Challenge #4 - 2D Shapes and Transformations

Computer Graphics

Presented by:

Santiago Zubieta / Jose Cortes

Teacher:

Helmuth Trefftz

Universidad EAFIT

Briefing

We'll work on transformations. These will be applied to certain shapes. Shapes are defined as a list of 2D points. Each point will be considered as a single column Matrix, and will be multiplied with the desired transformation Matrix.

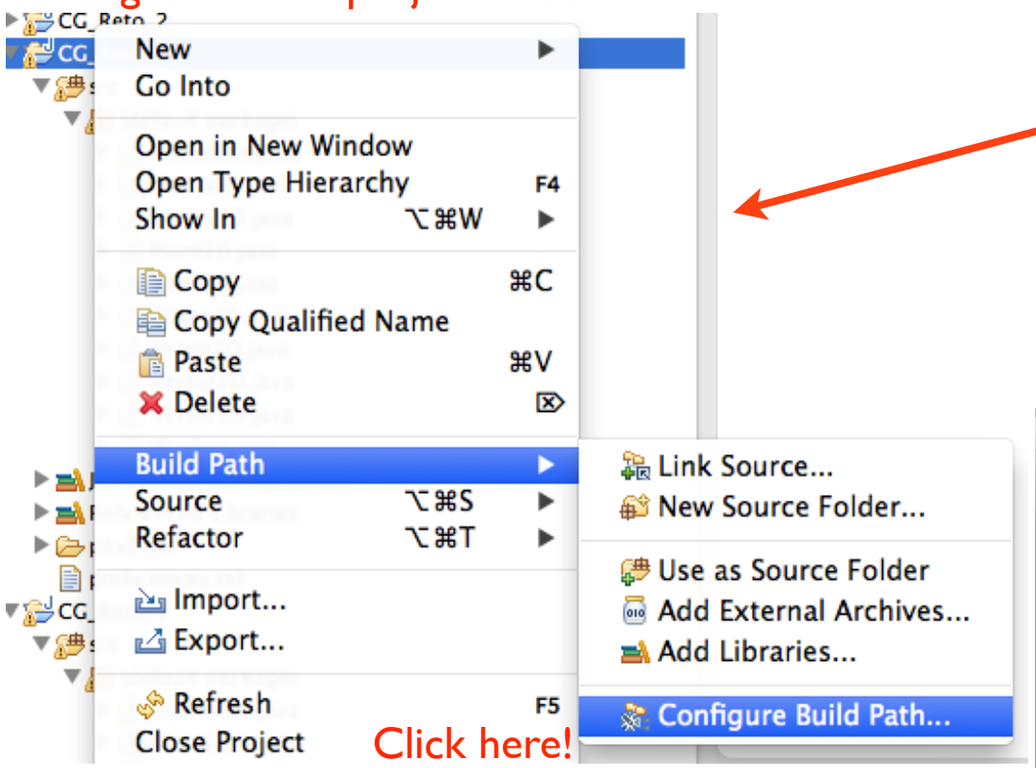
Transformation Matrix

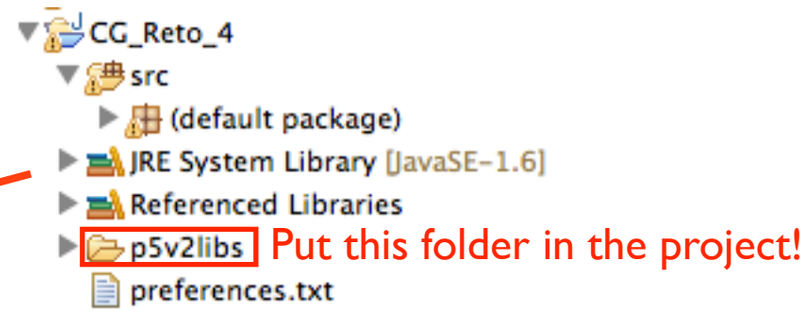
Point that will be Transformed

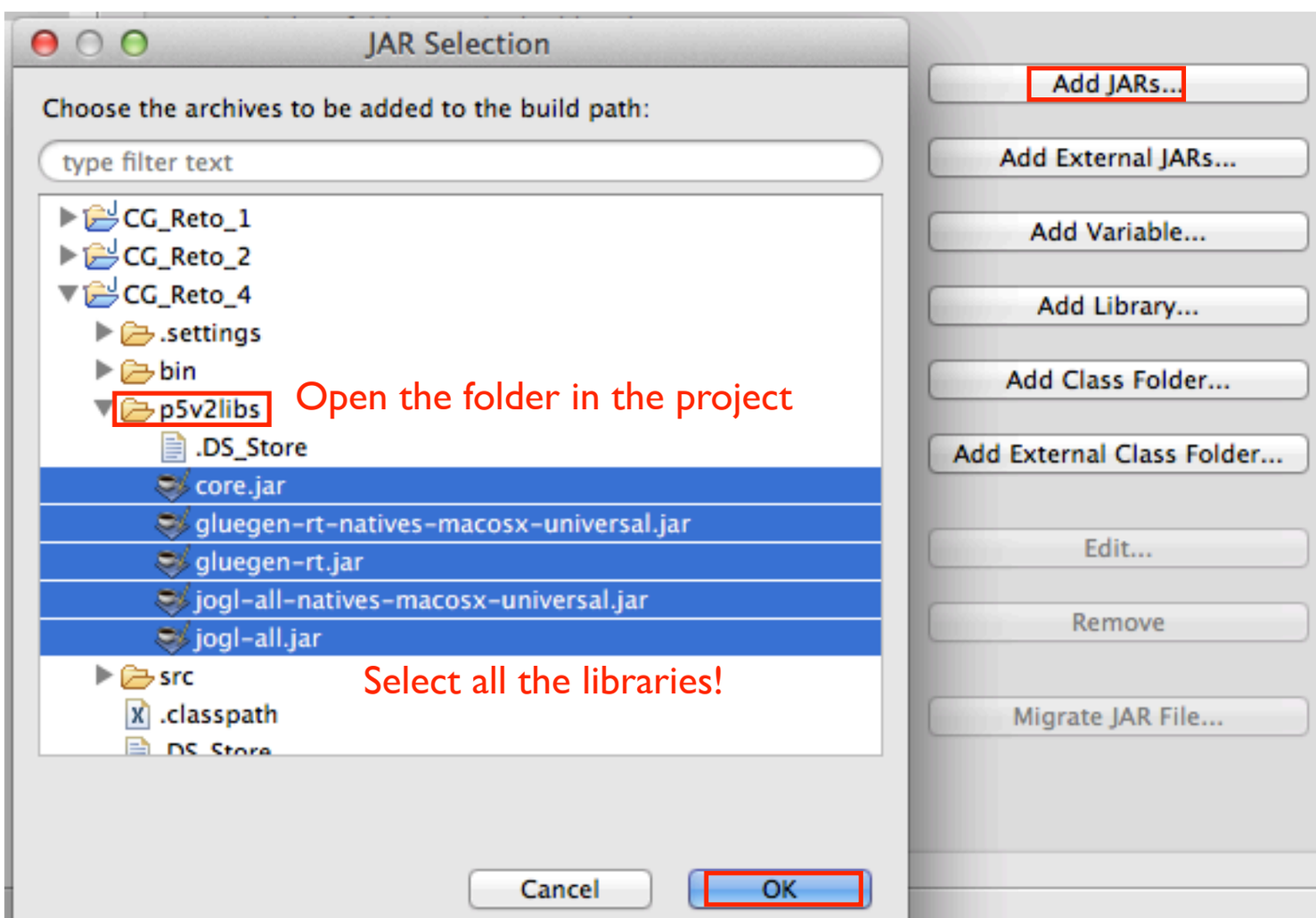
```
public static Point2D multiplyMatrixAndPoint(Matrix2D mat, Point2D p) {
    // It uses a 3D matrix to make us of Homogeneous Coordinates, to be
    // ..able to translate with matrix operations
    float pt[] = { 0, 0, 0 };
    float vals[] = { p.x, p.y, 1.0f };
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            pt[i] += mat.m[i][j] * vals[j];
        }
    }
    return new Point2D(pt[0], pt[1]);
}
```

Installing Processing

Right-click on project folder!







Processing is a framework built on top of Java which allows for awesome graphical capabilities, it has its own transformations but for the purposes of this course the transformation part we're doing on our own.

Translating

From the course material

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

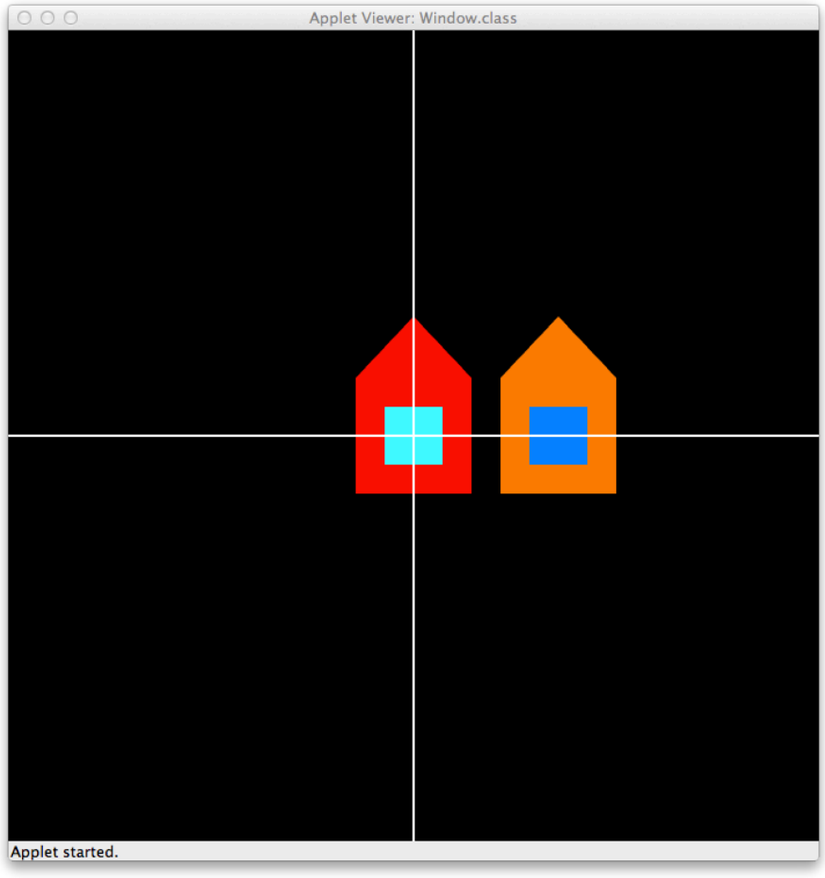
With Homogeneous Coordinates

$$x' = x + d_x, y' = y + d_y$$

$$\mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \mathbf{T} = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{P} + \mathbf{T}$$

Regular Operations



125 Pixels Translating

```
public void translate(float transX, float transY) {
    float matrix[][] = {
            { 1, 0, transX },
            { 0, 1, transY },
            { 0, 0,      1 }
    };
    Matrix2D translateMatrix = new Matrix2D(matrix);
    Point2D translatedPoint = Matrix2D.multiplyMatrixAndPoint(translateMatrix, this);
    this.x = translatedPoint.x;
    this.y = translatedPoint.y;
}
```

Method in the Point2D Object

Scaling

From the course material

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

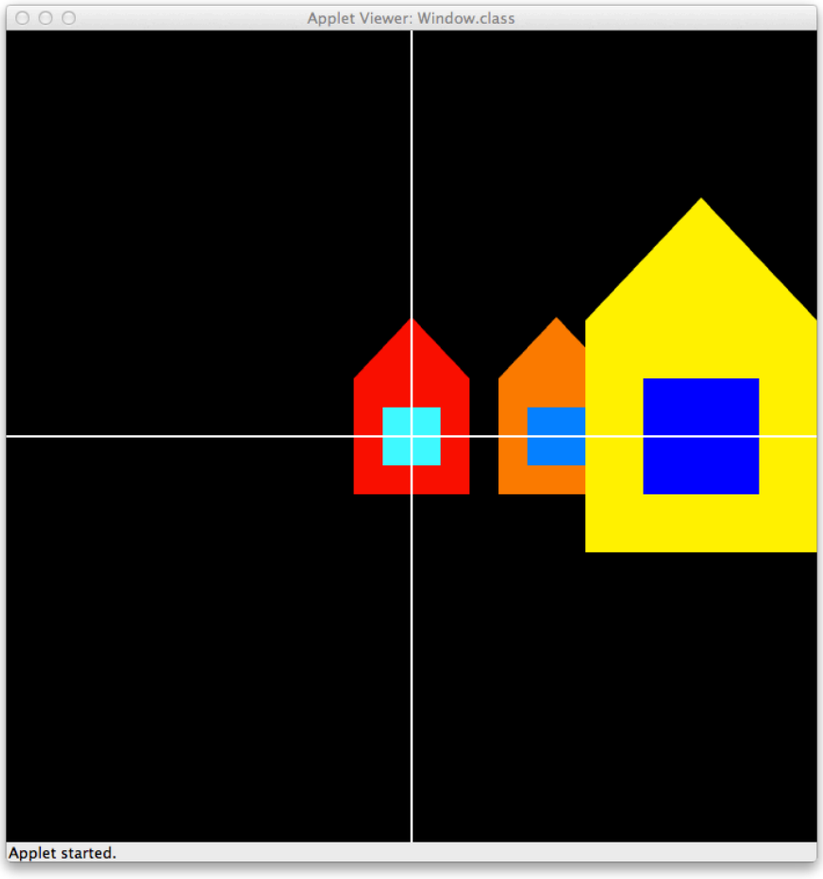
With Homogeneous Coordinates

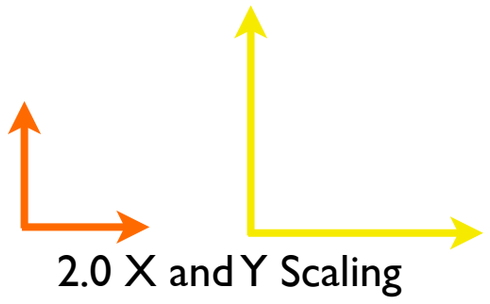
$$x' = s_x \times x, \; y' = s_y \times y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P'} = \mathbf{S} \times \mathbf{P}$$

Regular Operations





2.0 X and Y Scaling

```
public void scale(float scaleX, float scaleY) {
    float matrix[][] = {
            { scaleX,      0, 0 },
            {      0, scaleY, 0 },
            {      0,      0, 1 }
    };
    Matrix2D scaleMatrix = new Matrix2D(matrix);
    Point2D scaledPoint = Matrix2D.multiplyMatrixAndPoint(scaleMatrix, this);
    this.x = scaledPoint.x;
    this.y = scaledPoint.y;
}
```

Method in the Point2D Object

Rotating

From the course material

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

With Homogeneous Coordinates

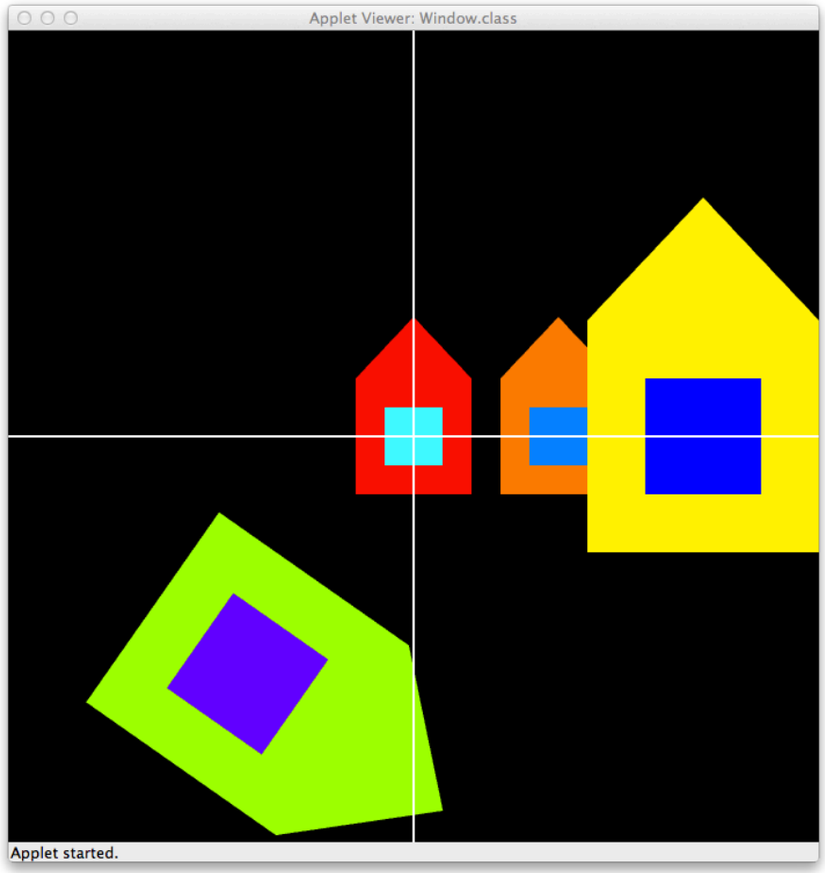
$$x' = x \times \cos\theta - y \times \sin\theta$$

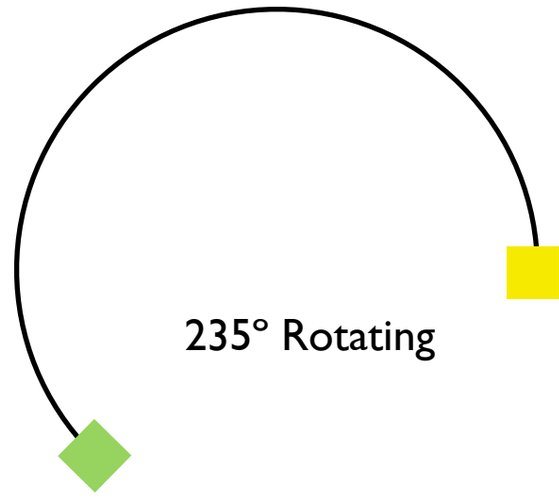
$$y' = x \times \sin\theta + y \times \cos\theta$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{P'} = \mathbf{R} \times \mathbf{P}$$

Regular Operations



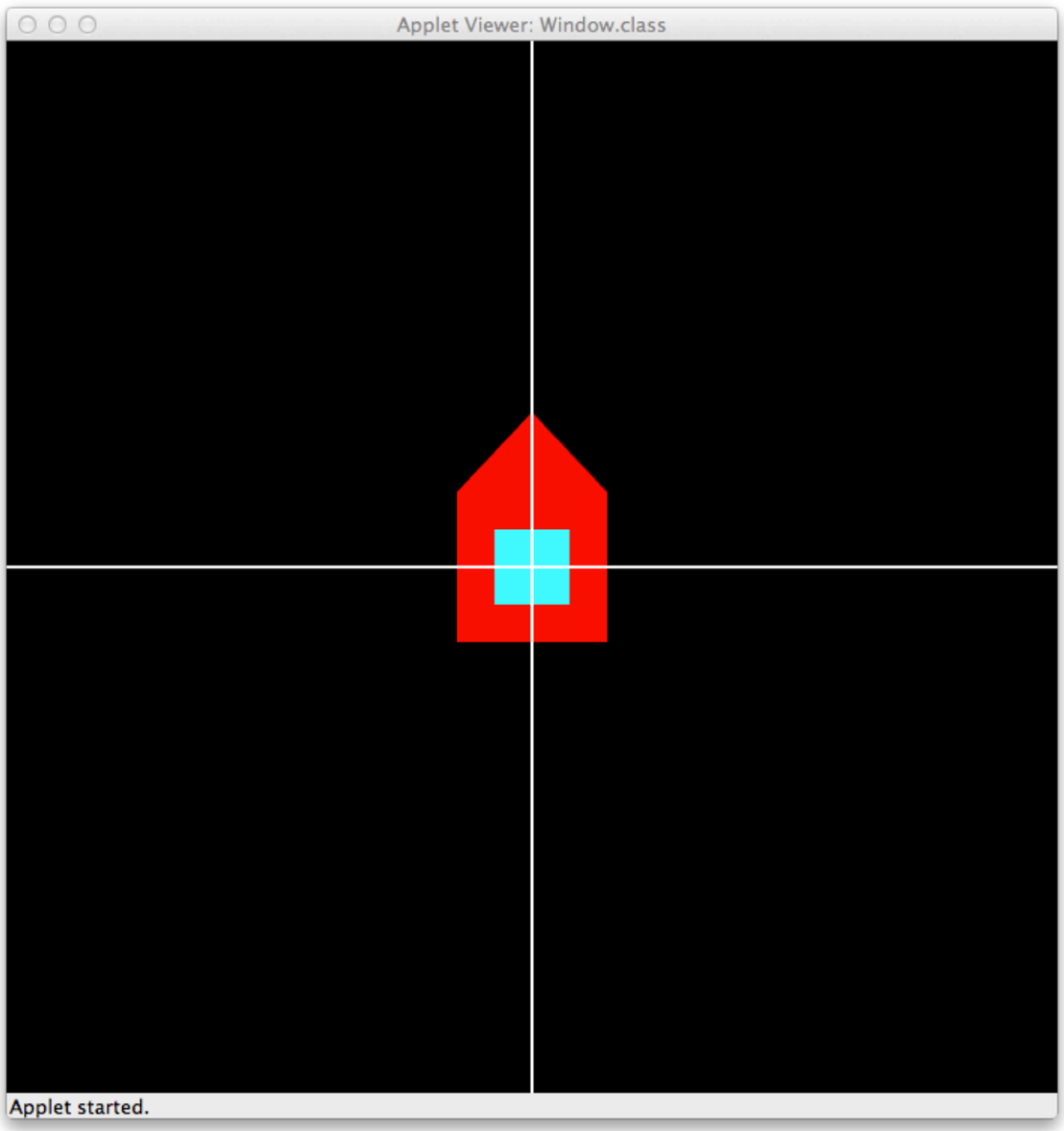


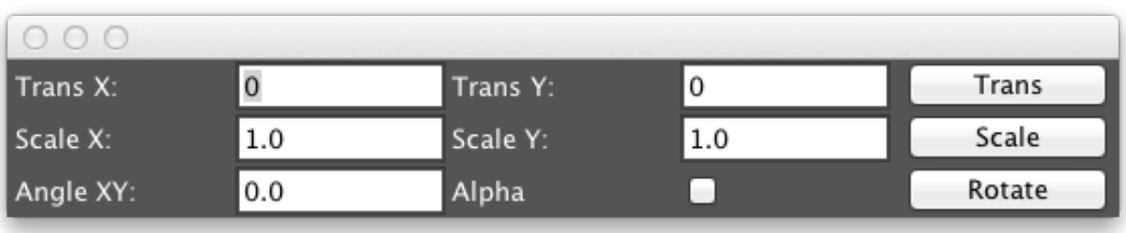
235° Rotating

```
public void rotate(float angle) {
    // Convert angle from Deg to Rad
    angle *= Math.PI / 180;
    float cos = (float) Math.cos(angle);
    float sin = (float) Math.sin(angle);
    float matrix[][] = {
            { cos, -sin, 0 },
            { sin,  cos, 0 },
            {   0,    0, 1 }
    };
    Matrix2D rotationMatrix = new Matrix2D(matrix);
    Point2D rotatedPoint = Matrix2D.multiplyMatrixAndPoint(rotationMatrix, this);
    this.x = rotatedPoint.x;
    this.y = rotatedPoint.y;
}
```

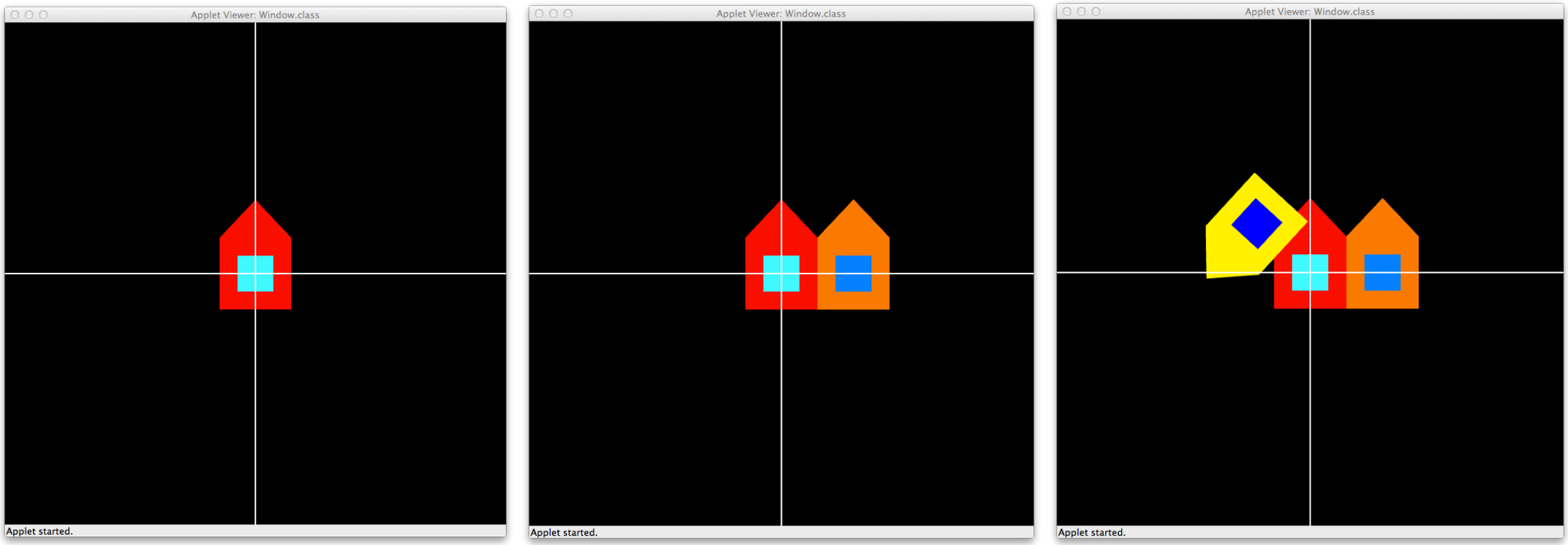
Method in the Point2D Object

The Application

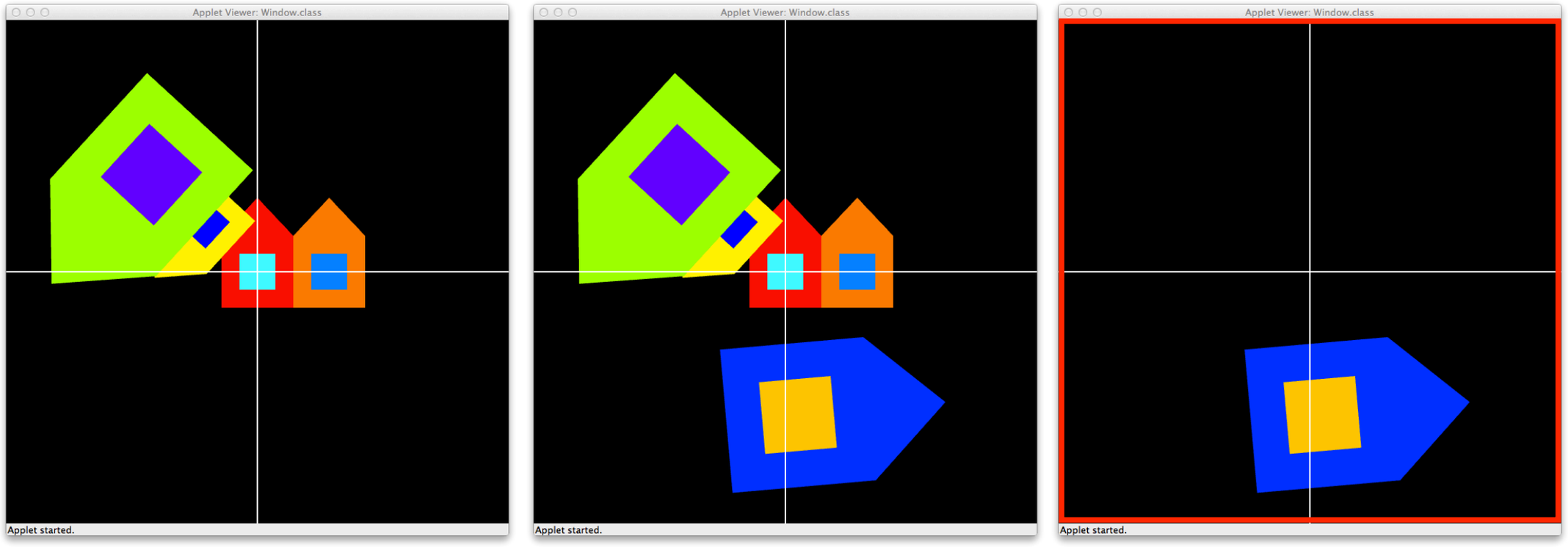




The Canvas

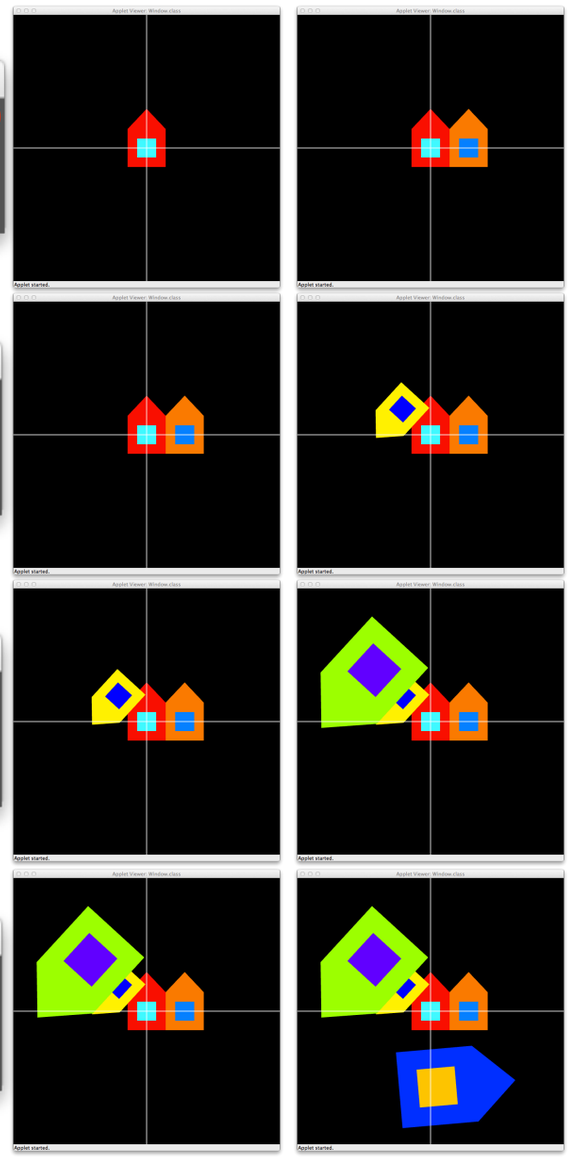


As you've seen this far, with each transformation, the color of the shape changes in an orderly fashion, but also the previous transformations are not removed, that is, unless you click on the canvas, then all the previous are deleted and only the most recent one remains! Try using this to make curious creative drawings with the House

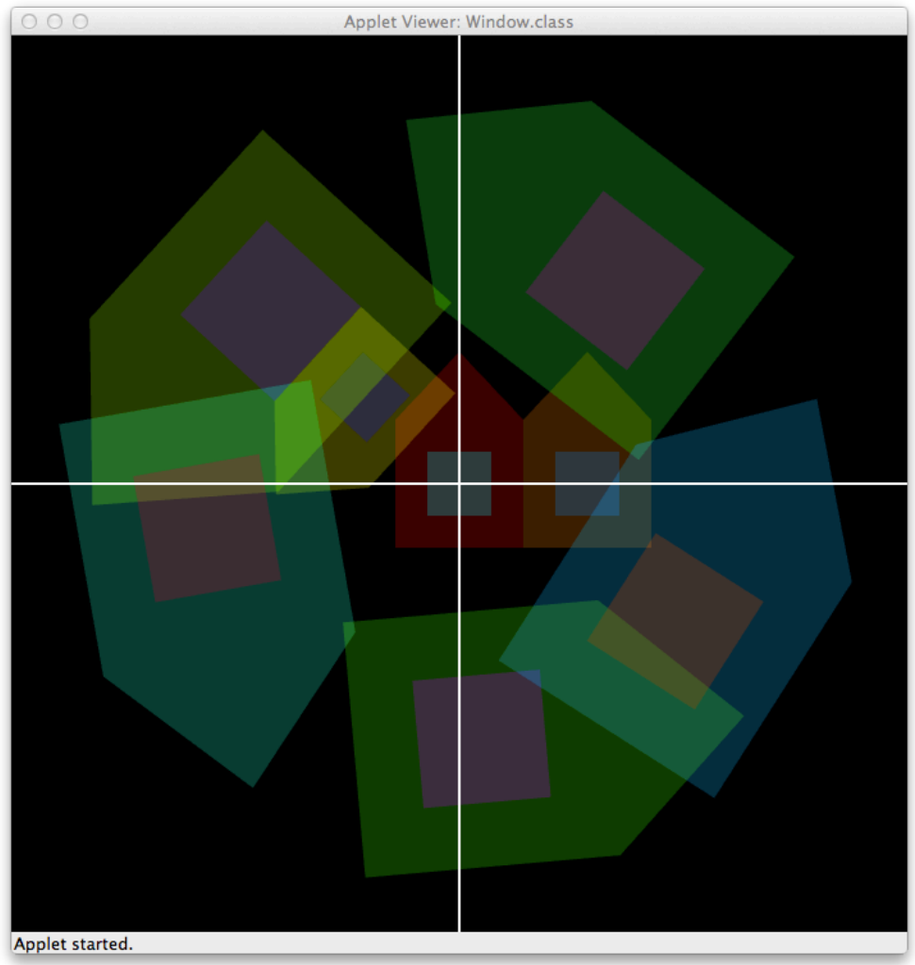


The Controls

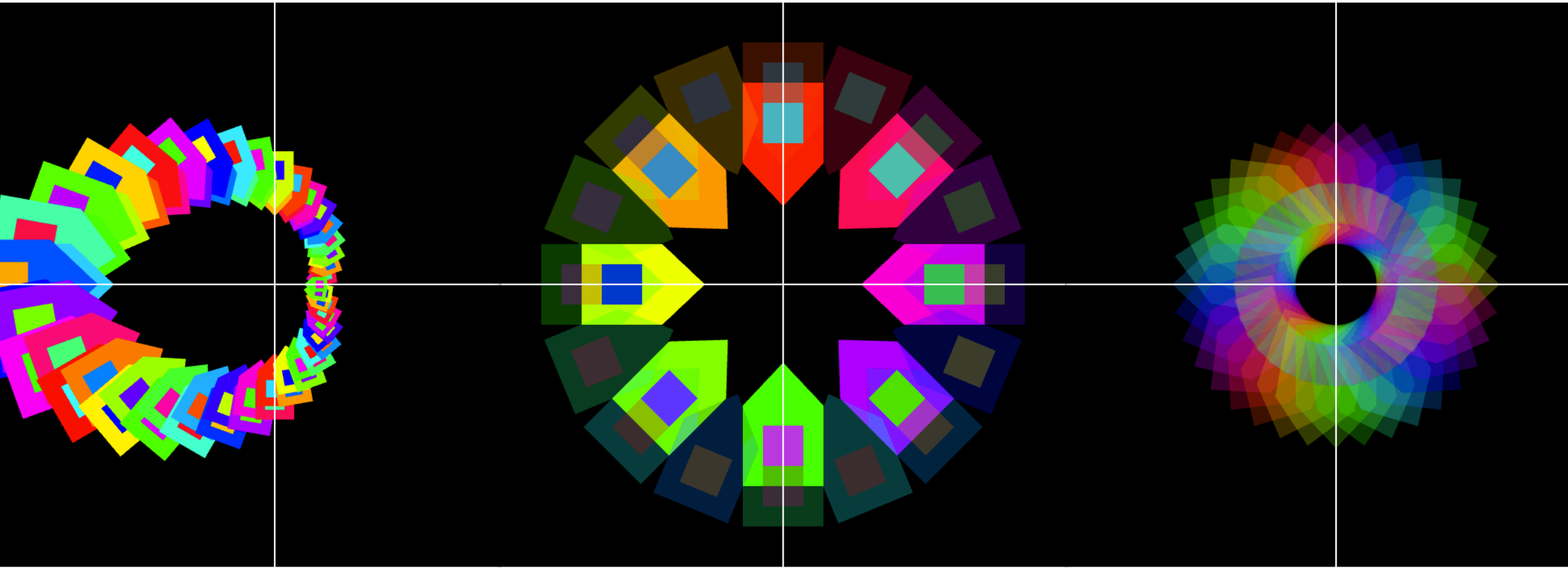
You can input different kind of transformation values in the controls. Only one kind of transformation will happen at a given time so you can see what each kind of transformation does. There's also a control for the transparency of the drawn shapes, so they are either 50% or 100% solid.



And with the Alpha enabled:



Some ‘Art’



Thanks for your time!

More challenges to follow!