UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Ex. No.6 2D Composite Transformations and Windowing in C++ using OpenGL

Date: 30/8/21 **Name:** Srinath S

Class: CSE-C **Roll:** 185001205

Question:

a) To compute the composite transformation matrix for any 2 transformations given as input by the user and applying it on the object. The transformation can be any combination of the following.

1) Translation

2) Rotation

3) Scaling

4) Reflection

5) Shearing Display the original and the transformed object. Calculate the final transformation matrix by multiplying the two individual transformation matrices and then apply it to the object.

Note: Use Homogeneous coordinate representations and matrix multiplication to perform transformations. Divide the output window into four quadrants. (Use LINES primitive to draw x and y axis)

b) Create a window with any 2D object and a different sized viewport. Apply window to viewport transformation on the object. Display both window and viewport.

Code:

```
#include<GL/glut.h>
#include<iostream>
#include<cmath>
using namespace std;
int flag;
double a[3][1], b[3][1], c[3][1];
double a1[3][1], b1[3][1], c1[3][1];
double t1[3][3], t2[3][3];

void myInit() {
    glClearColor(1.0,1.0,1.0,0.0);
    glColor3f(0.0f,0.0f,0.0f);
    glPointSize(2);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,1920.0,0.0,1080.0);
```

```
}
void transformObject(){
    double m[3][3];
    memset(m, 0, sizeof(m));
    for(int i=0; i<3; i++){
    for(int j=0; j<3; j++)cout<<t1[i][j]<<' ';
        cout<<'\n';
    }
    cout<<'\n';
    for(int i=0; i<3; i++){
        for(int j=0; j<3; j++)cout<<t2[i][j]<<' ';
        cout<<'\n';
    }
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            for(int k=0;k<3;k++){
                m[i][j] += t2[i][k]*t1[k][j];
            }
        }
    }
    memset(a1, 0, sizeof(a1));
    memset(b1, 0, sizeof(b1));
    memset(c1, 0, sizeof(c1));
    for(int i=0;i<3;i++){
        for(int j=0;j<1;j++){
            for(int k=0;k<3;k++){
                a1[i][j]+= m[i][k]*a[k][j];
                b1[i][j]+= m[i][k]*b[k][j];
                c1[i][j]+= m[i][k]*c[k][j];
                }
            }
        }
    }
void assignTM(double tm[3][3]){
    if(flag == 0){
        for(int i=0;i<3;i++)for(int j=0;j<3;j++)t1[i][j] = tm[i][j];
        flag = 1;
    }
    for(int i=0;i<3;i++)for(int j=0;j<3;j++)t2[i][j] = tm[i][j];
}
void drawTriangle(double a[3][1], double b[3][1], double c[3][1], int s=0) {
    a[0][0] += 960;
    b[0][0] += 960;
    c[0][0] += 960;
    a[1][0] += 540;
    b[1][0] += 540;
    c[1][0] += 540;
    glBegin(GL_TRIANGLES);
```

```
    if(s)
        glColor4f(0.1f, 0.3f, 0.9f, 0.5f);
    else
        glColor4f(0.9f, 0.2f, 0.1f, 0.5f);

    glVertex2d(a[0][0],a[1][0]);
    glVertex2d(b[0][0],b[1][0]);
    glVertex2d(c[0][0],c[1][0]);
    glEnd();
}
void translateObject(double tx, double ty){
    double tm[3][3];
    memset(tm, 0, sizeof(tm));
    tm[0][0] = tm[1][1] = tm[2][2] = 1;
    tm[0][2] = tx;
    tm[1][2] = ty;
    assignTM(tm);
}
void rotateObject(double deg, int ch){
    int tx = 0, ty = 0;
    if(ch == 2){
        cout<<"Enter pivot: ";
        cin>>tx>>ty;
    }
    deg = (3.14)*deg/180;
    double tm[3][3];
    memset(tm, 0, sizeof(tm));
    tm[2][2] = 1;
    tm[0][0] = cos(deg);tm[1][1] = tm[0][0];
    tm[1][0] = sin(deg);tm[0][1] = -tm[1][0];
    tm[0][2] = tx*(double)(1.0-cos(deg)) + ty*sin(deg);
    tm[1][2] = ty*(double)(1.0-cos(deg)) - tx*sin(deg);
    assignTM(tm);
}
void scaleObject(double sx, double sy, int ch){
    int tx = 0, ty = 0;
    if(ch == 2){
        cout<<"Enter pivot: ";
        cin>>tx>>ty;
    }
    double tm[3][3];
    memset(tm, 0, sizeof(tm));
    tm[2][2] = 1;
    tm[0][0] = sx;tm[1][1] = sy;
    tm[0][2] = tx*(double)(1.0-sx);
    tm[1][2] = ty*(double)(1.0-sy);
    assignTM(tm);
}
```

```
void reflectObject(int ch){
    double tm[3][3];
    memset(tm, 0, sizeof(tm));
    tm[2][2] = 1;
    switch(ch){
        case 1: tm[0][0] = 1, tm[1][1] = -1;
        break;
        case 2: tm[0][0] = -1, tm[1][1] = 1;
        break;
        case 3: tm[0][0] = -1, tm[1][1] = -1;
        break;
        case 4: tm[0][1] = 1, tm[1][0] = 1;
        break;
    }
    assignTM(tm);
}
void shearObject(double s, int x, int ch){
    double tm[3][3];
    memset(tm, 0, sizeof(tm));
    tm[0][0] = tm[1][1] = tm[2][2] = 1;
    double ref;
    if(ch==2){
        cout<<"Enter reference line: ";
        cin>>ref;
    }
    if(x)tm[0][1] = s, tm[0][2] = -s*ref;
    else tm[1][0] = s, tm[1][2] = -s*ref;
    assignTM(tm);
}
void chooseTransformation(int choice) {
    int ch;
    switch(choice){
        case 1: double tx, ty;
            cout<<"Enter translation vector: ";
            cin>>tx>>ty;
            translateObject(tx, ty);
            break;
        case 2:
            double deg;cout<<"Enter angle: ";cin>>deg;
            cout<<"Options:\n\t1. Origin\n\t2. Pivot\n\tChoice: ";
            cin>>ch;
            rotateObject(deg, ch);
            break;
        case 3:
            double sx, sy;cout<<"Enter scaling factor: ";cin>>sx>>sy;
            cout<<"Options:\n\t1. Origin\n\t2. Pivot\n\tChoice: ";
            cin>>ch;
            scaleObject(sx, sy, ch);
```

```
            break;
        case 4:
            cout<<"Options:\n\t1. X-axis\n\t2. Y-axis\n\t3. Origin\n\t4.
X=Y\n\tChoice: ";
            cin>>ch;
            reflectObject(ch);
            break;
        case 5:
            double s;cout<<"Enter shear parameter: ";cin>>s;
            int x;cout<<"Enter axis:(1-x, 0-y) ";cin>>x;
            cout<<"Options:\n\t1. Origin\n\t2. Different reference
line\n\tChoice: ";
            cin>>ch;
            shearObject(s, x, ch);
    }
}
void myDisplay(){
    glColor3f(1, 1, 1);
    glBegin(GL_LINES);
    glVertex2d(0,540);
    glVertex2d(1920,540);
    glVertex2d(960,0);
    glVertex2d(960,1080);
    glEnd();
    drawTriangle(a1, b1, c1, 1);
    drawTriangle(a, b, c);
    glFlush();
}
int main(int argc, char** argv) {
    memset(a, 0, sizeof(a));
    memset(b, 0, sizeof(b));
    memset(c, 0, sizeof(c));
    memset(t1, 0, sizeof(t1));
    memset(t2, 0, sizeof(t2));
    cout<<"Enter coordinates of triangle: ";
    cin>>a[0][0]>>a[1][0]>>b[0][0]>>b[1][0]>>c[0][0]>>c[1][0];
    int choice;
    a[2][0] = b[2][0] = c[2][0] = 1;
    cout<<"Select two transformations\n\n";
    cout<<"Transformation 1:\n\t1. Translation\n\t2. Rotation\n\t3.
Scaling\n\t4.Reflection\n\t5. Shearing\n\tChoice: ";
    cin>>choice;
    chooseTransformation(choice);
    cout<<"\nTransformation 2:\n\t1. Translation\n\t2. Rotation\n\t3.
Scaling\n\t4.Reflection\n\t5. Shearing\n\tChoice: ";
    cin>>choice;
    chooseTransformation(choice);
    transformObject();
```

```
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Composite Transformations");
    myInit();
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```

Outputs: