Lab Report

Week 4

Ritobroto Maitra (1301CS50)

19/02/2016

Title

- Applying transformations to basic primitives (a line, a circle and a polygon) about an arbitraty point in space.

Procedure

We build up on the work done in the previous class and again use a combination of in built matrix transformations along with some hard coded matrix multiplications to do the requisite modeling transformations.

The transformation about an arbitrary point **P** involves the following :

1. Apply a translation operation
$$T_t$$
to the origin so that it is shifted to **P**
2. Apply your regular transformation operations
3. Apply
$$T_t^{-1}$$
to bring the system back to original cordinate system.

There for, to apply a transformation **T** to a model **X**, obtaining **X'**, we do the following

$$X' = T_t^{-1} T T_t X$$

Transformation Matrices

1. Scaling
$$\begin{bmatrix} 0.33 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
2. Reflection
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
3. Shear
$$\begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4. Rotation

$$\begin{bmatrix} cos(45) & -sin(45) & 0 \\ sin(45) & cos(45) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5. Translation

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0.1 \\ 0 & 0 & 1 \end{bmatrix}$$

6. Co-ordinate shift Let us take *(0.5,0.5)* as our arbitrary point. We need to apply the following transformation to shift origin.

$$\begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$$

Drawing the Basic Shapes in OpenGL

The following are the functional parts of the code required to plot the shapes.

Line

We create two points and draw a line between them.

```
glVertex3f(0.0, 0.0, 0.0);
glVertex3f(0.5, -0.5, 0);
```

Circle

We create a 0.5 radius circle using sin(i) and cos(i) values as follows.

```
for(i=0;i<10*3.14;i+=0.0001){
  glVertex3f(cos(i)/2, sin(i)/2, 0.0);
}
```

Polygon

The polygon is created using the following lines.

```
  glBegin(GL_LINES);
  glVertex3f(0.5, 0.5, 0.0);
  glVertex3f(0.0, 0.0, 0.0);
  glVertex3f(0.0, 0.0, 0.0);
  glVertex3f(-0.5, 0.5, 0.0);
  glVertex3f(-0.5, 0.5, 0.0);
  glVertex3f(-0.7, -0.5, 0.0);
  glVertex3f(-0.7, -0.5, 0.0);
  glVertex3f(-0.2, -0.6, 0.0);
  glVertex3f(-0.2, -0.6, 0.0);
  glVertex3f(0.5, 0.5, 0.0);
```

Transformation Code in OpenGL

The transformation is fed by an input of options which is dealt by the following switch case mechanism. Observe **line 06** and **line 44**.

```
01 static void Key(unsigned char key, int x, int y)
02 {
03   GLfloat m[] = {1, 0, 0, 0,  3, 1, 0, 0,   0, 0, 1, 0,   0, 0, 0, 1};
04   GLfloat n[] = {1, 0, 0, 0,  -3, 1, 0, 0,  0, 0, 1, 0,   0, 0, 0, 1};
05   glTranslatef(0.5, 0.5, 0);
06   glMatrixMode(GL_MODELVIEW);
07     switch (key) {
08       case '1':
09       glScalef(0.33, 1, 1);
10       glutPostRedisplay();
11       break;
12         case '2':
13       glScalef(3, 1, 1);
14       glutPostRedisplay();
15       break;
16     case '3':
17       glRotatef(45, 0, 0, 1);
18       glutPostRedisplay();
19       break;
20     case '4':
21       glMultMatrixf(m);
22       glutPostRedisplay();
23       break;
24     case '5':
25       glMultMatrixf(n);
26       glutPostRedisplay();
27       break;
28     case '6':
29       glScalef(-1, 1, 1);
30       glutPostRedisplay();
31       break;
32     case '7':
33       glTranslatef(0, 0.1, 0);
34       glutPostRedisplay();
35       break;
36     case '8':
37       glTranslatef(0, -0.1, 0);
38       glutPostRedisplay();
39       break;
40         case 27:
41     exit(0);
42     }
43 }
44 glTranslatef(0.5, 0.5, 0);
```

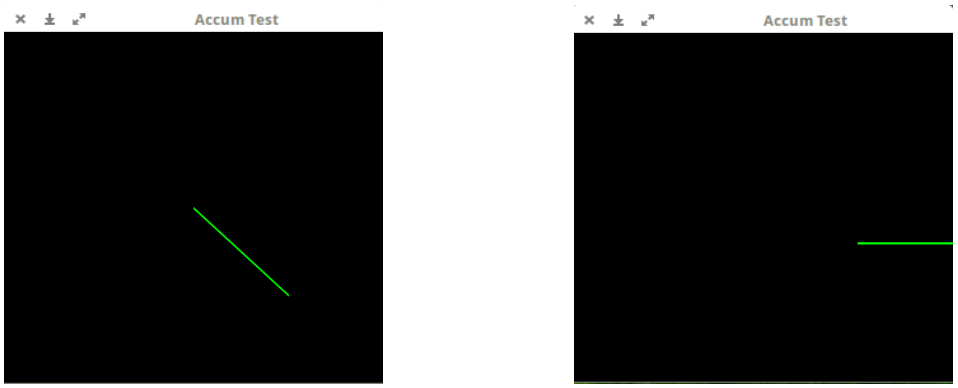
Transformation Code in MATLAB

Again observe line numbers - **1, 2, 33** and **34** to see how the origin shift operation is working.

```
01 trans = [1, 0, 2; 0, 1, 3; 0, 0, 1];
02 rtrans = [1, 0, -2; 0, 1, -3; 0, 0, 1];
03
04 prompt = 'Input';
05 in=-1;
06 while in ~= 0
07     in = input(prompt);
08     if in==1
09         % scaling
10         T = [2, 0, 0; 0, 1.5, 0; 0, 0, 1];
11     elseif in==-1
12         % rev scaling
13         T = [0.5, 0, 0; 0, 2/3, 0; 0, 0, 1];
14     elseif in==2
15         % reflection
16         T = [-1, 0, 0; 0, 1, 0; 0, 0, 1];
17     elseif in==3
18         % shear
19         T = [1, 1, 0; 0, 1, 0; 0, 0, 1];
20     elseif in==-3
21         % rev shear
22         T = [1, -1, 0; 0, 1, 0; 0, 0, 1];
```

```
    elseif in==4
        % rotation
        T = [cosd(45), -sind(45), 0; sind(45), cosd(45), 0; 0, 0, 1];
    elseif in==5
        % translation
        T = [1, 0, 1;0, 1, 0.5; 0, 0, 1];
     elseif in==-5
        % rev translation
        T = [1, 0, -1;0, 1, -0.5; 0, 0, 1];
    end
    pts = trans*T*rtrans*pts;
    plot(pts(1,:), pts(2,:), 'r*-');
    axis([-10, 10, -10, 10]);
    display(pts)
end
```

Examples

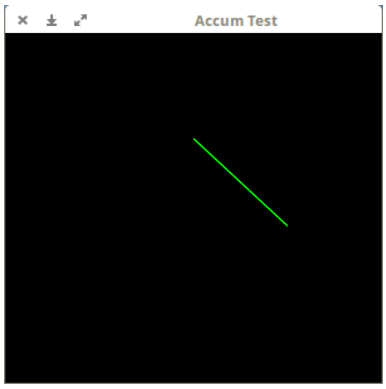


(a) Initial Line (b) Line : Rotation

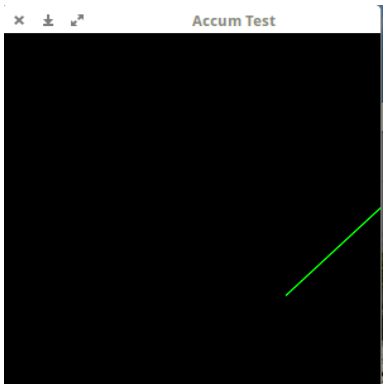
FIGURE 1 – Transformations on Line : Rotation

(a) Line : Shear (b) Line Scale

FIGURE 2 – Transformations on Line : Shear, Scale

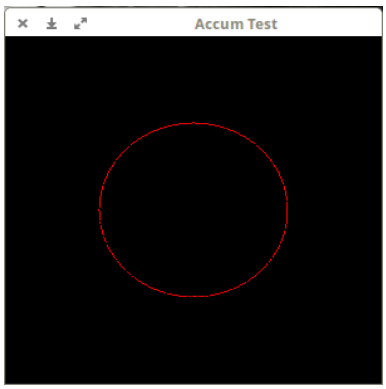


(a) Line : Translate

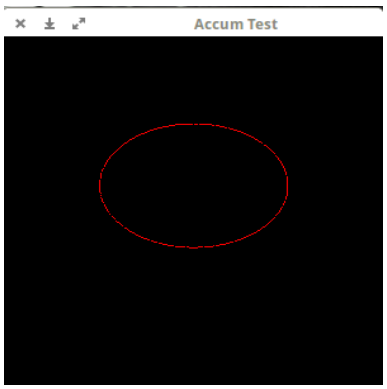


(b) Line : Reflect

FIGURE 3 – Transformations on Line : Translate, Reflect

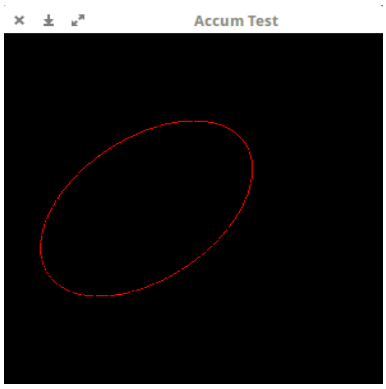


(a) Initial Circle

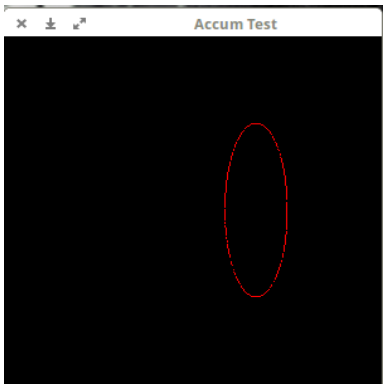


(b) Circle : Rotation

FIGURE 4 – Transformations on Circle : Rotation

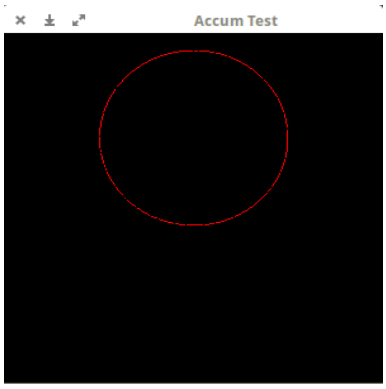


(a) Circle : Shear

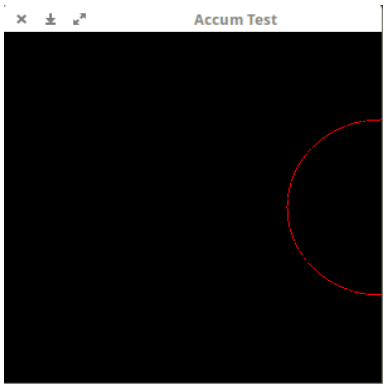


(b) Circle Scale

FIGURE 5 – Transformations on Circle : Shear, Scale

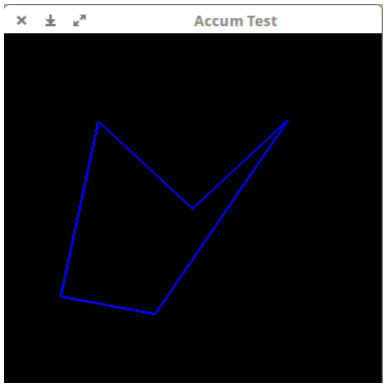


(a) Circle : Translate

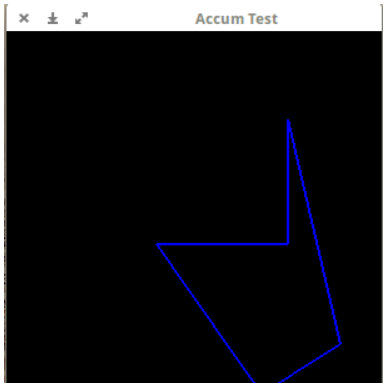


(b) Circle : Reflect

FIGURE 6 – Transformations on Circle : Translate, Reflect

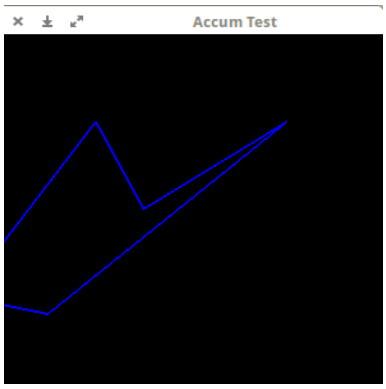


(a) Initial polygon

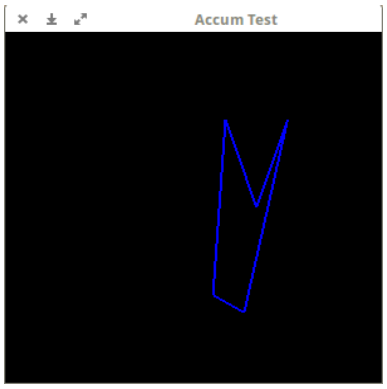


(b) polygon : Rotation

FIGURE 7 – Transformations on polygon : Rotation

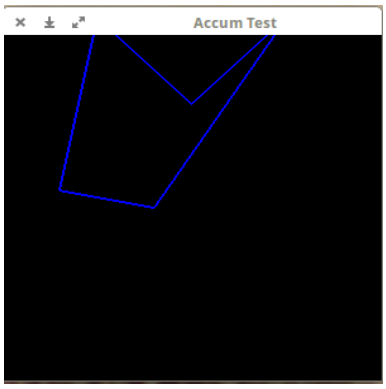


(a) Polygon : Shear

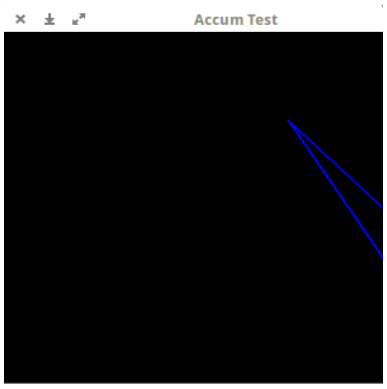


(b) Polygon : Scale

FIGURE 8 – Transformations on Polygon : Shear, Scale

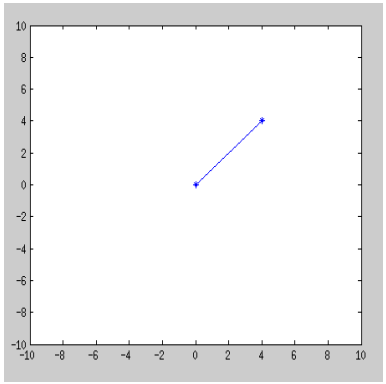


(a) Polygon : Translate

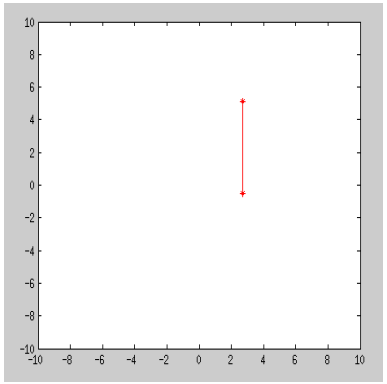


(b) Polygon : Reflect

FIGURE 9 – Transformations on Polygon : Translate, Reflect

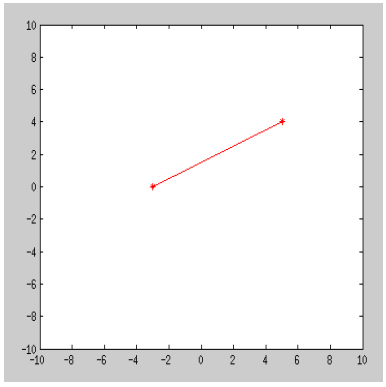


(a) Initial Line

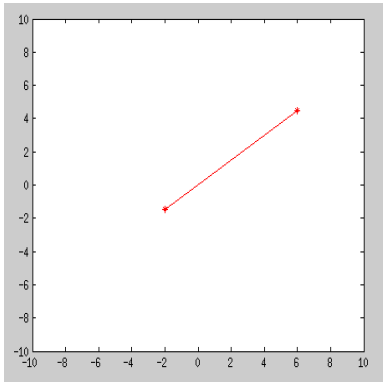


(b) Line : Rotation

FIGURE 10 – Transformations on Line : Rotation

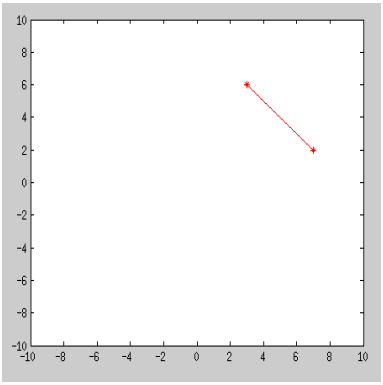


(a) Line : Shear

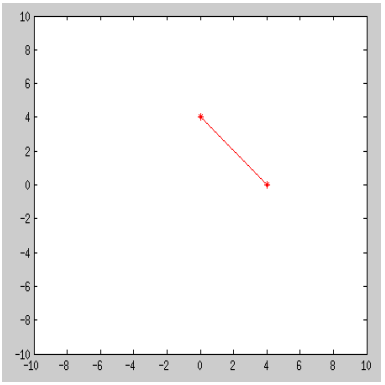


(b) Line Scale

FIGURE 11 – Transformations on Line : Shear, Scale

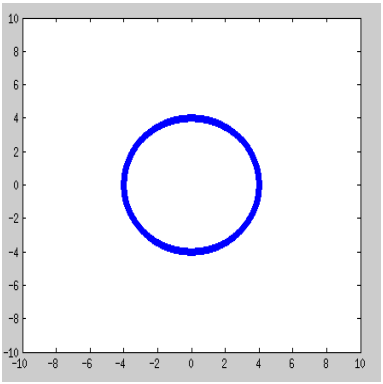


(a) Line : Translate

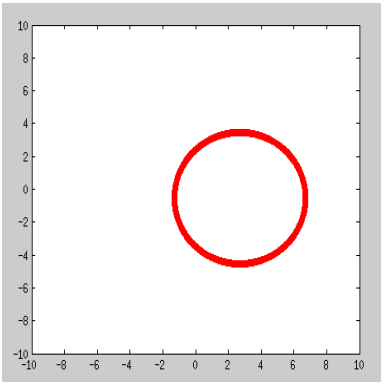


(b) Line : Reflect

FIGURE 12 – Transformations on Line : Translate, Reflect

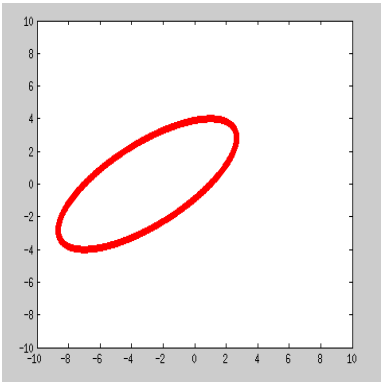


(a) Initial Circle

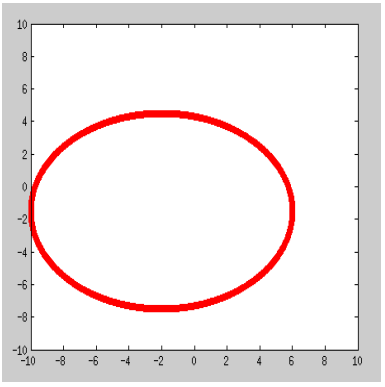


(b) Circle : Rotation

FIGURE 13 – Transformations on Circle : Rotation

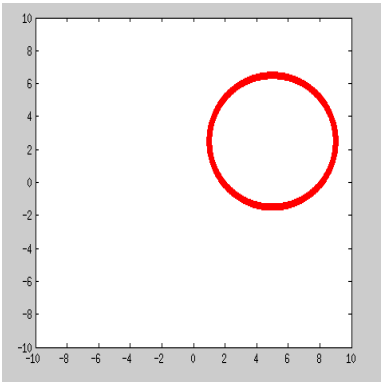


(a) Circle : Shear

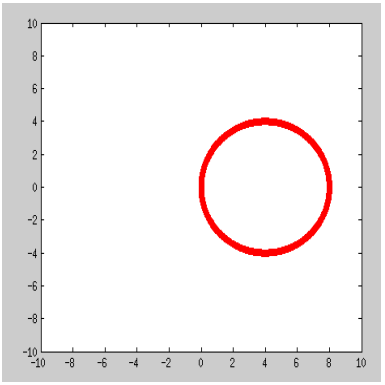


(b) Circle : Scale

FIGURE 14 – Transformations on Circle : Shear, Scale

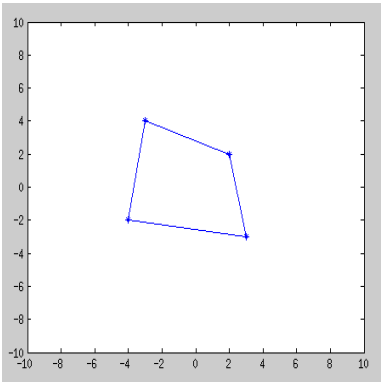


(a) Circle : Translate

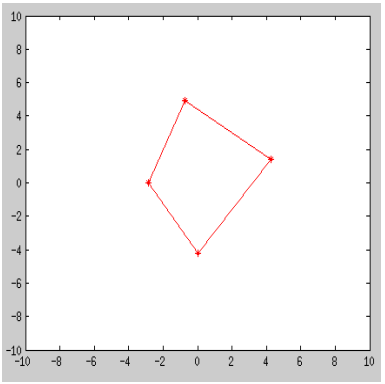


(b) Circle : Reflect

FIGURE 15 – Transformations on Circle : Translate, Reflect

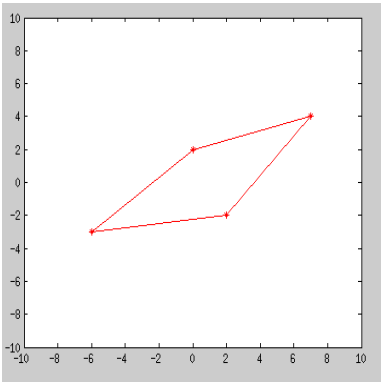


(a) Initial polygon

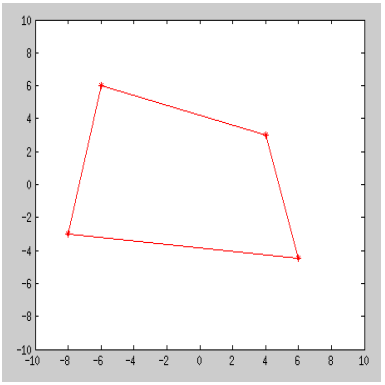


(b) Polygon : Rotation

FIGURE 16 – Transformations on Polygon : Rotation

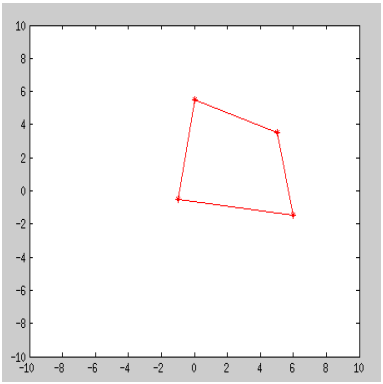


(a) Polygon : Shear

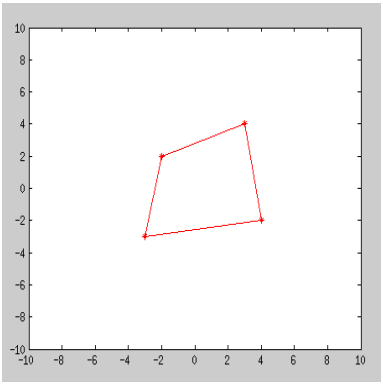


(b) Polygon : Scale

FIGURE 17 – Transformations on Polygon : Shear, Scale



(a) Polygon : Translate



(b) Polygon : Reflect

FIGURE 18 – Transformations on Polygon : Translate, Reflect