Mouse Theremin with p5.js (Intermediate)

Liam Baum
BELL Academy, Bayside, New York, United States

Description

A simple sketch using the p5.js programming language to code a simple theremin that changes pitch and volume by moving the mouse around the screen.

Materials

- computer (laptop, PC, etc.)
- internet connection to access the website: editor.p5js.org

Context

This project works well with middle and high school students who have a basic introduction to text-based coding languages. The basic part of this lesson can be done with little to no prior coding experience. However, there is a lot of opportunity for variation and extensions where understanding some basic coding concepts can be very useful.

Lesson Time

15-20 minutes of guided instruction with any remaining class time to be used for experimentation.

Lesson Instructions

1. Go to p5.js web editor: editor.p5js.org (This project can be done without having an account, however you will not be able to save any work) **(Figure 1.0)**

   The p5.js web editor has an area to write the code on the left and the canvas which will display the results of the code on the right. Keep in mind that this project is working primarily with sound so you will not see anything happen in the canvas when you run the code except a light grey background. The extensions provide opportunities to add visuals to this project. **(figure 1.1)**

   The text editor will already have some code written in it.
   function setup{} - This is where we enter code that will happen at the beginning of our program that only need to happen once. All the code should be written between the two curly brackets.

   createCanvas(400, 400) - This makes the canvas that our sketch will happen on. The canvas consists of an X axis and Y axis. The numbers inside the parentheses are how many pixels this canvas will be. As written, the canvas will be 400 pixels across the X axis and 400 pixels down the Y axis. These numbers can be changed. Have students experiment with different numbers. **(figure 1.2)**

function draw{} - This is where we will write our code that will need to change while we run the program. It will constantly be updating itself to account for any changes. This is known as a ***loop*** in computer programming. All code should be written between the two curly graphics.

background(220) - This provides the coloring of our canvas. You can provide a grayscale value from 0 (black) to 255 (white). Have students experiment with different numbers. **(figure 1.3)**

2. Click on line 1 in the text editor and hit return twice to space the existing code down. Go back to line 1 and write the following code:
   1 let osc;

   (figure 1.4)

   let is the command to declare a variable in p5.js. A variable can be made to hold information that we will refer to later in our program. **osc** is the name we are going to give this specific variable (You can name your variable whatever you want, but it is helpful to use words that connect with what you are going to use that variable for. You can not start a variable name with a number). This variable is going to hold what will make the sound in our program.

   Make sure to include the semicolon at the end of each line of code which is a part of the syntax of P5.js.

3. Go to line 5 and hit return two times to move the closed curly bracket down to make space for the next lines of code. Go back to line 5 and write the following line of code:
   5 osc = new p5.Oscillator();

   (figure 1.5)

   This line is adding an Oscillator into our code. An oscillator is a way to generate synthetic sound by simulating a vibration (All sound in the real world is created by a vibration). This oscillator is going to be stored in the variable we declared called **osc.**

4. In line 6, write the following line of code:
   6 osc.start();

   (figure 1.6)

   This line of code is going to start our oscillator when we run the program. Hit play and you will hear a pitch play on your computer.

5. Move down to line 11 and hit return a few times to move the closed curly bracket down to make space for the next lines of code. Go back to line 11 and write the following lines of code:
   11 osc.freq(440);

   (figure 1.7)

   This line of code will allow us to change the frequency of our oscillator. The number that goes inside the parentheses is the amount of hertz the frequency will play at. Have students experiment with different numbers to produce higher or lower frequencies.

   Note: Really high numbers may produce painfully high frequencies or frequencies you are unable to hear. This is a good opportunity to discuss the range of human hearing to find an acceptable frequency range.

6. Make a space on line 10 and write the following line of code:
   10 let pitch = mouseX;

   (figure 1.8)

   This line of code is creating a new variable called pitch. This variable is going to hold a value that we will use for the frequency of our theremin. **mouseX** is the value that represents the location of the mouse on the X axis of our canvas. So the value will be any number from 0 to the size of the canvas we are using. (Remember, we set the size of our canvas in the **createCanvas()** command. The first number will tell you how wide your X axis is. If you haven't changed this number since starting the program, it should be 400.)

   Go to osc.freq and change the value inside the parentheses to pitch =
   osc.freq(pitch);

   (figure 1.9)

   Run the program and move the mouse from left to right over the canvas. You should hear the pitch change from low to high as you move from left to right.

   Students will probably notice that the sound gets quieter as you move toward the left of the canvas. This is because you are getting into the lower range of frequencies that we cannot hear (from about 40Hz down to 0).

7. To choose our own range of frequencies for our mouse, we are going to use a process called **mapping.** Mapping takes a variable that contains one range of values and converts that to a new range of values. In this case, we want to convert the range of our X axis (the

width of our canvas) to a range of frequencies of our choosing starting on a low pitch and moving to a high pitch.

Go to line 10 and make the following changes to the code:
10 let pitch = map(mouseX, 0 , width, 200, 1000);

(figure 1.10)

This line of code is saying "Take the range of the value of mouseX which is between 0 and the width of the canvas and convert it to a range of values between 200 and 1000. Then put whatever that value is into the variable called pitch."

Run the program and move the mouse between the left and right side of the canvas. You should be able to hear the changes in frequency between the two values you mapped mouseX to. Have students experiment with different frequency values (the 200 and 1000 in the map command.) See what happens if they put a larger number before a smaller number)

Note - width can be used to represent the size of your canvas regardless of the value you have put in the createCanvas() line of code. This way if you decide to change the size of your canvas, you do not need to make the change elsewhere in your code.

8. Traditionally a theremin changes both frequency(pitch) and amplitude(volume), so now we will add a command to have the volume change when we move the mouse along the Y axis of our canvas which is from the top to the bottom.

   Make a space underneath the mapped pitch variable on line 10 and write the following code:
   11 let volume = map(mouseY, 0, height, 1, 0);

   (figure 1.11)

   This line of code is using several pieces of code we have already used with a few changes. We are declaring a new variable called volume which is going to hold the value for the volume of our oscillator.

   We are mapping the value of mouseY which is the location of the mouse on the Y axis at any given time. The range of the mouseY is from 0 to height of our canvas (The value of height is determined by whatever number you have as the second value in the createCanvas command. In our case, it is 400. But the same as width, if we change that value, height will update to that new value without us having to change anymore code).

   We are mapping the range of the value of mouseY from 0 to height to a new range which is between 0 and 1 which is the values that are used for the amplitude of the oscillator which we will get to next. It is important to point out that the Y axis of the canvas starts

at the top which is the value 0 and the bottom of the canvas is height which in this case is 400. This is somewhat counterintuitive and takes a little getting used to.

This is why we map the new value as 1 first and then 0. This means that when we are at the top of the canvas, the volume will be the loudest and when we are at the bottom of the canvas, the volume will be the quietest. This way as you move the mouse up, the volume increase. You could map the values from 0 to 1 but the mouse will get louder as it moves down and I have found this to be visually confusing in relation to what you are hearing.

9. Go to line 14 underneath the osc.freq(pitch); command and write the following code:
14 osc.amp(volume);

(figure 1.12)

This line of code changes the volume of the oscillator (amp is short for amplitude. This is an opportunity to discuss what amplitude means = The intensity at which a vibrations moves; Large vibrations produce louder sounds). This command takes a value between 0 and 1. Numbers above 1 will work but can distort the sound of the oscillator. In our case, we are using the variable we made called volume which changes between 0 and 1 depending on when our mouse is on the Y axis of our canvas.

Run the program and move the mouse around the canvas. Students should notice that now the pitch changes when moving side to side and the volume changes when moving up and down.

Complete Project code
(**figure 1.13)**

Modifications for Learners
With the p5 web editor, you can share projects or give students templates to work off of with a certain amount of code already entered in. Because this is text-based coding, you can also provide a sheet of commands or the code they would need to input.

Students who are able to complete this project quickly have several options. There is so much more which can be done in p5.js in terms of visuals to accompany this project. There are also ways to change the note output to make it more musical. Below are two completed examples which show examples of this. You can also encourage students to visit the p5.js reference page where they can find new commands, documentation and examples to experiment with.

P5 mouse theremin with color changes -
https://editor.p5js.org/mrbombmusic/sketches/ryUKXE_v7

Complete project code
(figure 1.14)

P5 mouse theremin with one octave chromatic scale -
https://editor.p5js.org/mrbombmusic/sketches/BkUfpQuD7

Complete project code
(figure 1.15)

Learning Outcomes

- Coding concepts - Syntax, loops, declaring variables, mapping values
- Basic graphics - Pixels, X and Y axis
- Sound Synthesis - Oscillators, frequency, hertz, amplitude

Assessment Considerations

The first and most obvious check for assessment is to make sure the program runs correctly without any error messages. This is easy to see since the editor will let you know with some type of error message indication.

For assessing a deeper understanding, students can be asked to explain how parts of the program work. This can be for both the code itself and the musical concepts expressed in the program. For example, they can explain why the pitch gets higher when moving the mouse from left to right and not up and down (Coding concept - mapping values to mouseX, not mouseY). They can also explain how the number values in the map function correspond to frequency and hertz (Sound concept - the higher the hertz, the higher the pitch). Students could be asked to explain these concepts to a teacher or peer assess each other.

Students could also be asked to change certain parts of their code to assess how well they understand all the parts and how they connect to the final result. For example, students could be asked to change the program so that the pitch gets lower when moving from left to right instead of higher or to switch the pitch to moving the mouse up and down and the volume to moving left to right. Having students do this and allowing them to explain the decisions they made can provide great opportunities to demonstrate understanding of the skills needed for this project.

More, Please (Further Reading/Resources)

Completed project - https://editor.p5js.org/mrbombmusic/sketches/ByM0KIPrX
P5.js Sound Library Reference - https://p5js.org/reference/#/libraries/p5.sound
The Coding Train tutorials on p5.js sound library by Dan Shiffman
(Search - p5 sound tutorial coding train)
https://www.youtube.com/watch?v=Pn1g1wjxl_0&list=PLRqwX-V7Uu6aFcVjlDAkkGIixw70s7jpW

About the Author

Liam is a lifelong music maker and learner, having started playing his first instrument at age 4. He holds a BS in Jazz Studies from SUNY New Paltz and a MS ed from CUNY Queens College. He has been a music teacher for 12 years, currently teaching instrumental and general music at the BELL Academy, a public middle school in Bayside, NY. Over recent years, Liam has become increasingly interested in how technology can be used for the purposes of creative

artistic expression. He is a Teacher Ambassador for Makey Makey and has led several workshops around the NYC area on creative coding and physical computing at events including Creative Coding Fest (ccfest.rocks/), Hip Hop Hackathon (hiphophacks.com) and Monthly Music Hackathon (monthlymusichackathon.org/). Liam currently lives in Queens with his wife and two daughters. Follow him on twitter: @mrbombmusic