

GRADE 6 TEACHERS' GUIDE

Creating efficient code

Welcome to NII Explore's *Coding in the Classroom* program for Grade 6 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 6 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.

In Grade 5, students are introduced to two main types of control structure – **conditional statements** and **loops**.

The **GRADE 6** curriculum is focused on **efficient code**. Efficiency is about solving problems using the simplest code possible. Every time a computer executes a line of code, it takes time and energy. If we write shorter, more efficient code, our programs will run faster and require less computing power.

There are many techniques that can make our code more efficient, but we will focus on these three.

SCHEDULE AT A GLANCE

WEEK 1

- **Online Lesson 1**
Intro to Scratch
- **Offline Activity 1**
Conditional Rock-Paper-Scissors

WEEK 2

- **Online Lesson 2**
Make an Efficient Scratch Game
- **Offline Activity 2**
Mystery Drawings

WEEK 3

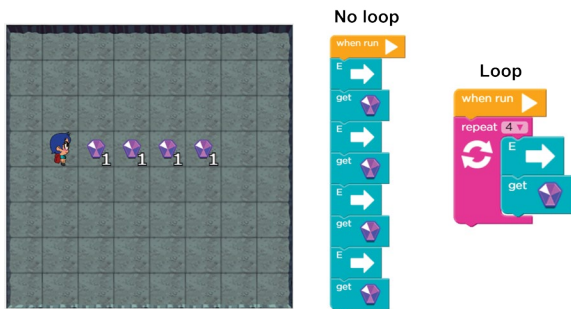
- **Online Lesson 3**
Dance Battle
- **Offline Activity 3**
Animation Storyboarding

WEEK 4

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

1. Use loops

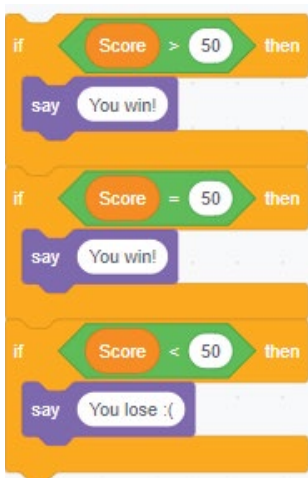
Loops make our code shorter by allowing us to repeat steps, rather than having to rewrite the same code multiple times. In this example, you can see that the code with a loop is much shorter than the one without.



Loop example. Both of these algorithms solve the maze, but the first solution requires 9 lines of code while the version with a loop uses only 4.

2. Use the right conditional statements

Efficiency is about minimizing the number of calculations that the computer needs to make. Let's say you've coded a game in which the player needs to score 50 or more points to win. You could evaluate the results using three "if-then" statements like this:



This code solves the problem, but the computer has to perform three separate calculations. It calculates if the score is *greater than 50*, then calculates if it is *equal to 50*, and finally calculates if it is *less than 50*. We can replace this code with a single "if-else" statement. Here's how:



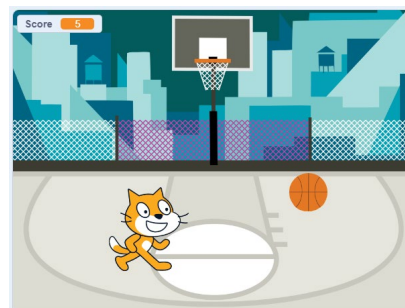
In this algorithm, the computer only needs to perform a single calculation – it checks if the score is greater than 49. If the statement is true, then the player wins. If it's not, the player loses.

This second version could run up to three times faster than the first example and use one-third of the computing power.

3. Put the code in the right order

The order of your code determines what your program will do, but it also affects your program's efficiency.

Consider this game made in Scratch. The cat chases the basketball and scores a point every time it touches the ball.



When the player reaches 10 points, the cat will say “You Win!”. The program needs an “if-then” block to check if the score is equal to 10. The algorithms below show two places where we could put that conditional statement, but one is much more efficient than the other. Can you figure out which one?



Option B is more efficient. In Option A, the “if-then” statement is right after the “forever” loop, which means that the computer will be checking the score constantly. In Option B, the check score statement comes immediately after changing the score. The computer only needs to check the score after a point is scored.

Most programs at a Grade 6 level will be small enough that efficiency isn’t a big problem, but it’s important to build good habits and develop problem solving skills. In the long run, efficient code will save us time, money, and energy.

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 some of the basic control structures to make their own “clicker” game.

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 – Conditional Rock-Paper-Scissors” before next online session.

See [Page 6.6](#) for lesson instructions and [Page 6.11](#) for a quick reference guide.

Offline Activity 1 – Conditional Rock-Paper-Scissors

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

See [Page 6.15](#) for activity instructions.

WEEK 2

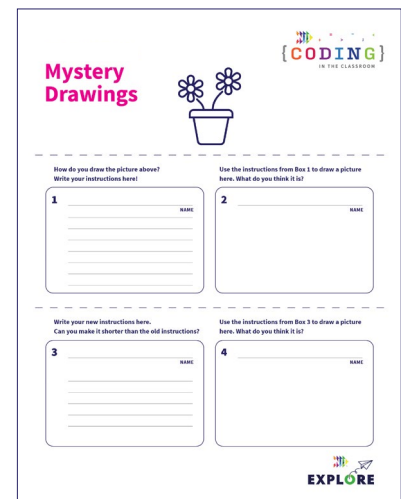
Online Lesson 2 – Make an Efficient Scratch Game

In this activity, students will edit an incomplete Scratch file to make a working game. Students will learn techniques to make their code more efficient.

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



Week 1 Conditional Rock-Paper-Scissors



Week 2 Mystery Drawings

POST Complete “Offline Activity 2 – *Mystery Drawings*” before next online session.

See **Page 6.20** for lesson instructions and **Page 6.24** for a quick reference guide.

Offline Activity 2 – Mystery Drawings

Your students will practice writing and editing efficient instructions with this drawing activity.

See **Page 6.29** for activity instructions.

WEEK 3

Online Lesson 3 – Dance Battle

Students will use control structures to create a “dance battle” animation. Students will learn how to use custom “My Blocks” in Scratch.

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 6.37** for lesson instructions and **Page 6.41** 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 6.45** for activity instructions.

WEEK 4

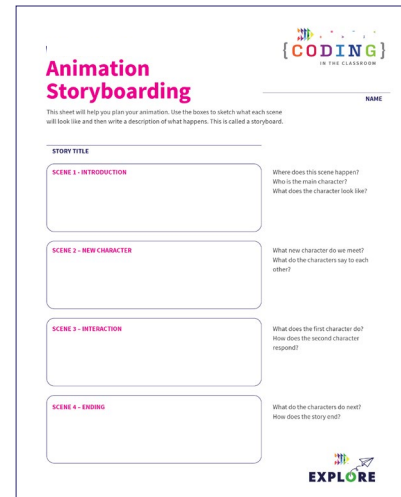
Final Project – Make Your Own Animated Short

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.

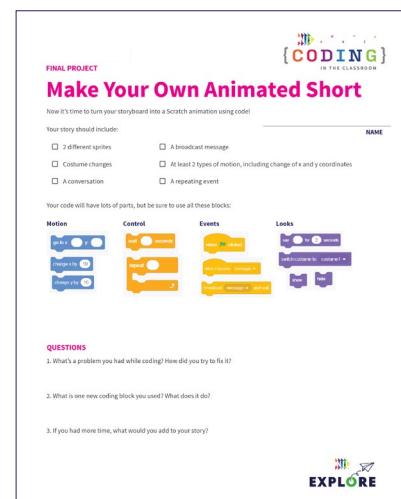
See **Page 6.49** for project instructions.

AFTER WEEK 4

Keep the coding going with the additional resources on **Page 6.54!**



Week 3 Animation Storyboarding



Week 4 Final Project

ONLINE LESSON 1

Intro to Scratch

(Clicker Game)

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 use motions, control structures (i.e., loops and conditional statements), and variables to make a simple clicker game.

CURRICULUM CONNECTIONS

CODING

- **Math C3.1** – solve problems and create computational representations of mathematical situations by writing and executing efficient 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 and the efficiency of the code

NUMBER SENSE

- **Math B1.3** – compare and order integers, decimal numbers, and fractions, separately and in combination, in various contexts

QUICK LINKS

Student Activity Link
scratch.mit.edu

Finished Example
scratch.mit.edu/projects/818521222

PowerPoint
Grade 6 – Week 1 – Clicker Game

Quick Reference
Intro to Scratch (Clicker Game) – 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 (Clicker Game) - 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 AND 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 student 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 basic clicker game to give students an idea of what they are working towards. You can make your own game or use this one.

Sample game:

scratch.mit.edu/projects/818521222

You can make the game full screen by clicking the four arrows icon. Click the green flag to start the game. Click the moving dog until you reach 10 points.

TOUR OF SCRATCH

NOTE: If students have used Scratch before, you can speed through this part.

Have students create a new project 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 6.12 for more details.**

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.

CREATE SCORE VARIABLE

We want to be able to keep score in our game. The score is something that can change or "vary" which is why we call it a variable. We use variables to store single pieces of data.

Go to Variables and then "Make a Variable". Name it "Score".

You can also delete the default "my variable" by right-clicking on it and choosing "delete".

INITIALIZE SCORE - STEP 1 IN QUICK REFERENCE

Ask students "What should the score be when we start a new game?" (Zero).

Add "**when green flag clicked**" to begin the program.

Then add "**set Score to 0**" right after the green flag block [Step 1].

Now the score will be reset to 0 whenever you click the green flag to start a new game.

SCORING POINTS - STEP 2 IN QUICK REFERENCE

Add "**when this sprite clicked**" then "**change Score by 1**" [Step 2].

The score will go up by 1 every time the target sprite is clicked.

TEST THE GAME

Have the students test the game – does the score go up? Does it reset to 0 when you start a new game?

When everyone is ready, run a friendly competition with the students. See who can get the most clicks in 10 seconds. Ask them to share their scores.

MOVING SPRITE - STEP 3 IN QUICK REFERENCE

To make the game harder, let's make the sprite move around the screen.

Under the green flag section, add a "**forever**" loop (found in Control) then put "**go to random position**" (found in Motions) inside the loop [Step 3a].

Have students test the game and see what happens. The sprite moves way too fast, but students find it funny.

SLOWING DOWN - STEP 3 CONTINUED

ASK: How can we make the sprite go slower?

HINT: Check out the options in Control.

Answer: Add "**wait 1 seconds**" after "go to random position" [Step 3b].

TEST AGAIN

Run another friendly competition to let the students test their games.

Testing the games regularly helps keep the students engaged, but it's also good practice to test code often to spot mistakes early.

SETTING TIMER - STEP 4 IN QUICK REFERENCE

We don't want our games to go on forever, so we will need to set a target score. The game should end when the player reaches this score (the target was 10 in the example we saw at the start of class).

To check the score, create a conditional statement as follows: **if (Score variable) = 10 then → hide → stop all** [Step 4a].

Ask the students: Where should we put this conditional statement? When does it make sense to check the score? (After scoring a point) Add the conditional statement under "change Score by 1".

Now the target sprite will disappear and stop moving when the target score is reached. To make the sprite appear again for a new game, add "**show**" at the start of the program after the green flag [Step 4b].

TEST

Have students do another test.

ASK: Why do you think we chose 10 as the target score for testing? (So that testing doesn't take too long. We can raise the target score after we know that the game works.)

OPTIONAL - ADD A "YOU WIN" MESSAGE [STEP 5 IN QUICK REFERENCE]

If time allows, students can add a "You Win" pop-up message when they reach the target score.

Create the sprite

First, we need to create a new sprite that says: "You Win" (or something similar). Hover over "Choose a Sprite" but choose the paintbrush instead of a sprite. Use the "T" to add text. Type a victory message like

"Good Game" or "You Win". Students can customize it by changing the colour, size, font, or by drawing in confetti [Step 5].

Drag the sprite to the desired position on the screen.

OPTIONAL - BROADCAST MESSAGES STEP 6 IN QUICK REFERENCE

Now that we've made the "You Win" sprite, we need to tell it when to appear. In Scratch, we use broadcast messages to send invisible signals between sprites. The moving sprite will be sending a secret message to the "You Win" pop-up sprite.

Sending the message

In the moving sprite's code, add **broadcast (message 1)** to the if-then statement before the **stop all** command. Rename the message as "winner" or something similar.

NOTE: This doesn't mean that the word "winner" will show up in our game – this name just helps us remember what the broadcast message does.

Receiving the message

Move over to the "You Win" sprite's code. You can toggle between sprites by clicking on the sprite name and icon near the bottom right of the screen. Add "**when I receive (winner) → show**" [Step 6b].

ASK STUDENTS: What's the last thing we need to add here? What should happen to this sprite when the game starts? (It should hide).

Add "**when green flag clicked → hide**" to the winner sprite [Step 6c].

FINAL TEST

Give students time to play through their games, try each other's games, or share with you and the rest of the class.

OPTIONAL ADD-ONS

Suggest some extra things students could add to their games.

- Change target score: What should the game go to?
- Change sprite size: Sprite gets smaller each time it's clicked or just starts and stays smaller
- Change game speed: Make the wait time shorter or longer
- Add sound effects: Make a sound every time the sprite is clicked (**NOTE:** This could quickly get annoying)
- Make sprite "pulse" when it gets clicked by setting size to 110%, waiting 0.05 seconds then returning to 100% size

SLIDE 6 - RECAP

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

Ask students to recall what they learned. "We used two types of control structure: loops and conditional statements. What does each one do?" (Loops make sections of code repeat. Conditional statements allow us to control (or select) what happens and when.)

See slide notes for more discussion points.

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 week. You will be making a new game, this time with a focus on efficiency.

QUICK REFERENCE

Intro to Scratch

(Clicker Game)

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 clicker game.

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. Open the finished example in a second tab.

QUICK LINKS

Student Activity Link

scratch.mit.edu/

Finished Example

[scratch.mit.edu/
projects/818521222](https://scratch.mit.edu/projects/818521222)

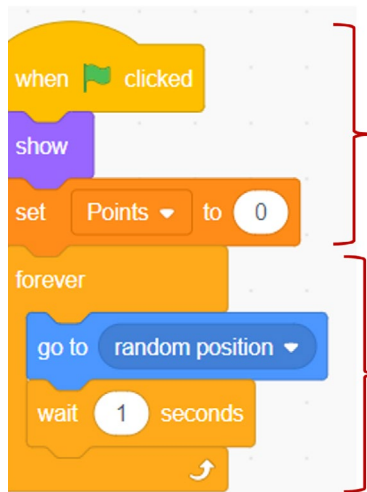
SCRATCH TOUR

The image shows the Scratch IDE interface with several components labeled:

- Switch between code and costumes:** Points to the 'Code' and 'Costumes' tabs at the top left.
- Rename project:** Points to the 'Untitled' text in the top blue header.
- Which sprite am I editing?:** Points to the small sprite icon in the top right of the workspace.
- Start/stop program:** Points to the green flag icon in the top right of the workspace.
- Preview window:** Points to the large white area on the right showing the current sprite.
- Show project full screen:** Points to the full-screen icon in the top right of the workspace.
- Coding block categories:** Points to the vertical list of colored categories on the left.
- Choose a coding block:** Points to a 'move' block in the 'Motion' category.
- Coding window:** Points to the main workspace area.
- List of sprites:** Points to the 'Sprite1' icon in the bottom right.
- Sprite properties:** Points to the 'Sprite1' panel in the bottom right.
- Choose a sprite:** Points to the 'Choose a sprite' button in the bottom right.
- Choose a backdrop:** Points to the 'Choose a backdrop' button in the bottom right.

THE CODE

Sprite #1 – Moving Sprite



```

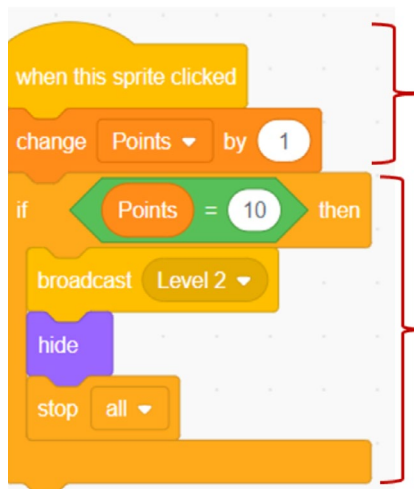
when green flag clicked
  show
  set Points to 0
  forever loop
    go to random position
    wait 1 seconds
  
```

Step 1. Initialization – When the game starts (green flag), the score is set to 0.

Step 4b. Add a **show** block to make the sprite reappear at the start of a new game.

Step 3a. The sprite goes to a random position and repeats that action forever.

Step 3b. Waiting 1 second before moving again makes the game more playable.



```

when this sprite clicked
  change Points by 1
  if Points = 10 then
    broadcast Level 2
    hide
    stop all
  
```

Step 2. The score will go up by 1 every time the sprite is clicked.

Step 4a. Each time a point is scored, the program checks if the score is equal to the target score (in this case 10). If it is, target sprite hides and stops all code. Without the stop command, the forever loop will keep running (i.e., the sprite will disappear but keep moving invisibly).

NOTE: Students might change the scoring increment or target score such that the target is never reached. For example, they might make the score go up by 2 each time while the target is an odd number. They can fix this with a > sign instead of an = sign.

Step 6a. To cue the Winner sprite, add a broadcast message after the target score is reached.

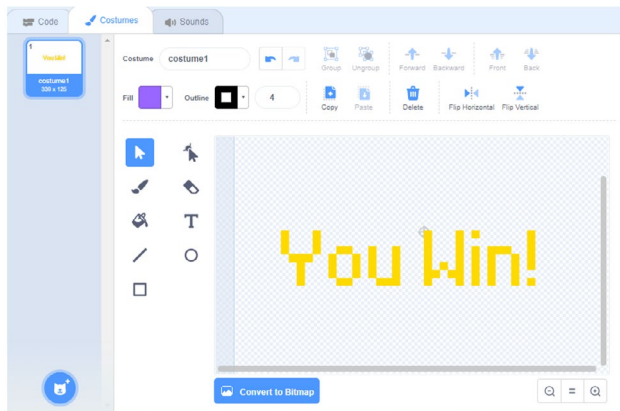
Sprite #2 – Winner Sprite



Sprite 6c. Sprite hides when the game starts.

Step 6b. Sprite appears when it receives the winner broadcast message.

Sprite #2 – Creating the Winner Sprite



Step 5. Hover over the “Choose a Sprite” icon then choose the Paintbrush option.

Select “T” for Text and type a message in the editing window.

Choose the arrow (“Select”) to drag a corner of the text box and make it bigger. You can also double click on the text box to change the fill colour and font.

In the preview window, drag the sprite where you want it to appear.

OFFLINE ACTIVITY 1

Conditional rock-paper-scissors

30 – 45 MINUTES

Practice evaluating conditional statements while engaging in some friendly competition!

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 6.17](#) to [6.19](#). 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 or in previous grades.
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.

MATERIALS

Each student will need:

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

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 1

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 1

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 1

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 2

Make an efficient game

In this lesson, students will be editing an existing Scratch file to make a catcher game. The game will include one sprite that moves horizontally along the bottom of the screen (catcher sprite) and a second sprite that repeatedly falls from the sky (falling sprite). Students will need to modify the code to make the game work and to make the program more efficient.

CURRICULUM CONNECTIONS

CODING

- **Math C3.1** – solve problems and create computational representations of mathematical situations by writing and executing efficient 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 and the efficiency of the code

NUMBER SENSE

- **Math B1.3** – compare and order integers, decimal numbers, and fractions, separately and in combination, in various contexts

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
- **Math E1.4** – describe and perform combinations of translations, reflections, and rotations up to 360° on a grid, and predict the results of these transformations

QUICK LINKS

Student Activity Link
scratch.mit.edu/projects/723255564

Finished Example
scratch.mit.edu/projects/818522286

PowerPoint
[Grade 6 - Week 2 - Video Games](#)

Quick Reference
[Make an Efficient Game - 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 **Make an Efficient Game – 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

Show students what the completed catcher game looks like to give them an idea of what they are working towards.

Link to finished game:
scratch.mit.edu/projects/818522286

Use the left and right arrows to move the frog back and forth. Try to catch the falling bugs until you reach the target score.

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 change the title to something of their choosing.

TEST THE GAME

Explain that their file is partially complete – they will need to make changes to make their code work and to make it more efficient.

Have students try the game to see what works and what doesn't (frog can only move to the right, bug doesn't fall).

ADD LEFT MOVEMENT - STEP 1 IN QUICK REFERENCE

Ask students to read the frog's code and identify what each part does. What part lets the frog move to the right?

Answer: if right arrow pressed? then → change x by 10 → point in direction 90

REMINDER: You can switch between sprites by choosing the appropriate one in the bottom right box. Make sure the frog is selected.

Based on the code for the right arrow, give students time to figure out what should go in the conditional statement for the left arrow. Let them try on their own but give hints if needed.

Answer: if left arrow pressed? then → **change x by -10** → **point in direction -90** [Step 1]

MAKE BUG FALL - STEP 2 IN QUICK REFERENCE

Switch to the bug sprite's code. Have students read through it and say what they think each line does (Starts game, resets score to 0, moves bug to random position at the top of the screen, i.e., y = 180).

NOTE: There is an unconnected section that begins with “if Score = 10 then...”. Tell students that you will be using that part later and to ignore it for now.

To make the bug fall, start by adding a **forever** loop to the bottom of the existing code. Next, ask the students what they could put inside the loop to make the bug fall down the screen.

Answer: change y by -10 [Step 2].

Have students test that this works by hitting the green flag. Common mistakes are mixing up x and y, or not using a negative. This is a good opportunity to teach/ remind students about the Cartesian coordinate system.

HAVE BUG RETURN TO THE TOP - STEP 3 IN QUICK REFERENCE

ASK “What’s the problem?” (The sprite only falls once – need to make it go back to the top and fall again).

We can do this using a conditional statement to check when the bug passes below a certain point on the screen. Add **if y position < -170 then** inside the forever loop. Common mistakes are missing the negative sign or using greater than instead of less than.

Next, we need to add instructions inside the conditional statement to send the bug back to the top. Which two blocks should we use? **HINT:** They are already part of our code.

Answer: go to random position → set y to 180 [Step 3].

Now whenever the sprite falls below a certain point it will move back to a random position at the top of the screen.

SCORING POINTS - STEP 4 IN QUICK REFERENCE

Next, we will start scoring points. A score variable has been pre-made for the students, but they will need to figure out when to use it.

Answer: Immediately after the first conditional statement add **if touching Frog then → change Score by 1 → go to random position → set y to 180.**

When the bug touches the frog the score will go up by 1 and the bug will return to a random position at the top of the screen.

PLAY TEST

At this point, the students’ games should be playable. Give them time to test it out and troubleshoot any issues.

SET TARGET SCORE - STEP 5 IN QUICK REFERENCE

If we don’t want the game to go on forever, we will need to set a target score.

Find the pre-written section of code that says **if Score = 10 then → broadcast winner → stop all.** We will be turning this entire section into a single block using a subprogram (also called a function). Subprograms are sections of code that perform a specific task. In this case, our subprogram will check if the score is equal to 10 and send the winner message if it is.

To turn this into a subprogram, go to “My Blocks” then “Make a Block”. Call this new block **check score** and click OK. A pink block called **define check score** will automatically appear in the coding window. Connect it to the top of the “If Score = 10 then...” section [Step 5a].

Now we need to tell our program when to check the score. Go to My Blocks and grab our new, pink **check score** block. Ask students to figure out where in their main program they should insert it. When is the best time to check the score?

Answer: There are several places that would work, but the most efficient spot is right after “change Score by 1”. We only need to check the score after the score changes [Step 5b].

PLAY TEST

The game should now be fully operational. Give students some time to test it out. If students have followed all the steps so far, a winner message has been pre-programmed to appear when the target score is reached.

If you are low on time, feel free to end the lesson here and give students the remaining time to test each other's games. Otherwise, you can go on to the optional next step.

OPTIONAL - MAKE "GO TO TOP" BLOCK - STEP 6 IN QUICK REFERENCE

One way to make our code more efficient is by putting steps in the ideal order. Another way is to eliminate redundancy. Have students look through the bug's code – is there anything that shows up multiple times? What? What does this code do?

Answer: go to random position → set y to 180; it makes the sprite go to a random position at the top.

We can replace this repetitive code with a subprogram or function. It makes our code easier to read and more efficient to write.

Go to My Blocks then "Make a Block". Title it "go to top". Under "**define go to top**" add "**go to random position → set y to 180**" [Step 6a].

Now we can replace those lines of code with the "**go to top**" block whenever they appear in our main code [Step 6b].

BONUS - CUSTOMIZE THE GAME

If time allows, you can give the students suggestions for customizing their games.

Change backdrop

Game could switch to a new backdrop when a certain score is reached.

Change sprites

Create the new sprite that you want to use then copy over the code from your existing sprite to your new sprite. You can copy code by dragging it to the new sprite's icon in the bottom right.

Change the game speed

Tweak the numbers to make the sprites (either frog or bug) move either faster or slower.

Add a second falling sprite

Create a second sprite and make it worth more points.

SLIDE 7 - RECAP

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

Ask students to recall what they learned. See slide notes for possible discussion points.

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. The students will use subprograms and repeat loops to make a fun animation.

Try **Offline Activity 2 – Mystery Drawings** before the next coding class.

QUICK REFERENCE

Make an efficient game

Students will be making a catcher game. The game includes a “catcher” sprite that the player controls with the arrow keys and a “falling” sprite that continuously falls from the top of the screen. Students will need to modify the existing code to make the game work and to make their program more efficient.

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

The code for the three sprites (frog, bug, and winner message) is shown below. Sections marked in **GREY** are pre-populated in the student version. You will be adding the steps marked in **PINK** during the class. The step numbers correspond with those in the lesson plan from **Pages 6.21 to 6.23**.

QUICK LINKS

Student Link
scratch.mit.edu/projects/723255564

Finished Example
scratch.mit.edu/projects/818522286

FROG SPRITE

```

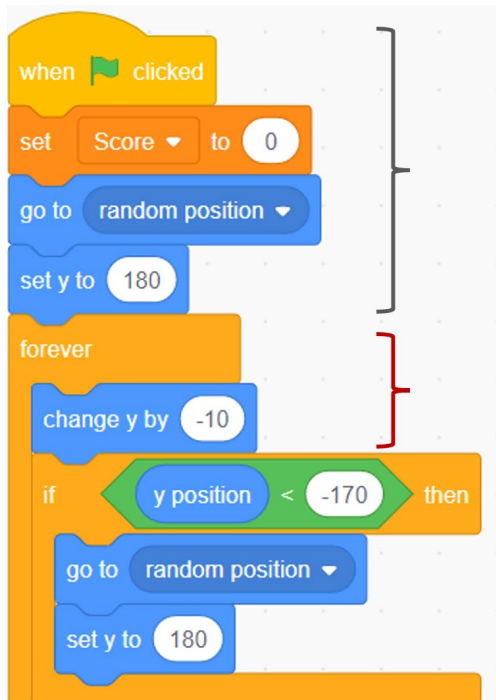
when green flag clicked
  go to x: 0 y: -140
  set rotation style to left-right
  point in direction 90
  forever loop
    if key right arrow pressed? then
      change x by 10
      point in direction 90
    if key left arrow pressed? then
      change x by -10
      point in direction -90
  
```

Initializes sprite position and orientation.

Moves the frog to the right when the right arrow key is pressed.

Step 1. Fill in this conditional statement to make the frog move left when the left arrow is pressed.

BUG SPRITE – MAIN PROGRAM



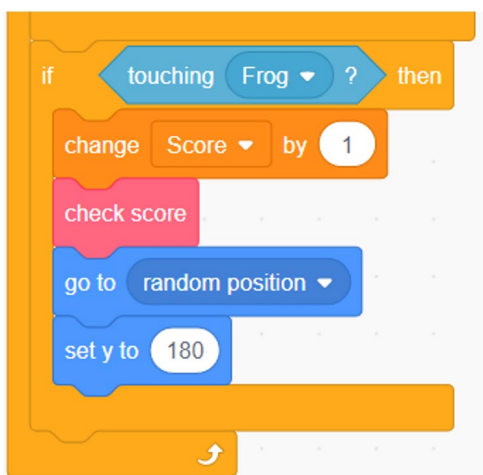
```

when clicked
  set Score to 0
  go to random position
  set y to 180
  forever
    change y by -10
    if y position < -170 then
      go to random position
      set y to 180
  
```

Sets the score to 0 at the start of the game, moves bug to a random position at the top of the screen.

Step 2. Makes the sprite fall. Code will repeat forever until told to stop.

Step 3. If the sprite ever reaches the bottom of the screen (i.e., has a y-coordinate less than -170), then send the sprite back to a random position at the top of the screen.



```

if touching Frog ? then
  change Score by 1
  check score
  go to random position
  set y to 180
  
```

Step 4. The player should score a point any time the bug touches the frog. The bug will then go back to a random position at the top.

Step 5b. Most efficient location for the “check score” block.

BUG SPRITE – CHECK SCORE SUBPROGRAM

```

define check score
  if Score = 10 then
    broadcast winner
    stop all
  
```

Step 5a. Connect the “define check score” block to the pre-written code below.

If the score variable is equal to 10, a “winner” broadcast message is sent and all code stops.

BUG SPRITE – “GO TO TOP” SUBPROGRAM

```

define go to top
  go to random position
  set y to 180
  
```

Step 6a. Create and define a custom “go to top” block.

BUG SPRITE – MAIN PROGRAM WITH “GO TO TOP”

```

when green flag clicked
  set Score to 0
  go to top
  forever loop
    change y by -10
    if y position < -170 then
      go to top
    if touching Frog ? then
      change Score by 1
      check score
      go to top
  
```

Step 6b. Replace redundant sections of code with the new **go to top** block.

WINNER MESSAGE SPRITE

Sprite hides at the start of the game and only appears when the winner broadcast message is received. This code is pre-written for the students.

```

when green flag clicked
  hide

when I receive winner
  show
  
```

OFFLINE ACTIVITY 2

Mystery drawings

30 – 45 MINUTES

Your students will practice giving and following efficient instructions with this fun, drawing activity.

LEARNING OBJECTIVES

- Write and execute instructions
- Follow and edit existing instructions
- Create 2D art works

CURRICULUM CONNECTIONS

- Math C3.1 & C3.2
- Arts D1

SET-UP

There are six versions of the worksheet, each with a different starting image. The worksheets can be found on **Pages 6.31 to 6.36**. Print some of each for your class. Each student will need a single sheet, not all six.

Students will also need a piece of paper and something to write and draw with.

INSTRUCTIONS

1. Ask students to recall what they learned in the online class.
2. Explain the premise of today’s activity. “Today, we are going to do some coding, but instead of writing instructions for a computer, we will write instructions for each other.”
3. Test the activity with your class. Without showing them the picture on the right, give the students instructions on how to draw it. The catch? You can only describe the shapes you’re drawing, not any specific things.



MATERIALS

Each student will need:

- 1 worksheet
- Something to write with

Here's what your "code" might be:

- Draw a medium-sized circle in the middle of the page
- Draw a smaller circle above it and then an even smaller one above that one
- Draw a rectangle above the smallest circle and shade it in
- On each side of the middle circle, draw a straight line pointing out at a 45° angle
- Add two short lines at the end of each of those lines
- Draw two dots and an upside-down semi-circle in the smallest circle

Have students follow along by drawing their picture on a blank scrap of paper.

- When you're done giving the instructions, have your students share their drawings with each other. **DISCUSS** Do they look the same? What do you think my original picture was?
- Hand out the worksheets. Remind students that the picture on their worksheet should be kept secret. Make sure that students sitting near each other have different images.
- Have students complete **Box 1** on their worksheet. They will be writing step by step instructions for drawing their image.
- When they're done writing instructions, have them fold their paper on the first dotted line to hide the original image, then pass their worksheet to the next person.
- Students will use the instructions written in Box 1 to draw a picture in **Box 2**. When everyone's done, let them unfold the paper and compare their drawing to the original image.

- Based on their results from Step 8, students will now change the code they received to make it more concise and efficient. They will write their revised code in **Box 3**. When they're done, have them fold their paper to hide the original image and drawing, then pass their papers to a new person.
- Students will now draw a picture in **Box 4** using the instructions from Box 3. When students are finished, they can unfold their paper and compare their drawing to the first drawing and the original picture.
- Allow the students to show their drawings to each other and discuss the activity among themselves.
- Wrap up the activity with a class discussion using the suggested discussion questions as a guide.

DISCUSSION QUESTIONS

How did your picture change from the original image to the final drawing?

What made the written instructions easy or hard to follow?

What changes did you make when you edited someone else's code?

Was it easier to write the first instructions or edit someone else's?

How does giving instructions to a person compare to writing computer code? Which one do you prefer?

Why should we practice giving instructions?

What's a time in our life when we might need to give someone instructions?

VERSION A

Mystery drawings



FOLD HERE

**How do you draw the picture above?
Write your instructions here!**

1 _____ **NAME**

**Use the instructions from Box 1 to draw a
picture here. What do you think it is?**

2 _____ **NAME**

FOLD HERE

**Write your new instructions here.
Can you make it shorter than the old
instructions?**

3 _____ **NAME**

**Use the instructions from Box 3 to draw a
picture here. What do you think it is?**

4 _____ **NAME**

VERSION B

Mystery drawings



FOLD HERE

**How do you draw the picture above?
Write your instructions here!**

**Use the instructions from Box 1 to draw a
picture here. What do you think it is?**

1 _____ **NAME**

2 _____ **NAME**

FOLD HERE

**Write your new instructions here.
Can you make it shorter than the old
instructions?**

**Use the instructions from Box 3 to draw a
picture here. What do you think it is?**

3 _____ **NAME**

4 _____ **NAME**

VERSION C

Mystery drawings



FOLD HERE

How do you draw the picture above?
Write your instructions here!

1

NAME

Use the instructions from Box 1 to draw a
picture here. What do you think it is?

2

NAME

FOLD HERE

Write your new instructions here.
Can you make it shorter than the old
instructions?

3

NAME

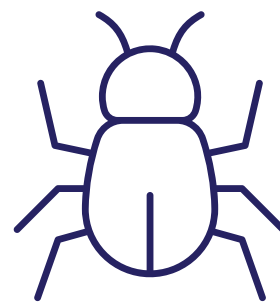
Use the instructions from Box 3 to draw a
picture here. What do you think it is?

4

NAME

VERSION D

Mystery drawings



FOLD HERE

**How do you draw the picture above?
Write your instructions here!**

1 _____ **NAME**

**Use the instructions from Box 1 to draw a
picture here. What do you think it is?**

2 _____ **NAME**

FOLD HERE

**Write your new instructions here.
Can you make it shorter than the old
instructions?**

3 _____ **NAME**

**Use the instructions from Box 3 to draw a
picture here. What do you think it is?**

4 _____ **NAME**

VERSION E

Mystery drawings



FOLD HERE

How do you draw the picture above?
Write your instructions here!

1

NAME

Use the instructions from Box 1 to draw a
picture here. What do you think it is?

2

NAME

FOLD HERE

Write your new instructions here.
Can you make it shorter than the old
instructions?

3

NAME

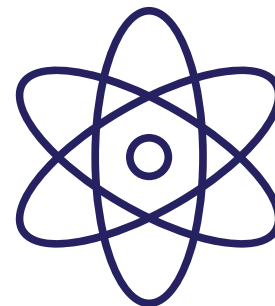
Use the instructions from Box 3 to draw a
picture here. What do you think it is?

4

NAME

VERSION F

Mystery drawings



FOLD HERE

**How do you draw the picture above?
Write your instructions here!**

1 _____ **NAME**

**Use the instructions from Box 1 to draw a
picture here. What do you think it is?**

2 _____ **NAME**

FOLD HERE

**Write your new instructions here.
Can you make it shorter than the old
instructions?**

3 _____ **NAME**

**Use the instructions from Box 3 to draw a
picture here. What do you think it is?**

4 _____ **NAME**

ONLINE LESSON 3

Dance battle animation

60 MINUTES

Students will animate a “dance battle” between an existing sprite and one that they make themselves. The lesson will cover repeat loops and subprograms (also known as functions or “My Blocks”). After this lesson, students will move on to the final project where they will create their own animation from scratch.

CURRICULUM CONNECTIONS

CODING

- **Math C3.1** – solve problems and create computational representations of mathematical situations by writing and executing efficient 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 and the efficiency of the code

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
- **Math E1.4** – describe and perform combinations of translations, reflections, and rotations up to 360° on a grid, and predict the results of these transformations

QUICK LINKS

Student Activity Link
scratch.mit.edu/projects/743978955

Finished Example
scratch.mit.edu/projects/818524309

PowerPoint
Grade 6 – Week 3 – Animation and Final Project

Quick Reference
Dance Battle Animation – Quick Reference

LESSON BREAKDOWN

SLIDE 1 - 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 **Dance Battle Animation - Quick Reference** for your own use during the lesson.

Ask students to recall what you discussed in the previous two lessons. What were the pink blocks called? (My Blocks or subprograms or functions) Why do we use them? (To make our code more organized and more efficient)

SLIDES 2 TO 7 - HISTORY MINUTE

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

FINISHED DEMO

Switch over to Scratch and show students what a completed dance animation might look like to give them an idea of what they are working towards.

Link to a finished animation:
scratch.mit.edu/projects/818524309

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 and then rename it to something of their choosing.

FIRST DANCE - SHOW STARTING CODE

Have students click the green flag to see what the existing program does (Anina talks to the student then performs a short dance).

PICK BACKDROP

The default backdrop is the Boardwalk, but we can start by picking a new one. Ask students to pick a backdrop – where will your dance battle take place?

PICK A DANCING SPRITE

Students will now choose a sprite to compete against Anina.

Go to “Choose a Sprite” and select the “Dance” filter along the top bar. Students can pick any of the dancers except Amon (he only has one pose or costume) and Ballerina (doesn’t have many moves).

Ask students to share which sprite they picked.

NOTE: We will be switching back and forth between Anina’s code and this new sprite’s code several times. Remind students how to switch between sprites by clicking the sprite icon near the bottom right corner of the screen.

INITIALIZE THE NEW SPRITE - STEP 1 IN QUICK REFERENCE

Set the initial size, position, direction, and costume for the new sprite. Students can look at Anina’s initialization code for reference.

It will look something like “**when green flag clicked** → **set size to** __% → **go to x:** __ **y:** __ → **point in direction** __ → **switch costume to (starting costume)**” [Step 1].

CREATE FIRST DANCE ROUTINE - STEP 2 IN QUICK REFERENCE

Switch back to Anina’s code and show the code for “anina dance 1” (It is contained in a subprogram/My Block). Explain that we will write a similar subprogram for their own sprite.

Select their sprite and navigate to the Costumes tab (top left corner).

“We can make a dance routine by switching through the different costumes. Each costume represents a different pose that this sprite can do. For our first dance, we will just cycle through all the possible poses.”

Create a My Block called “dance 1”. A **define dance 1** block will automatically appear.

Under define dance 1 add “**repeat __ times → next costume → wait 0.2 seconds**” [Step 2a].

In the repeat block, they will fill in the number of costumes that their sprite has. For example, Anina has 13 costumes, so her loop repeats 13 times. This code will cycle through all their costumes.

Add a **when I receive (your turn)** block and then attach the **dance 1** custom block. Click on this section of code to see what happens (The dance should play). They may also choose to add a **say** block to their code. Something like “Watch this, Anina!” [Step 2b].

TEST THE ANIMATION

Students are now ready to test everything they have so far. Have them click the green flag to start Anina’s script. When she’s done her dance, she will send the “your turn” broadcast message telling the other sprite to start.

SECOND DANCE ANINA - STEP 3 IN QUICK REFERENCE

Switch back to Anina’s code. Scroll to the right to show students that Anina has a second dance routine that is pre-written and stored in another subprogram. They can click on “define anina dance 2” to see what it does.

Have them add **anina dance 2** (found in My Blocks) to her main program after “anina dance 1” then click the green flag to run the program [Step 3].

CREATE SECOND DANCE - STEP 4 IN QUICK REFERENCE

Now students will create a second dance for their own sprite. Switch back to their sprite’s code and have them create a new block called **dance 2**. They will build a new dance routine by defining their new block. For this dance, they can use any combination of loops, costume changes, and wait times. Look at Anina’s second dance for reference [Step 4a].

They can test their dance routine as they go by clicking on their **dance 2** block. When they are finished, they can add dance 2 to their main program under “dance 1” [Step 4b]. Again, they can click the green flag to run the entire animation.

SHARING TIME

As students finish, ask for volunteers to share what they have so far.

THIRD DANCE ANINA - STEP 5 IN QUICK REFERENCE

Switch back to Anina and scroll to the right again to find Anina's third dance. Click on it to show what it does. Notice that this dance introduces other graphic effects. Also point out that this subprogram contains the "anina dance 1" subprogram. You can use subprograms within other subprograms (i.e., nesting). Add the "anina dance 3" block to her main program under the other two dances [Step 5].

FINAL DANCE - STEP 6 IN QUICK REFERENCE

Now it's time for the students to create a final dance for their own sprite. Have them create a My Block called "dance 3". Encourage them to use any combination of loops, costume changes, and wait commands plus at least one graphic effect (found in Looks) or some other kind of movement. They can also use one of their existing My Blocks inside this new one [Step 6a].

When done, they can add their dance subprogram into their main program [Step 6b] and watch the entire animation. They can also mix and match their three dance subprograms.

SLIDE 8 - RECAP

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

Ask students to recall what they've learned about so far. Some possible topics include: control structures (loops and conditional statements), subprograms (My Blocks or functions), Motion and Looks, or efficiency. See slide notes for additional information.

SLIDES 9 AND 10 - 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 today 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/819337693

NOTE: Student projects are not expected to be this long or detailed.

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

As a bonus, NII Explore is always accepting submissions to our "Scratch Film Festival" contest. Email your students' projects to explore@nii.ca - we hand out prizes periodically for some of our favourite animations!

SLIDE 11 - POEM OF THE DAY

Share the final Poem of the Day.

SLIDE 12 - 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

Dance battle animation

Students will animate a “dance battle” between an existing sprite and one that they make themselves. The lesson will cover repeat loops and subprograms (also known as functions or “My Blocks”).

SET UP

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

THE CODE

The step numbers here correspond with the instructions on **Pages 6.38** to **6.40**. Sections marked in **GREY** are pre-populated for the students. Steps marked in **PINK** will be added during the coding class.

QUICK LINKS

Student Activity Link
scratch.mit.edu/projects/743978955

Finished Example
scratch.mit.edu/projects/818524309

SPRITE 1 (ANINA) – MAIN PROGRAM

NOTE: Anina also has three pre-coded dance routines not shown here. See Scratch file.

The code for Sprite 1 (Anina) consists of the following blocks:

- Initialization:** when clicked, set size to 90%, go to x: -105 y: -29, point in direction 90, switch costume to anina stance.
- Dialogue:** say join Hey join username ! for 2 seconds, say Check this out! for 2 seconds.
- Dance:** anina dance 1, anina dance 2, anina dance 3.
- Broadcast:** say Now show me your moves! for 2 seconds, broadcast your turn.

Annotations and steps:

- Initialization – sets Anina’s starting size, position, orientation, and costume.
- Sprite talks to user before dance begins.
- Performs first dance.
- Step 3.** Add second dance to main program.
- Step 5.** Add third dance to main program.
- Sends broadcast message to cue **SPRITE 2** to start its dance.

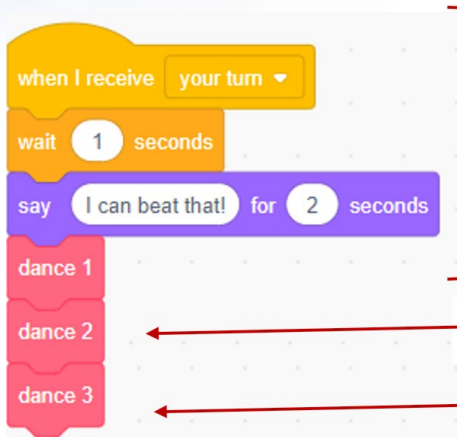
SPRITE 2 (STUDENT’S CHOICE)

NOTE: The messages in the “say” blocks are left up to the students.

The code for Sprite 2 (Student's Choice) consists of the following blocks:

- when clicked
- set size to 80%
- go to x: 131 y: -35
- point in direction 90
- switch costume to lb stance

Step 1. Initialize the new sprite’s size, position, direction, and costume.

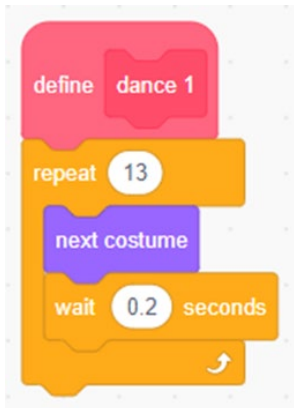


Step 2b. After receiving the “your turn” broadcast message from Anina, sprite can say something (optional) then perform first dance routine.

Step 4b. Add second dance into main program.

Step 6b. Add third dance into main program.

FIRST DANCE



Step 2a. Create a subprogram (My Block) for the first dance routine. This simple routine loops through all the sprite’s available costumes. **NOTE:** Number of repeats will depend on the number of costumes the sprite has (usually 12 or 13).

SECOND DANCE

Step 4a. Students can experiment with repeat loops, costume changes, and wait commands to make their own dance.

```

define dance 2
repeat 2
  switch costume to lb pop front
  wait 0.2 seconds
  switch costume to lb pop down
  wait 0.2 seconds
  switch costume to lb pop front
  wait 0.2 seconds
  switch costume to lb pop left
  wait 0.2 seconds
  switch costume to lb pop front
  wait 0.2 seconds
  switch costume to lb pop down
  wait 0.2 seconds
  switch costume to lb pop front
  wait 0.2 seconds
  switch costume to lb pop right
  wait 0.2 seconds
  
```

THIRD DANCE

Step 6a. Students can experiment with the same commands as dance 2 plus motion and graphic effects. They can also put previous pink blocks into this new one (nesting).

```

define dance 3
switch backdrop to Moon
switch costume to lb top L leg
repeat 10
  change size by -3
  change ghost effect by 5
  wait 0.1 seconds
point in direction pick random -180 to 180
repeat 150
  move 30 steps
  if on edge, bounce
set size to 80 %
go to x: 131 y: -35
point in direction 90
switch costume to lb stance
  
```

NOTE: These dance routines are examples. Students will come up with their own routines.

OFFLINE ACTIVITY 3

Animation storyboarding

30 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. They will use the front for this activity and the back for their final projects. Open this [animation example](#) on your computer to show your students. 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:

- Activity 3 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 character would they meet?
 - What kind of relationship do these characters have? What would they say to each other?
 - How do the characters interact with each other? How do they move?
 - 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 out 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 happen?
Who is the main character?
What does the character look like?

SCENE 2 - NEW CHARACTER

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

SCENE 3 - INTERACTION

What does the first character do?
How does the second character respond?

SCENE 4 - ENDING

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

FINAL PROJECT WORKSHEET

Make your own animated short

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

Your story should include:

NAME _____

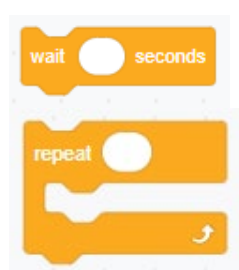
- 2 different sprites
- A broadcast message
- Costume changes
- At least 2 types of motion, including change of x and y coordinates
- A conversation
- A repeating event

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

Motion



Control



Events



Looks



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 efficient Scratch code with many different block types
- Code with control structures, including loops and 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 sheets 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 6.52](#).

1. Reiterate challenge to students. "You will be using everything we've learned about Scratch so far to make your own animated short."
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.
3. Have students create a new Scratch project. Remind them to name their projects to make it easier to find later.

MATERIALS

Each student will need:

- Laptop or tablet
- Scratch account
- Storyboard worksheet from Activity 3

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:

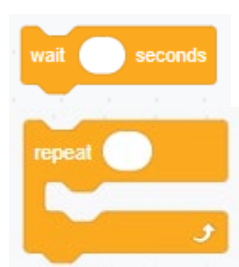
- | | |
|--|--|
| <input type="checkbox"/> 2 different sprites | <input type="checkbox"/> A broadcast message |
| <input type="checkbox"/> Costume changes | <input type="checkbox"/> At least 2 types of motion, including change of x and y coordinates |
| <input type="checkbox"/> A conversation | <input type="checkbox"/> A repeating event |

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



Control



Events



Looks



Students will probably use more blocks than the ones listed above, but these are the ones specified on their 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 some 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 Rarely looks for ways to make code more efficient	Sometimes uses control structures (i.e., repeat loops, conditional statements, wait commands) when appropriate Sometimes looks for ways to make code more efficient	Often uses control structures (i.e., repeat loops, conditional statements, wait commands) when appropriate Often looks for ways to make code more efficient	Always uses control structures (i.e., repeat loops, conditional statements, wait commands) when appropriate Always looks for ways to make code more efficient

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 animation, either orally or in writing, with limited effectiveness	Explains code and animation, either orally or in writing, with some effectiveness	Explains code and animation, either orally or in writing, with considerable effectiveness	Explains code and animation, either orally or in writing, with a high degree of effectiveness

APPLICATION

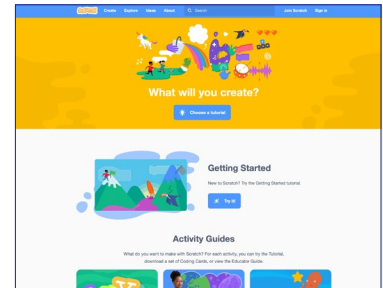
	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
Application of knowledge and skills in familiar contexts	Follows coding lessons with much assistance	Follows coding lessons with assistance	Follows coding lessons with some assistance	Follows coding lessons with little or no assistance
Application of knowledge and skills to new contexts	Applies coding knowledge to make animation with much assistance	Applies coding knowledge to make animation with assistance	Applies coding knowledge to make animation with some assistance	Applies 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

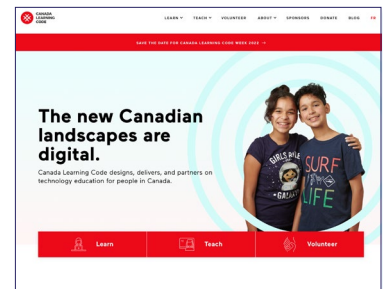


Scratch

CANADA LEARNING CODE

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

canadalearningcode.ca

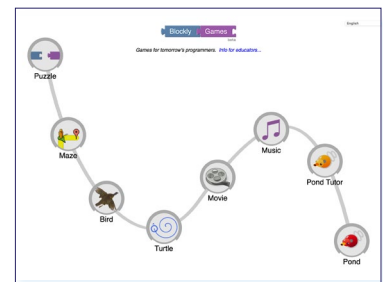


Canada Learning Code

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 6 students.

blockly.games



Blockly Games

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