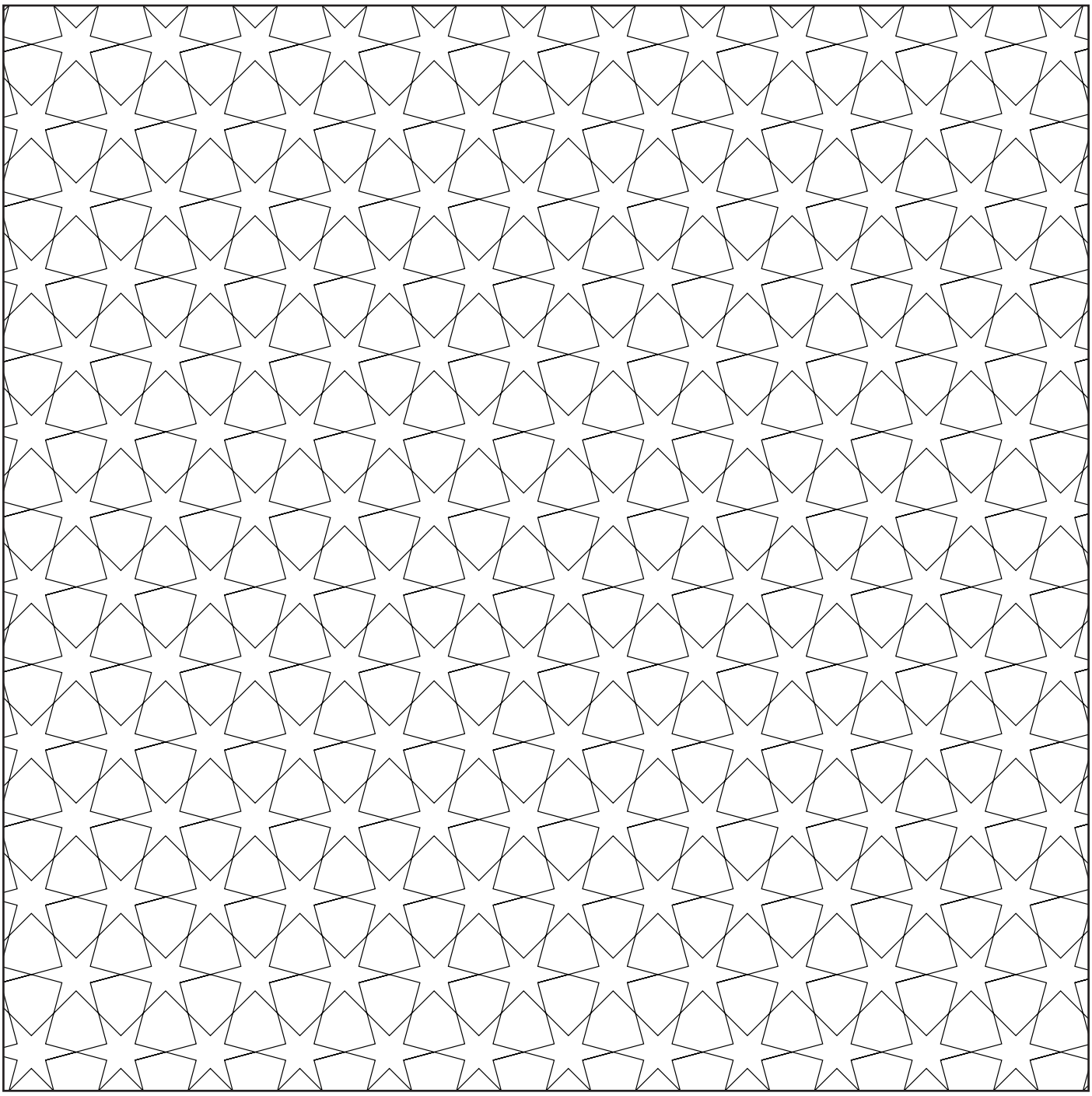
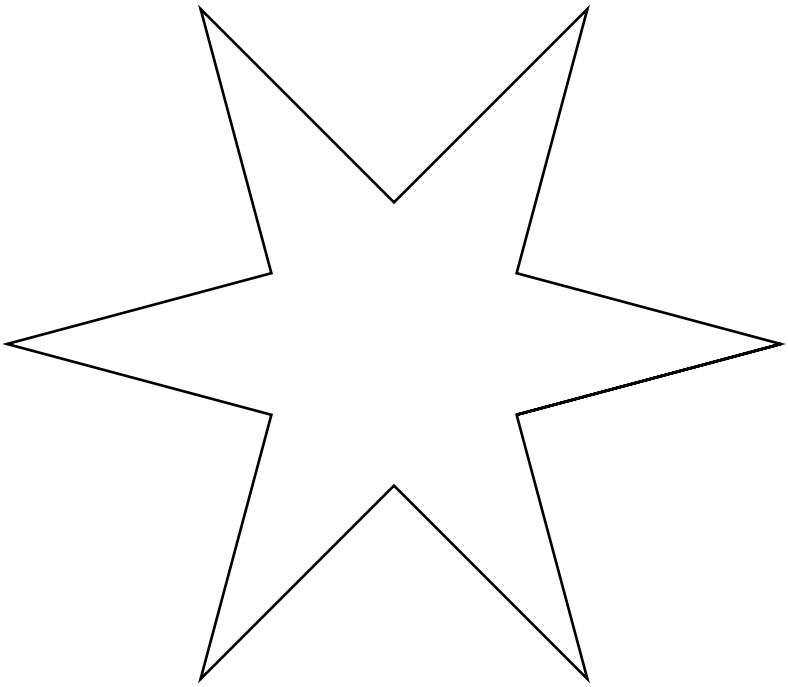
Geometrik Deseni Kodlamak

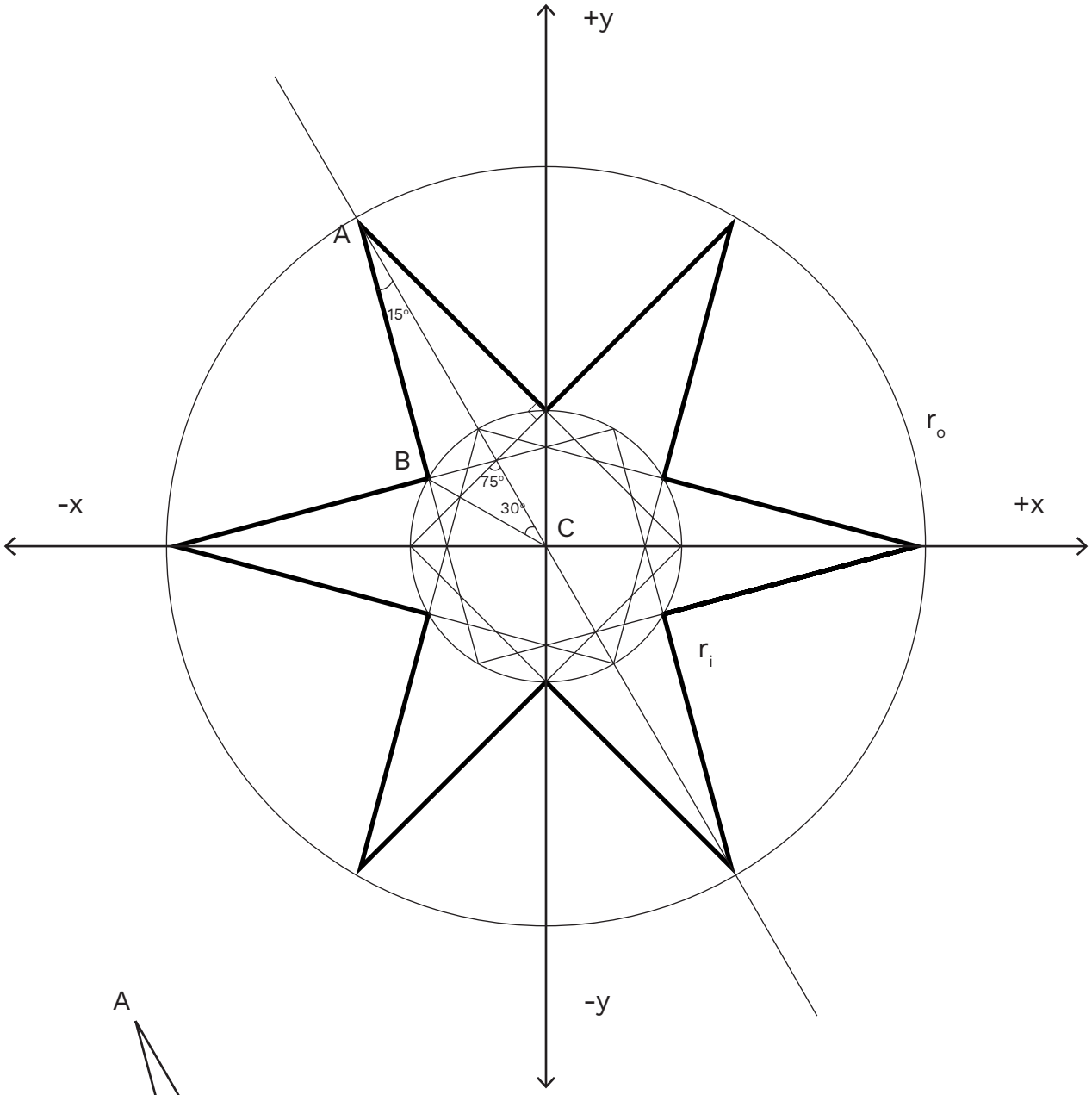
Aşağıdaki deseni inceleyin ve bu deseni oluşturan temel görsel bileşeni bulmaya çalışın.

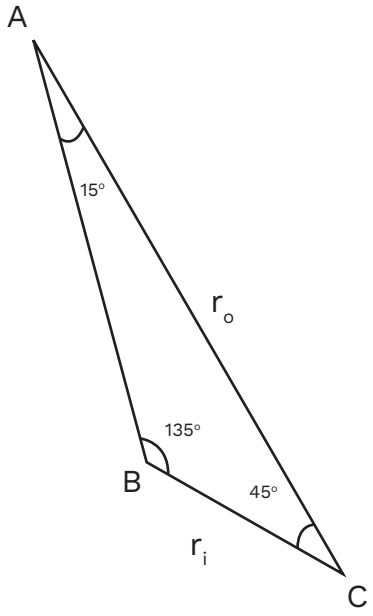


Motif



Temel Görsel Bileşini İnceleyelim

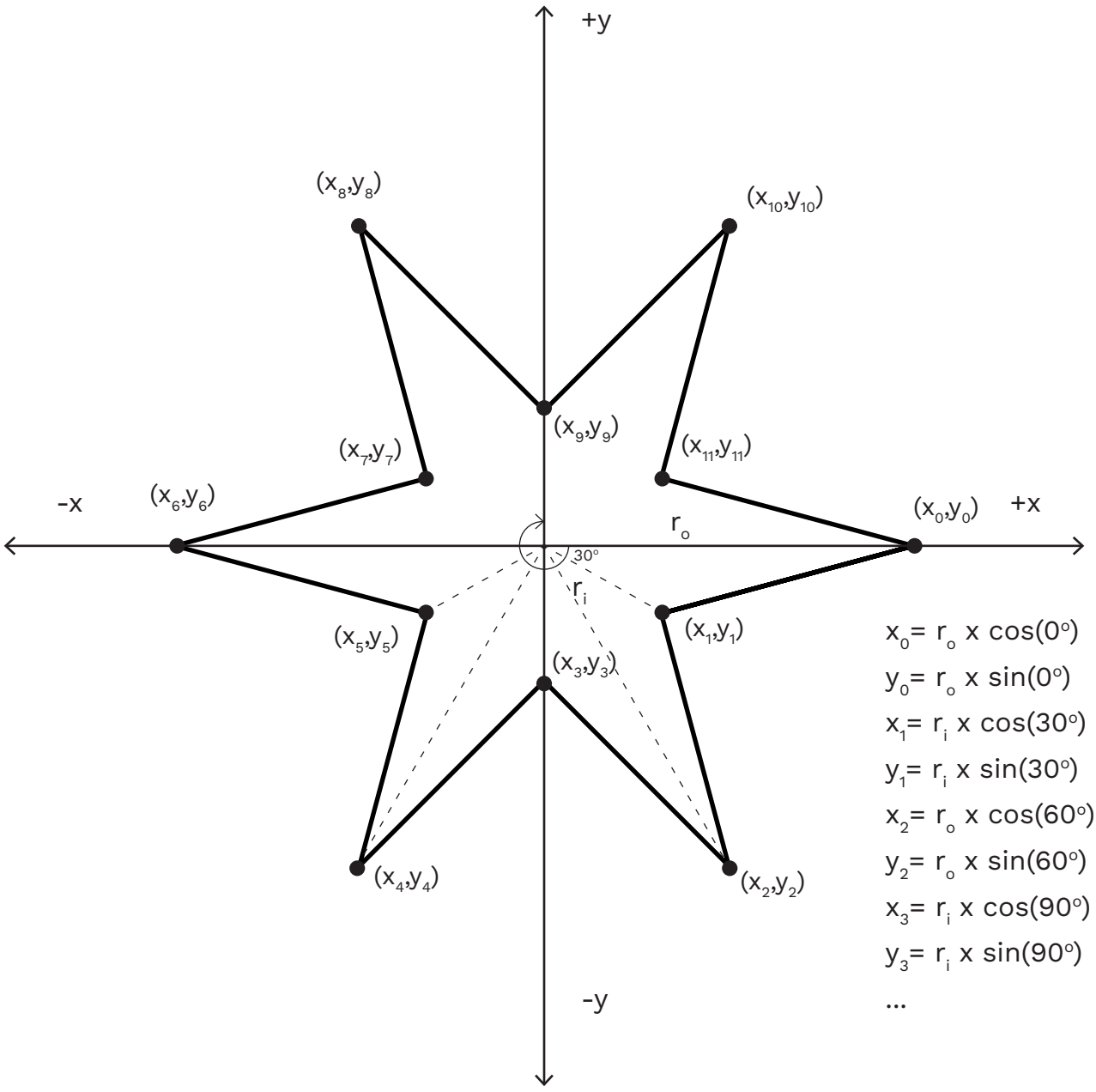




$$\frac{r_i}{\sin(135^\circ)} = \frac{r_o}{\sin(135^\circ)}$$

Açıları ve Vertex noktalarını tespit etmek

Aşama 1 : Vertex noktalarını bulalım



$x_0 = r_o \times \cos(0^o)$

$y_0 = r_o \times \sin(0^o)$

$x_1 = r_i \times \cos(30^o)$

$y_1 = r_i \times \sin(30^o)$

$x_2 = r_o \times \cos(60^o)$

$y_2 = r_o \times \sin(60^o)$

$x_3 = r_i \times \cos(90^o)$

$y_3 = r_i \times \sin(90^o)$

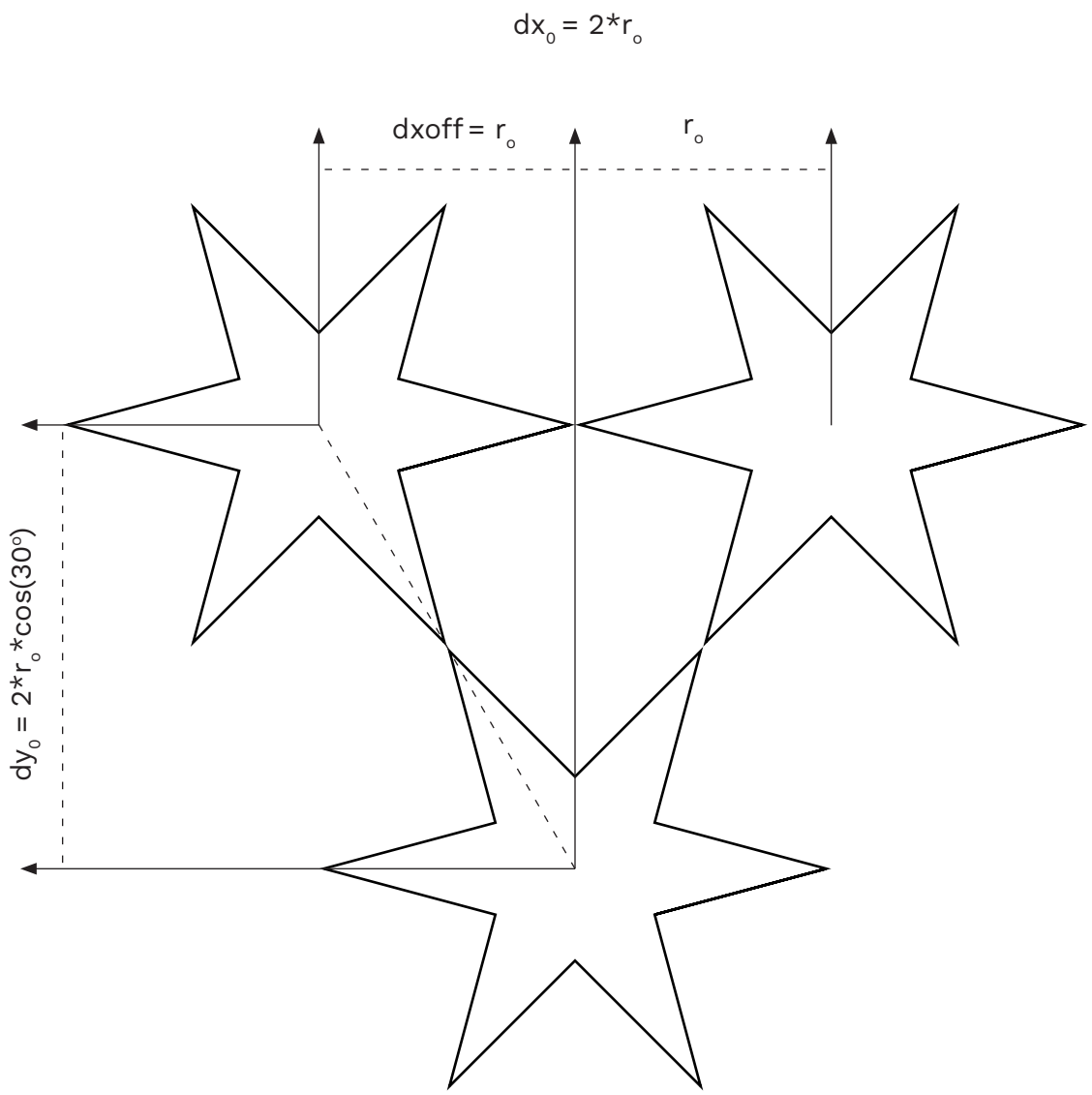
...

Motifi Oluşturmak

```
let innerRadius,outerRadius;

function setup() {
  createCanvas(400, 400);
  angleMode(DEGREES);
  innerRadius = 24;
  outerRadius = innerRadius * (sin(135) / sin(15));
}

function draw() {
  background(255);
  noFill();
  let angle = 30;

  push();
  translate(width*0.5,height*0.5);
  beginShape();
  for (let i = 0; i < 12; i++) {
    let sx,sy;
    if(i%2==0){
      sx = cos(i*angle) * outerRadius;
      sy = sin(i*angle) * outerRadius;
    }else{
      sx = cos(i*angle) * innerRadius;
      sy = sin(i*angle) * innerRadius;
    }

    vertex(sx, sy);
  }
  endShape(CLOSE);
  pop();
}
```

Bezeme Yapısını İnceleyelim

Aşama 2 : Yukarı ve aşağıya kaymaları belirleyecek xoffset ve yoffset değerlerini hesaplayalım.



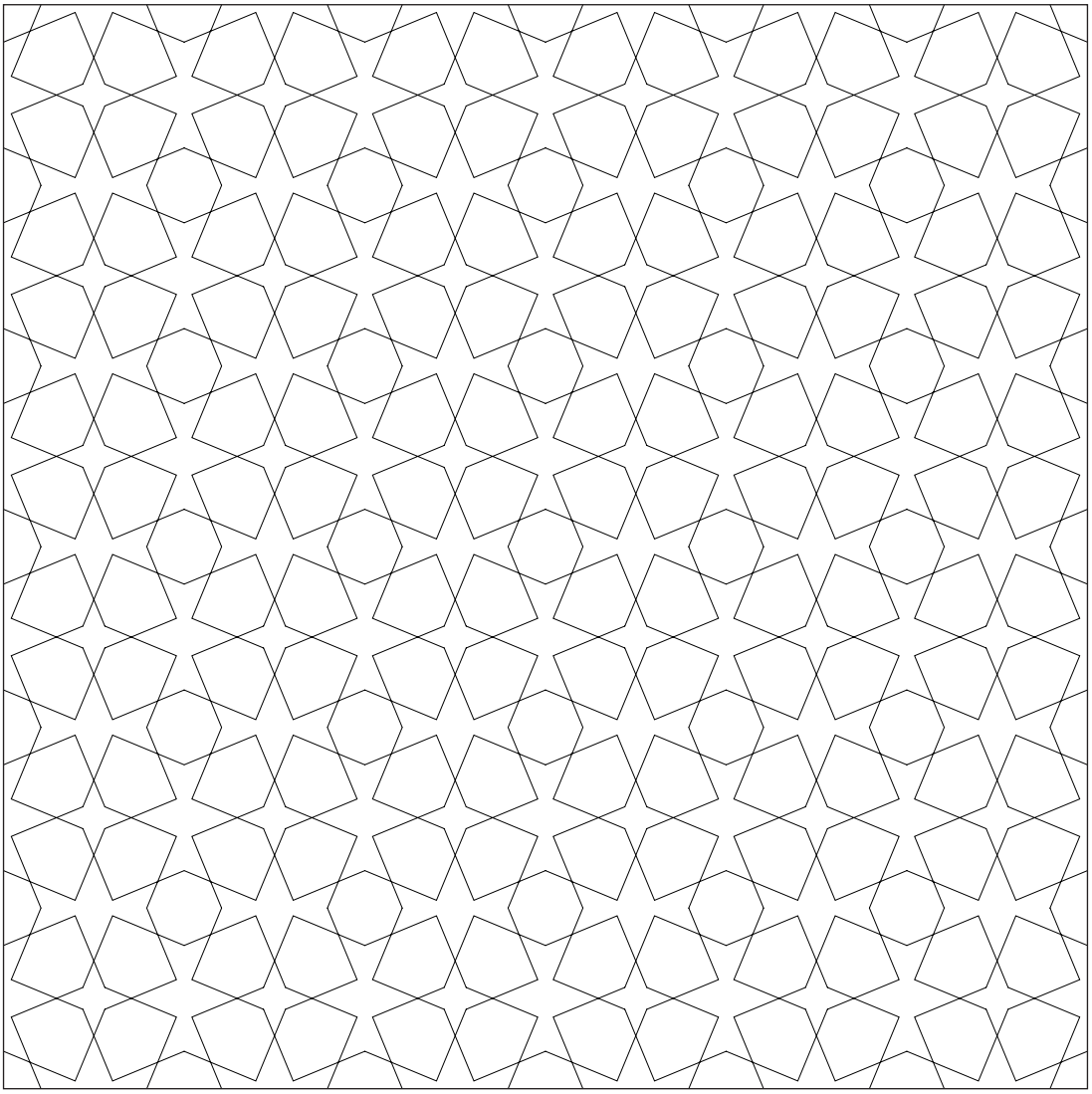
Bezeme Kodu

```
// Tile class
class Motif {
  constructor(r) {
    this.innerRadius = r;
    this.outerRadius = r * (sin(135) / sin(15));
  }

  display() {

  let angle = 30;

  beginShape();
  for (let i = 0; i < 12; i++) {
    let sx,sy;
    if(i%2==0){
      sx = cos(i*angle) * this.outerRadius;
      sy = sin(i*angle) * this.outerRadius;
    }else{
      sx = cos(i*angle) * this.innerRadius;
      sy = sin(i*angle) * this.innerRadius;
    }
    vertex(sx, sy);
  }
  endShape(CLOSE);
  }

}

let innerRadius,outerRadius;
let xOff,yOff;
let nRow;
let nCol;

function setup() {
  createCanvas(600, 600);
  angleMode(DEGREES);
  innerRadius = 12;
  outerRadius = innerRadius * (sin(135) / sin(15));
  xOff = 2 * outerRadius;
  yOff = 2 * outerRadius * cos(30);
  nRow = floor(height / (2 * innerRadius));
  nCol = floor(width / (2 * innerRadius));
}

function draw() {
  background(255);
  let motif = new Motif(innerRadius);
  noFill();
```

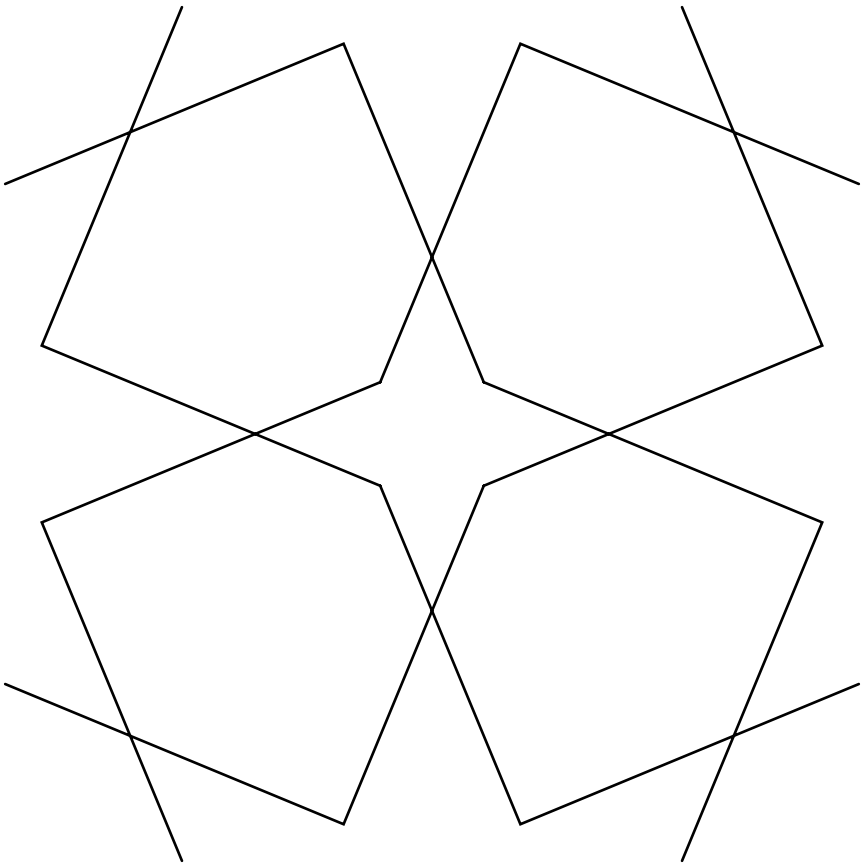
```
  for (let r = 0; r < nRow; r++) {
    for (let c = 0; c < nCol; c++) {
      push();
      if(r%2==0){
        //rows 0,2,4,6
        translate(xOff * c,  yOff * r);
      }else{
        //rows 1,3,5,7
        translate(xOff * c + outerRadius,  yOff * r);
      }
      motif.display();
      pop();
    }
  }
}
```

Geometrik Deseni Kodlamak

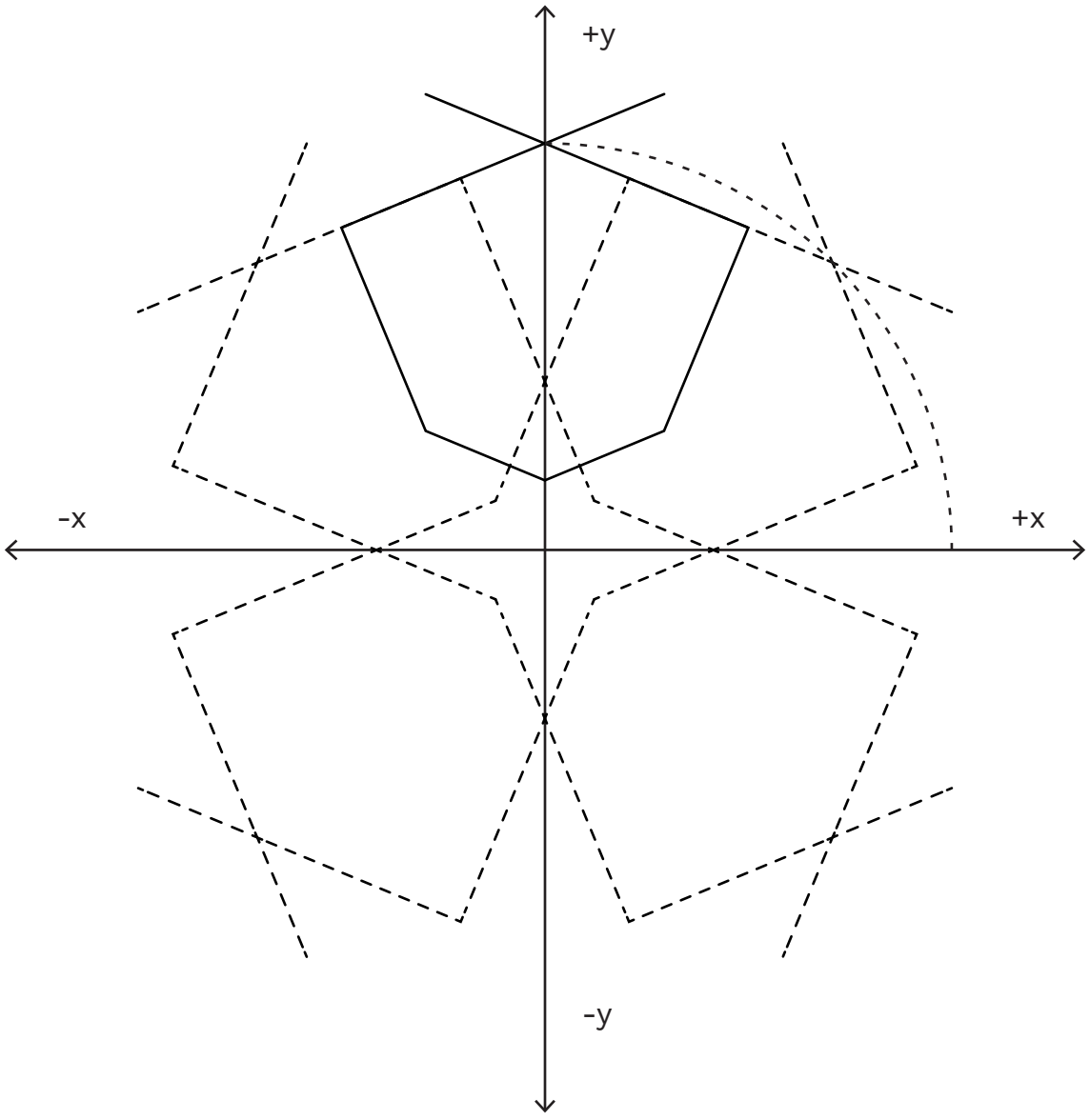
Aşağıdaki deseni inceleyin ve bu deseni oluşturan temel görsel bileşeni bulmaya çalışın.



Motif



Temel Görsel Bileşini İnceleyelim



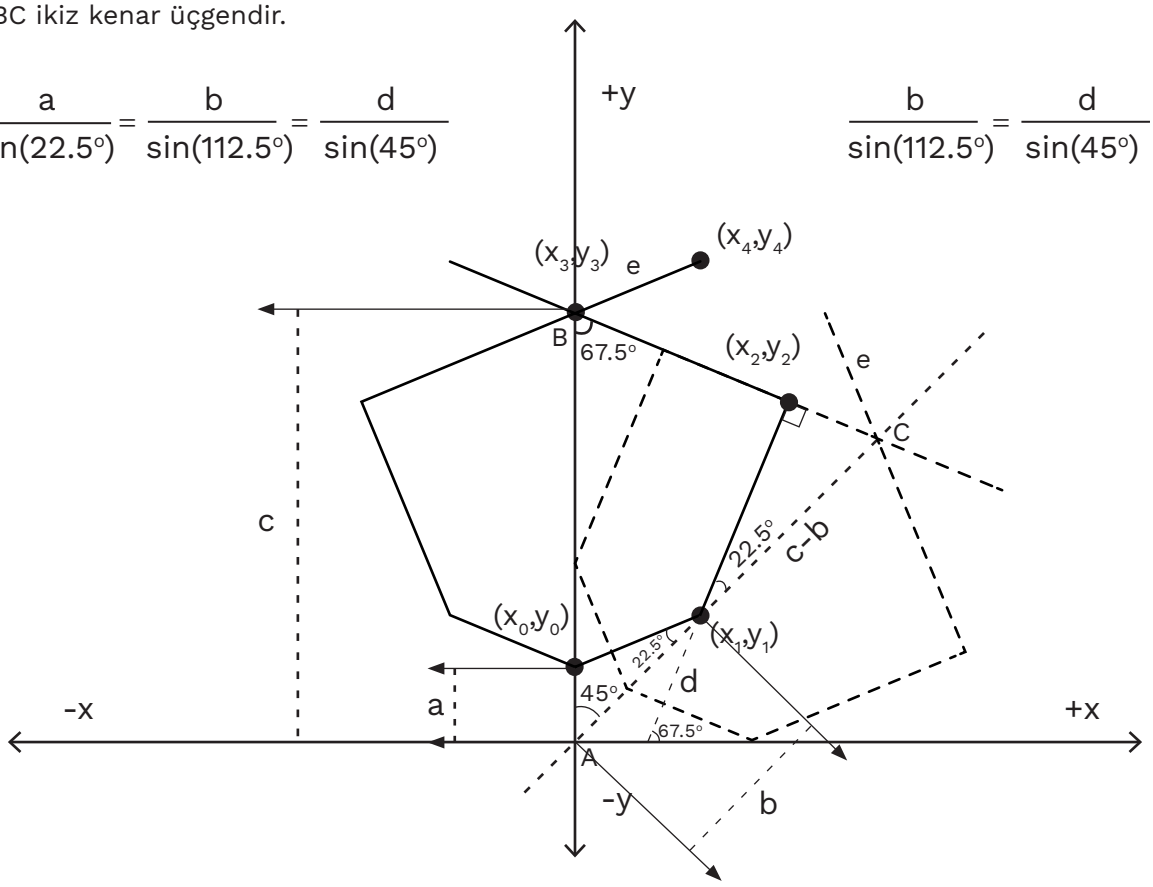
Açıları ve Vertex noktalarını tespit etmek

Aşama 1 : Vertex noktalarını bulalım

△ABC ikiz kenar üçgendir.

$$\frac{a}{\sin(22.5^o)} = \frac{b}{\sin(112.5^o)} = \frac{d}{\sin(45^o)}$$

$$\frac{b}{\sin(112.5^o)} = \frac{d}{\sin(45^o)}$$



$x_0 = 0$

$y_0 = a$

$x_1 = b \times \cos(45^o)$

$y_1 = b \times \sin(45^o)$

$x_2 = (d+(c-b)\times\cos(22.5^o))\times \cos(67.5^o) + a$

$y_2 = (d+(c-b)\times\cos(22.5^o))\times \sin(67.5^o)$

$x_3 = 0$

$y_3 = c$

$x_4 = e \times \cos(22.5)$

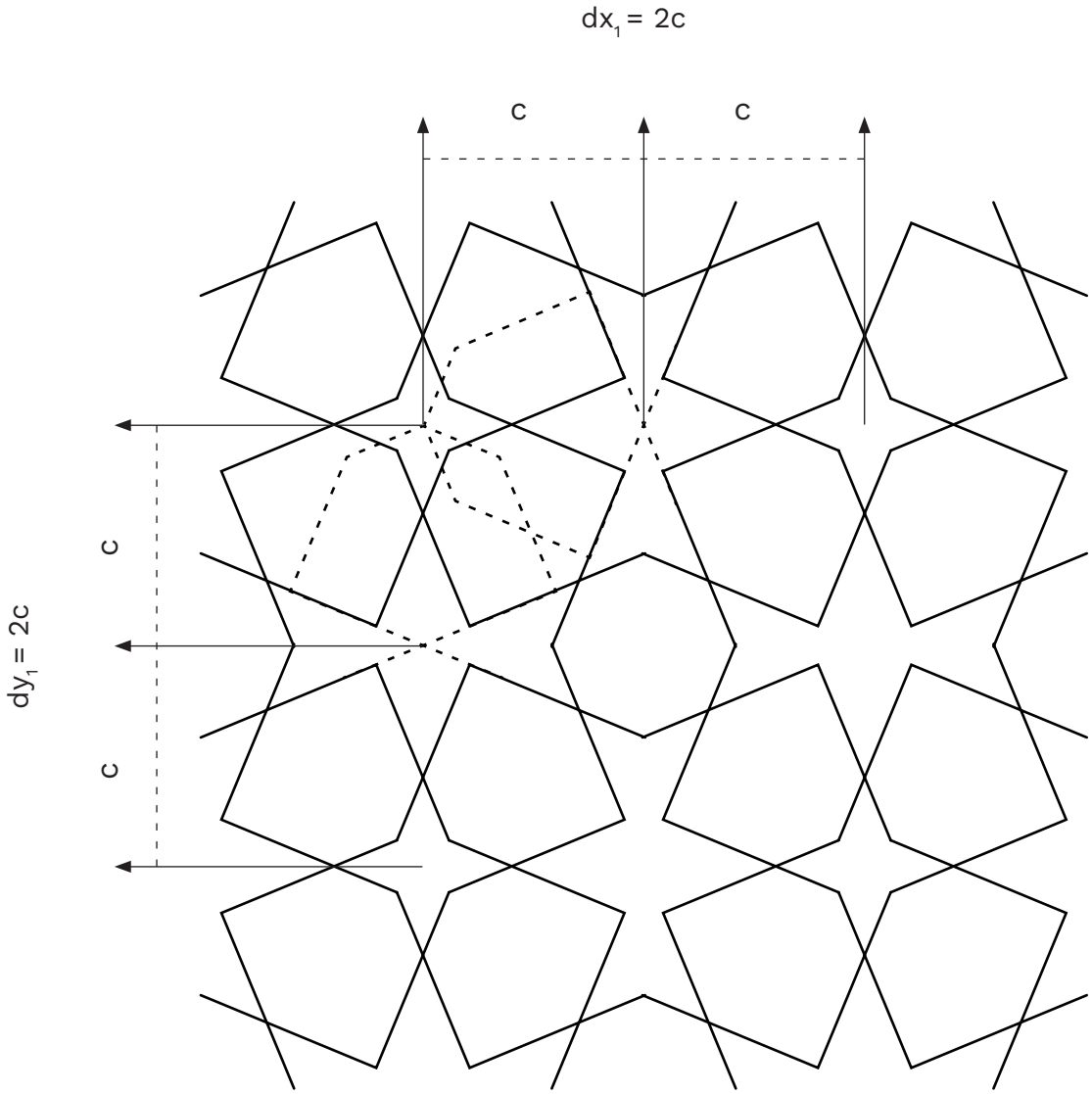
$y_4 = e \times \sin(22.5) + c$

Motifi Oluşturmak

```
let a, b, c, d, e;

function setup() {
  createCanvas(400, 400);
  angleMode(DEGREES);
  noFill();
  c = 120;
  a = 20;
  b = a*sin(112.5)/sin(22.5);
  d = a*sin(45)/sin(22.5);
  e = ((c-b)*sin(22.5))/sin(45);
}

function draw() {
  background(255);
  push();
  translate(width * 0.5, height * 0.5);
  rotate(45);
  for (let n = 0; n < 4; n++) {
    push();
    rotate(90*n);
    let mirror = 1;
    for(let i = 0; i < 2; i++){
      //loop for mirroring
      beginShape();
      let x0 = 0*mirror;
      let y0 = -a;
      vertex(x0, y0);
      let x1 = b * cos(45)*mirror;
      let y1 = -b * sin(45);
      vertex(x1, y1);
      let x2 = ((d+(c-b)*cos(22.5))* cos(67.5) + a)*mirror;
      let y2 = -1*((d+(c-b)*cos(22.5))* sin(67.5));
      vertex(x2, y2);
      let x3 = 0*mirror;
      let y3 = -c;
      vertex(x3, y3);
      let x4 = e * cos(22.5)*mirror;
      let y4 = -1*(e * sin(22.5) + c);
      vertex(x4, y4);
      endShape();
      mirror = mirror * -1;
    }
    pop();
  }
  pop();
  noLoop();
}
```

Bezeme Yapısını İnceleyelim

Aşama 2 : Yukarı ve aşağıya kaymaları belirleyecek xoffset ve yoffset değerlerini hesaplayalım.



Bezeme Kodu

```
/*
Code written by Selcuk ARTUT 2022
Geometric Patterns with Creative Coding
All rights reserved
*/


let tiles = []; // Declare array
let nRow;
let nCol;
let a = 20;
let c = 120;

// Tile class
class Tile {
  constructor(a,c) {
    this.a  = a;
    this.c  = c;
    this.b = this.a*sin(112.5)/sin(22.5);
    this.d = this.a*sin(45)/sin(22.5);
    this.e =  ((this.c-this.b)*sin(22.5))/sin(45);
  }

  display() {
    rotate(45);
    for (let n = 0; n < 4; n++) {
      push();
      rotate(90*n);
      let mirror = 1;
      for(let i = 0; i < 2; i++){
        //loop for mirroring
        beginShape();
        let x0 = 0*mirror;
        let y0 = -this.a;
        vertex(x0, y0);
        let x1 = this.b * cos(45)*mirror;
        let y1 = -this.b * sin(45);
        vertex(x1, y1);
        let x2 = ((this.d+(this.c-this.b)*cos(22.5))* cos(67.5) + this.a)*mirror;
        let y2 = -1*((this.d+(this.c-this.b)*cos(22.5))* sin(67.5));
        vertex(x2, y2);
        let x3 = 0*mirror;
        let y3 = -this.c;
        vertex(x3, y3);
        let x4 = this.e * cos(22.5)*mirror;
        let y4 = -1*(this.e * sin(22.5) + this.c);
        vertex(x4, y4);
        endShape();
```

```
        mirror = mirror * -1;
      }
      pop();
    }

  }
}



function setup() {
  createCanvas(1080, 1080);
  angleMode(DEGREES);
  noFill();
  strokeWeight(1);
  nRow = floor(height / (2*c));
  nCol = floor(width / (2*c));
  for (let i = 0; i < nRow * nCol; i++) {
    tiles.push(new Tile(a,c));
  }
}

function draw() {
  background(255);

  for (let r = 0; r < nRow; r++) {
    for (let k = 0; k < nCol; k++) {
      push();
      translate(k * c * 2,  c * r * 2);
      tiles[r+k*nRow].display();
      pop();
    }
  }
  noLoop();
}
```