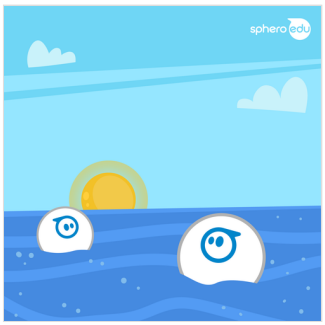
Recursion & Ocean Colors



In this activity, you will learn about an advanced programming technique called recursion. At the same time, your Sphero will visualize why the ocean appears blue by showing the connection between ocean depth on color wavelength.

LEARNING OBJECTIVES:

- I can practice industry standards, including:
  - Printing data to the console
- I can define and use CS fundamentals, including:
  - Recursion and base case
  - Writing pseudocode
- I can use JavaScript to create a program.
- I can execute the created program using Sphero.

Tags: functions text coding JavaScript recursion base case wavelengths console

Grades: 7 to 12+ | **Duration**: 1-2 Hours

CCSS: CCSS.Math.Content.6.EE.C.9

NGSS: HS-PS4-1

Step #1: Exploration - Blue Oceans

Do you know why the ocean appears blue?

It is because visible colors of light penetrate differently into the ocean depths. The longer the wavelength, the more quickly it is absorbed. Blue is a short wavelength, so it penetrates to a deeper depth. Watch the video below for more information.

In this activity, you will use your Sphero to visualize which colors are absorbed most quickly and which colors you can see at a greater depth.

YouTube video: https://youtu.be/XA3rNgyEmwA

Step #2: Skills Building - Deep Dive

Make your Sphero dive to the bottom of the ocean! The following depths (in meters) are just past the threshold for where colors are absorbed into the ocean: 2, 5 ,7, 12. You will use the Sphero to mimic this dive.

Copy and paste the code for the full program below into your Sphero Edu app. Try running the code!

- *How far did the Sphero travel?*
- *What color appeared?*

Copy

```
const rainbow = {
   5: { r: 255, g: 0, b: 0 },
   4: { r: 255, g: 140, b: 0 },
   3: { r: 255, g: 255, b: 0 },
   3: { r: 0, g: 255, b: 0 },
   1: { r: 0, g: 0, b: 255 },
}

function getNumberOfColors(depth) {
   if (depth <= 4) {
       return 5;
   } else if (depth > 4 && depth <= 7) {
       return 4;
   } else if (depth > 7 && depth <= 12) {
       return 3;
   } else if (depth > 12 && depth <= 22) {
       return 2;
   } else {
       return 1;
   }
}

function getOceanColors(colors, count) {
   if (count === 0) {
       return colors;
   } else {
       colors.push(rainbow[count]);
       return getOceanColors(colors, count - 1);
```

```
    }
}

async function startProgram() {
    const duration = 2;  // CHANGE ME
    const speed = 60;

    await roll(0, speed, duration);

    const numberOfColors = getNumberOfColors(duration);
    const oceanColors = getOceanColors([], numberOfColors);

    let i = 0;
    while (i < oceanColors.length) {
        await strobe(oceanColors[i], 1, 1);
        await delay(0.5);
        i++;
    };

    await speak("Ocean depth is " + duration + "meters");
    exitProgram();
}
```

Step #3: Skills Building - Changing Variables

Now, change the duration variable in your program to the following numbers. Record what colors show for each number.

- 2
- 5
- 7
- 12
- 22

Based on your observations above, make a hypothesis for which colors are absorbed into the ocean at each distance. *Can you now explain why the ocean looks blue?*

Step #4: Challenge - Recursion

Recursion is an advanced computer science concept: it involves calling a function within a function. Recursive programs have two components:

- A base case—a scenario, that when reached, ends the function from being called.
- A set of rules that makes sure the base case is eventually reached.

Take a look at the function below and answer the following questions:

- *What is the base case?*
- If the count is 3 when you first call the function, *how many times will getOceanColors be called in total?*
- *Why is an empty array passed into the function the first time you call it?*
- *What do you think colors and count equal every time the function is called?*

Copy

```
const rainbow = {
   5: { r: 255, g: 0, b: 0 },
   4: { r: 255, g: 140, b: 0 },
   3: { r: 255, g: 255, b: 0 },
   2: { r: 0, g: 255, b: 0 },
   1: { r: 0, g: 0, b: 255 },
}

function getOceanColors(colors, count) {
   console.log(`count is ${count} and colors are ${JSON.stringify(colors)}`
   if (count === 0) {
       return colors;
   } else {
       colors.push(rainbow[count]);
       return getOceanColors(colors, count - 1);
   }
}

let numberOfColors = 4;
getOceanColors([], numberOfColors);
```

Let's test your answer in the last question! You might have seen console.log(colors) in the code. You can use this line to print data. It is often used in simple debugging. We are going to use it to see more clearly how this recursive function works.

- First, open up an Internet browser on a laptop or PC. Did you know that you have a JavaScript engine in your browser? You can access it using your developer tools. If you aren't sure where to find your developer tools, ask a teacher or do a search on the Internet.
- Try pasting the code above into your developer console. When you press enter, console.log() will print the colors and count each time the function runs. *Does it matched what you guessed?*
- Try changing numberOfColors to a different number. *What do you expect it to print this time?* Test your answer in the JavaScript console again. *Were you correct?*
    - **HINT:** Your console remembers the variables you've declared, so you only need to paste the following lines in each new time you want to call it!

```
numberOfColors = 3;
getOceanColors([], numberOfColors);
```

Copy