Challenge #5 - 3D 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 3D points. Each point will be considered as a single column Matrix, and will be multiplied with the desired transformation Matrix.

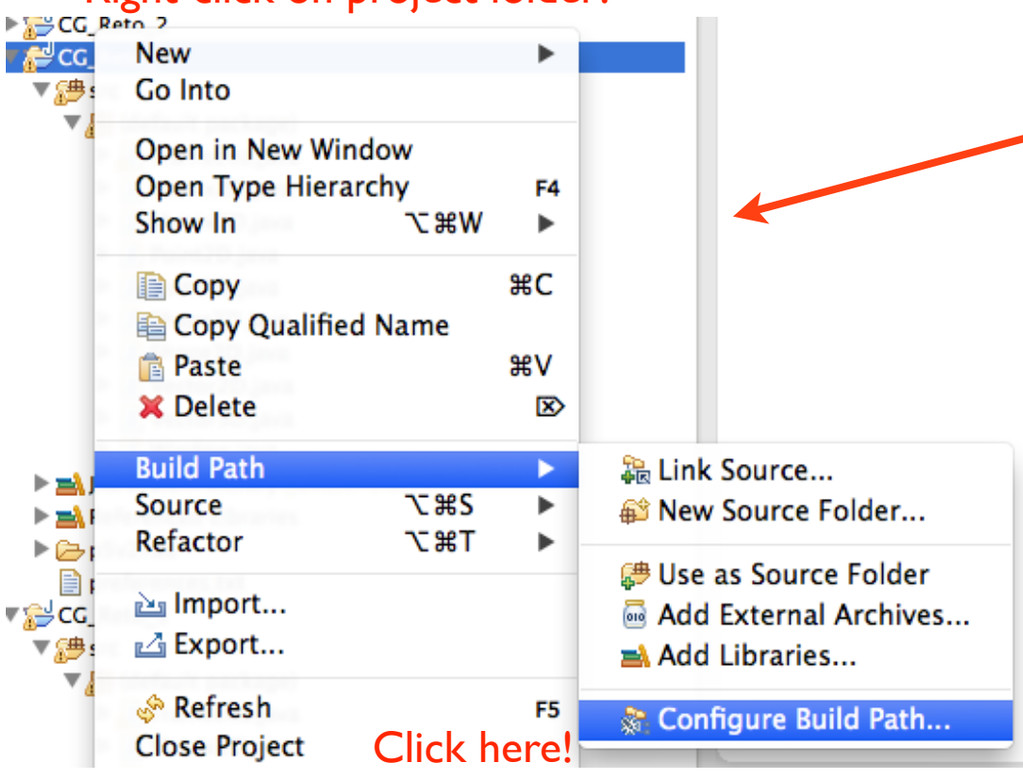
Transformation Matrix

Point that will be Transformed

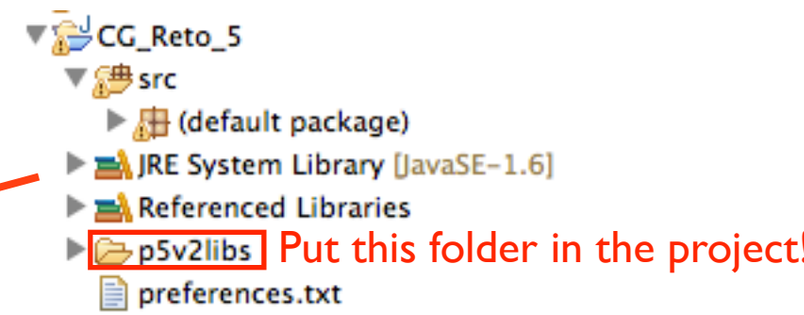
```
public static Point3D multiplyMatrixAndPoint(Matrix3D mat, Point3D p) {
    // It uses a 4D matrix to make us of Homogeneous Coordinates, to be
    // ..able to translate with matrix operations
    float pt[] = { 0, 0, 0, 0 };
    float vals[] = { p.x, p.y, p.z, 1.0f };
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            pt[i] += mat.m[i][j] * vals[j];
        }
    }
    return new Point3D(pt[0], pt[1], pt[2]);
}
```

Installing Processing

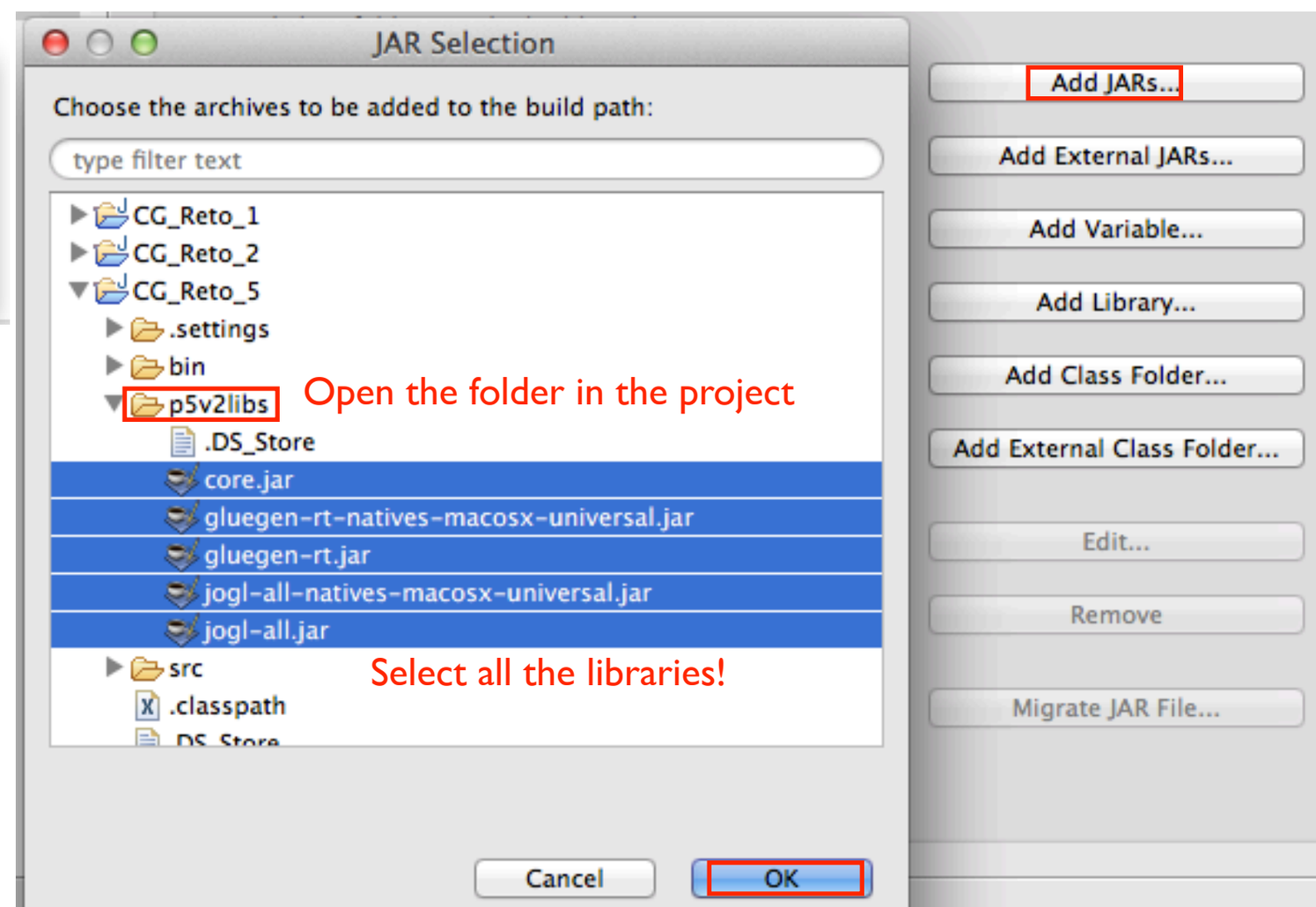
Right-click on project folder!



Click here!



Put this folder in the project!



Open the folder in the project

Select all the libraries!

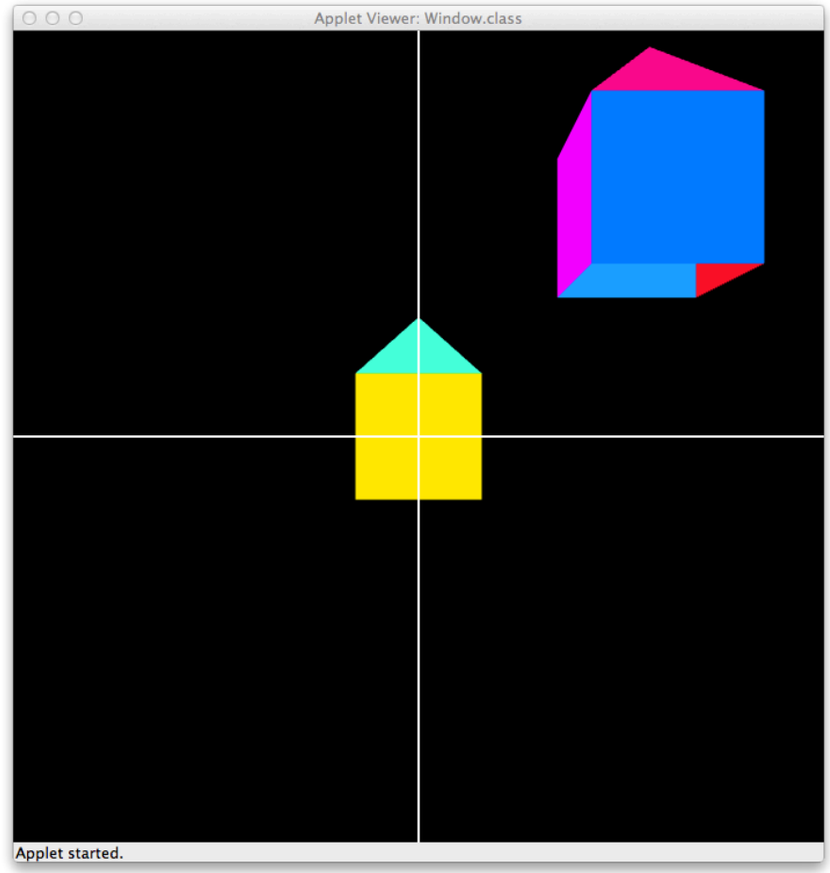
Processing is a framework built on top of Java which allows for awesome graphical capabilities, it has its own transformations but for the purposes of this course the transformation part we're doing on our own.

Translating

From the course material

$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With Homogeneous Coordinates



100 Pixels Translating in X →
100 Pixels Translating in Y ↑
100 Pixels Translating in Z ↘

```
public void translate(float transX, float transY, float transZ) {
    float matrix[][] = {
            { 1, 0, 0, transX },
            { 0, 1, 0, transY },
            { 0, 0, 1, transZ },
            { 0, 0, 0,      1 }
    };
    Matrix3D translateMatrix = new Matrix3D(matrix);
    Point3D translatedPoint = Matrix3D.multiplyMatrixAndPoint(translateMatrix, this);
    this.x = translatedPoint.x;
    this.y = translatedPoint.y;
    this.z = translatedPoint.z;
}
```

Method in the Point3D Object

Scaling

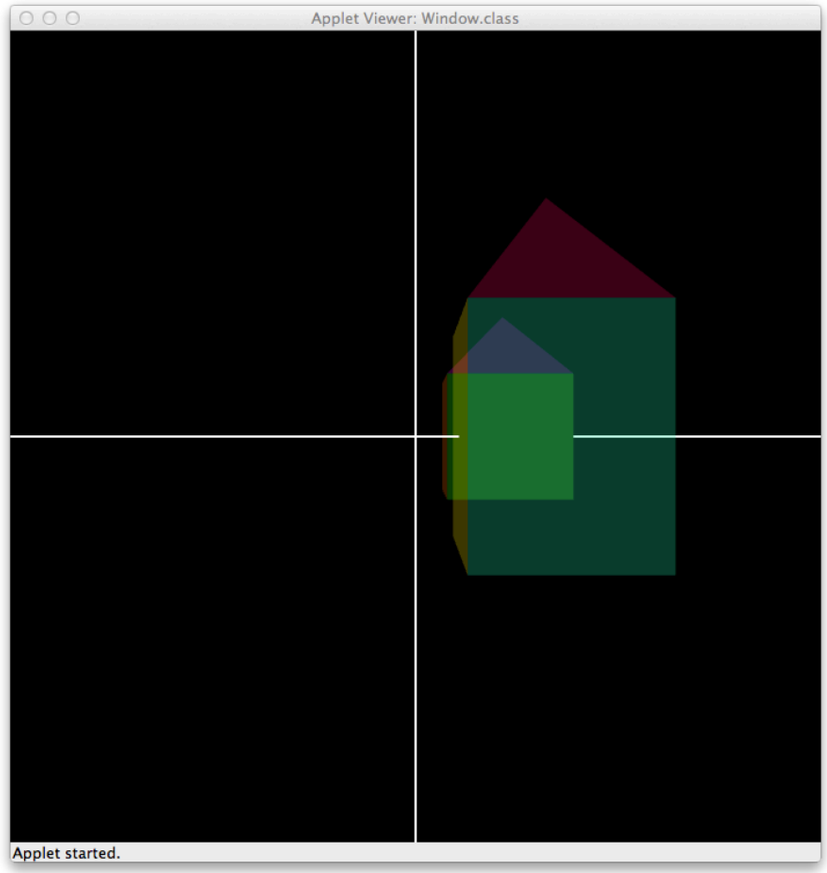
From the course material

$$S(s_x, s_y, s_z)$$

$$=$$

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With Homogeneous Coordinates



1.5 X Scaling
2.0 Y Scaling
2.0 Z Scaling

```
public void scale(float scaleX, float scaleY, float scaleZ) {
    float matrix[][] = {
            { scaleX,      0,      0, 0 },
            {      0, scaleY,      0, 0 },
            {      0,      0, scaleZ, 0 },
            {      0,      0,      0, 1 }
    };
    Matrix3D scaleMatrix = new Matrix3D(matrix);
    Point3D scaledPoint = Matrix3D.multiplyMatrixAndPoint(scaleMatrix, this);
    this.x = scaledPoint.x;
    this.y = scaledPoint.y;
    this.z = scaledPoint.z;
}
```

Method in the Point3D Object

Rotating XY

From the course material

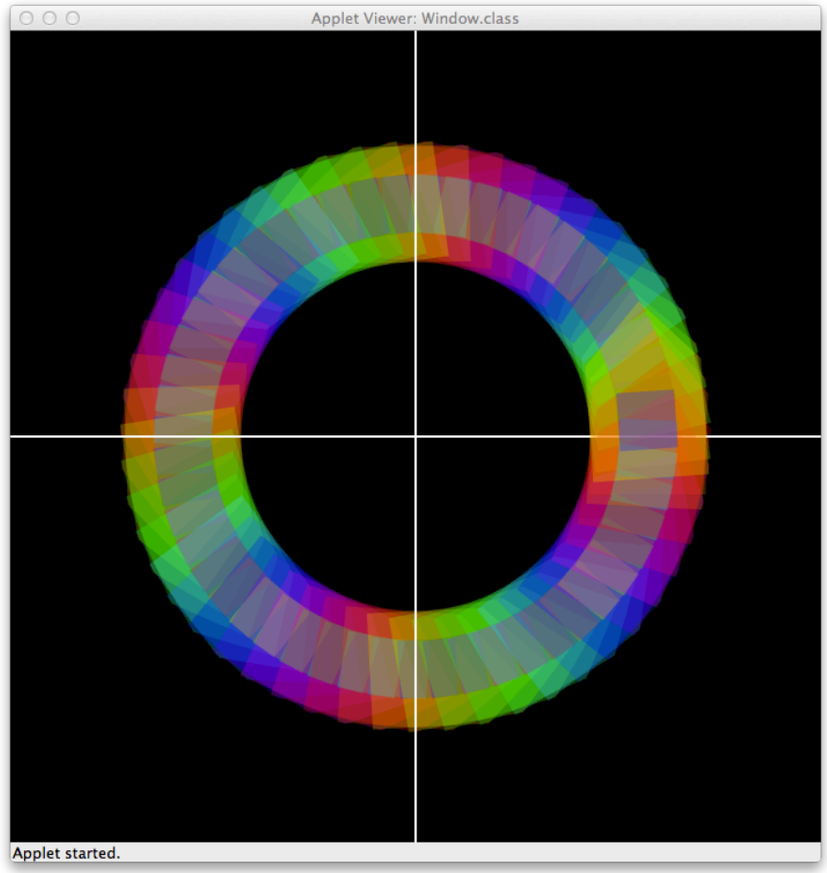
XY plane = Around Z axis

$$R_z(\theta)$$

$$=$$

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With Homogeneous Coordinates



Rotating in XY 7° several times

```
public void rotateXY(float angleXY){
    // Convert angle from Deg to Rad
    angleXY *= Math.PI / 180;
    float cos = (float) Math.cos(angleXY);
    float sin = (float) Math.sin(angleXY);
    float matrixXY[][] = {
            { cos, -sin, 0, 0 },
            { sin,  cos, 0, 0 },
            {   0,    0, 1, 0 },
            {   0,    0, 0, 1 }
    };
    Matrix3D rotationMatrixXY = new Matrix3D(matrixXY);
    Point3D rotatedPoint = Matrix3D.multiplyMatrixAndPoint(rotationMatrixXY, this);
    this.x = rotatedPoint.x;
    this.y = rotatedPoint.y;
    this.z = rotatedPoint.z;
}
```

Method in the Point3D Object

Rotating YZ

From the course material

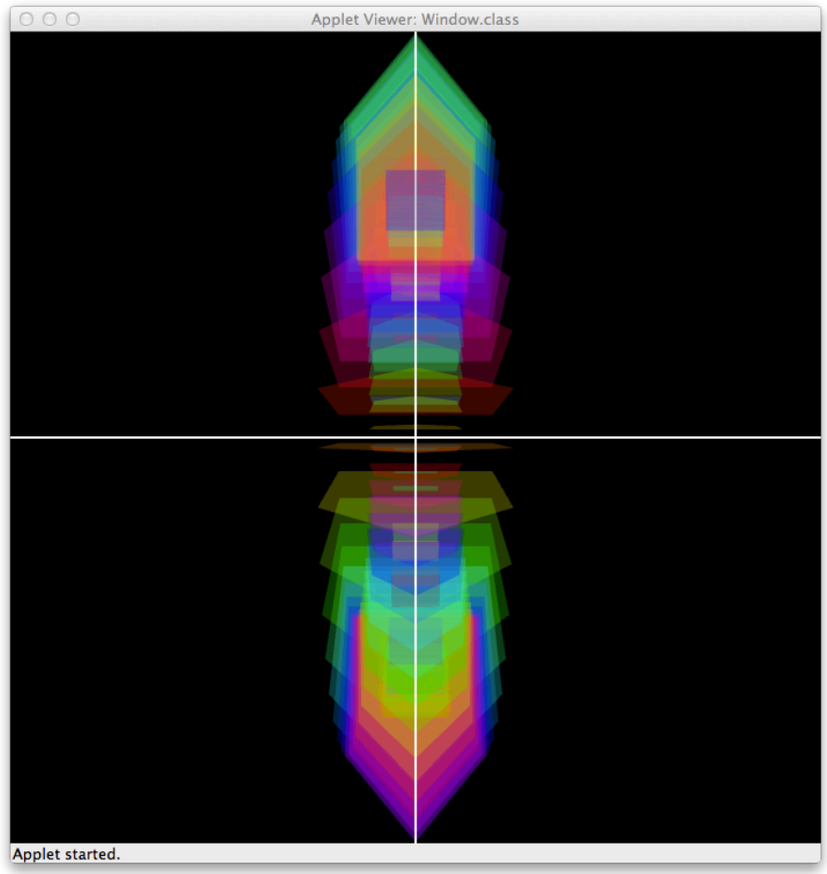
YZ plane = Around X axis

$$R_x(\theta)$$

$$=$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With Homogeneous Coordinates



Rotating in YZ 7° several times

```
public void rotateYZ(float angleYZ){
    // Convert angle from Deg to Rad
    angleYZ *= Math.PI / 180;
    float cos = (float) Math.cos(angleYZ);
    float sin = (float) Math.sin(angleYZ);
    float matrixYZ[][] = {
            { 1,   0,    0, 0 },
            { 0, cos, -sin, 0 },
            { 0, sin,  cos, 0 },
            { 0,   0,    0, 1 }
    };
    Matrix3D rotationMatrixYZ = new Matrix3D(matrixYZ);
    Point3D rotatedPoint = Matrix3D.multiplyMatrixAndPoint(rotationMatrixYZ, this);
    this.x = rotatedPoint.x;
    this.y = rotatedPoint.y;
    this.z = rotatedPoint.z;
}
```

Method in the Point3D Object

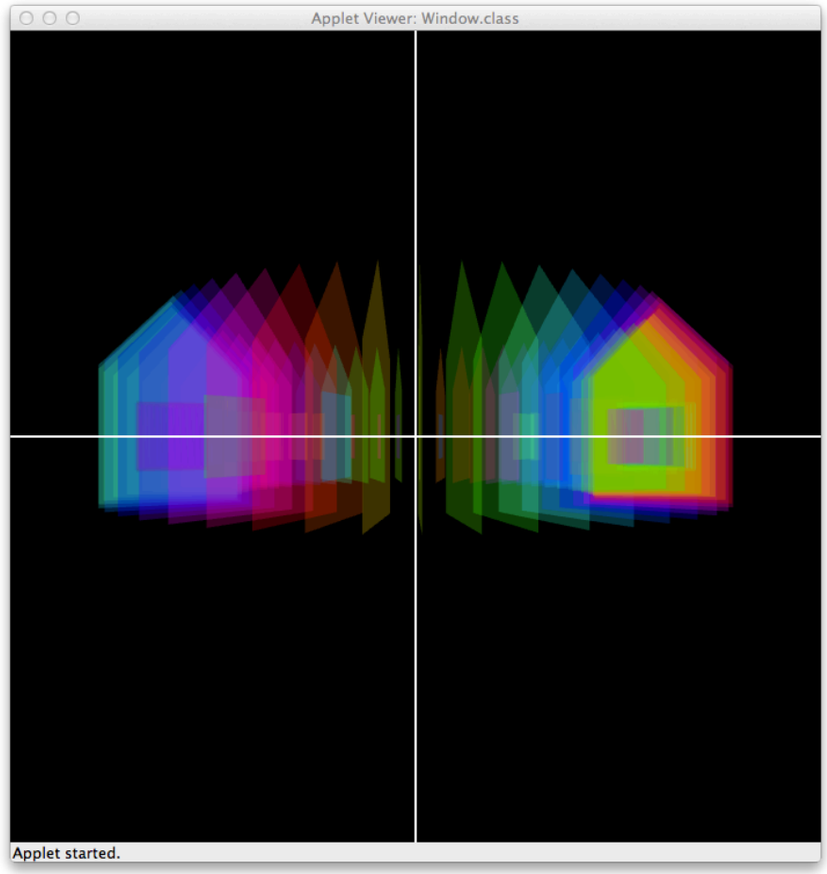
Rotating ZX

From the course material

ZX plane = Around Y axis

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With Homogeneous Coordinates

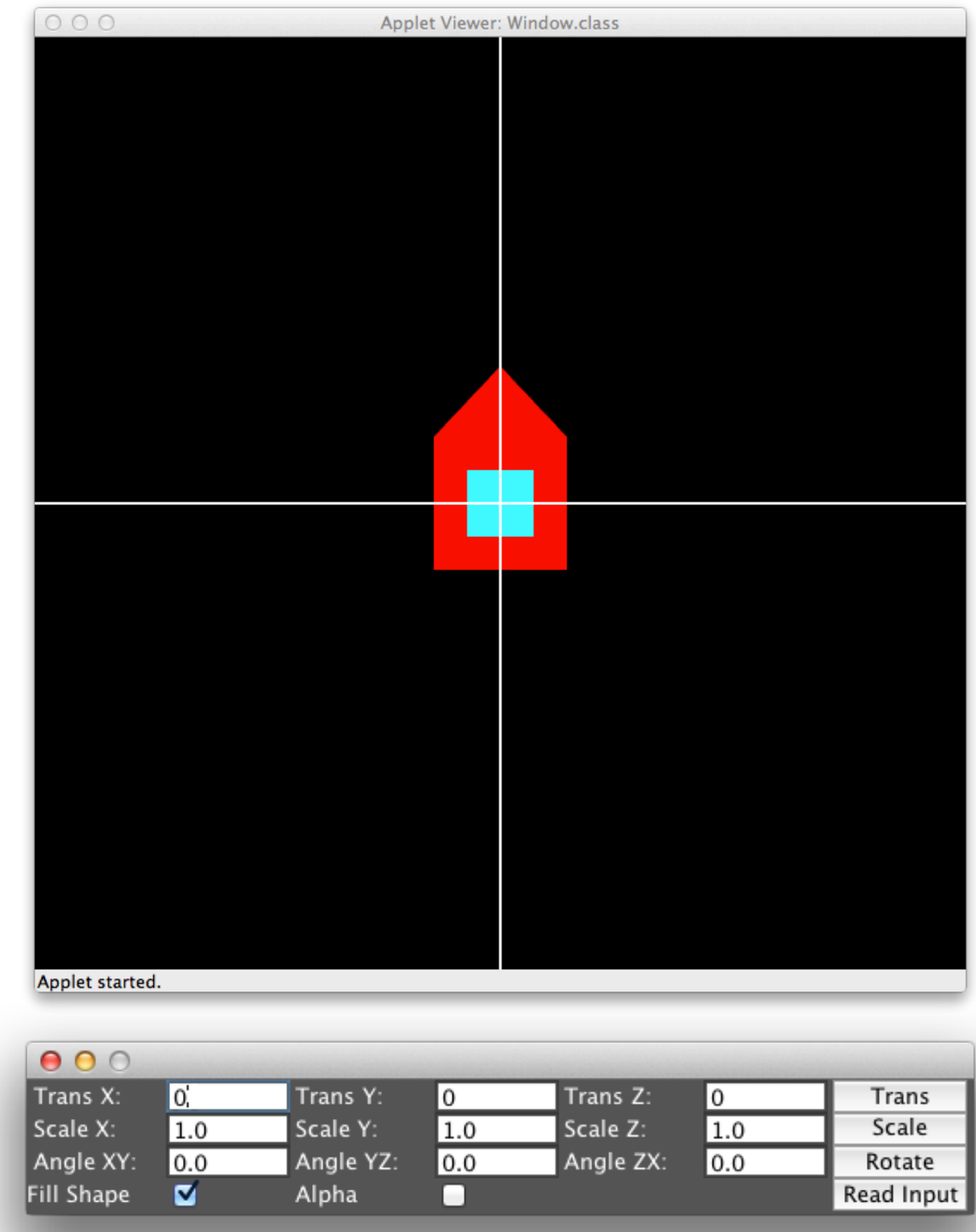


Rotating in ZX 7° several times

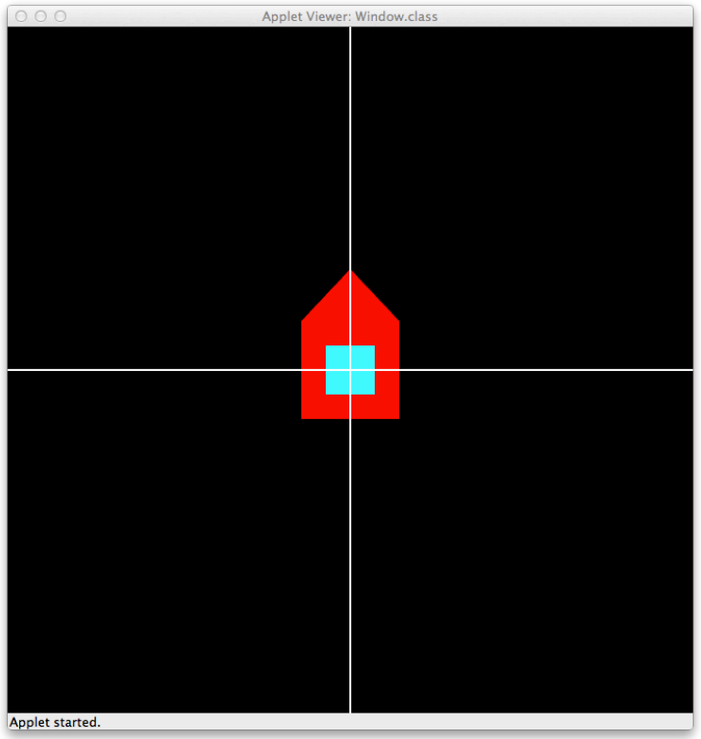
```
public void rotateXY(float angleXY){
    // Convert angle from Deg to Rad
    angleXY *= Math.PI / 180;
    float cos = (float) Math.cos(angleXY);
    float sin = (float) Math.sin(angleXY);
    float matrixXY[][] = {
            { cos, -sin, 0, 0 },
            { sin,  cos, 0, 0 },
            {   0,    0, 1, 0 },
            {   0,    0, 0, 1 }
    };
    Matrix3D rotationMatrixXY = new Matrix3D(matrixXY);
    Point3D rotatedPoint = Matrix3D.multiplyMatrixAndPoint(rotationMatrixXY, this);
    this.x = rotatedPoint.x;
    this.y = rotatedPoint.y;
    this.z = rotatedPoint.z;
}
```

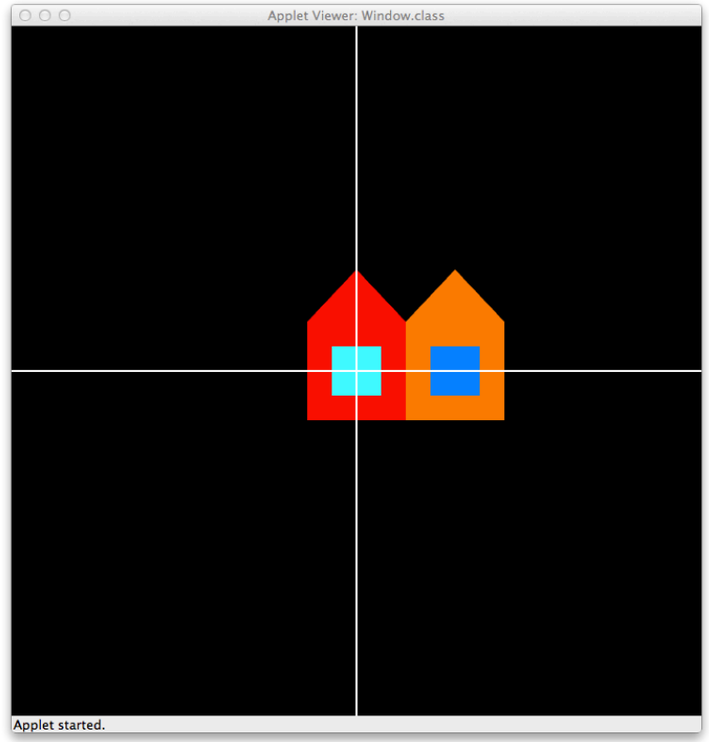
Method in the Point3D Object

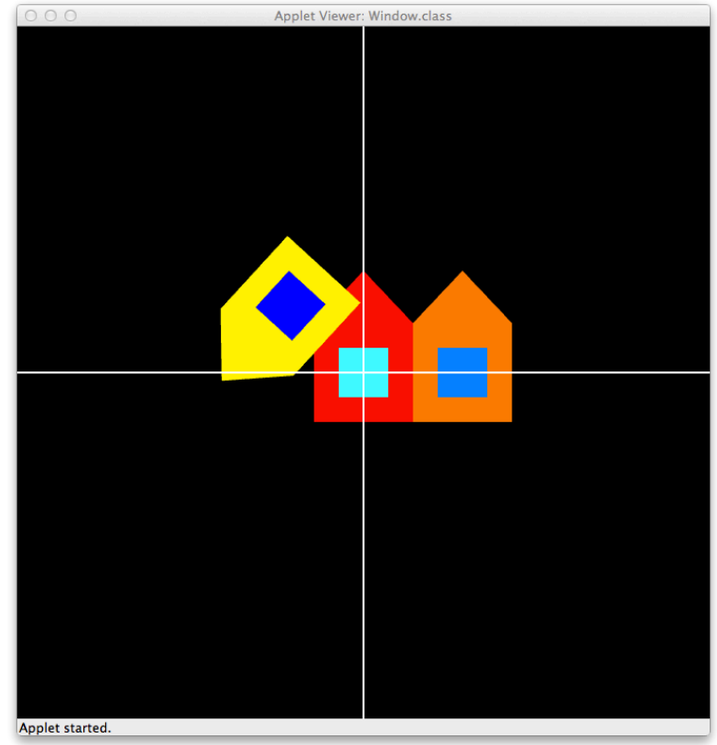
The Application



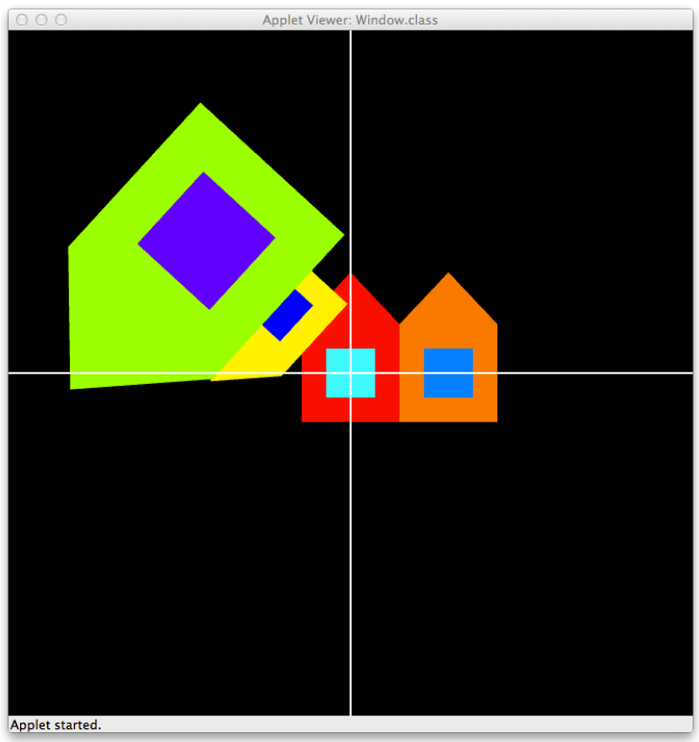
The Canvas

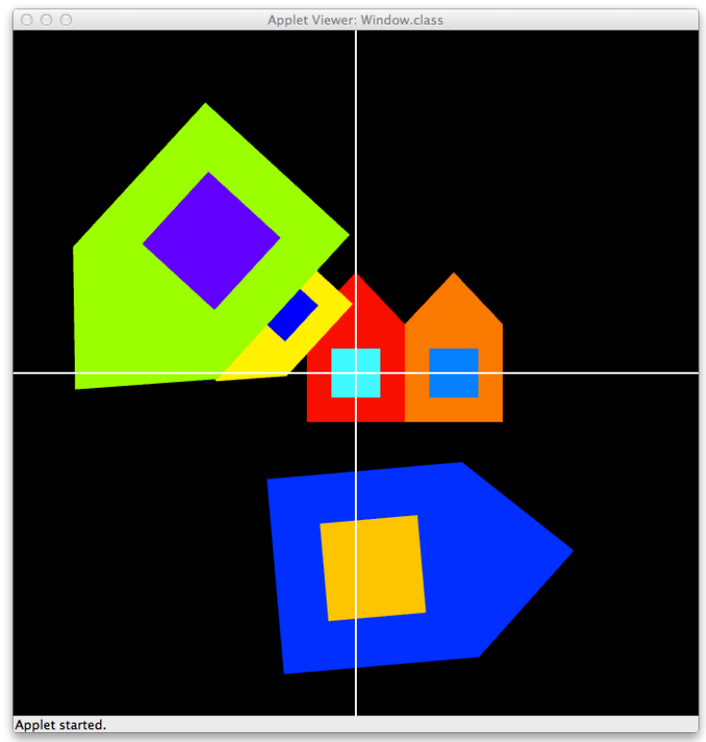


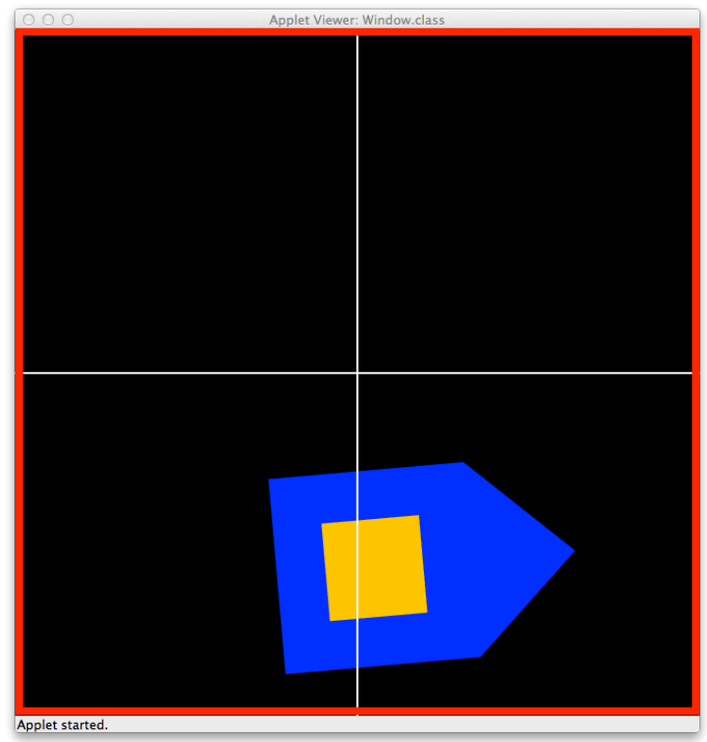




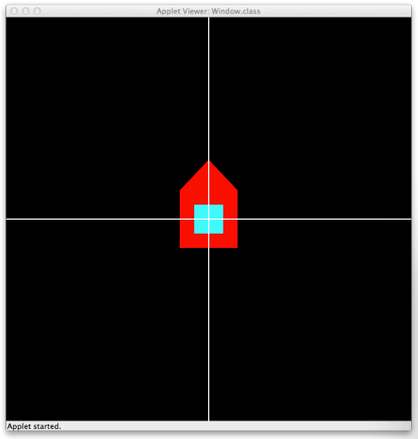
As you've seen this far, with each transformation, the color of the shape changes in an orderly fashion, but also the previous transformations are not removed, that is, unless you click on the canvas, then all the previous are deleted and only the most recent one remains! Try using this to make curious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



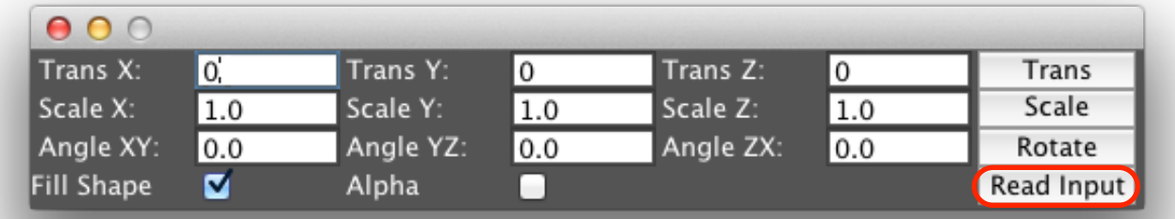


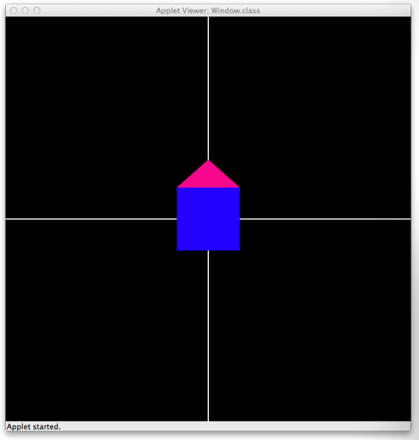


The Shapes

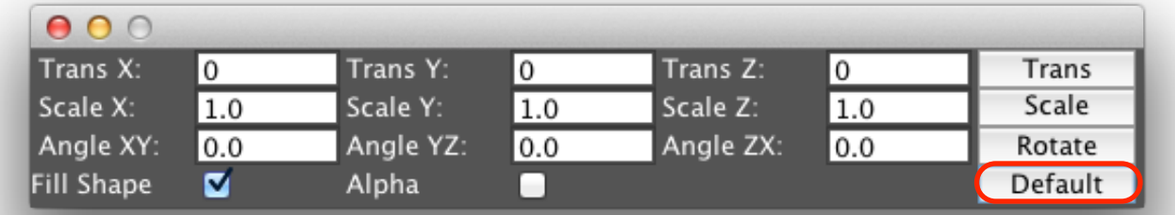


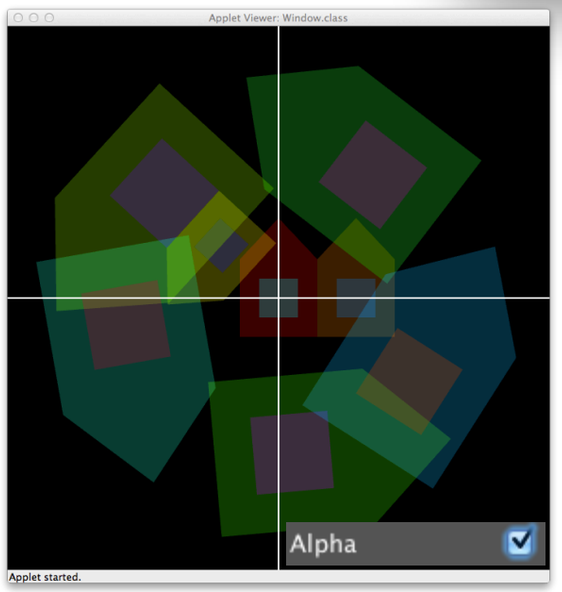
Default Mode: its the good old flat 2D house, but this time in a 3D environment, so 3D transformations can be applied to it for interesting results (such as the ones displayed in the 'Rotating XY/YZ/ZX' section'). To change to the other mode, press '**Read Input**'. After this, the button will change to ask if you want to use **Default Mode**.



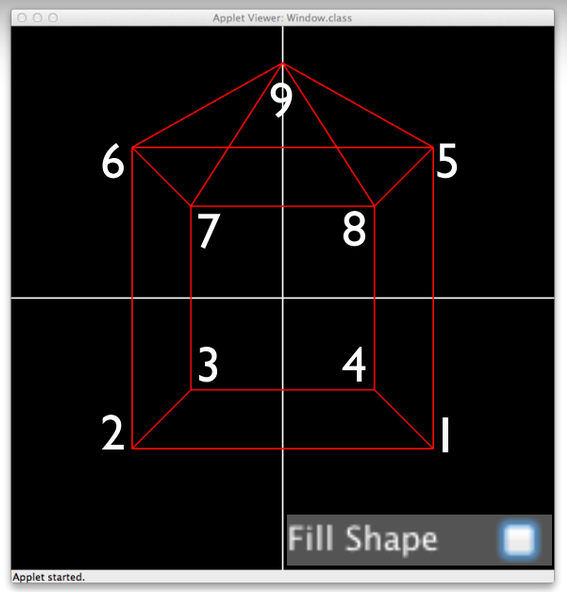


Read Input: It reads from a text file located in src/data/input.txt a set of points, followed some polygons described by those given points. In this case, its a 3D house, and every face will have a different color. To change to the other mode, press '**Default**'. After this, the button will change to ask if you want to **Read Input** from file.





With **Alpha** enabled, 25% transparency is enabled



With **Fill Shape** disabled, shapes are wireframes!

This set of numbered points and the set of polygons that use these points, describe this house!

src/data/input.txt

Number of points →

9 points, defined as:

n x y z

n: point number

x: x position

y: y position

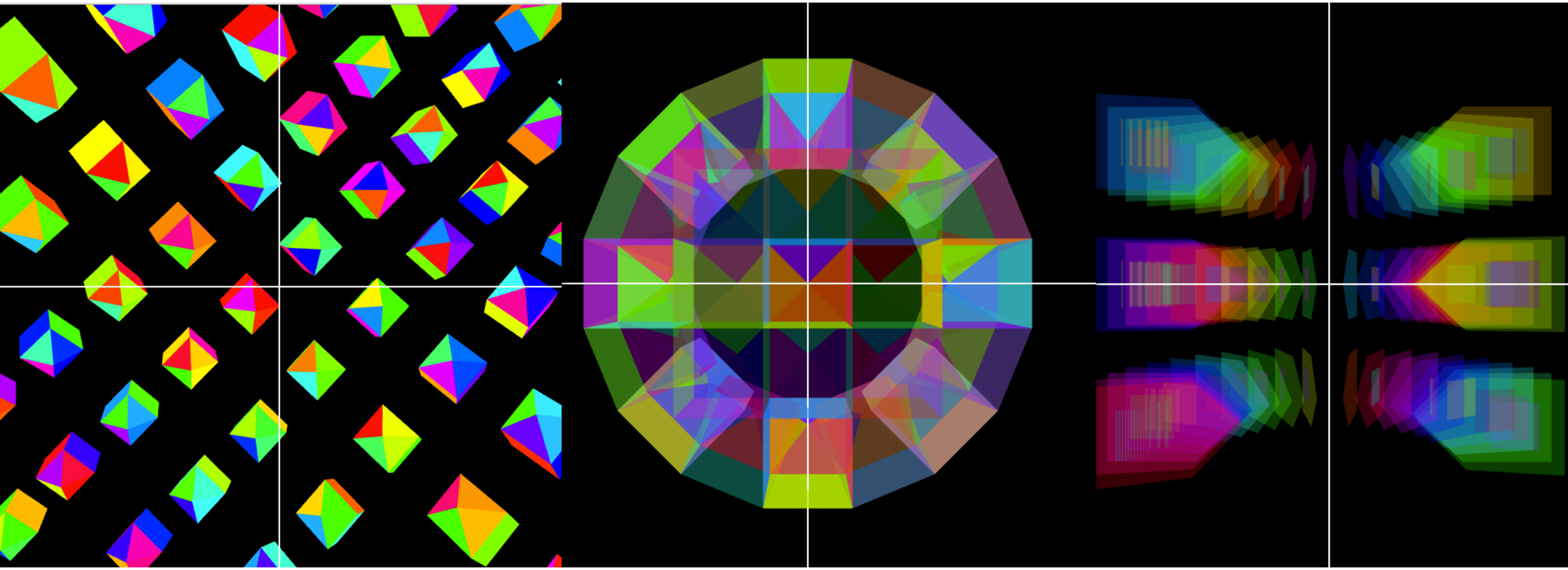
z: z position

the lines until the end of the file will describe a set of polygons, one per line, defined as a list of the points (by number) defining them

```
9
1 50 -50 50
2 -50 -50 50
3 -50 -50 -50
4 50 -50 -50
5 50 50 50
6 -50 50 50
7 -50 50 -50
8 50 50 -50
9 0 103 0
1 2 3 4 1
1 2 6 5 1
2 3 7 6 2
3 4 8 7 3
4 1 5 8 4
5 6 9 5
6 7 9 6
7 8 9 7
8 5 9 8
```

please no empty lines!

Some ‘Art’



Thanks for your time!

More challenges to follow!