SCIENCE NORTH SCIENCE NORD
SUDBURY, ONTARIO, CANADA

Let's Get Moving!	Grade 1 and 2

Lesson Plan

Coding Tool	ScratchJr
Time Required	Two periods

Math Curriculum Connections

Algebra
C3. Coding

Overall Expectations
C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills

Specific Expectations
C3.1 solve problems and create computational representations of mathematical situations by writing and executing code, including code that involves sequential events

Spatial Sense
E1. Geometric and Spatial Reasoning

Overall Expectations
E1. Describe and represent shape, location and movement by applying geometric properties and spatial relationships in order to navigate the world around them

Specific Expectations
E1.1 sort and identify two-dimensional shapes by comparing number of sides, side lengths, angles and number of lines of symmetry
E1.5 describe the relative positions of several objects and the movements needed to get from one object to another

Description

In this lesson, students will have the chance to explore movement in 2D spaces both on paper and on screen. They will learn how to communicate position changes using code blocks. This knowledge will be applied in ScratchJr to solve problems and have fun!

Success Criteria

- The 1st and 2nd grade students will be able to create shapes and navigate on a grid using both unplugged and computer-based activities

Materials and Media

- Paper
- Pencils
- Shape and Maze Handout
- Shape and Maze Answer Key
- Coding Handout
- Devices with ScratchJr

Computational Thinking Skills

This lesson introduces the concept of an algorithm. Typically in programming, an algorithm is a set of steps designed to accomplish a task. By communicating a task via a series of steps, students will apply algorithmic thinking. Students will use block code as a tool to build algorithms that describe movement.

The coding portion of this lesson uses ScratchJr. This is available as a free app geared to students aged 5-7. It uses an introductory block-based language that allows students to code the steps for their algorithm. The 'Coding Handout' for this lesson includes step by step examples. ScratchJr is also

available as a desktop version that teacher can use to project the program onto a smartboard. This can be found using this link:
https://jfo8000.github.io/ScratchJr-Desktop/

Introduction

Giving and following directions and being able to interpret or visualize directions using tools like maps and grids are useful skills that we use every day. These skills are what allow us to travel to new places (e.g., following a route drawn on a map or following directions described by a GPS system), to communicate locations to others (e.g., giving a friend directions to visit your home or a favourite spot, using landmarks), and to find or replace objects in their proper location (e.g., "the cereal goes on the top shelf, next to the crackers). Understanding directions as they relate to location and movement improves our spatial awareness and allows us to better navigate the world around us.

When we give directions for moving from one location to another, we break those directions down into a set of instructions, or steps. In coding, these instructions are called an algorithm. These instructions should consider direction (e.g., forward, backward, left, right), distance (e.g., steps or centimetres), and any obstacles that might be on the path of movement. A person might be able to make a decision if it comes upon an obstacle, but a computer program can only do exactly what it has been instructed.

In this lesson, students will direct movement in several contexts, from following paths, to navigating mazes, to drawing geometric shapes on a grid. Students will learn how to direct movement, verbally and using coding algorithms, using specific step-by-step instructions.

Action

Walk the robot

When you're describing directions for movement, it's important to be as specific as possible. Humans can make decisions and interpret your directions, but as we'll see later when we code instructions, a robot or a computer will only do exactly what you tell it to do.

For this offline activity, students will practice breaking movements down into steps and communicating those steps to a partner.

One student can pretend to be a robot. The robot cannot make any decisions on its own about its movements. It can only follow instructions and must follow them exactly as given.

The other student will be the programmer, who creates the instructions (also known as an algorithm), that the robot student will follow. To best model coding, the programmer should write the code out in advance before their partner executes their program. Students can also

give these instructions using directional arrows and numbers, or they can give the instructions verbally.

Some activities that the programmer can code their robot to do include:

- For an in-class activity, the programmer can create a set of directions that will have their robot travel to an object placed in a specific location on the floor while avoiding obstacles, such as other students' desks and walls
- Alternatively, the programmer can create a set of directions that will have their robot trace a path that creates a geometric shape, like a square, a rectangle, or a triangle. This can be done by walking or by drawing it on grid paper.

Once the robot is executing the program, the programmer should refrain from making corrections or calling out additional verbal instructions, just as the robot should refrain from making movement choices that are outside of the programmer's instructions. An important part of this exercise is recognizing the importance of specificity and accuracy when communicating directions. So, if the algorithm is missing a step, or if the directions are inaccurate (e.g., the directions lead the robot into an obstacle like a wall, or they do not lead the robot to the established goal), this will require the coder to de-bug their program. This means they will find their mistake, fix it and try the program again by having their partner repeat the exercise with the revised algorithm.

For the first attempt, a simpler approach would be to have the programmer read one step of their instructions at a time and the robot will execute that step before moving on to the next instruction. As an extension, the programmer can create a new path, but this time they must read the complete code (all of the steps in order) aloud before the robot begins executing the instructions. If time allows, both partners should have the opportunity to try being the programmer and the robot.

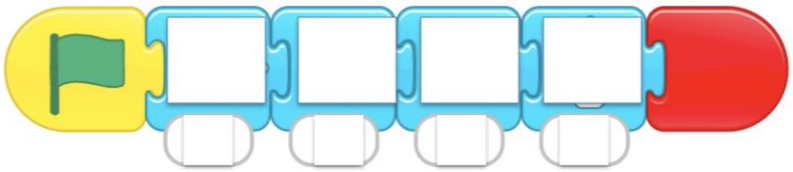
Unplugged Activity

The un-plugged activity will apply the concepts practiced during the Walk the Robot activity to a written format.

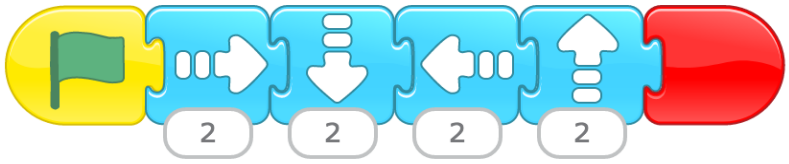
In the handout provided, students will use arrow blocks to create algorithms that communicate movement on a page. They will be tasked with creating an algorithm for two different shapes as well as navigating two mazes. The final task will allow students to follow given code to complete a maze.

Students will be using a grid where each movement code block indicates a step to the next square in the direction of the arrow. The students will also need to indicate how many steps(squares) in the given direction. Below is an example of the incomplete code. The start

block indicates that the robot will move when the green flag is clicked. The robot needs to know when to start the movement. The next block will need a direction arrow as well as a number to say how many squares to move. The last red block indicates that the code is complete.



Below, we see a completed algorithm. For ease of completion, students can draw a simple directional arrow without the dashed lines.



There are five tasks that go with the *Shape and Maze Handout*. An answer key is also provided for the teachers.

Task 1: Geometric shapes

For the first task, students will instruct the robot to walk along the largest square possible within the grid. The largest square possible will follow the perimeter of the grid.

Task 2: Rectangle

For the second task, students will have them move the robot through a rectangle that is wider than it is tall. There can be many correct answers here. The number of squares for the width (left, right) must be greater than the number of squares for the height (up, down).

Task 3-4: Mazes

The next two tasks within the handout have the students communicating steps using code blocks to navigate through the grids provided. Each grid will have a character to start as well as a goal to navigate to. Students may want to start by drawing the path on their papers and then determining which code blocks will communicate that path. There will be many possible paths. Students won't always need to use all the spaces given.

Encourage students to use the shortest path possible. Often, map algorithms will determine the shortest path to a given destination. There are also cases where we can change the search parameters to get alternate routes. For example, if we want to avoid traffic congestion or toll roads. If we change the rules of the algorithm, there is a chance we obtain a different result.

Task 5: Treasure Hunt

In the fifth and final part of the handout, the students are given a grid and a completed algorithm. They will need to move through the given code in order to determine where to place the goal, an X to mark the spot.

Coding Math Activity

In the culminating math activity, students will apply what they have learned to drag and drop code blocks within ScratchJr in order to write algorithms that describe movement.

Note that the C*oding Handout* illustrates step-by-step instructions for coding geometric shapes within the app.

After introducing ScratchJr and how the motion blocks work, students will have three tasks that build off the un-plugged activities. As there are three tasks, teachers can choose to do the different tasks either together using a smartboard, independently on iPads or a combination of the two depending on the comfort level of students. From our perspective, this activity is likely best done as a group, by projecting the desktop version of ScratchJr onto the smartboard and guiding the students, but feel free to adjust the activity based on the skill-level of your students.

Task 1: Large Rectangle

The first task on Scratch-Jr will require students to move their sprite through the largest rectangle possible within the grid space of ScratchJr.

Task 2: Large Square

Once students have made the largest rectangle they can, they will apply the knowledge from the first challenge to complete the code for the largest square possible on the project stage as well.

Task 3: Visible Rectangle

Their final task will be to code the largest rectangle they can while keeping their sprite in full view throughout the movement. As students work through their tasks, have them test their code often and then show their work to gauge their learning.

Closure and Assessment

- By the end of the lesson, students should be able to give and follow directions for moving from one location to another, including navigating movement needed to get from the position of one object to another. They will also be able to describe and modify the movement of a sprite using block code.
- The Shape and Maze Handout can be collected by teachers and used for assessment of learning. Look for accuracy of their code to assess that students can apply the key concepts and skills related to the activity.

Adaptations

- The Walk the Robot activity can be done sitting down by having students draw paths instead of walking them.
- Both the unplugged and coding activity are broken into smaller tasks. Teachers can choose to do tasks together or in groups before students attempt them to build familiarity.

Extensions

- Students can add obstacles into the paths in their mazes. They can adjust their code to go around objects to the goal.
- Students can try to create rectangles with different conditions in ScratchJr, or more challenging shapes all together.

Additional Resources

- ScratchJr desktop download: https://jfo8000.github.io/ScratchJr-Desktop/