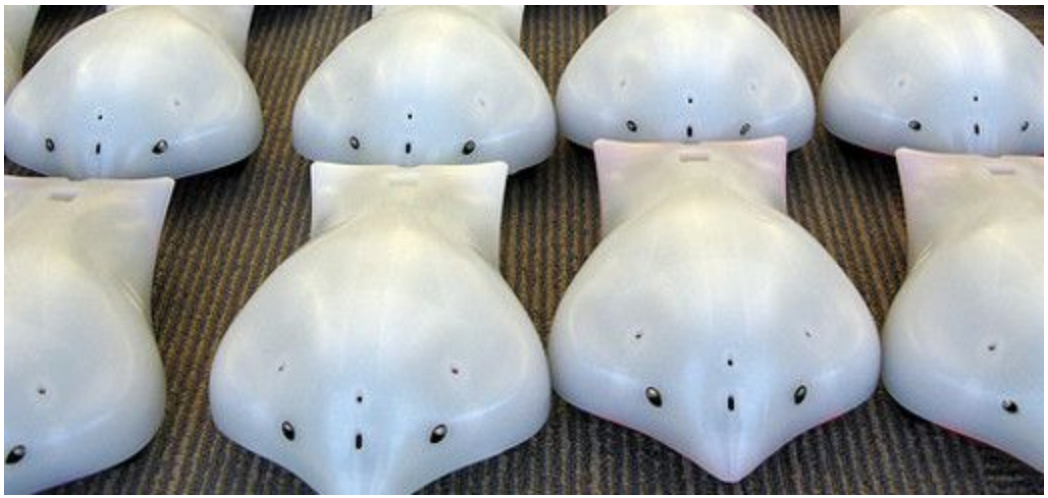
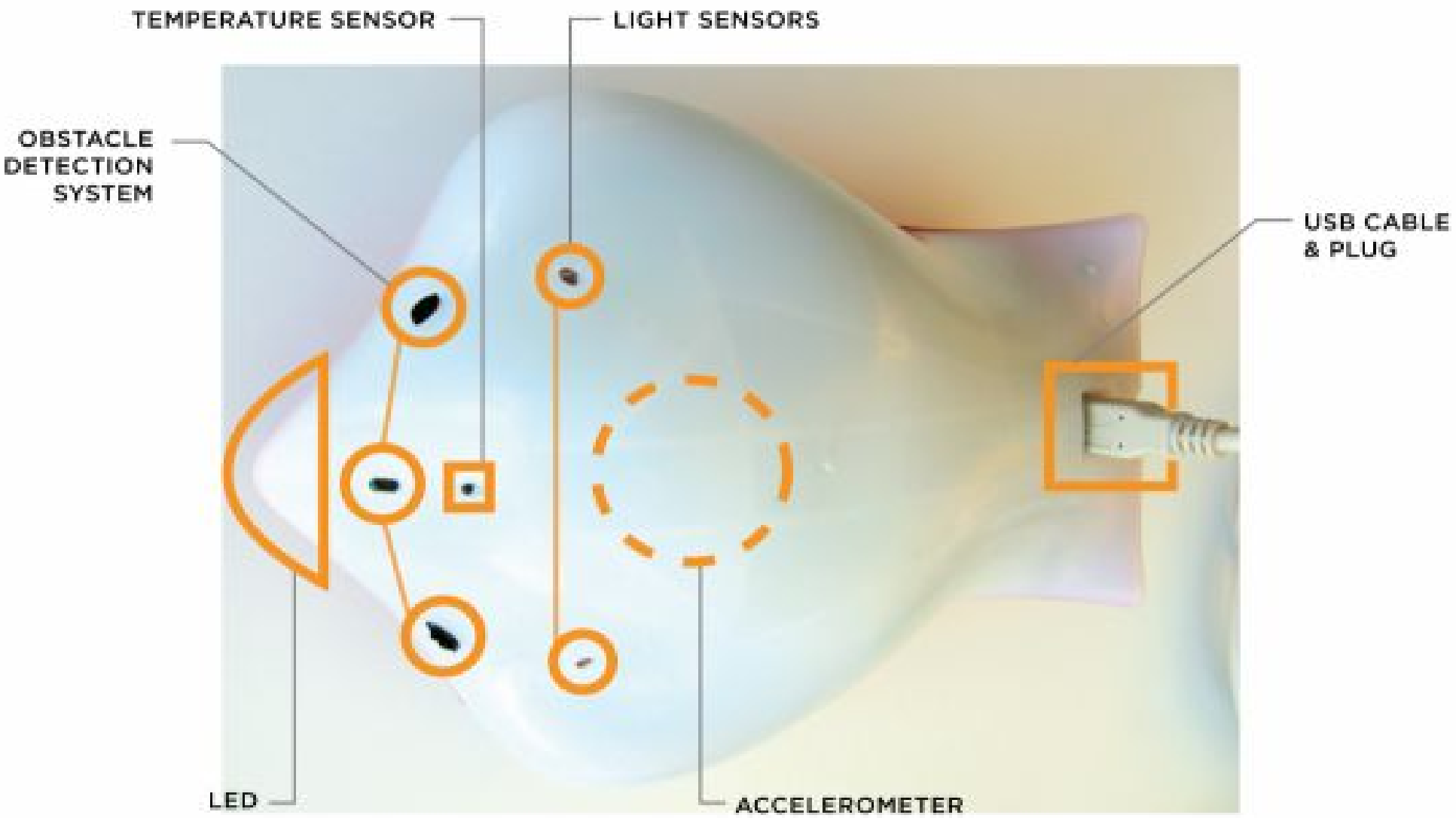
The Finch Robot Challenge Book



The Finch Robot



Things to Remember

- The Finch Robot needs to be connected to the computer with the USB cable at all times.

- You may find that the Finch moves more easily if you gently hold up the USB cable so it doesn't get tangled or caught on anything.

OPENING SCRATCH

Click on the Finch icon on the bottom of your screen.

You will see this ---------->

Make sure it says “Finch Connected”

Click “Open Scratch”

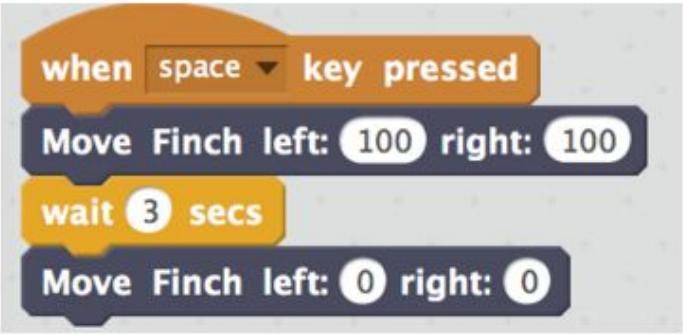
The Scratch blocks for the Finch are located in the More Blocks menu.

MOVING THE FINCH

Sets the power to the left and right wheels. The range is -100 to 100.

Setting the numbers to 0 for both wheels stops the Finch.

MINI CHALLENGE: Try out the program below (the wait block is in the Control menu). It should move the Finch forward for three seconds. Then try several motor speeds between 0 and 100 in the first Move Finch block (keep the speeds of the left and right motors the same). Describe two ways that you can change how far the Finch moves.



What happens when the speed is negative?

What happens when the speed for each wheel is different?

What happens when one wheel is set to 0?

MINI CHALLENGE: Write a program that makes the robot turn a full circle to the left and then a half circle to the right.

COLOR WITH THE FINCH

This sets the color of the Finch's beak. The R, G, and B values control the strength of the red, green, and blue color in the Finch's beak. The range is 0 to 255 for each color.

- How can you make the Finch’s beak turn red?

  R: G: B:

- How can you make the Finch’s beak turn blue?

  R: G: B:

- How can you make the Finch’s beak turn purple?

  R: G: B:

- How can you make the Finch’s beak turn yellow?

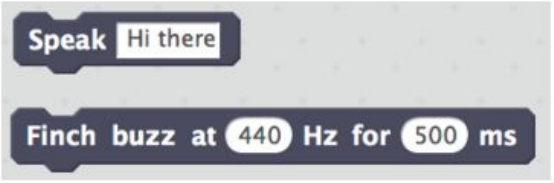
  R: G: B:

MINI CHALLENGE: Make the beak blink on and off with different colors. Do you only see one color? Try including a Wait block.

Where else do you use a Wait block? Why is it important?

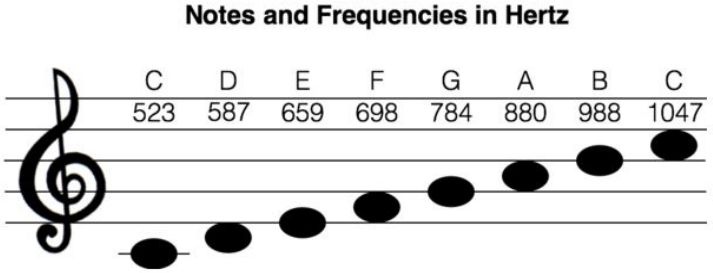
Sounds with the Finch

The More Blocks menu for the Finch contains two sound blocks. The Speak block will cause the computer to speak whatever text you type into the block.



The Finch buzz block activates the Finch's buzzer. This block requires two numbers. The number on the left (Hz) gives the frequency of the sound; keep in mind that humans can only hear sounds in the range of 20 to 20,000 Hz. The number on the right (ms) gives the length of the sound in milliseconds. If you set the buzzer to 262 Hz, and set the duration to 3000 ms, the Finch will play a middle C, which is the C note in the middle of a piano.

MINI CHALLENGE: Can you make the Finch play a tune by inputting different values for the buzzer? Remember to use the Wait block in between notes. Use the chart below for the values of different notes.



Challenge #1:
Finch Dance Party

Combine motion, looks, and sound commands to make the Finch dance. Choose a piece of music at least 20 seconds long (or use part of a larger piece). Then program the Finch to turn and move in steps that match the tempo of the music. Change colors and add some buzzers for a light and sounds effect.

Your Task: Create a program where the Finch performs 6-8 "dance steps." For each step, the Finch should:

1. Move in a new direction
2. Change color
3. Make a different sound

Don't forget: Reset the Finch's wheels to 0 at the end of the dance. Use Wait blocks in between dance steps, color changes, and sound changes.

TRY IT: Use loops to create repetitive movements and color patterns in your dance.

CHALLENGE #2:
THROUGH THE MAZE

Your Task: Write a program that enables the robot to complete a maze autonomously. This means that the robot must move through the maze on its own. You may press a key to start the script, but after that, you may not use the keyboard or move the robot.

Create a maze with masking tape on the floor. The maze should have a start and stop, at least 2 turns, and at least one place with a “change color” sticker.

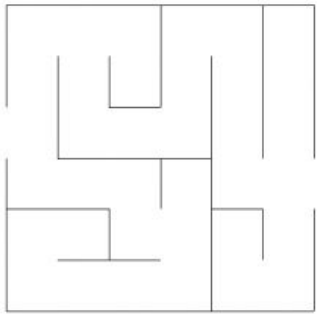
When you start, turn the Finch’s beak to GREEN.

When you complete the maze, turn the Finch’s beak to RED.

When you reach a “change color” sticker, change the beak to any color besides green or red.

Don’t forget: You need a wait block in between moves. Experiment with speeds, turns and wait time to guide the Finch through the maze.

TIME IT! Use a timer to see how fast your Finch gets through the maze.



Challenge #3:
Through the Maze 2

Your Task: Write a program that allows you to control the Finch with the keyboard and guide it through the maze. Decide which arrow key moves the Finch in each direction. For example, maybe you want the robot to move forward when you press the up arrow. What movements will the Finch need to make? Make sure you program a key for each movement.

You will also need to program keys to change the Finch's beak color, one key for each color.

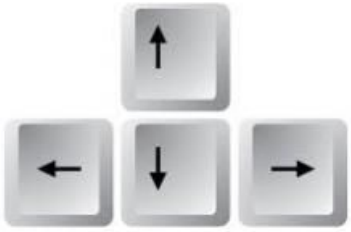
When you start, turn the Finch's beak to GREEN.

When you complete the maze, turn the Finch's beak to RED.

When you reach a "change color" sticker, change the beak to any color besides green or red.

When you complete your scripts, use them to guide the robot through the maze. Once you start, you may not touch the Finch with your hands - control it with the keyboard alone.

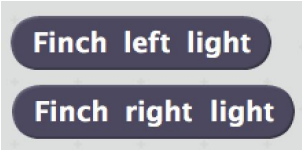
TIME IT! Time how long it takes for the robot to complete the maze via remote control. Which is faster getting the Finch through the maze - the remote control version or the programmed version?



Sensing with the Finch

The Finch has sensors that provide input to the robot. A sensor makes measurements and sends them to the program that the Finch is running. The program can use the sensor information to set outputs or make a decision. The Finch has two light sensors, an acceleration sensor (accelerometer), two obstacle sensors, and a temperature sensor.

The blocks for the Finch are at the bottom of the More Blocks menu in Scratch. The Finch left light and the Finch right light blocks can be used to measure the values of the two light sensors. Each of these blocks has a value from 0 (no light) to 100 (maximum light).



Drag a light sensor block into the Scripts area. Click on the block to see the current value of the sensor. Next, cover the light sensor with your hand and click on the block again. You should see the value of the light sensor change.

What is the value when it's covered? When it's uncovered?

MINI CHALLENGE: Make the Finch's beak light turn on when the light sensor is covered (or when the light in the room is turned off), and have it turn off when the light sensor is uncovered (or the light in the room is turned back on).

Hint: You should check for a range using > and <, instead of an exact value, as the light sensor value can change quickly. You will find these comparison blocks in the Operators menu.

CHALLENGE #4:
"SIMON SAYS"

Your Task: In this program, the Finch will say to turn it in different orientations, and wait to continue until the player makes the right move. For example, the program might say, "Simon says: point beak up." After giving that command, the Finch should wait until the player points the beak up. To reward success, the Finch should blink its beak for 3 seconds and speak the phrase "good job." Then the Finch should give another command. Your game should cover all six possible orientations (you should skip "In Between").

Hint: How do you check for the Finch's orientation?

Check the box next to the Orientation block in the More Blocks menu. Look at its value on screen. Then move the Finch to different orientations to see how it changes.

The value of the Finch Orientation block will be equal to the names of the different orientations, depending on what the Finch is doing. Note that there are underscores between the words in the names of the different orientations and the first letter of each word is capitalized (for example, Beak_Up, Beak_Down).

Challenge #5:
Follow the Light

Your Task: Write a program to make your robot follow a flashlight as quickly as possible. When no light is shining on the robot, it should stop. Combine the sensor blocks with move blocks and blocks from the Operators menu to increase the robot's speed when the light gets stronger.

You can make the speed of the robot depend upon the value of the light sensors using the block shown below.

This block will set the speed of the robot using the values of the light sensors. However, this will only happen once. If you want the speed of the robot to change when the sensor value changes, you need to use a loop. Use a flashlight to try out the program shown below.

Finch moving too slowly? Use math blocks from the Operators menu to speed it up.

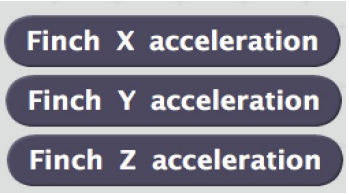
You can use the multiplication block to make the robot move twice as fast.

See how fast you can make the Finch move using this technique with the flashlight. Don't forget to stop moving when the flashlight turns off.

TRY IT: Program your Finch to avoid the light instead of following it.

Challenge #6:
Finch Remote Control

Your Task: Write a program to make your Sprite follow the movements of the Finch.



We will be using the acceleration blocks for this challenge. These blocks measure how much the Finch is tilted (or changing its speed). When the Finch is sitting on a flat surface, these blocks will not change. There are three different blocks because the Finch can measure tilt in three different directions. The value of each block is a decimal number between -1.5 and 1.5.

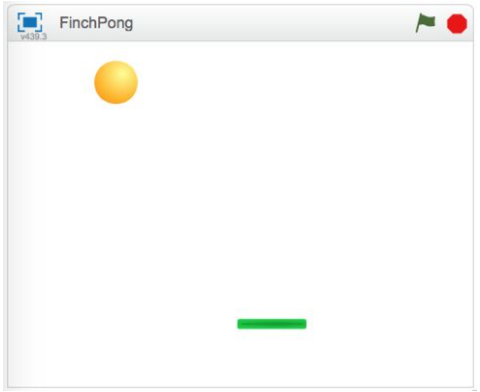
TRY IT: Click on the three acceleration box checkboxes so they appear on screen. Tilt the Finch in different directions. What type of tilt makes each one change?

Ready for the challenge? Use one of the acceleration blocks (X or Y will work best) to move the sprite on the screen. You will need to use a move block with the acceleration block inside it. You will also need to use the arithmetic blocks.

Challenge #7:
Finch Pong

Your Task: Turn the Finch into a game controller! You will create a version of Pong. A ball will fall from the top of the screen, and you will try to catch it with a sprite that you control with the Finch.
To start, you will need two sprites, a ball and a paddle (or a basket, if you prefer). The sprites shown are from Scratch.

Start by writing a script for your ball. You want the ball to start at a random position at the top of the screen. It should then move down the screen. How quickly or slowly it moves it up to you. Test and revise this script to make sure it is working before you move on to the next step.

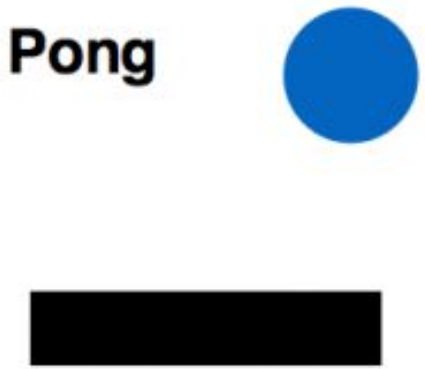


Next, start the script for the paddle. The vertical (*y*) position of the paddle should be fixed. The horizontal (*x*) position of the paddle should be determined by one of the acceleration blocks for the Finch. Test and revise this script until you can use the Finch to move the paddle back and forth on the screen. Now you can move the paddle to catch the ball!

Create a variable to hold the user's score. How will you measure whether the paddle catches the ball? Change your score variable by 1 when this happens.

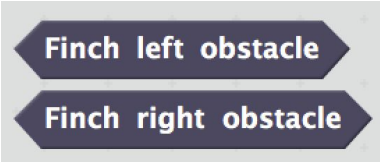
You have created a basic Pong game. Now it is time to make it your own! What features do you want to add? Try adding sounds and graphics to your game. Use your imagination!

Save your Pong program, because we will be coming back to it later to add more features!



Avoiding Obstacles

The Finch obstacle blocks, Finch left obstacle and Finch right obstacle, are Boolean blocks, which means the value of the block is either true or false. If an object is in front of the right obstacle sensor, then the Finch right obstacle block is true. Otherwise, it is false. Each obstacle sensor can detect objects about 1-4 inches from the sensor.

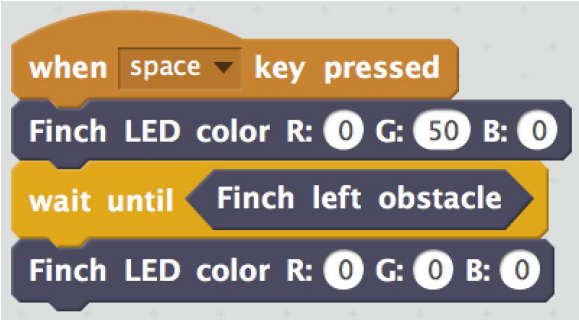


You can use these Boolean blocks in an "if else" block. The "if else" block is very similar to the if block. If the Boolean block is true, the program runs the blocks in the top of the "if else." Otherwise, the program runs the blocks in the bottom of the "if else." For example, the program below turns the Finch's beak red when an object is close to the left sensor. Otherwise, the beak is green.

Try It #1: Write a program that will make the Finch turn right when the left obstacle sensor detects an object. Otherwise, the Finch should move straight ahead.

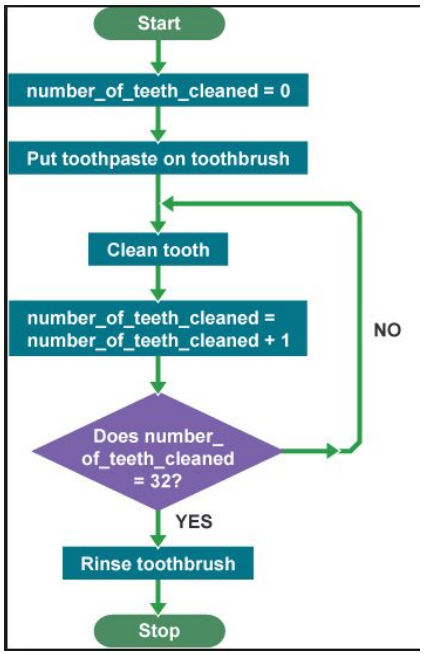
You can also use Boolean blocks inside a wait until block. This block stops the program until the Boolean statement in the

wait until block is true. For example, the program below turns the Finch's beak red and then waits until the left obstacle sensor detects an object. Then it turns the beak off.



Try It #2: What will happen if you run this script when there is already an object next to the left obstacle sensor?

Note: As you start to write programs that make decisions, you will notice that you need to think carefully before you begin to write a program. Planning a program carefully can save you from spending a lot of time debugging. Experienced programmers often draw a diagram (called a flowchart) of how a program will work or write out the steps that the robot will follow (this is called pseudocode).



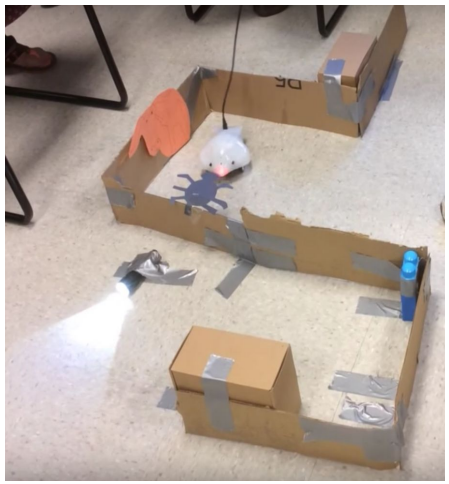
Challenge #8:
Through the Maze 3

Your Task: Write a program that enables the robot to complete a maze autonomously -- but this time, using its sensors. Instead of telling the robot how many seconds to move forward or when to turn, have the Finch stop and turn when it senses an obstacle. You may press a key to start the script, but after that, you may not use the keyboard or move the robot.

You will need a 3-D maze for this to work! Build a maze out of cardboard, paper cups -- anything that will create an obstacle for the Finch.

Can you add features to the maze to help your robot solve it? For instance, adding a shaded area might mean that you can use the light sensors as well as the obstacle sensors.

Test it! Your robot should be able to go through the maze even if you make changes to it. How much could the current maze change before your program would no longer work? What changes can you make to your program to enable it to solve different mazes?



Challenge #9:
SquirrelBot

The SquirrelBot must stay alert for hungry birds of prey. When he senses the dark shadow of a hawk above him, he needs to take random evasive action to confuse the hungry bird. Program four different escape maneuvers to allow your SquirrelBot to survive another day!

Your Task: The requirements for this challenge are as follows:

1. The Finch must detect when a shadow falls on either of the light sensors.
2. When there is no shadow, the Finch should be still.
3. When a shadow is detected, the Finch must make an escape maneuver.
4. The Finch should have at least four possible escape maneuvers.
5. Each time the Finch detects a shadow, it should randomly choose one of the possible escape maneuvers.

Hint: Make a variable called *escapePlan*. Each time the Finch detects a shadow, set this variable equal to a random number between one and the number of escape maneuvers.

Challenge #9:
Finch Pong Plus!

Your Task: Take your Finch Pong game to the next level! Make the game more exciting by adding horizontal direction to the ball, increasing the speed as the player improves, and making the ball bounce when it hits the paddle and the walls.

Adding Bouncing

In your first version of the game, the user just tried to intercept the ball with the paddle. Now you want to make the ball bounce off the paddle and the sides of the screen.

Start by changing the direction of the ball when it hits the paddle. When the ball hits the paddle, you want it to reverse its direction. What block can use you to do this? Where in your program should you place this block?

Next, you want to make the ball bounce off the edges of the screen. You can use use the if on edge, bounce block, which is found near the bottom of the Motion menu. Where in you program should you place this block?

Adding Horizontal Movement

Right now, your ball only bounces back and forth between the top and the bottom of the screen, which isn't very exciting. You need to add some horizontal movement!

Modify your program so that the ball is pointing in the 135° direction when it starts to fall from the top of the screen.

When the ball hits the paddle at an angle, it should turn 90° when it bounces off the paddle. If the ball is falling from the left (135° direction), it should turn 90° to the left. If the ball is falling from the right, it should turn 90° to the right. Make these

changes to your program, and be sure to test that the ball is bouncing properly.

Hint: To find the direction of the ball, you can use the direction variable block at the bottom of the Motion menu.

Changing the Speed of the Ball

Create a variable called *speed*. You will use this variable to change how fast the ball falls. You should place *speed* inside the block you use to move the ball. For instance, if you use a move steps block for the ball, you should place *speed* inside that block.

Remember that you need to set an initial value for *speed* at the beginning of the program. What is an appropriate starting value for *speed*? You want the ball to start slow and speed up when the user is doing well. Add a block to increase *speed* when the user hits the ball with the paddle.

How can you detect that the user has missed the ball? When this happens, you want to reduce the speed of the ball and start a new ball from the top of the screen. You can do this by ending the game and telling the user to restart, or you can give automatically start a new ball when the user misses.

Hint: What is the *y* position of the ball when the user has missed?