```
/**
 This example builds on the census visualization. Here, we free
 ourselves from the constraints that the visualization needs to be
 immediately understandable.

 Instead, we break out our bouncing ball code. fields from the census
 data are mapped to various properties of our bouncing balls. to create
 a mass on tangled color.

 C. Andrews
 2014-01-21
 **/



Ball[] balls; // our array of Ball objects
void setup() {
  size(700, 700);

  colorMode(HSB, 360, 100, 100);
  background(0, 0, 100);

  // load the table from the file
  Table table;
  table =loadTable("census.csv", "header");

  // create the Ball array using the number of rows in the table
  // to control the number of Balls
  balls = new Ball[table.getRowCount() ];

  int currentBall = 0;
  // iterate over each row, creating a Ball to represent each one
  for (TableRow row: table.rows()) {
    // fetch the data from the row
    String state = row.getString("State");
    int income = row.getInt("income");
    float degree = row.getFloat("% degree");
    int population = row.getInt("Population");

    // create the ball and set its properties with the data
    Ball ball = new Ball();
    ball.diameter = map(population, 570000, 37000000, 5, 20);
    // note that I only took the hue out to 300, to avoid the wrap around back to red
    ball.hue = map(income, 30000, 70000, 0, 300);
    ball.vx =map(degree, 17, 50, 1, 25);
    ball.vy =map(degree, 17, 50, 2, 24);

    // add the ball to the array
    balls[currentBall] = ball;
    currentBall++;
  }
}


/**
 Iterate over the Balls, drawing each one.

 Note that we are using our new for loop syntax.
 **/
void draw() {

  for (Ball ball: balls) {
    ball.update();
    ball.paint();
  }
}


/**
 This is a basic Ball class. It has a size, velocity, position and hue.
 It is placed in a random location on the screen and then just bounces
 when its update() and paint() methods are called. Note that this knows
 nothing about visualization or census data -- it is just a ball.
 **/
class Ball {
  float x, y; // position
  float vx, vy; // velocity
  float diameter; // diameter of the ball
  float hue; // the hue of the ball


  /**
   Start the ball in a random location on the canvas.
   I picked 15 pixels in from the side to make sure no matter what size
   the ball was it would start fully on the canvas.
   */
  Ball() {
    x = random(15, width-15);
    y = random(15, height-15);
  }


  /**
   This is our standard update code.
   We change the balls position based on the velocity. If we hit a wall,
   we reset the position and reverse the velocity along that axis.
   **/
  void update() {
    // update the ball’s position
    x += vx;
    y += vy;


    if (x -diameter/2 < 0) {
      // the ball hit the left wall
      x = diameter/2;
      vx = -vx;
    }
    else if (x + diameter/2 > width) {
      // the ball hit the right wall
      x = width - diameter/2;
      vx =-vx;
    }

    if (y -diameter/2 < 0) {
      // the ball hit the top
      y = diameter/2;
      vy = -vy;
    }
    else if (y + diameter/2 > height) {
      // the ball hit the bottom
      y = height - diameter/2;
      vy =-vy;
    }
  }

  /**
   Paint a simple circle in the chosen hue.
   **/
  void paint() {
    fill(hue, 100, 100);
    ellipse(x, y, diameter, diameter);
  }
}
```