```
//  Austin Matel
// 2/25/20
//  This is a comment
//  The setup function function is called once when your program
begins
//  Make sure to start with the mouse on the canvas, press control r
to start
var ships = [];
var squares = [];
var balls = [];
var repellor,attractor;
function setup() {
  var cnv = createCanvas(800, 800);
  cnv.position((windowWidth-width)/2, 30);
  background(5, 5, 5);
  loadObjects(50);
}

//  The draw function is called @  30 fps
function draw() {
  background(5,5,5,20);
  runObjects();
}
// This puts the ships, boids, and squares into their lists and
variables, and gives them their attributes
function loadObjects(n){
  for (var i = 0; i < n; i++){
    ships[i] = new Ship(random(width), random(height), random(-3,3),
random(-3,3));
    squares[i] = new Square(random(width), random(height), random(-3,
3), random(-3,3));
    balls[i] = new Ball(random(width), random(height), random(-3, 3),
random(-3,3));
  }
  repellor = new Boid(width/2, height/2);
  attractor = new Boid(width/2, height/2);
}
// When called, this displays the ships, boids, and squares and makes
them move around
function runObjects(){
  for(var i = 0; i < ships.length; i++){
    ships[i].run();
    squares[i].run();
```

```
    balls[i].run();
  }
  repellor.run();
  attractor.run();
}

//Austin Matel
//2/25/20
class Ship{
  constructor(x, y, dx, dy){
    this.loc = createVector(x, y);
    this.vel = createVector(dx, dy);
    this.clr = color(0, 0, random(225));
    this.angle = 0;
    this.acc = createVector(0,0.1);
  }
  // Runs all of the ship code
run(){
  this.checkEdges();
  this.update();
  this.render();
  this.makeLine();
  this.touchingMouse();
}
  // Creates the lines between squares and triangles and circles
```

```
makeLine(){
  for(var i = 0; i < squares.length; i++){
    if(this.loc.dist(squares[i].loc) < 75){
      stroke(255,255,255);
      line(this.loc.x, this.loc.y, squares[i].loc.x + 5, squares[i].loc.y + 5);
      noStroke();
    }
    if(this.loc.dist(balls[i].loc) < 75){
      stroke(255,255,255);
      line(this.loc.x, this.loc.y, balls[i].loc.x + 5, balls[i].loc.y + 5);
      noStroke();
    }
  }
}
```

```
  // Makes the ship warp to the other side of the screen when it
contacts the side
```

```
checkEdges(){
  if(this.loc.x < 0){
    this.loc.x = width;
  }
  if(this.loc.x > width){
    this.loc.x = 0;
  }
  if(this.loc.y < 0){
    this.loc.y = height;
  }
  if(this.loc.y > height){
      this.loc.y = 0;
  }
}
  // Makes the ships attracted or repelled from the mouse and makes it
move
  update(){
  var disToAttractor;
  var disToRepellor;
  disToAttractor = this.loc.dist(attractor.loc);
  disToRepellor = this.loc.dist(repellor.loc);
      if(disToAttractor > 200){
    this.acc = p5.Vector.sub(attractor.loc, this.loc);
    this.acc.normalize();
    this.acc.mult(0.1);
  }
  if(disToRepellor < 200){
    this.acc = p5.Vector.sub(repellor.loc, this.loc);
    this.acc.normalize();
    this.acc.mult(-0.5);
    }
  this.vel.limit(4);
  this.vel.add(this.acc);
  this.loc.add(this.vel);
  }
    // This draws the ships and orients the ship to point towards the
mouse
render(){
  fill(this.clr);
  this.angle = this.vel.heading() + 360;
  this.angle = this.angle +0.1;
  push();
    translate(this.loc.x,this.loc.y);
```

```
    rotate(this.angle);
    triangle(-5,8,5,8,0,-8);
  pop();
  }
  touchingMouse(){
    if(this.loc.x === mouseX && this.loc.y === mouseY){
            background(random(255), random(255), random(255),
random(255));
    }
  }
}

//Austin Matel
//2/25/20
class Boid{
  constructor(x, y){
    this.loc = createVector(x, y);
  }
  // This runs all of the Boid code
  run(){
    this.checkEdges();
    this.followMouse();
  }
  // This makes sure that the boid doesn't follow the mouse all over
the screen
  checkEdges(){
    if (this.loc.x < 0){
      this.loc.x = 0;
    }
    if (this.loc.x > width){
      this.loc.x = 800;
    }
    if (this.loc.y < 0){
      this.loc.y = 0;
    }
    if (this.loc.y > height){
        this.loc.y = 800;
    }
  }
  // This makes the boid follow the mouse
  followMouse(){
    this.loc.x = mouseX;
    this.loc.y = mouseY;
```

```
  }
}

//Austin Matel
//2/25/20
```

```
class Square{
  constructor(x, y, dx, dy){
    this.loc = createVector(x, y);
    this.vel = createVector(dx, dy);
    this.clr = color(random(225), 0, 0);
    this.angle = 0;
    this.acc = createVector(0,0.1);
  }
// This runs all of the square code
  run(){
    this.checkEdges();
    this.update();
    this.render();
    this.makeLine();
  }
```

```
    // Creates the lines between squares and triangles and circles
  makeLine(){
    for(var i = 0; i < squares.length; i++){
      if(this.loc.dist(balls[i].loc) < 75){
        stroke(255,255,255);
        line(this.loc.x, this.loc.y, balls[i].loc.x + 5,
balls[i].loc.y + 5);
        noStroke();
      }
    }
  }
  // This makse the squares warp to the other side of the screen when
they contact the side of the canvas
  checkEdges(){
    if(this.loc.x < 0){
      this.loc.x = width;
    }
    if(this.loc.x > width){
      this.loc.x = 0;
    }
    if(this.loc.y < 0){
      this.loc.y = height;
    }
    if(this.loc.y > height){
```

```
      this.loc.y = 0;
    }
  }
  // This attracts and repels the squares to the mouse and give it its
speed
  update(){
    var disToAttractor;
    var disToRepellor;
    disToAttractor = this.loc.dist(attractor.loc);
    disToRepellor = this.loc.dist(repellor.loc);
    if(disToRepellor > 200){
      this.acc = p5.Vector.sub(repellor.loc, this.loc);
      this.acc.normalize();
      this.acc.mult(0.1);
      }
    if(disToAttractor < 200){
      this.acc = p5.Vector.sub(attractor.loc, this.loc);
      this.acc.normalize();
      this.acc.mult(-0.5);
      }
    this.vel.limit(4);
    this.vel.add(this.acc);
    this.loc.add(this.vel);
    }
  // This displays the squares and orients them to face a flat side
towards the mouse
  render(){
    fill(this.clr);
    this.angle = this.vel.heading() + 360;
    this.angle = this.angle +0.1;
    push();
      translate(this.loc.x,this.loc.y);
      rotate(this.angle);
      rect(-5,8,10,10);
    pop();
    }
}

//Austin Matel
//2/25/20
class Ball{
  constructor(x, y, dx, dy){
    this.loc = createVector(x, y);
```

```
    this.vel = createVector(dx, dy);
    this.clr = color(0, random(255), 0);
    this.angle = 0;
    this.acc = createVector(0,0.1);
  }
  // This runs all of the square code
  run(){
    this.checkEdges();
    this.update();
    this.render();
  }
  // This makes the squares warp to the other side of the screen when
they contact the side of the canvas
  checkEdges(){
    if(this.loc.x < 0){
      this.loc.x = width;
    }
    if(this.loc.x > width){
      this.loc.x = 0;
    }
    if(this.loc.y < 0){
      this.loc.y = height;
    }
    if(this.loc.y > height){
      this.loc.y = 0;
    }
  }
  // This attracts and repels the squares to the mouse and give it its
speed
  update(){
    var disToAttractor;
    var disToRepellor;
    disToAttractor = this.loc.dist(attractor.loc);
    disToRepellor = this.loc.dist(repellor.loc);
    if(disToRepellor > 200){
      this.acc = p5.Vector.sub(repellor.loc, this.loc);
      this.acc.normalize();
      this.acc.mult(0.1);
      }
    if(disToAttractor < 200){
      this.acc = p5.Vector.sub(attractor.loc, this.loc);
      this.acc.normalize();
      this.acc.mult(-0.5);
```

```
      }
    this.vel.limit(4);
    this.vel.add(this.acc);
    this.loc.add(this.vel);
    }
  // This displays the squares and orients them to face a flat side
towards the mouse
  render(){
    fill(this.clr);
    ellipse(this.loc.x, this.loc.y, 10, 10);
    }
  }
```