```
// Author: Laura Scholl, 1/7/2020
// Written in P5.js

let myTriangle = [];
var c, numRows, numColumns, halfSide;
const ellipseRad = 200;
let lastEllipse = -1;

function row(yPos, r) {
  // declare and initialize variables local to the function row();
  var radians = 0;
  var localCenterX = 0;
  var localCenterY = yPos;

  for (let i = 0; i < numColumns; i++) {
    if (i % 2 == 0) {
      radians = PI;
      localCenterY = yPos + halfSide / 2;
    } else {
      radians = 0;
      localCenterY = yPos + halfSide;
    }
    myTriangle[r][i].display(radians, localCenterX, localCenterY);
    localCenterX = localCenterX + halfSide;
  }
}

function setup() {
  //createCanvas(windowWidth, windowHeight);
  createCanvas(640, 360);
  c = color(0, 0, 0);
  background(c);
  halfSide = width / 8;
  numRows = width / halfSide + 1;
  numColumns = height / (height / 8) + 1;
  strokeWeight(2);
  for (let i = 0; i < numRows; i++) {
    myTriangle[i] = [];
```

```
    for (let j = 0; j < numColumns; j++) {
      // initialize each object by calling its constructor
      myTriangle[i][j] = new ConcentricTri(4, halfSide);
    }
  }
  // the draw loop stops after the first time through
  //noLoop();
  colorMode(HSB, width, height, 100);
}
function draw() {

  let whichEllipse = mouseX / ellipseRad;
  if (whichEllipse !== lastEllipse) {
    noStroke();
    let ellipseX = whichEllipse * ellipseRad;
    c = color(ellipseX, mouseY, 72);
    fill(c);
    ellipse(mouseX, mouseY, ellipseRad, ellipseRad);
    lastEllipse = whichEllipse;
  }
  let y = 0;
  for (let i = 0; i < numRows; i++) {
    // call the row function to create rows of triangles
    row(y, i);
    y = y + (halfSide * 1.6);
  }
}

class ConcentricTri {
  constructor(pNum, pRadius) {
    this.numTri = pNum;
    this.radius = pRadius;
  }
  display(angle, centerX, centerY) {
    let localRadius = this.radius;
    var tp1X, tp1Y, tp2X, tp2Y, tp3X, tp3Y;

    push();
```

```
    translate(centerX, centerY - 1);
    rotate(angle);
    for (let i = 0; i < this.numTri; i++) {
      tp1X = (cos(radians(30)) * localRadius);
      tp1Y = (sin(radians(30)) * localRadius);
      tp2X = (cos(radians(150)) * localRadius);
      tp2Y = (sin(radians(150)) * localRadius);
      tp3X = (cos(radians(270)) * localRadius);
      tp3Y = (sin(radians(270)) * localRadius);
      stroke(255);
      fill(0);
      triangle(tp1X, tp1Y, tp2X, tp2Y, tp3X, tp3Y);
      localRadius = localRadius - (this.radius / this.numTri);
    }
    pop();
  }
}
```