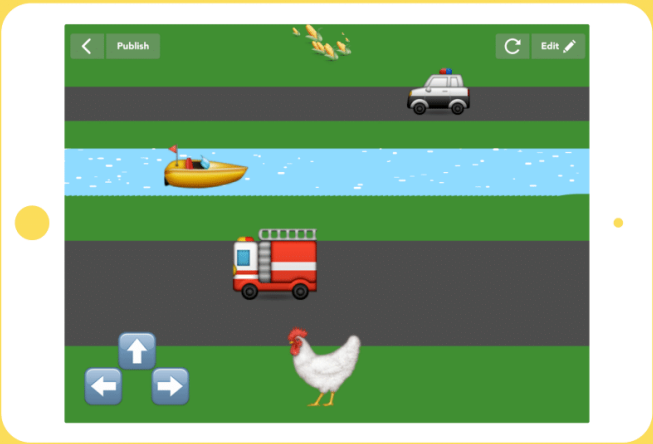
HOPSCOTCH

Crossy Road



Teacher notes for student-led tutorial

TIME

45 minutes, or 60 if you include
15 minutes of free code time

BIG IDEA

If you can code, you can make things that you like and use, and that may not have existed before. Coding is a superpower!

SKILL FOCUS

- Using Hopscotch
- CCSS.MATH.PRACTICE.MP5 Use appropriate tools strategically

KEY VOCABULARY

Event: When something happens
Sequence: A list of instructions, in order
Loop: Code that repeats
Random: The lack of a pattern
Range: The lowest and highest numbers that *Random* can choose from

TRANSFER GOALS

1. Students will understand that coding means telling computers what to do, and can think of some things that are made with code (Apps, car software, medical equipment).
2. Students will be familiar with how to use the Hopscotch app to create projects, add objects, and write and edit code.
3. Students will be able to name two Hopscotch Events and understand that an Event is "when something happens".
4. Students will understand that a loop is code that repeats, and be able to see loops in their daily life.

MATERIALS

– 1 iPad or iPhone per student, or 1 device per 2 students, for pair programming
– Video available on YouTube: https://youtu.be/nLICdNRz5Ig
– Complete project available: http://hop.sc/TrafficDodgersProject

TEACHER BRIEF

Hi!

We're really excited that, this Hour of Code, you're programming with your students—both for them and for you. Kids have remarkable imaginations, and creating computer programs is an amazing way for them to express themselves. We've seen kids create astonishing things using our simple but powerful tool. We know you'll see the same when using Hopscotch, and hope you share what your students create.

Anyone, regardless of their experience in programming, can teach this Hour of Code lesson. Just as Hopscotch was built on the principle that anyone can become a great programmer, this lesson is designed on the premise that anyone can teach basic programming, including you!

In this lesson, students will build a game in which their character runs across a road with fast-moving obstacles. It's simple but fun, and very quickly allows them to experience the satisfaction of telling their computer what to do.

You can teach this Hour of Code in several ways:

1. Students independently complete their games, following along with the video tutorial in the app. The tutorial is designed to be used without any outside help (though we encourage kids to pause it as they're coding). The tutorial is available at https://youtu.be/nLICdNRz5Ig.
2. Students independently complete their games but you ask them to pause their own work for discussion and group work. We offer discussion ideas, as well as differentiation techniques and reflection questions, in the following pages.
3. You project the video to the class and use it as a supplement to your instruction. You lead discussion and group work, and adjust the directions for the project.

After a discussion of what needs to be built and, if desired, how it might be coded, students can start coding. Depending on how many devices you have, you can have students work independently or in pairs. At Hopscotch, we do a lot of pair programming (two programmers share one computer) because it helps us write smarter, less-buggy code. We recommend trying it! All students should get into the habit of testing their code frequently by running (playing) it. It is much easier to find and solve mistakes when you're constantly testing.

Have fun and we can't wait to see what your students build. Share their projects on social media and tag us either with #madeonhopscotch or @hopscotch on Twitter and @gethopscotch on Instagram :)

Yours,
Jocelyn Leavitt
Co-founder and CEO, Hopscotch

LESSON

0. Discussion (5 minutes)
The first and most important lesson of computer science is that computers do what they are told, and only what they are told, in the order they are told to do it.

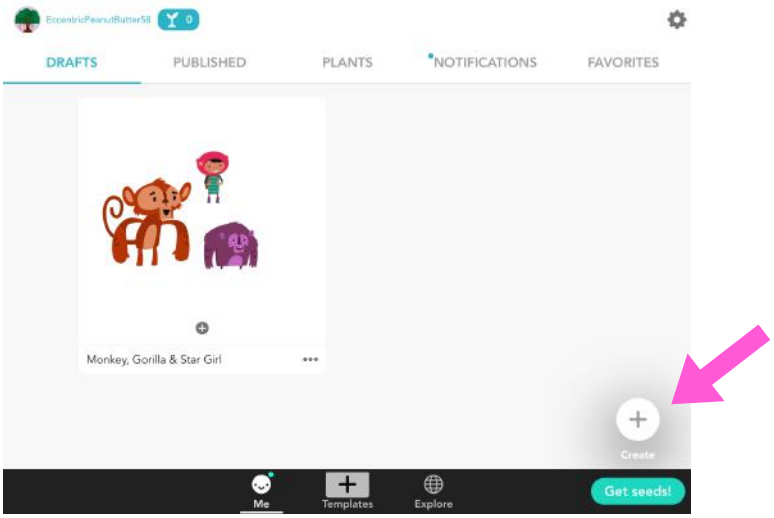
If you fully understand this concept and begin to think of everyday processes (making a sandwich, getting to school) as a set of instructions, you will begin to think like a programmer without trying very hard! A programmer is a person who codes, or writes computer programs. A program is a set of instructions a computer can understand. We refer to these instructions as a **sequence**. This term also refers to the idea that computers must follow the instructions in the order, or in the "sequence", in which they're given.

Ask your students to name some programs they use. Consider all their games and apps, but also the software a DJ uses to mix tracks, the database your doctor uses to keep track of your health, and the video games you play after school. All are programs and all were created by programmers.

How many times a day do you interact with computers? Are there computers in surprising places? How about a car? How about a phone? If you can control these computers and write programs for them, you can make things that millions of people use every day!

1. Using Hopscotch (5 minutes)
First, get your students acquainted with Hopscotch.

1.1 Finding the Hopscotch app on your iPad
1.2 Signing into your account (Email us at classrooms@gethopscotch.com for bulk student accounts.)
1.3 Making a new project: Tap on the white circle "+" button labelled "create"



2. Control Pad (E) (10 minutes)
The point of Cross the Road is to navigate a character across a field filled with obstacles (in this case, cars!). The player will use control buttons to direct their character. This control pad is one of the most universally recognizable video game elements.

One of the most important lessons of this activity is learning that the programmer must not only put together all the components of the game (buttons, background, character), but also explicitly tell the computer how they should work. For this, we need to create a rule, or code that tells the computer what to do and when to do it. A rule has two components: an event and commands (or action).

An event is a trigger that the computer recognizes and causes it to do some action. In Hopscotch, all events start with the word "When" and are the first thing you choose when you write a rule. Think of it as completing a "WHEN….., THEN….." sentence.

Events are deeply important for computer engineers because they tell the computer when it should do something. When you touch the phone icon on your home screen, then your phone brings up the interface to make calls. When an Angry Bird hits a block, then the block falls down.

Discuss some events (triggers) that happen in the classroom. Identify the trigger and resulting action: When I raise my hand (trigger), then stop talking (action), when the bell rings (trigger), then put down your pencil and turn in your test (action).

After a general discussion of rules and events, you can transition to talking about programming Cross the Road.

LESSON

In Cross the Road, when the up button is tapped (trigger), we want the hero to move up (action). Pose this challenge to your students and as a class discuss the steps the computer must take to complete it. Once the class agrees on what should happen, you can encourage them to begin working on their own control pads. You can have them do this as a class in several small groups, in pairs, or on their own.

Students should add the buttons that will be used as a control pad (right, left, and up) and a protagonist or "hero" that the buttons will move around the screen. In Hopscotch, we program objects or characters. They can be found by tapping on the "+" button on the bottom of the screen. You can also add a new character from the code editor by tapping "+ New object". Buttons can be implemented by using a text object and then typing in a block from the emoji keyboard.

For each button they want to include in the game, they will need to add a rule associated with it. They can do this by tapping the character that will be affected by the buttons (the hero) and giving it new rules. Encourage students to explore the events (When) menu in Hopscotch by tapping their hero, then "See code", and then testing out different events from the magenta menu. They can swipe to see events triggered by the iPad (iPad), interactions between characters (collisions), or logic (conditionals).

Challenge students to complete the code for the other two buttons in pairs or small groups. Ask them to consider: What code will they need to add to create a button that moves the character right when the right arrow is tapped and one that moves the character left when the left button is tapped? To which character should these rules be added? (The hero)

The following is sample code.

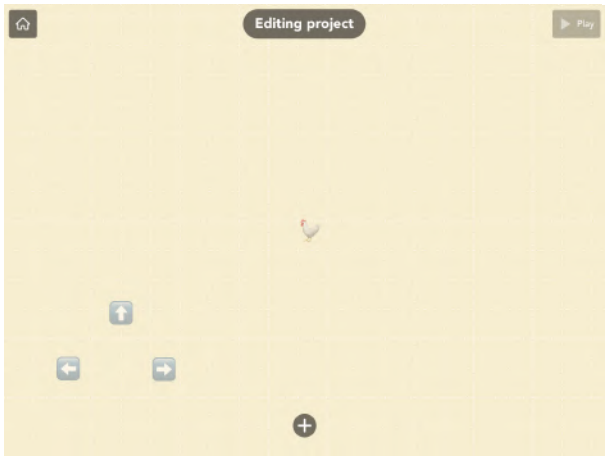
2.1 Add hero object and place at bottom of screen

You can also choose an emoji as your hero by selecting a text object and then choosing from the emoji keyboard.

**Using text and custom art is a premium Hopscotch feature that requires a subscription to Hopscotch or a one-time purchase to unlock.*

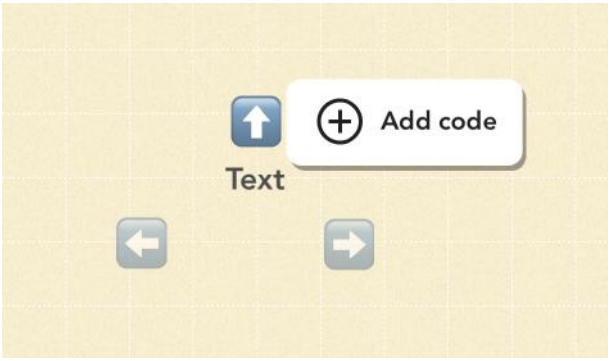
LESSON

2.2 Add 3 control buttons (up, right, and left) that the player will use to move their character



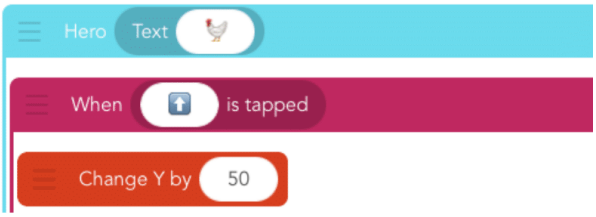
Use three different text objects to create the buttons. You can find the arrow buttons in your emoji keyboard. If you don't have emojis, you can enable them via your iPad settings.

2.3 Name the control buttons "up", "right", and "left"



Tap an arrow button, then the bolded text below it to rename it. Give each arrow button a name that corresponds to the direction it will move the character (up, right, and left). In programming, it is important to name your objects well so that other people can easily read your code in the future.

2.4 Write code to move the hero forward

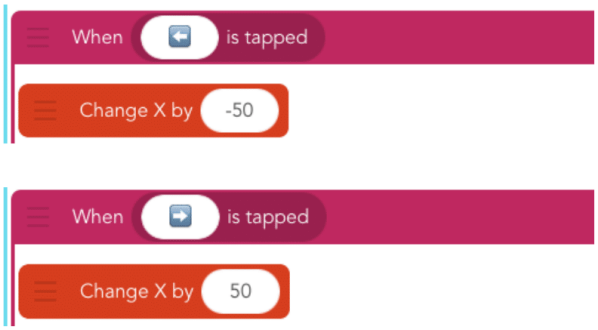


Choose the up button by tapping the iPhone icon in the "When tapped" header.

Review the Coordinate Plane. In Hopscotch, (0,0) is at the bottom left of your iPad screen, so the whole screen is in Quadrant I. Moving up is changing the Y position by a positive amount.

LESSON

2.5 Complete the buttons to move the hero left and right



Moving left is changing the X position by a negative amount, and moving right is changing the X position by a positive amount.

To test out your code, select the play button (turquoise triangle in the upper right corner of your screen).

3. Like a Boss (LS) (10 minutes)

Once the students have their hero working, the next step is to add some drama to the game by introducing a challenge—cars that drive back and forth across the screen indefinitely. These cars will be controlled by the computer (in game design, we call these kinds of automated characters bosses or non-player characters).

This is a good time to discuss **sequence** and **loops.**

Sequence is the order in which instructions are given to the computer. The idea of putting instructions in the correct sequence seems obvious and basic, but it's a vital concept in computer programming.

You can reference a real-life example: making sandwiches for their friends. Ask the class what process they would need to employ in order to make and wrap 10 tuna sandwiches. Does it matter if the process happens in the same order for each sandwich? What if they added mayo after putting canned tuna on bread? Or what if you put the bread in the bag before opening the tuna? Silly, but order matters.

Computers have a finite set of kinds of tasks they can accomplish. But when these tasks are combined properly, amazing things can be built. In addition to running instructions sequentially, computers are very good at repeating sets of instructions. In computer science we call this a "**loop**", or code that repeats.

Consider using a loop to repeat the sandwich making process: For the number of sandwiches I need: open the tuna, add mayo, stir, put on bread, put in bag.

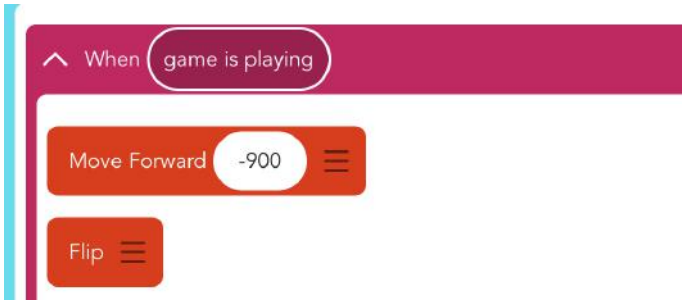
As a class, discuss the behavior of these cars and together make a list of the steps they take. Ask students to consider the difference between using "Repeat 10 Times" and "Repeat Forever". Which is appropriate for the sandwich? Which is appropriate for the car's movement? Also, consider what happens if instructions are out of order.

LESSON

When students have a hypothesis about how the cars should move, they can begin coding.

3.1 Add car emoji *Name car emoji "Car 1".*

3.2 Add new code to car: Make car move back and forth across the screen



(If you decide to break this lesson into two sessions, this would be a good place to stop.)

4. Randomness (5 minutes)
We can make our game more interesting by randomizing the speed of the cars. **Randomness** is a lack of pattern or predictability in events.

The concept of randomness is very important in computer programming because the most useful computer programs must be able to solve generalized (rather than specific) problems. For instance, it is much more useful to write a program that could find the factorial of any random number than a program that could only find the factorial of, say, the number seven. Having one generalized solution that can be used for a variety of specific inputs is at the heart of what makes computer programs powerful. And randomness can be used to test how robust that program is.

Randomness can also be used to make computer programs better. The Roomba vacuum can accomplish its task of cleaning any room anywhere by moving forward until it hits a wall, and then turning in a random direction. Imagine if the people who programmed the Roomba had to write specific directions for it to clean a square room, a rectangular room with two sofas in it, a long and narrow room…you get the idea. It might be more efficient in those specific instances, but they would never be able to account for all the potential rooms the Roomba might have to clean.

Randomness is very useful for programming computer games, because it drives the luck aspects of games. For instance, how often or when a block in Tetris appears is driven by randomness.

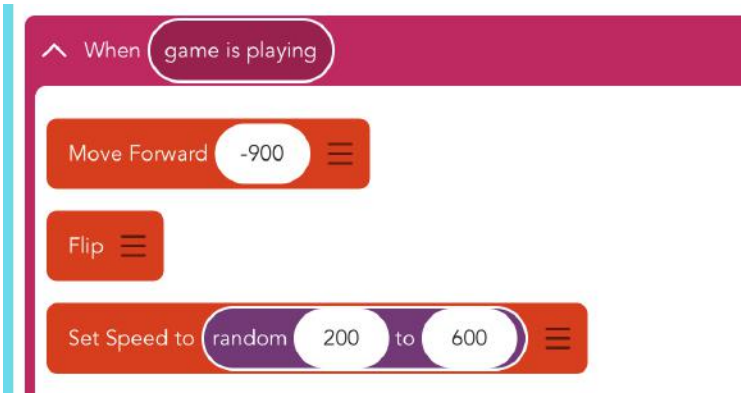
LESSON

Randomness depends on giving the computer options to choose from, or a range. You can discuss a real-life example of range. Ask your students to "Pick a number between 1 and 10". Imagine if you had just told them that you are thinking of a number and asked them to guess it. They would have been guessing for days. Instead, you gave them a range ("1 to 10"), or the lowest and highest number for Random to choose between. We will use randomness to make our games more fun and challenging.

Ask students to consider what would happen if the cars in the game all drove at the same speed. Would the game be fun? What would happen if you randomly set the speed of the cars? Would that make it more fun? (We think so!)

To set the car's speed, you need to determine its range. Ask your students to play around with the range and see what happens. What if you try (1,10)? What if you try (100,1000)? The default speed in Hopscotch is 400, so a range of (200,600) is pretty good.

4.1 Edit car's code: Set the speed to random each time



5. Collisions (E) (10 minutes)
The last obligatory element in Cross the Road is to establish collisions and then add more cars. A collision is a type of event, and in Hopscotch, it is represented as "When __ bumps __". When the hero bumps into a car, the hero should disappear.

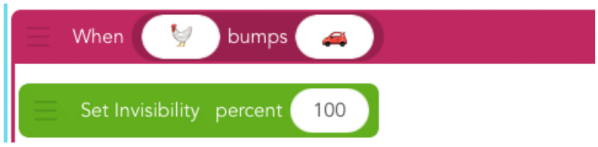
To finish the game, we need to add at least one other car to make it fun.

Students will need to add the collision rule to their hero, and then add and program more cars. Allow a set amount of time for this activity. In that time, some students will be able to add multiple cars and program their movement and collisions (using the same code as the first car). Others will achieve only one. As a class, discuss collisions and depending on the age of your students, see if they can implement the code on their own. Circulate and help the students who are struggling. This process is repetitive, but offers good practice and gives

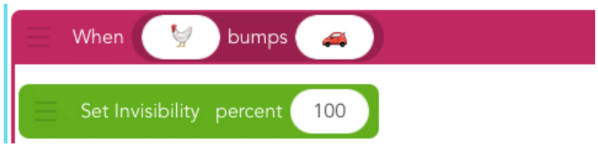
students a chance to see how one of the most important programming concepts (writing functions) is useful.

Use the refresh button to start the game over.

5.1 Add code to hero: Disappear when it collides with the car



5.2 Add and program more cars with the rule established above in 4.1
5.3 Add collision code to the hero for each new car

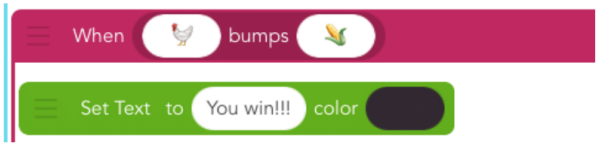


Name each additional car "Car 2", "Car 3", etc.

5.4 Test program, adjust position of cars

6. Victory (optional) (E) (5 minutes)
It's not a game if you can't win! Add a goal destination, or target, to give your hero somewhere to go. When the hero bumps the target, the game should say, "You win." We can program a text object to display this message when triggered by the collision.

6.1 Add a target object (corn)
6.2 Add a win message text object, and don't set the text



When you add a text object, if you tap "X" on the upper left corner instead of writing a name for your new text, it starts out invisible.

7. Publishing (5 minutes)
Share what you made with the world! Ask students to publish their programs, giving the game a descriptive name that they'll remember and pinching the image to adjust the screenshot. See if they can find their own and each other's projects in the community.

7.1 Publish your program

DIFFERENTIATION

(15 minutes, optional)

- Put in lots of cars
- Draw lanes
- Set speed
- Customize control pad with better emojis, different sizes, or by moving a different amount
- Animate the “You Win” text and give it a cool color

REFLECTION

(5 minutes, optional)

- What is coding? (telling computers what to do)
- What can you make with code? (apps, games, medical software)
- What is an event? (when to do something)
- Can you name some events, in Hopscotch or in real life? (“When the play button is tapped”, “When _ bumps _”)
- What is a collision? (when two things bump into each other)
- What do you think about coding? Is it fun? Hard? Rewarding?