```
// Elliptical Orbit Simulation
// This is a simulation of an orbiting body around a large body, such
// as a planet or comet around the sun.
// shared under Creative Commons license
// https://creativecommons.org/licenses/by-nc-sa/3.0/
// Written for Processing by Jason Galbraith
// Rewritten for p5.js by Sophia Wang, 10.16.2025

/*
************************************************************
Change the numbers below to see how it changes the simulation!
************************************************************
 */

// The following variable is the eccentricity of the orbit.
// An eccentricity of 0 is a circle, while an eccentricity of 0.9 is a
// narrow ellipse.
let eccentricity = 0.39;

// The following is how many trailing dots to plot.
let TRAILLENGTH = 10;

// This is how much of an area the planet will sweep out at each step.
let DELTA_T = 5000;

// This is how much to delay the simulation between steps.
let DELAY = 20;

// This is the planet size in pixels.
let planetSize = 5;

// This is the sun size (focus) in pixels
let sunSize = 30;

/*
**************************************************************
Below here is the actual simulation. You probably shouldn't mess
with this section. Or at least, you should expect weird results.
**************************************************************
 */

let WIDTH = 600, HEIGHT = 600;
let a = WIDTH/2 * 0.8; //center to major
let c = eccentricity * a; //focus
```

```
let b = Math.sqrt((a*a)-(c*c)); //center to minor
let x_trail = new Array(TRAILLENGTH);
let y_trail = new Array(TRAILLENGTH);
let counter = 0, currentAngle = 0, currentTrail = 0, currentDelay = 0;

// setup function called initially, only once
function setup() {
  createCanvas(600, 600);
  background(0);
  ellipseMode(RADIUS);
  for (let i = 0; i < TRAILLENGTH; i++) {//Set trail to draw off screen
     x_trail[i] = -100;
     y_trail[i] = -100;
  }
}

// Draw function loops
function draw() {
  currentDelay++;
  if (currentDelay == DELAY) {//If it is time to draw the ellipse
      currentDelay = 0;
      //These are the x and y coordinates on the ellipse
      let startX = x_trail[currentTrail]; //start from most recent position in trail
      let startY = y_trail[currentTrail];
      let finalX = x_trail[currentTrail];
      let finalY = y_trail[currentTrail];
      currentTrail++; //set new trail position
      if (currentTrail == TRAILLENGTH) {
          currentTrail = 0;
      }
      let totalArea = 0.0;
      let changeInAngle = 0.0;
      let startH = 0.0; //hypotenuse of first line from focus
      let finalH = 0.0; //hypotenuse of second line from focus
      let area = 0.0;
      let angleRadians = (currentAngle / 180.0) * PI;
      let startChangeX = startX-(WIDTH/2+c); //x position from focus of first line
      let startChangeY = (HEIGHT/2)-startY; //y position from focus of first line
      let startAngle = Math.atan(Math.abs(startChangeY)/Math.abs(startChangeX)); // angle of
first line from focus
      if ((startChangeX <= 0) && (startChangeY >= 0)) { //figure out what quadrant you are in
          startAngle = PI/2-startAngle + PI/2;
      }
      else if ((startChangeX <= 0) && (startChangeY <= 0)) {
```

```
        startAngle = PI + startAngle;
    }
    else if ((startChangeX >= 0) && (startChangeY <= 0)) {
        startAngle = PI/2-startAngle + 3*PI/2;
    }
    let finalChangeX = finalX-(WIDTH/2+c); //X position from focus of second line
    let finalChangeY = (HEIGHT/2)-finalY; //Y position from focus of second line
    let finalAngle = startAngle;

    while (totalArea < DELTA_T) { //while you have not swept out enough area
      startX = finalX;
      startY = finalY;
      startChangeX = startX-(WIDTH/2+c);
      startChangeY = (HEIGHT/2)-startY;
      startAngle = finalAngle;
      currentAngle += 0.1; //change the angle slightly
      angleRadians = (currentAngle / 180.0) * PI; //degrees to radians
      finalX = a * cos(angleRadians) + WIDTH/2; //calculate x position of new point using center
of ellipse formula
      finalY = b * sin(angleRadians) + HEIGHT/2; //calculate y position of new point using center
of ellipse formula
      finalChangeX = finalX-(WIDTH/2+c); //calculate distance from focus in x
      finalChangeY = (HEIGHT/2)-finalY; //calculate distance from focus in y
      finalAngle = Math.atan(Math.abs(finalChangeY)/Math.abs(finalChangeX)); //angle from
second line to focus
      if ((finalChangeX <= 0) && (finalChangeY >= 0)) { //figure out your quadrant
        finalAngle = PI/2 - finalAngle + PI/2;
      }
      else if ((finalChangeX <= 0) && (finalChangeY <= 0)) {
        finalAngle = PI + finalAngle;
      }
      else if ((finalChangeX >= 0) && (finalChangeY <= 0)) {
        finalAngle = PI/2-finalAngle + 3*PI/2;
      }
      changeInAngle = startAngle - finalAngle; //calculate the change in angle
      startH = Math.sqrt(startChangeX*startChangeX+startChangeY*startChangeY); //figure out
the hypotenuse of the first line
      finalH = Math.sqrt(finalChangeX*finalChangeX+finalChangeY*finalChangeY); //figure out
the hypotenuse of the second line
      area = startH * finalH * Math.sin(changeInAngle) * 0.5; //calculate the area of that triangle
      totalArea += area; //Add that to the total area
    }//end while
    angleRadians = (currentAngle / 180.0) * PI;
    x_trail[currentTrail] = a * cos(angleRadians) + WIDTH/2; //figure out planet's x
```

```
    y_trail[currentTrail] = b * sin(angleRadians) + HEIGHT/2; //figure out planet's y
    background(0);
    stroke(255);
    fill(0);

    ellipse(WIDTH/2, HEIGHT/2, a, b); //draw ellipse
    fill('#ffffba');
    ellipse((WIDTH/2+c), HEIGHT/2, sunSize, sunSize);
    for (let i = 0; i < TRAILLENGTH; i++){
      fill(255);
      ellipse(x_trail[i], y_trail[i], planetSize, planetSize); //draw trail
    }
    stroke(0, 0, 255);
    fill(0, 0, 255);
    ellipse(x_trail[currentTrail], y_trail[currentTrail], planetSize, planetSize); //draw planet
  }
}
```