CODING IN THE CLASSROOM

GRADE 2 TEACHERS' GUIDE

Coding with concurrent events

Welcome to NII Explore's *Coding in the Classroom* program for Grade 2 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 2 CODING CURRICULUM

As of 2020, Ontario's math curriculum includes coding expectations. Put simply, coding is when we make 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 **GRADE 2**, students learn to code with both sequential and concurrent events.

Sequential events are instructions that happen in a specific order.
We use sequential events to solve coding problems and in our everyday lives.
Here's an example of each:

Sequential event examples. *The coding puzzle and solution and the morning routine on the left are both examples of sequential events. They are a series of steps or instructions that happen in a certain order.*

SCHEDULE AT A GLANCE

WEEK 1
- **Online Lesson 1** Programming with Angry Birds
- **Offline Activity 1** Robot Teacher

WEEK 2
- **Online Lesson 2** Debugging in Maze
- **Offline Activity 2** Morning Sequences

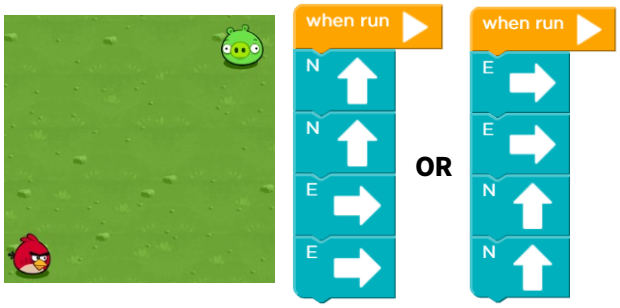
WEEK 3
- **Online Lesson 3** Making Sprites
- **Offline Activity 3** Simultaneous Simon Says

WEEK 4
- **Final Project** Sprites in Action

EXPLORE

In both coding and in everyday life, sometimes the **exact order** of the steps matters…

… and sometimes there are **multiple sequences** that work.



Sequencing is the foundation of all computer programming which is why we're starting here.

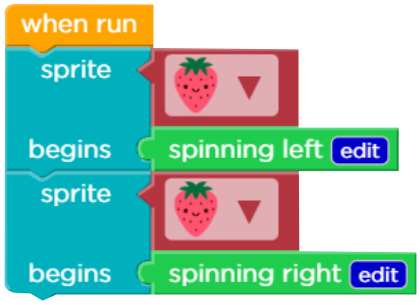
Concurrent events are things that happen at the same time. As humans, we are often multi-tasking. Take dance, for example. You need to listen to the music and move different parts of your body all at the same time.

Dancing is an example of concurrent events. We might move our feet at the same time that we move our arms.

With the correct code, computers can execute concurrent events too:

***Concurrent events.** When this program runs, the sprite will start wobbling then, without stopping the first instruction, it will also start moving east.*

Computers can only do exactly what we tell them to do, and some events cannot happen concurrently. In the example below, the animated character (or sprite) doesn't move. See if you can figure out what's wrong with the code!



***Some events can't happen concurrently.** In this example, we tell the sprite to start spinning right without first telling it to stop spinning left. These two events cancel each other out and the sprite doesn't move.*

Concurrent events are an important part of our daily lives. Can you imagine if you couldn't sleep and breathe at the same time? That would be quite inconvenient! Concurrent events are also an essential coding skill, especially for making things like animations and games.

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 – Programming with Angry Birds

This lesson introduces students to block-based coding and sequential events. Students will write code to guide an Angry Bird through a series of mazes.

PREP Log students onto computers and open activity link.

POST Complete "Offline Activity 1 – *Robot Teacher*" before next online session.

See **Page 2.5** for lesson instructions.

Offline Activity 1 – Robot Teacher

Students will have to guide their "robot teacher" through a series of seemingly simple tasks.

See **Page 2.9** for activity instructions.

WEEK 2

Online Lesson 2 – Debugging in Maze & Collecting Treasure with Laurel

The first activity introduces students to **debugging** – the process of finding and fixing mistakes in computer code. If time allows, try the second activity for extra coding practice.

PREP Same as Week 1.

POST Complete "Offline Activity 2 – *Morning Sequences*" before next online session.

See **Page 2.11** for lesson instructions.

Offline Activity 2 – Morning Sequences

Students will make an ordered sequence for their morning routines.

See **Page 2.14** for activity instructions.

WEEK 3

Online Lesson 3 – Making Sprites

Students will learn how to create, customize, and animate characters called sprites.

PREP Same as Week 1.

POST Complete "Offline Activity 3 – *Simultaneous Simon Says*" and the Final Project.

See **Page 2.16** for lesson instructions.

Offline Activity 3 – Simultaneous Simon Says

Practice concurrent events with this fun twist on an old classic.

See **Page 2.19** for activity instructions.

WEEK 4

Final Project – Sprites in Action

Students will build on what they learned in Week 3 to complete another sprite lesson and make their own animation.

See **Page 2.21** for project instructions and **Page 2.23** for assessment criteria.

AFTER WEEK 4

If you haven't already done so, try the rest of the offline activities then keep the coding going with the additional resources on **Page 2.25**!

ONLINE LESSON 1

Programming with Angry Birds

60 MINUTES

The three online lessons and final project all use **code.org** and block-based coding. If you are new to code.org, you can check out their **teacher resources** and try the lessons for yourself before each class.

This lesson will introduce students to the basics of block-based coding. Students will practice sequencing to help an Angry Bird find its target. The levels include a mix of writing code and editing existing code.

QUICK LINKS

Student Activity Link
studio.code.org/s/coursec-2022/lessons/3/levels/1

PowerPoint
Grade 2 – Week 1 – Intro to Coding

CURRICULUM CONNECTIONS

CODING

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

PATTERNS AND RELATIONSHIPS

- **Math C1.2** – create and translate patterns using various representations, including shapes and numbers

GEOMETRIC AND SPATIAL REASONING

- **Math E1.5** – describe the relative positions of several objects and the movements needed to get from one object to another

LESSON BREAKDOWN

SLIDE 1 - SET UP AND INTRODUCTION

Open the PowerPoint slides and the code.org activity on your own computer. Project for the students to see.

Get the students logged onto their computers with the activity link open. They may get a pop-up message that says: "You are not signed in". They can dismiss this message by clicking anywhere else on the screen. When everyone is ready, have them turn their attention to you for a quick discussion before the coding begins.

SLIDE 2 - WHAT IS CODING?

Check if your students have coding experience.
Ask them what coding means.

Use slides to explain what coding is -> "Coding is when we give instructions to a computer".

SLIDE 3 - WHAT CAN YOU DO WITH CODE?

Ask students what they think – why are we learning to code? What can we use it for?

Use slides to share possible answers.

SLIDE 4 - THE CODER'S CODE

Have students recite the "Coder's Code" by repeating after you.

Helps set expectations for the students.

LEVEL 1

Switch your screen over to code.org.

Watch the introductory video together.
(Level 1 of the lesson)

LEVEL 2

Proceed to Level 2. Give students a tour of the code.org layout – where do we find everything?

Levels are listed in the bar along top, preview window is on the left, blocks are in the middle, workspace is on the right, instructions are above the blocks and workspace. You should take some time to familiarize yourself with the layout before you teach these lessons.

NOTE: For the most part, your students won't need to read the instructions, but some may need help reading the words on the coding blocks (e.g., move forward)

LEVEL 2 - CONTINUED

Do Level 2 as a class. Explain the goal of the level, read the instructions together. Help students read the instructions on the blocks as needed.

Demo how to connect blocks and run the program.

NOTE: You will probably want students to turn down their computer volume or mute their speakers entirely. You can also make your preview window larger (i.e., the area where the Angry Bird appears) by dragging the right side of the window.

LEVEL 3

Read instructions together. Show students how to add extra blocks from the "Blocks" section.

Give students a chance to try it and then take up the solution together.

LEVEL 4

Explain that the grey blocks can't be deleted. Show that we are using "6/5 blocks". Ask "How many blocks do we need to take away?"

Give students time to "debug" (fix) their code by finding the block they need to remove. If needed, show students how to remove blocks by dragging.

LEVEL 5

Practice left vs right. E.g., "Everyone turn your head to the left. Now turn to the right." If you have any tricks to help students remember, now is a good time to introduce them.

Encourage students to try the level on their own. If their Angry Bird turns the wrong way, they can try changing left to right or vice versa. Point out that the answer should have 6 blocks.

Take up the level as a class.

LEVELS 6 & 7

If students are getting the hang of it, let them work at their own pace through these next two levels. Remind students to check the target number of blocks for each level as a hint.

NOTE: You can monitor students' progress by checking the bar at the top of their screen. A solid green dot shows that they fully completed a level. A partially filled dot means that they solved the level but didn't use the intended method. For example, they may have used more than the recommended number of blocks.

LEVEL 8

Show students how to use the Step button to read through their code one line at a time. Remind students that grey blocks can't be deleted, only rearranged. If students have trouble, do the level together using the step button.

LEVEL 9

Let students try Level 9 on their own.

LEVEL 10

Explain that Level 10 is a prediction question. Read through the code together one line at a time. Have students imagine/visualize what is happening to the Angry Bird each step. Then read all the answer choices together. Have the class vote on the answer.

LEVEL 11

Let students try Level 11 on their own. If there is extra time, you could show them what the repeat block does. If you're unsure about it, test it for yourself first.

NOTE: The concept of repeating events isn't formally introduced until Grade 3.

SLIDE 6 - RECAP

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

Ask students -> What did we learn about today? What does coding mean? [Giving instructions to computers]

Making programs

A program is a bunch of code that does something. When we code, we are writing programs.

Fixing mistakes

Sometimes our Angry Bird crashed into the wall because our code had mistakes. A big part of coding is finding those mistakes and fixing them (also called debugging)

Turning left and right

Not every puzzle is straightforward, sometimes we have to turn!

See slides for other talking points.

SLIDE 7 - POEM OF THE DAY

Introduce the concept of the poem of the day, read poem -> “We will end every lesson with a poem to help us remember what we learned about”.

SLIDE 8 - WHAT’S NEXT?

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

OFFLINE ACTIVITY 1

Robot teacher

20 MINUTES

Students will practice giving step-by-step instructions to their "robot" teacher.

MATERIALS

None required

LEARNING OBJECTIVES

- Practice giving sequential instructions
- Use speaking skills to clearly communicate information

CURRICULUM CONNECTIONS

- Math C3.1 & C3.2 (Coding)
- Oral Communication 2.3 & 2.4

SET-UP

Read through this lesson plan and collect optional materials if using. (See Step 5)

INSTRUCTIONS

SUMMARY

As a class, students will tell you (or another adult) how to perform a task like drawing a picture. The trick is that, like a robot, you will follow their instructions literally. The goal is for students to practice giving very specific instructions.

1. Ask students to recall what they learned about in the online class.
   ASK What is coding? [Giving instructions to a computer]
2. Explain premise of today's activity. "Today, we are going to do some coding, but instead of writing instructions for a computer, you will be giving me instructions. I will pretend to be a robot and follow your instructions exactly."
3. Grab a marker or chalk and an eraser then head to the board. Tell students that you'd like to draw a stick figure (or something of your choosing), but as a robot, you don't know what it's supposed to look like.

4. Pick some volunteers to tell you how to draw your picture one step at a time. Here's the trick that makes this game fun: intentionally make mistakes whenever the instructions aren't specific enough.

   For example, if a student tells you to "draw a circle", make your circle really big (or really small) or draw it in the wrong position on the board. If the students want you to re-draw something, they will need to tell you to erase the old part first. The point is to encourage the students to make their instructions as detailed as possible.

5. Try the activity again with a **different picture.**

   Variation

   Instead of drawing pictures, you could have students give you instructions for making or building something. For example, you could bring in the ingredients for a sandwich and ask the students for step-by-step help making it. Or you could bring in a toy that needs assembly and ask for assistance.

6. Wrap up the activity using some of the discussion questions as a guide.

DISCUSSION QUESTIONS

What was the problem when we first started this activity?

Possible answer: Our "robot" wasn't doing what we thought it would do.

Why do you think it's so important to give specific instructions to computers?

Answer: Computers are powerful, but not very smart. They can only do exactly what we tell them to do.

What's it called when we give instructions to a computer?

Answer: Coding!

ONLINE LESSON 2

Debugging in maze & collecting treasure with Laurel

60 MINUTES

Students will continue coding with sequential events as they practice debugging code. The debugging process connects to the curriculum expectation of reading and altering existing code. If time allows, your class can try the second activity to reinforce what they've learned so far.

QUICK LINKS

Debugging in Maze
studio.code.org/s/coursec-2022/lessons/4/levels/1

Collecting Treasure with Laurel
studio.code.org/s/coursec-2022/lessons/5/levels/2

PowerPoint
Grade 2 – Week 2 – Debugging

CURRICULUM CONNECTIONS

CODING

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

PATTERNS AND RELATIONSHIPS

- **Math C1.2** - create and translate patterns using various representations, including shapes and numbers

GEOMETRIC AND SPATIAL REASONING

- **Math E1.5** - describe the relative positions of several objects and the movements needed to get from one object to another.

LESSON BREAKDOWN

SET UP

Open the PowerPoint slides and the code.org activities on your own computer. Project for the students to see.

NOTE: There are two activities this week because each one is somewhat shorter.

Get the students logged onto their computers with the activity link open.

SLIDE 1 - WEEK 1 RECAP

Ask students to recall what they learned about in the first coding lesson -> how to connect blocks, how to run a program.

SLIDES 2 & 3 - READY TO START - CODER'S CODE

OPTIONAL: Choose a point from the Coder's Code to emphasize this week .[slide is hidden in the PowerPoint file if you want to use it]

Switch your screen to code.org. Make sure students have the right lesson open. [Start with Debugging in Maze]

DEBUGGING IN MAZE - LEVEL 1

Watch the introductory video together (Level 1 of the lesson). The video explains what debugging is.

LEVEL 2

Explain that you will be debugging today which is what we call it when we fix mistakes in code. For each level, there will be some mistake that we need to fix. Ask if students can figure out what is wrong in this level.

LEVELS 3 & 4

Explain that there are different things that could be wrong. Sometimes the code is missing a step and sometimes there are extra steps. Let students try the next two levels on their own.

LEVEL 5

Remind students about left vs right. Encourage them to use the step button to help spot the mistake in their code. The step button allows us to execute the code one block at a time and makes it easier to spot our mistakes.

LEVEL 6

Remind students that grey blocks cannot be deleted, only rearranged.

LEVEL 7

Point out the target number of blocks. Ask "How many do we need to remove?" Encourage them to use the step button again.

LEVEL 8

Let students try this challenge level on their own then take up the answer as a class.

LEVEL 9

Level 9 is a prediction question. Read through the code together one line at a time. Have students imagine/ visualize what is happening to Scrat each step. Then read all the answer choices together. Have the class vote on the answer.

LEVEL 10

Have students try this level on their own.

SLIDES 4 TO 10 - INTERMISSION: HISTORY MINUTE

When you are roughly halfway through the period, take a break to share the history minute about Grace Hopper and the origin of "debugging". You will need to switch back to the PowerPoint slides. There are speaking points in the slide notes.

It's a good opportunity to check how students are doing and re-focus their attention.

COLLECTING TREASURE WITH LAUREL - LEVEL 2

After students finish Level 10 of "Debugging in Maze", they will be taken to this new lesson automatically. Alternatively, you can have them open the second activity link.

Level 1 is an introductory video that students can skip.

In Level 2, show them how to make Laurel move forward and collect a piece of treasure. Note that this is a demo level – they only need to collect a single gem to pass the level.

LEVELS 3 ONWARDS

Give students the rest of the time to work through these levels at their own pace. Let them know it's okay if they don't finish them all. We have included this second lesson because the Debugging lesson tends to finish early.

SLIDE 11 - RECAP

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

Ask students -> What did we learn about today? [Debugging] What does debugging mean? [Fixing mistakes in our code] What are some tricks we can use to help us debug? [step button, "rubber duck" debugging]

SLIDE 12 - POEM OF THE DAY

Share this week's Poem of the Day.

SLIDE 13 - WHAT'S NEXT?

Give a quick preview of next lesson -> "We will learn how to make characters called sprites and then animate them"

You can also try **Offline Activity 2 – Morning Sequences** before the next online lesson.

OFFLINE ACTIVITY 2

Morning sequences

30 MINUTES

Students will create an ordered sequence for their morning routine then try to sort each other's steps.

LEARNING OBJECTIVES

- Understand sequencing of steps
- Communicate ideas through pictures and writing

CURRICULUM CONNECTIONS

- Math C3.1 & C3.2 (Coding)
- Arts D1
- Writing 1.5 & 1.6

SET-UP

Gather the required materials. You may choose to pre-cut the paper into quarters.

MATERIALS

Each student will need

- A piece of blank paper
- Something to write and draw with
- Scissors

INSTRUCTIONS

SUMMARY

Students will write and illustrate an ordered sequence of events to represent their morning routine. They will then trade with a partner and try to put each other's sequences in the right order.

1. Ask students to recall what they learned about in the online class. **ASK** What's it called when we fix mistakes in our code? [Debugging] What were some mistakes we had to debug? [Missing or extra steps, steps in the wrong order]
2. Explain premise of today's activity. **SAY** "Just like in coding, we often do steps in a certain order. Today, you are going to draw a routine (or program) to show us what you do in the morning."
3. Hand out paper and have students divide their page into quarters. You may choose to pre-cut their pages into quarters if you have a paper slicer.

4. Students will now pick four different things they do in the morning. For example, "brush teeth, get dressed, ride bus, watch TV". They will draw and label these activities in the four quarters of their page.
5. When students are finished, they will cut their page into quarters. Students should now have four cards, each with a different activity on it.
6. Ask students to put their four cards in the correct order based on their own morning routine. To use the example above, the sequence might be **"watch TV -> brush teeth -> get dressed -> ride bus"**. After you've checked their order, flip over their cards and write the numbers 1 to 4 on the backs (i.e., Card 1 has a 1 on the back, Card 2 has a 2, etc.). These numbers will serve as an answer key.
7. Have students mix up the order of their four cards and then trade with a partner. Alternatively, you could collect each student's set and facilitate the exchanges yourself.
8. Students will now try to find the correct sequence for their classmate's cards. They can check their answer by flipping over the cards to see the numbers on the back.
9. You may choose to repeat Steps 7 and 8 for several rounds. When done, debrief the activity using the following discussion questions as a guide.

DISCUSSION QUESTIONS

Did anyone have a different order than you expected? Why do you think that is?

Possible answer: There are many correct sequences for our morning routines.

In coding, what's it called when you make a mistake with the order of your steps? What's it called when we fix it?

Answers: a bug, debugging

Can you think of any morning tasks that have to be done in the right order?

What about ones where the order doesn't matter?

Possible answers: You have to wake up before you can get dressed, but you can brush your teeth before or after you get dressed.

ONLINE LESSON 3

Making sprites

60 MINUTES

Students will learn how to create, customize, and animate sprites. There are not many code.org lessons that deal with concurrent events, but the "behavior" blocks in this lesson allow sprites to perform multiple actions at the same time. The blocks in this lesson involve a fair amount of reading, so be prepared to help students in that regard.

QUICK LINKS

Student Activity Link
studio.code.org/s/coursef-2022/lessons/3/levels/2

PowerPoint
Grade 2 – Week 3 – Animation & Final Project

CURRICULUM CONNECTIONS

CODING

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

GEOMETRIC AND SPATIAL REASONING

- **Math E1.5** - describe the relative positions of several objects and the movements needed to get from one object to another

LESSON BREAKDOWN

SET UP

Open the PowerPoint slides and the code.org activity on your own computer. Project for the students to see.

Get the students logged onto their computers with the activity link open.

SLIDE 1 - WEEK 2 RECAP

Ask students to recall what they learned about in Week 2 -> debugging, sequencing.

SLIDES 2 TO 7 - HISTORY MINUTE: PIXAR

Take a minute to share this history minute about Pixar and computer animation. See slide notes for talking points.

SLIDES 8 & 9 - READY TO START

Make sure students have the right lesson open.

The Coder's Code is hidden in the slides if you want to refer to it, otherwise feel free to go right into the lesson.

LEVEL 2

Watch the introductory video (Level 2). We recommend skipping Level 1 because it jumps ahead to a more advanced topic without first providing context.

LEVEL 3

IMPORTANT NOTE: Code.org uses cookies to remember your progress if you tried this lesson beforehand. After opening the activity link, you will have to click "Version History" and then "Start Over", otherwise your previous answers will be visible. You will have to do this on every level.

Give students a tour of the coding area and point out what's different. The biggest change is that the "Blocks" section now has a menu of options to choose from.

As a group, demonstrate how to add a new sprite. Let students use the dropdown menu to change the appearance of their sprite. Ask for some volunteers to share which sprite they chose.

LEVEL 4

Point out the new "Location" tab and then demonstrate how to change the location of the sprite. If students are readers, they can try reading the instructions for themselves.

LEVEL 5

Explain the goal to the students "We will be making four different sprites and putting one in each corner." Click the example image to show students what you mean.

Remind students where to find the blocks they will need. Some students will understand right away, others will need some hands-on support.

If some students finish faster, show them that they can also customize their background.

LEVEL 6

Show students the new “set size” block and where it is found. Explain the challenge to them then give them time to try it. Have students experiment with different sizes. How big can their sprite get? How small?

LEVEL 7

Show students where to find the new “behaviors” blocks and explain what they do. Let students experiment with different behaviors. If they finish before others are ready, challenge them to make their sprite do multiple behaviors at once (i.e., concurrent events).

LEVEL 8

Level 8 contains four separate challenges. Briefly explain the goal of each one, before asking the students to start on A.

Once in Level A, click on the image in the instructions section to show students what their end goal looks like. Remind them to check the image if they are ever unsure of what they’re supposed to be making.

Give students time to work through the four levels on their own, offering instructions and support as needed.

OPTIONAL - LEVEL 9

If time allows, have students skip ahead to Level 9. This is a free-play level where students can make their own animation using sprites. Invite students to share what they make with you.

SLIDE 10 - RECAP

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

Ask students -> What have we learned about coding so far?

Programs

We make programs by putting instructions in a certain order. We might also call this an algorithm.

Debugging

Debugging is when we fix mistakes in our code.

Sprites

Sprites are characters/images we can control with code. We can set their size and location, and even make them perform multiple behaviors at once!

SLIDE 11 - PREVIEW THE FINAL PROJECT

Students will try one more code.org lesson, mostly on their own.

Explain that they will be doing more work with sprites and learning about new things that sprites can do.

SLIDE 12 - POEM OF THE DAY

Share the final Poem of the Day.

SLIDE 13 - 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!

OFFLINE ACTIVITY 3

Simultaneous Simon Says

20 MINUTES

Practice following concurrent events with a new twist on a classic game.

MATERIALS

None required

LEARNING OBJECTIVES

- Understand concurrent events
- Perform movement skills

CURRICULUM CONNECTIONS

- Math C3.1 & C3.2 (Coding)
- Health and Physical Education C1 & C2

SET-UP

This activity requires some space. Consider moving to the gym or re-arranging the classroom.

INSTRUCTIONS

SUMMARY

In a normal game of Simon Says, students only perform one motion at a time. In this version, students will have to follow multiple instructions at once.

1. Ask students to recall what they learned about in the online class. **ASK** What are sprites? [Characters we can control with code] What "behaviours" did we have our sprites perform? [spinning, growing, moving, etc.]
2. Explain premise of today's activity. "In our last coding class, you told sprites what to do. Today, you're going to be my sprites. I'm going to tell you what to do in a special game of Simon Says."

3. Have students stand up and spread out. Play a round of the basic version of Simon Says as a warm up, especially if your class is unfamiliar with the game. For example, if you say "Simon says spin around" students should spin around, but if you just say "spin around" they shouldn't spin because you didn't say "Simon says".
4. Next, explain the new twist. In coding, if we tell a sprite to spin, it will keep spinning until we tell it to stop. For example, if you say "Simon says spin" followed by "Simon says clap your hands" the students should start spinning then keep spinning as they clap their hands. They will only stop spinning when you say "Simon says stop spinning". Demonstrate using this example.
5. Begin a new game, starting very slowly at first. Here is an example sequence:
    - Simon says walk on the spot
    - Simon says stop walking
    - Simon says put your hands on your head
    - Simon says spin around – Students should spin around with their hands still on their head
    - Simon says stop spinning – Hands should still be on their head
    - Simon says take your hands off your head – Students should be back to neutral
6. Continue playing the game, gradually adding more simultaneous actions as your class gets the hang of it. See if they can get up to doing three or even four actions at the same time.
7. Debrief the activity using the following discussion questions as a guide.

DISCUSSION QUESTIONS

Did you find it hard to do two things at once? What made it harder?

In this activity, I had to tell you when to stop doing something. How come computers need us to tell them when to stop?

Answer: Computers can't "think" like we can. We need to tell them exactly what to do.

Can you think of a time in your life when you might have to do two things at once?

Possible answers: Hand signal while riding bike, sing "Happy Birthday" while washing hands, listening while taking notes

FINAL PROJECT

Sprites in action

60 MINUTES

Students will apply their coding knowledge to complete a Code.org lesson on their own or with a partner. At the end of the lesson, students will have "free play" time to make their own short animation.

QUICK LINKS

Student Activity Link
studio.code.org/s/coursef-2022/lessons/4/levels/1

LEARNING OBJECTIVES

- Write and execute code, including code with concurrent events
- Read and alter existing code

CURRICULUM CONNECTIONS

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

SET-UP

Open the activity link and try the lesson for yourself. It should take you 15-20 minutes to complete. Afterwards, you will have to click "Version History" and then "Start Over" on each level, otherwise your previous answers will be visible when you show the students.

INSTRUCTIONS

SUMMARY

You and your students will be completing a Code.org lesson titled "Sprites in Action" together. In each level, students will use coding blocks to make sprites perform different behaviours. At the end, there is a "free play" level where students can make whatever they'd like.

1. Ask students to recall what they learned about in the last online class.
   ASK What are the characters called that we move around? [Sprites] What can we control about our sprites? [Size, position, behaviours]
2. Start by trying the prediction question in **Level 1.**
   Read the question and code out loud for the students.

3. Next, watch the lesson's introductory video as a class **(Level 2)**. The video explains what you will be doing in the lesson.
4. Begin working through each level while reading all the instructions together, or you can let students try it for themselves.
5. Levels 6 and 7 are about stopping behaviours. If you've already done Offline Activity 3, you can use it to help explain this concept.
6. Level 8 has six different challenges to try. If you have time, let students pick a challenge to try on their own. Otherwise, skip ahead to Level 9.
7. Level 9 is a "free play" level. Give students time to play around with the different coding blocks to make whatever they'd like. Try to save at least 15 minutes at the end for this.
8. Walk around the class to help students and ask them to show you what they made.
9. After the class, review the **ASSESSMENT FRAMEWORK** on **Page 2.23**. It is meant as a tool to help you evaluate each student's work during this final project and their progress through the ***Coding in the Classroom*** program as a whole.

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 block types (set size, behaviours, etc.) in final project Does not answer questions during online classes, even with assistance	Uses some block types (set size, behaviours, etc.) in final project Answers questions during online classes with some assistance	Uses most block types (set size, behaviours, etc.) in final project Answers some questions during online classes	Uses many block types (set size, behaviours, etc.) in final project Answers many questions during online classes
Understanding of content	Rarely uses correct sequencing to solve coding puzzles Rarely uses concurrent events (e.g., performs two behaviours at once) when appropriate	Sometimes uses correct sequencing to solve coding puzzles Sometimes uses concurrent events (e.g., performs two behaviours at once) when appropriate	Often uses correct sequencing to solve coding puzzles Often uses concurrent events (e.g., performs two behaviours at once) when appropriate	Always uses correct sequencing to solve coding puzzles Always uses concurrent events (e.g., performs two behaviours at once) when appropriate

THINKING

	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
Use of planning skills	Rarely predicts which coding blocks are needed to solve a puzzle or make an animation	Sometimes predicts which coding blocks are needed to solve a puzzle or make an animation	Often predicts which coding blocks are needed to solve a puzzle or make an animation	Usually predicts which coding blocks are needed to solve a puzzle or make an animation
Use of processing skills	Converts ideas into code with limited effectiveness	Converts ideas into code with some effectiveness	Converts ideas into code with considerable effectiveness	Converts ideas into code with high degree of effectiveness
Use of critical/ creative thinking processes	Troubleshoots and "debugs" code with much assistance Does not experiment with new ideas in final project	Troubleshoots and "debugs" code with assistance Experiments with a new idea in final project	Troubleshoots and "debugs" code with some assistance Experiments with a few new ideas in final project	Troubleshoots and "debugs" code with little assistance Experiments with many new ideas in final project

COMMUNICATION

	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
Expression and organization of ideas and information in oral, visual, and/or written forms	Rarely organizes code clearly or uses correct terminology	Organizes codes somewhat clearly and sometimes uses correct terminology	Organizes code clearly and mostly uses correct terminology	Always organizes code clearly and uses correct terminology
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

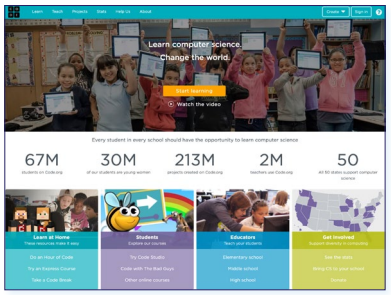
	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
Application of knowledge and skills in familiar contexts	Completes coding levels with much assistance	Completes coding levels with assistance	Completes coding levels with some assistance	Completes coding levels with little or no assistance
Application of knowledge and skills to new contexts	Applies coding knowledge to complete final project with much assistance	Applies coding knowledge to complete final project with assistance	Applies coding knowledge to complete final project with some assistance	Applies coding knowledge to complete final project 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

CODE.ORG

Code.org is the same platform we used for the online lessons. For Grade 2 students, we recommend any of the lessons from Courses A, B or C.

code.org/educate/curriculum/



Code.org

SCRATCHJR

Scratch Jr. is a free app version of the popular Scratch coding platform. It is meant for younger learners and pre-readers, and it's available on tablets and mobile devices. There are many built-in lessons that teach students to solve problems and create their own animations.

scratchjr.org

Scratch Jr.

KODABLE

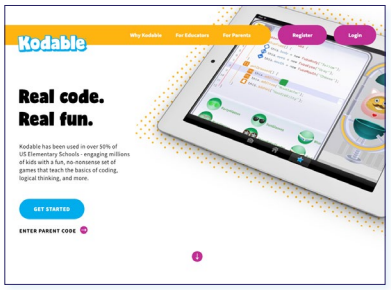
Kodable offers a range of simple, unplugged coding activities to try with your class.

kodable.com/learn/unplugged-coding-activities

CANADA LEARNING CODE

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

canadalearningcode.ca



Kodable

TVO CODING IN THE CLASSROOM

Watch webinars and read suggestions for teaching Ontario's coding curriculum.

outreach.tvolearn.com/codingintheclassroom

*Coding screenshots are sourced from **code.org**