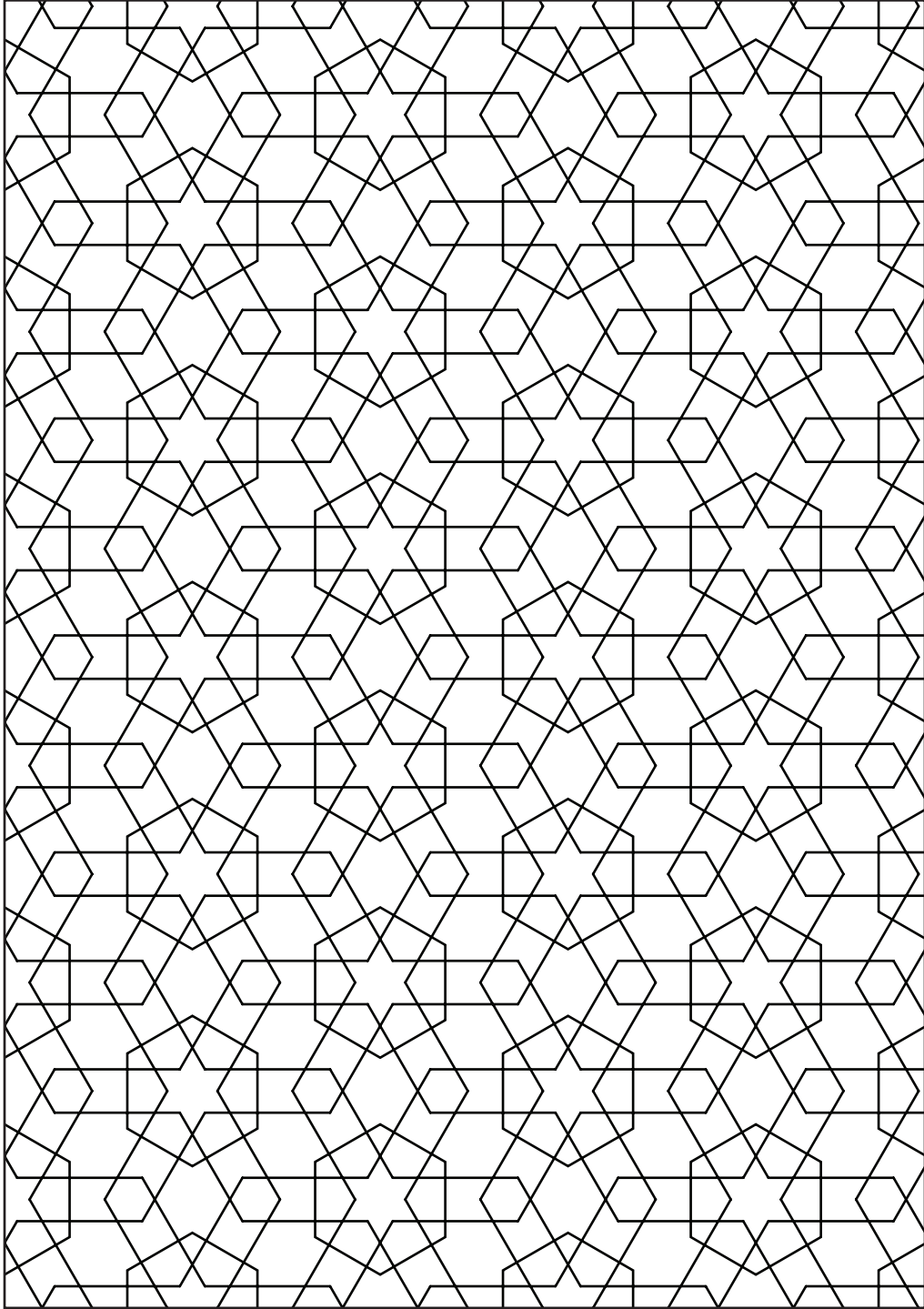
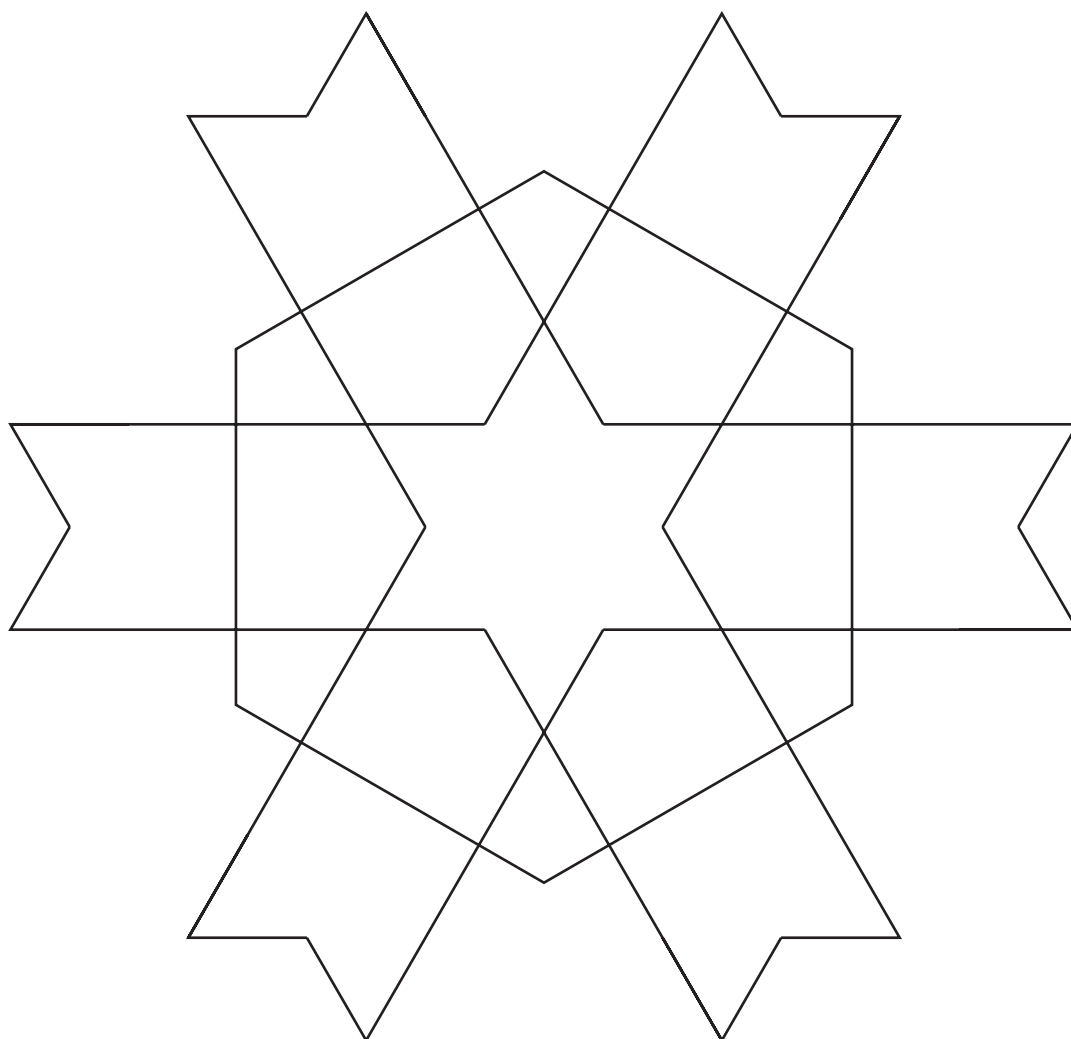


Geometrik Deseni Kodlamak

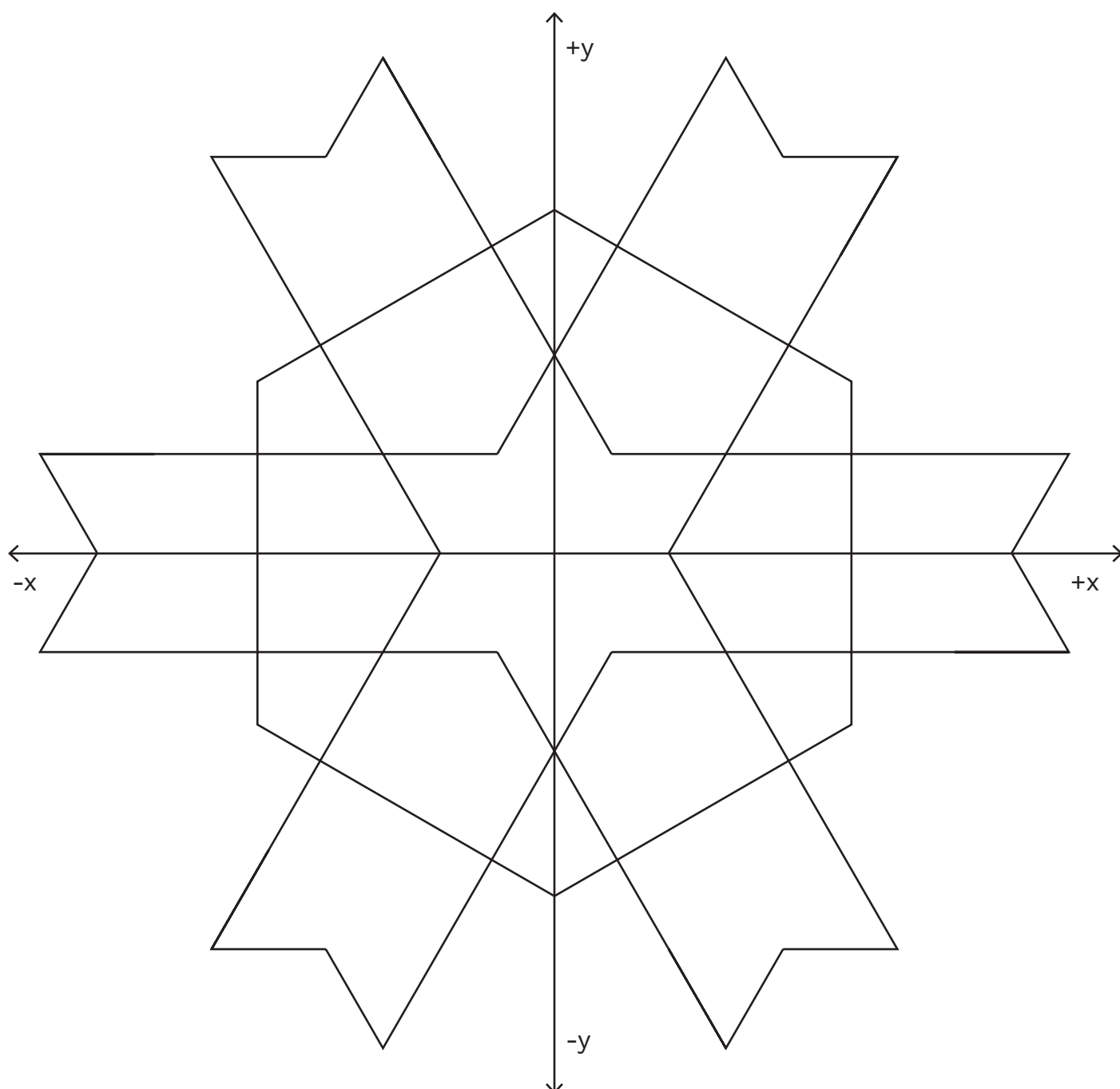
Aşağıdaki deseni inceleyin ve bu deseni oluşturan temel görsel bileşeni bulmaya çalışın.



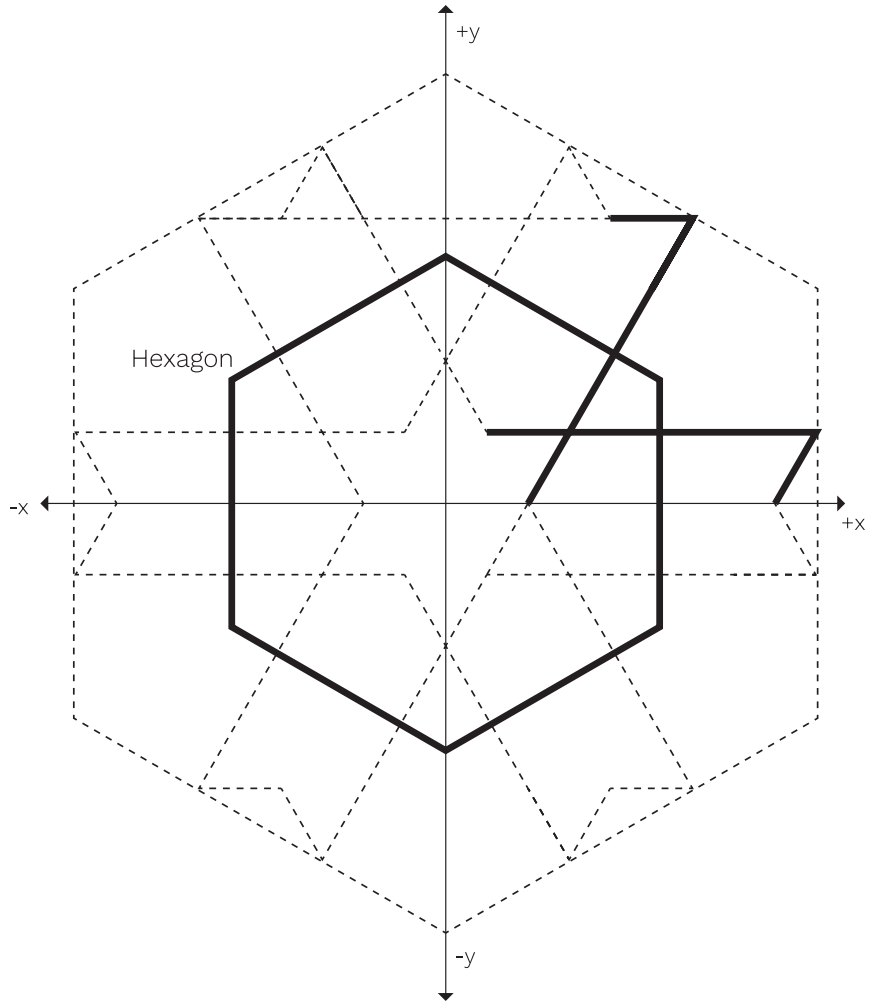
Motif



Motif

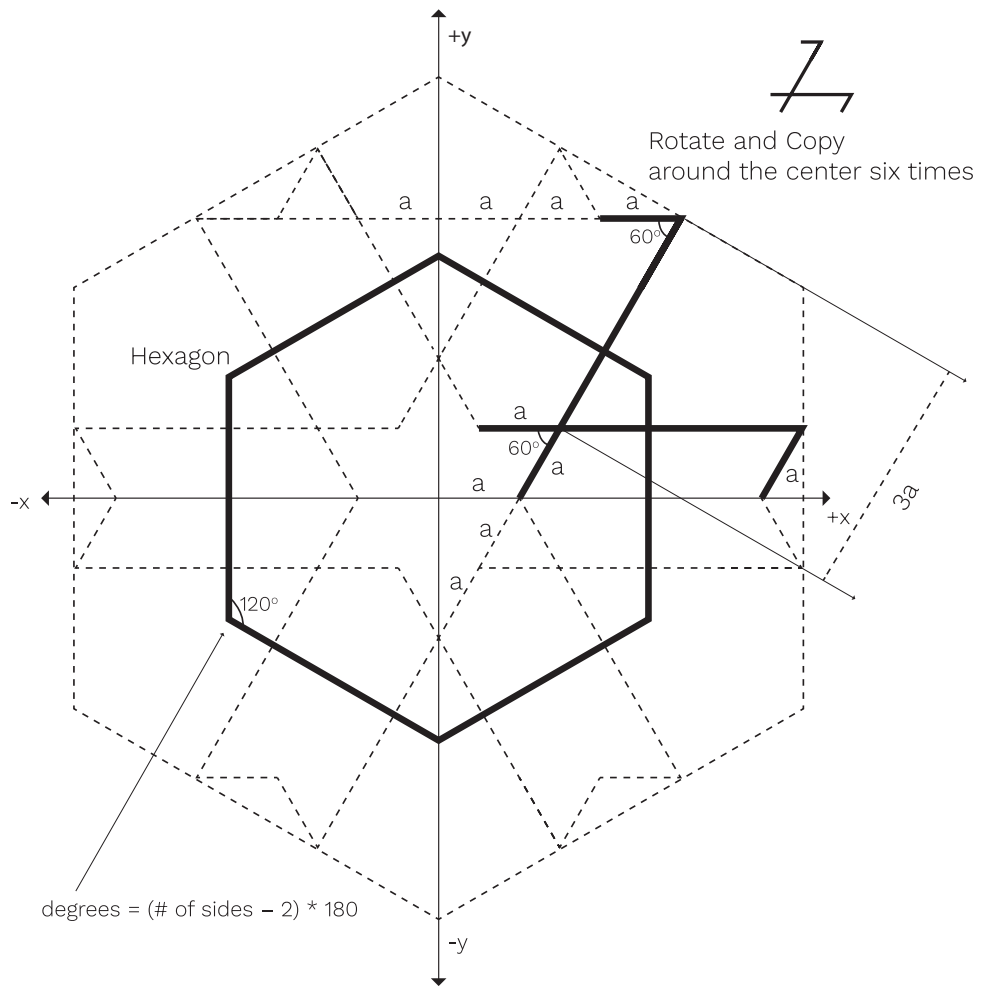


Temel Görsel Bileşini İnceleyelim



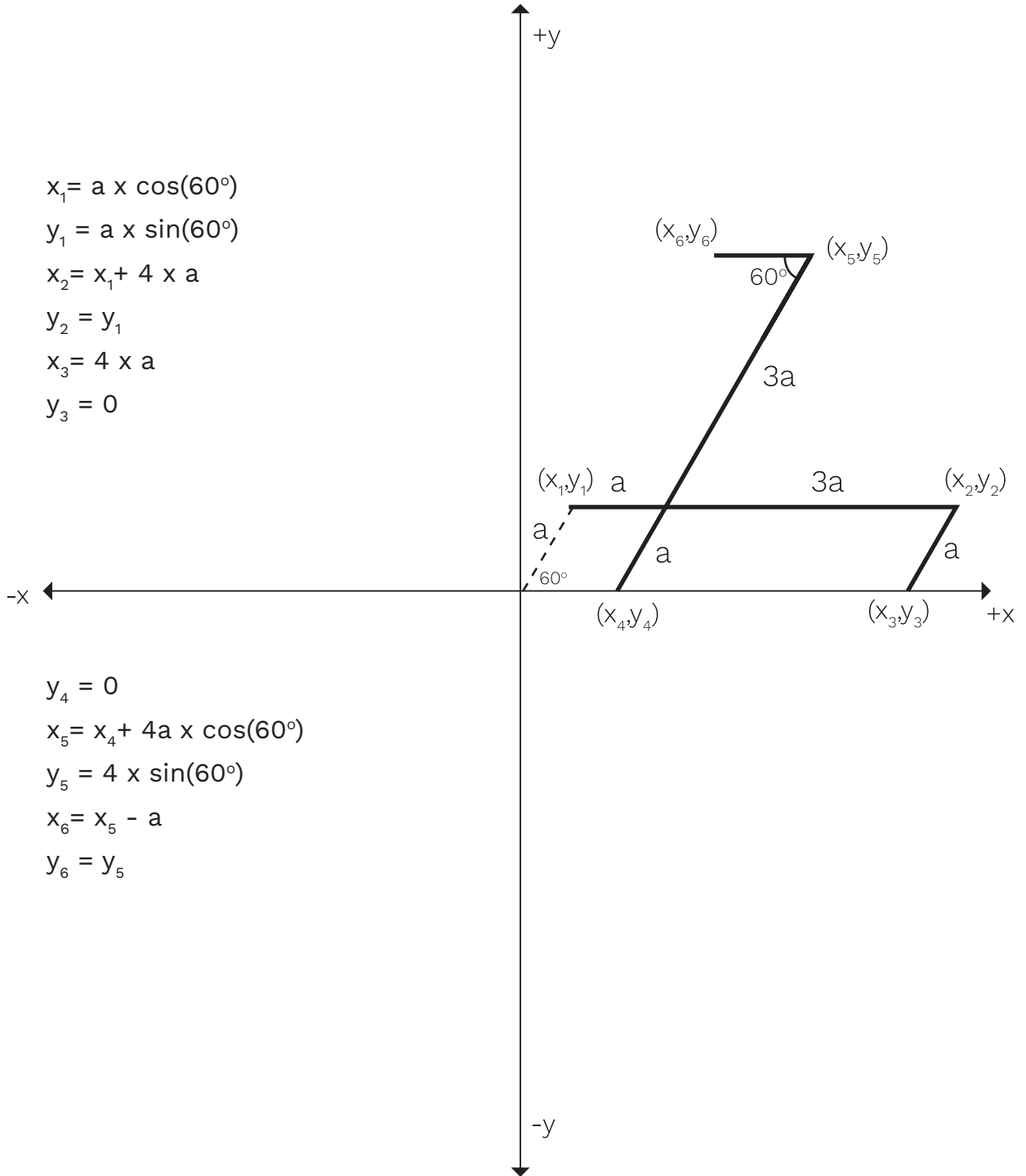
Açıları ve Vertex noktalarını tespit etmek

Aşama 1 : Vertex noktalarını bulalım



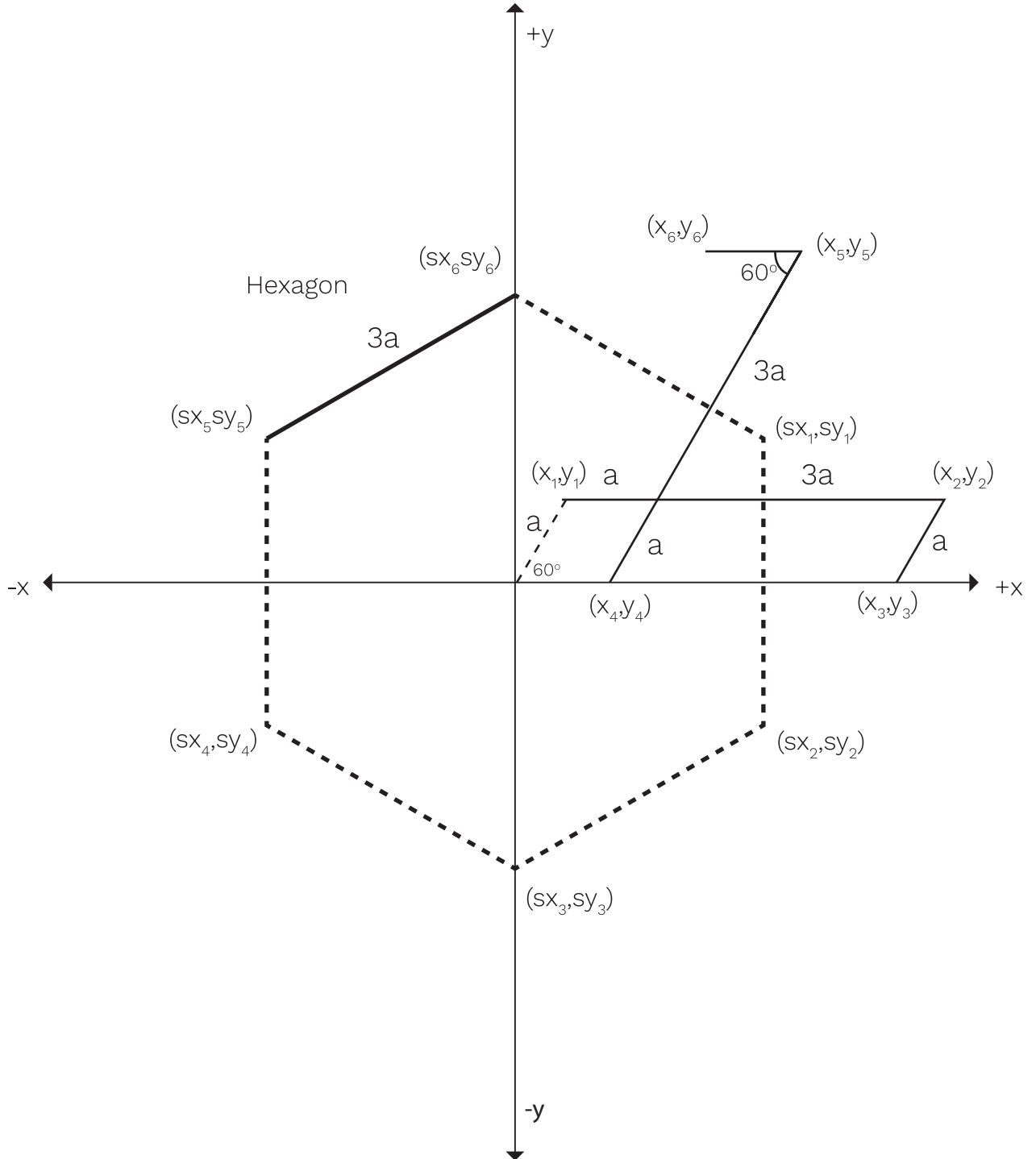
Açıları ve Vertex noktalarını tespit etmek

Aşama 2 : Önce aşağıdaki şekli oluşturalım



Açıları ve Vertex noktalarını tespit etmek

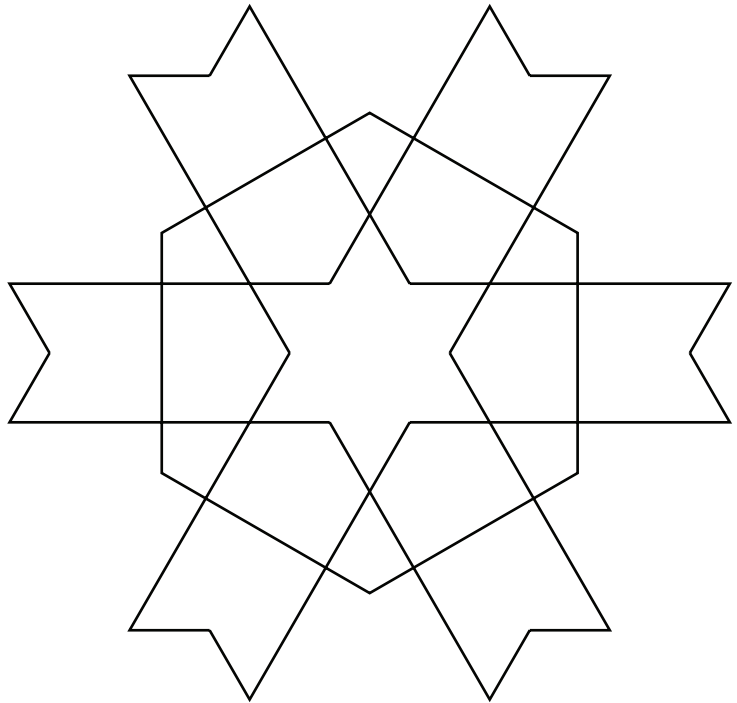
Aşama 3: Şimdi altıgene konsantre olalım



Motifi Oluşturmak

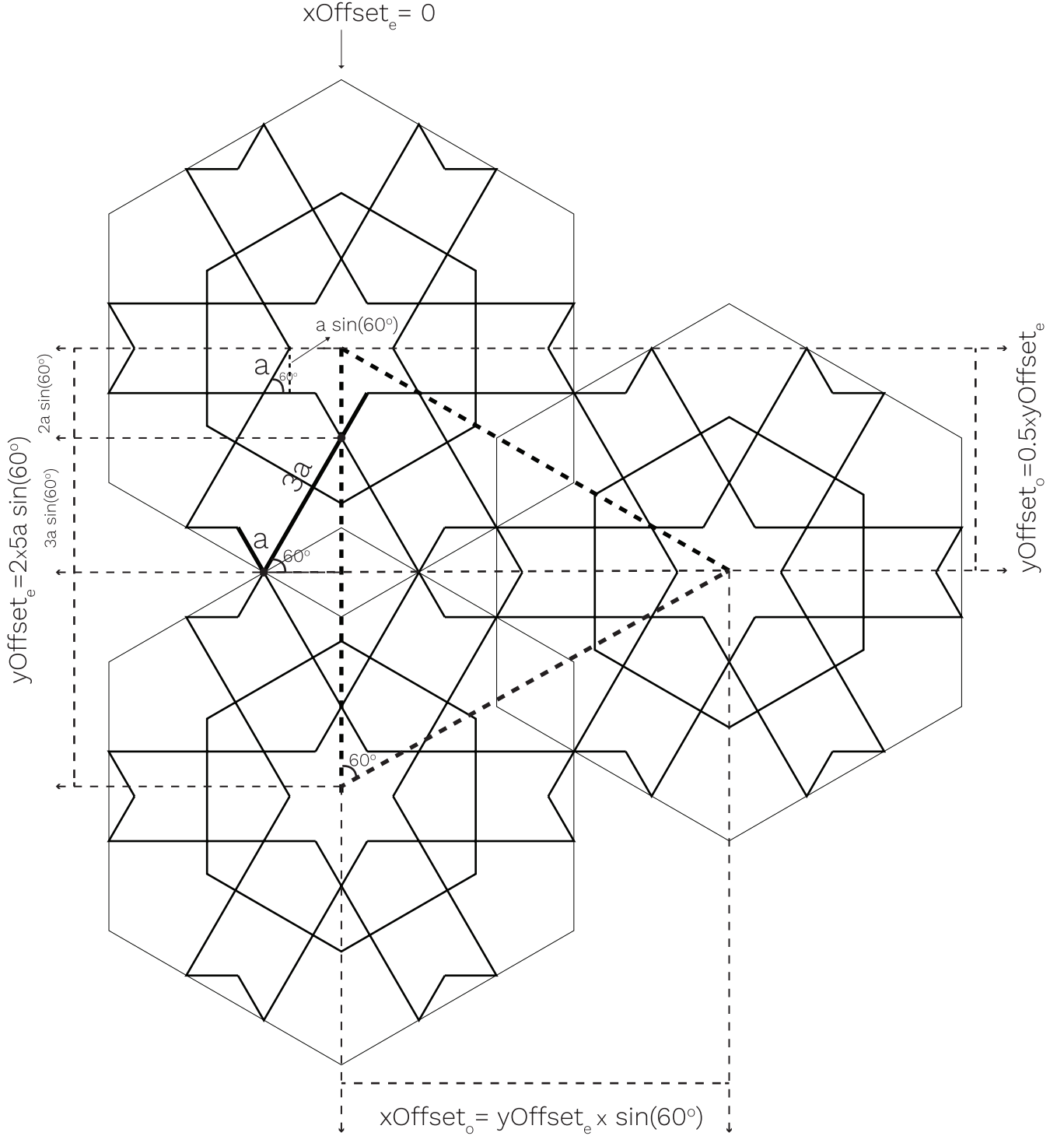
```
let a = 50; //scale factor
function setup() {
  createCanvas(600, 600);
  angleMode(DEGREES);
}
```

```
function draw() {
  background(255);
  stroke(0);
  noFill();
  push();
  translate(width * 0.5, height * 0.5); //center on screen
  for (let i = 0; i < 6; i++) {
    push();
    rotate(60 * i);
    beginShape();
    let x1 = a * cos(60);
    let y1 = -a * sin(60);
    let x2 = x1 + 4 * a;
    let y2 = y1;
    let x3 = 4 * a;
    let y3 = 0;
    vertex(x1, y1);
    vertex(x2, y2);
    vertex(x3, y3);
    endShape();
    beginShape();
    let x4 = a;
    let y4 = 0;
    let x5 = x4 + 4 * a * cos(60);
    let y5 = -4 * a * sin(60);
    let x6 = x5 - a;
    let y6 = y5;
    vertex(x4, y4);
    vertex(x5, y5);
    vertex(x6, y6);
    endShape();
    pop();
  }
  //hexagon
  rotate(90);
  let angle = 360 / 6;
  beginShape();
  for (let ang = 0; ang < 360; ang += angle) {
    let sx = cos(ang) * 3 * a;
    let sy = sin(ang) * 3 * a;
    vertex(sx, sy);
  }
  endShape(CLOSE);
  pop();
  noLoop();
}
```



Bezeme Yapısını İnceleyelim

Aşama 4 : Yukarı ve aşağıya kaymaları belirleyecek xoffset ve yoffset değerlerini hesaplayalım.



Bezeme Kodu

```
// Motif class
class Motif{
  constructor(r) {
    this.a = r;
  }

  display() {
    for (let i = 0; i < 6; i++) {
      push();
      rotate(60 * i);
      beginShape();
      let x1 = this.a * cos(60);
      let y1 = -this.a * sin(60);
      let x2 = x1 + 4 * this.a;
      let y2 = y1;
      let x3 = 4 * this.a;
      let y3 = 0;
      vertex(x1, y1);
      vertex(x2, y2);
      vertex(x3, y3);
      endShape();
      beginShape();
      let x4 = this.a;
      let y4 = 0;
      let x5 = x4 + 4 * this.a * cos(60);
      let y5 = -4 * this.a * sin(60);
      let x6 = x5 - this.a;
      let y6 = y5;
      vertex(x4, y4);
      vertex(x5, y5);
      vertex(x6, y6);
      endShape();
      pop();
    }

    //hexagon
    rotate(90);
    let angle = 360 / 6;
    beginShape();
    for (let ang = 0; ang < 360; ang += angle) {
      let sx = cos(ang) * 3 * a;
      let sy = sin(ang) * 3 * a;
      vertex(sx, sy);
    }
    endShape(CLOSE);
  }
}
```

```

let motives = []; // Declare array
let nRow;
let nCol;
let a = 16;
let xoffsetEven, yoffsetEven, xoffsetOdd, yoffsetOdd;

function setup() {
  createCanvas(600, 600);
  angleMode(DEGREES);

  //even columns 0,2,4,...
  //odd column 1,3,5,...
  xoffsetEven = 0;
  yoffsetEven = 2 * 5 * a * sin(60);
  xoffsetOdd = yoffsetEven * sin(60);
  yoffsetOdd = yoffsetEven * 0.5;

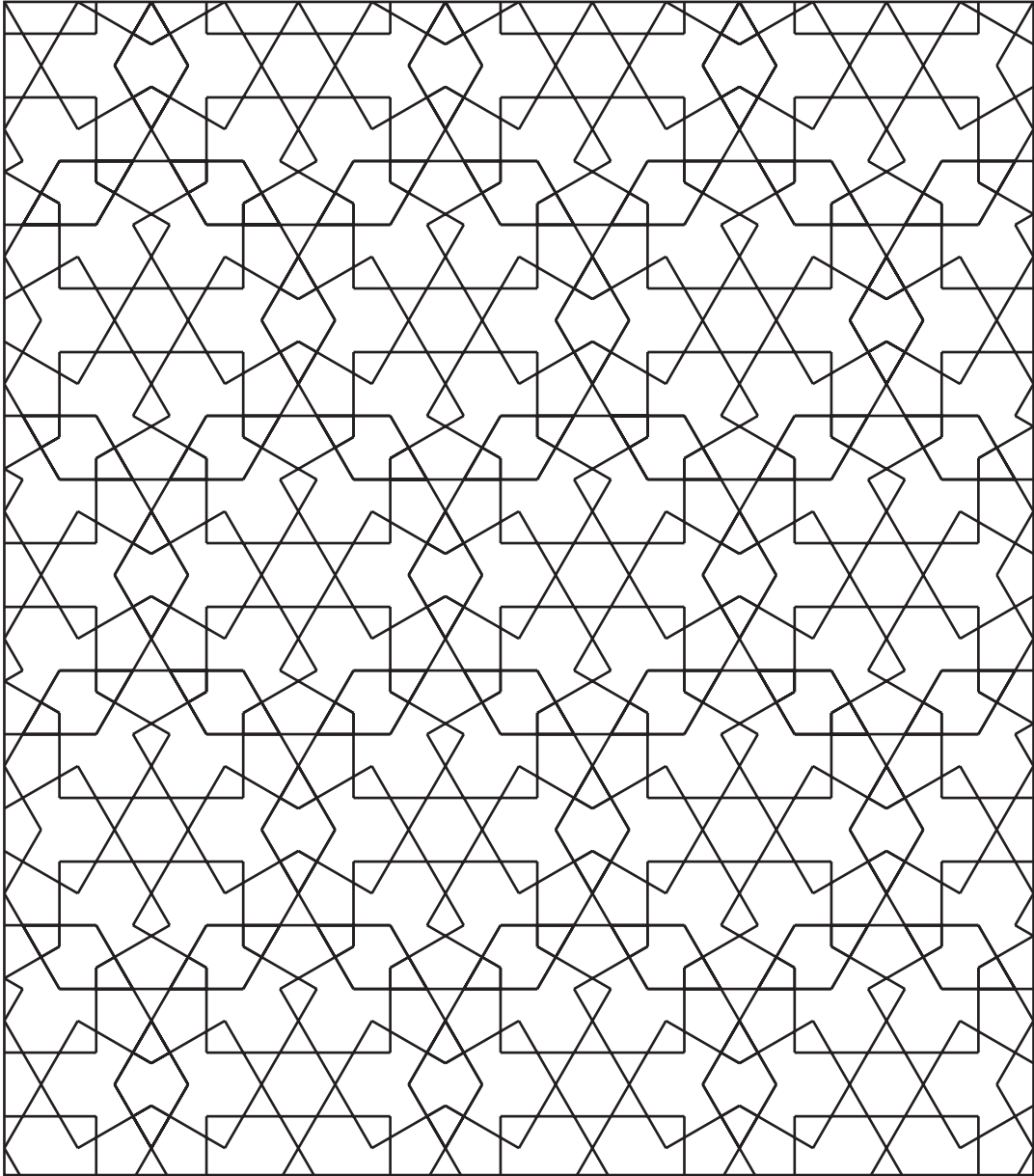
  //approximate the nRow and nCol values
  //try and see the results
  nRow = ceil(width / xoffsetOdd);
  nCol = 1+ceil(height / yoffsetEven);
  //generate the motives array
  for (let i = 0; i < nRow * nCol; i++) {
    motives.push(new Motif(a));
  }
}

function draw() {
  background(0);
  stroke(255);
  noFill();
  noLoop();
  for (let c = 0; c < nCol; c++) {
    for (let r = 0; r < nRow; r++) {
      push();
      if (c % 2 == 0) {
        //columns 0,2,4,6
      } else {
        //columns 1,3,5,7
        translate(0, yoffsetOdd);
      }
      translate(xoffsetEven * c + xoffsetOdd * c, yoffsetEven * r);
      tiles[c + r * nCol].display();
      pop();
    }
  }
}

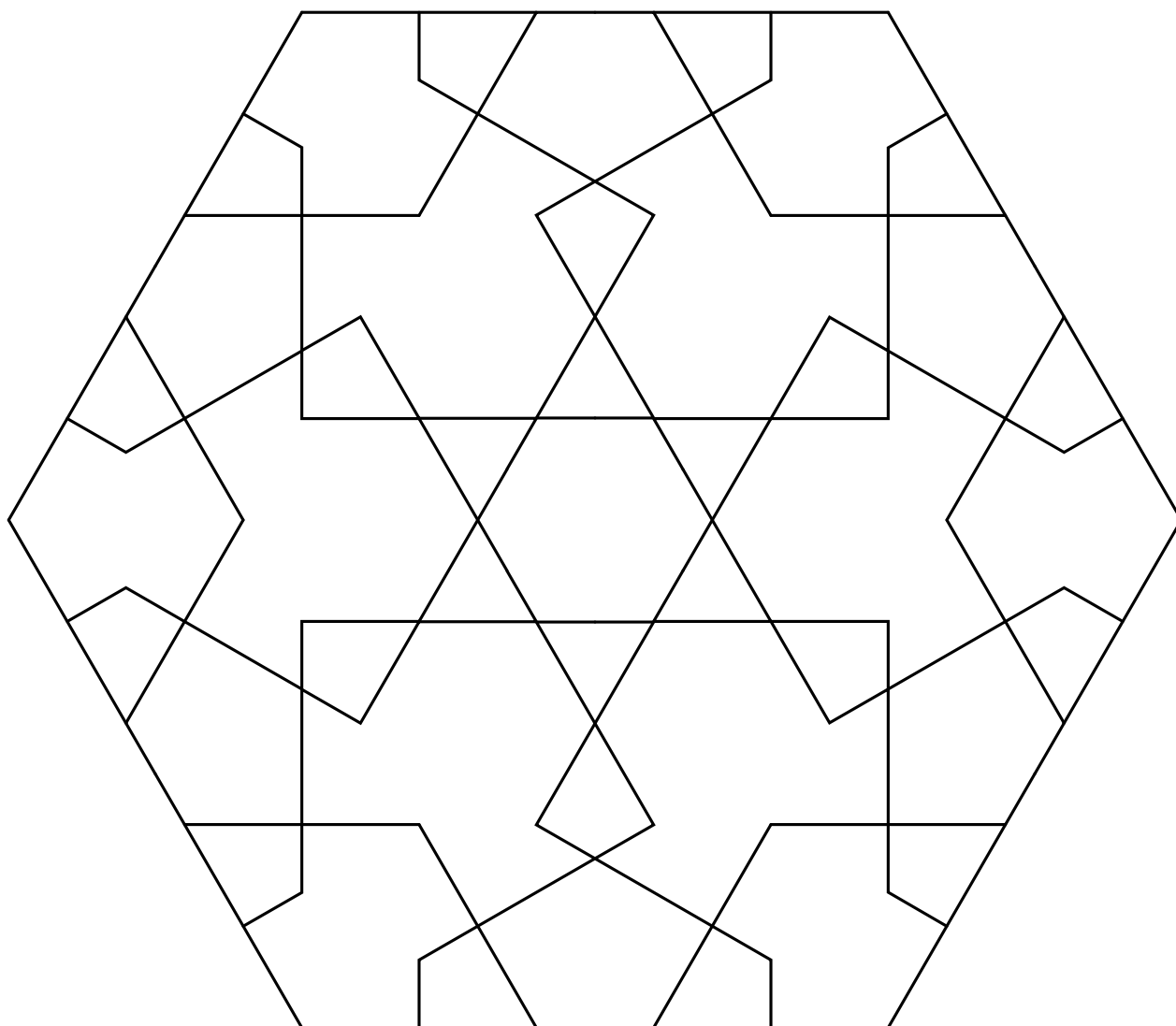
```

Geometrik Deseni Kodlamak

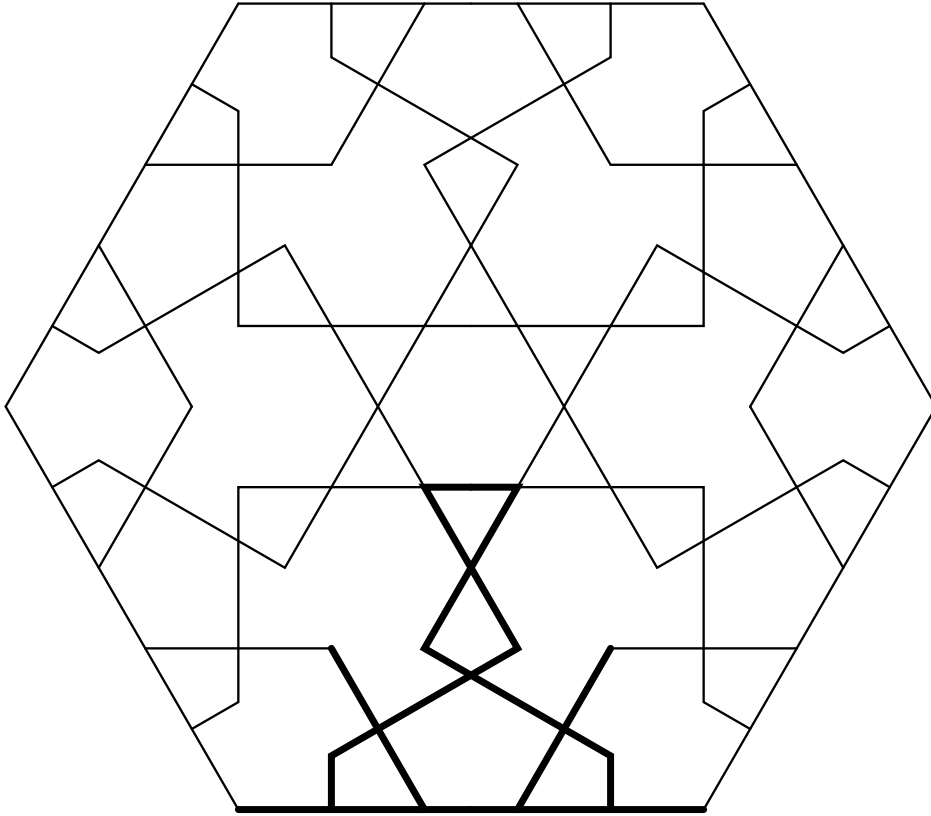
Aşağıdaki deseni inceleyin ve bu deseni oluşturan temel görsel bileşeni bulmaya çalışın.



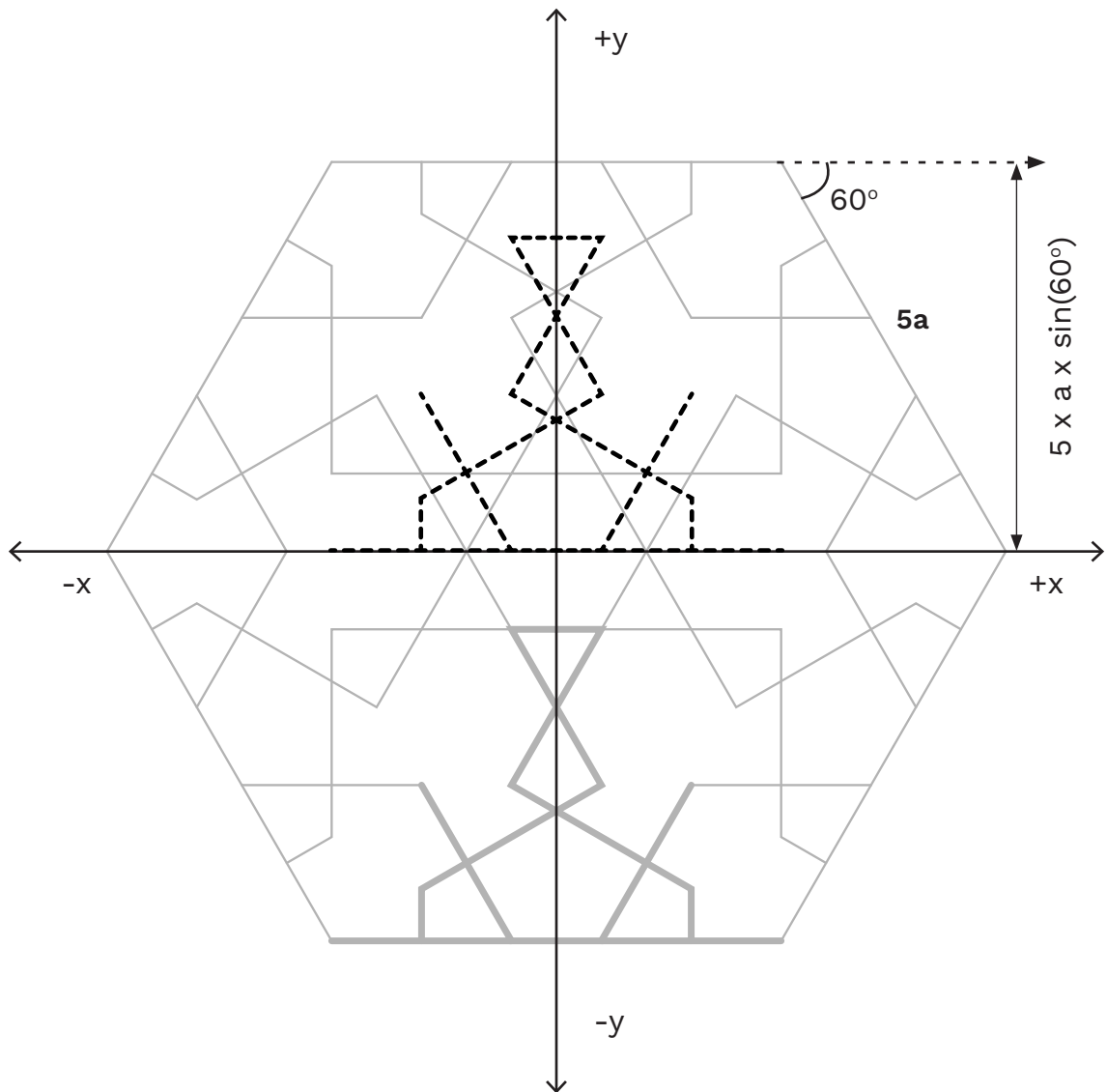
Motif



Temel Görsel Bileşini İnceleyelim



Aşama 2 : Motif aşağıya kaydırılmalı ve merkeze göre 60 derece rotasyon uygulanarak 6 defa oluşturulmalı



Motifi Oluşturmak

```
let a = 40;
```

```
function setup() {  
  createCanvas(600, 600);  
  noFill();  
  angleMode(DEGREES);  
}  
function draw() {  
  let x0,y0,x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,x8,y8,x9,y9,x10,y10;  
  background(255);  
  push();  
  translate(width*0.5, height*0.5);  
  x0 = 0;  
  y0 = 0;  
  x1 = 0.5 * a;  
  y1 = 0;  
  x2 = 2 * a * cos(60) + 0.5 * a;  
  y2 = -(2 * a * sin(60));  
  x3 = 1.5 * a;  
  y3 = 0;  
  x4 = 2.5 * a;  
  y4 = 0;  
  x5 = 1.5 * a;  
  y5 = -a / tan(60);  
  x6 = 0;  
  y6 = (-2.5 * a) / tan(60);  
  x7 = a * sin(30);  
  y7 = -(2 * a * sin(60));  
  x8 = 0;  
  y8 = -(3 * a * sin(60));  
  x9 = a * sin(30);  
  y9 = -(4 * a * sin(60));  
  x10 = 0;  
  y10 = -(4 * a * sin(60));  
  
  for (let i = 0; i < 6; i++) {  
    push();  
    rotate(i * 60);  
    translate(0, 5 * a * sin(60));  
    beginShape();  
    vertex(x0, y0);  
    vertex(x1, y1);  
    vertex(x2, y2);  
    endShape();  
  
    beginShape();  
    vertex(x1, y1);  
    vertex(x4, y4);  
    endShape();
```

```

beginShape();
vertex(x3, y3);
vertex(x5, y5);
vertex(x6, y6);
vertex(x7, y7);
vertex(x8, y8);
vertex(x9, y9);
vertex(x10, y10);
endShape();

//mirror on y axis
beginShape();
vertex(-x0, y0);
vertex(-x1, y1);
vertex(-x2, y2);
endShape();

beginShape();
vertex(-x1, y1);
vertex(-x4, y4);
endShape();

beginShape();
vertex(-x3, y3);
vertex(-x5, y5);
vertex(-x6, y6);
vertex(-x7, y7);
vertex(-x8, y8);
vertex(-x9, y9);
vertex(-x10, y10);
endShape();

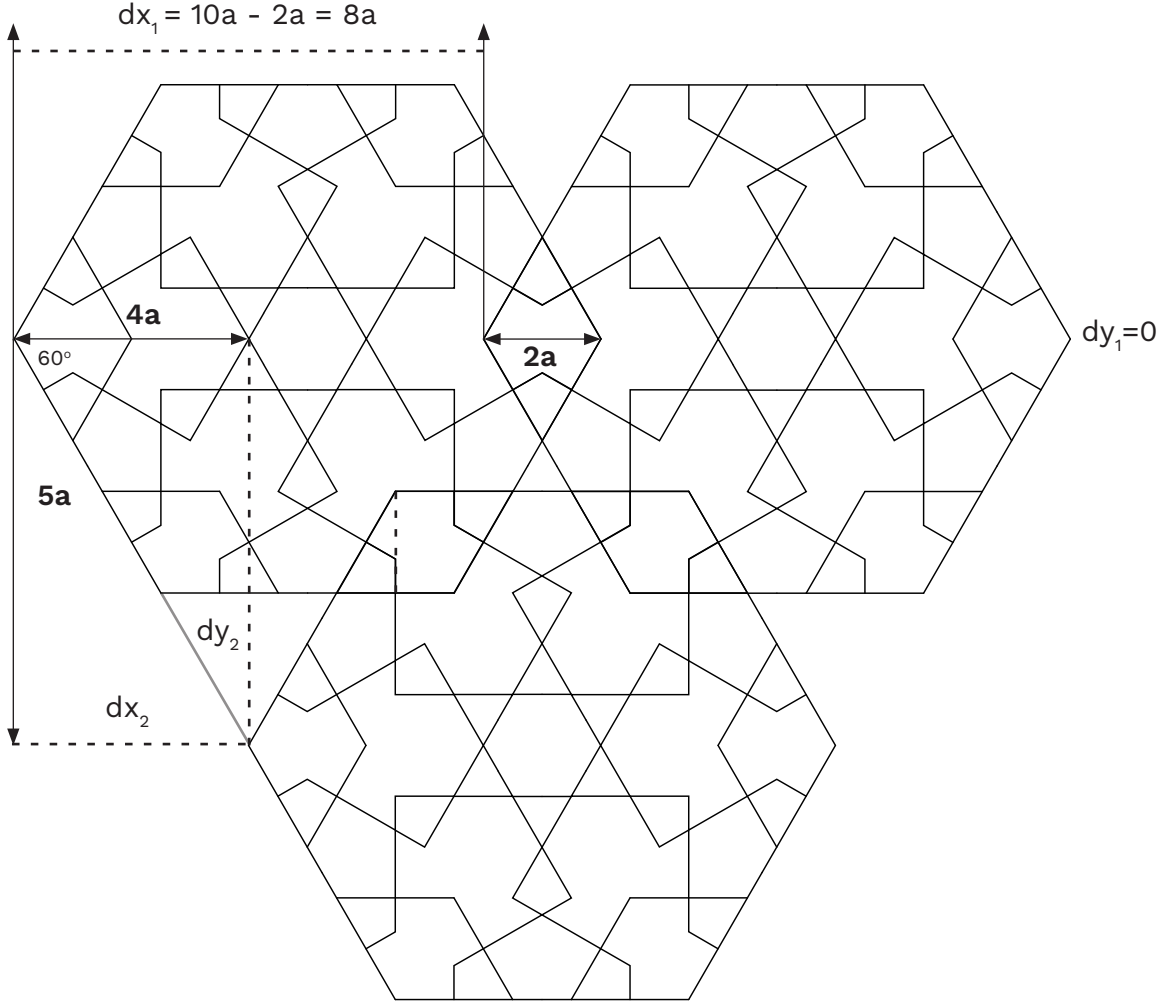
pop();
}

pop();
noLoop();
}

```

Bezeme Yapısını İnceleyelim

Step 3 : Yukarı ve aşağıya kaymaları belirleyecek xoffset ve yoffset değerlerini hesaplayalım.



$$dx_1 = 8a$$

$$dy_1 = 0$$

$$dx_2 = 4a$$

$$dy_2 = \tan(60^\circ) \times 4a$$

Bezeme Kodu

```
/*  
Code written by Selcuk ARTUT 2022  
Geometric Patterns with Creative Coding  
All rights reserved  
*/  
// Tile class  
class Tile {  
  constructor(r) {  
    this.a = r;  
  }  
  
  display() {  
  
    let x0,y0,x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,x8,y8,x9,y9,x10,y10;  
    x0 = 0;  
    y0 = 0;  
    x1 = 0.5 * this.a;  
    y1 = 0;  
    x2 = 2 * this.a * cos(60) + 0.5 * this.a;  
    y2 = -(2 * this.a * sin(60));  
    x3 = 1.5 * this.a;  
    y3 = 0;  
    x4 = 2.5 * this.a;  
    y4 = 0;  
    x5 = 1.5 * this.a;  
    y5 = -this.a / tan(60);  
    x6 = 0;  
    y6 = (-2.5 * this.a) / tan(60);  
    x7 = this.a * sin(30);  
    y7 = -(2 * this.a * sin(60));  
    x8 = 0;  
    y8 = -(3 * this.a * sin(60));  
    x9 = this.a * sin(30);  
    y9 = -(4 * this.a * sin(60));  
    x10 = 0;  
    y10 = -(4 * this.a * sin(60));  
    for (let i = 0; i < 6; i++) {  
      push();  
      rotate(i * 60);  
      translate(0, 5 * this.a * sin(60));  
  
      beginShape();  
      vertex(x0, y0);  
      vertex(x1, y1);  
      vertex(x2, y2);  
      endShape();  
  
      beginShape();  
      vertex(x1, y1);
```

```
vertex(x4, y4);  
endShape();
```

```
beginShape();  
vertex(x3, y3);  
vertex(x5, y5);  
vertex(x6, y6);  
vertex(x7, y7);  
vertex(x8, y8);  
vertex(x9, y9);  
vertex(x10, y10);  
endShape();
```

```
//mirror on y axis  
beginShape();  
vertex(-x0, y0);  
vertex(-x1, y1);  
vertex(-x2, y2);  
endShape();
```

```
beginShape();  
vertex(-x1, y1);  
vertex(-x4, y4);  
endShape();
```

```
beginShape();  
vertex(-x3, y3);  
vertex(-x5, y5);  
vertex(-x6, y6);  
vertex(-x7, y7);  
vertex(-x8, y8);  
vertex(-x9, y9);  
vertex(-x10, y10);  
endShape();
```

```
pop();  
}  
}  
}
```

```
let tiles = []; // Declare array  
let nRow;  
let nCol;  
let dx1, dy1, dx2, dy2;
```

```
let r = 16;
```

```

function setup() {
  createCanvas(1080, 1080);
  angleMode(DEGREES);
  noFill();
  strokeWeight(1);

  nRow = floor(height / (2*r));
  nCol = floor(width / (2*r));

  for (let i = 0; i < nRow * nCol; i++) {
    tiles.push(new Tile(r));
  }

  dx1 = 8.0*r;
  dy1 = 0;
  dx2 = 4.0*r;
  dy2 = 4.0*r*tan(60);

}

function draw() {
  background(255);
  stroke(0);
  for (let r = 0; r < nRow; r++) {
    for (let c = 0; c < nCol; c++) {
      push();
      translate(c * dx1, dy1 + dy2*r);
      if (r % 2 == 1) {
        //rows 1,3,5,7
        translate(dx2,0);
      }
      tiles[r + c * nRow].display();
      pop();
    }
  }
}

```

Geometrik Deseni Kodlamak

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

