



Girls Who Code At Home

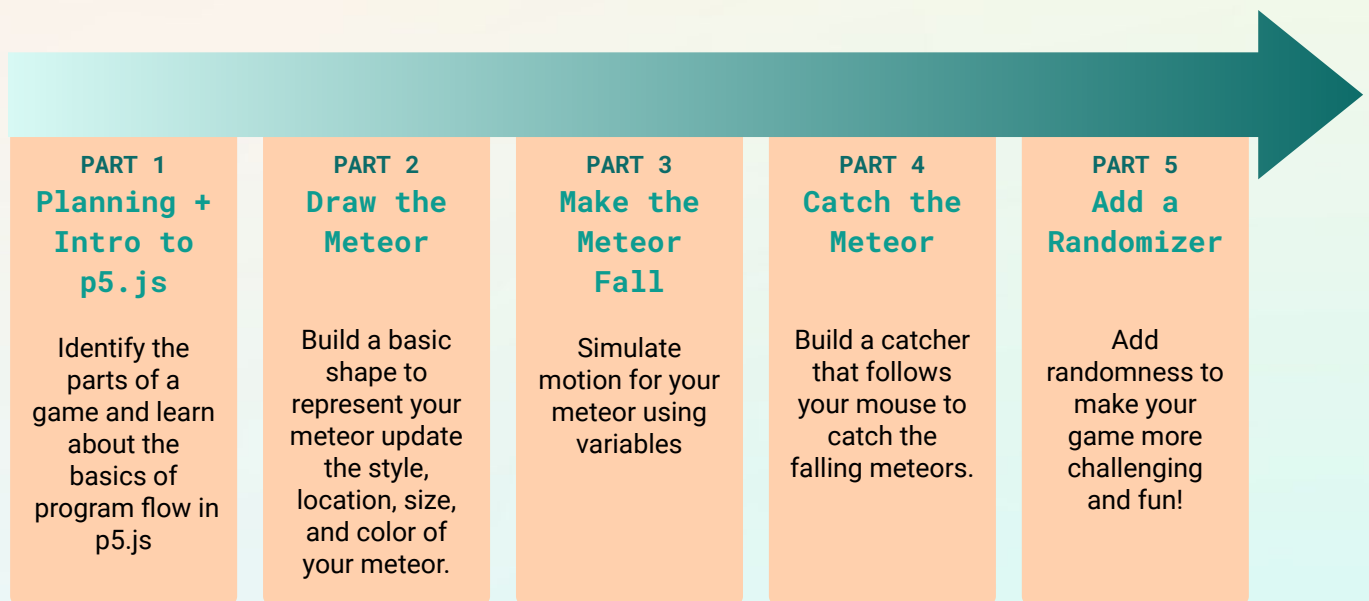
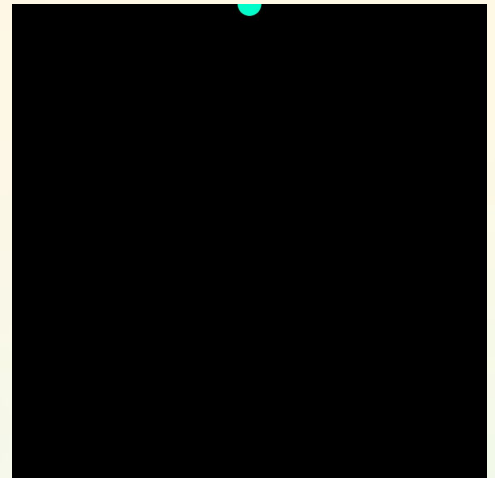
Meteor Catcher Game: Part 2

Draw the Meteor

Activity Overview

Welcome back! Last week you explored the p5.js environment and began planning the parts of your meteor catcher game. In this part, you will use the coordinate system to draw the first component of your game: the meteor! You will also learn more about how to use color in p5.js and set the color for your meteor and sketch background.

You should have already completed [Part 1](#) of the **Meteor Catcher Game Series** before embarking on this activity.



Learning Goals

By the end of this activity you will be able to...

- ❑ describe the p5.js coordinate system and its relationship to pixels on the screen.
- ❑ use built in functions and commands to draw basic shapes on the coordinate plane.

Materials

- ➔ [p5.js Online Editor](#)
- ➔ [Meteor Catcher Game Sample Project](#)
- ➔ [Meteor Catcher Game Part 2 Reference Guide](#)

You can also follow along with Part 2 in the [Meteor Catcher Game video tutorials!](#)

Women in Tech Spotlight: Phoenix Perry



Image Source: [Hackaday.io](https://hackaday.io)

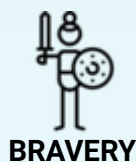
When Phoenix was younger, her parents bought her an [Atari](#) home game console. Being introduced to this new world of technology sparked Phoenix's interest in technology and she began learning how to program in [BASIC](#). After graduating with an undergraduate degree, Phoenix began her career as a Web Designer for [Evite](#). Here she endured many endless nights, a normal behaviour in the company, and took pain killers from the strains on her wrists due to lack of rest. She then developed [carpal tunnel](#), a common nerve syndrome common in programmers due to increased tensions in the bones and ligaments on your hand.

Phoenix spent many years away from the technology industry due to her condition, working as an Art Director. She then came across the [Integrated Digital Media program at NYU Tandon](#), where she became an Adjunct Professor and Researcher to teach about the design, play, and embodiment of game development. It was through this experience she went on to become a co-founder of [Code Liberation](#). Code Liberation hopes to teach, prepare, and support women, nonbinary, femme, and girl-identifying people to pursue jobs in STEAM. They offer free classes, workshops, game jams, hackathon, and social game nights to women of all ages and at different stages of their careers. Phoenix uses her experience to help the next generation of women develop skills needed to join the technology industry. She has now partnered with [University of Arts London](#) to open a chapter of Code Liberation in the United Kingdom.

Watch this [video](#) to learn more about Phoenix and why diversity is important in technology. Want to learn more about Phoenix? Watch her [TED Talk](#), explore her [personal website](#), and read this [article](#) to learn more about Phoenix's academic journey and hardships to get to where she is today!

Reflect

Being a computer scientist is more than just being great at coding. Take some time to reflect on how Phoenix and her work relates to the strengths that great computer scientists focus on building - bravery, resilience, creativity, and purpose.



After developing carpal tunnel, Phoenix found herself unable to work and move. This made it difficult for her return to the technology industry. Discuss how Phoenix displayed bravery in returning to industry while also supporting and preparing other women at Code Liberation.

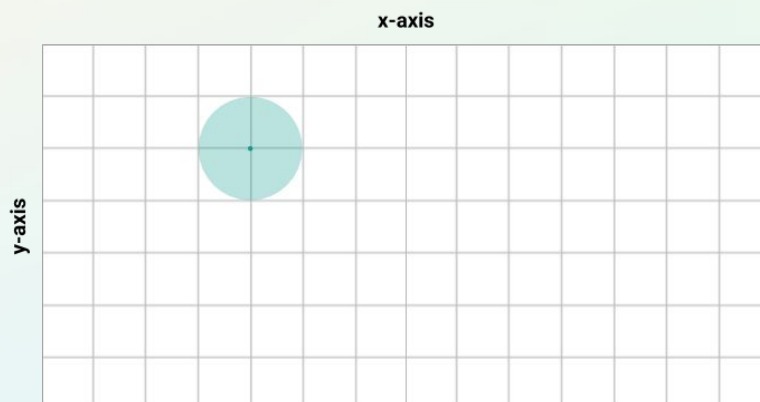
Share your responses with a family member or friend. Encourage others to read more about Phoenix to join in the discussion!

Step 1: Command a circle (2-4 mins)

Let's say your friend gives you an instruction to draw a circle on a piece of paper. You could either just draw the circle or ask for more information. If you ask for more information, you might inquire - where on the paper? How big? What color? A perfect circle or more of an oval? To answer the question of where, your friend might say "A third of the way from the left side and three-fourths of the way down towards the bottom." That kind of instruction might work if you are talking to a human and don't need to be specific. But it won't work for a computer! Instead we can use the coordinate system to specify a location for elements we want to display in our program.

The coordinate system is a system that uses one or more numbers to identify the location of a point in space. Coordinate systems can be on a 2D plane or in 3D space. Coordinate planes (i.e. 2D) have an x-axis that runs horizontally and a y-axis that runs vertically to form a grid. They use an ordered pair to signify a point: (x position, y position).

p5.js can also work in three dimensional space with a z-axis. You may see the option for a z-axis parameter in p5 documentation on some functions, but you should not include it.



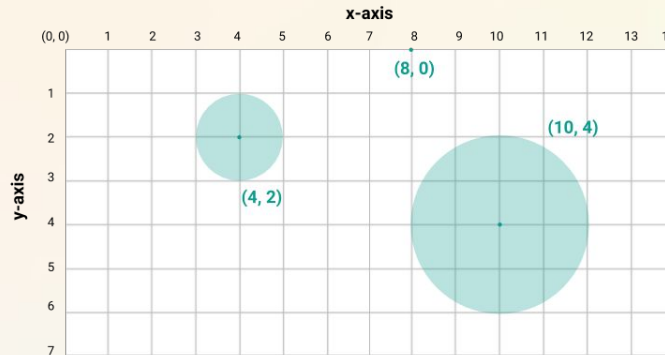
Consider the coordinate plane above. Think about the instructions you would give to a computer to draw the circle in that location.



Don't forget to check your ideas with the Reference Guide on pg 2.

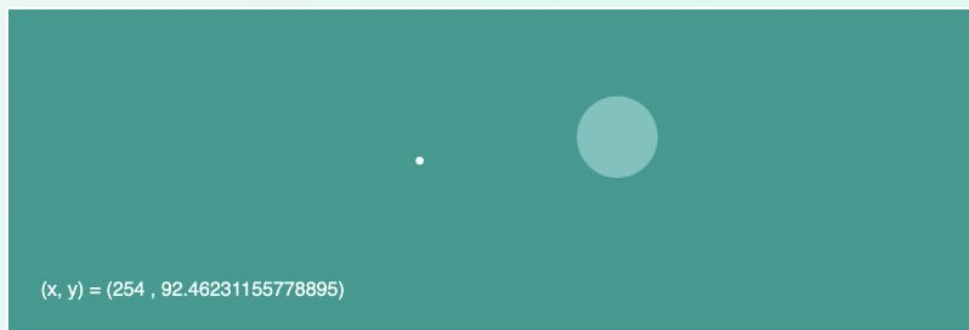
Step 2: Practice using the coordinate system (5-8 mins)

Each pixel on your screen has a unique address in the coordinate system. In order to draw pixels to the screen, we must give our program the x coordinate (i.e. the location on the x-axis) and the y coordinate (i.e. the location on the y-axis) of the pixel.



The origin or (0, 0), on the screen coordinate system is located at the top left corner. As you move right on the screen, the value of the x-coordinate increases. As you move down on the screen the value of the y-coordinate increases. This may appear a little different than the coordinate system layouts you have seen in math class, where the origin is in the center or at the bottom left corner.

Explore this [sketch](#) and use your mouse around to try to estimate the center, width, and height of the circle.

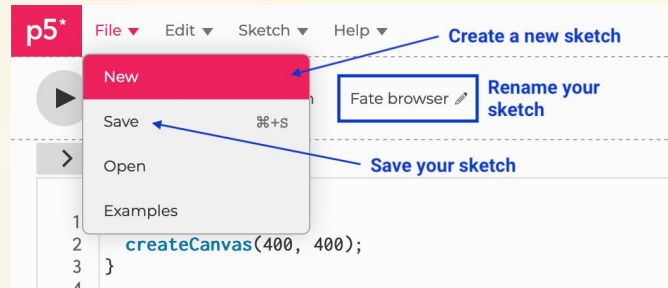


Don't forget to check your ideas with the Reference Guide on pg 2.

Step 3: Create your project sketch (5-10 mins)

In the remaining sections, we will start writing the code for your game. First we need to create your main project sketch.

- ❑ **Log into the p5.js online editor.** The editor automatically gives you a blank sketch with starter code. *Alternatively, you can create a new sketch by going to File > New.*
- ❑ **Click the pencil icon to name the sketch** to something that you can easily recognize like Meteor Catcher Game v1. *Note: This is in the sketch information area below the toolbar.*
- ❑ **Next, go to File and click Save.** You can also save by using the keyboard shortcut Command S (Mac) or Control S (Windows). Be sure to navigate inside the editor before using these shortcuts.
- ❑ **Create a multiline comment at the very top of your sketch with the following information:**
 - ❑ **Title of program:** This should be the same as the sketch name.
 - ❑ **Version of program:** Is this the first version or second? If you make big changes, it's good practice to create a new version.
 - ❑ **Author:** By (your first name and last name initial).
 - ❑ **Description:** A sentence or two about what it does.



At the top of your sketch, you will include a comment that gives basic information about the sketch. Use **code comments** to remind yourself of how something works, the reasoning for a decision, or a follow up task.

- ➔ Single line comments use a double forward slash, `//`.
- ➔ Multiline comments use a forward slash and asterisk, `/*`, to open it and an asterisk and forward slash, `*/`, to close it.

```
// This is a single line comment

/*
This is a
multiline comment
*/
```

Step 4: Draw the meteor (3-5 mins)

Now that we have our project sketch saved, let's create our first game component: the meteor! First we will learn how to draw the shape at a specific location, then we will fill it with color to change the appearance.

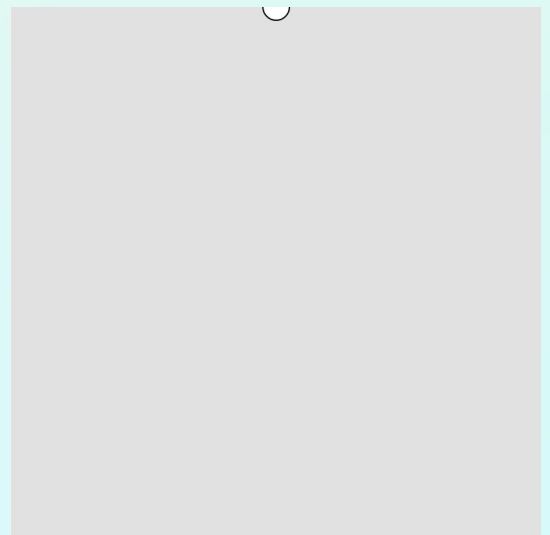
Shape and Location

Think back to the last step where we learned about the coordinate system. Based on what we know now, if we want to draw a shape, we have to tell the computer to draw each pixel individually. This gets pretty tedious very quickly, so p5 created pre-made functions that draws shapes for us. All we need to do is define the location where we want the shape, then give the width and height. Let's examine the syntax for a circle below:

JAVASCRIPT	DESCRIPTION
<code>ellipse(x, y, width, height);</code>	<ul style="list-style-type: none">→ ellipse: The function name. Ellipse is another word for oval. p5.js Reference→ (): We use parentheses to tell our program that it needs to call the function. Sometimes we include parameters or inputs in the function inside our parentheses.→ x: The x-coordinate at the center of the ellipse.→ y: The y-coordinate at the center of the ellipse.→ width: Sets the width of the ellipse in pixels.→ height: Sets the height of the ellipse in pixels.→ ,: We use commas to separate the different parameters or inputs in the functions.→ ;: All lines of code in p5.js must end with a semicolon.

Use the `ellipse()` function to draw the meteor to the screen:

- ❑ Add the `ellipse()` function to your sketch inside of the `draw()` function.
- ❑ Place it in the center of the canvas (your x-axis position) at the very top (your y-axis position). Reference the graph above if you need a refresher.
- ❑ Set the size of the circle, or ellipse, to be 20 pixels wide and 20 pixels high.
- ❑ Add a comment to remind yourself that this is the meteor.
- ❑ Run your code by pressing the play button to test it.



Don't forget to check your code with the Reference Guide on pg 3.

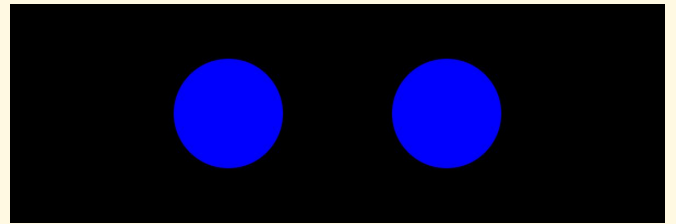
Step 5: Fill the meteor with color (5-8 mins)

To add color to shapes, we can use the `fill()` function. This will fill any shape drawn after the command with the specified color. In the example below, both circles are blue because they come after the `fill()` function:

JAVASCRIPT

```
function setup() {  
  createCanvas(600, 200);  
}  
  
function draw() {  
  background(0);  
  
  // Make both circles blue  
  fill(0, 0, 255);  
  ellipse(width / 3, height / 2, 100, 100);  
  ellipse(width / 3*2, height / 2, 100, 100);  
}
```

RESULT



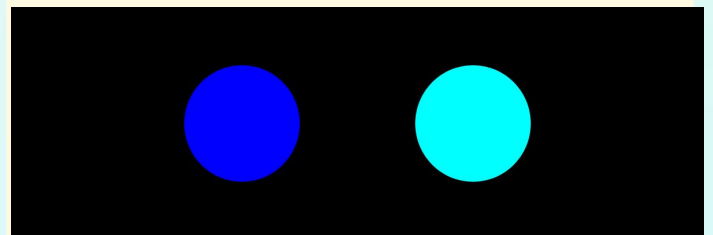
But what if we want the second shape to be teal instead of blue? We need to add another `fill()` function before that shape:

JAVASCRIPT

```
function setup() {  
  createCanvas(600, 200);  
}  
  
function draw() {  
  background(0);  
  
  // Blue circle  
  fill(0, 0, 255);  
  ellipse(width / 3, height / 2, 100, 100);  
  // Teal circle  
  fill(0, 255, 255);  
  ellipse(width / 3*2, height / 2, 100, 100);  
}
```

RESULT

This change would result in one blue circle and one teal circle



The `fill()` method uses RGB color mode that uses a combination of red, green, and blue light to create a spectrum of colors.

- (0,0,255) would display a **blue** color
- (0,255,255) would give a **teal** color.

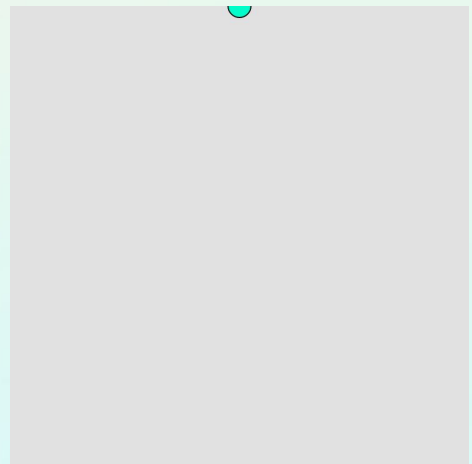
Step 5: Fill the meteor with color (cont.)

Check out the syntax for the `fill()` function using RGB color mode below. RGB color mode uses combinations of red, green, and blue light to create a range of digital colors. You can assign a value to each color between 0 to 255. For example, (255, 0, 0) would be red, (0, 0, 0) would be black, and (255, 136, 0) would be orange. You can play around with different values and colors using tools like [color pickers](#) or palette tools like [Coolers](#).

JAVASCRIPT	DESCRIPTION
<code>fill(redValue, greenValue, blueValue);</code>	<ul style="list-style-type: none">→ <code>fill</code>: The function name.→ <code>()</code>: We use parentheses to tell our program that it needs to call the function. Sometimes we include parameters or inputs in the function inside our parentheses.→ <code>redValue</code>: The red value between 0 and 255.→ <code>blueValue</code>: The blue value between 0 and 255.→ <code>greenValue</code>: The green value between 0 and 255.→ <code>;</code>: All lines of code in p5.js must end with a semicolon.

Use the `fill()` function to add color to your meteor:

- ☐ Pick a color for your meteor and make a note of the RGB values.
- ☐ Call the `fill()` function with your meteor's RGB values (remember that location is important!).
- ☐ Press the play button to run your program when you are finished. The color of your meteor should change to the one you specified.



Don't forget to check your code with the Reference Guide on pg 3.

Step 6: Change the outline (3-5 mins)

You might have noticed that there is a black outline around your meteor. We can remove this outline using the `noStroke()` function. Calling this function means that none of the shapes in your sketch will have outlines. If you want to enable outlines, you need to call the `stroke()` function above that shape.

You can use other functions like `stroke()` and `strokeWeight()` to control the color and thickness of shape borders.

Remove the outline using the `noStroke()` function:

- ❑ Add the `noStroke()` function under the `background()` function. Since we don't want outlines on any of our shapes, we will place it at the top.
- ❑ Run the program when you are done.

An outline will no longer be visible on your meteor. Here is a close up screenshot of how the meteor should look at the top of your sketch.



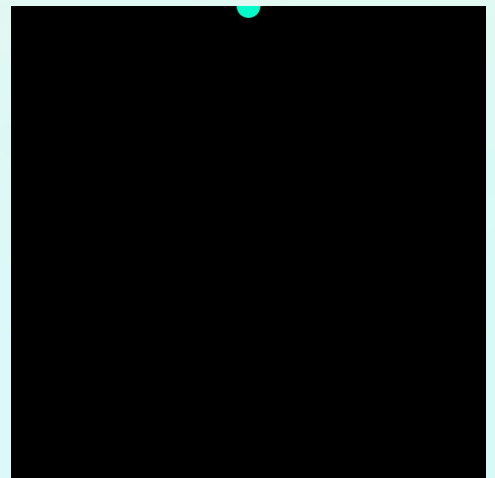
Don't forget to check your code with the Reference Guide on pg 4.

Step 7: Add the background color (3-5 mins)

Put your color knowledge to the test! Fill in the canvas using the `background()` function. Right now, it only has one value, but similarly to the `fill()` function, it can take parameters for other color modes like RGB.

Change the background of your game:

- ❑ Choose a background color and make note of its RGB values. (Here are the [color pickers](#) and palette tools like [Coolers](#) mentioned earlier.)
- ❑ Update the `background()` function inside `draw()` with your new color values.

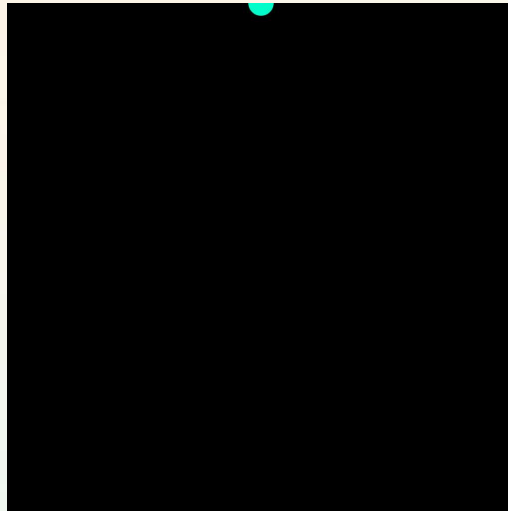


Don't forget to check your code with the Reference Guide on pg 4.

Step 8: Test your code (3-5 mins)

Let's test what we have written so far to make sure our program runs the way we want it to. Click the play button to run your sketch. You should have:

- A 400 by 400 canvas with a color background.
- A circle meteor in the top center of the canvas with a new color and no border.



Not working the way you want it to? If you have an error that prevents the code from compiling and running, p5.js will display an error message in the console. When something isn't working properly, start there to figure out the problem.

Otherwise, try these **debugging tips**:

- Is your code inside the correct curly brackets?
- Do you have semicolons at the end of each line of code?
- Did you spell the variable and function names correctly? Remember that JavaScript is also case-sensitive!
- Are your functions in the correct location and sequence? Remember that order matters in program flow!
- Are your parameter values within the correct range for the function? For example, is there an x value of 500 even though the canvas is 400 pixels wide? Are your RGB values between 0 and 255?

If you need a refresher on best practices for debugging, check out [this fantastic post](#) from the p5.js community.

Step 9: Check for Understanding

Describe the location, size, and color of the shape in the code below:

```
function setup() {  
  createCanvas(100, 100);  
}  
  
function draw() {  
  fill(0, 0, 255);  
  ellipse(50, 50, 5, 5);  
}
```



Don't forget to check your ideas with the Reference Guide on pg 5.

Step 10: Share Your Girls Who Code at Home Project! (5 mins)

We would love to see your work and we know others would as well. Share your pseudocode with us! Don't forget to tag [@girlswhocode](#) [#codefromhome](#) and we might even feature you on our account!

Stay tuned for more Girls Who Code at Home projects!

