Moon Landing Scratch Programming Detailed Guide (ages 7+)

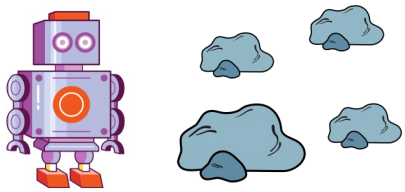
get.with.the PROGRAM

About this activity

get.with.the PROGRAM

Can you make a robot moon rock sorting game?

Our Scratch robot is sorting moon rocks to take back to Earth and needs to work out which are the best ones for the journey. Can you use Scratch to design, test and debug an algorithm to help her do this?



Learning intention:

- Design, test and debug an algorithm in Scratch to control a robot and sort moon rocks

I can…

- Design an algorithm to achieve a goal
- Test and debug an algorithm
- Include repetition and explain how it makes the algorithm more efficient
- Include selection in an algorithm
- Use examples of input and variables
- Understand the concept of initialisation

Key words

algorithm
design
test*
debug
variable
repetition
selection
input/output
initialisation*

* term not found in KS2 computing curriculum

v3.1 2026

1

Key Terms

get.with.the PROGRAM

The coding adventure focused on computing terms like algorithm, debugging, decomposition, design and test (see our glossary). This Scratch activity also talks about the curriculum terms below.

Repetition	**Repetition** is looping or repeating sections of an algorithm so the steps can be reused.
Selection	**Selection** is making a choice to take a path in the algorithm, based on other information.
Input/Output	**Input** when you have some information that comes into or out of the algorithm.
Variable	A **Variable** is the place in an algorithm where you put information to use again.
Initialisation	**Initialisation** is a section of an algorithm that is at the start, usually resetting things.

2

Getting started

Open Scratch 3.0

You can do this online here: https://scratch.mit.edu/ or use a local version.
This project only uses resources available in Scratch, so there's no need for students to sign in.

- Open Scratch
- Click Start creating
- Close the tutorial

Setting up your sprites and background

Let's remove the initial sprite and add in the background you will need for this activity:

- First delete Sprite1 (the cat)
- Now click Choose a Backdrop
- Our robot is sorting moon rocks so let's choose Moon
- You can now see the Moon backdrop in the top right area of the screen (called the stage)

Now you are ready to begin!

Note: Saving projects - if you would like to save your work as you go and are not logged in (recommended), you can save your project onto the computer: In the menu click File > Save to your computer. You can find it in Downloads saved as 'Scratch Project.sb3'

Challenge 1 – sorting the sprites

For this first Challenge, can you program the Moon rocks to sort themselves when clicked on?

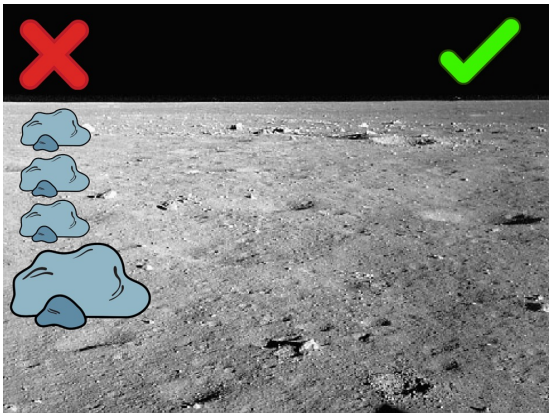
If you have used Scratch before you might have a few ideas how to do this!

For our solution, we are going to:

- Create 4 sorting sprites. Here we're using four Moon 'rocks'.
- We'll resize and rename the sprites, then reposition them on the stage (see image on left).
- We'll add in in the 'Tick' and 'Cross', rename and reposition these, and change the 'Cross' from black to red.
- Finally, we'll program the sorting sprites to move to 'Tick' or 'Cross' when clicked on, before being reset back to their starting point.

We'll do this using blocks from the following menus:

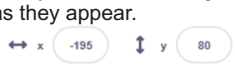
Motion Events

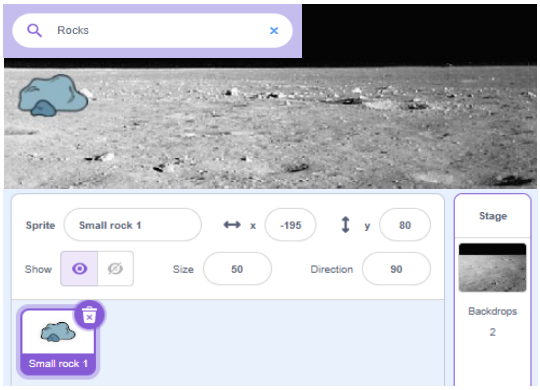


Challenge 1: step 1

Let's resize, re-name and position our sorting sprites.

- Go to Choose Sprite, search for Rocks and select the sprite.
- Right-click (mouse) or tap and hold (tablet) on your sprite to duplicate it until you have 4 rocks.
- Look below the stage to see each sprite's information.
- Reduce the first 3 sprites to 50 in the size box. Click inside the sprite box to rename them 'Small rock 1', 'Small rock 2', and 'Small rock 3'. Leave the final rock at size 100 and rename it 'Large rock'.
- Drag the sprites across your stage to the left, leaving space at the bottom, or use the co-ordinates on the right to reposition. Note down each sprite's co-ordinates (the x and y numbers) as you'll need these in step 3. Remember, some of the numbers may have a minus sign (see below) so write them down as they appear.





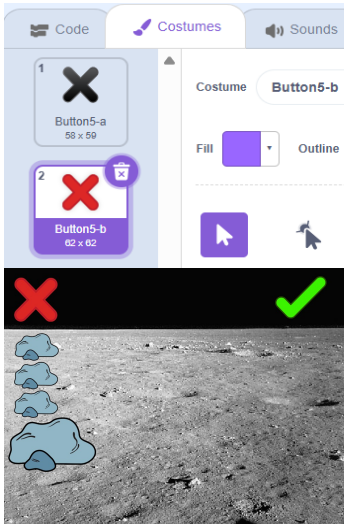
Names and co-ordinates

Small rock 1:	x =	-195	y =	80
Small rock 2:	x =	-195	y =	40
Small rock 3:	x =	-195	y =	0
Large rock:	x =	-195	y =	-45

Challenge 1: step 2

Let's create 'Tick' and 'Cross' sprites for the rocks to be sorted to.

- Go to Choose Sprite and search for 'button'. Select the 'Tick' and position it to the top right of the stage.
- Repeat this to find the 'Cross' sprite by searching for button' again, and position this to the top left of the stage.
- Select and rename them 'Tick' and 'Cross'.
- Click the 'Cross' sprite, then click the Costumes tab (top-left of the Scratch screen). Select the red costume to change your sprite from black to red.

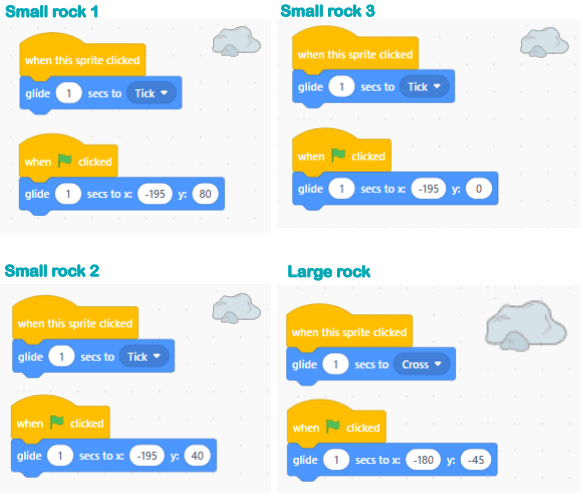


Challenge 1: step 3

get.with.the PROGRAM

Let's make the sorting sprites move to the 'Tick' or 'Cross' when we click on them.

- Select the first sorting sprite ('Small rock 1').
- Add a **when this sprite clicked** (Events) input block. This starts the algorithm whenever the sprite on the stage is clicked (initialisation).
- Add a **glide 1 secs to [random position]** block (Motion). Change the [position] to 'Tick'. Click the sprite to test.
- Add a **when**  **clicked** (Event) block (input). Drag a **glide 1 secs to x: [ ] y: [ ]** (Motion) block underneath. Update its starting co-ordinates (the x and y numbers you wrote down in step 1) in the correct spaces (remembering some may have had a minus sign). This will reset the sprite.
- Now test. Did it return to its starting position? If not, debug.
- Repeat for all the sorting sprites, updating their co-ordinates and sending the Large rock to the 'Cross'. Test and debug.



Challenge 2 – the sorting robot!

get.with.the PROGRAM

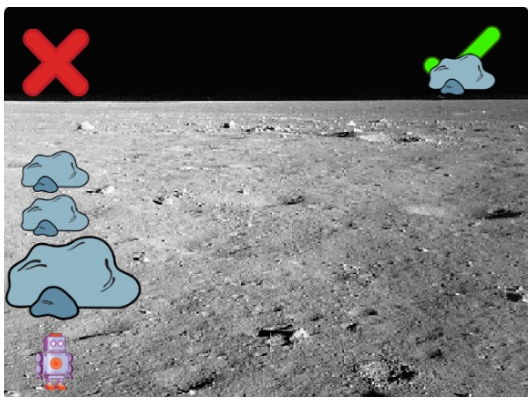
For this Challenge, can you code a robot that sorts the sorting sprites and sends them to the 'Tick' or the 'Cross'?

You may already have some ideas of how to do this, if you've used Scratch before!

For our solution we're going to:

- Create an 'AL the Robot' sprite.
- Program each sorting sprite to move to 'AL the Robot' when clicked.
- Program the sorting sprites to glide to the 'Tick' or the 'Cross' after touching 'AL the Robot'
- Return to their starting position when the green flag is clicked to reset.

We will do this using blocks from the following menus:



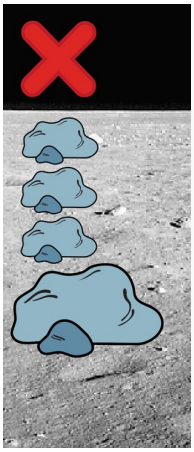
Challenge 2: step 1

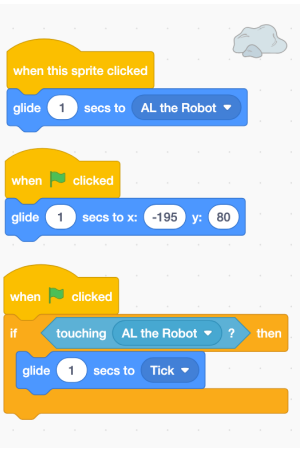
get.with.the PROGRAM

Let's resize the robot sprite, rename her and position her, and then test it.

- Create a new sprite by searching for Retro Robot.
- Reduce her size to 35%, rename her 'AL the Robot', and position her below your 'rock' sorting sprites.
- Change each sorting sprite's **when sprite clicked** (Event block) instruction, so they glide to 'AL the Robot' when clicked.
- For the first sorting sprite, add a new **when**  **clicked** (Event block) input.
- Drag a **forever** (Control) block beneath it and place an **if-then** (Control) block (selection) inside it.
- Add a **touching [mouse pointer]** (Sensing) block. Change the dropdown to 'AL the Robot'.
- Drag a **glide 1 secs to [random position]** block (Motion) inside the **if-then** block. Change the [position] to 'Tick'.
- To test, click the  then click the sorting sprite (on the stage). **Does it work as expected?**

Motion Sensing Events Control





9

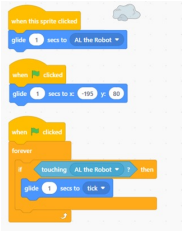
Challenge 2: step 2

get.with.the PROGRAM

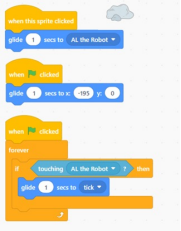
Let's debug our code until we get what we hoped for and then copy it to the other sorting sprites.

- If the sorting sprite didn't move to the 'Tick' when it touched the robot, try clicking the  again to reset it. Why didn't it work as we'd hoped? Let's debug.
- The program needs to repeatedly check if the sprite is touching 'AL the Robot'.
- So, to do this, place your **if-then** (Control) block (selection) inside a **forever** (Control) loop (repetition).
- To stop the **forever** loop, click the **red dot** (top right of Scratch screen) after you've tested.
- If it then works as expected, copy this code to the other sorting sprites.
- Test each sprite individually, then test all together (remembering to start with the  and stop with the **red dot** each time!).

Small rock 1

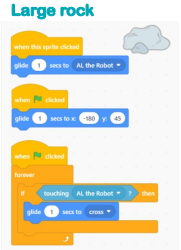


Small rock 3



Small rock 2

Large rock



10

Extension Challenge 1 – catching the sprites

get.with.the PROGRAM

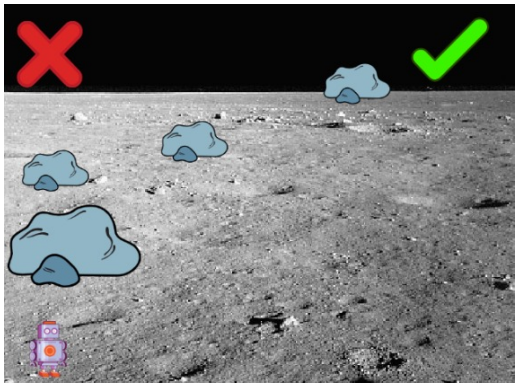
For this first Extension Challenge, can you make the game harder by making all the sorting sprites fly across the stage and catch them as they move?

You may already have some ideas of how to do this, if you have used Scratch before!

For our solution we are going to:

- Make each sorting sprite now move across the stage from left to right when the  is clicked.
- Test, and debug if needed.

We will do this using blocks from the following menus:

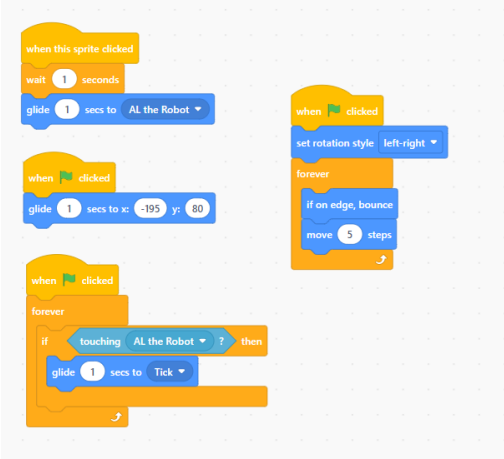


Extension Challenge 1: step 1

get.with.the PROGRAM

Let's make the sorting sprites fly across the stage to make the game harder!

- In the first sorting sprite, add another **when**  **clicked** (Event) block (input).
- Add a **forever** (Control) loop block (repetition) and place **a move [10] steps** (Motion) block inside it, changing the 10 to a 5. Test.
- To debug, add an **if on edge, bounce** (Motion) block above your **move [5] steps** block.
- Drag a **set rotation style left-right** (Motion) block above the **forever** loop. Test.
- Copy this sequence of blocks into each sorting sprite and test again.
- To make the sprites start at different times, add a **wait 1 seconds** (Control) block under **when sprite clicked.** Set a different wait time for each sprite.



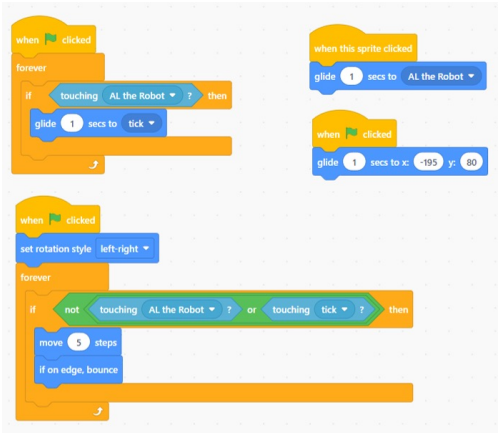
Extension Challenge 1: step 2

get.with.the PROGRAM

Let's debug our code so our sorting sprite stops after being sorted. Then we can copy it to the other sorting sprites.

- Drag an **if-then** (Operator) block inside your **forever** loop. (logic).
- Drag a **not** (Operator) block into the **if-then** (Operator) hexagonal space. Then, place an **or** (Operator) block inside the **not** block.
- Drag two **touching [mouse button]** (Sensing) blocks into the **or** block and use the dropdown to select 'AL the Robot' in one, and 'Tick' or 'Cross' in the other.
- Now, the sprite should only move if it's NOT touching 'AL the Robot', the 'Tick' or the 'Cross'.
- Now, test to check it works, copy to the other sorting sprites, and test again!

Operators Sensing



Extension Challenge 2

get.with.the PROGRAM

Can you make 'AL the Robot' say, 'I've sorted all the moon rocks ready for Earth!' and create a timer to see how quickly you can sort them?

You may already have some ideas of how to do this, if you have used Scratch before!

For our solution we are going to:

- Create variables to track each sorting sprite and show it it's been sorted.
- Change each variable to Yes when a sprite is sorted and reset them all to No when they are reset.
- Check whether the sprites have been sorted and make 'AL the Robot' say something like either: 'Working on it …' or 'I sorted all the moon rocks for Earth!'
- Add a timer that counts how quickly the rocks are sorted.

We will do this using blocks from the following menus:

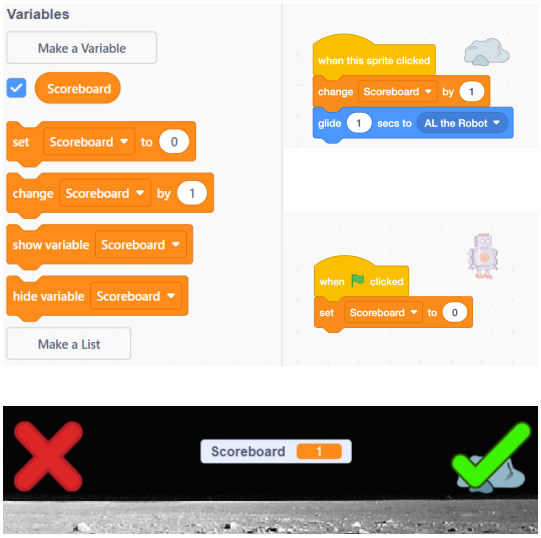
Variables Operators Motion Looks Events Control

Extension Challenge 2: step 1

Let's create a variable to keep score when the sorting sprites are clicked.

- Create a variable (Variable) called Scoreboard. Make sure it is set to *for all sprites*.
- Leave the box ticked so you can see the Scoreboard on the stage.
- Select 'Small rock 1' sprite and drag a **change [my variable] by 1** (Variables) block into your **when sprite clicked** (Event) block from earlier. Use the dropdown arrow to select 'Scoreboard' - this adds 1 to the Scoreboard each time the sprite is clicked. Copy this code to all the other sorting sprites.
- In AL the Robot's code, add a **set [my variable] to 0** (Variables) block to the **when clicked** (Event) block to reset the Scoreboard to 0.
- Test your code to check the score increases and resets correctly. Debug, if needed, then copy to your other sprites.

Variables Events



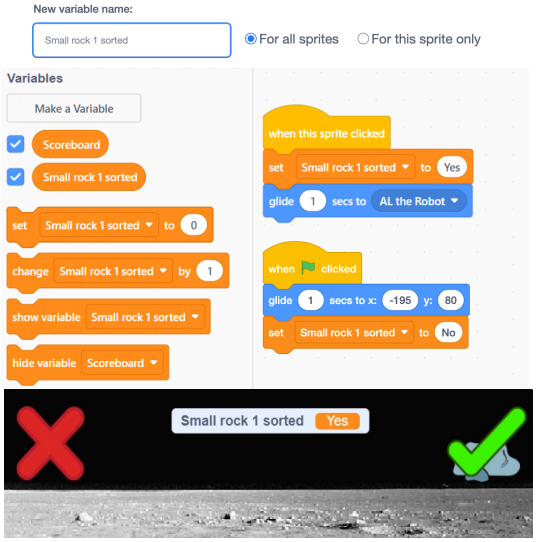
15

Extension Challenge 2: step 2

Let's create more variables to see when each sorting sprite has been sorted.

- Create a variable (Variables) called 'Small rock 1 sorted'. Make sure **for all sprites** is selected.
- Leave the box ticked to show the variable on the stage.
- Select 'Small rock 1' sprite and drag a **set [my variable] to 0** (Variables) block into the coding area.
- Place one under the **when sprite clicked** (Event) block, select 'Small rock 1' from the dropdown and replace the 0 with Yes. Test your code works.
- Drag another **set [my variable] to 0** (Variables) block into the coding area and select 'Small rock 1' again. Now place it under the **glide to [starting co-ordinates]** (Event) block, and replace the 0 with No. Test your code works.
- Repeat this for all your sorting sprites and test again. Each sorting sprite's variable should be No before they're sorted and change to Yes when the sprite is clicked and sorted.

Variables Events



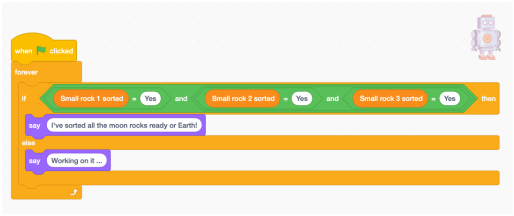
16

Extension Challenge 2: step 3

Let's create code for 'AL the Robot' to check the variables, and an output saying, 'I've sorted all the moon rocks, ready for Earth!'

- Select 'AL the Robot' and drag a **when clicked** (Event) block into her coding area.
- Drag an **if-else** (Control) block beneath it, to create two pathways in the algorithm (selection).
- Drag 2 **say [Hello!]** (Looks) blocks into the **if** and **else** pathways and overwrite them with something like, 'I've sorted all the moon rocks, ready for Earth!' and 'Working on it …'
- Drag 2 **and** blocks (Operators) into the **if** section and then add in 3 **=** blocks (Operators), one for each tick-sorting sprite, as in the image to the right, to check when all are sorted.
- Then test this out to check it works. Debug, if needed.
- Finally, give the whole game a test, hiding the variables you don't need on your stage … and have fun playing!

Looks Events Control

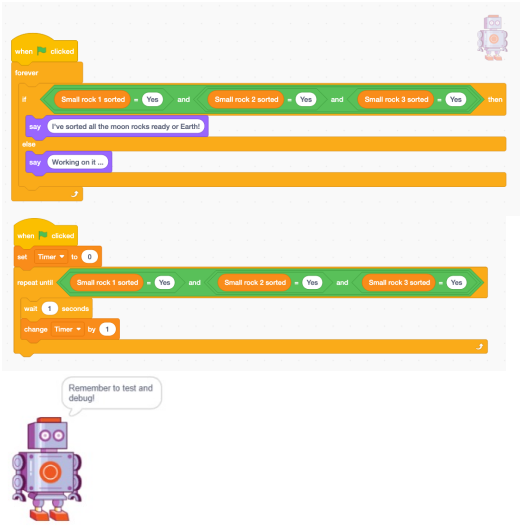


Extension Challenges – more ideas

Let's create a timer to see how quickly you can sort the rocks!

- Select 'AL the Robot' and drag in a new **when clicked** (Event) block.
- Create a Timer variable.
- Drag a **set variable to [0]** (Variable) block underneath the new **when clicked** block. Change the variable to Timer.
- Add a **repeat until** (Control) block underneath this to show when to stop the timer.
- Find the "[rock] sorted = Yes" code you created earlier. Right-click (mouse) or tap and hold (tablet) to duplicate it, then drag this code into the **repeat until** block's 'hexagonal space'.
- Drag a **wait [1] seconds** (Control) block inside the **repeat until** block.
- Drag a **change [my variable] by [1]** (Variable) block under the **wait block.** Change the variable to Timer.

Variables Operators Looks Events Control



Plenary

get.with.the PROGRAM

Now let's review what happened!

*KS2 curriculum terms

Do you know what these words mean?

algorithm design test debug variable

repetition selection input output initialisation

Tell me about the project you made today:

- What was your favourite part of this activity?
- What did you learn about repeating a section of your algorithm?
- What did you learn about using 'if' statements?
- What was the most challenging part of this activity?

Plenary definition answers

get.with.the PROGRAM

Here are our definitions of the key words:

algorithm	a step-by-step set of instructions
design	when you think things through before you start
test	when you see if everything works
debug	trying different things until you get what you hoped for
variable	the place in an algorithm where you put information to use again
repetition	the process of looping or repeating sections of an algorithm
selection	making a choice/decision depending on information available
input/output	when you have something that comes into or out of the algorithm
initialisation	a section of an algorithm that is at the start, usually resetting things