Activity P03:

Sending A Rocket To Mars

TAHI RUA TORU TECH

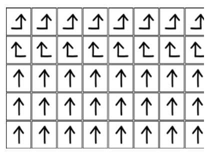
KIA HIHIKO AOTEAROA!

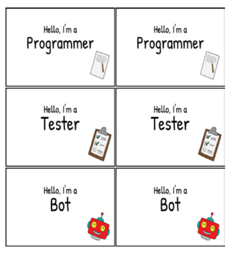
Discovery

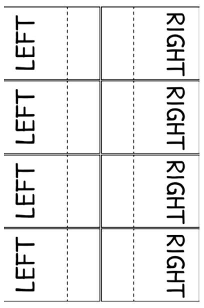
Computer programs are all around us in things like fitness devices, smartphones and alarm systems. These programs are referred to as “applications”, “apps” or “software” and often they are nearly invisible. In devices, software “boots up” when you switch it on and keeps going until you switch it off. Despite this variety, they all work on similar principles that are accessible to students of all ages, and this unit explores some of the fundamental principles in programming.

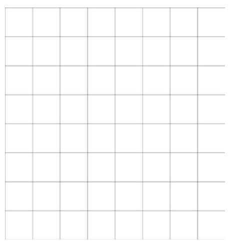
Mathematics: Numeracy, Geometry
Literacy: Speaking

Downloadable Resources (One Per Class):









Note: Be sure to print double-sided!

Classroom Resources: Blocks, clipboards, handheld whiteboards, whiteboard pens, paper and pens.

What is It?

Writing a computer program involves planning what you're going to do: “coding” the instructions, testing them, tracking down any bugs, and changing the program so that it works correctly. Most substantial programs are written by a team of people and often the roles of design, coding and testing are separated out. This team is usually comprised of the Programmers (the people who write the program), and the Testers (the people who test the program).

Why?

These Kidbots activities separate the programming from the testing to avoid the programmer adjusting their program on the fly and also support students to understand that programming is about working together, thinking through what you want to have happen and collaborating to solve problems.

Link to Digital Technologies Curriculum

Creating a sequence of instructions for this lesson exercises **algorithmic problem solving**, as it requires students to create an algorithm to accomplish a task. Computational algorithms are introduced in the first **Computational Thinking** Progress Outcome and are based on input, output, storage, sequence, selection and iteration.

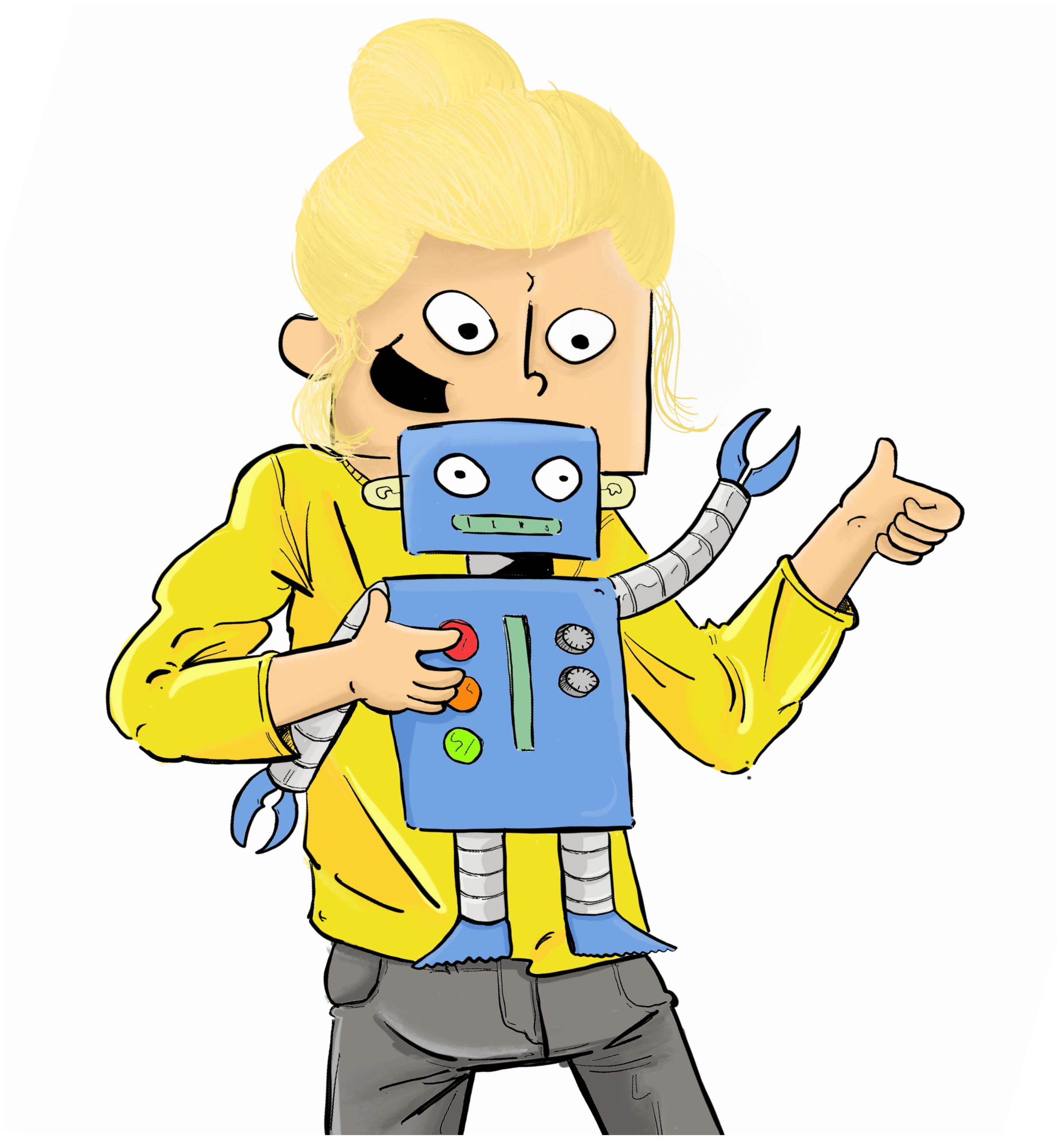
This activity utilises the CS Unplugged material and is released under Creative Commons CC BY-SA 4.0.

Activity Background

Ideally this lesson should take place around a large grid such as:

- An outside painted chess board.
- Grids in your classroom carpet.
- Making masking tape grids on the floor in your classroom.
- Draw a chalk grid either in your classroom or outside.

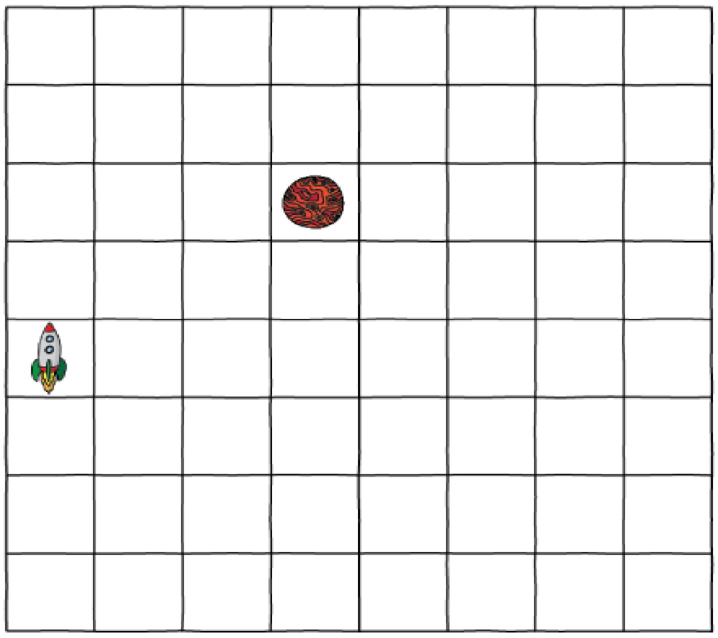
Ask for two volunteers and give yourself and them the roles of:
Role 1: The Developer (who writes the program) - The teacher will model this initially
Role 2: The Tester (who instructs the Bot and looks for bugs)
Role 3: The Bot (who runs the program)



This activity utilises the CS Unplugged material and is released under Creative Commons CC BY-SA 4.0.

The Activity

Teacher: "I'm going to be the programmer, but I'm going to need your help." "We are **programming** "the Bot", not just remote controlling it, because ALL the instructions are written before the Bot can follow those instructions.



"Debugging is fun because you get a chance to change your program after it's finished when you notice it's not working how you thought it should."

"It's my job to write down clear instructions for the Bot, who is going to be (say the person's name) and "The Tester" (who is...) is going to give the instructions to the Bot and be on the lookout for bugs."

"First of all I need to decide, what programming language are we are going to use for this? I've chosen arrows to represent move forward, turn left and turn right."

If students aren't sure about the left and right direction, you can print the "left and right hand cards", and stick them to their shoes or have them hold them in their hands.

Have the Bot act out the individual instructions: forward means step one square forward, and left and right mean a 90 degree turn on the spot in the square (not moving to another square).

Teacher: "We're going to write our own program that gets the rocket to fly to Mars. The goal is to get the rocket to the square that Mars is in. Let's write the first two steps on the board together." (Draw two forward arrows.)

"So let's try that, and see what happens.

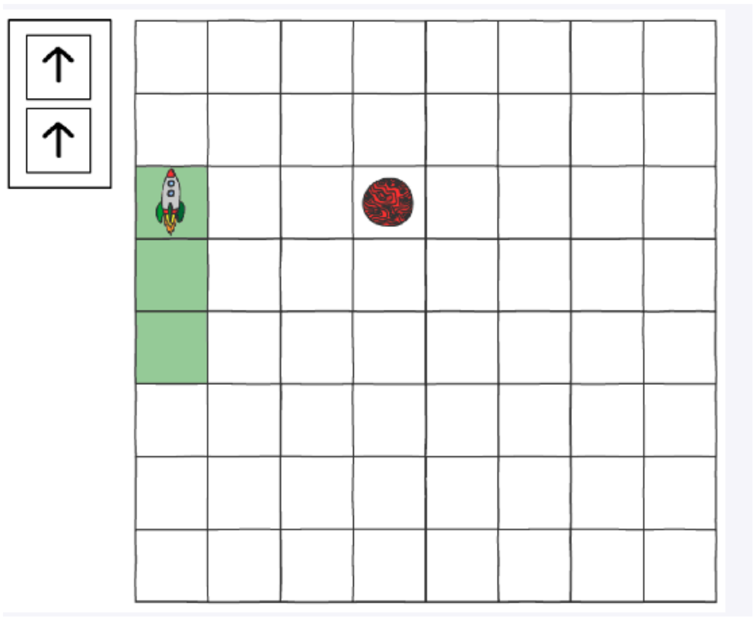
"Tester - could you please take these instructions and pass them onto the Bot. Be ready to underline what doesn't work when you see the Bot doing something that doesn't look right, and hand the whiteboard back to me to figure out how to fix the bug."

Teacher: "Bot - please pick up the rocket ready to receive the instructions for the tester." (The bot can carry a toy or token representing the rocket; or they can imagine that they are guiding it).

Tester then reads off the board: "move forward, move forward."

This activity utilises the CS Unplugged material and is released under Creative Commons CC BY-SA 4.0.

Rocket to Mars program:

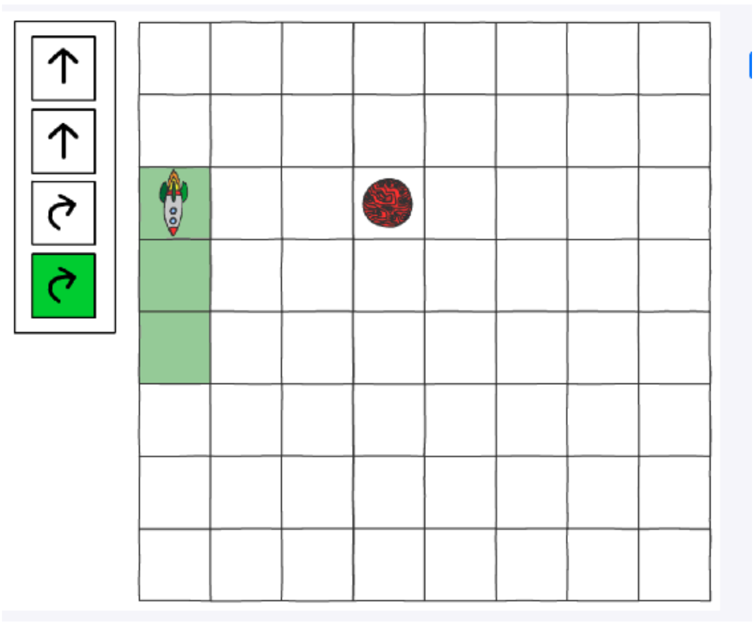


"Tester, did the program run as you expected it to?" Depending on the tester's response, if it did then carry on programming, otherwise fix what didn't work and run that again. In this example the rocket should be in the square three to the left of Mars.

Now let's add to it. What would we program next?

Point to where the next piece of code needs to be added and add turn right, turn right. (This is deliberately incorrect.)

Rocket to Mars program:



"I think it's ready to test now. Tester, please test my program (the programmer hands the program on the whiteboard to the tester and the bot should return to the starting square ready to rerun the program".

Teacher: "Remember Tester, it's your job to find any "bugs" in my program. A bug is when my program isn't doing what was expected. Your job is to draw a line under the piece of code where they notice the instructions seem to be going wrong. You can stop the Bot at the point that you think there is a bug.

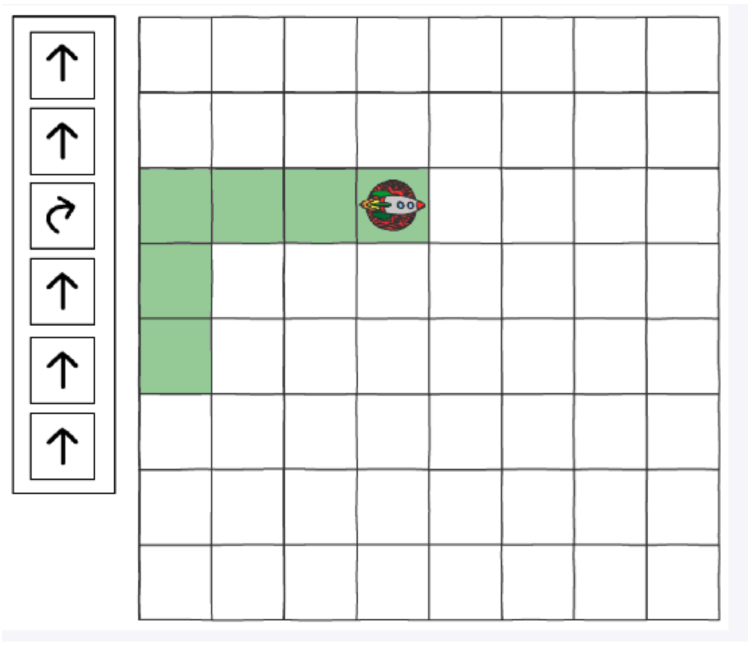
Tester then reads the instructions in the program off the board and the Bot executes them as they are read.

1. Move forward
2. Move forward
3. Turn right
4. Turn right

Teacher: "Excellent, you found a bug! I love finding bugs, so I can start solving them. Now class, let's work through this together to find my bug. Tester, you've done a great job finding it, but it's the programmer's job to find and fix the bug."

This activity utilises the CS Unplugged material and is released under Creative Commons CC BY-SA 4.0.

Once the bug has been identified then ask the Tester to test again. Ask the Bot to pick up the rocket and go back to the start position, then the Tester reads them the instructions.



Did we successfully program the rocket to land on Mars? How do we know?

Are there other ways we could have programmed the rocket to get to Mars? (There will be lots of ways; for example, Right, Forward, Forward, Forward, Left, Forward, Forward will work.) Discuss the programming options and test each one. What if we want the rocket to get to Mars, and then come back safely?

Have the students choose their own two toys (one to be a space travelling object, the other to be the destination) and have them practise this task, as follows.

5. Place the traveller on a square on the edge of the grid, facing inwards.
6. Place the destination toy inside the grid.
7. The programmer writes down the program on a whiteboard.
8. The tester then takes the whiteboard and a different coloured whiteboard pen. The tester tells the Bot each instruction in the program. The tester puts a tick next to the code that is correct and underlines when the code is different to what the Bot should be doing. If this happens the Tester says “Stop” and the Bot stops and goes back to the start. The Tester gives the whiteboard to the Developer, who then debugs the code, and gives the Tester a revised version.
9. Repeat step 4 until the program is free of bugs and works as intended.
10. Change roles and move the Bot (space travelling object) starting point and the toy that represents the destination until everyone has had a turn

Extending The Lesson

It’s quite common to think that programming is some kind of special talent that people either have or don’t have, but this couldn’t be further from the truth!

Like all skills, programming is something you learn through practise, making mistakes, and learning from them. The most important skill that programmers need is to be able to communicate with others, especially when they are finding and describing bugs.

Bugs happen all the time in programming, so being able to identify where the bug occurs and problem solving how to fix it is incredibly important. It doesn’t matter how experienced you are at programming, there will always be bugs that need to be found and fixed. That’s why the word “debugging” is so important to programmers.

This activity utilises the CS Unplugged material and is released under Creative Commons CC BY-SA 4.0.