GRADE 4 | GRADE 5 | GRADE 6
Nested Events | Conditional Statements | Efficient Code

{CODING}
IN THE CLASSROOM

GRADE 5 TEACHERS' GUIDE

Conditional statements and control structures

Welcome to **NII Explore's** ***Coding in the Classroom*** program for Grade 5 students. During the 4-week program, you and your class will complete:

- 3 online lessons
- 3 offline activities
- A final coding project

This teacher guide includes everything you need to get started!

THE GRADE 5 CODING CURRICULUM

As of 2020, Ontario's math curriculum includes coding expectations. Put simply, coding is when we write instructions, or "code", for a computer to follow. There are two core expectations that run through every grade level of the coding curriculum.

1. **Writing and executing code**
2. **Reading and altering existing code**

Each grade level introduces students to a new coding skill. Students can practice this new skill while also using the skills learned in previous grades. In Grades 1 to 4, students learn how to use four types of events – sequential, concurrent, repeating, and nested events.

Sequential events are steps that happen in a specific order, while concurrent events happen at the same time. Repeating events, also called **loops**, happen multiple times, and nested events are when we put a loop inside of another loop.

In **GRADE 5**, your students will start coding with **conditional statements**. A conditional statement is a piece of code that checks if a condition is true or false and then executes instructions based on the result. The most basic type of conditional is the "**if-then**" statement.

SCHEDULE AT A GLANCE

WEEK 1
- **Online Lesson 1**
  Intro to Scratch
- **Offline Activity 1**
  Everyday Algorithms

WEEK 2
- **Online Lesson 2**
  Build a Maze Game
- **Offline Activity 2**
  Conditional Rock-Paper-Scissors

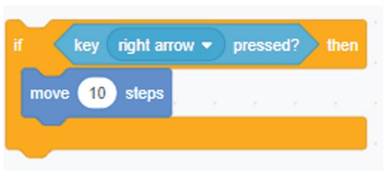
WEEK 3
- **Online Lesson 3**
  Make an Interactive Animation
- **Offline Activity 3**
  Animation Storyboarding

WEEK 4
- **Final Project**
  Make Your Own Animated Short

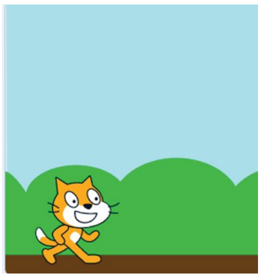
Here's an example from our everyday lives:

IF it's raining
THEN bring an umbrella

And here's one from Scratch, an online coding platform:

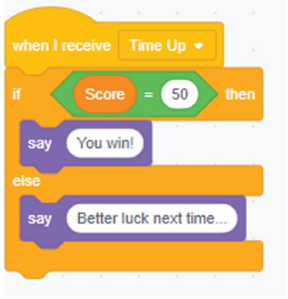


```
if <key [right arrow] pressed?> then
    move (10) steps
```

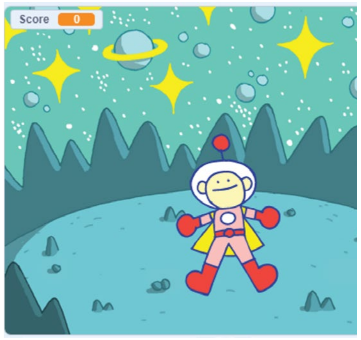


Example of an "if-then" statement.
If you press the right arrow key, then the cat will move 10 steps.

"**If-Else**" statements are another type of conditional statement. They tell the computer what to do whether the condition is true *or* false. Here's an example from a video game made in Scratch:



```
when I receive [Time Up]
if <(Score) = (50)> then
    say [You win!]
else
    say [Better luck next time...]
```



When time runs out in this game, the computer will check to see if the score is equal to 50. If the condition is true (the score is 50) then it will say "You win!". If the statement is false (the score is *not* 50), it will say "Better luck next time…"

We can also write "if-else" statements to make decisions in our everyday lives:

If
temperature > 20ºC
Then
wear shorts
Else
wear pants

In this algorithm, we would start by checking the temperature. ***If*** it's warmer than 20°C, ***then*** we will wear shorts. Otherwise (***else***) we will wear pants.

Conditional statements and loops (repeating events) are the two main types of **control structure**. Control structures are a key part of any programming language because they determine what a program will do.

In Scratch, these control structures are found under the Control tab and they are represented by **light orange** blocks. We can combine loops and conditional statements from the Control tab to make more sophisticated control structures.

In Grade 6, students will continue working with conditional statements and other control structures, this time with the goal of making their code more efficient.

PROGRAM SCHEDULE

The *Coding in the Classroom* program will last four weeks.
Here is a detailed guide of what you will be doing each week.

BEFORE WEEK 1

Read through this teacher guide, including the instructions for the three online lessons. If you have time, you may want to try the online activities for yourself.

Make sure your class has access to devices (laptops or tablets) for each of the online lessons. Your class will also need devices for the final project.

WEEK 1

Online Lesson 1 – Intro to Scratch

This lesson introduces students to Scratch, an online coding platform. Students will use basic commands to make a short animation.

PREP Log onto computers and open Scratch. Ask students to "Join Scratch" and make an account using their school email address.

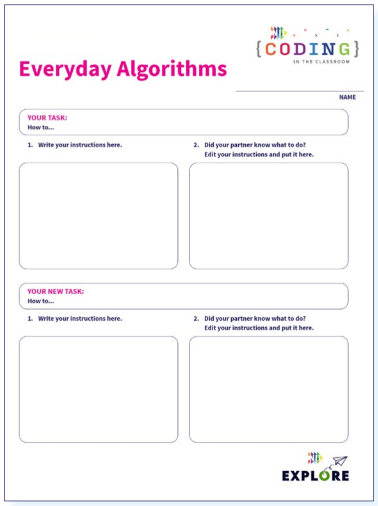
POST Complete "Offline Activity 1 – *Everyday Algorithms*" before next online session.

See **Page 5.5** for lesson instructions and **Page 5.9** for a quick reference guide.

Offline Activity 1 – Everyday Algorithms

Students will practice thinking like a computer as they write step-by-step instructions for everyday activities.

See **Page 5.13** for activity instructions.



Everyday Algorithms

NAME

YOUR TASK:
How to...

1. Write your instructions here.
2. Did your partner know what to do? Edit your instructions and put it here.

YOUR NEW TASK:
How to...

1. Write your instructions here.
2. Did your partner know what to do? Edit your instructions and put it here.

***Week 1** Everyday Algorithms*

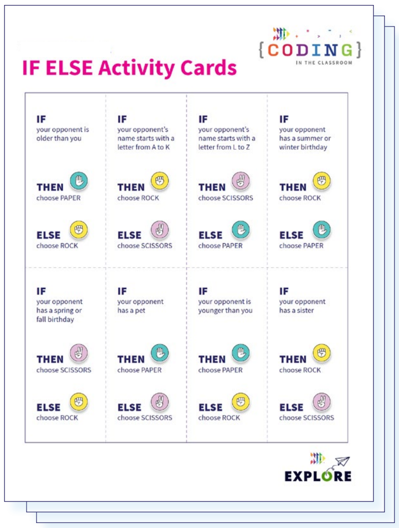
WEEK 2

Online Lesson 2 – Build a Maze Game

Students will edit and add to an existing Scratch file to create their own maze game. You will use conditional statements and loops, and students will learn about translations and rotations.

PREP Have students log into their Scratch accounts and open the activity link.

POST Complete "Offline Activity 2 – *Conditional Rock-Paper-Scissors*" before next online session.



IF ELSE Activity Cards

IF your opponent is older than you	IF your opponent's name starts with a letter from A to K	IF your opponent's name starts with a letter from L to Z	IF your opponent has a summer or winter birthday
THEN choose PAPER	THEN choose ROCK	THEN choose SCISSORS	THEN choose ROCK
ELSE choose ROCK	ELSE choose SCISSORS	ELSE choose PAPER	ELSE choose PAPER
IF your opponent has a spring or fall birthday	IF your opponent has a pet	IF your opponent is younger than you	IF your opponent has a sister
THEN choose SCISSORS	THEN choose PAPER	THEN choose PAPER	THEN choose ROCK
ELSE choose ROCK	ELSE choose SCISSORS	ELSE choose ROCK	ELSE choose SCISSORS

***Week 2** Conditional Rock-Paper-Scissors*

See **Page 5.16** for lesson instructions and **Page 5.21** for a quick reference guide.

Offline Activity 2 – Conditional Rock-Paper-Scissors

Your class will practice using and interpreting conditional statements with a fun, rock-paper-scissors tournament.

See **Page 5.25** for activity instructions.

WEEK 3

Online Lesson 3 – Make an Interactive Animation

Students will modify a program that uses inputs and conditional statements to make an interactive animation.

PREP Have students log into their Scratch accounts and open the activity link.

POST Complete “Offline Activity 3 – *Animation Storyboarding*” and the Final Project.

See **Page 5.30** for lesson instructions and **Page 5.35** for a quick reference guide.

Offline Activity 3– Animation Storyboarding

Students will prepare for their final coding project by planning out their animated story.

See **Page 5.40** for activity instructions.

WEEK 4

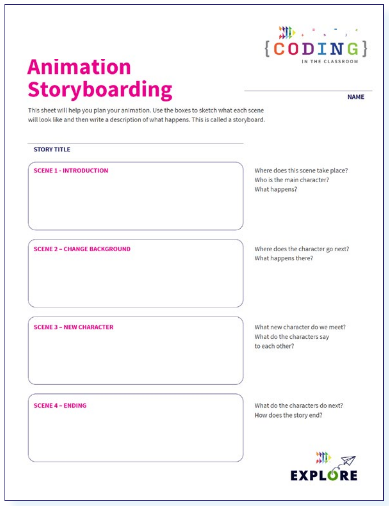
Final Project – Make Your Own Animated Short

Students will apply their learning to make their own animated short film in Scratch. When they’re finished, they will share their projects for you to evaluate.

See **Page 5.44** for project instructions and **Page 5.46** for assessment criteria.

AFTER WEEK 4

Keep the coding going with the additional resources on the **Page 5.49**!


Animation Storyboarding

NAME

This sheet will help you plan your animation. Use the boxes to sketch what each scene will look like and then write a description of what happens. This is called a storyboard.

STORY TITLE

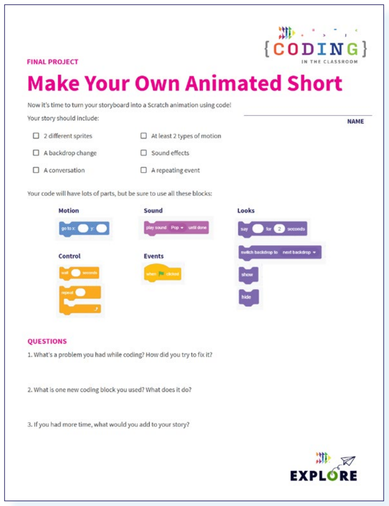
SCENE 1 - INTRODUCTION — Where does this scene take place? Who is the main character? What happens?

SCENE 2 – CHANGE BACKGROUND — Where does the character go next? What happens there?

SCENE 3 – NEW CHARACTER — What new character do we meet? What do the characters say to each other?

SCENE 4 – ENDING — What do the characters do next? How does the story end?

***Week 3** Animation Storyboarding*


FINAL PROJECT

Make Your Own Animated Short

Now it’s time to turn your storyboard into a Scratch animation using code!

NAME

Your story should include:

- ☐ 2 different sprites
- ☐ At least 2 types of motion
- ☐ A backdrop change
- ☐ Sound effects
- ☐ A conversation
- ☐ A repeating event

Your code will have lots of parts, but be sure to use all these blocks:

QUESTIONS

1. What’s a problem you had while coding? How did you try to fix it?

2. What is one new coding block you used? What does it do?

3. If you had more time, what would you add to your story?

***Week 4** Final Project*

ONLINE LESSON 1

Intro to Scratch

(Animation)

60 MINUTES

The three online lessons and final project all use Scratch. If you are new to Scratch, you may want to check out their "**Getting Started**" tutorial.

In this lesson, you will introduce students to Scratch and have them make their own accounts. The goal is to familiarize students with Scratch and some of its basic commands. Along the way, students will make a short animation. Lessons 2 and 3 will dive deeper into conditional statements and other control structures.

CURRICULUM CONNECTIONS

CODING

- **Math C3.1** – solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves conditional statements and other control structures
- **Math C3.2** – read and alter existing code that involves conditional statements and other control structures, and describe how changes to the code affect the outcomes

GEOMETRIC AND SPATIAL REASONING

- **Math E1.4** - plot and read coordinates in the first quadrant of a Cartesian plane using various scales, and describe the translations that move a point from one coordinate to another
- **Math E1.5** - describe and perform translations, reflections, and rotations up to 180° on a grid, and predict the results of these transformations

QUICK LINKS

Student Activity Link
scratch.mit.edu

Finished Example
scratch.mit.edu/projects/685760737

PowerPoint
Grade 5 – Week 1 – Intro to Coding

Quick Reference
Intro to Scratch (Animation) – Quick Reference

LESSON BREAKDOWN

SLIDE 1 - SET UP AND INTRODUCTION

Open the PowerPoint slides and Scratch on your own computer. Project for the students to see. Open or print the **Intro to Scratch (Animation) - Quick Reference** for your own use during the lesson.

SLIDE 2 - WHAT IS CODING?

Check if your students have coding experience and if they've used Scratch before. Ask them what coding means. "Coding is when we give instructions to a computer."

SLIDE 3 & 4 - WHAT TO EXPECT

Your class will complete three online lessons (using Scratch), three offline activities, and a final project (make their own animation).

If your students make an animation that they are particularly proud of, please share it with us at **explore@nii.ca**. NII Explore periodically awards prizes to some of our favourite coding projects.

SLIDE 5 - READY TO START

Have students open the activity link on their devices. It should take them to the Scratch home page.

MAKING SCRATCH ACCOUNTS

Start the first week by having all the students make Scratch accounts. This will let them save their projects and access them at home or on another day. It may take a few minutes but will be worthwhile in the long run.

Click "Join Scratch" in the top right. Create a username and password. Have students choose a username that will be easy for them to remember. For their password, they should choose something that is hard to guess. As an added measure, encourage them to write down their login credentials in a safe place.

It is not the best practice from a security standpoint, but for simplicity, they could use the same password that they use to log into their computers.

If their username is taken, have them add numbers at the end of it.

They do not need to give out their personal details other than an email address. Have them use their school email address.

If students already have Scratch accounts, they can log into them to start.

DEMO

Show a sample animation to give students an idea of what they are working towards. You can make your own animation or use this one:

Example:
scratch.mit.edu/projects/685760737

Click the green flag to start the animation.

TOUR OF SCRATCH

Have students create a new project (click "Create" in top left corner) then give them a tour of Scratch. Show them where the coding window and preview window are, and where they can access blocks for their code. **See Page 5.10 for more details. (Use same page number linking/formatting we used previously)**

CHOOSING A SPRITE AND BACKDROP

Have students choose a backdrop (bottom right corner).

Have students delete the default cat sprite (garbage can beside the sprite icon in the bottom right) and pick a new sprite (blue cat button in bottom right). Ask students to share which backdrop and sprite they picked so you know when they're ready to move on.

EVENTS AND DIALOGUE - STEP 1 IN QUICK REFERENCE

Each section of code begins with a yellow event block.

Go to "Events" and add "**when green flag clicked**" [Step 1a] to the coding window.

Now we will make our sprite say something.

From "Looks" find "**say hello for 2 seconds**" [Step 1b] and add it under the green flag block.

Let students change the "hello" message to say whatever they want.

From "Control", add a "**wait 1 seconds**" [Step 1c] block and explain what it does.

Have students test their program by hitting the green flag above the preview window.

INITIALIZE SPRITE - STEP 2 IN QUICK REFERENCE

Before we make our sprite move, we will need to initialize its position and direction.

Drag the sprite to the starting position that you want.

From Motions, add "**go to x:__ y:__**" [Step 2a] right after the green flag block. Check that the x and y coordinates in the block match the current position of the sprite in the sprite properties. If they don't, change the coordinates in the blue block.

Next initialize the direction/orientation of the sprite.

Add "**point in direction 90**" [Step 2b] under the go to x, y block.

Now no matter how much we move our sprite, it will always go back to the starting position when we click the green flag.

ADD MOTIONS - STEP 3 IN QUICK REFERENCE

Have students look through the Blue Motions tab and choose a few to add into their sprite's code. They can test what a block does by clicking on it. **See Quick Reference** for some motion examples [Step 3].

RECAP

Give time for students to do some experimentation with motions, dialogue, and waiting.

Ask for volunteers to share their work.

SWITCH COSTUMES - STEP 4 IN QUICK REFERENCE

Sprites can change their appearance by changing costumes.

In the top left corner, have students switch from "Code" to "Costumes". They can see here which costumes are available for their chosen sprite.

NOTE: Some sprites only have one costume. If they wish, students can make a new costume by right clicking on their single costume and picking "duplicate". They can then modify the copy to make it into a different costume (e.g., change its colour).

Switch back to "Code" and click "**switch costume to [ ]**" to see what the block does. Add it to the animation [Step 4].

Students may ask how to switch back to their original costume. They will need to add in another “switch costume to [   ]” at some point in their program (usually at the start).

SWITCH BACKDROPS - STEP 5 IN QUICK REFERENCE

Finish the Scratch intro by showing students how to switch between backdrops.

Add a second backdrop

In the bottom right, pick “Choose a Backdrop” then pick any option. Now click the “Stage” section in the bottom right then “Backdrops” in the top left. They will be able to see all their backdrops listed down the left side, just like the costumes earlier.

Switch back to the sprite’s “Code” and click “**switch backdrop to [   ]**” to see what the block does. Add it to the animation [Step 5]. If students want to switch back to their original backdrop, they will need to add another “**switch backdrop to [   ]**” block (usually at the start).

BONUS – EXPERIMENT TIME

If you still have some time, give students time to experiment with different motions and looks.

Give students the opportunity to share their animations with you or the class.

SLIDE 6 - RECAP

With about 5 minutes left in the class, switch back to the PowerPoint slides.

Ask students to share something that they learned in this lesson.

SLIDE 7 - POEM OF THE DAY

Each slideshow ends with a Poem of the Day to recap the lesson. Introduce the concept of the Poem of the Day then read the poem together.

SLIDE 8 - WHAT’S NEXT?

Let students know when you will be coding again. We recommend alternating between the online lessons and the offline activities. It requires less screen time for your class and will give students more time to absorb the new information.

Quickly preview what you will be doing next time (We are going to make our own game).

QUICK REFERENCE

Intro to Scratch
(Animation)

After making Scratch accounts, you will show students some basic commands. The goal is to familiarize students with Scratch and block-based coding. By the end of the class, each student will have made a simple animation.

SET UP

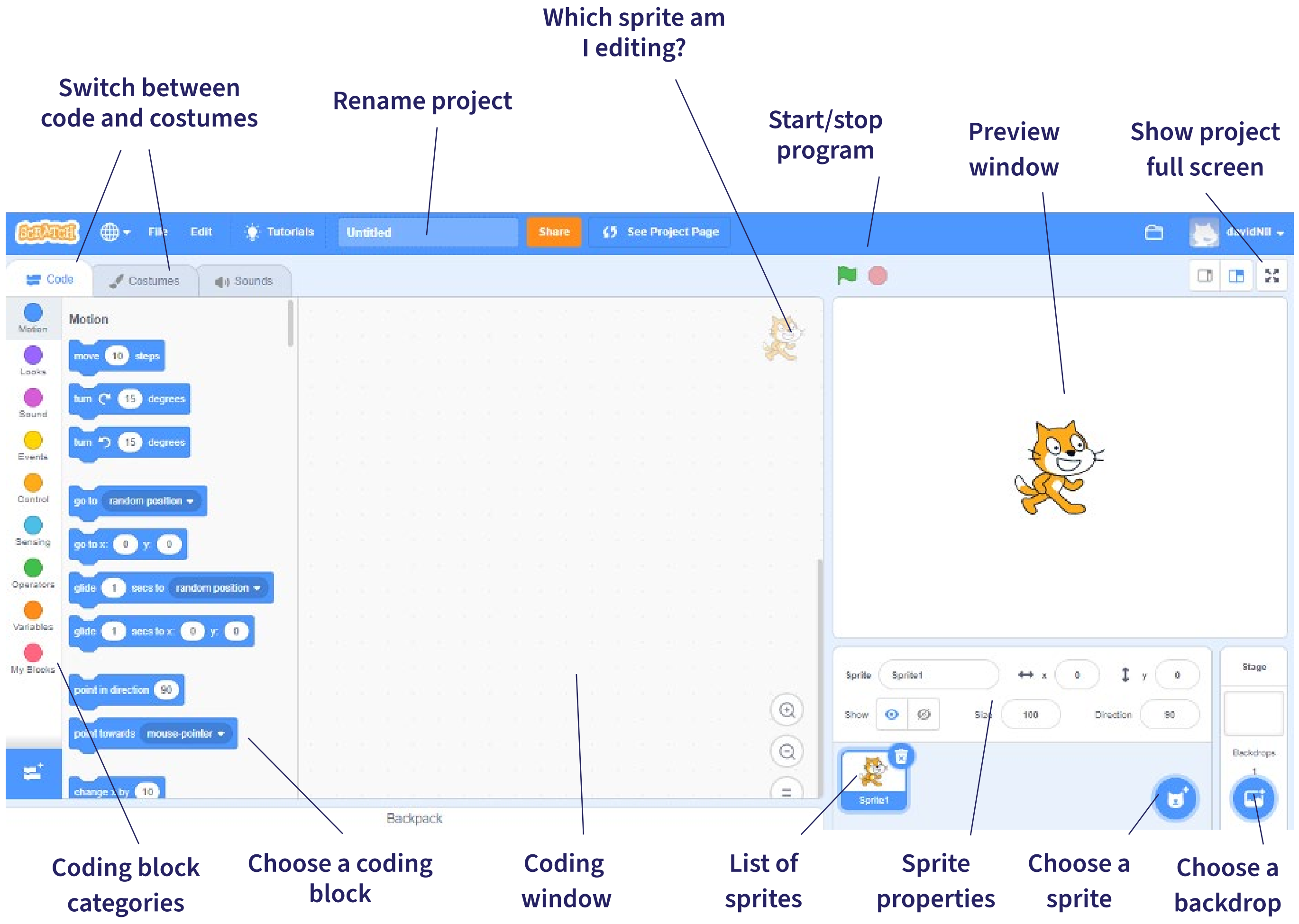
- Have students open Scratch and either create a new account or log into an existing one
- Open Scratch on your own computer and create a blank project

QUICK LINKS

Student Activity Link
scratch.mit.edu

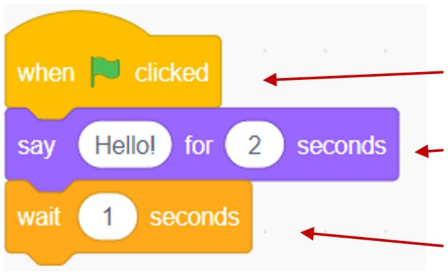
Finished Example
scratch.mit.edu/projects/685760737

SCRATCH TOUR



THE CODE

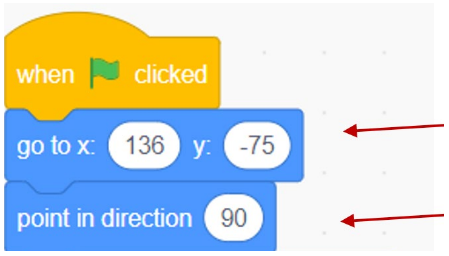
FINAL CODE


```
when [flag] clicked
say (Hello!) for (2) seconds
wait (1) seconds
```

Step 1a. Starts program.

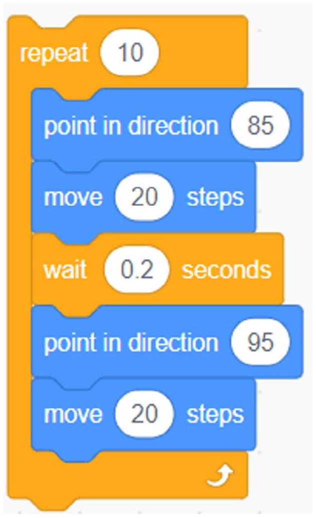
Step 1b. Sprite says "hello!" Students can change what sprite says.

Step 1c. Wait before executing next command.

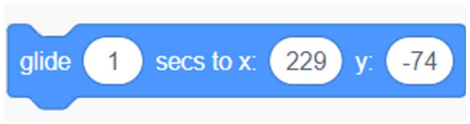

```
when [flag] clicked
go to x: (136) y: (-75)
point in direction (90)
```

Step 2a. Set initial location of sprite. Sprite will move back to starting position when program begins.

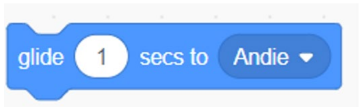
Step 2b. Set direction/orientation.


```
repeat (10)
    point in direction (85)
    move (20) steps
    wait (0.2) seconds
    point in direction (95)
    move (20) steps
```

Option 1


```
glide (1) secs to x: (229) y: (-74)
```

Option 2


```
glide (1) secs to (Andie ▾)
```

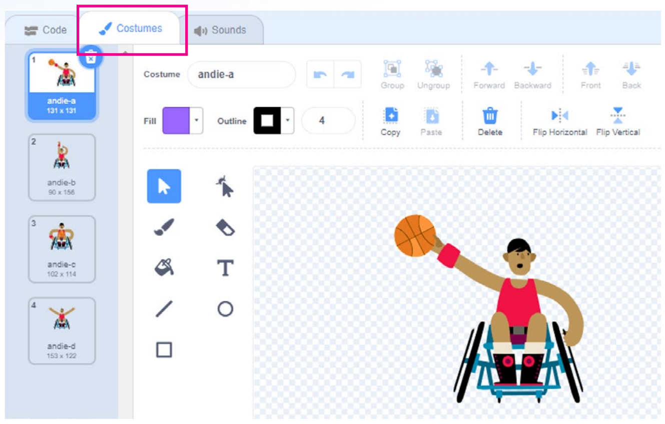
Option 3

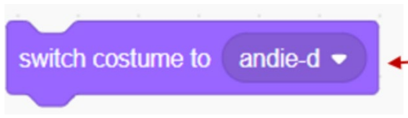
Step 3. Three ways to make your character move around the screen.

Option 1 – Simulate walking by having the sprite rotate back and forth while moving forward.

Option 2 – Have the sprite glide to a new position.

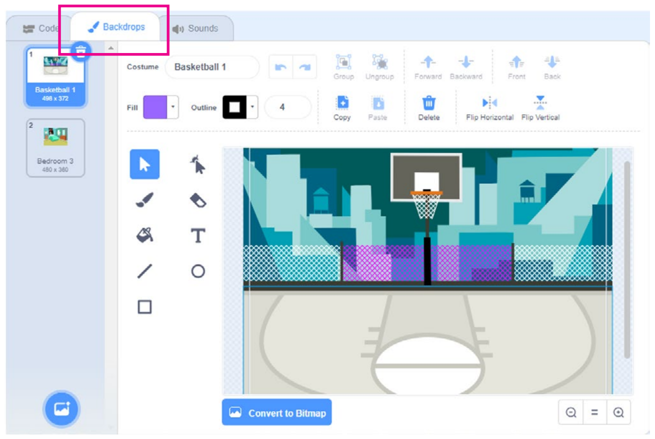
Option 3 – Have the sprite glide to another sprite.

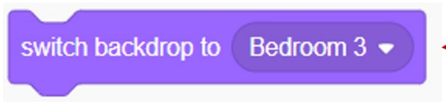




Step 4. Change costumes

You can see and edit the costumes in the sprite's "Costumes" tab. You can switch between costumes using the appropriate purple Looks blocks.





Step 5. Change backdrops

You can see and edit the backdrops by clicking "Stage" in the bottom right corner and then choosing the "Backdrops" tab. You can switch between backdrops using the appropriate purple Looks blocks.

OFFLINE ACTIVITY 1

Everyday algorithms

45 MINUTES

Think like a computer by writing algorithms – a series of step-by-step instructions – for everyday activities.

MATERIALS

Each student will need:

- 1 worksheet
- Something to write with

LEARNING OBJECTIVES

- Write and execute algorithms for everyday activities

CURRICULUM CONNECTIONS

- Math C3.1 & C3.2 (Coding)
- Language – Writing

SET-UP

Print and distribute worksheets.

INSTRUCTIONS

SUMMARY

Students will write detailed instructions for everyday activities like tying their shoes. Students will then test their algorithms by having a partner follow the instructions.

1. Ask students to recall what they learned about coding in the first online lesson. **ASK** What is coding? Why do computers need code?
2. Explain the premise of the activity. **SAY** “I will give you each a simple task and you will have to write step-by-step instructions telling someone how to do it. When you’re done, you’ll check how well your instructions worked by having a partner follow them.”
3. Demonstrate the activity with an example. Pick a volunteer and give them step-by-step instructions for how to do some activity around your classroom (e.g. sanitize hands, turn on computer and projector, clean the board).

4. Hand out worksheets and number students from 1 to 4. Their number determines what task they will write instructions for as follows:

   Group 1 – How to tie your shoes
   Group 2 – How to draw a dog
   Group 3 – How to fold a paper airplane
   Group 4 – How to sharpen a pencil

   Have students write their instructions or "code" in the first section of their worksheet. Remind them to be as detailed as possible.

   NOTE These tasks are provided as suggestions. Feel free to assign different everyday tasks or have students choose their own.

5. When students are done writing their instructions, split them into pairs. They will take turns with one person being the "coder" reading their instructions and the other being the "computer" following the instructions. Remind students that computers can only do exactly what they're told – only do something if your coder explicitly told you to do it.

6. Ask students to revise their code based on their test. **ASK** What changes can you make so that your instructions are more clear?

   They will write their revised code in the second section of their worksheet.

7. Pair students with a new partner and have them try giving and following their revised instructions.

8. If time allows, have the students try writing instructions for a different task (e.g., Group 1 does the Group 2 task, etc.). They can put their work in sections 3 and 4.

9. Debrief the activity using the discussion questions as a guide.

DISCUSSION QUESTIONS

What did you find the most difficult about this activity?

What changes did you make to your instructions to make them easier to follow?

Note: Fixing mistakes in code is called "debugging".

When might you need to give step-by-step instructions to someone instead of just saying something like "tie your shoes"?

Answer: The person might have never done that task before.

Why do you think we need to write such detailed instructions for the computer when we code?

Answer: Computers don't really know how to do anything. They have to be told exactly what to do as if they've never done it before.

Do you find it easier to give instructions to a person or a computer? Why?

Possible answers: Computers because they will do exactly what you say. Humans because they can fill in the gaps if you forget to say something.

ACTIVITY 1 WORKSHEET

Everyday algorithms

NAME

YOUR TASK:
How to...

1. Write your instructions here.

2. Did your partner know what to do?
Edit your instructions and put it here.

YOUR NEW TASK:
How to...

3. Write your instructions here.

4 . Did your partner know what to do?
Edit your instructions and put it here.

ONLINE LESSON 2

Build a maze game

60 MINUTES

Students will modify ("remix") an existing Scratch file to create their own maze game. The maze is designed so that students will have to add new code to complete each level. Students will learn about conditional statements and loops, while also reinforcing their knowledge of translations and rotations from math class.

CURRICULUM CONNECTIONS

CODING

- **Math C3.1** – solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves conditional statements and other control structures
- **Math C3.2** – read and alter existing code that involves conditional statements and other control structures, and describe how changes to the code affect the outcomes

GEOMETRIC AND SPATIAL REASONING

- **Math E1.4** - plot and read coordinates in the first quadrant of a Cartesian plane using various scales, and describe the translations that move a point from one coordinate to another
- **Math E1.5** - describe and perform translations, reflections, and rotations up to 180° on a grid, and predict the results of these transformations

QUICK LINKS

Student Activity Link
scratch.mit.edu/projects/721121679

Finished Example
scratch.mit.edu projects/818522508

PowerPoint
Grade 5 – Week 2 – Video Games

Quick Reference
Build a Maze Game (Mouse) – Quick Reference

LESSON BREAKDOWN

SET UP

Open the PowerPoint slides and the Scratch links on your own computer. Project for the students to see. Open or print the **Build a Maze Game (Mouse) - Quick Reference** for your own use during the lesson.

SLIDE 1 - WEEK 1 RECAP

Ask students to recall what they learned about in the last online lesson.

SLIDES 2 TO 6 - HISTORY MINUTE

Use slides to discuss video games as an application of coding. See slide notes for talking points.

DEMO

Quickly show students what the completed maze game looks like to give them an idea of what they are working towards.

Click the green flag to start the game then use the arrow keys to guide the mouse to the donut.

Link to finished game:
scratch.mit.edu/projects/818522508

LOG INTO SCRATCH

Have students log into their Scratch accounts and open the student activity link.

Have them click "Remix" to make their own version of the project and give it a new title.

The project should start with Level 1 when they open the file (a mouse between two straight black lines).

LEVEL 1 - PICK A TARGET SPRITE

Have students create a new sprite to be the target. In our finished example, the target sprite is a donut.

Ask students to share what they picked so you know they're ready.

INITIALIZE THE TARGET SPRITE

Change the size of target sprite (e.g., donut) to make it fit inside the maze. "100 is the default size, set the size to a smaller number to make it smaller." You can edit the size in the sprite properties area.

Drag the target sprite to the end of the maze.

NOTE: We won't write any code for the target sprite in this lesson.

SET END CONDITION - STEP 1 IN QUICK REFERENCE

Switch back over to the mouse's code. In the code for the mouse, we have pre-set a "repeat until" loop. Ask students when the game should repeat until. In other words, what is the goal?

Answer: In the code for the mouse, go to Sensing and add "**touching (target sprite)**" to the "repeat until" block [Step 1]. Explain: "We want the game to keep running until the mouse reaches the target."

ENABLE LEFT/RIGHT TRANSLATIONS - STEP 2 IN QUICK REFERENCE

Now it's time to make our mouse sprite move.

Point out the pink **define enable translations** block. Ask students if they remember what translations are from math class **Answer:** Translations are when we shift a shape left/right/up/down.

Under “define enable translations” we have pre-populated four conditional statements for the right/left/up/down arrows. For this level, we will start with the right/left arrows (we will come back to up/down later in the lesson).

It’s up to the students to figure out which of the blue Motion blocks to put inside the first two if-then blocks. Give them some time to do this. **HINT:** The x-coordinate tells us our left-right position.

Answer: if key (right arrow) pressed? then → **change x by 4** and if key (left arrow) pressed? then → **change x by -4** [Step 2].

The default is to change x by 10, but we recommend using 3 or 4 to allow finer control when navigating the maze. The mouse might move too quickly if you use bigger values.

PRELIMINARY TESTING

Let students test their left and right arrows by completing the first level (Remember to hit the green flag to start the game). After they reach the target sprite, a broadcast message called “level complete” is sent. A text sprite is pre-programmed to appear when it receives this message.

LEVEL 2

Have students change the backdrop to Level 2. They can do this by going to Looks and then clicking “next backdrop” or they can click on “Backdrops” in the “Stage” area (bottom right corner) and change to the Level 2 backdrop.

SETTING UP THE WALLS - STEP 3 IN QUICK REFERENCE

This maze has walls, but at this point they don’t actually do anything. Show students that you can beat Level 2 by going straight through the walls.

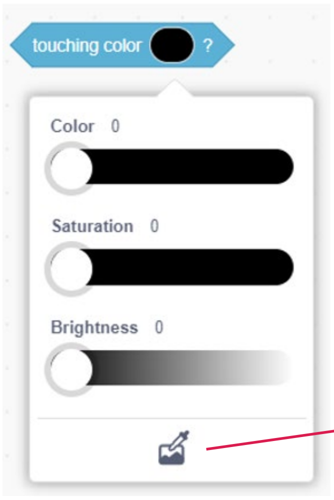
To make our walls work, we need to use something called a **conditional statement**. Explain if-then statements and where to find them in Scratch (the Control section).

“We use conditional statements, like these “if-then” blocks, to control what our programs do. The block checks IF a certain thing is true and if it is THEN it tells it what to do next.”

From Control, put an **if <blank> then** block inside the repeat until loop after “enable translations” [Step 3a].

Go to Sensing and find "**touching color ( )?**". "Scratch can sense what colour your sprite is touching. What colour should we set this to?" (**Answer:** Black because it's the colour of the walls).

Add “touching color (black)?” into the blank of the conditional statement [3a continued]. To change the colour, click on the bubble. You can set the colour to black using the sliding bars (Saturation = 0, Brightness = 0) or by using the eye dropper tool and clicking on the maze walls.



Eyedropper Tool

When you touch the black walls, what should happen? (Hit the wall).

Go to My Blocks and find the one called “hit wall”. Add the **hit wall** block inside the if-then statement. [Step 3b]

NOTE: “Hit wall” is a custom block that we made. You can scroll to the far right of the code to see how it’s defined.

OPTIONAL: Students could also add a sound effect that plays whenever they hit the wall. Just note that this could get out of hand if too many students are playing sound effects at the same time.

TEST THE WALLS

Have students test the game by running into the walls on purpose.

Challenge them to see how far they can get – what’s the issue? (Can only move right and left, can’t go up and down).

ADD UP AND DOWN COMMANDS - STEP 4 IN QUICK REFERENCE

Under “enable translations” there are empty conditional statements for the up and down arrows. Challenge students to figure out what needs to go inside the two statements.

Answer: if up arrow pressed? then → **change y by 4** and if down arrow pressed? then → **change y by -4** [Step 4].

TEST ALL FOUR MOVEMENTS

Have students test their new code by trying to beat this level. **NOTE:** The most common mistake is that they’ve mixed up their x’s and y’s. Remind students that x is the horizontal direction and y is the vertical direction.

LEVEL 3

Have students change the backdrop to Level 3. They can do this by going to Looks and then clicking “next backdrop” or they can click on “Backdrops” in the “Stage” area and change to the Level 3 backdrop.

ADD ROTATIONS - STEP 5 IN QUICK REFERENCE

Ask students to try Level 3. What’s the problem? (Mouse needs to turn).

Check if they remember the math term for turning. (Rotation) Rotations and translations are two types of transformation. In math, transformations are when we change shapes in some way. Show them that the **define enable rotations** block has two blank conditional statements.

In the first blanks, students will need to choose two keys (e.g., Z and C because they can be controlled with the left hand easily) to represent counterclockwise and clockwise rotation. Next, they will need to add the appropriate “turn ___ degrees” blocks.

Example: if key (z) pressed? then → turn ↺ 4 degrees and **if key (c) pressed? then → turn ↻ 4 degrees** [Step 5].

TEST AGAIN

Have the students test their latest controls (translations and rotations) by completing Level 3.

BONUS - EXTRA LEVELS

If time permits, it's fun for students to create their own level.

In backdrops, you'll find **Level 4**. This level is completely blank. Students can create their own maze using either the rectangle tool, line tool, or paintbrush tool. The colour should be pre-set to black, but students can change the colour using the same method as before.

Give students a chance to test their games with each other.

Finally, they could make their game automatically cycle through all the levels by putting all of their main program inside a **repeat (number of levels) loop** with a **wait 3 seconds** and **next backdrop** block at the end. See Steps 6a and 6b in Quick Reference.

SLIDE 7 - RECAP

With about 5 minutes left in the class, switch back to the PowerPoint slides.

Ask students to recall what they learned.
Ask if anyone can explain what a conditional statement is and where you find them in Scratch.

SLIDE 8 - POEM OF THE DAY

Share this week's poem as a recap.

SLIDE 9 - WHAT'S NEXT?

Quickly preview what you will be doing next week. You will be making an interactive animation.

Try **Offline Activity 2 (Conditional Rock-Paper-Scissors)** before the next online lesson.

{CODING}
IN THE CLASSROOM

QUICK REFERENCE

Build a maze game
(Mouse)

The maze game involves moving a mouse sprite through a maze to reach a target sprite. The user controls the mouse using the keyboard and a "Level Complete" message appears when the mouse reaches its target. Some parts of the game are pre-made for the students, but they will need to add to the code to make it functional.

QUICK LINKS

Student Activity Link
scratch.mit.edu/projects/721121679

Finished Example
scratch.mit.edu/projects/818522508

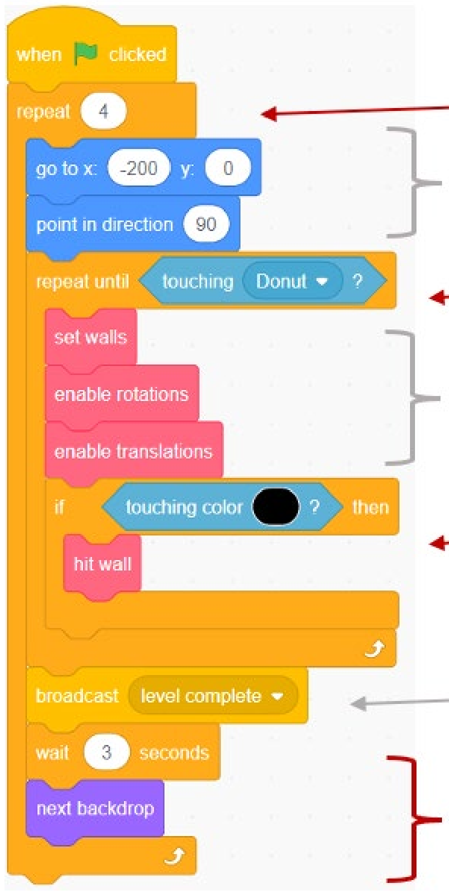
SET-UP

- Have students log into their Scratch accounts and open student link
- Open both the student version and the finished game on your computer. You will demo the finished version then work from the student version.

THE CODE

Here is the final code for each sprite. Areas in **GREY** are pre-coded for the students. Parts marked in **PINK** need to be added during the lesson. See the lesson plan on **Pages 5.17** to **5.20** for detailed instructions.

MOUSE SPRITE - MAIN PROGRAM



```
when [flag] clicked
repeat 4
    go to x: -200 y: 0
    point in direction 90
    repeat until touching Donut ?
        set walls
        enable rotations
        enable translations
        if touching color [●] ? then
            hit wall
    broadcast level complete
    wait 3 seconds
    next backdrop
```

Step 6a. OPTIONAL
Cycle through all the levels.

Initialization of sprite position and direction.

Step 1. Make game repeat until the mouse touches the target sprite (e.g., donut).

Enables all sprite movements.

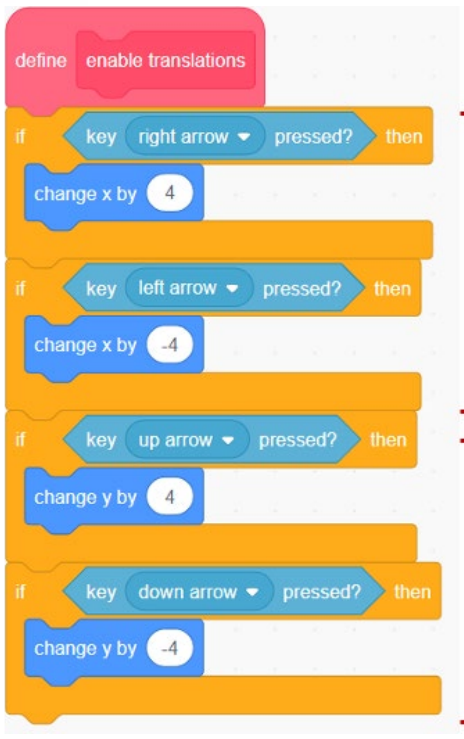
Step 3a. Add a conditional statement for when the mouse touches the black walls

Step 3b. This custom “My Block” blocks the mouse from going through walls.

Triggers “Level Complete” message.

Step 6b. Wait a few seconds and then advance to the next level after reaching the target.

ENABLE TRANSLATIONS SUBPROGRAM

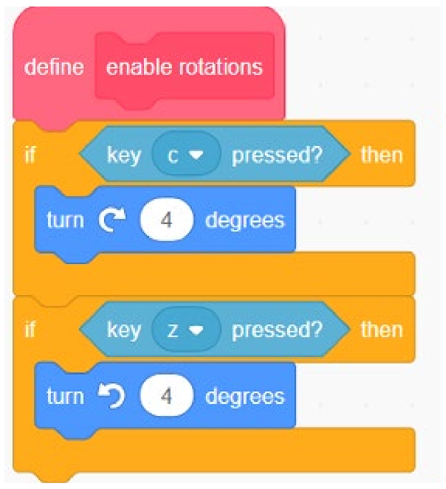


```
define enable translations
if key right arrow pressed? then
    change x by 4
if key left arrow pressed? then
    change x by -4
if key up arrow pressed? then
    change y by 4
if key down arrow pressed? then
    change y by -4
```

Step 2. Fill in these two conditional statements to enable left and right movement.

Step 4. Fill in these two conditional statements to enable up and down movement. Note that these blocks use **y not x**.

ENABLE ROTATIONS SUBPROGRAM

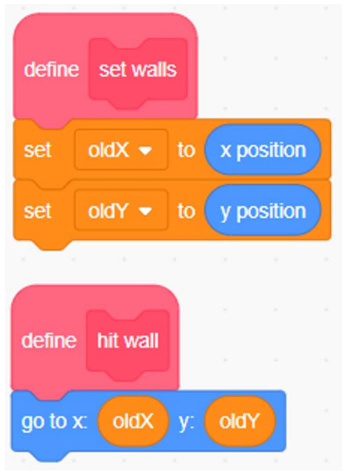


```
define enable rotations
if key c pressed? then
    turn ↻ 4 degrees
if key z pressed? then
    turn ↺ 4 degrees
```

Step 5. Fill in these two conditional statements to enable left and right rotations. Students will also need to choose which two keys to use (e.g., C and Z).

"HIDDEN" CODE

This code is off to the far right of the coding window. It prevents the mouse from going through walls but is too finicky to make students code themselves. You can show them if they are interested or ask what it does.

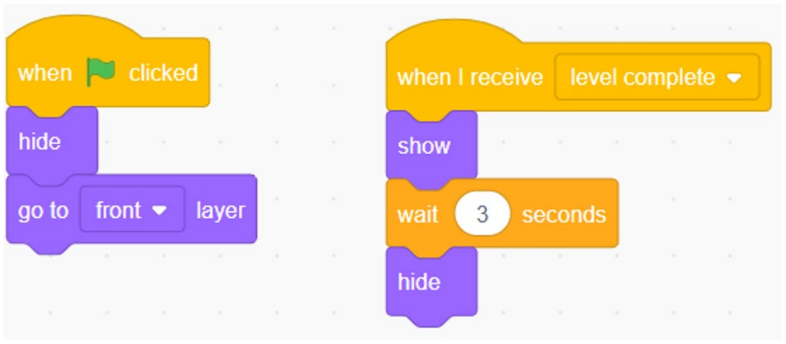


```
define set walls
set oldX to x position
set oldY to y position

define hit wall
go to x: oldX y: oldY
```

LEVEL COMPLETE SPRITE

This sprite is pre-made and coded for the students.
The sprite will hide when game begins and only appear once the player has reached the target sprite and the corresponding broadcast message has been sent.



```
when 🏁 clicked
hide
go to front layer

when I receive level complete
show
wait 3 seconds
hide
```

OFFLINE ACTIVITY 2

Conditional rock-paper-scissors

30 MINUTES

Practice evaluating conditional statements while engaging in some friendly competition!

MATERIALS

Each student will need:

- 3 popsicle sticks or some other small markers (e.g. pencils)
- 1 "If-Else" card

LEARNING OBJECTIVES

- Evaluate and execute "If-Else" conditional statements

CURRICULUM CONNECTIONS

- Math C3.1 & C3.2 (Coding)

SET-UP

Procure popsicle sticks or some other objects to use as counters.
Print and cut out "If-Else" cards found on **Pages 5.27** to **5.29**.
You will need 1 card per student.

INSTRUCTIONS

SUMMARY

Students will follow the conditional statements on their "If-Else" cards to play several games of rock-paper-scissors. They will use the popsicle sticks to track their wins and losses.

1. Ask students to recall what they learned about conditional statements in the online class.
2. Explain the premise of today's activity. **SAY** "We are going to play a rock-paper-scissors tournament, but instead of picking your moves, you'll each get a conditional statement that will tell you what to do."
3. Show an "If-Else" card and demonstrate how you would read it. For example, the card might say "**IF** your opponent has a pet **THEN** play rock **ELSE** play scissors". You would start by asking your opponent if they have a pet. If they do, then you will play rock in your game. If they don't, you will play scissors.

4. Pick a volunteer and give them an “If-Else” card. Play an example game against them.
5. Hand out 3 popsicle sticks and 1 “If-Else” card to each student. Give students a chance to read their card.
6. Have each student find a partner and play a practice match against them. Circle the room and check if students are having any trouble understanding their “If-Else” card.
7. Start the tournament. Students will move around the room playing a single game against different opponents. They will always choose their move based on their “If-Else” card. The loser gives one of their popsicle sticks to the winner. If it’s a tie, both players keep their sticks and trade their "If-Else" cards. When a student runs out of sticks, they are out of the round and can play against other eliminated students for fun.
8. After several minutes, check which students have the most popsicle sticks.
9. Have students trade “If-Else” cards and redistribute the popsicle sticks so everyone has three sticks again. Play another round to give students practice evaluating their new conditional statement.
10. Play as many rounds as time allows, leaving a few minutes for a debrief discussion at the end.

DISCUSSION QUESTIONS

What are the key words that help you recognize a conditional statement?

Answer: If, then, else.

What was your conditional statement and how did you test if it was true or false?

Answers will vary.

Why do you think the conditional statements were always about your opponent and not about yourself?

Answer: If the statement was about yourself, the answer would always be the same (e.g., you either have a pet or you don’t) and we wouldn’t really need to set conditions. Your opponent changes every time so you always have to re-check if the statement is true or false.

We use conditional statements in our everyday lives, like “IF it’s raining THEN bring an umbrella”. What are some other examples you can think of?

Possible answers: IF hungry THEN have something to eat; IF it’s recess time and sunny THEN go outside ELSE stay inside; IF tired or after 9 THEN go to bed; IF shoes are untied THEN tie them; IF someone sneezes THEN say “Bless you.”

OFFLINE ACTIVITY 2

IF ELSE cards (Page 1 of 3)

IF
your opponent is older than you

THEN
choose PAPER

ELSE
choose ROCK

IF
your opponent's name starts with a letter from A to K

THEN
choose ROCK

ELSE
choose SCISSORS

IF
your opponent's name starts with a letter from L to Z

THEN
choose SCISSORS

ELSE
choose PAPER

IF
your opponent has a summer or winter birthday

THEN
choose ROCK

ELSE
choose PAPER

IF
your opponent has a spring or fall birthday

THEN
choose SCISSORS

ELSE
choose ROCK

IF
your opponent has a pet

THEN
choose PAPER

ELSE
choose SCISSORS

IF
your opponent is younger than you

THEN
choose PAPER

ELSE
choose ROCK

IF
your opponent has a sister

THEN
choose ROCK

ELSE
choose SCISSORS

OFFLINE ACTIVITY 2

IF ELSE cards (Page 2 of 3)

IF
your opponent has a brother

THEN
choose SCISSORS

ELSE
choose PAPER

IF
your opponent can whistle

THEN
choose ROCK

ELSE
choose PAPER

IF
your opponent likes to code

THEN
choose SCISSORS

ELSE
choose ROCK

IF
your opponent is wearing green or red

THEN
choose PAPER

ELSE
choose SCISSORS

IF
your opponent is wearing blue or pink

THEN
choose PAPER

ELSE
choose ROCK

IF
your opponent's favourite number is greater than 10

THEN
choose ROCK

ELSE
choose SCISSORS

IF
your opponent's favourite number is less than 10

THEN
choose SCISSORS

ELSE
choose PAPER

IF
your opponent's favourite ice cream is chocolate

THEN
choose ROCK

ELSE
choose PAPER

OFFLINE ACTIVITY 2

IF ELSE cards (Page 3 of 3)

IF
your opponent's name has 6 or more letters

THEN
choose SCISSORS

ELSE
choose ROCK

IF
your opponent's name has less than 6 letters

THEN
choose PAPER

ELSE
choose SCISSORS

IF
your opponent takes the bus to school

THEN
choose PAPER

ELSE
choose ROCK

IF
your opponent likes pineapple on pizza

THEN
choose ROCK

ELSE
choose SCISSORS

IF
your opponent's favourite class is math or gym

THEN
choose SCISSORS

ELSE
choose PAPER

IF
your opponent's favourite class is science or art

THEN
choose ROCK

ELSE
choose PAPER

IF
your opponent has been in your class before

THEN
choose SCISSORS

ELSE
choose ROCK

IF
your opponent sits on the same side of the class as you

THEN
choose PAPER

ELSE
choose SCISSORS

ONLINE LESSON 3

Make an interactive animation

60 MINUTES

Students will modify ("remix") an existing Scratch file to create an interactive animation. The story follows a magician who performs three different tricks. Their code will take user inputs and then use conditional statements to decide what will happen next.

CURRICULUM CONNECTIONS

CODING

- **Math C3.1** – solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves conditional statements and other control structures
- **Math C3.2** – read and alter existing code that involves conditional statements and other control structures, and describe how changes to the code affect the outcomes

GEOMETRIC AND SPATIAL REASONING

- **Math E1.3** – plot and read coordinates in all four quadrants of a Cartesian plane, and describe the translations that move a point from one coordinate to another

QUICK LINKS

Student Activity Link
scratch.mit.edu/projects/721404702

Finished Example
scratch.mit.edu/projects/818524594

PowerPoint
Grade 5 – Week 3 – Animation and Final Project

Quick Reference
Make an Interactive Animation – Quick Reference

LESSON BREAKDOWN

SLIDES 1 AND 2 - WEEKS 1 AND 2 RECAP

Open the PowerPoint slides and the Scratch links on your own computer. Project for the students to see. Open or print the **Make an Interactive Animation - Quick Reference** for your own use during the lesson.

Ask students to recall what you discussed in the previous two lessons. "What are two types of control structure?" (Repeat loops and conditional statements) How do they work?

SLIDE 3 - PREVIEW

Show "Today…" slide then preview this week's activity by showing the finished animation as an example.

Finished animation:
scratch.mit.edu/projects/818524594

"We will be making an animation like Week 1, but this time it's interactive. The viewer gets to decide what happens next."

NOTE: For each question, the code is written so you only have to type the first letter for your answer. For example, "**(D)esert**" means that you can enter "D" to choose desert.

LOG INTO SCRATCH

Have students log into their Scratch accounts and open the activity link.

Have them click "Remix" to make their own version of the project and then change the title to something of their choosing.

INPUTTING ANSWERS

Show students the starting code and talk through what it does.

Talk about input and answers – "When you type in an answer, Scratch saves it as a temporary variable called "answer". In coding, we call this an input."

Here the code takes our answer and uses it in the say block so that the character says "My name is (answer)".

TRICK 1: TELEPORTATION - ADD TWO BACKDROPS

Explain the first trick "We are going to switch to a new backdrop, but the viewer will get to pick which one."

Add two new backdrops

Click "Choose a Backdrop" in the bottom right and choose a new one. Do this twice. You can view all your backdrops by clicking "Backdrops" in the bottom right and then "Backdrops" again in the top left.

Ask students which backdrops they chose.

CHANGING BACKDROPS - STEP 1 IN QUICK REFERENCE

"Now we need to add code so that the viewer can choose which of these backdrops we go to."

Connect the section that begins "**say For my first trick, I'm going to teleport.**" to the main program.

In the **ask and wait** block, students will type "**Where should I go? (Option 1) or (Option 2)?**" [Step 1a]. Students can stylize their answer choices like this: (D)esert or (S)pace – that way the user knows that they only have to type a single letter to make a choice and reduces the chance of spelling mistakes. Instead of desert and space, students will use whichever two backdrops they chose.

Next, students will see two conditional (if-then) statements that say **if answer = (blank) then**. In the first blank, they will add the letter that represents their first backdrop (e.g., D for desert). Then, inside the conditional statement, they will add **switch backdrop to (Backdrop 1)**. They will repeat these steps for both backdrops.

Here's an example of what their code may look like:

If answer = D then switch backdrop to Desert and **If answer = S then switch backdrop to Space** [Step 1b]

TESTING FIRST TRICK

Let students test everything they have so far by clicking the green flag – does their first trick work?

TRICK 2: MAKE A ROCK GROW OR LEVITATE

Explain what the second trick will be. "We will make a rock either grow or levitate."

MAKE ROCK APPEAR - STEP 2 IN QUICK REFERENCE

First, we need to make the rock appear on-screen.

Switch to the rock's code (choose Rock at the bottom) and add the following code to make it appear:

When backdrop switches to Desert → show

When backdrop switches to Space - > show [Step 2]

NOTE: Instead of desert and space, students will use whichever two backdrops they chose.

ASKING AUDIENCE

"We have two tricks we can choose from. We can make the rock either grow or levitate."

Back in the magician's code, connect the section of code that begins "**Pretty cool, huh?**" to the main program.

This section already contains the code that asks the audience to choose either grow or levitate. However, students still need to tell the rock what to do when it receives the signal to perform one of these actions.

GROW - STEP 3 IN QUICK REFERENCE

Back in the Rock's code, find the block that says **when I receive (grow)**. This is where we tell the rock how to grow.

If the rock isn't visible, go to Looks and click the purple "show" block so students can see it as they work.

Ask: Under "when I receive (grow)" what could we add?

Let students play around with different options in "Looks". Our finished version uses a repeat loop [Step 3a] but the simplest way to do it is with one "**set size to (blank)%**" block [Step 3b].

LEVITATE - STEP 4 IN QUICK REFERENCE

Under "when I receive (levitate)" students will need to add some code.

Let them play around with blue Motion blocks and see what they come up with. The simplest solution is with a "**glide 1 secs to x:___ y:_____**" [Step 4b] but there are other options that work [Step 4a].

To use the glide option, have students drag the rock to the position they want the rock to end up and then add in the glide block. Have them check that the x and y coordinates match the position they want.

TESTING

Test out what we have so far – do both tricks work?

NOTE: Check the time. You might not be able to get to the third trick so feel free to end this lesson here.

TRICK 3 - TRANSFIGURING THE ROCK

If time allows, move on to the third and final trick. Start by explaining what the trick is "We will be transforming the rock into some other sprite."

OPTIONAL: PICKING SPRITES

For simplicity, we have pre-chosen the Unicorn and Donut as the two options, but you could have students pick something else. If they make new sprites, have the students drag their chosen sprites to the position they want them (i.e., wherever the rock was) then add "**When green flag clicked → hide**" to initialize both new sprites.

ASKING THE QUESTION - STEP 5 IN QUICK REFERENCE

In the magician's code, connect the final section of code (begins with "**Ta-da!**") to the main program. Students will need to add their answer choices (e.g., U for Unicorn, D for Donut) into the **ask and wait** block and the conditional statements.

Inside the two if-then statements, students need to add the appropriate broadcast messages like we did for the second trick e.g., "**broadcast (donut)**" if the user enters D for Donut [Step 5].

SHOWING SPRITES - STEP 6 IN QUICK REFERENCE

Ask students what we will need to add to the new sprites' code to get them to appear at the right time.

Answer: "**when I receive (donut) → show"** and **"when I receive (unicorn) → show**".

NOTE: The Rock has been pre-programmed to hide when either of these new sprites appear.

TEST ENTIRE ANIMATION

Have students play through their entire animation – does it work? Have them share with fellow students.

SLIDE 4 - RECAP

With about 5 minutes left in the class, switch back to the PowerPoint slides.

Review what students learned today and the entire coding program. Possible topics: Inputs, if-then (conditional) statements, repeat loops, Cartesian coordinates. See slide notes for talking points.

SLIDES 5 AND 6 - FINAL PROJECT PREVIEW

Use the slides to explain the final project.

"You will be applying everything we've talked about to make your own animation. You can keep working on the one we made together and make it even better, or you can create something new."

"We will start by making something called a storyboard. Then we will make your story come to life using Scratch."

You can show this animation as an example:
scratch.mit.edu/projects/675171912

When students are done, they will share their work for you to evaluate.

NII Explore is always accepting student submissions. Email your students' projects to **explore@nii.ca** - we hand out prizes periodically for some of our favourite animations!

SLIDE 7 - POEM OF THE DAY

Share the final Poem of the Day.

SLIDE 8 - WHAT'S NEXT?

Complete the final project in the next couple weeks while this lesson is still fresh. If you haven't already, try out the offline activities with your class. Happy coding!

QUICK REFERENCE

Make an interactive animation

The students will be animating a magic show in which the magician performs three different magic tricks. For each trick, the user (viewer) will choose between two options. Students will learn how to process inputs in Scratch and use conditional statements to control the output. Some of the code is pre-made, so students will be adding to the existing code.

QUICK LINKS

Student Activity Link
scratch.mit.edu/projects/721404702

Finished Example
scratch.mit.edu/projects/818524594

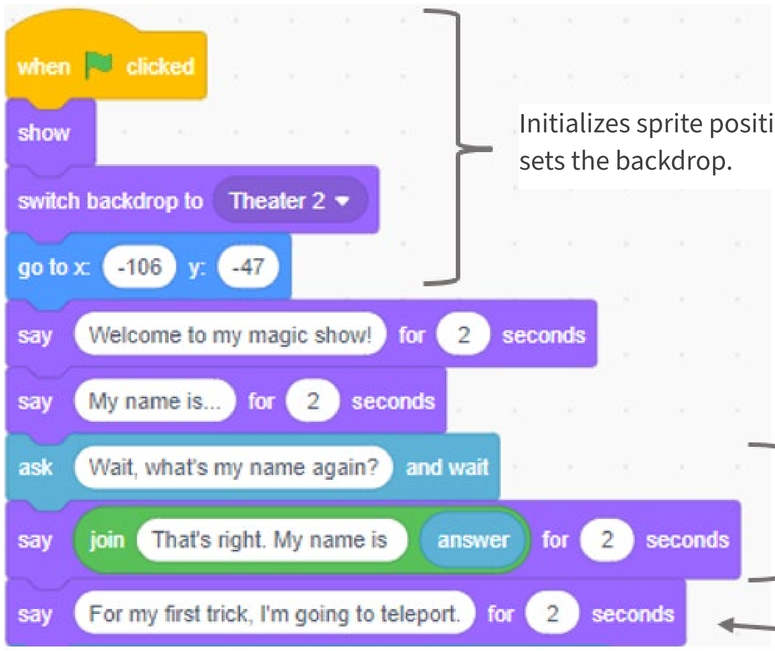
SET UP

- Have students log into their Scratch accounts and open student link
- Open both the student version and the finished game on your computer. You will demo the finished version then work from the student version

THE CODE

Sections marked in **GREY** are pre-populated for the students. Steps marked in **PINK** will be added during the coding class. See the lesson plan on **Pages 5.32** to **5.34** for more detailed instructions.

MAGICIAN (MAIN SPRITE)

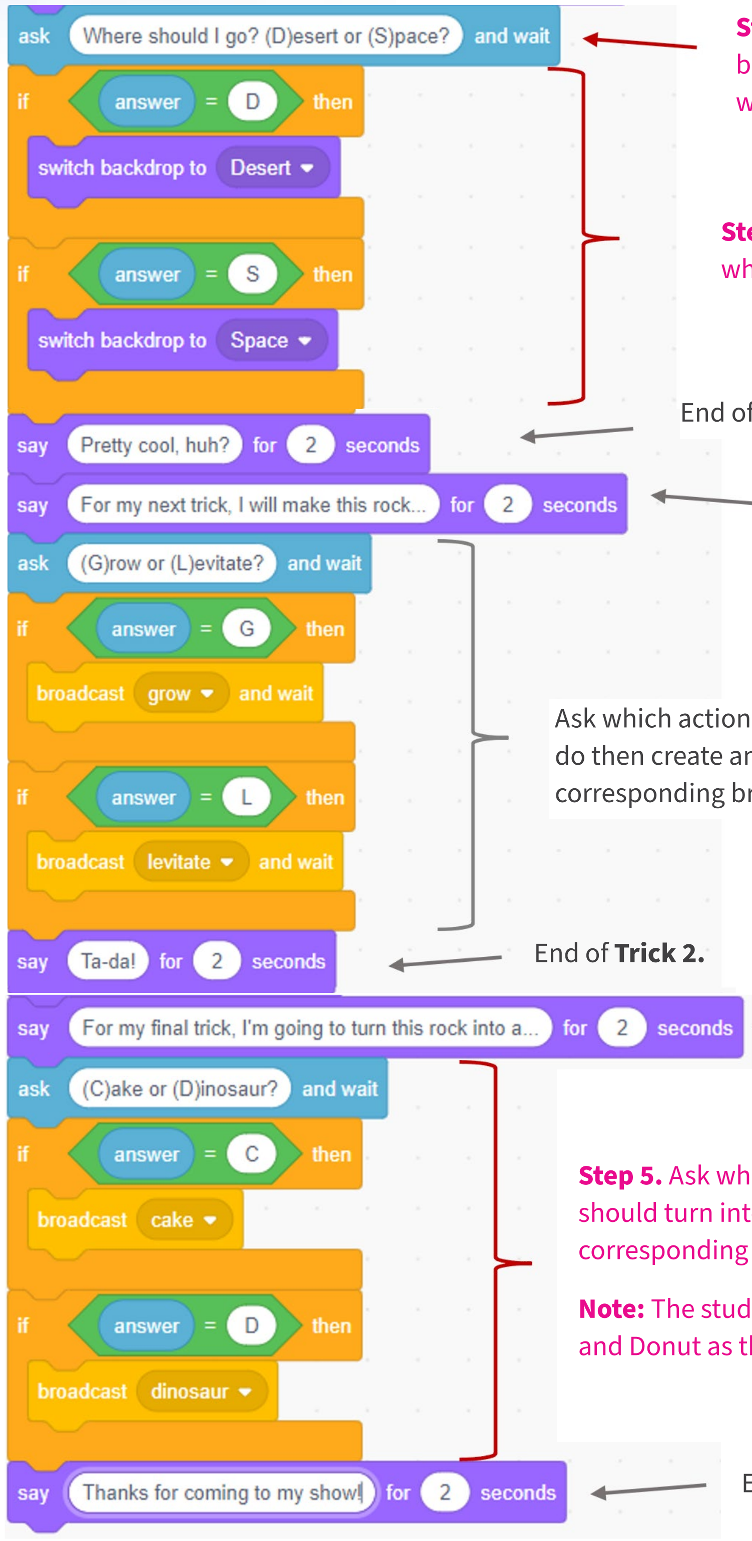


```
when [flag] clicked
show
switch backdrop to (Theater 2 ▾)
go to x: (-106) y: (-47)
say (Welcome to my magic show!) for (2) seconds
say (My name is...) for (2) seconds
ask (Wait, what's my name again?) and wait
say (join (That's right. My name is ) (answer)) for (2) seconds
say (For my first trick, I'm going to teleport.) for (2) seconds
```

Initializes sprite position and sets the backdrop.

Takes name as input and uses it in a sentence.

Start of **Trick 1.**



```
ask [Where should I go? (D)esert or (S)pace?] and wait
if <(answer) = [D]> then
    switch backdrop to [Desert ▾]
if <(answer) = [S]> then
    switch backdrop to [Space ▾]
say [Pretty cool, huh?] for (2) seconds
say [For my next trick, I will make this rock...] for (2) seconds
ask [(G)row or (L)evitate?] and wait
if <(answer) = [G]> then
    broadcast [grow ▾] and wait
if <(answer) = [L]> then
    broadcast [levitate ▾] and wait
say [Ta-da!] for (2) seconds
say [For my final trick, I'm going to turn this rock into a...] for (2) seconds
ask [(C)ake or (D)inosaur?] and wait
if <(answer) = [C]> then
    broadcast [cake ▾]
if <(answer) = [D]> then
    broadcast [dinosaur ▾]
say [Thanks for coming to my show!] for (2) seconds
```

Step 1a. After picking two backdrops, ask which one user wants to go to.

Step 1b. Use answer to determine which backdrop to change to.

End of **Trick 1.**

Start of **Trick 2.**

Ask which action the rock should do then create and send the corresponding broadcast message.

End of **Trick 2.**

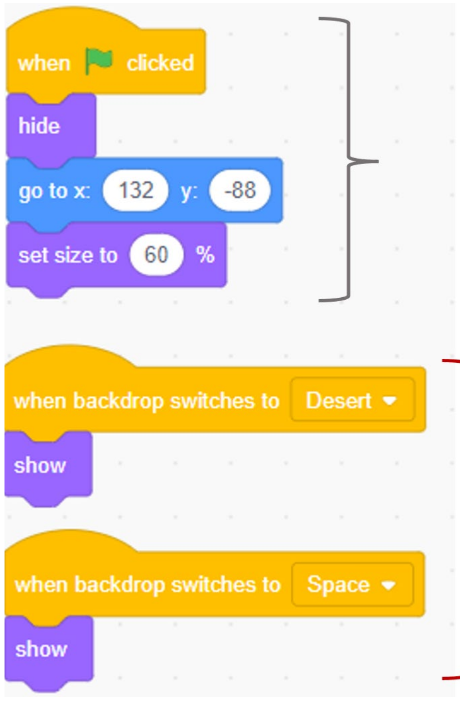
Start of **Trick 3.**

Step 5. Ask which sprite the rock should turn into then send the corresponding broadcast message.

Note: The student version uses Unicorn and Donut as the default sprites.

End of **Trick 3.**

ROCK CODE - SECONDARY SPRITE



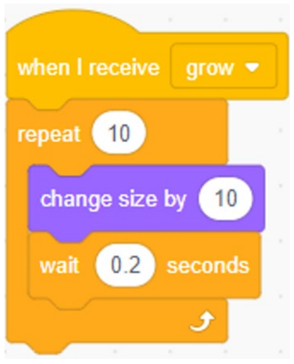
```
when [flag] clicked
hide
go to x: 132 y: -88
set size to 60 %
```

Initializes sprite position and size.

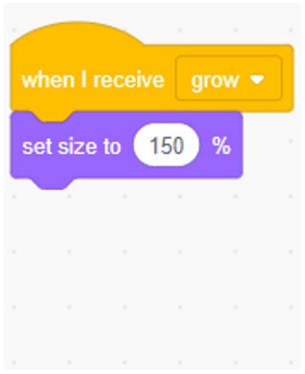
```
when backdrop switches to Desert ▾
show

when backdrop switches to Space ▾
show
```

Step 2. Appears when the backdrop changes.



```
when I receive grow ▾
repeat 10
    change size by 10
    wait 0.2 seconds
```

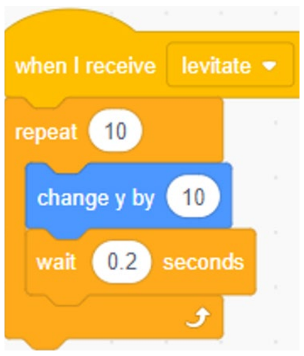


```
when I receive grow ▾
set size to 150 %
```

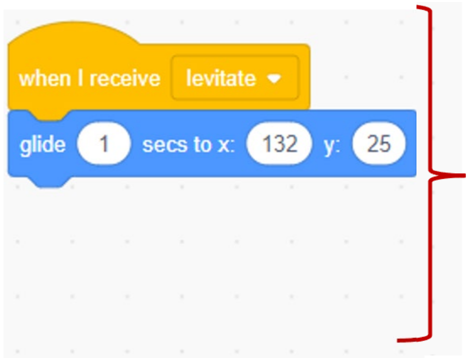
Step 3. Two options to make the rock grow.

Option 1 (Step 3a): Use a repeat loop and wait command.

Option 2 (Step 3b): Simply set size to something bigger.



```
when I receive levitate ▾
repeat 10
    change y by 10
    wait 0.2 seconds
```

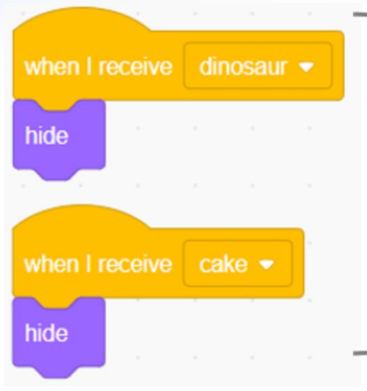


```
when I receive levitate ▾
glide 1 secs to x: 132 y: 25
```

Step 4. Two options to make the rock levitate.

Option 1 (Step 4a):
Use a repeat loop and wait command.

Option 2 (Step 4b)
Glide to a new position (same x value, higher y).

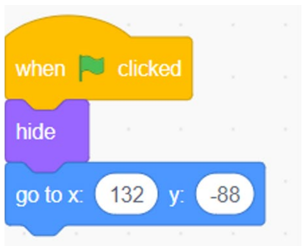

```
when I receive [dinosaur ▾]
hide

when I receive [cake ▾]
hide
```

Trick 3

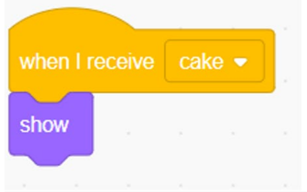
Disappear when either of the new sprites appears.

NEW SPRITES - WHICHEVER SPRITES APPEAR IN TRICK 3, E.G., DONUT AND UNICORN

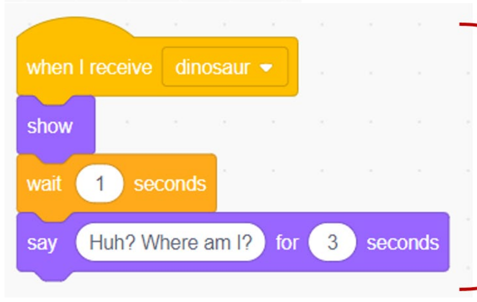

```
when ⚑ clicked
hide
go to x: (132) y: (-88)
```

Initialize

NOTE: “go to” command is optional because we probably won’t be moving this sprite. Just drag sprite to correct position.


```
when I receive [cake ▾]
show
```

Step 6. Sprite appears when correct broadcast message is received.


```
when I receive [dinosaur ▾]
show
wait (1) seconds
say (Huh? Where am I?) for (3) seconds
```

Step 6 Optional: Have the sprite say something at the end.

{CODING}
IN THE CLASSROOM

OFFLINE ACTIVITY 3

Animation storyboarding

30 – 45 minutes

Students will prepare for their final coding project by planning an animated story.

LEARNING OBJECTIVES

- Create the storyboard for an original animation
- Plan a coding project

CURRICULUM CONNECTIONS

- Math C3.1 & C3.2 (Coding)
- Language - Writing

SET-UP

Print the worksheets for the Animation Storyboard and the Final Project on the front and back of a single page. Open this **animation example** on your computer and project. Watch this **video tutorial** about the Final Project.

INSTRUCTIONS

1. Ask students to recall what they learned in the online coding classes.
2. Explain the premise of the activity. **SAY** "You will be planning an animation like the ones we made in our coding classes. The next time we have the computers, you will be coding your animation in Scratch."
3. Show the animation example. **ASK** What kind of motions did they use? Does this give you any ideas for your story?

   NOTE: This animation is more complex than what students are expected to make. It is meant to demonstrate many possibilities for what they could include in their own work.

MATERIALS

Each student will need:

- 1 worksheet
- Something to write with

4. Hand out worksheets and review the instructions together. Demonstrate how to fill in the storyboard. They will sketch what each scene will roughly look like, and write a description of what happens in that scene.

   There are guiding questions to help them plan each scene. The storyboard will provide some structure to their stories rather than being completely open-ended. Remind them that they will be using Scratch to make whatever they come up with.

5. Give students time to work on their stories. Circle the room to assist students as needed. You may want to have Scratch open on your computer to remind students of what characters and backdrops are available if they need inspiration.

   POSSIBLE PROMPTS

   - How would this character move?
   - Does your character have a name?
   - What backdrop might this character move to?
   - What character would they meet?
   - What kind of relationship do these characters have?
   - What would they say to each other?
   - What sound effects would make sense in this scene?
   - How could your story end?

6. When students are finished their storyboards, collect their worksheets. You will hand them back when you have computer time for the final project.

ACTIVITY 3 WORKSHEET

Animation storyboarding

NAME

This sheet will help you plan your animation. Use the boxes to sketch what each scene will look like and then write a description of what happens. This is called a storyboard.

STORY TITLE

SCENE 1 – INTRODUCTION

Where does this scene take place?
Who is the main character?
What happens?

SCENE 2 – CHANGE BACKGROUND

Where does the character go next?
What happens there?

SCENE 3 – NEW CHARACTER

What new character do we meet?
What do the characters say to each other?

SCENE 4 – ENDING

What do the characters do next?
How does the story end?

FINAL PROJECT WORKSHEET

____
NAME

Make your own animated short

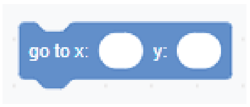
Now it's time to turn your storyboard into a Scratch animation using code!

Your story should include:

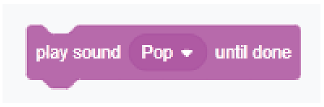
- ☐ 2 different sprites
- ☐ A backdrop change
- ☐ A conversation
- ☐ At least 2 types of motion
- ☐ Sound effects
- ☐ A repeat loop

Your code will have lots of parts, but be sure to use all these blocks:

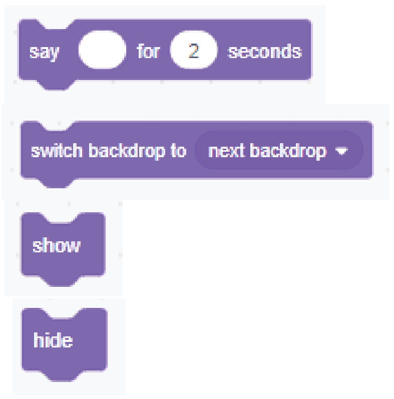
Motion



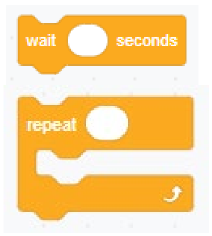
Sound



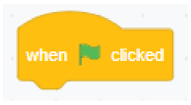
Looks



Control



Events



QUESTIONS

1. What's a problem you had while coding? How did you try to fix it?

2. What is one new coding block you used? What does it do?

3. If you had more time, what would you add to your story?

FINAL PROJECT

Make your own animated short

60 MINUTES +

Students will apply their coding knowledge to make their own animated short film. It's a fun opportunity to demonstrate what they've learned.

LEARNING OBJECTIVES

- Build Scratch code with many different block types
- Code with control structures, including loops or conditional statements
- Write the script and dialogue for an original story

CURRICULUM CONNECTIONS

- Math C3.1 & C3.2 (Coding)
- Media Literacy 3.4

SET-UP

Have students log onto computers and open their Scratch accounts. Hand back storyboard worksheets from Activity 3.

INSTRUCTIONS

SUMMARY

Students will be using their storyboards from Activity 3 and turning them into an animation in Scratch. At the end, students will answer questions on their worksheet, and save their code for you to assess.
See **ASSESSMENT** on **Page 5.47**.

1. Reiterate challenge to students. "You will be using everything we've learned about Scratch so far to make your own animated movie."
2. Hand back their worksheets and review the instructions on the back. It has a checklist of what elements to include in their animation and which Scratch blocks to use.

MATERIALS

Students will need:

- **Laptops or tablets**
- **Scratch accounts**
- **Worksheets from Activity 3**

3. Have students create a new Scratch project. Remind them to name their projects to make it easier to find later.
4. Give students the rest of the period to work on their projects. Encourage students to try out different blocks if they aren't sure how to do something. Remind them to follow their storyboard, but let them know it's okay to make changes if needed.
5. Scratch will save their projects automatically as long as they are logged into their account. If students need more time, consider using another computer period to finish.
6. **PROJECT SUBMISSION**
   Students will answer the reflection questions on their worksheet and hand it in. On Scratch, students will share their project, copy the link, and send it to you. Make sure to leave time (about 10 minutes) to do this.

   Check out our video tutorial for instructions on sharing Scratch projects.

FINAL PROJECT

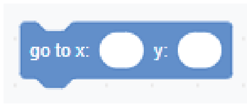
Assessment and evaluation

When students have sent you their Scratch projects, you can view their code by clicking the link and then "See Inside". If they have included it in their code, you will be able to start viewing their animation by clicking the green flag. You can start your assessment by checking that the students used the following elements in their animation:

- ☐ 2 different sprites
- ☐ At least 2 types of motion
- ☐ A backdrop change
- ☐ Sound effects
- ☐ A conversation
- ☐ A repeat loop

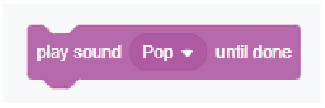
Next, you can look through their code to see which blocks they used. You can switch between sprites by clicking on the sprite names in the bottom right corner. The students were asked to use all of these blocks as a starting point:

Motion



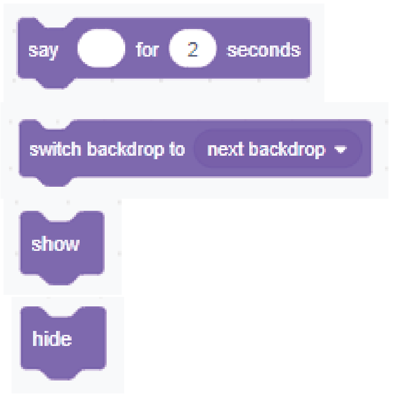
go to x: ( ) y: ( )

Sound



play sound Pop ▾ until done

Looks



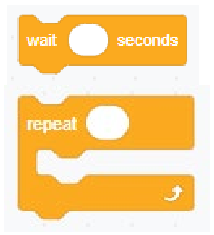
say ( ) for (2) seconds

switch backdrop to next backdrop ▾

show

hide

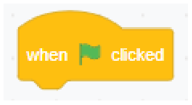
Control



wait ( ) seconds

repeat ( )

Events



when 🏁 clicked

Students will probably use more blocks than the ones listed above, but these are the ones specified on their activity worksheet.

Read through both their code and their worksheet responses, then use the **ASSESSMENT FRAMEWORK** on the next page to evaluate their work.

Assessment framework

This chart will help you assess your students' work during the Final Project and the Coding in the Classroom program as a whole. It is based on the Ontario Mathematics (2020) curriculum.

KNOWLEDGE AND UNDERSTANDING

	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
Knowledge of content	Uses few required block types in final code Does not answer questions during online classes, even with assistance	Uses most required block types in final code Answers questions during online classes with some assistance	Uses all required block types in final code Answers some questions during online classes	Uses all required block types and many others in final code Answers many questions during online classes
Understanding of content	Rarely uses control structures (i.e., repeat loops, conditional statements, wait commands) when appropriate	Sometimes uses control structures (i.e., repeat loops, conditional statements, wait commands) when appropriate	Often uses control structures (i.e., repeat loops, conditional statements, wait commands) when appropriate	Always uses control structures (i.e., repeat loops, conditional statements, wait commands) when appropriate

THINKING

	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
Use of planning skills	Creates storyboard with few of the required elements	Creates storyboard with some of the required elements	Creates storyboard with most of the required elements	Creates storyboard with all of the required elements
Use of processing skills	Uses code to convert storyboard into animation with limited effectiveness	Uses code to convert storyboard into animation with some effectiveness	Uses code to convert storyboard into animation with considerable effectiveness	Uses code to convert storyboard into animation with high degree of effectiveness
Use of critical/ creative thinking processes	Troubleshoots and "debugs" code with much assistance Does not experiment with blocks	Troubleshoots and "debugs" code with assistance Experiments with required block types	Troubleshoots and "debugs" code with some assistance Experiments with some new block types	Troubleshoots and "debugs" code with little assistance Experiments with many new block types

COMMUNICATION

	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
Expression and organization of ideas and information in oral, visual, and/or written forms	Uses some visual or written elements Does not clearly express a story	Uses either visual or written elements to express an animated story	Uses both visual and written elements to express an animated story	Combines visual and written elements to express a cohesive, animated story
Communication for different audiences and purposes in oral, visual, and/or written forms	Explains code and digital art, either orally or in writing, with limited effectiveness	Explains code and digital art, either orally or in writing, with some effectiveness	Explains code and digital art, either orally or in writing, with considerable effectiveness	Explains code and digital art, either orally or in writing, with a high degree of effectiveness

APPLICATION

	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
Application of knowledge and skills in familiar contexts	Follows coding lessons with much assistance	Follows coding lessons with assistance	Follows coding lessons with some assistance	Follows coding lessons with little or no assistance
Application of knowledge and skills to new contexts	Transfers coding knowledge to make animation with much assistance	Transfers coding knowledge to make animation with assistance	Transfers coding knowledge to make animation with some assistance	Transfers coding knowledge to make animation with little or no assistance
Making connections within and between various contexts	Rarely participates in offline coding activities Rarely makes connections between coding concepts and everyday life	Participates somewhat in offline coding activities Sometimes makes connections between coding concepts and everyday life	Participates in offline coding activities Makes connections between coding concepts and everyday life	Participates fully in offline coding activities Often makes connections between coding concepts and everyday life

Additional resources

SCRATCH

Scratch has a series of activity guides under the “Ideas” tab. There are also countless tutorials available on YouTube.

scratch.mit.edu/ideas

CANADA LEARNING CODE

From lesson plans to professional development, this website has a wealth of resources for teaching coding.

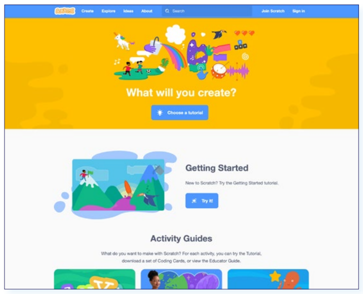
canadalearningcode.ca

BLOCKLY GAMES

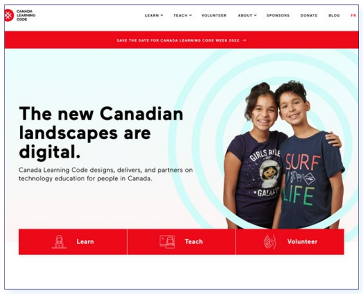
These coding games cover a range of topics. Blockly offers a mix of block-based and text-based coding. The later levels of some lessons are quite tricky, but the first few levels should be accessible for Grade 5 students.

blockly.games

*Coding screenshots are sourced from **scratch.mit.edu/**



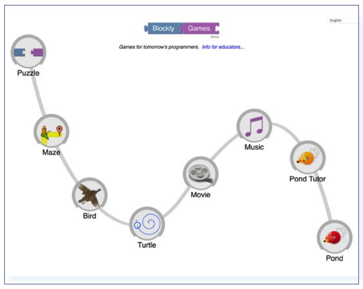
Scratch



The new Canadian landscapes are digital.

Canada Learning Code designs, delivers, and partners on technology education for people in Canada.

Canada Learning Code



Blockly Games