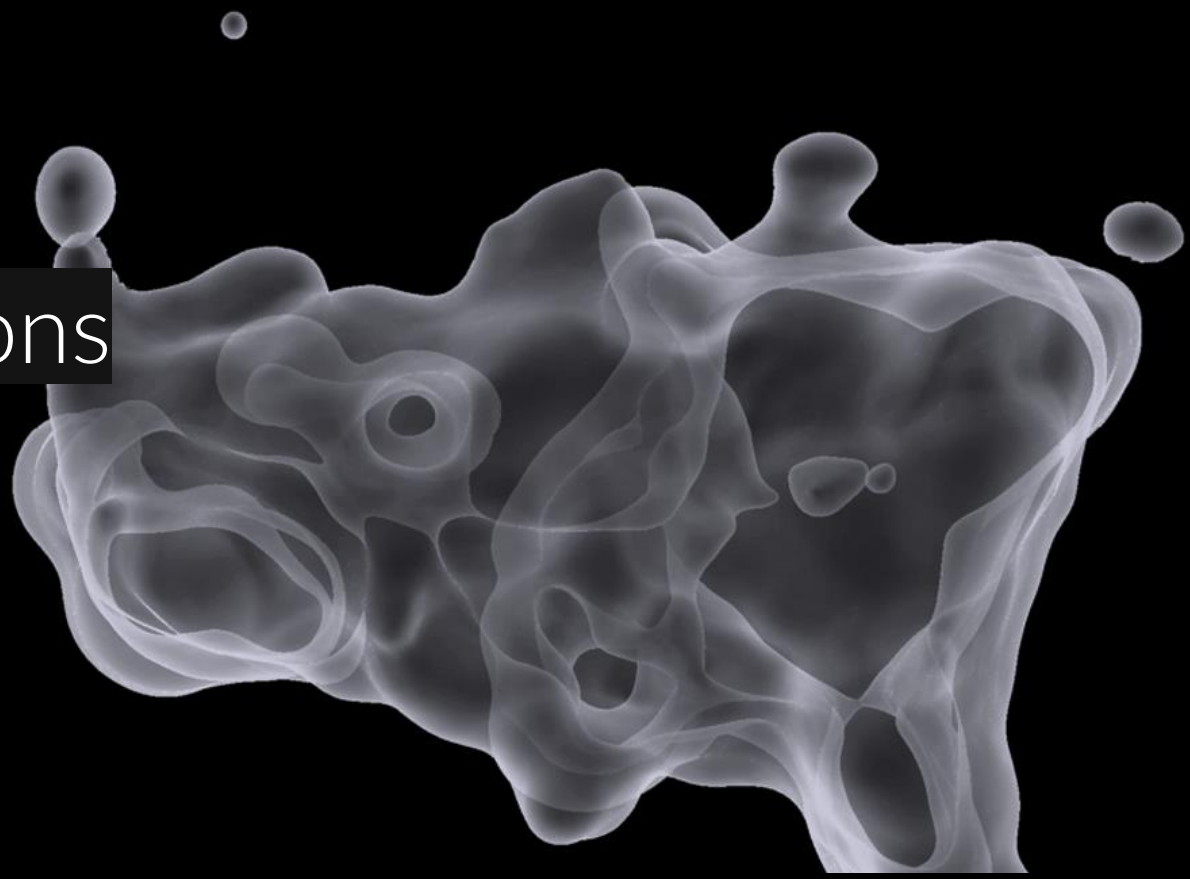
Artwork: Aaron Koblin



Transformations

VA345 Creative Coding

Course Instructor : Assoc. Prof. Dr. Selcuk ARTUT

Food for thought : Aaron Koblin

http://www.aaronkoblin.com/

Fall2021 - VA345 Creative Coding Assignments 002

Assignment 003 - Solution

Draw this exercise below with Illustrator and p5.js

Drawing an Islamic Geometric Pattern / Elhambra, Granda Spain (1302-1391)

saveCanvas()

Saves a numbered sequence of images, one image each time the function is run.

Syntax

```
saveCanvas(c, 'myCanvas', 'jpg');
```

Save as Vector

https://github.com/zenozeng/p5.js-svg

Let’s do an example

Add this line in your projects index.html :

<script src="https://unpkg.com/p5.js-svg@1.1.1"></script>

createCanvas(200, 200, SVG); // Create SVG Canvas

save("mySVG.svg"); // give file name print("saved svg");

noLoop(); // we just want to export once}

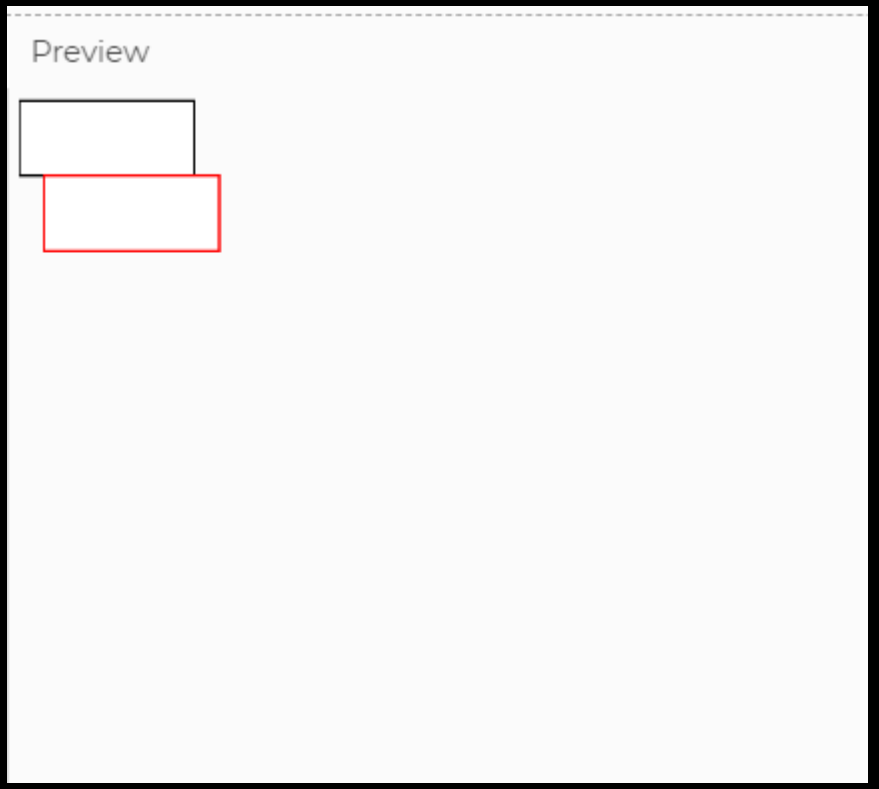
Translate

The translate() function moves the origin from the upper-left corner of the screen to another location. It has two parameters. The first is the x-coordinate offset and the second is the y-coordinate offset:

translate(x, y)

The values of the x and y parameters are added to any shapes drawn after the function is run.

Translate

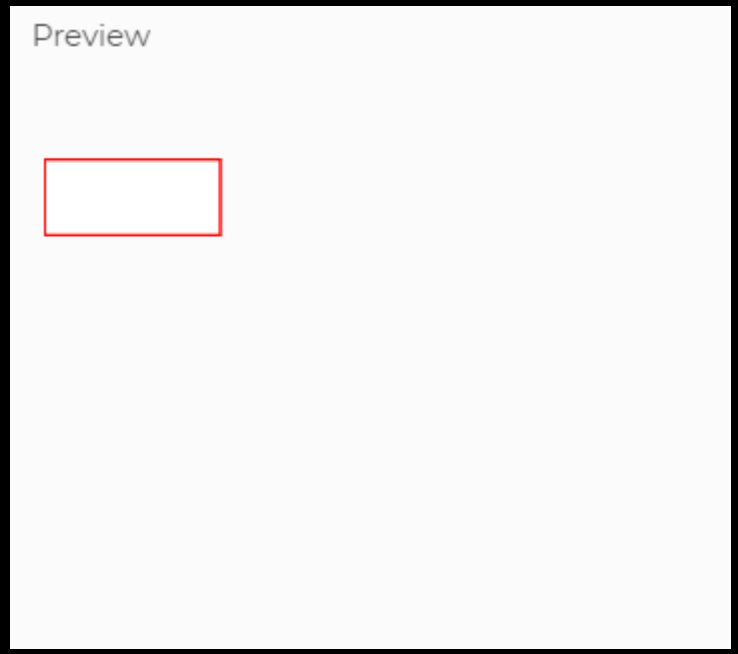


Code Examples (Translate Example 1)

Let's try

```
stroke("black");
rect(0, 5, 70, 30);
translate(10, 30); // Shifts 10 pixels right and 30 down
stroke("red");
rect(0, 5, 70, 30);
```

Translate



Code Examples (Translate Example 2)

vs

```
translate(10, 30); // Shifts 10 pixels right and 30 down
stroke("black");
rect(0, 5, 70, 30);
stroke("red");
rect(0, 5, 70, 30);
```

Translate

The translate() function is additive. If translate(10,30) is run twice, all the elements drawn after will display with an x-offset of 20 and a y-offset of 60 (Translate Example 3)

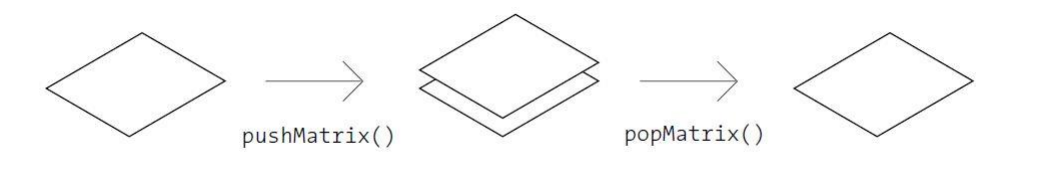
```
stroke("black");
rect(0, 5, 70, 30);
translate(10, 30); // Shifts 10 pixels right and 30 down
stroke("red");
rect(0, 5, 70, 30);
stroke("blue");
translate(10, 30); // Shifts everything again for a total
rect(0, 5, 70, 30); // 20 pixels right and 60 down
```

Translate

Controlling transformations

The transformation matrix is a set of numbers that defines how geometry is drawn to the screen. Transformation functions such as translate() alter the numbers in this matrix and cause the geometry to draw differently.

The push () function records the current state of all transformations so that a program can return to it later. To return to the previous state, use pop ().



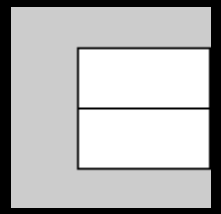
Each push() must have a corresponding pop(). The function push() cannot be used without pop(), and vice versa.

Translate

Using push() and pop()

Example 4

```
translate(33, 0); // Shift 33 pixels right
rect(0, 20, 66, 30);
rect(0, 50, 66, 30);
```

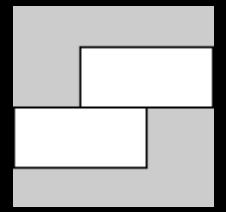


Translate

Using push() and pop()

Example 5

```
push();
translate(33, 0); // Shift 33 pixels right
rect(0, 20, 66, 30);
pop(); // Remove the shift
// This shape is not affected by translate() because
// the transformation is isolated between the push()
// and pop()
rect(0, 50, 66, 30);
```



Translate

Rotation

The rotate() function rotates the coordinate system so that shapes can be drawn to the screen at an angle. It has one parameter that sets the amount of the rotation as an angle:

rotate(angle)

The rotate function assumes that the angle is specified in units of radians

angleMode(DEGREES);

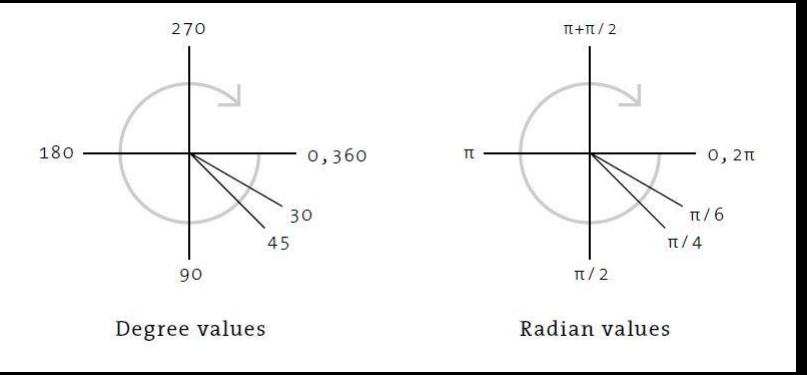
Description

Sets the current mode of p5 to given mode. Default mode is RADIANS.

Radian? Uh?

Degrees are a common way to measure angles. A right angle is 90°, halfway around a circle is 180°, and the full circle is 360°.

In working with trigonometry, angles are measured in units called radians. Using radians, the angle values are expressed in relation to the mathematical value π, written in Latin characters as “pi” and pronounced “pie.” In terms of radians, a right angle is π/2, halfway around a circle is simply π, and the full circle is 2π.

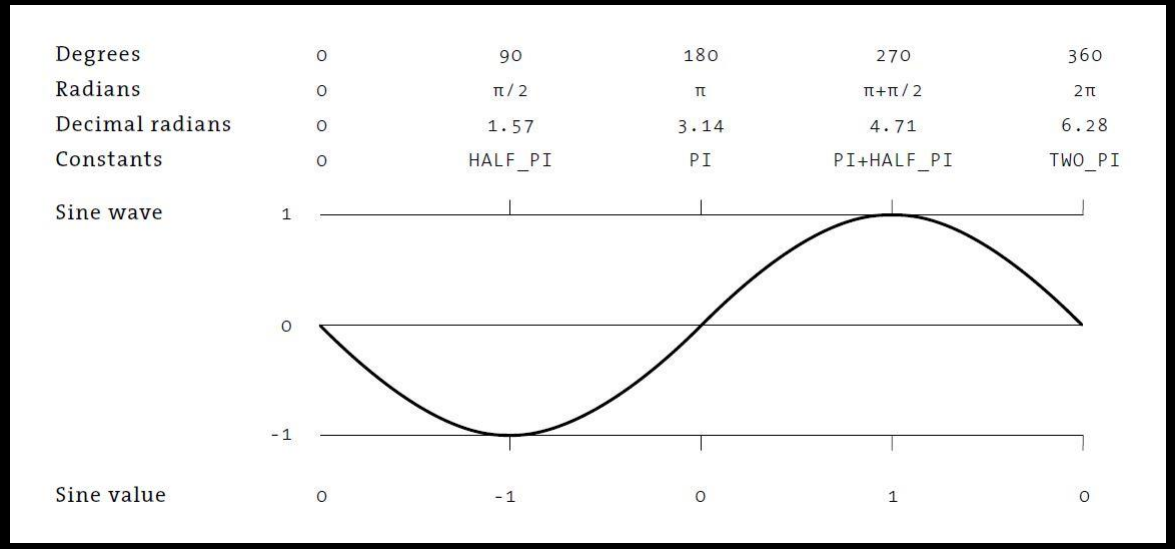


Trigonometry? Oh no!

The sin() and cos() functions are used to determine the sine and cosine value of any angle. Each of these functions requires one parameter:

sin(angle)

cos(angle)



Let's code an example

Make a shape move with numbers returned from sin() and cos().

Translate

Rotation Example 1

```
smooth();
rect(55, 0, 30, 45);
rotate(PI/8);
rect(55, 0, 30, 45);
```

Translate

Scaling

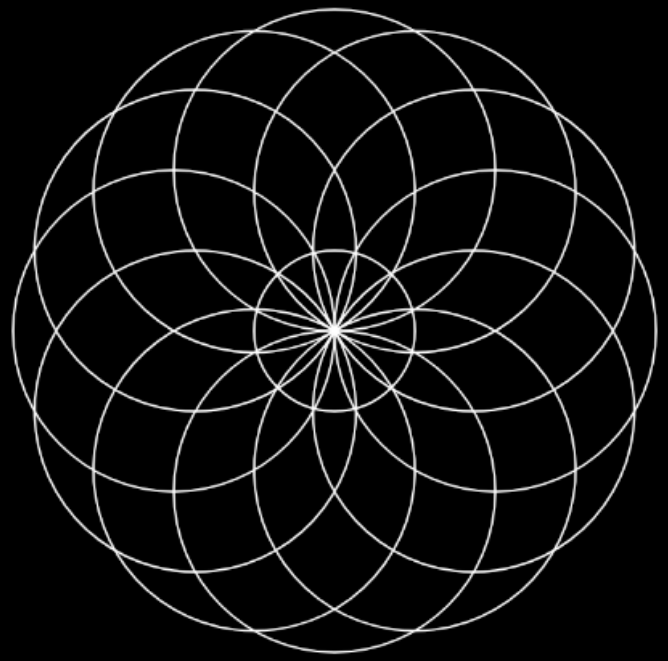
The scale() function magnifies the coordinate system so that shapes are drawn larger. It has one or two parameters to set the amount of increase or decrease:

scale(size)

scale(xsize, ysize)

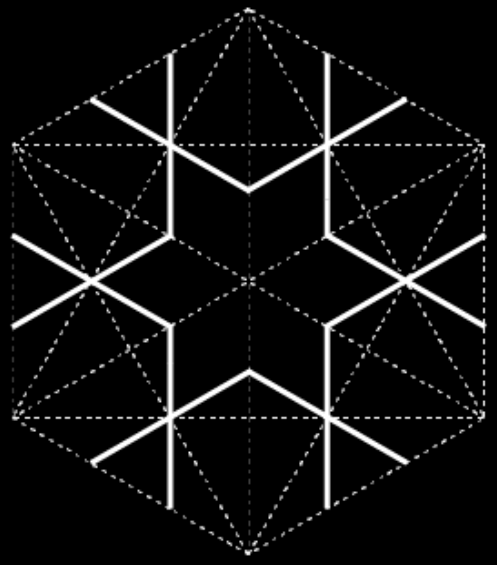
Scaling Example 1 :

```
smooth();
ellipse(32, 32, 30, 30);
scale(1.8);
ellipse(32, 32, 30, 30);
```

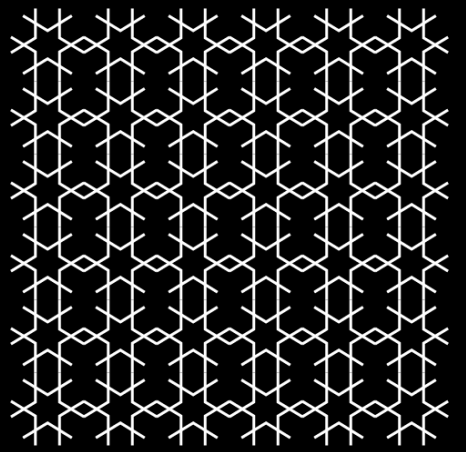


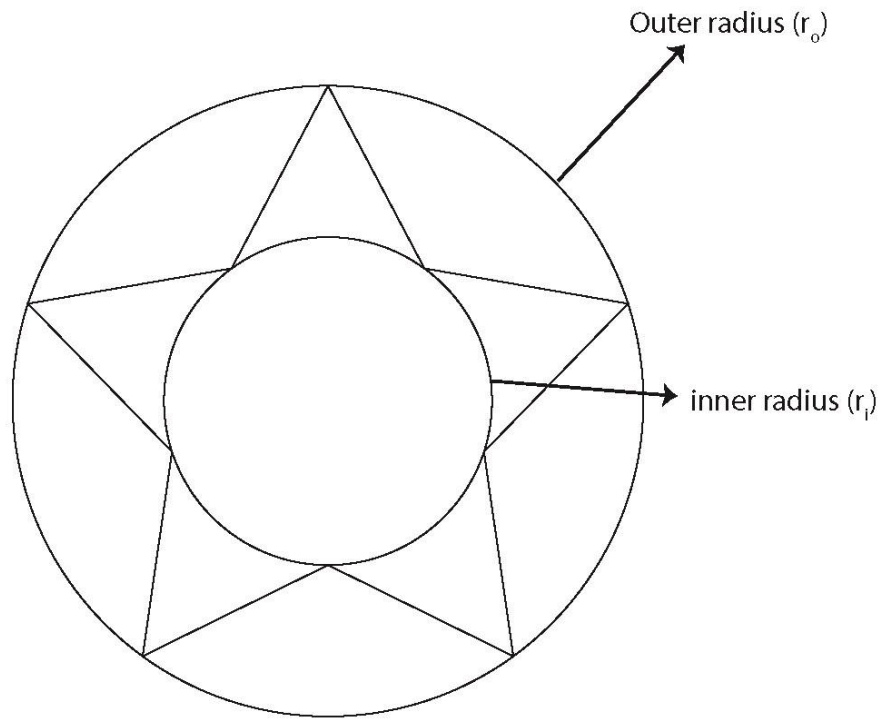
Example

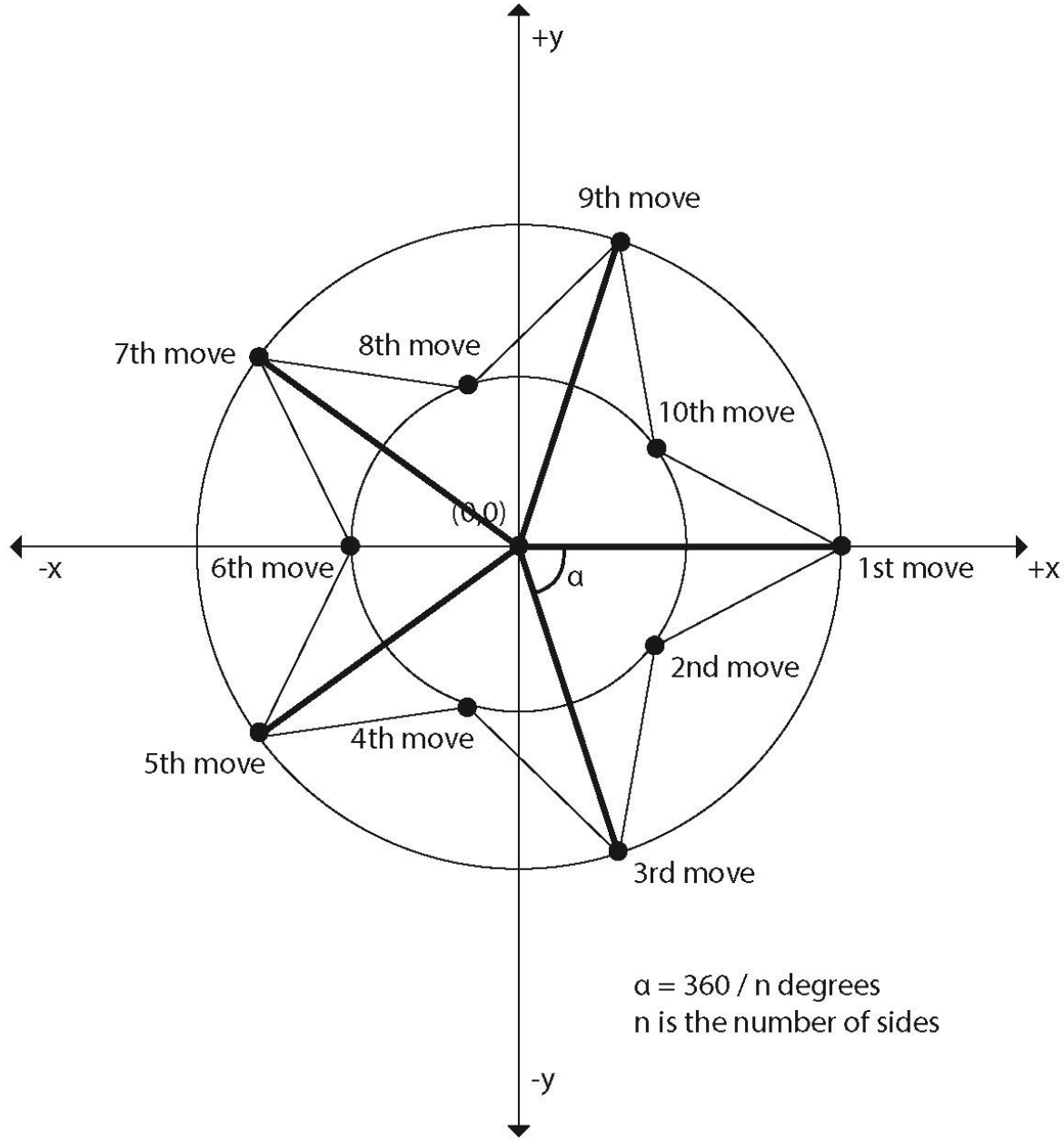
Use Rotation, Transformation functions to create this visual with p5js



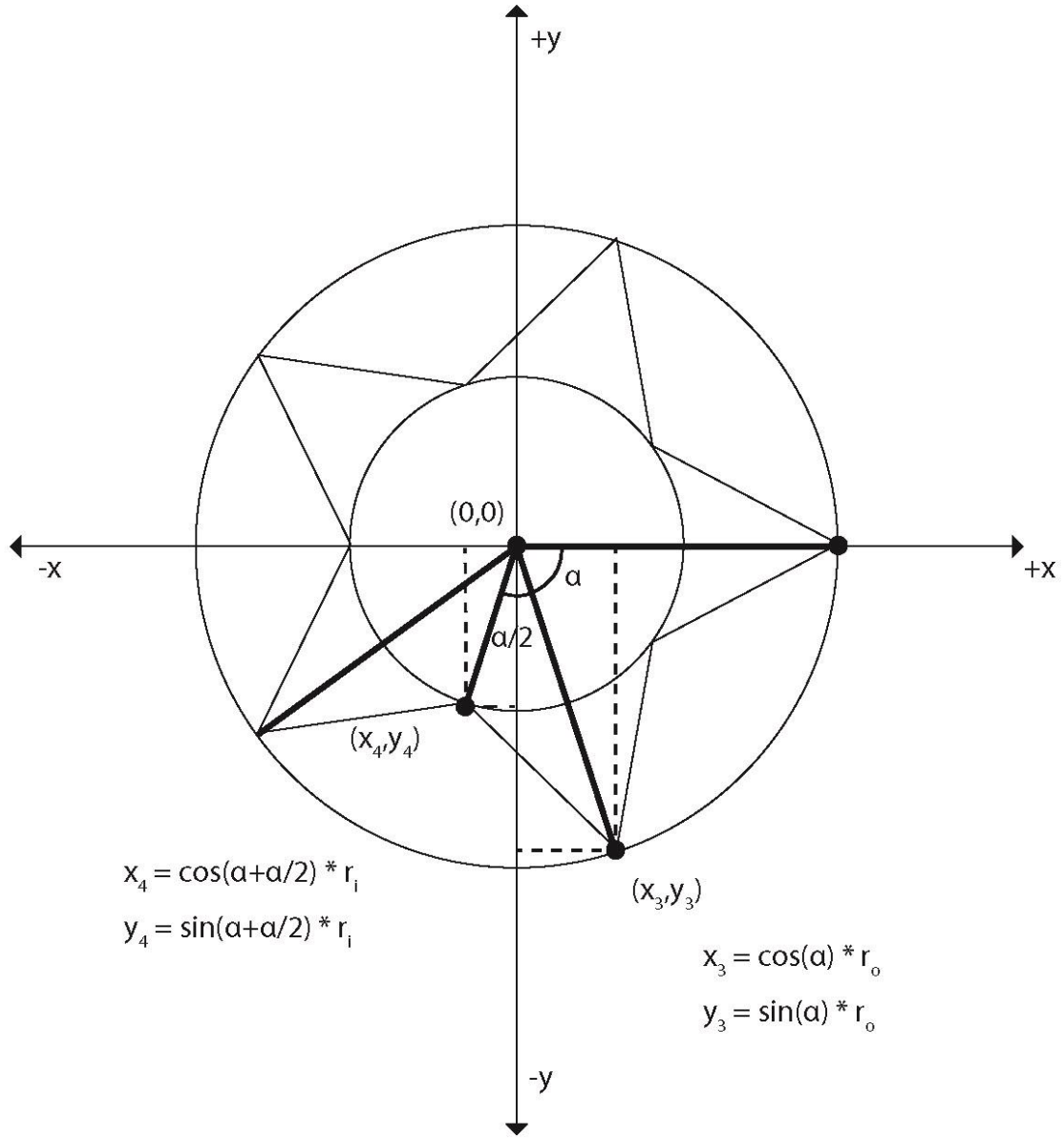
Ex: Coding an Islamic Geometric Pattern / Eşrefoğlu Mosque, Beyşehir




Outer radius (r_o)
inner radius (r_i)



α = 360 / n degrees
n is the number of sides

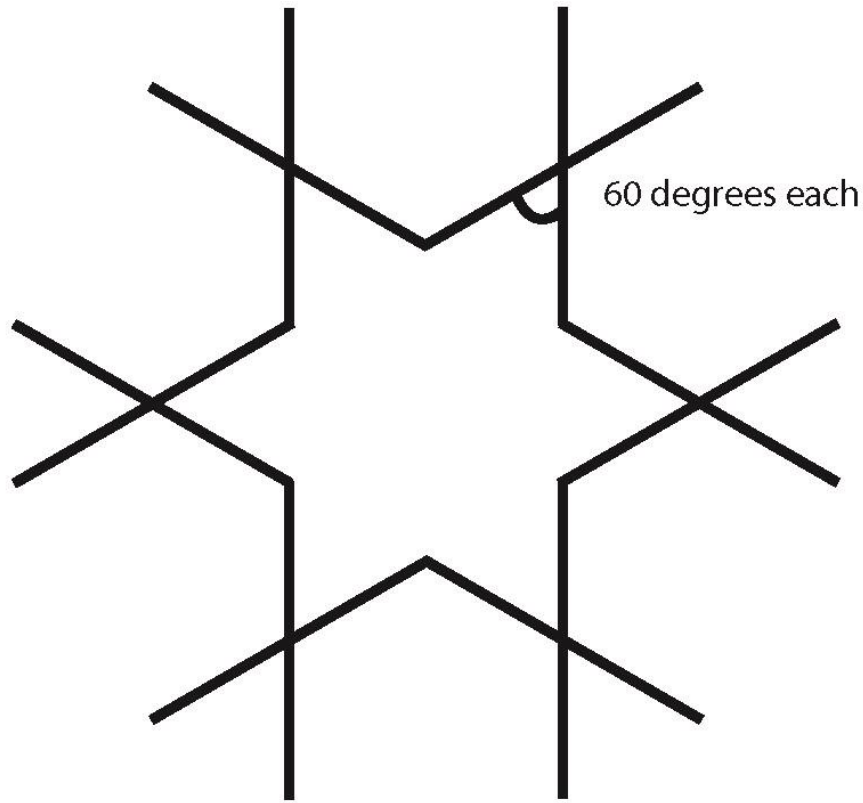

+y
-x
+x
-y
(0,0)
α
α/2
(x_4,y_4)
(x_3,y_3)
$x_4 = \cos(\alpha+\alpha/2) * r_i$
$y_4 = \sin(\alpha+\alpha/2) * r_i$
$x_3 = \cos(\alpha) * r_o$
$y_3 = \sin(\alpha) * r_o$

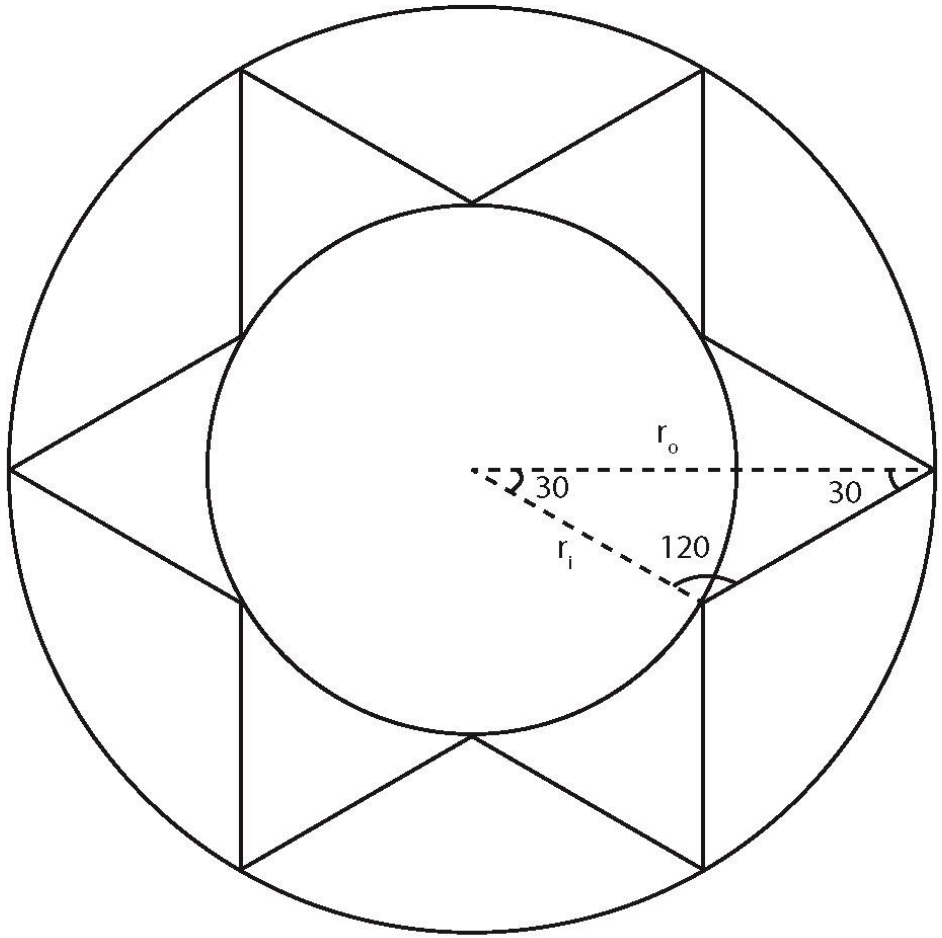
VA345 Creative Coding
Instructor: Selcuk ARTUT

Coding an Islamic Geometric Pattern / Eşrefoğlu Mosque, Beyşehir

The loop can be based on this algorithm. Note we are using Radians in here.

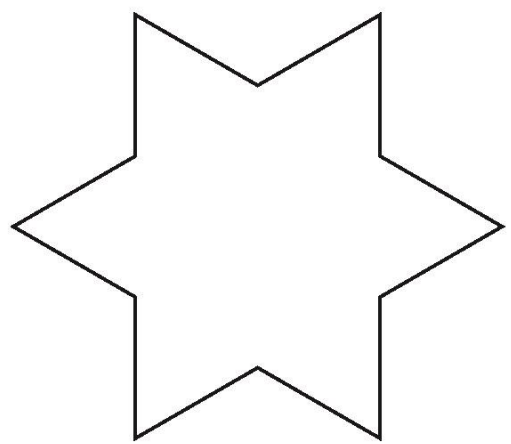
```
let angle = TWO_PI / npoints;
let halfAngle = angle / 2.0;

for (let a = 0; a < TWO_PI; a += angle) {
    let sx = x + cos(a) * radius2;
    let sy = y + sin(a) * radius2;
    vertex(sx, sy);
    sx = x + cos(a + halfAngle) * radius1;
    sy = y + sin(a + halfAngle) * radius1;
    vertex(sx, sy);
}
```

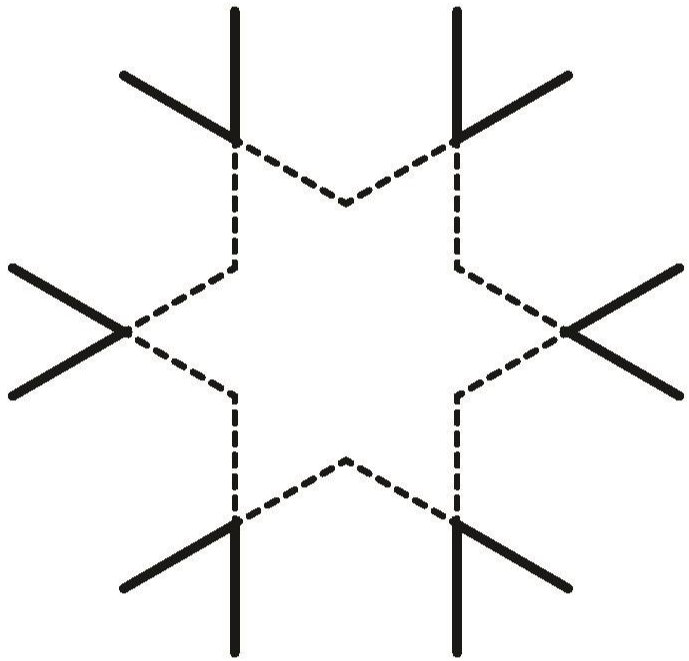

60 degrees each

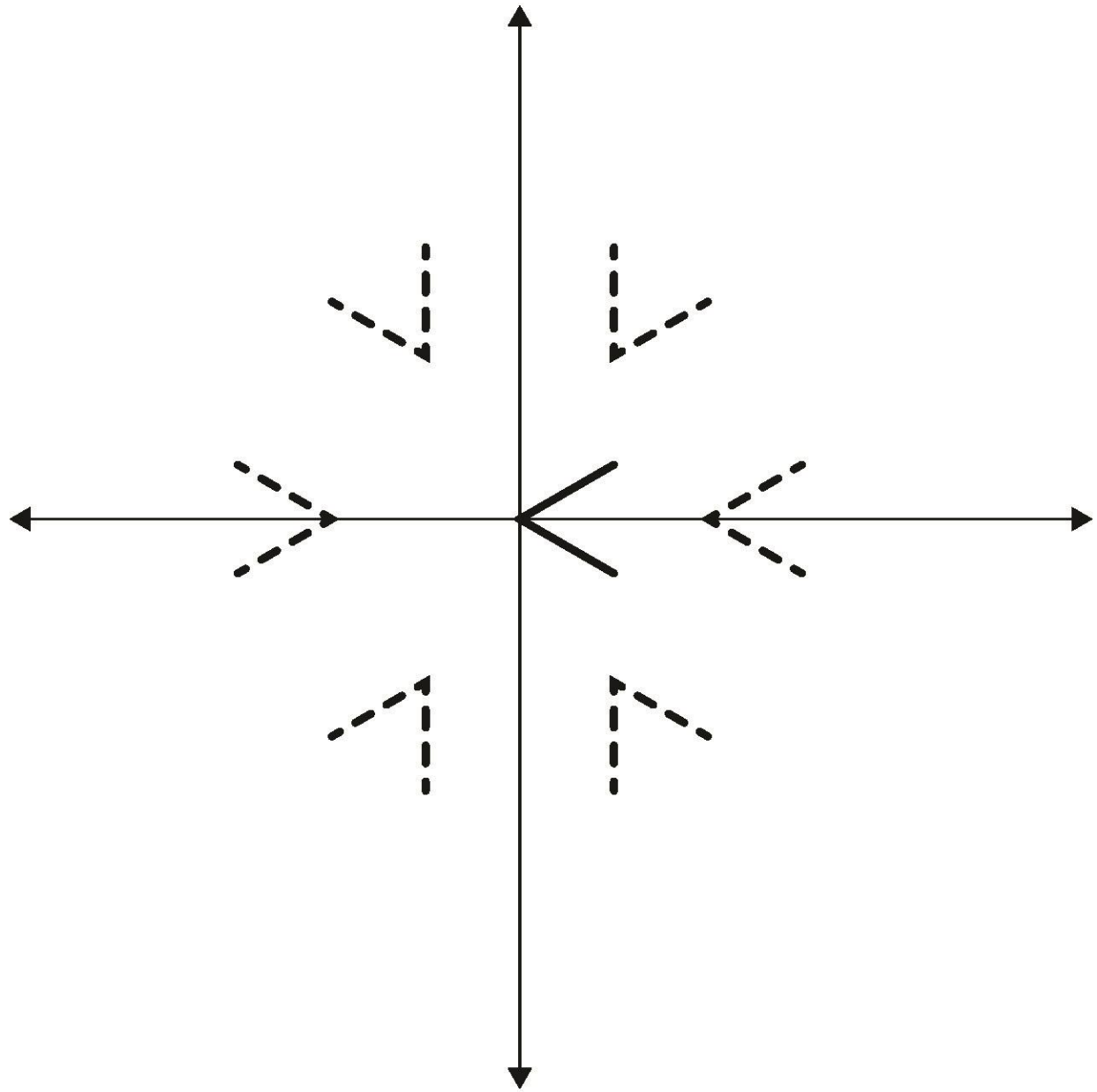


$$\frac{r_i}{\sin 30^\circ} = \frac{r_o}{\sin 120^\circ}$$

```
let outerRadius = 80;
let innerRadius;

let angle = TWO_PI / npoints;
let halfAngle = angle / 2.0;

innerRadius = outerRadius * (sin(radians(30)) / sin(radians(120)));

for (let a = 0; a < TWO_PI; a += angle) {
    let sx = x + cos(a) * outerRadius ;
    let sy = y + sin(a) * outerRadius ;
    vertex(sx, sy);
    sx = x + cos(a + halfAngle) * innerRadius;
    sy = y + sin(a + halfAngle) * innerRadius;
    vertex(sx, sy);
}
```







```
for (let a = 0; a < TWO_PI; a += angle) {
    push();
    rotate(a);
    translate(radius2, 0);
    beginShape();
    sx = x + cos(halfAngle) * radius1;
    sy = y - sin(halfAngle) * radius1;
    vertex(sx, sy);
    vertex(0, 0);
    sx = x + cos(halfAngle) * radius1;
    sy = y + sin(halfAngle) * radius1;
    vertex(sx, sy);
    endShape();
    pop();
  }
```

Arrays

The term array refers to a structured grouping or an imposing number—“The dinner buffet offers an array of choices,” “The city of Los Angeles faces an array of problems.” In computer programming, as array is a set of data elements stored under the same name.

Arrays can be created to hold any type of data, and each element can be individually assigned and read. There can be arrays of numbers, characters, sentences, boolean values, etc. Arrays might store vertex data for complex shapes, recent keystrokes from the keyboard, or data read from a file.

example demonstation

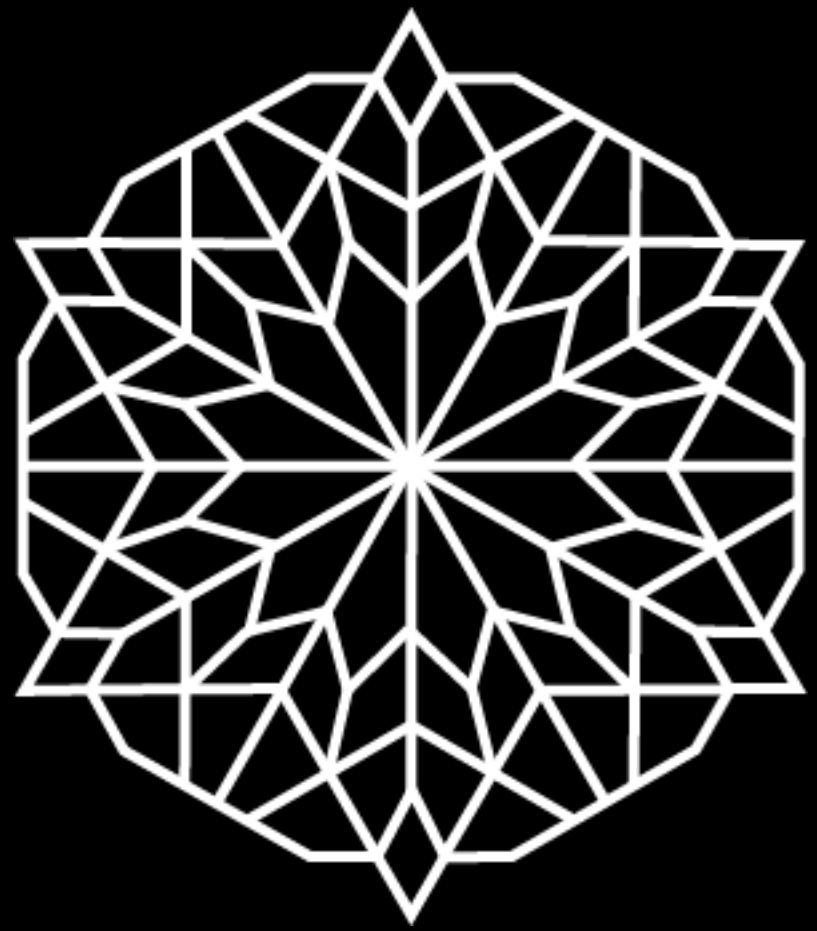
let x = [50, 61, 83, 69]

Arrays

```
let x = [ 50, 61, 83, 69, 71, 50, 29, 31, 17, 39 ];
let y = [ 18, 37, 43, 60, 82, 73, 82, 60, 43, 37 ];
beginShape();
// Reads one array element every time through the for()
for (let i = 0; i < x.length; i++) {
vertex(x[i], y[i]);
}
endShape(CLOSE);
```

Declaring Arrays

```
let data = []; // Declare that it is an array
function setup() {
createCanvas(100, 100);
data[0] = 19; // Assign
data[1] = 40;
data[2] = 75;
data[3] = 76;
data[4] = 90;
}
```



Assignment 004

Follow the Drawing002-Handdrawn.pdf guidebook.
Draw a Persian Orosi Glass Islamic Geometric Pattern
Ref: https://www.youtube.com/watch?v=K2VrRdbZhf8