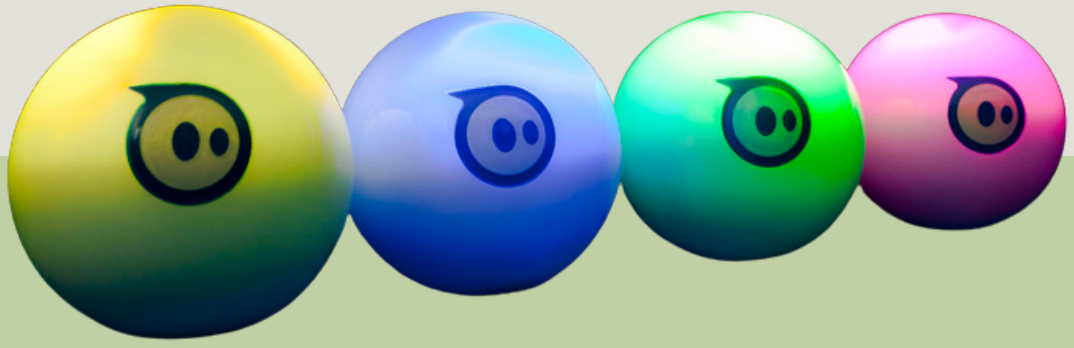
SPHERO MINI

APP-ENABLED ROBOTIC BALL
LESSON PLANS

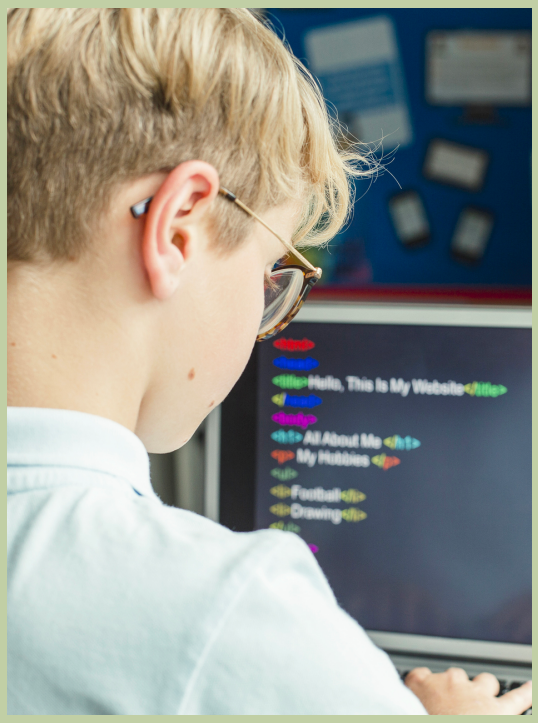


Created by: Sonika Tamilarasan
Bit-by-Bit

TABLE OF CONTENTS

MEET SPHERO

This lesson plan for Spheros aims to give students, ages 6 and above, an understanding of coding. Spheros are a wonderful way for beginners to start. There are three different levels included in the Sphero Edu app: draw, blocks, and text. In this lesson plan, students will learn the basics of coding using both block-based and text-based programming and create their own programs to make Sphero move, change colors, and respond to sensor input.



Get Started

Materials:

- A charged Sphero
- 6 mini bowling tests, and 3 cones
- Paper, pencil
- Smartphones, tablets, or laptop

Get Started:

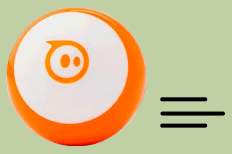
- Download the **Sphero Edu** app: https://edu.sphero.com/d
  - Select the type of device you are using
  - Select I'm a School user
  - Click "Let's Code"
- Look at the top-right corner and click on the "Sphero-phone" to connect to your Sphero.
  - Scroll to find Sphero Mini
  - Your Sphero will light up when it is connected.
- Using the toolbar to the left, click on programs.
- On the bottom right-hand corner of the screen click on the plus button.
- Makes sure Sphero-mini and either click (Blocks/Text) is selected. You now have an IDE (Integrated Development Environment), that you can use to program!

*Alternatively, you can also download the **Sphero Play** to use it as a console for the first 2 projects. This one does not involve coding, so it's very beginner friendly.*

WARM UP

Let's see the Sphero in action! Drag out each block as instructed and observe the Sphero execute the code.

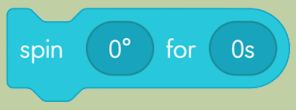
Drag out a speed block. What happen?



The Sphero moves!

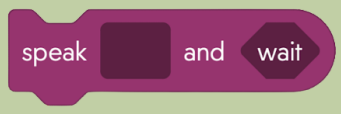
Now what happens when you change it to a greater number?

It moves faster!



The Sphero changes direction! If you want the Sphero to move forward make sure the light is pointing towards you.

You can change the color of the Sphero!



The Sphero can also talk! The sound outputs from the device it is connected to so make sure your volume is up!

There are plenty of blocks in the IDE that will help you navigate your Sphero. The next couple of pages will guide you through each individual block.

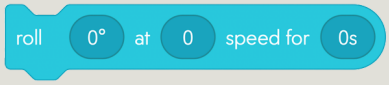
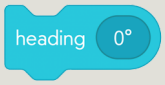
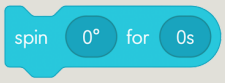
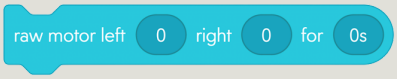
HOW TO CODE SPHERO

Aim Button

Use the circle to drag the light toward you. See the image below for an example.

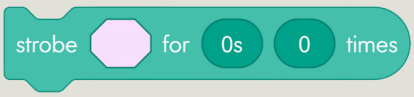
Movement Blocks

These light blue blocks can be used to move or stop your Sphero, take a look at each block to see it's function!

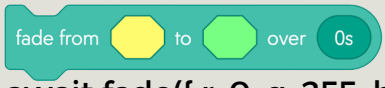
 await roll(0, 0, 0)	Makes the robot roll in a specific direction(1) at a specific speed(2) for a number of seconds(3).
 stopRoll()	Sets the speed to zero to stop the robot, which is the same as the setSpeed(0) command.
 setSpeed(0)	Sets the speed of the robot from -255 to 255, where a positive number is forward, and a negative is backward.
 setHeading(0)	Sets the direction the robot rolls, 0˚ is forward, and 180˚ is backward (towards you).
 await spin(0, 0)	Spins the Sphero at a certain degree for a given amount of seconds. This is very useful in making the Sphero move in a circle!
 await rawMotor(0,0,0)	Controls the electrical power sent to the left and right motors separately. It can be used to make your Sphero jump.
 resetAim()	Sets the aim back to 0˚ when using block, but when using the text editor you have the ability to input a direction of your choice.

Light Blocks

Control the color of Sphero with these blocks!

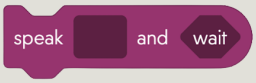
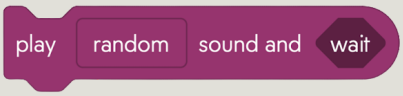
 await strobe({ r: 255, g: 57, b: 66 }, 3, 15)	This command repeatedly blinks the main LED lights. The time is how long the light stays on for a specific blink, and cycles in the total number of blinks.
 setBackLed(255)	Sets the brightness of the back aiming light. The parameter is from 0 to 255, with 255 being the brightest.

HOW TO CODE SPHERO

 setMainLed({r: 90, g:255, b:90})	Changes the color of the main LED light. The text editor uses RGB values!
 await fade({ r: 0, g: 255, b: 0 }, { r: 0, g: 0, b: 255 }, 3.0)	Changes the main LED lights from one color to another over a period of seconds.

Sound Blocks

The Sphero can speak? Yes they can! Use the following blocks to control sound.

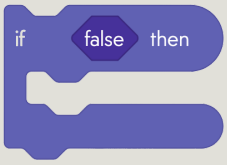
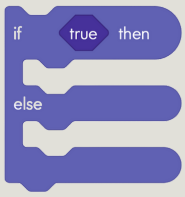
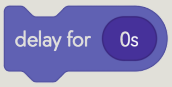
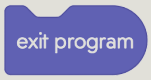
 await speak('Hello World', true)	Uses your device's text-to-speech engine to speak the 'string' you typed in.
 await Sound.Category.SoundName.play(true)	Plays a random noise or selected sound. If true, then waits for command to be fully executed to move on to the next step.

Control Blocks

These blocks are very useful in making your code concise. They can be used to insert conditional statements, loops, repeats, and causes.

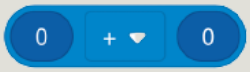
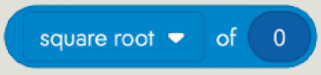
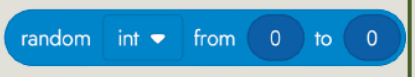
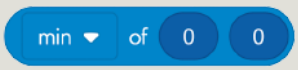
 while (true) {}	Repeats a section of code for a infinite amount of times.
 while (!testCondition) {}	Repeats a section of code until the specified condition is met.
 for (initialize loop variable; check loop condition; update loop variable) {}	Repeats a section of code a certain amount of times.

HOW TO CODE SPHERO

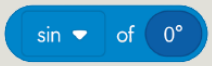
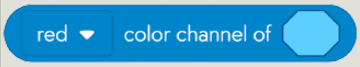
 if (condition) {}	If a condition is true then it executes a section of code, if not it doesn't execute the code.
 if (condition) {} else {}	If the condition is true then it executes the code in the first section, if not it executes the code int he second section.
 await delay(4)	Creates a pause between two lines of code for a number of seconds.
 exitProgram()	Quits the program.

Operator Blocks

Math is everywhere! Use these operator blocks to perform mathematical operations for your program!

 (+, - , *, /, **, %, --, ++)	Use the dropdown button to access various mathematical function. Performs arthmetic operations between 2 numbers.
 getRandomColor()	Changes the LED of Sphero to a random color.
 sqrt(x) , round(x), floor(x), ceiling(x), abs(x), sign(x)	Use the dropdown menu to access various mathematical methods. Ceiling means to round up, and floor means to round down.
 getRandomInt(min, max) getRandomFloat(min, max)	Returns a random whole number between the 2 inputted values. It includes the min/max values. This can also be used for float (decimal).
 min(x, y, ..., n), max(x, y, ..., n), pow(x, y), random(), round(x), sign(x), sqrt(x) ,trunc()	Multiple functions. pow(x, y) returns the value of x to the power of y, random() returns a random number between 0 and 1, round(x) rounds a value to the nearest integer value (whole number), sign(x) returns the sign of a number, trunc() returns the integer part of a value by removing any fractional digits.

HOW TO CODE SPHERO

 sin(x), cos(x), tan(x)	Gives the trigonometric value of the given degree.
 buildString()	Combines multiple values to a single string. Values can be variables, parameters, sensors, integers, strings, booleans, or colors.
 { r: 255, g: 255, b: 255 }.x	Gets an individual color channel value of the main LED lights.
 Object.assign({}, { r: 255, g: 72, b: 222 }, { r: 0 })	Returns a new color by modifying a single channel (red, green, blue) of a given color.

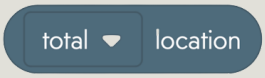
Comparator Blocks

Comparator blocks are typically used to compare values that are gathered from the sensor.

 getConnectedRobotType() === RobotType.mini	Only executes code for a given type of robot.
 && or ||	&& - ("and") both values need to be true || - ("or") only one needs to be true.
 ===, ==, !=, !=\=, <, <=, >, >=	=== Evaluates if 2 values are equal (does not compensate for different data types), == Evaluates if 2 values are equal, compensating for different data types, != Evaluates if 2 values are not equal compensating for different data types , !=\= Evaluates if 2 values are not equal, does not compensate for different data types.

Sensory Blocks

Gives you access to real-time values from your robot

 getOrientation().pitch getOrientation().roll getOrientation().yaw	Provides the tilt angle along a given axis measured by the Gyroscope. Pitch is the forward or backward tilt angle, roll is left or right tilt angle, yaw is the spin (twist) angle.
 getLocation().x getLocation().y	x provides distance from the origin of the program start (cm) horizontally. y provides distance from the origin of the program start (cm) vertically.

HOW TO CODE SPHERO

distance getDistance()	Returns the total distance traveled in cm.
total ▾ velocity getVelocity().x getVelocity().y	x is the right (+) or left (-) velocity. y is the forward (+) or back (-) velocity.
heading getHeading()	Returns target directional angle, in degrees.
speed getSpeed()	Returns the current target speed of the robot. Positive numbers are forward, negative numbers are backward.
main LED getMainLed(), getMainLed().r, getMainLed().g, getMainLed().b	Returns the RGB color of the main LEDs, from 0 - 255 for each color channel. r return red, g returns green, b returns blue.
back LED getBackLed() getBackLed().b	Returns the brightness of the back LED.
time elapsed getElapsedTime()	Returns the amount of time the that the program has run for.

Sensory Blocks

Allows you to incorporate conditional logic. When the Sphero is doing something, it can do something else. These also use sensors to detect it's motion.

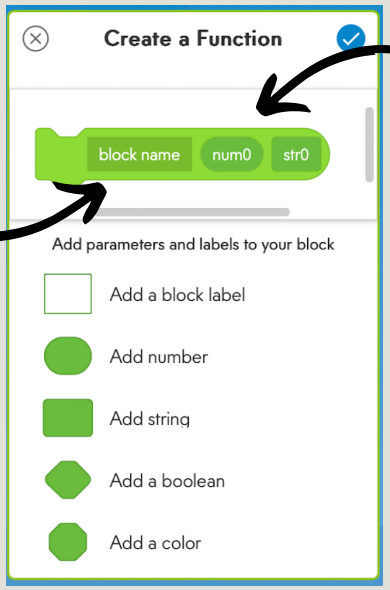
on freefall async function onFreefall(){}	Executes code when gravity is the only force acting on the robot.
on collision async function onCollision(){}	Executes when the robot collides with an object (collision needs to be somewhat impactful for the Sphero to detect it).
on gyro max async function onGyroMax(){}	Executes the robot exceeds 5.5 revolutions per second.
on landing async function onLanding(){}	Executes after free fall.

CREATE FUNCTION/VARIABLE

Create Function Block

This block can be used to create functions (a "chunk" of code that can be reused over and over again.)

Name the function



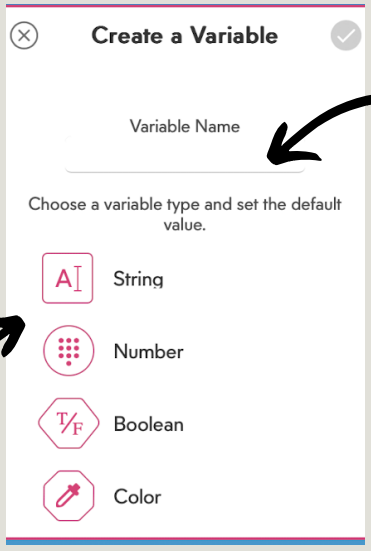
Add input variables (these variables can be used when you construct your function).

Add your commands after the define block to create your function. This function block can be used to simplify your code.

Create Variable Block

This block can be used to create a variable (a storage space for information).

Choose variable type



Name your variable

Variable blocks are very useful to store items such as scores, names, etc... The data store can be retrieved and manipualted.

CODING WITH THE TEXT EDITOR

I know the text editor might look scary, but fear not! It is very similar to block coding except now you just have to type the commands! The following lesson will walk you through some main syntax you need to get started!

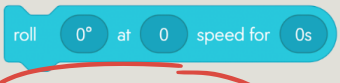
To begin, it's important to comprehend that computers do not communicate in the same way that we do. Therefore, you can't simply convey instructions to the computer in the same manner as you would to a friend. To make the computer do what you want, you have to write your code in a specific way, called "syntax."

On the previous pages, you may have noticed that underneath the block codes, there is a corresponding syntax attached to each command. It's important to note that these syntaxes must be followed precisely in order for your code to work correctly.

One essential rule to keep in mind when writing Java code is to end each line of code with a semicolon (;). This is similar to how we end sentences with a period in English. The semicolon tells the computer that the command on that particular line is complete and that it should move on to the next line of code.

Failing to include a semicolon at the end of each line can lead to errors in your code and cause your program to malfunction. So, be sure to always double-check your code to ensure that every command is properly punctuated with a semicolon!

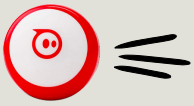
Remember, coding can be challenging at first, but with practice and patience, you can learn to write amazing programs with JavaScript!

 await roll(0, 0, 0)	Makes the robot roll in a specific direction(1) at a specific speed(2) for a number of seconds(3).
 stopRoll()	Sets the speed to zero to stop the robot, which is the same as the setSpeed(0) command.
 setSpeed(0)	Sets the speed of the robot from -255 to 255, where a positive number is forward, and a negative is backward.

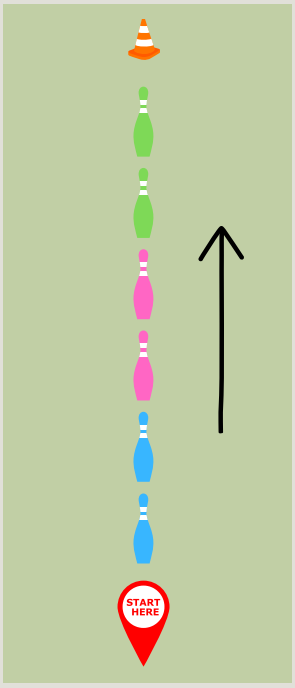
For more detailed explanations, take a look at the following website:
https://sites.google.com/sphero.com/sphero-edu-javascript-wiki/movement

PROJECT 1: KNOCK EM' DOWN

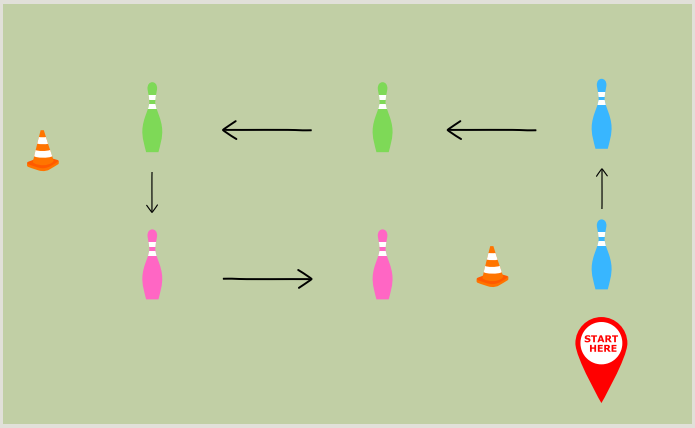
Set up the bowling pins and cones in the given formation. Try knocking the pins down with your Sphero and avoid touching the cones!



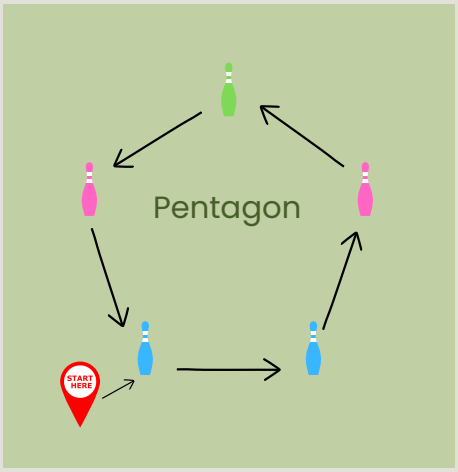
1.



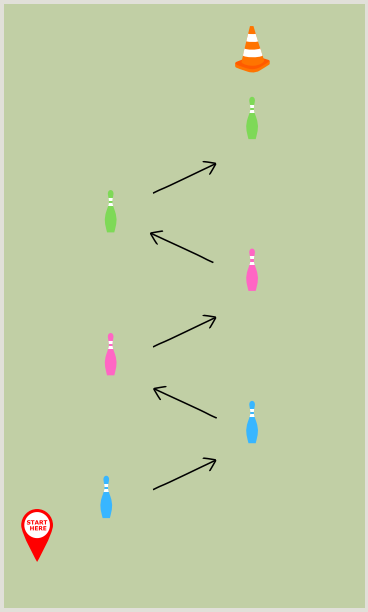
2.



3.



4.

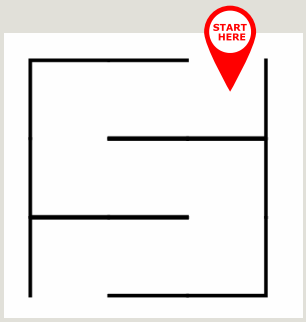


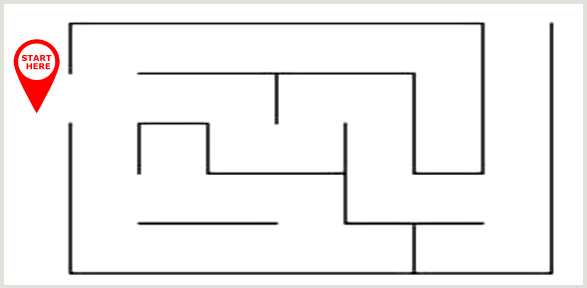
Make sure to space your cones out evenly and to leave at least 2 inches of space in between them!

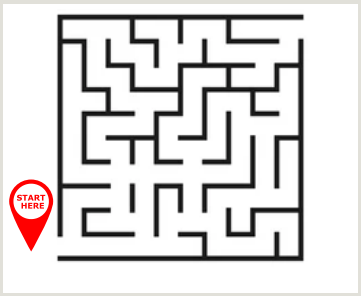
Challenge: create a formation for a friend and try to knock each other's down!

PROJECT 2: MAZE RUNNER

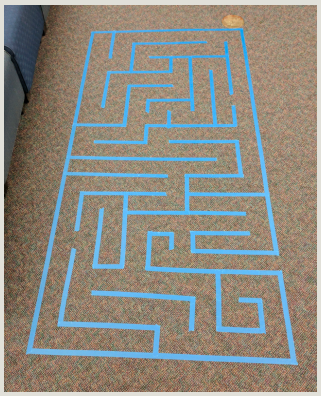
All you need are tape and a floor to execute this project. Create a Maze, and program your Sphero to navigate through it!







Easy **Medium** **Hard**

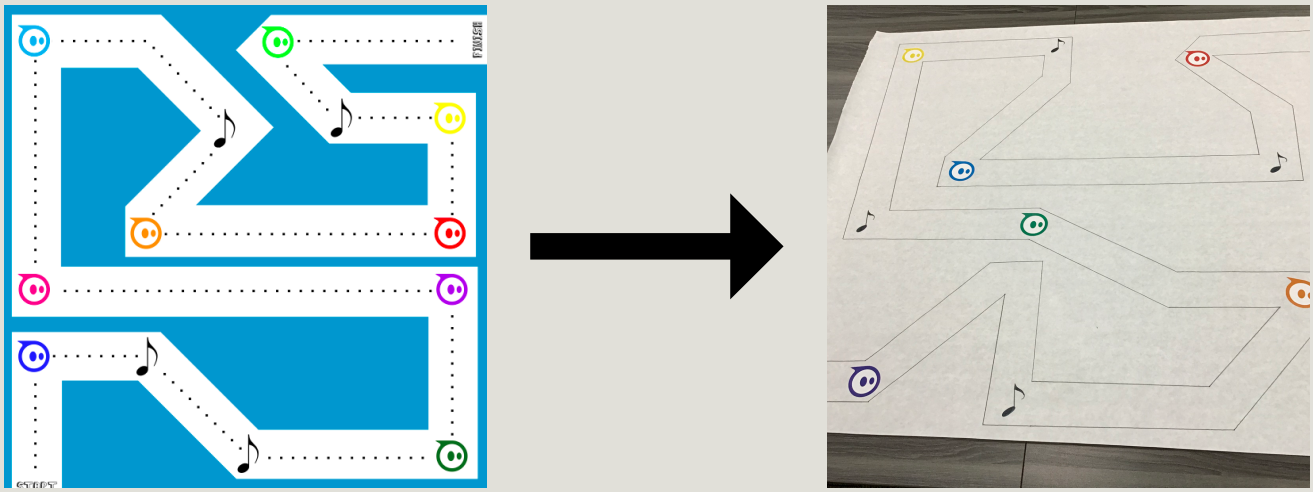


Materials needed:
Painters Tape/Masking Tape
Floor Space

Use the masking tape to recreate the mazes on a bigger scale for easier Sphero mobility.
Tip: Use algorithms to find patterns in the maze to simplify your code.

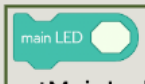
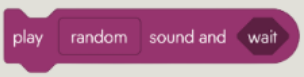
PROJECT 3: DANCE PARTY

Either create your own map, or use the one given by your instructor for this activity!



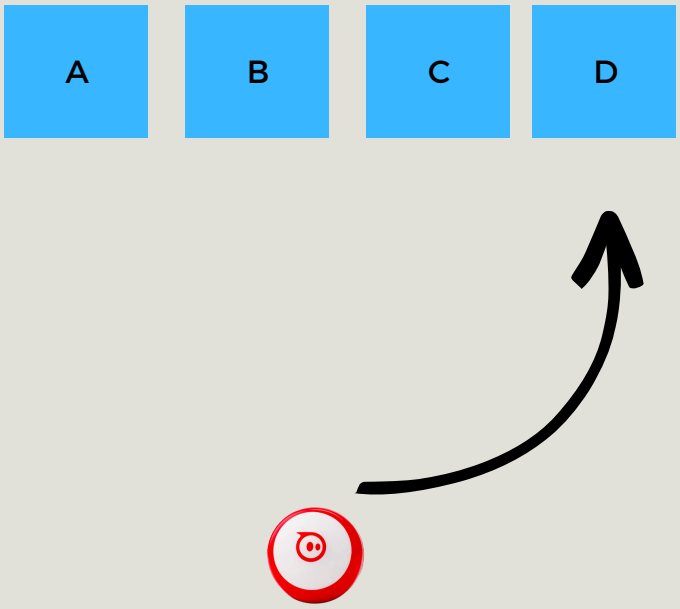
This might look similar to the maze runner, but it incorporates music and colors as well. Recreate the given maze on a large piece of paper for Sphero mobility. When the Sphero crosses the music note play a sound, when it crosses a color change it to that color, and make sure to stay on the path!

Potential code blocks:

 setMainLed({r: 90, g:255, b:90})	Changes the color of the main LED light. The text editor uses RGB values!
 await Sound.Category.SoundName.play(true)	Plays a random noise or selected sound. If true, then waits for command to be fully executed to move on to the next step.

COLLABORATE

Trivia



Make a list of trivia questions with. multiple choice answers (a through d). Quiz your students and let them roll their Sphero to the correct answer.

Obstacle Course

Gather cardboard, books, and any other random materials laying around you and create an obstacle course. Race with your friends to see who can finish first.

COLLABORATE

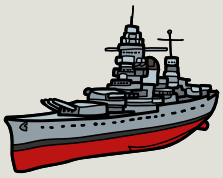
Sphero Sumo

To play this game all you need are plastic cups and painters tape for boundaries. Battle your friends and whoever goes out of the boundary. first loses!

Battle Ship

Make a 10x10 grid on the floor with the columns labeled A-J and the rows labeled 1-10. Print a 10x10 grid for a student/teacher to place their ships on, and let the rest of the students hit targets by rolling their Sphero to a square. Battleship print-out attached to the next page.

BATTLESHIP



	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										

THANK YOU!

Created by: Sonika Tamilarasan
Images used from: ideas.demco.com, Sphero.edu