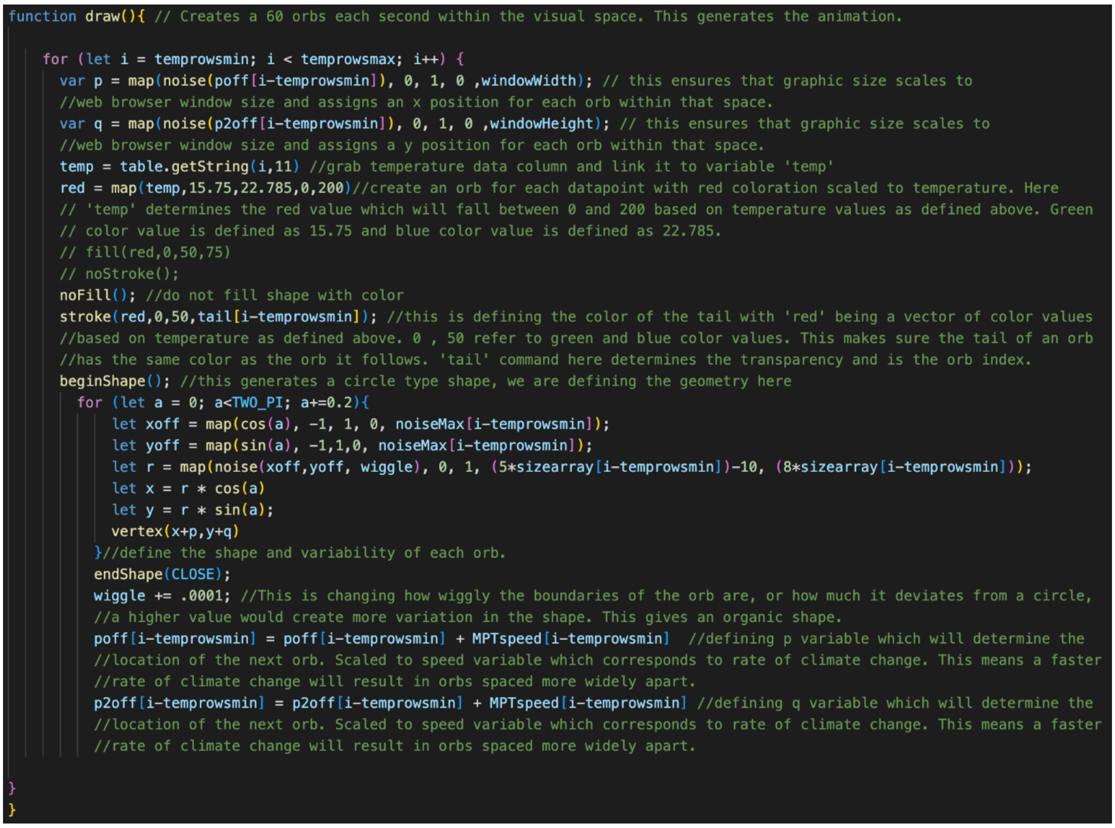

```
function draw(){ // Creates a 60 orbs each second within the visual space. This generates the animation.

    for (let i = temprowsmin; i < temprowsmax; i++) {
      var p = map(noise(poff[i-temprowsmin]), 0, 1, 0 ,windowWidth); // this ensures that graphic size scales to
      //web browser window size and assigns an x position for each orb within that space.
      var q = map(noise(p2off[i-temprowsmin]), 0, 1, 0 ,windowHeight); // this ensures that graphic size scales to
      //web browser window size and assigns a y position for each orb within that space.
      temp = table.getString(i,11) //grab temperature data column and link it to variable 'temp'
      red = map(temp,15.75,22.785,0,200)//create an orb for each datapoint with red coloration scaled to temperature. Here
      // 'temp' determines the red value which will fall between 0 and 200 based on temperature values as defined above. Green
      // color value is defined as 15.75 and blue color value is defined as 22.785.
      // fill(red,0,50,75)
      // noStroke();
      noFill(); //do not fill shape with color
      stroke(red,0,50,tail[i-temprowsmin]); //this is defining the color of the tail with 'red' being a vector of color values
      //based on temperature as defined above. 0 , 50 refer to green and blue color values. This makes sure the tail of an orb
      //has the same color as the orb it follows. 'tail' command here determines the transparency and is the orb index.
      beginShape(); //this generates a circle type shape, we are defining the geometry here
        for (let a = 0; a<TWO_PI; a+=0.2){
            let xoff = map(cos(a), -1, 1, 0, noiseMax[i-temprowsmin]);
            let yoff = map(sin(a), -1,1,0, noiseMax[i-temprowsmin]);
            let r = map(noise(xoff,yoff, wiggle), 0, 1, (5*sizearray[i-temprowsmin])-10, (8*sizearray[i-temprowsmin]));
            let x = r * cos(a)
            let y = r * sin(a);
            vertex(x+p,y+q)
          }//define the shape and variability of each orb.
          endShape(CLOSE);
          wiggle += .0001; //This is changing how wiggly the boundaries of the orb are, or how much it deviates from a circle,
          //a higher value would create more variation in the shape. This gives an organic shape.
          poff[i-temprowsmin] = poff[i-temprowsmin] + MPTspeed[i-temprowsmin]  //defining p variable which will determine the
          //location of the next orb. Scaled to speed variable which corresponds to rate of climate change. This means a faster
          //rate of climate change will result in orbs spaced more widely apart.
          p2off[i-temprowsmin] = p2off[i-temprowsmin] + MPTspeed[i-temprowsmin] //defining q variable which will determine the
          //location of the next orb. Scaled to speed variable which corresponds to rate of climate change. This means a faster
          //rate of climate change will result in orbs spaced more widely apart.

}
}
```