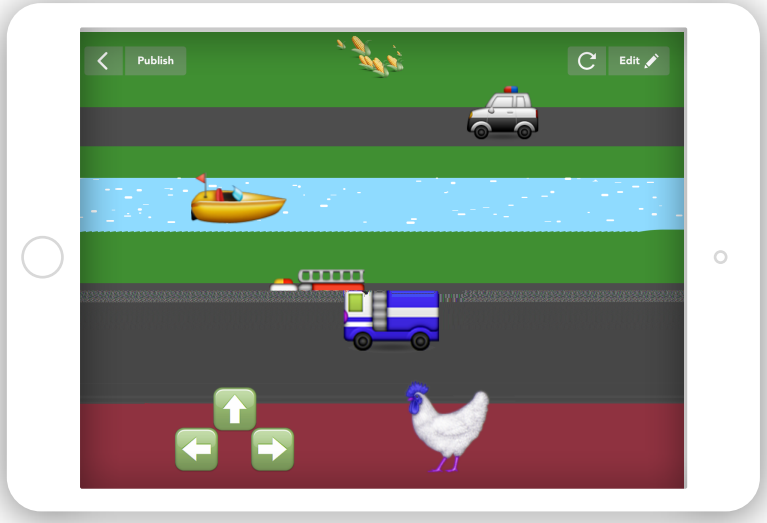
ScratchJr. to Hopscotch

Transition Guide



INTRODUCTION

Hello! This is a resource designed to help facilitate the transition from coding on PBS KIDS ScratchJr to Hopscotch. Use it to familiarize yourself with Hopscotch, and then to help your kids continue their exploration of coding. (Note: you don't have to be familiar with ScratchJr.,—this can serve as an introduction to coding in general through Hopscotch.)

What is Hopscotch?

Hopscotch is a free programming app for iPad and iPhone. With it, you can drag easy-to-read code blocks to write programs. It is completely open-ended, and kids can code whatever they want on Hopscotch, including interactive stories and art, games, puzzles, generative art, and more. Hopscotch also has a community—kids can publish their work for others to play and remix. They can play and look under the hood of other kids' projects. The community inspires, connects, and teaches.

Hopscotch is most popular among kids 7-13, though it's appropriate for beginning coders of any age. It is compatible with iPad 2+, iPhone 4+, and iOS 8+. Hopscotch is available for free from the Apple App Store. Hopscotch is not yet available on Android devices or the browser.

You may code on Hopscotch offline, but to publish or play community projects, you must be online.

Link to download: https://hop.sc/get_hopscotch

Goals

This activity guide will:

- Help you become comfortable facilitating coding on Hopscotch
- Familiarize you with Hopscotch's interface
- Help you learn how to do things you've learned in ScratchJr in Hopscotch
- Introduce you to new coding concepts that extend beyond what you've explored in ScratchJr (don't worry—you've totally got this!)

Key terms and ideas:

This activity guide will build on the key concepts explored through PBS KIDS's ScratchJr Activity Guide, and will focus on the following ideas:

- Sequence - An ordered set of instructions.
- Event - A trigger that a computer recognizes and that causes it to do something
- Loop - A repeating set of instructions
- Value/Variable - A holder for a number. (This is a new concept!)
- Conditional - A command that tells the computer to do something "if" a certain condition is true. (This is a new concept!)

Who is this guide for?

Anyone who wants to learn how to code, or teach others how to code.

While creating this guide, we imagined you being a parent whose children are interested in coding or a teacher/community leader who facilitates coding programs for kids.

But anyone can use this guide to either learn how to code on Hopscotch yourself or to help others do the same.

Format:

The first two activities recreate activities you might have done with PBS KIDS ScratchJr. In these activities, you will revisit concepts, skills, and challenges, and will implement them in Hopscotch.

In these activities, you will also explore Hopscotch and become familiar with the interface. There are side-by-side comparisons to show you how something in PBS KIDS ScratchJr works in Hopscotch.

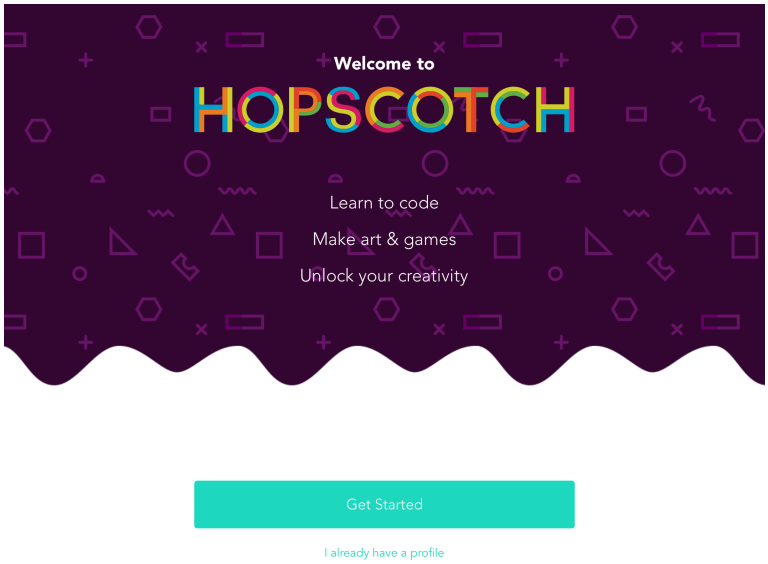
The second two activities will be new! They will introduce you to skills and concepts that you have not previously explored in PBS KIDS ScratchJr and will help you learn to apply them in Hopscotch.

Setting up Hopscotch

It's easy to get started using Hopscotch. For each device you will use, complete the following steps. You can do this ahead of time or walk your class through them on your first day.

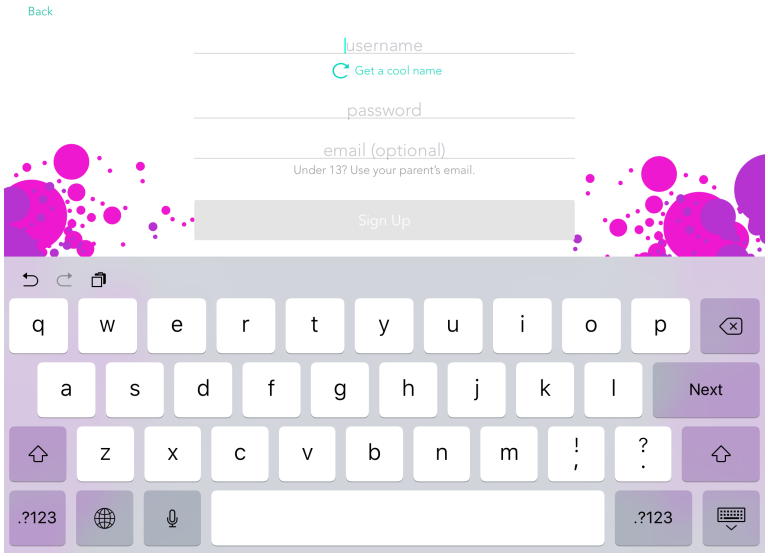
1. Open Hopscotch app

2. Create a new account. (Accounts allow multiple students to share the same iPad, as well as access their work from home.)

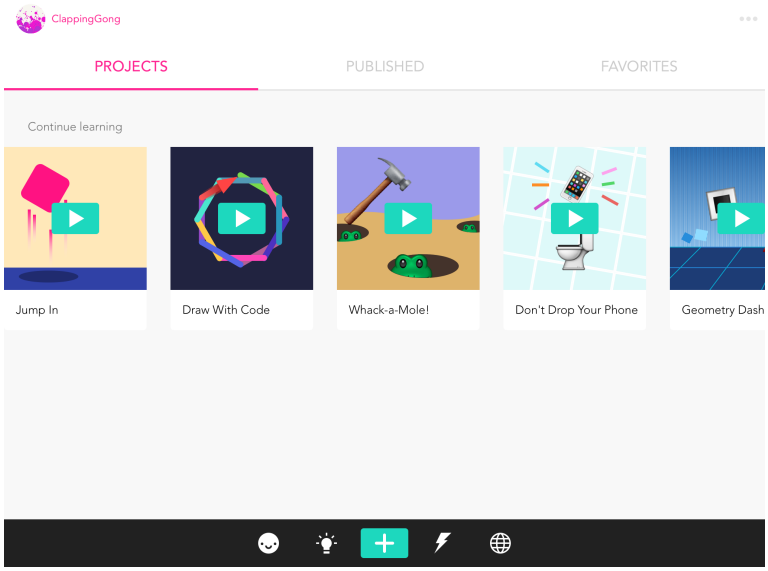


3. Tap the arrow to choose a non-personally identifiable nickname and enter a password. Email is **not** required to use Hopscotch (though it can help you retreive your password in the future).

You can also email Hopscotch to request bulk accounts. (help@gethopscotch.com)



4. Your profile is your homebase in Hopscotch. Get started by tapping one of the first t hree tutorials (Jump In, Draw with Code, or Whack-a-Mole). Follow along to get familiar with the Hopscotch interface. After you're done, you can now access the full app via the bottom nav bar: on the far right you can access the Hopscotch Community. Here, you can play other Hopscotchers' projects and learn from their code. The lightning icon opens your notifications, including when someone has liked or remixed your project. Most importantly, the green plus button allows you to create a new project. Each project you create will appear below the video tutorials.



Lesson 1

Animal Adaptations

LESSON 1

TIME

60 minutes, depending on amount of exploration and share-out time

OVERVIEW

In this activity, your group or class will be challenged to create Hopscotch projects that explore different animals and their unique behaviors and traits. They will also explore the Hopscotch interface and get up to speed on this new programming language.

LEARNING GOALS

Kids will learn how to create projects, add characters, and how to use the programming blocks to make their characters animate and move on screen. They will apply coding and computational thinking practices as they use technology as a tool for creativity, expression and learning with the Hopscotch app.

KEY WORDS

- Programs and Coding
- Sequence/Algorithm

MATERIALS

– 1 iPad or iPhone per student, or 1 device per 2 students, for pair programming
– The Hopscotch app installed on each device

Getting Started (5 minutes)

Take the time to set-up cues, preview what you're going to be doing and why, and to get the group ready and focused for a fun, creative Hopscotch challenge.

Today everyone will be making their own programs using Hopscotch just like they've done with ScratchJr (this is a great starting place for those who haven't done ScratchJr, also).

In this activity, kids will learn how to add characters and how to use the When, Movement, and Looks & Sounds blocks. They will choose an animal and will create a project that shows off what makes that animal special.

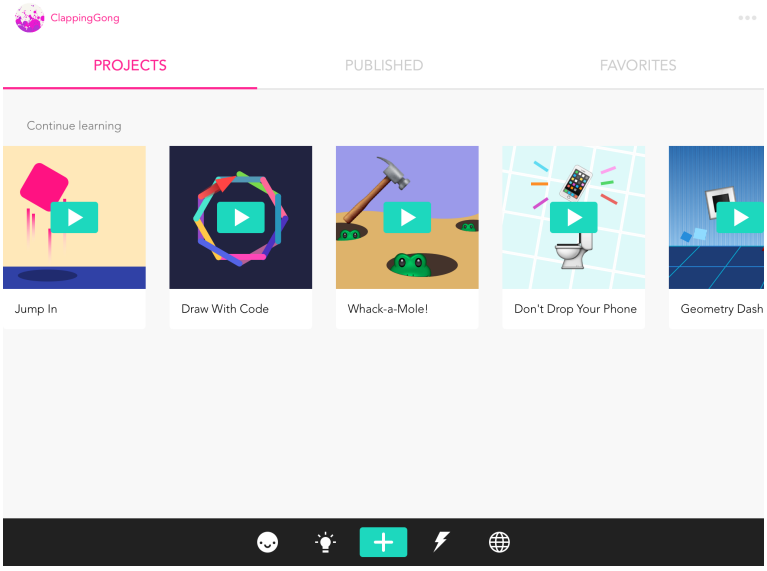
If your group has completed this activity in ScratchJr, remind them of that experience. Ask focusing questions and have each kid answer: What is your favorite wild animal? Why is that animal your favorite? If you have done this activity in ScratchJr, what did you learn?

Before you start, make sure you give the group some free time to openly explore the app on their own. Give them opportunities to share any new and exciting discoveries with the group, and after the lesson, give them time to explore Hopscotch's community.

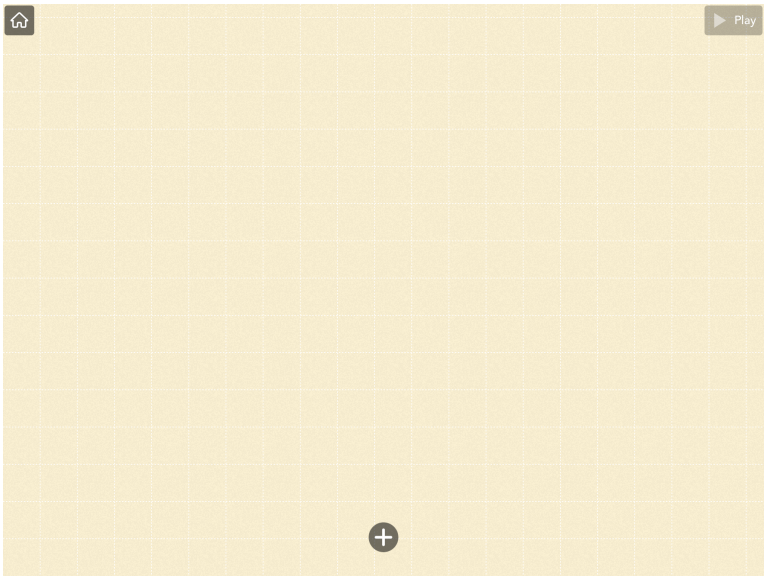
Explore

Model and have the group follow along as you explore how to use Hopscotch. Start with a blank project and demonstrate how to add a new object. Review the range of objects available in Hopscotch (shapes, characters, emojis, and other text).

Go to the editor by tapping the turquoise plus button. If you don't see this button, complete one of the first three video tutorials on your profile (accessed from the smiley face icon) to make the navigation bar appear.



You will see a new blank graph paper canvas with a small plus button at the bottom. Tap the plus button and see the object library:



Each of these are objects, the equivalent of sprites like the Scratch cat in ScratchJr. If you want to delete an object, drag it around the screen and a trash can you can drag it to will appear.

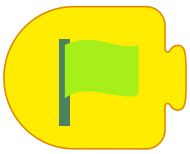
After selecting one, tap "+ Add code." The event library will open up and you can begin coding your object.

Events (pink trigger "when" blocks)

In ScratchJr, the yellow Trigger blocks are what cause (or, trigger) a program to begin. Each of the yellow Trigger blocks represents a different way in which a program can begin. The "Start on Green Flag" block creates a program that will begin whenever the Green Flag is tapped.

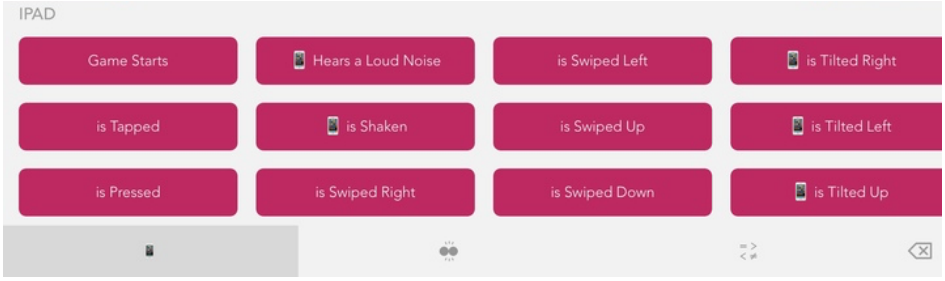
In Hopscotch, the magenta "When" blocks are the triggers! The equivalent of "Start on Green Flag" is "When Game Starts":

Scratch Jr. Hopscotch



Each of the other "When" blocks is a different trigger that initiates code based on some event, like "When the iPad is tapped" or "When an object is swiped."

Partial set of events in Hopscotch (scroll in the app for more):



Code blocks overview

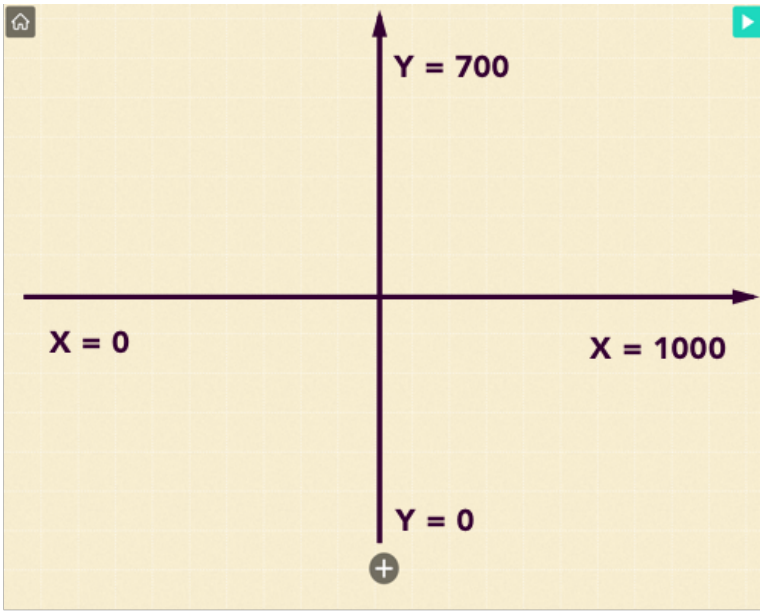
Try pressing play after only adding a when block and see what happens—nothing. That's because you haven't told the object what to do yet by adding additional blocks.

To create a complete program in Hopscotch, combine an event block (a "When") and additional code blocks.

In the exercise explained below, explore and model the blocks that affect objects' movement and appearance. For each block that you explore, create a new rule: an event and the code that follows it. Delete blocks by pressing down and dragging them off the screen.

Movement blocks

In ScratchJr, there is a different block for each direction users might want their objects to go: up, down, left, and right. In Hopscotch, moving an object requires thinking about it in relation to the coordinate plane. If your group is not familiar with the coordinate plane (the "graph paper" on the Hopscotch screen), you might want to quickly introduce them to this concept.



- "Change X by" moves an object across the screen horizontally (along the X-axis).
  - Changing X by a positive number moves an object right
  - Changing X by a negative number moves an object left
- "Change Y by" moves an object across the screen vertically (along the Y-axis).
  - Changing Y by a positive number moves an object up
  - Changing Y by a negative number moves an object down
- Move forward moves an object forward in whatever direction it is facing. For example, if the object is facing the top of the screen, it will move in the that direction.
  - Choosing a positive number moves an object forward
  - Choosing a negative number moves an object backwards

Model these blocks by adding them to a "When" block in the editor. This creates a rule!

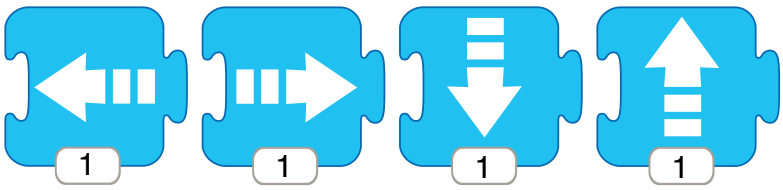
Each time you try a different block, ask your group to predict how it will affect the object. Work with them to make an object go up go, down, left, and right. Show them that the object moves farther when the value of the block is increased. Ask them to share examples of how their animals move around their habitat.

Movement blocks in Hopscotch compared to ScratchJr:

Move

Change X by

Change Y by



Changing direction

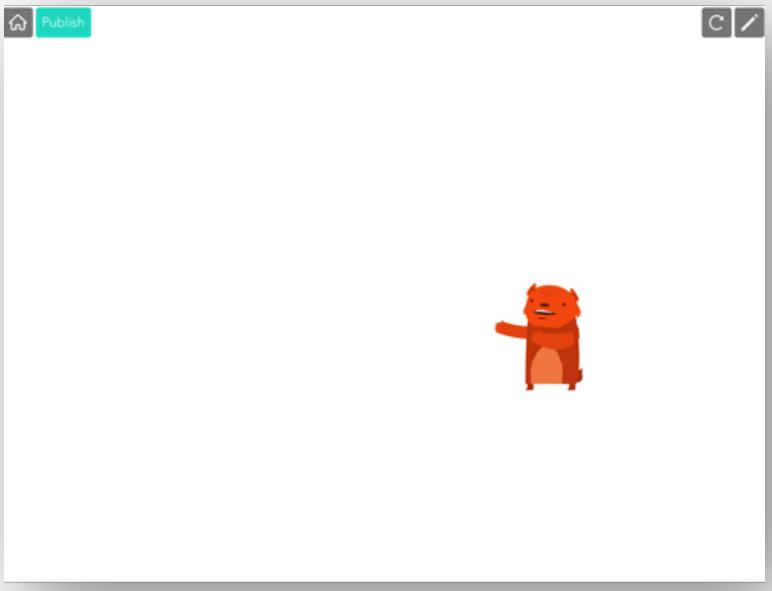
In ScratchJr, kids can change an object's direction by turning it left or right. In Hopscotch, there are three ways to change direction.

Take Bear, whose starting position is standing straight up and down, perpendicular to the screen.

You can make Bear move by:

1. "Flip": Make an object face the opposite direction. If an object is facing right, "Flip" makes it face left. Now look at Bear:

Flip

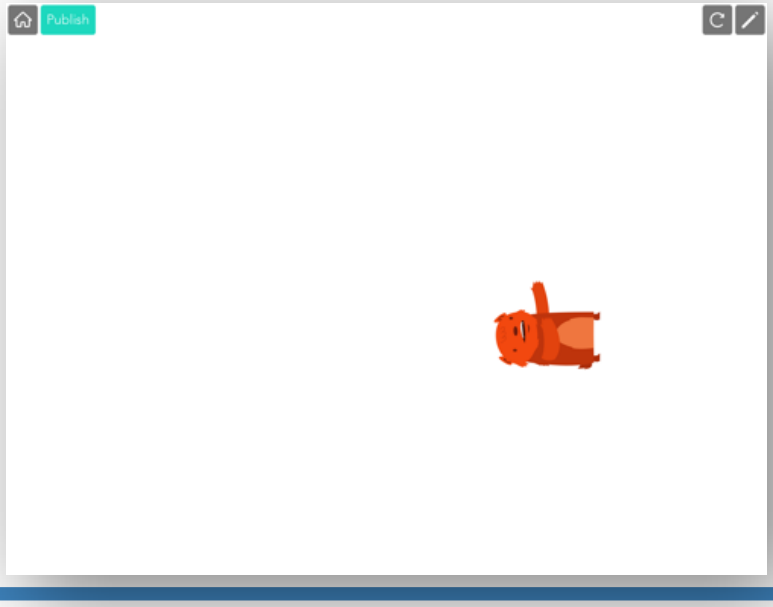


Or,

2. "Turn": Turns an object by a number of degrees. Below, Bear has turned 90 degrees.

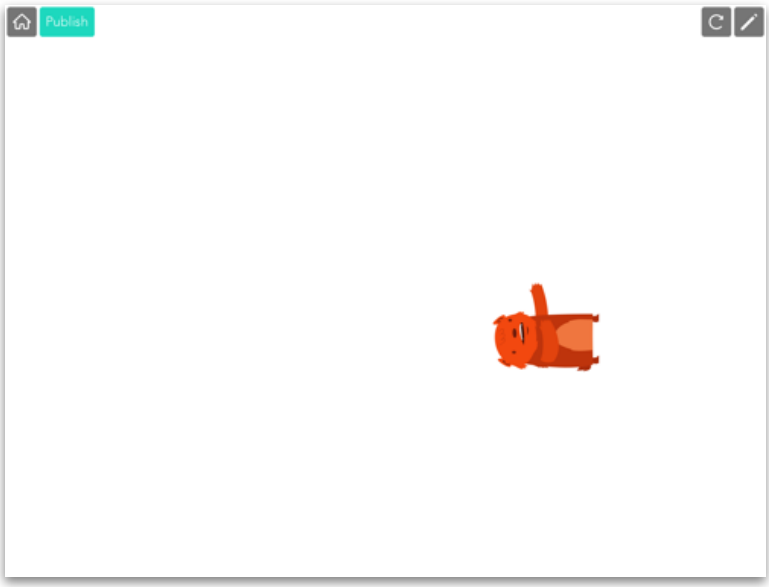
- Turning an object by a positive turns it left (counter-clockwise).
- Turning an object by a negative number turns right (clockwise).

Turn



Or,
3) "Set Angle" rotates an object (in relation to the bottom of the screen). For example, if "Set Angle" is 90, the object will rotate by 90 degrees. Then, if programmed to move forward, the object will move forward from that starting place (up towards the top of the screen). Below, Bear's angle has been set to 90 degrees from the bottom of the screen.

Set Angle



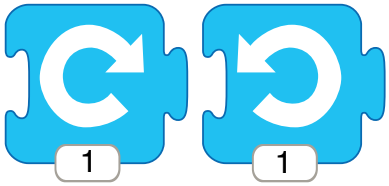
Drag each of these commands into the editor one by one and test the difference. Try using the "When iPad is tapped" and then testing out the "Turn" block. Each time you tap the iPad, the object turns. Ask kids when they might want to make their animal turn or change direction. Try swapping the "Turn" block for a "Set Angle" block and have the group predict what will happen. Then try it out. See if they can explain why "Set Angle" does not cause the object to turn more than once.

These are the turn blocks in Hopscotch and ScratchJr:

Flip

Turn

Set Angle



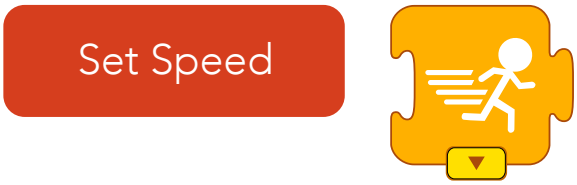
Speed block

In ScratchJr and Hopscotch, there is a block to adjust an object's speed. Without it, the object moves according to a predetermined speed. The "Set Speed" block in Hopscotch changes an object's speed from the initial default speed of 400.

- Choosing a number greater than 400 will make an object move faster
- Choosing a number less than 400 will make an object move slower

Model this block by dragging it above a "Move Forward" block. Play the program, leaving "Set Speed" blank, and then change it to 800 and then 100 to show how the block affects the movement of your object. Ask the group to share examples of when their animals might move faster or slower.

Here's the block in Hopscotch compared to ScratchJr:



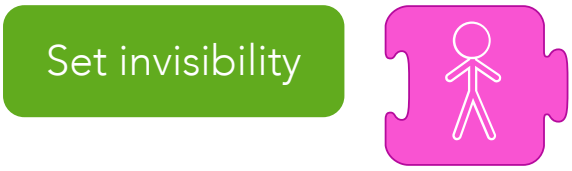
Set Invisibility

Whereas ScratchJr has a separate blocks for showing and hiding characters, Hopscotch has just one block that set an object's invisibility. This green Looks & Sounds block can make characters appear and disappear.

- Setting invisibility to 100 makes an object 100% invisible (hides it).
- Setting invisibility to 0 makes an object 0% invisible (shows it).
- Setting invisibility to somewhere between 0-100 makes an object partially invisible (makes it somewhat transparent). Play around with this one.

Model by dragging the block into the editor and adjusting the value. Ask the group when they think it would be a good time to make an animal visible or invisible.

Here's the block in Hopscotch compared to ScratchJr:



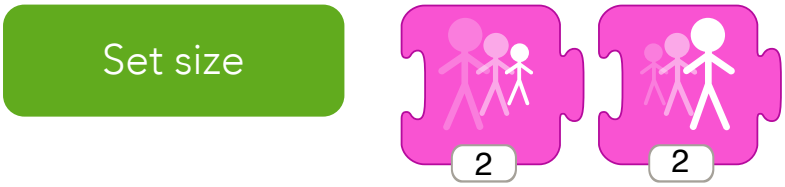
Set Size

As with visibility, ScratchJr has three blocks that adjust the size of a character—one makes it bigger, one makes it smaller, and one resets it. In Hopscotch, there are several blocks that affect an object's size, but we're going to focus on one: "Set Size". This block changes the size of an objects according to percentages.

- Setting size to something greater than 100 makes an object bigger.
- Setting size to something less than 100 makes an object smaller.
- Setting size to 100 returns your object to its normal size (it's 100% normal).

Model this block by dragging into a new rule and adjusting the value. Test what happens when you go from a big number to a small number. Ask your group to brainstorm times when they would want to change the size of their animal.

Here's the block in Hopscotch compared to ScratchJr:



Set Color (*NEW BLOCK*)

Hopscotch has a block called "Set Color" that allows users to change the color of shape objects and trails they draw. This can be helpful if they're using shapes to create other objects or parts of their game (e.g. a wall or clouds).

- Setting color to random means that the object will change color once each time the rule is triggered.
- Setting color to a color will change it to that color.
- Setting color to HSB or RGB allows users to set a precise color value. If your group is older or more advanced, you can explore these concepts. Otherwise, stick with the set color palette for now.

Model this block by dragging into a new rule and adjusting the color. Test what happens when it's inside a "When game starts" versus "When iPad is tapped" block.

Check out this block in Hopscotch:

Hopscotch's color palette:

Encourage the group to freely explore the app and in particular, how to add characters, make them move, show, hide, grow, shrink, and change speed and color. Keep your tablet connected to the projector so that you can continue to provide support.

Create (30 minutes)
Have everyone choose an animal that they would like to use for their project (a Hopscotch character or emoji can suffice!).

The Animal Challenge
Challenge everyone to use all of the new blocks introduced, including Whens, Movement and Looks & Sounds blocks to create a project that shows off something unique/special/awesome about the animal they chose.

- Is their animal really big and that's what protects it from other animals? Encourage using the "Set Size" block.
- Is their animal really fast and that's how it hunts? Or how it escapes being hunted? Encourage using the "Set Speed" block.
- Does the animal blend into its environment using camouflage and only shows itself when it's safe? Encourage playing with the "Set Invisibility" and "Set Color" blocks.

Make sure everyone has a tablet and make yourself available to lend support as everyone works on their projects. Also encourage everyone to turn to each other for support and inspiration.

Make it simpler
Have the group focus exclusively on the "Whens" and "Movement" blocks to make their animal move around the screen.

Share
Have kids pair up and share their projects with one another. Model and encourage giving and receiving feedback. Also encourage the group to share and discuss the following:

- Are there similarities between coding on Hopscotch and coding on ScratchJr?
- Are there differences between coding on Hopscotch and coding on ScratchJr?
- What makes your animal unique and special?
- How do those unique traits help your animal survive in its habitat?

Lesson 2

Build a Jungle

TIME

60 minutes, depending on amount of exploration and share-out time

OVERVIEW

In this activity, kids will explore cause and effect relationships and what plants need to grow and thrive. They will be challenged to create projects in Hopscotch where they make their own plants and trees grow. If your group has completed this challenge in ScratchJr, they will learn how to apply their computational thinking skills in a new programming environment.

LEARNING GOALS

Kids will learn how to use the Hopscotch programming blocks to make animated stories and interactive projects. They will explore coding and computational thinking practices as they utilize technology as a tool for creativity, expression and learning with the Hopscotch app.

KEY WORDS

- Programs and Coding
- Sequence/Algorithm
- Events

MATERIALS

– 1 iPad or iPhone per student, or 1 device per 2 students, for pair programming
– The Hopscotch app installed on each device

Getting Started (5 minutes)

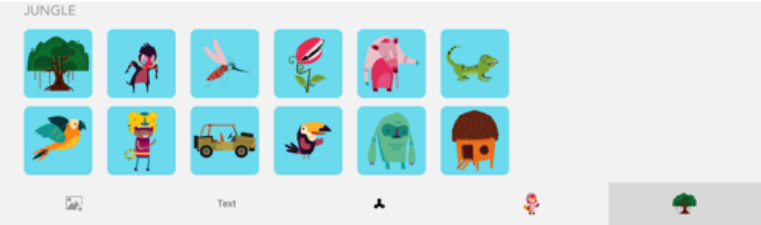
The context in which you are engaging in this activity will impact how you get started. Take the time to set-up cues, preview what you're going to be doing and why, and get the group ready and focused for a fun creative challenge. They will get to use their knowledge of nature to become gardening experts and grow their own digital jungles. Ask a focusing question and have each person answer: If you could create your own jungle, what would live there?

Explore (20 minutes)

Model and have the group follow along as you explore how to use the "Set Size", "When iPad is tapped", "Wait", and other blocks inside Hopscotch by dragging each of them into the editor and discovering what they do. Explore how an object can cause another object to do something (an object responds to an event triggered by another object, like Message Sending in ScratchJr).

Additionally, make sure you give the group some free time to openly explore the app on their own. Give them opportunities to share any new and exciting discoveries with one another.

Check out the jungle characters in Hopscotch (at the far end of the keyboard):

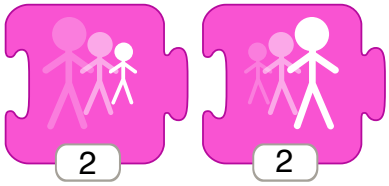


"Grow by" / "Shrink by"

In Hopscotch, like ScratchJr, you can grow or shrink an object. Hopscotch uses percents to determine how much the object's size should change. The larger the number, the greater the change in the object's size.

Model these blocks by dragging them into a new rule and adjusting the value. See what happens when you go from a big number to a small number. Ask your group to brainstorm times when they would want to change the size of a jungle plant or animal.

Here are the blocks in Hopscotch compared to ScratchJr:



"Wait" block

The "Wait" block is a powerful way to control the action in programs. For example, users can have an object pause between running two commands in the same rule (like "Move forward", "Wait", "Turn" around or "Set text", "Wait", and "Set text" in a story). They can also use the "Wait" block to delay the start of a rule, like if they want a rule to be triggered five seconds after the iPad is tapped.

Like in ScratchJr, the "Wait" block in Hopscotch has a time parameter:

- Add the block into a program where you would like a pause.
- Adjust how long the pause is by tapping the number on the block and using the number pad to change it.

Here's the "Wait" block in Hopscotch compared to ScratchJr:

Ask your group when they might want to use "Wait" in their jungle programs.

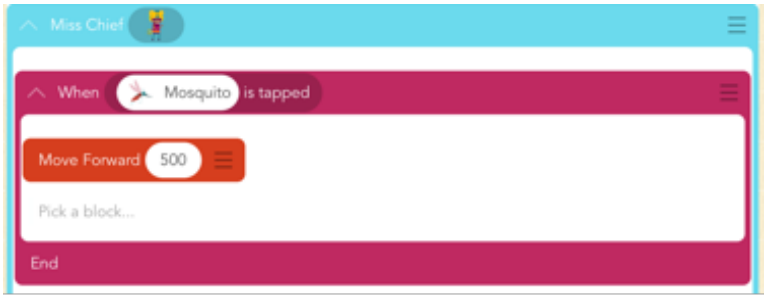
"When Tapped"

"When Tapped" is Hopscotch's equivalent of "Start on Tap" in ScratchJr., and is a powerful alternative to "When game starts" or "Start on Green flag". When using this block, instead of triggering code to automatically run at the outset of the program, users introduce interactivity—something can happen in the course of the program that causes a reaction.

Additionally, in Hopscotch, this event can be triggered in two ways:

- By default, this rule is triggered when the iPad is tapped anywhere (e.g. in FlappyBird, the screen is tapped to keep the bird aloft.)
- You can change the trigger, though, to when a specific object (e.g. a control button) is tapped. Note that in the below example, Miss Chief's rule includes an event triggered by another object (Mosquito).

Here is an example of Miss Chief moving forward when Mosquito is tapped:



Ask your group to think about examples of how "When tapped" could be used, including "When iPad is Tapped" and "When [object] is Tapped".

Interactions between two objects

In ScratchJr, Message Sending blocks trigger events between two objects. There is no set of blocks in Hopscotch that does the exact same thing, but users can use the structure of the Hopscotch language to achieve the same effect. But first, review why message sending or interaction is useful:

Imagine that you wants to make a tree grow every time someone taps on the sun:

- First, you're going to need to add both a sun and a tree to the project as
- objects. (Use a text object and emoji for each.)
- Now start with the tree. Since the tree will react to the trigger, you should add a "When tapped" block to the tree's code. Tap the iPad icon in the when block, and select the sun object instead of the iPad.
- Then add a "Grow by" block.
- Test it out.
- Tapping the sun should cause the tree to grow.

Every time the sun is tapped, it sends a behind-the-scenes message and when the tree receives that message, it grows. In Hopscotch, the objects interact directly, without a message. Users can designate the object to which they want to add the rule (the object that will carry out the action) and the the object that will trigger the event.

Take a look at how direct interaction works in Hopscotch compared to message passing in ScratchJr:

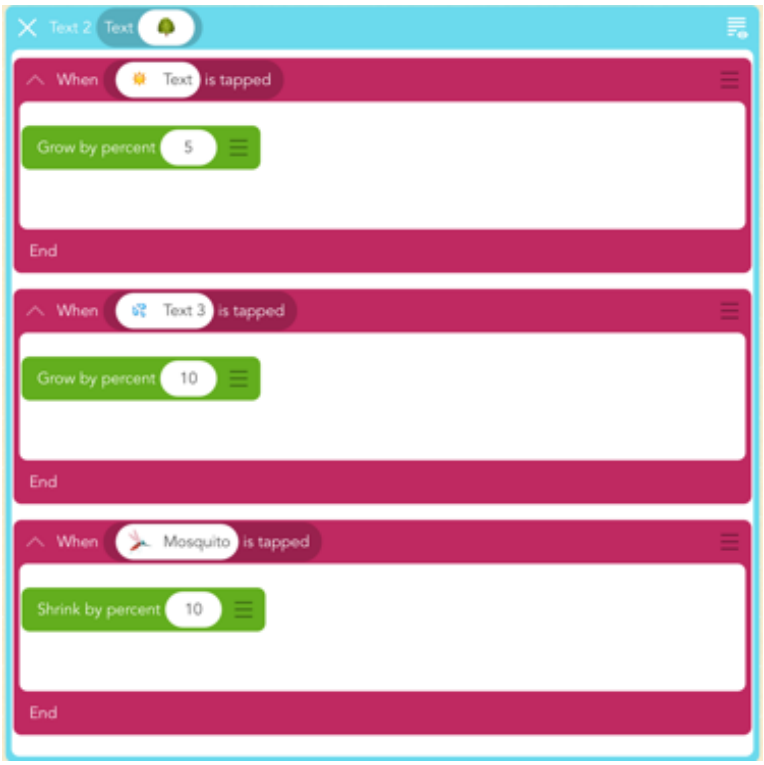
Once you understand this, ask your group to discuss other examples of when message passing or interactivity might be relevant in a jungle project.

Create (30 minutes)

Now it's time to for everyone to make their own jungle projects. Make sure everyone has a tablet to work on (or is working on one tablet in pairs) and make yourself available to lend support as everyone works on their projects. Write the goals clearly at the front of the room.

- Have everyone add jungle objects (including some plants) to their project as well as some other objects that their plant/tree needs to grow. Hopscotch has a special set of jungle characters at the end of the keyboard and emojis can easily stand in for sun, water, etc.
- Challenging option: Encourage the group to use the "When Game Starts", "Set Size", and "Wait" blocks to create projects where exposure to the sun, water, and animals helps plants grow.
    - Encourage them to use the "When Game Starts" block for all of their programs.
    - Encourage them to make add animations for the sun, water, and/or worm characters that trigger right away.
    - Encourage them to make their plants grow after being triggered and after waiting for other objects to complete their animations.
- Super challenging option: Encourage everyone to use the "When [ ] is Tapped and "Grow" blocks to create programs that make their plants grow when other objects are tapped.
    - Tap on the Sun, and the plant should grow. Tap on the water, same thing.
        - Is there anything that can be added and programmed that would make their plant shrink (snow, animals, storms etc.)?

If anyone gets stuck, encourage them to reach out to each other for support. Sample code follows.



Make it simpler

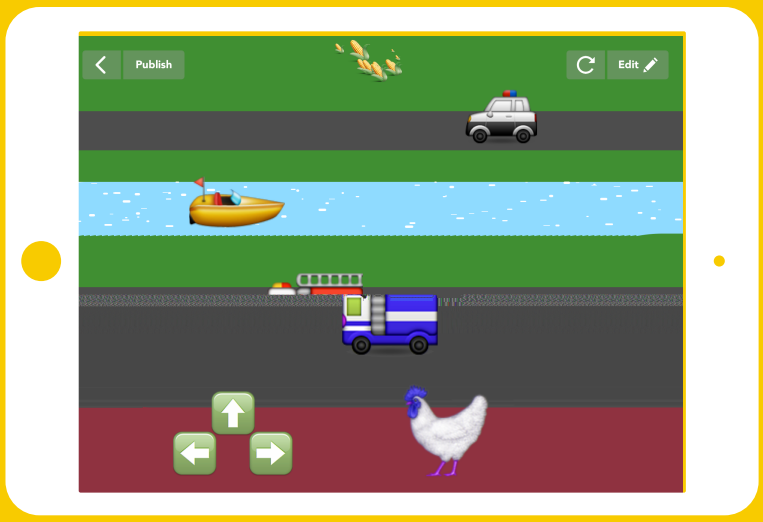
Forget all of the other extra characters and just practice making the plant grow or shrink when it is tapped on.

Share (15 minutes)

Have everyone swap tablets with a partner to check out each other's projects. Encourage them to explore which programming blocks their partner used and why. Additionally, continue to model and encourage giving and receiving feedback. Is there anything they would change about their project after seeing their partner's project? After five minutes get everyone's attention and ask them to share something awesome about their partner's project with the group.

Lesson 3

Cross the Road



LESSON 3

TIME

60 minutes, depending on amount of exploration and share-out time

OVERVIEW

In this activity, your class or group will expand on cause and effect relationships to build a game in which they control a character who is trying to cross a busy street. They will be challenged to create their own version of this game after exploring the main components as a group. This activity introduces them to a new programming concept: loops.

LEARNING GOALS

Kids will learn how to use the Hopscotch programming blocks to make animated stories and interactive projects. They will explore coding and computational thinking practices as they utilize technology as a tool for creativity, expression and learning with the Hopscotch app. They will be able to execute their own ideas after learning new coding concepts as a group.

KEY WORDS

- Programs and Coding
- Sequence/Algorithm
- Loops
- Collisions

MATERIALS

– 1 iPad or iPhone per student, or 1 device per 2 students, for pair programming
– The Hopscotch app installed on each device

Getting Started (5 minutes)

The context in which you are engaging in this activity will impact how you get started. Take the time to set-up cues, preview what you're going to be doing and why, and get the group ready and focused for a fun creative challenge. They will apply their computational thinking skills to create an interactive game where they lead a character across a busy street. Ask a focusing question: If you were to guide someone across a street, what kind of directions might you give? (Go, stop, left, right, look for cars coming).

Explore (20 minutes)

Model and have the group follow along as you explore how to use "Set invisibility", "Set text", "Repeat forever", "When bumps", and "When [] is tapped" inside Hopscotch by dragging each of them into the editor and discovering what they do.

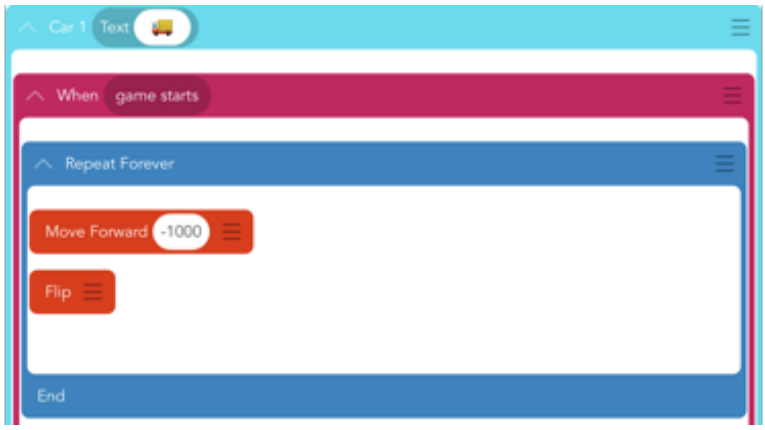
Additionally, make sure you give everyone some free time to openly explore the app on their own. Give them opportunities to share any new and exciting discoveries with the group.

Repeat and Repeat forever

Like ScratchJr, Hopscotch has two loop blocks: "Repeat" and "Repeat forever". Take this opportunity to introduce (or reintroduce) the group to the idea of a programming loop: code that repeats. Ask them to describe loops in their everyday lives (for example, making the same sandwich ten times, a stoplight).

- "Repeat" will repeat the the code inside of the block according to the number (parameter) the user sets. It will run the code from top to bottom, and then start again at the top.
- "Repeat forever" will repeat the the code inside of the block until the program stops. It will run the code from top to bottom, and then start again at the top.

Try putting a block below "Repeat forever" but within the same rule (e.g. above the bottom of the magenta when block). Ask the group to predict what will happen. Test it out. Ask the group to figure out, in pairs, why that code never runs. Then switch the "Repeat forever" to a "Repeat" block and see what happens. How might each of these blocks be useful in creating a game where objects are driving across a screen? (See sample code below.)

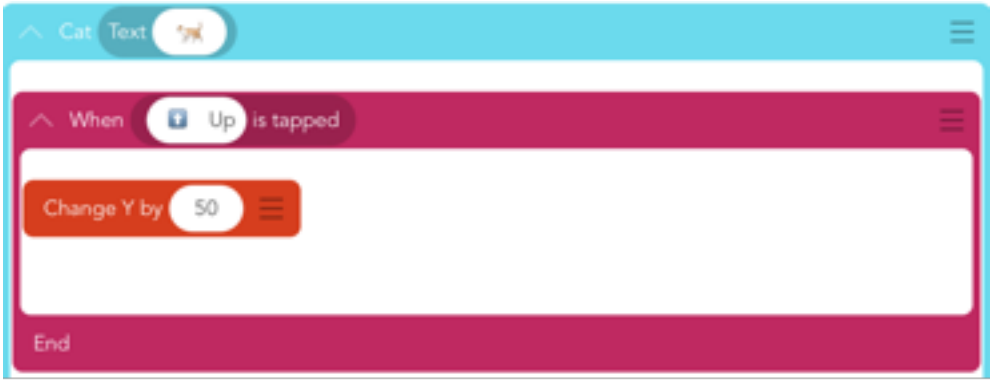


"Repeat" and "Repeat forever" in Hopscotch compared to ScratchJr:

When tapped as Message Sending

In ScratchJr, "Message sending" uses one character to trigger a reaction in another. This same effect can be achieved in Hopscotch by adding a rule to one object with an event that is triggered by another object. This sounds a little tricky, but when you test it out, it's actually very simple.

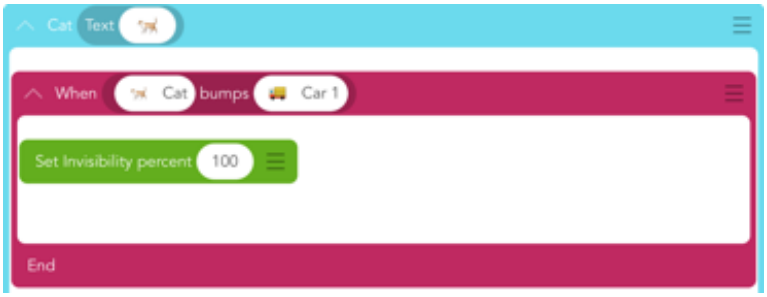
Try modeling this for your group by adding a direction arrow emoji (up, down, left, or right). Then create a rule for your character with "When [arrow] is tapped" and "Move forward". Test it out; when you tap the arrow, the object moves. Ask everyone how to create the same effect for other directions. What combinations of objects, Whens, and code blocks would they need? When might they need this in their Road Crosser games?



Bumps

Like ScratchJr, in Hopscotch users can trigger an event when two objects "bump" into, or collide with, one another. Note that they need to separately program the two objects to move towards one another in order for them to "bump".

Model this block by dragging it into the editor and, using the code you have established previously for the direction pad and repeating cars, create a collision between a car and your character.



Start Sound

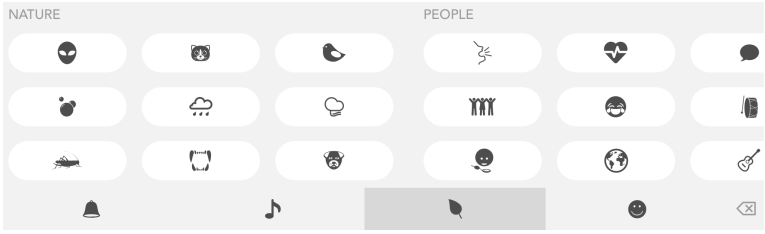
Sounds effects and music bring projects to life. Hopscotch has a library of musical notes and sound effects that can be used in any combination. Unlike ScratchJr, you cannot record your own sounds in Hopscotch.

To add sounds to a project, use the "Start sound" block, choose the desired sound effect, and set the "Wait" value. The number entered in "Wait" will determine how long the code waits between before playing the next note or sound. The larger the number, the greater the time between the two sounds.

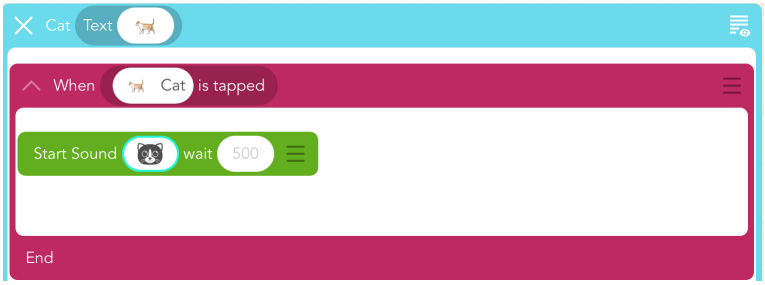
Model this block by adding it to your code, and then testing out the different notes and sends. Try adding several "Start sound" blocks in a row and adjusting the "Wait" values. Ask your class to guess how the sound effect will change each time you adjust the "Wait" value.

"Start sound" in Hopscotch compared to the play and record sound blocks in ScratchJr:

A sample of Hopscotch's sound library:



"Start sound" in code:



Set Text

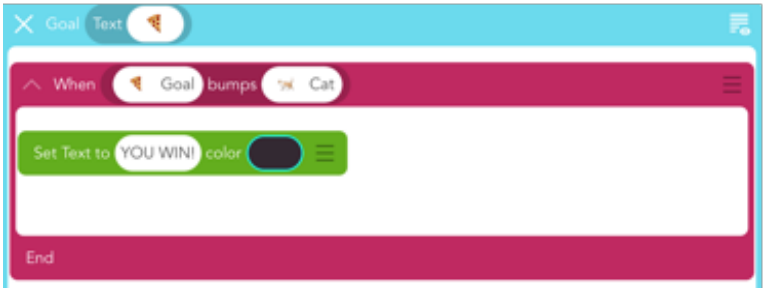
This helpful Looks & Sounds block can be used to turn any object into text.

- Use "Set text" and change any object into text or emoji
- Use "Set text" several times in the same sequence to display a long or changing idea (add a "Wait" between "Set text" blocks to make text more readable)

Model this block by adding a goal object that your character wants to reach at the top of the screen. When the character reaches the goal, set the goal's text to be something congratulatory.

Set text block in Hopscotch:

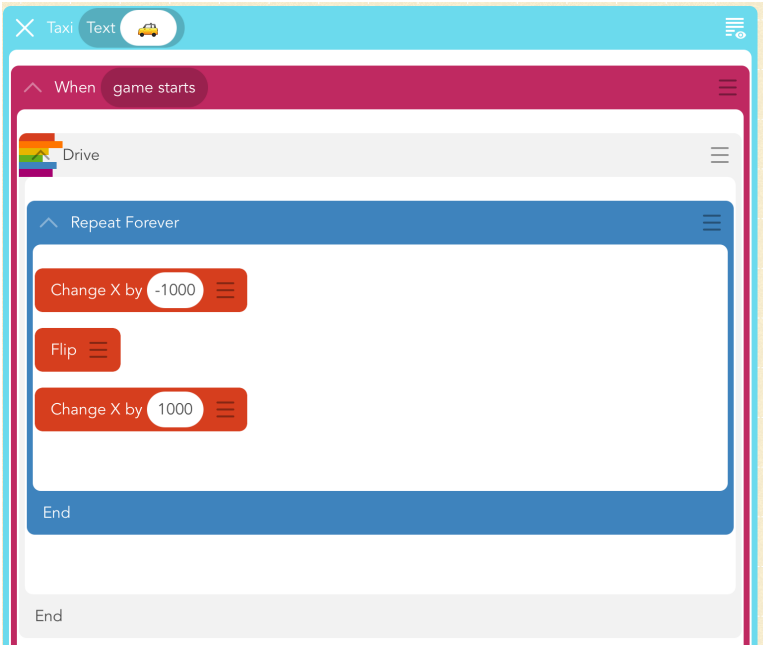
And in context:



Create (30 minutes)

Now it's time to for everyone to make their own Road Crosser projects. Make sure everyone has a device to work on (or is working on one device in pairs) and make yourself available to lend support as everyone works on their projects. Write the goals clearly at the front of the room.

- Have everyone program a character that must cross the road, as well as the controls to move it and cars to get in its way.
- Challenge option: Add gameplay elements that make the project exciting and challenging for the player.
    - Encourage the group to use "Repeat forever" to make a single car drive back and forth forever.
    - Encourage them to use "Bumps", "Set invisibility," and "Set text" to create a lose state, or the result of the character colliding with the car.
    - Add more cars, program each to move, and create additional "Bumps" rules for them, too. (Note that naming cars well can make code easier to read.)
        - Create a custom block for the cars' movement. Custom blocks are a group of code (a function) that can be applied to any block. They save the user the hassle of rewriting code more than once. Note that if they change the code inside a custom block in one place, it will change the code everywhere.
        - Here's an example:



- Super challenge option: Make the cars' movement random. Use "Set speed" and the "Random" math operator to change how fast they go. Ask the group to discuss what random means in a programming context (the lack of a pattern). The "Range" is the lowest and highest numbers that Random can choose from.

If anyone gets stuck, encourage them to reach out to each other for support. Or, if you have Wifi, check out this sample project: hop.sc/crossyroadproject. You can also search the community for "Crossy Road" to see other implementations.

Make it simpler

Program a character to get past a single car.

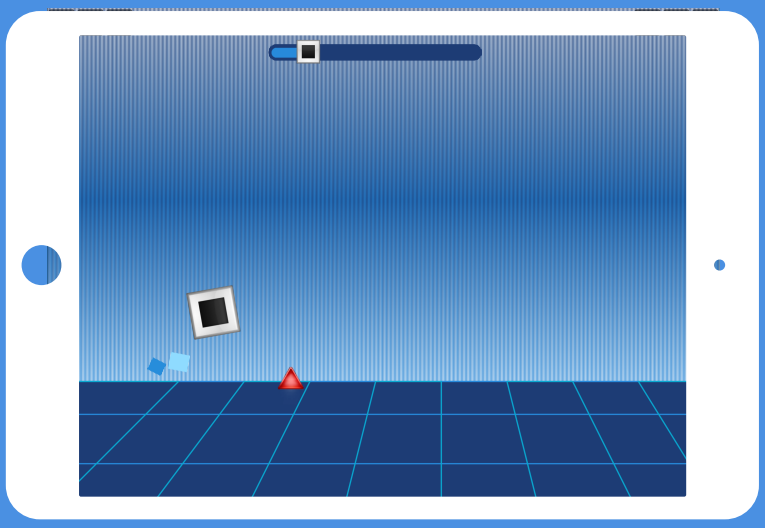
Share (15 minutes)

Have everyone swap tablets with a partner to check out each other's projects. Encourage them to explore which programming blocks their partner used and why. Additionally, continue to model and encourage giving and receiving feedback. Is there anything they would change about their project after seeing their partner's project? After five minutes, get everyone's attention and ask them to share something awesome about their partner's project with the group.

- What is a loop?
- What's an example of a loop in your regular life? In programming?
- What is Random?
- What's an example of random in your regular life? In programming?

Lesson 4

Geometry Jumper



LESSON 4

TIME

60 minutes, depending on amount of exploration and share-out time

OVERVIEW

In this activity, kids will explore conditionals (“if/then” statements), and variables (numbers that can change programmatically). They will be challenged to create their own game in which an object jumps over fast-moving obstacles after exploring the major themes and elements as a group.

LEARNING GOALS

Kids will learn how to use the Hopscotch programming blocks to make interactive games. They will understand that they can program a computer to make choices and “think”. They will be able to apply what they learn to create a game in Hopscotch.

KEY WORDS

- Programs and Coding
- Variables / Values
- Conditionals

MATERIALS

– 1 iPad or iPhone per student, or 1 device per 2 students, for pair programming
– The Hopscotch app installed on each device

Getting Started (5 minutes)

The context in which you are engaging in this activity will impact how you get started. Take the time to set-up cues, preview what you're going to be doing and why, and to get the group ready and focused for a fun creative challenge. Today, they will apply their computational thinking skills to create an interactive game where their character jumps over a fast-moving object. Describe the game premise, and ask the group to discuss the rules they might need to create in pairs. (Character can jump; obstacle can move; if obstacle and character collide, they lose.) Games like Super Mario, Geometry Dash, and Tiny Wings are examples of this mechanism.

Explore (20 minutes)

Model and have the group follow along as you explore how to use "Set position", "Leave a trail", "Set value," "Increase value," and "Check once if" inside Hopscotch by dragging each of them into the programming area and discovering what they do.

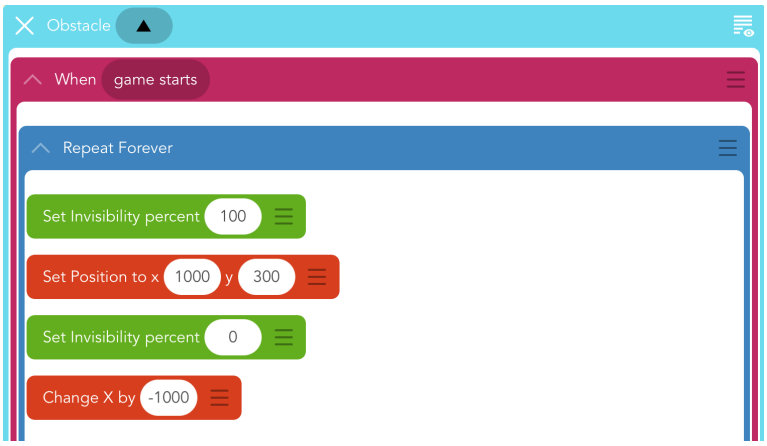
Additionally, make sure you give everyone some free time to openly explore the app on their own. Give them opportunities to share any new and exciting discoveries with the group.

Set Position

In Hopscotch, users can determine an object's starting position by dragging it wherever they want on the graph paper screen (the "stage"). But they can also programmatically assign it a position using the red movement block, "Set position."

Model this block in Hopscotch by dragging an object that will be used as an obstacle onto the stage. Another way to make one obstacle look like many is to reset its position after it travels across the screen. (Pairing this with "Set invisibility" makes the effect even more powerful.)

The following sample code can be used to demonstrate the effect:

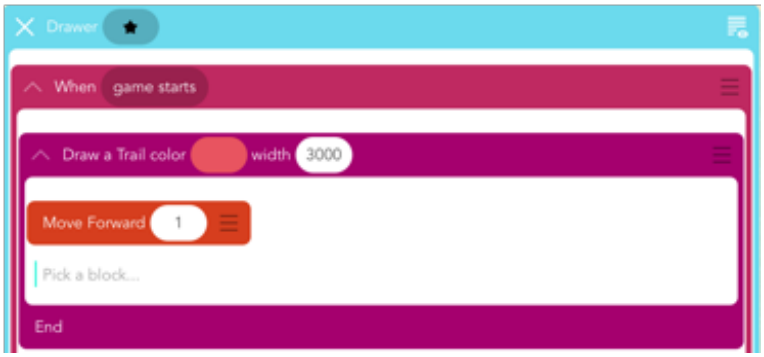


Draw a trail

Whereas ScratchJr has a range of fun background choices, Hopscotch leaves the background up to the programmer. A simple way to create a background is the purple "Draw a trail" block in "Drawing". This block essentially draws a colored path behind it as it moves around the screen and the programmer decides its color, width, and speed and path of creation.

Model this block by dragging it into the editor and adjusting each of these components. To make a background, set the width to 3000 and the distance moved to 1.

In context:



Values

Values, also known as variables, hold pieces of information. Values can be set, changed, or checked. They can be used inside events and other blocks, and can stand in any place where a number is used. What makes values so powerful is that they can actually be changed programmatically.

Begin modeling values by asking the group to describe examples of values in their lives, such as age. Then, drag out a "Set value" block into the editor and set it to zero. In a new rule, set the value to increase when the iPad is tapped. Demonstrate how each time you tap the iPad, the value goes up.

Value blocks in Hopscotch:

Set Value

Increase Value

As a group, brainstorm some ways that values could be used in the game. Try using values to keep track of lives. The below sample code shows how to start a character with three lives. Each time the character collides with the obstacle, the "Lives" value should decrease (increase by a negative number). Note that you can use a separate text object to constantly display the number of lives left. (You can set text to a value by tapping the toggle next to the turquoise check mark.)

Sample value code:

Check once if

The "Check once if" block is a conditional—one of the most powerful blocks in Hopscotch. Conditionals are statements of the form "IF (something is true) THEN (do an action)". It executes code only under the condition specified, like IF the score is greater than 10 or IF a character is invisible. We use conditionals in our lives all the time. Stoplights are a great example of conditions: "If the light is green, then go." They are also extremely important in programming since computers need explicit directions.

There are two types of conditionals in Hopscotch: "Check Once If" is used when there are only two possibilities: 1. If the condition is true, do something, and 2. If the condition is not true, move on. "Check Once…Else", by contrast, lets you check two things before moving on: 1. If something is true, do something, and 2. If this thing is NOT true, do something else, then move on. For example, say you can go in the ocean only if you're wearing sunscreen or else you will have to put some on first. If you are wearing sunscreen (the condition is true), then you may go into the ocean. If you are not wearing sunscreen (the condition is false), then you must put some on first.

Model these blocks and discuss the difference between the two types of conditionals. When might one be better than the other?

In this game, the group can use a conditional to check if the game is still in play; that is, if the character's lives have not expired. On the jumping character, forever check if the value "Lives" is less than 1. If yes, end the game (by setting invisibility to 0, turning into a flame emoji, or making the whole screen go black). Else, do nothing.

Here's an example:

Create (30 minutes)

Now it's time to for everyone to make their own Geometry Jumper games. Make sure everyone has a device to work on (or is working on one device in pairs) and make yourself available to lend support as everyone works on their projects. Write the goals clearly at the front of the room.

- Have everyone program a character that jumps up when you tap on the iPad and an obstacle that moves across the screen towards it. When the obstacle collides with the jumping character, the game ends.
- Challenge option: Add gameplay elements that make the project exciting and challenging for the player:
    - Encourage the group to use "Set position," "Set invisibility", and "Repeat forever" to make their obstacle's movement more interesting. They can even use "Set Speed" and "Random"!
    - Encourage them to draw a background that makes the game visually appealing.
- Super challenge option: Introduce the concepts of lives by creating a "Lives" value and decreasing it when the jumper and obstacle collide. Display the current number of lives using a text object.
    - They can also "Check once if, Else" to check the game's score and end the game at the appropriate moment.

If anyone gets stuck, encourage them to reach out to each other for support. If you are online, you can also check out a sample project: https://hop.sc/geometrydashproject or search "Geometry Dash" in the community.

Make it simpler

Program a character that jumps when you tap the iPad and an obstacle that moves repeatedly across the screen.

Sample jumper code:

Share (15 minutes)

Have everyone swap devices with a partner to check out each other's projects. Encourage them to explore which programming blocks their partner used and why. Additionally, continue to model and encourage giving and receiving feedback. Is there anything they would change about their project after seeing their partner's project? After 5 minutes get everyone's attention and ask them to share something awesome about their partner's project with the group.

- What are conditionals?
- What is an example of a conditional in your regular life? In programming?
- What are values? Why are they useful?
- How do you use values in programming?
- How do you create a background in Hopscotch?