**Lesson 7** Randomize with mousePressed()

**How can we control a sketch by clicking the mouse?**

| **Overview** | |
| --- | --- |
| This lesson introduces students to the mousePressed() function so that they can control a sketch by clicking the mouse. Students will also use the random() function to generate different values for shapes. | |
| **Lesson Objectives** | |
| Students will be able to   * Use mousePressed() to control a p5 sketch * Assign random values to variables * Use the console to get information about a program | |
| **Suggested Duration** | |
| Two periods (90 minutes) | |
| **Blueprint Foundations Student Outcomes (**https://blueprint.cs4all.nyc/outcomes/) | |
| Abstraction  Analyze | **Describe** how I might use patterns to express an idea. |
| Abstraction  Prototype | **Explain characteristics** or patterns that informed a function or an interface I created. |
| Algorithms  Prototype | **Demonstrate** the benefit of using an event, conditional or loop in my prototype. |
| Algorithms  Communicate | **Compare and contrast** how conditionals or loops were used in classmates’ prototypes. |
| Programming  Communicate | **Present** the challenges, and benefits of using tools to program collaboratively. |
| **Vocabulary** | |
| * **Console**: Used by programmers to test and debug code by displaying information about the program | |
| **Planning Notes** | |
| * In the student activity, they will make a [click button](https://editor.p5js.org/mparker/sketches/qCQ-bZv5l). You may choose to have them work in groups of 3-4 (as with the traffic light activity in [Lesson 4](https://docs.google.com/document/d/1pd-SJZ5_WjUOZ6JAajiMzAahBASvNMVQ0xbdfkOmDbM/edit?usp=sharing)) or in pairs.   + If working in groups, you may provide **chart paper** for their pseudocode. * For pair programming, decide on pairs in advance, preassign first roles, and make sure students switch roles every 5 - 10 minutes. | |
| **Resources** | |
| * Video tutorial: [2.5 The random() function](https://youtu.be/nfmV2kuQKwA) | [Code](https://github.com/CodingTrain/website/blob/master/Tutorials/P5JS/p5.js/02/2.5_p5.js_random/sketch.js) * Random in p5: [Happy Coding Tutorial](https://happycoding.io/tutorials/p5js/random) * Interactivity with p5.js: [Mouse Events](https://p5js.org/learn/interactivity.html) section * Do Now video: [Is Anything Truly Random?](https://www.youtube.com/watch?v=tClZGWlRLoE) (consider stopping at 3:11 because end of video uses a lot of jargon that may overwhelm your students) | |
| **Assessments** | |
| * Assess the **Student Activity**. Check for the ability to:   + Declare and use variables that control the x and y parameters of a shape   + Use mousePressed() to control a sketch   + Generate random values using the random() function * Assess the **Wrap Up**. Check for the ability to:   + Describe debugging strategies | |

| **Do Now:** |
| --- |
| * Show students this [video](https://www.youtube.com/watch?v=tClZGWlRLoE) and have them answer the following prompts in their **design journals**:   + Based on the video, what does it mean to be random?   + Why is randomness so important to our society? What are real-life situations where random numbers are needed?   + What are some ways you might use random numbers in a p5 sketch? |
| **Discussion: Randomness and p5’s random() function** |
| * Discuss students’ responses from the **Do Now**. In this lesson, they will randomize different parts of their sketch using p5’s **random()** function. * Explain that the random() function uses two parameters to generate a random number: a minimum and maximum value.   + If only one value is used, **the default minimum is 0.** So random(0, 300) is equivalent to random(300). |
| **Teacher Demo: mousePressed() and random()** |
| Code Along   * With your students, build out the following [example](https://editor.p5js.org/mparker/sketches/ETlGnSqL2) that randomly changes the background color of the canvas when the mouse is pressed. Point out that:   + Just like keyPressed(), the mousePressed() function should be written **after** the draw loop, not inside it.   + You’re using a minimum and maximum of 0 to 255 because that’s the range of values used for colors. If you used a range like (255, 500), the background would always be white!     Using the console   * Add the following to the draw loop to display the actual number that the random() function generates.   + Ask: What do you notice about the numbers?   + Ask: What value might bgColor be when the text “disappears”?      * Students should notice that by default, random() generates long decimal numbers. For functions like background(), p5 will round the decimal to a whole number behind the scenes. * When values are very close to 255, it gets harder to read the white text. In a lot of cases, it isn’t practical to use text() to draw information on the screen. Instead, developers use the **console** to display helpful information about the program:      * Conclude this part of the demo by saying that as a class you will transition toward using the console for displaying information only meant to be seen by the **programmer**, and using text() for displaying information meant to be seen by **users**.   Independent Exercise: Randomize RGB   * Right now, the background color can only be black, white, or gray. Students should duplicate this [starter code](https://editor.p5js.org/mparker/sketches/SB4KkDtje) and follow the steps to make the background **any** random color when the mouse is pressed. If they get stuck, remind them to follow debugging protocols and ask a neighbor for help. * Extension for early finishers: Update the sketch so that when the mouse is pressed, the background is a random shade of purple.   + [Solution](https://editor.p5js.org/mparker/sketches/iKsSWVuS4) for teacher reference   + [Challenge solution](https://editor.p5js.org/mparker/sketches/N-1BgFfYD) for teacher reference |
| **Student Activity: Click Buttons** |
| * In lesson 5, students used conditionals and the && operator to turn a rectangle into a hover button. To make a click button, all students need to do is put that conditional inside mousePressed(). * Expand on the sketch from the independent exercise by drawing a rectangle to be used as a button. Ask students to help you write a conditional that only changes the background color when the mouse is on the rectangle. Below is an [example](https://editor.p5js.org/mparker/sketches/tC6s6-2cS) of what the complete sketch might look like:     Independent Exercise: Randomize Size   * Students should duplicate this [starter code](https://editor.p5js.org/mparker/sketches/a7J3ECzFm) and follow the steps to randomize a button’s size when it is pressed. Here is the [solution](https://editor.p5js.org/mparker/sketches/AKIuMsVXK) for teacher reference. * Turn and Talk: Why does the clickable area stay the same, even though the button size changes? *Answer: Because the conditional only checks to see if the mouse is in the upper left corner.* * Explain that the conditional “assumes” that the button has a fixed width and height of 75, even though the *actual* width and height is changing to a new random amount.      * You could rewrite the conditional to look like this, and it would behave the same way:      * Turn and Talk: Instead of adding 75 to 50, what amount do we actually want to add to 50? *Answer: Whatever the random width and height currently is.* * Conveniently, the current value of the button’s width and height is stored inside the **variables** rectW and rectH. Rewrite the conditional to use those variables and make the whole button clickable, no matter what the width and height are:        * + Sketch can be found [here](https://editor.p5js.org/mparker/sketches/BKcsZoVUl)   Group Activity: The Moving Button   * In this activity, students will work in groups or pairs to create a sketch that moves a rectangular button to a new random position whenever it is pressed. Here is the [solution](https://editor.p5js.org/mparker/sketches/qCQ-bZv5l) for teacher reference.      * Before students begin coding they should develop a plan using pseudocode either on **chart paper** or in their **design journals**.   Instructions:   1. Create a rectangular button 2. Declare two **variables** that control button position 3. The button should move to a new **random** position when it is pressed 4. Use **console.log()** to display position variables in the console when the mouse is pressed 5. CHALLENGE: Sometimes the button’s edges end up off the canvas. Update the sketch so the entire button is always on the canvas. |
| **Wrap Up** |
| Share Out:   * [**Design Journal**] First have students reflect on the following:   + What was the most challenging part of making this?   + What strategies did you use to get unstuck?   + How might you change or add onto this project in the future? * Invite a few pairs to come up and share their process:   + Describe your project and how it works.   + Share your responses to the reflection questions. * Make sure students share their code with each other and with you before they leave. |
| **Extension: N/A** |
|  |