



# Girls Who Code At Home

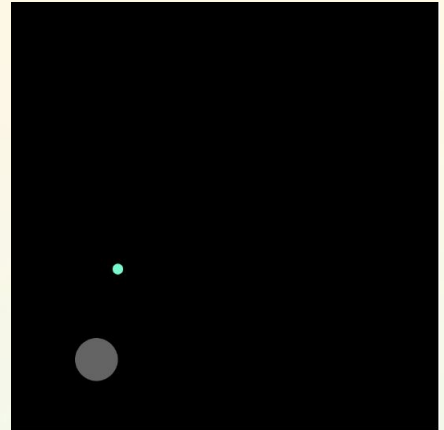
**Meteor Catcher Game: Part 5**

Add a Randomizer

## Activity Overview

In this final part, we will explore one way to make your game more challenging - and more fun! You will learn how to use the [random\(\)](#) function to add complexity to the behavior of your meteor. Finally, customize your project by trying out one or more of our extensions. Click [here](#) to preview what you will learn by the end of the activity.

You should have already completed [Part 1](#), [Part 2](#), [Part 3](#), and [Part 4](#) of the **Meteor Catcher Game Series** before embarking on this activity.



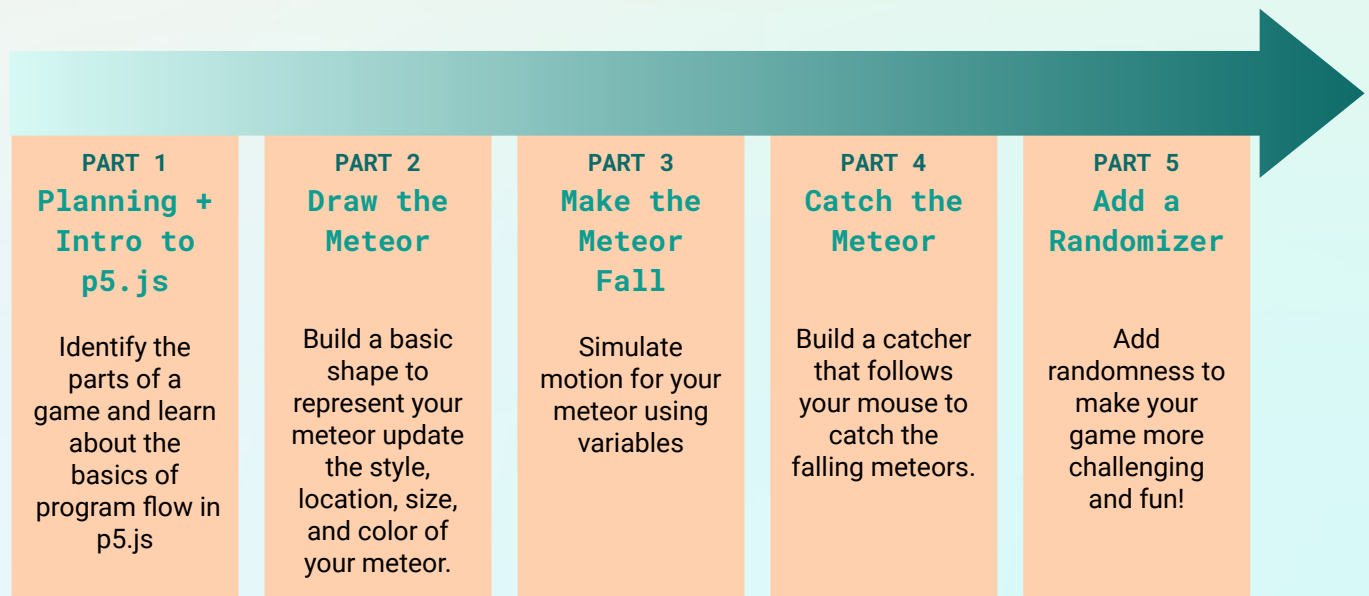
## Learning Goals

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

- explain how the `random()` function can be used to increase or decrease a game's challenge.

## Materials

- [p5.js Online Editor](#)
- [Meteor Catcher Game Sample Project](#)
- [Meteor Catcher Game Part 5 Reference Guide](#)



You can also follow along with Part 6 in the **Meteor Catcher Game [video tutorials!](#)**

## Women in Tech Spotlight: Lisette Titre-Montgomery



Image Source: [Lisette Titre-Montgomery](#)

When you jam out to moves on Dance Central or create characters on The Sims, do you ever wonder about the technology that took you there? Lisette Titre-Montgomery works as an engineer and designer behind the most popular games on the market. With over seventeen years of experience in the game industry and thirteen shipped titles, she has successfully bridged the gap between the video gamer and the developer.

Lisette also dedicates her time to bringing more diversity into the gaming industry through mentoring and inclusive hiring initiatives with The White House. She continues to be an advocate for game based curriculums in K-12 education to engage students in STEAM education and careers with a program called [Gameheads](#), based in Oakland, CA.

Watch this [video](#) to learn more about Lisette and some of the challenges she faced working on various games and how she overcame them.

Want to learn more about Lisette and her work?

- Check out her [personal website](#) to learn more about some of her various works and projects.
- Read this [article](#) about Lisette's background and how she came to the gaming industry.
- Look up [Lisette's profile](#) to see some highlights of her career and her advice for students interested in starting in the gaming industry!

## Reflect

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



**BRAVERY**

Lisette had never worked in the gaming industry until her first job. What steps did she take to succeed in her role?

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

## Step 1: Reflect on your game's challenge (2-4 mins)

Right now, we have all the critical parts of our game covered: the components, core mechanic, game and space. But the game isn't very fun yet, mostly because it is not challenging enough. Finding the right difficulty level is key for game designers. One tool you can use to increase difficulty (and there are many) is to include an element of randomness. This prevents the player from anticipating what will come next by surprising them with new information that they must adapt to.

Where can we add randomness to the game to make it more challenging? Take one minute to think about it, then compare your ideas to the ones below.



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

## Step 2: Explore the `random()` function (5 mins)

We can use the `random()` function to add randomness to our code. This function returns, or outputs, a random floating point number between a specified range of values. Floating point means the number might contain a decimal. This allows for a much larger spectrum of numbers. Explore this [example sketch](#) where each mouse click inside the canvas generates a random value for the red, green, and blue values in the `fill()` function.



Let's take a look at the syntax of the `random()` function.

JAVASCRIPT	DESCRIPTION
<code>random(min, max);</code>	<ul style="list-style-type: none"><li>→ <code>random</code>: The function name.</li><li>→ <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.</li><li>→ <code>min</code>: The lower end of the random range you want that includes this number as an option.</li><li>→ <code>max</code>: The higher end of the random range you want that excludes this number as an option.</li><li>→ <code>;</code>: All lines of code in JavaScript must end with a semicolon.</li></ul>

## Step 3: Update your pseudocode (2-4 mins)

Before we add in our new lines of code, let's brainstorm what we want to happen in human language first. Write pseudocode for each `random()` function. Try to be as specific as possible and use the value ranges specified. When you are done, check your answer below:

- **Meteor starting location.** Value range: 0 to 400
- **Meteor speed.** Value range: 0.5 to 4
- **Meteor size.** Value range: 10 to 30



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

## Step 4: Add your code (5-8 mins)

Now we need to translate your pseudocode into actual code. All of the code we write will go inside both conditional statements underneath the existing statement. Right now, we are only redrawing the meteor to the top of the screen if it intersects with the catcher or bottom wall. Adding these statements in the conditional will randomize the location, speed, and size of the new meteor.

**Locate the first conditional statement. Add these new lines of code under `meteorY = 0`;**

- ☐ Write the line of code that randomizes the meteor starting location to a value between 0 and 400.
- ☐ Write the line of code that randomizes the meteor's speed to a value between 0.5 and 4.
- ☐ Write the line of code that randomizes the meteor's size to a value between 10 and 30.

**Locate the second conditional statement.**

- ☐ Copy the statements you just wrote for the first conditional, and copy them under `meteorY = 0`; in the second conditional.

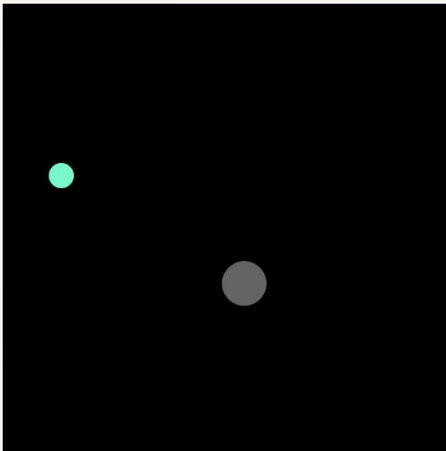
You may have noticed the lines of code inside the conditionals look the same! That's true! You could combine these conditionals with the **logical operator** "or" - `||`. You could also combine them by nesting them together. We opted to separate them to make the extensions a bit less confusing. If you would like to combine these conditionals, you are welcome to!

## Step 5: Test your code (5-10 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.

### Run your code and test the following:

- ❑ Try to make your meteor and catcher intersect. When they intersect, the current meteor should disappear and a "new" meteor should appear at a random x position, with a new size, and new speed.
- ❑ Wait for the meteor to intersect with the bottom of your canvas. When they intersect, the current meteor should disappear and a "new" meteor should appear at a random x position, with a new size, and new speed.



Click [here](#) to run the example sketch.

**Not working the way you want it to?** Try these debugging tips:

- Is your code inside the correct curly brackets?
- Do you have semicolons at the end of each line of code?
- Are your functions in the correct location? Remember that order matters in program flow!
- Are the parameters in your `random()` function correct?



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

## Step 7: Test your code and receive feedback (5-10 mins)

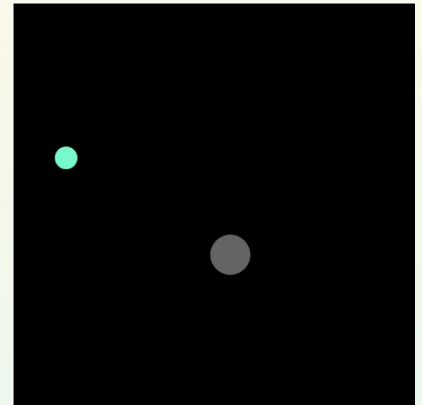


You have unlocked a major milestone: you finished building the core part of your project! Before moving forward, we will test your code to make sure your game is working properly. This is also a great opportunity to get feedback on your game from friends and family. Share your project and this checklist with your tester!

### Checklist

**Run your sketch, then use the checklist below to test your code:**

- ☐ You have a 400 by 400 canvas. The background is filled with the color of your choice.
- ☐ Your game begins with a meteor (i.e. an ellipse) falling slowly from the center of the top of the canvas. The meteor has no outline and is filled with a color of your choice.
- ☐ There is a catcher (i.e. an ellipse) that is white in color and semi-transparent. The center of the catcher follows the movement of your mouse.
- ☐ Your program prints the distance between the meteor and catcher to the console.
- ☐ If the meteor and the catcher intersect, the game updates to draw a new meteor.
- ☐ If the meteor and the bottom of the canvas intersect, the game updates to draw a new meteor.
- ☐ New meteors begin at the top of the canvas at a random location on the x axis with a random speed and random size.



Click [here](#) to run the example sketch.

### Debugging Tips

If your sketch isn't working the way you want it to, check the console first to see if there is an error. You can try to fix your code or try googling to find the answer. You can also review [this resource](#) from p5.js.

- 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?
- Are your functions in the correct location and sequence? Remember that order matters in program flow!
- Are the parameter values in your functions correct? Specifically the `dist()` and `random()` functions?
- Print values to the console using `print()` then check any `if` statements.

**You did it!** You finished building the foundation for your project. In the next part of this activity you will have the chance to personalize your project further with some optional extensions.



## Step 8: Extensions (5-45 mins)

### Extension 1: Change the story with graphics (15-45 mins)

Right now, the game is about space and meteors - but it doesn't have to be! You can alter the narrative of the game by adding new graphics or images. In game design, this is referred to as skinning. The skin of a game is the appearance of the components and background of the game. You can reskin the game without changing the mechanics, but it can vastly change the meaning of the game and how the player feels about it.



Click [here](#) to see an example sketch of this extension

In this extension, try replacing one or more shapes and the background with images. Here are the basic steps you need to complete this extension:

- ❑ Find **.png** image files with a transparent background. You may need to resize them in Google Draw, then export them as a **.png** image file.
- ❑ Upload your **.png** files to your sketch.
- ❑ Declare a variable to store the image file.
- ❑ Initialize each image variable in **setup()** with the **loadImage()** function.
- ❑ Draw the images to the screen using the **image()** function in **draw()**. Be sure to use the correct x and y position variables.
- ❑ Pass the background image variable through the **background()** function.

#### Extension Resources

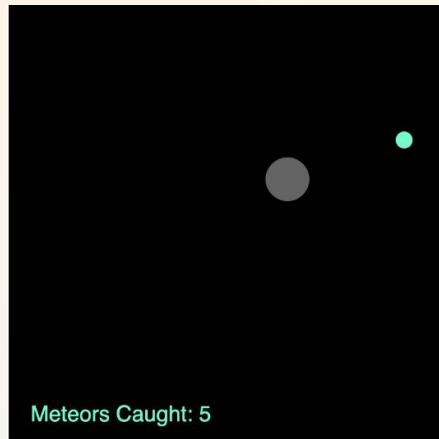
Below are some helpful resources we used to create this Extension. These will help you get started, but remember that there are lots more resources only a search engine away!

- [p5.js Web Editor: Uploading Media Files - p5.js Tutorial](#) by The Coding Train (start around 0:59)
- [Load and Display an Image Example](#) from p5.js. Note: Disregard the text that says To run this example locally, you will need an image file, and a running local server. This does not apply to the online editor.
- [loadImage\(\)](#)
- [image\(\)](#)
- [imageMode\(\)](#)
- [background\(\)](#)

Here is a link to our [solution code for this extension](#). We recommend trying it yourself first and using this resource when you get really stuck or want to check your work.

### Extension 2: Track the score (5-15 mins)

At some point in the tutorial, you may have asked yourself, “But what about points?!” Keeping track of a player’s score isn’t necessary to make it a game, but it is one way to give players immediate feedback about their gameplay. You can create a variable to keep track of the score, then add or increment the value of that variable each time the player earns points. You could also have the player start with a score, then subtract or decrement a value when they take a specific action.



Click [here](#) to see an example sketch of this extension

In this extension, try creating your own simple scoring system, then drawing the score to the screen as text. Here are the basic steps you need to complete this extension:

- ❑ **Create a variable to keep track of the score and set it to zero.** *Hint: Where do you think you'd want to place this variable if you wanted to access it anywhere in your sketch?*
- ❑ **Increment this variable if the meteor and catcher intersect.** *Hint: Where in your code do you check to see if the meteor and catcher intersect?*
- ❑ **Draw the value of the variable to the screen as text.**

#### Extension Resources

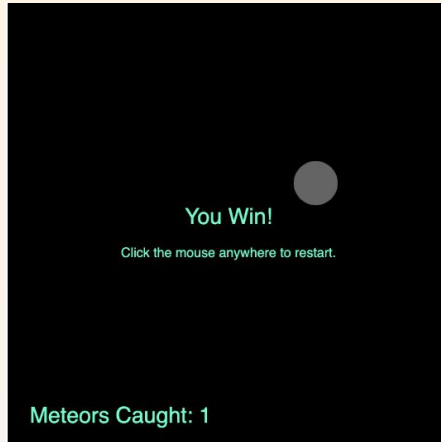
Below are some helpful resources we used to create this Extension. These will help you get started, but remember that there are lots more resources only a search engine away!

- [Increment Decrement Example](#) from p5.js
- [Words Example](#) from p5.js
- [textSize\(\)](#)
- [text\(\)](#): You can combine [strings](#) (i.e. a series of text characters) with variables in the `text()` function.
- [textAlign\(\)](#)
- [string](#)

Here is a link to our [solution code for this extension](#). We recommend trying it yourself first and using this resource when you get really stuck or want to check your work.

### Extension 3: Add a win state (10-20 mins)

Whether individual or collaborative, games often have a win state. Win states occur when players have successfully overcome the main challenge of the game, like the highest number of goals or most gold coins in 30 seconds or the completion a space mission. There are many different ways you can win a game.



Click [here](#) to see an example sketch of this extension

In this extension, create a system to test and alert a player if they have won, then reset the game so they can play again. Before you get started, think about what “winning” means in your game. *For example, is it the amount of times played, the number of points, etc.* In the extension example [here](#), we use the number of points scored to determine if a player has won. If they have won and press the mouse, the game restarts. This extension requires you to create a function of your very own, so if you want to learn more about defining functions this is a good opportunity!

Here are the basic steps you need to complete this extension:

- ❑ Create a function to restart the game by resetting the position, score, and speed variables to their original values.
- ❑ Create a conditional to test if the score is equal to the number of points needed to win.
- ❑ Write a message to the screen alerting the player they have won and how to restart the game.
- ❑ Create a second conditional inside the first to test if the mouse is pressed. Call the restart function inside of it.

#### Extension Resources

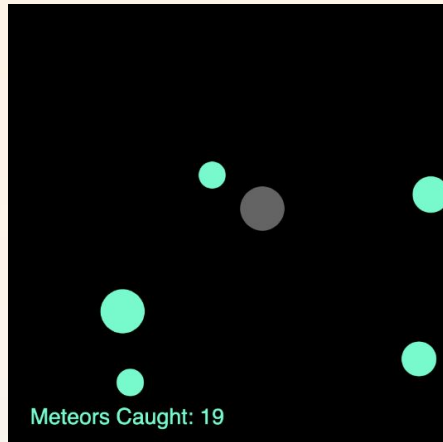
Below are some helpful resources we used to create this Extension. These will help you get started, but remember that there are lots more resources only a search engine away!

- [Function Basics - p5.js Tutorial](#) video from The Coding Train
- [function](#)
- [mouselsPressed](#)
- [Increment Decrement Example](#) from p5.js
- [Words Example](#) from p5.js
- [textSize\(\)](#)
- [text\(\)](#): You can combine [strings](#) (i.e. a series of text characters) with variables in the `text()` function.
- [textAlign\(\)](#)
- [string](#)

Here is a link to our [solution code for this extension](#). We recommend trying it yourself first and using this resource when you get really stuck or want to check your work.

#### Extension 4: Add more meteors (25-40 mins)

In the last part of the Building phase, we explored ways we could make the game more challenging by randomizing properties of the meteor. Another way to make the game more challenging is to make more meteors!



Click [here](#) to see an example sketch of this extension

**Note:** We also included a score tracker so you can see when a meteor has intersected, but you are not required to do that for this extension.

There are a couple of ways you can approach this.

- **Approach #1:** Create new variables for a second meteor. Then duplicate all the code you wrote for the first meteor, but updating it with variables for the second meteor. If you think this sounds tedious, you are correct! It will work, but your code will be long and cumbersome.
- **Approach #2:** Use [arrays](#) and [for](#) loops to add more meteors! Arrays allow you to store an ordered list of elements in a single variable. That is, instead of just storing one value for a meteor's diameter, you can store 10, or 15, or 20 or 300 - and you can access them all using the index number. For loops allow you to loop through a section of code multiple times based on a condition. You could say that for loops and arrays are BFFs because you can use a [for](#) loop to iterate or cycle through an [array](#). See the **Extension Resources** below to learn more.

In this extension, try using arrays and for loops to make four more meteors for a total of five. Arrays will allow you to store the meteorX variables you need for all the meteors in one place. For loops will allow you to cycle through each of those variables and use it or perform an action on it.

Here are the basic steps you need to complete this extension:

- ❑ Declare **arrays** to store the X position of the meteor, Y position of the meteor (these values should all be 0), the meteor diameter, the distance, and the speed.
- ❑ Create **for** loops in the **setup()** that will pre-populate the following arrays with initial random values: X position, meteor diameter, and speed.
- ❑ Create a **for** loop that draws the meteors to the screen.
- ❑ Create a **for** loop that cycles through the meteors to make them fall at different speeds.
- ❑ Draw the catcher.
- ❑ Create a **for** loop that cycles through meteors to determine the distance between each meteor and the catcher.
- ❑ **Create a **for** loop that cycles through meteors to test if one of the meteors has intersected with the catcher.** If it has intersected, redraw that meteor to the top of the screen at a random X position, and set a new random speed for that meteor.  
*Tip: You will need to add a conditional statement inside the for loop.*
- ❑ **Create a **for** loop that cycles through meteors to test if one of the meteors has intersected with the bottom of the screen.** If it has intersected, redraw that meteor to the top of the screen at a random X position, and set a new random speed for that meteor.  
*Tip: You will need to add a conditional statement inside the for loop.*

### Extension Resources

Below are some helpful resources we used to create this Extension. These will help you get started, but remember that there are lots more resources only a search engine away!

- p5.js Tutorial videos from The Coding Train  
**Note:** some of these videos were created before the online editor was built, but it will work the same!
  - ◆ [What is an Array?](#)
  - ◆ [Arrays and Loops](#)
  - ◆ [while and for loops](#)
- [Program Flow](#) tutorial
- [Iteration](#) tutorial
- **for** If you are having trouble getting started, try using the first approach and create a second meteor by duplicating code from the first meteor. Any time you see a double (e.g. **meteorY\_1** and **meteorY\_2** or **speed\_1** and **speed\_2**) chances are you will need an array and a for loop to cycle through it.

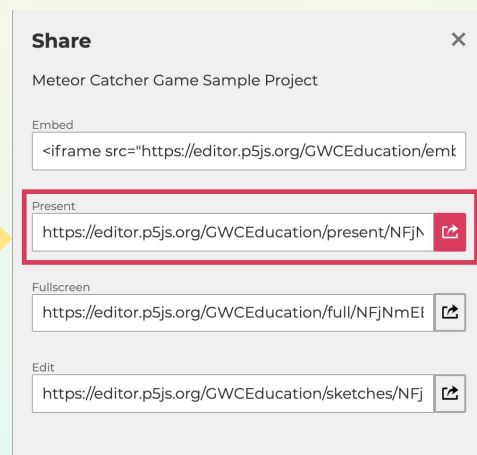
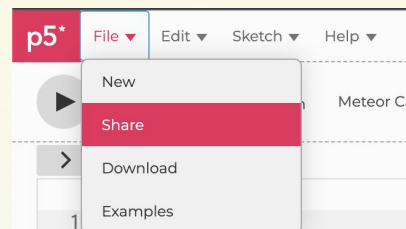
Here is a link to our [solution code for this extension](#). We recommend trying it yourself first and using this resource when you get really stuck or want to check your work.

## Step 9: 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 game with us! Don't forget to tag [@girlswhocode](#) [#codefromhome](#) and we might even feature you on our account!

**Follow these steps to share your project:**

- Save your project first.
- In the **File** Menu, choose the **Share** option in the dropdown menu.
- Choose the **Link** option in the dropdown menu.
- Copy the **Present** Link paste it wherever you would like to share it.



**Project Link**

**Stay tuned for more Girls Who Code at Home projects!**

