

Lab Report

Jitendra Kumar , 1401CS19

Week 4
09/02/2017

Title

- ▶ Choose any primitive and apply all basic 2D transformations 1). OpenGL 2). MatLab

Procedure

OpenGL

- 1). Choose any primitive and apply all basic 2D transformations.
 - ▶ Create a C file and name it as *transformations.c*.
 - ▶ Following is the final code for all basic 2D transformations :

```
#include <stdio.h>
#include <math.h>
#include <GL/glut.h>
#include <stdlib.h>
#define PI 3.14159265
int flag, count; int i = 0 ; int j = 0 ; int k = 0 ;
double input_pts[10][2];
double final_pts[10][2];
double trans_matrix[3][3];

double x = 0 ;

void displayPolygon()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glLineWidth(3);
    glBegin(GL_LINES);
        glColor3f(1.0f, 1.0f, 1.0f);
        glVertex2f(0.0f,400.0f);
        glVertex2f(0.0f,-400.0f);
        glVertex2f(400.0f,0.0f);
        glVertex2f(-400.0f,0.0f);
    glEnd();

    glBegin(GL_LINE_LOOP);

    glColor3f(1.0f,0.0f,0.0f);

    glVertex3f(input_pts[0][0]/100.0, input_pts[0][1]/100.0, 1.0f);
    glVertex3f(input_pts[1][0]/100.0, input_pts[1][1]/100.0, 1.0f);
    glVertex3f(input_pts[2][0]/100.0, input_pts[2][1]/100.0, 1.0f);
    glVertex3f(input_pts[3][0]/100.0, input_pts[3][1]/100.0, 1.0f);
    glEnd();

    glBegin(GL_LINE_LOOP);

    glColor3f(0.0f,1.0f,0.0f);
    glVertex3f(final_pts[0][0]/100.0, final_pts[0][1]/100.0, 1.0f);
    glVertex3f(final_pts[1][0]/100.0, final_pts[1][1]/100.0, 1.0f);
    glVertex3f(final_pts[2][0]/100.0, final_pts[2][1]/100.0, 1.0f);
    glVertex3f(final_pts[3][0]/100.0, final_pts[3][1]/100.0, 1.0f);
    glEnd();

    glFlush();
}
```

```

        glutSwapBuffers();
    }

void matrix_multiplication()
{
    int i = 0 , j = 0 , k = 0 ;
    double a = 0 ; double b = 0 ;

    for(i = 0 ; i < 4 ; i++)
    {
        a = trans_matrix[0][0]*input_pts[i][0] + trans_matrix[0][1]*input_pts[i][1] + trans_matrix[0][2];
        b = trans_matrix[1][0]*input_pts[i][0] + trans_matrix[1][1]*input_pts[i][1] + trans_matrix[1][2];

        final_pts[i][0]=a;
        final_pts[i][1]=b;
    }
}

void translate(double x , double y)
{
    trans_matrix[0][0] = 1;
    trans_matrix[0][1] = 0;
    trans_matrix[0][2] = x;
    trans_matrix[1][0] = 0;
    trans_matrix[1][1] = 1;
    trans_matrix[1][2] = y;
    trans_matrix[2][0] = 0;
    trans_matrix[2][1] = 0;
    trans_matrix[2][2] = 1;
}

void scale_x_y(double sx, double sy)
{
    trans_matrix[0][0] = sx;
    trans_matrix[0][1] = 0;
    trans_matrix[0][2] = 0;
    trans_matrix[1][0] = 0;
    trans_matrix[1][1] = sy;
    trans_matrix[1][2] = 0;
    trans_matrix[2][0] = 0;
    trans_matrix[2][1] = 0;
    trans_matrix[2][2] = 1;
}

void reflectAroundX(void)
{
    trans_matrix[0][0] = 1;
    trans_matrix[0][1] = 0;
    trans_matrix[0][2] = 0;
    trans_matrix[1][0] = 0;
    trans_matrix[1][1] = -1;
    trans_matrix[1][2] = 0;
    trans_matrix[2][0] = 0;
    trans_matrix[2][1] = 0;
    trans_matrix[2][2] = 1;
}

void shearaboutY(double b)
{
    trans_matrix[0][0] = 1;
    trans_matrix[0][1] = 0;
    trans_matrix[0][2] = 0;
    trans_matrix[1][0] = b;
    trans_matrix[1][1] = 1;
    trans_matrix[1][2] = 0;
    trans_matrix[2][0] = 0;
    trans_matrix[2][1] = 0;
    trans_matrix[2][2] = 1;
}

void rotate(double a)
{
    x = PI / 180 ;
    a = a*x;
    trans_matrix[0][0] = cos(a);
    trans_matrix[0][1] = -1 * sin(a);
    trans_matrix[0][2] = 0;
    trans_matrix[1][0] = sin(a);

```

```

        trans_matrix[1][1] = cos(a);
        trans_matrix[1][2] = 0;
        trans_matrix[2][0] = 0;
        trans_matrix[2][1] = 0;
        trans_matrix[2][2] = 1;
    }

void transformPoints()
{
    int choice ;
    printf("\nEnter your choice:\n1. Translation\n2. Scaling\n3. Reflection\n4. Shear\n5. Rotate\n");
    printf("Your choice : ");
    scanf("%d",&choice);

    double x , y , scale_factor_x, scale_factor_y, shear_factor ,rotation_angle ;

    if(choice==1)
    {
        printf("Enter translate_x : ");
        scanf("%lf",&x);
        printf("Enter translate_y : ");
        scanf("%lf",&y);
        translate(x,y);
    }

    if(choice==2)
    {
        printf("Enter scale_factor_x : ");
        scanf("%lf",&scale_factor_x);
        printf("Enter scale_factor_y : ");
        scanf("%lf",&scale_factor_y);
        scale_x_y(scale_factor_x,scale_factor_y);
    }

    if(choice==3)
    {
        reflectAroundX();
    }

    if(choice==4)
    {
        printf("\nEnter the shearing Factor : ");
        scanf("%lf",&shear_factor);
        shearaboutY(shear_factor);
    }

    if(choice==5)
    {
        printf("\nEnter the rotation_angle : ");
        scanf("%lf",&rotation_angle);
        rotate(rotation_angle);
    }
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB);
    glutInitWindowSize(800, 800);
    for(i = 0 ; i < 10 ; i++)
    {
        for(j = 0 ; j < 3 ; j++)
        {
            input_pts[i][j]=0;
        }
    }
    printf("*****\nWe'll be applying transformations to a square polygon\n*****\n");
    printf("Enter the polygon coordinates : ");
    count = 4 ;

    for( i = 0 ; i < count ; i++)
    {
        scanf("%lf %lf",&input_pts[i][0],&input_pts[i][1]);
    }

    for(i = 0 ; i < 10 ; i++)
    {
        for(j = 0 ; j < 2 ; j++)

```

```

        {
            final_pts[i][j]=0;
        }
    }
    for(i = 0 ; i < 3 ; i++)
    {
        for(j = 0 ; j < 3 ; j++)
        {
            trans_matrix[i][j]=0;
        }
    }

    transformPoints();
    matrix_multiplication();
    glClearColor(1.0, 1.0, 1.0, 1.0);
    gluOrtho2D(-400, 400, -400, 400);
    glutCreateWindow("\nPolygon transformations");
    glutDisplayFunc(displayPolygon);
    glutMainLoop();

    return 0;
}

```

- Compile and run the executable file in terminal by typing in the following commands :
 - (a) `gcc transformations.c -lGL -lGLU -lglut -lm`
 - (b) `./a.out`

■ MatLab

1). Choose any primitive and apply all basic 2D transformations

- Open a new Script and construct a function transformations().
- Following is the Matlab Script Code for all basic 2D transformations.

```

function [] = transformations()
    disp("*****")

    input_matrix = [0 0 1; 0 10 1; 10 10 1; 10 0 1 ; 0 0 1];
    disp("\nEnter your choice:\n1. Translation\n2. Scaling\n3. ReflectAroundX\n4. ShearaboutY\n5. Rotate\n");
    choice = input("Your choice : ");

    temp_matrix = [1 0 0; 0 1 0 ; 0 0 1];
    identity_matrix = [1 0 0; 0 1 0 ; 0 0 1];

    if(choice <= 5)
        if(choice==1)
            translate_x = input('Enter translate_x:');
            translate_y = input('Enter translate_y:');
            temp_matrix = [1 0 translate_x ; 0 1 translate_y ; 0 0 1];
        end

        if(choice==2)
            scale_factor_x = input('Enter scale_factor_x:');
            scale_factor_y = input('Enter scale_factor_y:');
            temp_matrix = [scale_factor_x 0 0 ; 0 scale_factor_y 0 ; 0 0 1];
        end

        if(choice==3)
            temp_matrix = [1 0 0 ; 0 -1 0 ; 0 0 1];
        end

        if(choice==4)
            shear_factor = input('Enter the shearing_factor:');
            temp_matrix = [1 0 0 ; shear_factor 1 0 ; 0 0 1];
        end

        if(choice==5)
            theta= input('Enter the rotation_angle:');
            theta = (theta*3.14159265) / 180 ;
            %theta = theta*pi;
            temp_matrix = [cos(theta) -sin(theta) 0;
                            sin(theta) cos(theta) 0;
                            0 0 1];
        end
    else
        disp("\nYou choosed an invalid choice\n")
    end
end

```

```

end

trans = transpose(temp_matrix);
final = input_matrix*trans;
plot(input_matrix(:,1),input_matrix(:,2),'--','LineWidth', 3);
hold on
plot(final(:,1),final(:,2),'b-','LineWidth', 3);

```

Output

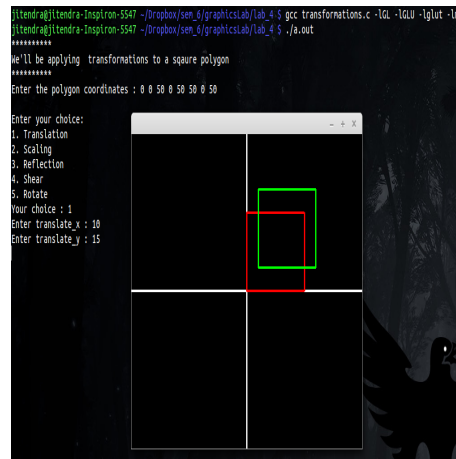


FIGURE 1 – translation in OpenGL

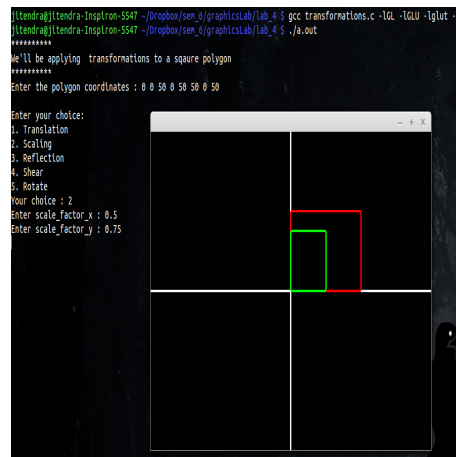


FIGURE 2 – Scaling in OpenGL

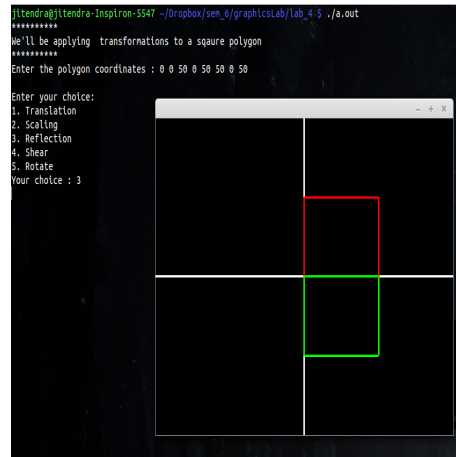


FIGURE 3 – reflection in OpenGL

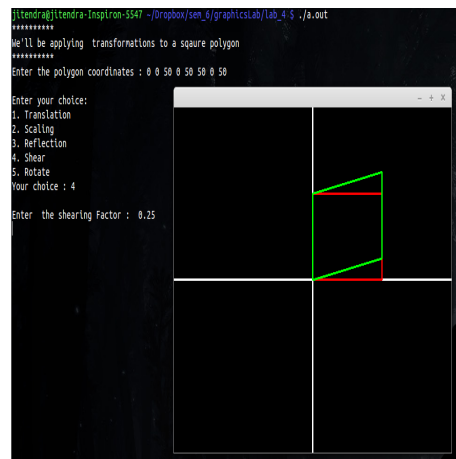


FIGURE 4 – Shear in OpenGL

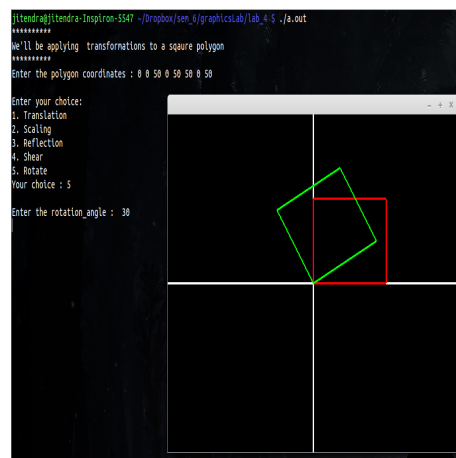


FIGURE 5 – rotation in openGL

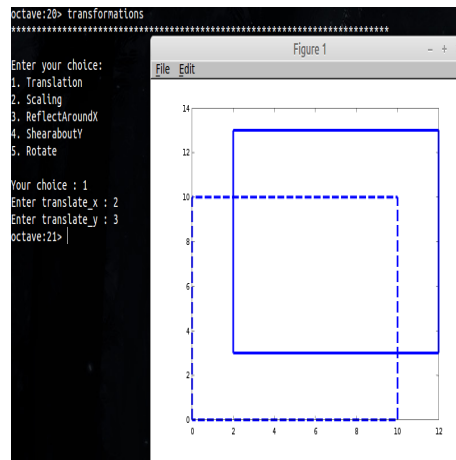


FIGURE 6 – translation in matlab

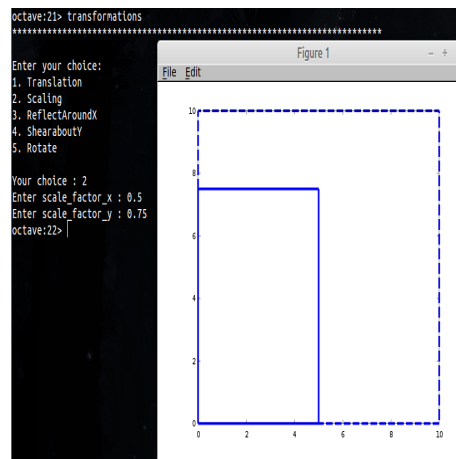


FIGURE 7 – scaling in Matlab

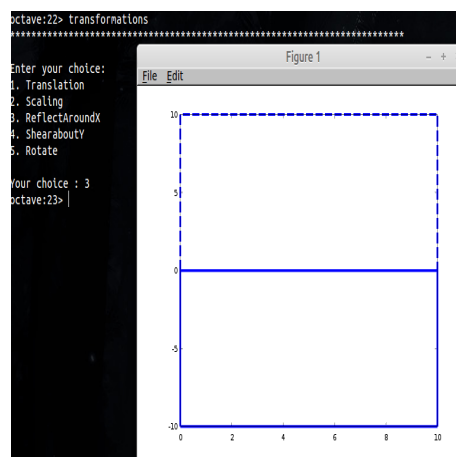


FIGURE 8 – reflection in Matlab

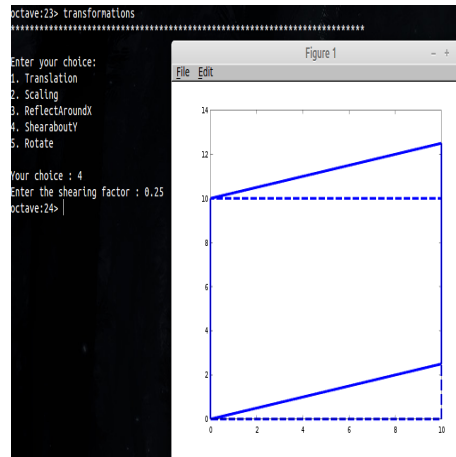


FIGURE 9 – shear in Matlab

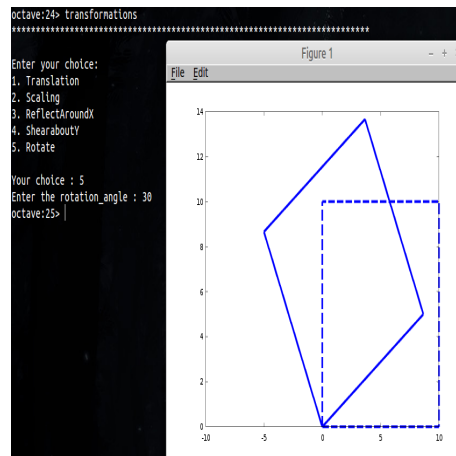


FIGURE 10 – Rotation in Matlab