Graphics and Visualizations - SE3032
Lab 5

IT23226128 – D.M.D.G.B.Seneviratne

Part A – Manual Exercises

Task 1: Translation

Question 1: A quadrilateral has vertices at the points: P(0, 0), Q(6, 0), R(6, 4), and S(0, 4). Apply a translation with the translation vector Tx = 2 and Ty = -5. What are the new coordinates of the quadrilateral after translation?

- P'(2, -5)
- Q'(8, -5)
- R'(8, -1)
- S'(2, -1)

Question 2: On a map grid, a treasure chest is located at point T(150, 80). The instructions state: "Move 50 units East and 30 units North to find the key." Assuming East is the positive x-direction and North is the positive y-direction, what are the coordinates of the key?

- Key's coordinates = (150 + 50, 80 + 30) = (200, 110)

Question 3: After a translation, a triangle's vertices are at A'(0, -2), B'(3, 5), and C'(-1, 1). The translation vector used was Tx = -2, Ty = 4. What were the original coordinates of the triangle before the translation?

- Original = Final – Vector
  (x, y) = (x' – (-2), y' – 4) = (x' + 2, y' – 4)
  - A: (0 + 2, -2 -4) = (2, -6)
  - B: (3 + 2, 5 – 4) = (5, 1)
  - C: (-1 + 2, 1 – 4) = (1, -3)
- Original triangle: A(2, -6), B(5, 1), C(1, -3)

Task 2: Scaling

Question 4: A quadrilateral has vertices at P(2, 3), Q(5, 3), R(5, 6), and S(2, 6). Apply scaling with Sx = 3 and Sy = 2 about the fixed point (4, 5). Find the new coordinates of the quadrilateral.

- Scale Sx = 3, Sy = 2 about F(4,5)
    - P'(-2, 1)
    - Q'(7, 1)
    - R'(7, 7)
    - S'(-2, 7)

Question 5: A triangle has vertices at A(4, 2), B(1, -1), and C(3, -3). 5 Apply scaling with Sx = 0.5 and Sy = 1.5 about the fixed point (0, 0). Find the new coordinates of the triangle.

- Scale Sx = 0.5, Sy = 1.5 about F(0,0)
    - A'(-3, -2)
    - B'(-5, -4)
    - C'(_6, -1)

Task 3: Rotation

Question 6: A triangle has vertices at A(3, 2), B(5, 4), and C(6, 1). Rotate the triangle by 180 degrees counterclockwise about the origin (0, 0). What are the new coordinates of the triangle after rotation?

(180° about origin ⇒ (x′,y′) = (−x, −y))

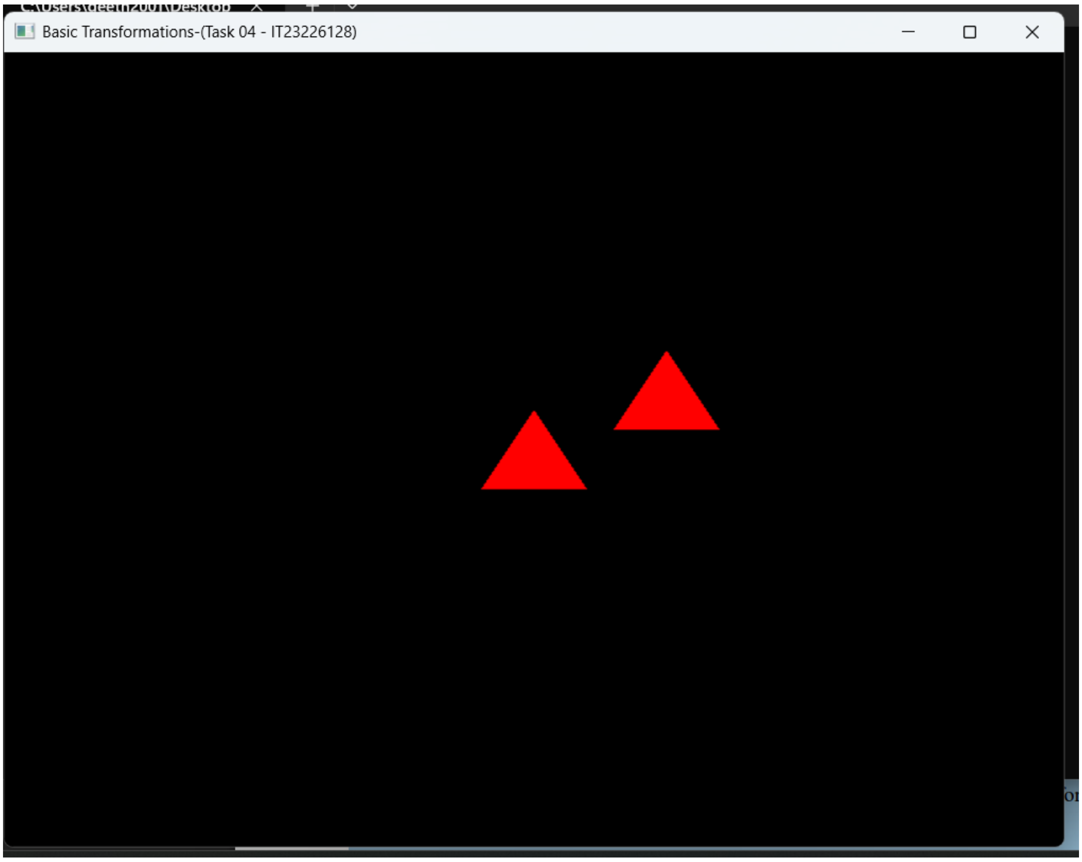
- A′(−3, −2)
- B′(−5, −4)
- C′(−6, −1)

Question 7: A triangle has vertices at A(2, 3), B(5, 3), and C(5, 6). Rotate the triangle by 90 degrees counterclockwise about the fixed point (3, 3). What are the new coordinates?

- Rotate 90' CCW about F(3, 3)
    - A'(3, 2)
    - B'(3, 5)
    - C'(0, 5)

Part B – Coding Exercise

Task 4: Implement Basic Transformations



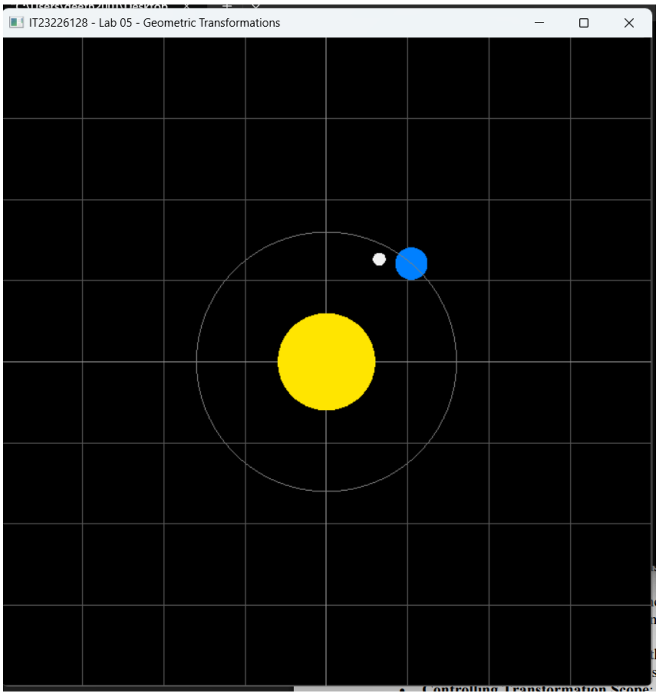
Code

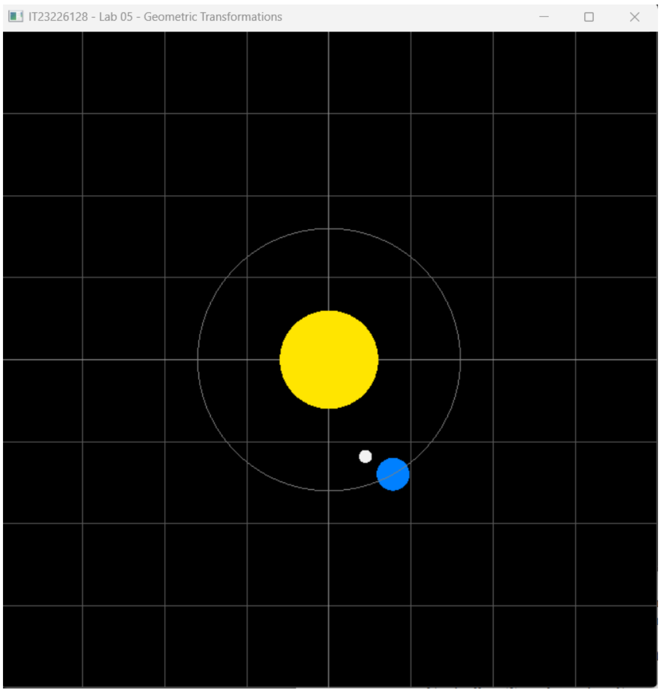
```
      #include <GL/glut.h>
#include <math.h>
#include <stdio.h>
// Function to draw a simple triangle
void drawTriangle(void) {
// Set color to red (RGB values between 0.0 and 1.0)
   glColor3f(1.0, 0.0, 0.0);

   // Begin drawing triangles
   glBegin(GL_TRIANGLES);

   // Define three vertices to form a triangle
   // Bottom left vertex
   glVertex2f(-0.2, -0.2);
   // Bottom right vertex
   glVertex2f(0.2, -0.2);
   // Top vertex
   glVertex2f(0.0, 0.2);

   // End drawing
   glEnd();
}
```

```
// Translation transformation example
void Translate() {
   glClear(GL_COLOR_BUFFER_BIT);

   // Draw original triangle
   glColor3f(1.0, 0.0, 0.0); // Red
   drawTriangle();

   // Apply translation transformation
   // Save current transformation matrix
   glPushMatrix();

   // Translate by (0.5, 0.3, 0.0)
   glTranslatef(0.5, 0.3, 0.0);

   // Draw translated triangle in blue
   glColor3f(0.0, 0.0, 1.0);
   drawTriangle();

   // Restore previous transformation matrix
   glPopMatrix();

   glutSwapBuffers();
}

// Scaling transformation about origin
void ScaleAboutOrigin() {
   glClear(GL_COLOR_BUFFER_BIT);

   // Draw the original triangle first for reference
   glColor3f(1.0, 0.0, 0.0); // Red
   drawTriangle();

   // Apply scaling transformation about origin
   glPushMatrix();
   // Scale by 2.0 in X, 1.5 in Y, 1.0 in Z
   glScalef(2.0, 1.5, 1.0);

   // Draw scaled triangle in blue
   glColor3f(0.0, 0.0, 1.0);
   drawTriangle();
   glPopMatrix();

   glFlush();
}

// Scaling transformation about a fixed point
void ScaleAboutFixedPoint() {
   glClear(GL_COLOR_BUFFER_BIT);

   // Draw the original triangle first for reference
   glColor3f(1.0, 0.0, 0.0); // Red
   drawTriangle();
```

```
    // Implement scaling about a fixed point (0.25, 0.25)
    // Strategy: Move fixed point to origin, scale, then move back
    glPushMatrix();

    // Move fixed point to origin
    glTranslatef(0.25, 0.25, 0.0);
    // Apply scaling
    glScalef(2.0, 1.5, 1.0);
    // Move back to original position
    glTranslatef(-0.25, -0.25, 0.0);

    // Draw scaled triangle in green
    glColor3f(0.0, 1.0, 0.0);
    drawTriangle();
    glPopMatrix();

    glFlush();
}

// Rotation transformation about origin
void RotationAboutOrigin() {
    glClear(GL_COLOR_BUFFER_BIT);

    // Draw the original triangle first for reference
    glColor3f(1.0, 0.0, 0.0); // Red
    drawTriangle();

    // Apply rotation transformation about origin
    glPushMatrix();
    // Rotate 45 degrees around Z-axis
    glRotatef(45.0, 0.0, 0.0, 1.0);

    // Draw rotated triangle in blue
    glColor3f(0.0, 0.0, 1.0);
    drawTriangle();
    glPopMatrix();

    glFlush();
}

// Rotation about a fixed point (e.g., first vertex)
void RotationAboutFixedPoint() {
    glClear(GL_COLOR_BUFFER_BIT);

    // Draw the original triangle first for reference
    glColor3f(1.0, 0.0, 0.0); // Red
    drawTriangle();

    // Implement rotation about a fixed point (0.0, 0.0)
    // Strategy: Move fixed point to origin, rotate, then move back
    glPushMatrix();

    // Move fixed point to origin (in this case it's already at origin)
    glTranslatef(0.0, 0.0, 0.0);
    // Rotate 45 degrees around Z-axis
    glRotatef(45.0, 0.0, 0.0, 1.0);
    // Move back to original position
    glTranslatef(0.0, 0.0, 0.0);

    // Draw rotated triangle in green
    glColor3f(0.0, 1.0, 0.0);
    drawTriangle();
    glPopMatrix();

    glFlush();
}

void init() {
    glClearColor(0.0, 0.0, 0.0, 0.0); // Black background
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-2.0, 2.0, -2.0, 2.0); // 2D orthographic projection
    glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char** argv) {
    // Initialize GLUT
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Basic Transformations-(Task 04 - IT23226128)");

    init();

    // Uncomment the function you want to test:
    glutDisplayFunc(Translate);
    // glutDisplayFunc(ScaleAboutOrigin);
    // glutDisplayFunc(ScaleAboutFixedPoint);
    // glutDisplayFunc(RotationAboutOrigin);
    // glutDisplayFunc(RotationAboutFixedPoint);

    // Enter the main event loop
    glutMainLoop();
    return 0;
}
```

Task 5: Composite Transformation





Code

```
#include <GL/glut.h>
#include <math.h>
#include <stdio.h>
// ------- Helpers -------
static void drawTriangle(void) {
   glColor3f(1.0f, 0.0f, 0.0f);          // red
   glBegin(GL_TRIANGLES);
      glVertex2f(-0.5f, -0.5f);         // bottom-left
      glVertex2f( 0.5f, -0.5f);         // bottom-right
      glVertex2f( 0.0f,  0.5f);         // top
   glEnd();
}

static void drawAxes(void) {
   glLineWidth(1.0f);
   glColor3f(0.7f, 0.7f, 0.7f);
   glBegin(GL_LINES);
      glVertex2f(-1.0f, 0.0f); glVertex2f(1.0f, 0.0f);    // X
      glVertex2f(0.0f, -1.0f); glVertex2f(0.0f, 1.0f);    // Y
   glEnd();
}

// ------- Transform demos -------
void Translate() {
    glClear(GL_COLOR_BUFFER_BIT);
    drawAxes();

    // Original
    drawTriangle();

    // Translated copy
    glPushMatrix();
        glTranslatef(0.5f, 0.3f, 0.0f);
        glColor3f(0.0f, 0.8f, 1.0f);  // cyan to distinguish
        drawTriangle();
    glPopMatrix();

    glutSwapBuffers();
}

void ScaleAboutOrigin() {
    glClear(GL_COLOR_BUFFER_BIT);
    drawAxes();

    // Original
    drawTriangle();

    // Scaled (origin is the pivot)
    glPushMatrix();
        glScalef(2.0f, 1.5f, 1.0f);
        glColor3f(0.2f, 1.0f, 0.2f);
        drawTriangle();
    glPopMatrix();

    glFlush();
}

void ScaleAboutFixedPoint() {
    glClear(GL_COLOR_BUFFER_BIT);
    drawAxes();

    // Fixed point
    const float fx = 0.25f, fy = 0.25f;

    // Original
    drawTriangle();

    // Move pivot to origin -> scale -> move back
    glPushMatrix();
        glTranslatef(fx, fy, 0.0f);
        glScalef(1.8f, 0.7f, 1.0f);
        glTranslatef(-fx, -fy, 0.0f);
        glColor3f(1.0f, 0.6f, 0.0f);
        drawTriangle();
    glPopMatrix();

    glFlush();
}
```

```
void RotationAboutOrigin() {
    glClear(GL_COLOR_BUFFER_BIT);
    drawAxes();

    // Original
    drawTriangle();

    // Rotated 45° about origin
    glPushMatrix();
        glRotatef(45.0f, 0.0f, 0.0f, 1.0f);
        glColor3f(0.6f, 0.0f, 1.0f);
        drawTriangle();
    glPopMatrix();

    glFlush();
}

void RotationAboutFixedPoint() {
    glClear(GL_COLOR_BUFFER_BIT);
    drawAxes();

    // Fixed point (the triangle's lower-left vertex roughly around -0.5,-0.5)
    const float fx = -0.5f, fy = -0.5f;

    // Original
    drawTriangle();

    // Move pivot to origin -> rotate -> move back
    glPushMatrix();
        glTranslatef(fx, fy, 0.0f);
        glRotatef(45.0f, 0.0f, 0.0f, 1.0f);
        glTranslatef(-fx, -fy, 0.0f);
        glColor3f(0.0f, 0.9f, 0.5f);
        drawTriangle();
    glPopMatrix();

    glFlush();
}

// ------- Composite: Sun–Earth–Moon -------
static float earthOrbitAngle = 0.0f;
static float earthRotationAngle = 0.0f;
static float moonOrbitAngle  = 0.0f;

static void drawCircle(float cx, float cy, float r, int segs, float rcol, float gcol, float bcol) {
    glColor3f(rcol, gcol, bcol);
    glBegin(GL_TRIANGLE_FAN);
        glVertex2f(cx, cy); // center
        for (int i = 0; i <= segs; ++i) {
            float t = 2.0f * 3.14159265f * (float)i / (float)segs;
            glVertex2f(cx + r * cosf(t), cy + r * sinf(t));
        }
    glEnd();
}

static void drawGrid() {
   glColor3f(0.35f, 0.35f, 0.35f);
   glBegin(GL_LINES);
      for (float x = -2.0f; x <= 2.01f; x += 0.5f) { glVertex2f(x, -2.0f); glVertex2f(x, 2.0f); }
      for (float y = -2.0f; y <= 2.01f; y += 0.5f) { glVertex2f(-2.0f, y); glVertex2f(2.0f, y); }
   glEnd();
   // axes a bit brighter
   glColor3f(0.6f, 0.6f, 0.6f);
   glBegin(GL_LINES);
      glVertex2f(-2.0f, 0.0f); glVertex2f(2.0f, 0.0f);
      glVertex2f(0.0f, -2.0f); glVertex2f(0.0f, 2.0f);
   glEnd();
}

void SolarSystemDisplay() {
   glClear(GL_COLOR_BUFFER_BIT);
   drawGrid();

   // Sun (scaled at origin)
   glPushMatrix();
      glScalef(1.5f, 1.5f, 1.0f);
      drawCircle(0.0f, 0.0f, 0.2f, 50, 1.0f, 0.9f, 0.0f); // yellow
   glPopMatrix();

   // Earth: orbit around Sun, then self-rotate
   glPushMatrix();                             // world
      glRotatef(earthOrbitAngle, 0, 0, 1);          // orbit around Sun
      glTranslatef(0.8f, 0.0f, 0.0f);               // orbital radius
      glPushMatrix();                          // earth local
         glRotatef(earthRotationAngle, 0, 0, 1);    // self rotation
         drawCircle(0.0f, 0.0f, 0.10f, 36, 0.0f, 0.5f, 1.0f); // blue
      glPopMatrix();

      // Moon: nested relative to Earth
      glPushMatrix();                          // moon local (relative to Earth)
         glRotatef(moonOrbitAngle, 0, 0, 1);
         glTranslatef(0.2f, 0.0f, 0.0f);
         drawCircle(0.0f, 0.0f, 0.04f, 24, 0.95f, 0.95f, 0.95f);
      glPopMatrix();
   glPopMatrix();

   // (Optional) Earth orbit path
   glColor3f(0.5f, 0.5f, 0.5f);
   glBegin(GL_LINE_LOOP);
      for (int i = 0; i < 100; ++i) {
         float t = 2.0f * 3.14159265f * i / 100.0f;
         glVertex2f(0.8f * cosf(t), 0.8f * sinf(t));
      }
   glEnd();

   glutSwapBuffers();
}
```

```
void timer(int v) {
   earthOrbitAngle     += 0.2f;   // slow
   earthRotationAngle  += 1.0f;   // faster
   moonOrbitAngle      += 0.5f;   // medium
   glutPostRedisplay();
   glutTimerFunc(16, timer, 0);   // ~60 FPS
}

void init(void) {
   glClearColor(0,0,0,1);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-2.0, 2.0, -2.0, 2.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

static void printInstructions(void) {
   printf("Transforms: Translate / Scale / Rotate; Composite: Sun-Earth-Moon\n");
   printf("In main(), switch which glutDisplayFunc to run.\n");
}

int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize(900, 700);
   glutCreateWindow("IT23226128 - Lab 05 - Geometric Transformations");

   init();
   printInstructions();

   // === Pick what to show: uncomment ONE ===
   //glutDisplayFunc(Translate);
   //glutDisplayFunc(ScaleAboutOrigin);
   //glutDisplayFunc(ScaleAboutFixedPoint);
   //glutDisplayFunc(RotationAboutOrigin);
   //glutDisplayFunc(RotationAboutFixedPoint);
   glutDisplayFunc(SolarSystemDisplay);  // composite demo

   glutTimerFunc(0, timer, 0);
   glutMainLoop();
   return 0;
}
```