y)

- Regular trees: Composed of a brown rectangle (trunk) and green triangle (leaves).
- Fruit trees: Include red circles positioned on the tree leaves using translation.
- Uses translation (glTranslatef) to position trees in the scene

9. drawWheel(x, y, radius)

- Multi-layered wheel design with dark gray outer circle (tire), light gray middle circle (rim), and medium gray center (hub).
- Used for both locomotive and train car wheels with different scales.

10. drawLocomotive(x, y)

- Main locomotive body composed of:
  - Dark blue main body rectangle.
  - Blue cabin rectangle on top.
  - Red chimney rectangle.
  - Light blue window rectangle.
  - Dark front bumper.
  - Gray wheel axles.
  - Two wheels positioned below.
  - Gray coupling point for connecting to train car.
- Uses nested rectangles and glPushMatrix/glPopMatrix for organization.

11. drawTrainCar(x, y)

- Train cargo car composed of:
  - Red bottom body rectangle.
  - Yellow top cargo rectangle.
  - Red chimney rectangle.
  - Gray axles.
  - Two wheels with different scale than locomotive.
  - Gray coupling connection point.
- Demonstrates scaling of wheels to different sizes.

12. drawSmoke(x, y)

- Three overlapping circles of decreasing size positioned vertically.
- Creates smoke effect rising from the locomotive chimney.

13. 2D Transformations Used

- **Translation**: Positioning all objects (locomotive, train car, wheels, trees, clouds) using glTranslatef().
- **Scaling**: Creating different-sized trees and wheels using glScalef().
- **Matrix Operations**: Using glPushMatrix() and glPopMatrix() to manage transformation hierarchies.

Code

```
#include <GL/glut.h>

#include <math.h>

#include <vector>


const float PI = 3.14159265359;


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 * PI / 180;
        glVertex2f(cx + r * cos(theta), cy + r * sin(theta));
    }
    glEnd();
}

void drawWheel(float x, float y, float r) {
    glPushMatrix();
    glTranslatef(x, y, 0);
    glColor3f(0.6, 0.6, 0.6);
    drawCircle(0, 0, r);
    glColor3f(0.8, 0, 0);
    drawCircle(0, 0, r * 0.6);
    glPopMatrix();
}

void drawCloud(float x, float y) {
```

```
    glPushMatrix();
    glTranslatef(x, y, 0);
    glColor3f(0.85, 0.85, 0.85);
    drawCircle(0, 0, 0.05);
    drawCircle(0.06, 0.01, 0.06);
    drawCircle(0.12, 0, 0.05);
    drawCircle(0.06, 0.04, 0.05);
    glPopMatrix();
}

void drawCustomSun(float x, float y) {
    glPushMatrix();
    glTranslatef(x, y, 0);
    glColor3f(1, 1, 0);
    drawCircle(0, 0, 0.08);

    glLineWidth(3.0);
    glBegin(GL_LINES);
    for (int i = 0; i < 8; i++) {
        float theta = i * (2 * PI / 8);
        float r_inner = 0.1;
        float r_outer = 0.18;
        glVertex2f(r_inner * cos(theta), r_inner * sin(theta));
        glVertex2f(r_outer * cos(theta), r_outer * sin(theta));
    }
    glEnd();
    glLineWidth(1.0);
```

```
    glPopMatrix();
}

void drawPineTree(float x, float y) {
    glPushMatrix();
    glTranslatef(x, y, 0);
    glColor3f(0.4, 0.25, 0.1);
    drawRectangle(-0.02, -0.2, 0.04, 0.2);
    glColor3f(0.1, 0.6, 0.1);
    drawTriangle(-0.12, -0.05, 0.12, -0.05, 0, 0.15);
    drawTriangle(-0.1, 0.1, 0.1, 0.1, 0, 0.3);
    drawTriangle(-0.08, 0.25, 0.08, 0.25, 0, 0.4);
    glPopMatrix();
}

void drawAppleTree(float x, float y) {
    glPushMatrix();
    glTranslatef(x, y, 0);
    glColor3f(0.4, 0.25, 0.1);
    drawRectangle(-0.03, -0.2, 0.06, 0.3);
    glColor3f(0.1, 0.7, 0.1);
    drawCircle(0, 0.2, 0.18);
    glColor3f(0.9, 0, 0);
    drawCircle(-0.08, 0.15, 0.02);
    drawCircle(0.05, 0.12, 0.02);
    drawCircle(-0.02, 0.25, 0.02);
    drawCircle(0.1, 0.22, 0.02);
```

```
    drawCircle(-0.06, 0.3, 0.02);
    glPopMatrix();
}

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

    glColor3f(0.1, 0.1, 0.7);
    drawRectangle(0, 0, 0.45, 0.25);

    glColor3f(0.8, 0, 0);
    drawRectangle(0.25, 0.25, 0.2, 0.2);
    drawRectangle(0.23, 0.45, 0.24, 0.03);

    glColor3f(0.5, 0.8, 1.0);
    drawRectangle(0.28, 0.28, 0.14, 0.14);

    glColor3f(0.8, 0, 0);
    glBegin(GL_POLYGON);
    glVertex2f(0.05, 0.25);
    glVertex2f(0.15, 0.25);
    glVertex2f(0.18, 0.45);
    glVertex2f(0.02, 0.45);
    glEnd();

    glColor3f(0.8, 0, 0);
```

```
    drawTriangle(0, 0, 0, 0.15, -0.1, 0);

    glColor3f(1, 1, 0);
    drawCircle(0, 0.18, 0.04);

    glPopMatrix();
}

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

    glColor3f(0.1, 0.1, 0.7);
    drawRectangle(0.05, 0, 0.3, 0.1);

    glColor3f(0.8, 0, 0);
    glBegin(GL_POLYGON);
    glVertex2f(0, 0.1);
    glVertex2f(0.4, 0.1);
    glVertex2f(0.38, 0.3);
    glVertex2f(0.02, 0.3);
    glEnd();

    glColor3f(1, 1, 0);
    drawTriangle(0.05, 0.3, 0.35, 0.3, 0.2, 0.45);

    glPopMatrix();
```

```
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(0.3, 0.7, 1.0);
    drawRectangle(-1, -1, 2, 2);

    glColor3f(0.3, 0.8, 0.3);
    drawCircle(0, -1.3, 1.5);

    drawCustomSun(-0.75, 0.75);
    drawCloud(-0.4, 0.6);
    drawCloud(0.0, 0.7);
    drawCloud(0.4, 0.6);

    drawPineTree(0.4, 0.1);
    drawAppleTree(0.75, 0.05);

    float trainY = -0.25;

    glColor3f(0.6, 0.6, 0.6);
    drawRectangle(-0.1, trainY + 0.05, 0.2, 0.02);

    drawEngineBody(-0.55, trainY);
    drawWheel(-0.45, trainY, 0.08);
    drawWheel(-0.2, trainY, 0.08);
```

```
    drawWagonBody(0.1, trainY);
    drawWheel(0.2, trainY - 0.02, 0.06);
    drawWheel(0.4, trainY - 0.02, 0.06);

    glFlush();
}

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

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Train Scenery");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

Output

Discussion

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

- The **background** (sky and ground) was created using large rectangles with different colors.
- The **sun** was created using sun rays with appropriate positioning and coloring.
- **Clouds** were formed by overlapping circles arranged in a symmetric pattern.
- The **locomotive** was built using multiple rectangles layered together (body, cabin, chimney) with a complex color scheme to represent the vehicle structure.
- The **train car** was similarly constructed with rectangles for the body and top cargo section.
- **Wheels** were created using concentric circles to show tire, rim, and hub details, with scaling applied to create wheels of different sizes.
- **Trees** were positioned across the scene using translation transformations, with fruit trees including red circles for apples/fruits using positioned circles.
- **Smoke** rising from the chimney was created using three circles of decreasing size.

By modularizing the drawing functions (wheel, locomotive, train car, cloud, tree, smoke), the entire scenery becomes structured, easier to understand, and reusable. The use of 2D transformations (translation, scaling) demonstrates how complex compositions can be formed from simple primitive shapes. The hierarchical use of glPushMatrix() and glPopMatrix() ensures that transformations applied to parent objects (like the locomotive's position) correctly affect all child objects (like its wheels).