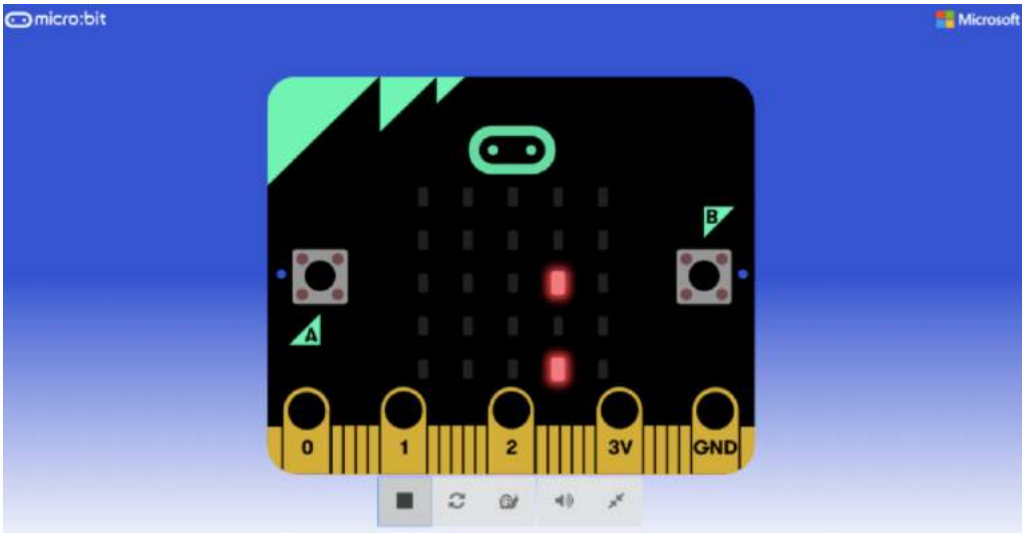
MOONHACK

MOONHACK 2018 MICRO:BIT PROJECT



INTRODUCTION

On 20 July 1969 the first humans to ever touch down on the Moon did so in the Lunar Module. Landing this machine was no easy task. The pilot, Neil Armstrong had to guide the craft to the optimal landing zone. In this activity we are going to create a game based on this first moon landing from 49 years ago. Follow the instructions below to create your own Lunar Module Simulator using the BBC micro:bit.

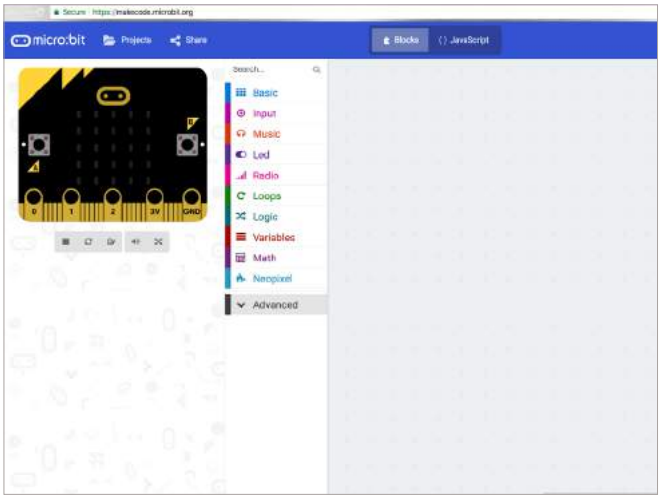
Even if you do not have access to a micro:bit you can complete this project! The free virtual editor allows anyone to participate.

STEP 1: POSITION THE SPRITES

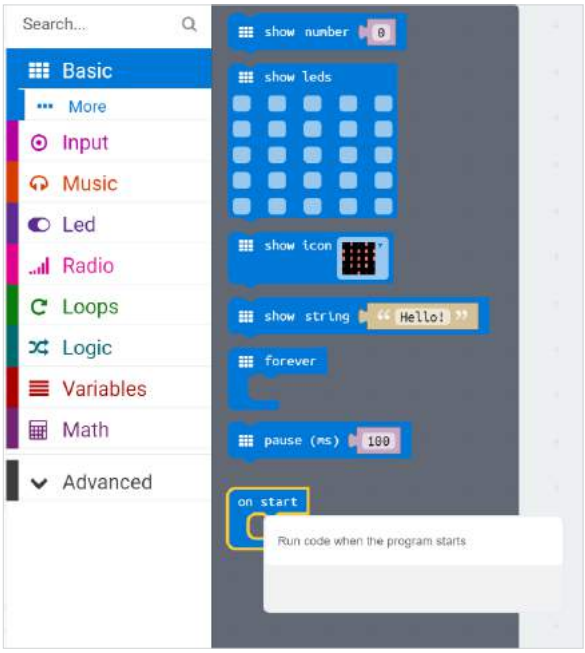
In this step we will set the initial positions of the Lunar Module and the Landing Site.

Activity Checklist

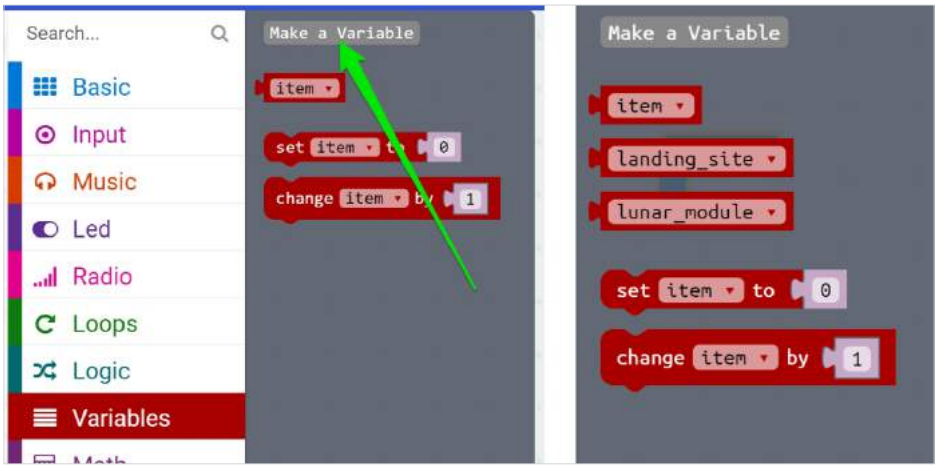
- [ ] In your browser, go to makecode.microbit.org and remove the code that is given to you by dragging it to the left sidebar until you see a rubbish bin. Do this until you have a clear project like the image:



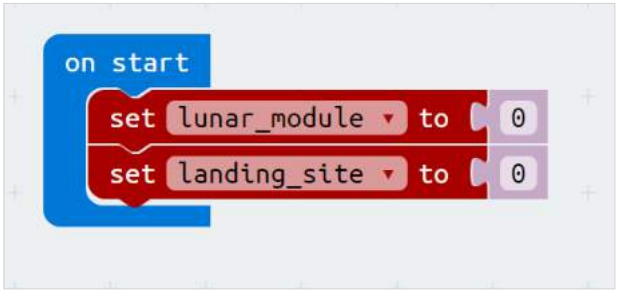
- [ ] We need to take an on start block from the basic panel and add it to our program. This will run our code when the micro:bit starts up.



- [ ] Next we will need to create two variables: lunar_module and landing_site. Go to the variables panel and select "Make a Variable" for both of these.



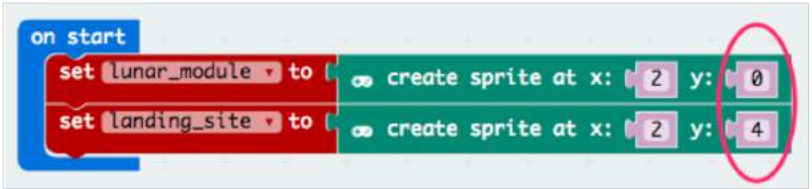
- [ ] Next, we will assign values to those variables. Drag the set item to 0 block into the on start block twice, and use the drop down arrow to change item to the two variables you just created.



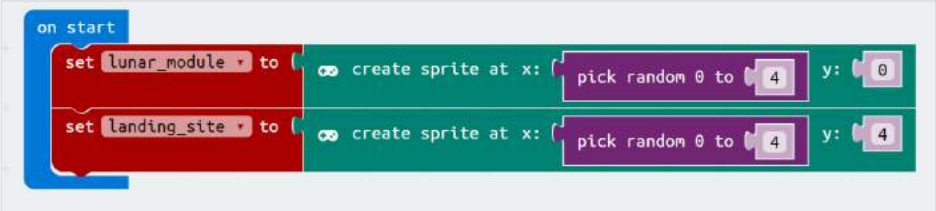
- [ ] Finally, we will assign sprites, which are our game elements, to the two variables. To do this we will need to go into the advanced menu, and open the game panel.

- [ ] Drag the create sprite block into each variable assignment.

- [ ] We will need to place the sprites in the correct positions by assigning the correct values. The lunar_module will start on the top row (row 0) and the landing_site will start on the bottom row (row 4). Set your y values to these rows values.



- [ ] Finally we will need to assign an x value (left and right position) to each sprite. Because we want to make this game difficult, we will have both the lunar_module and the landing_site appear in random x positions. To do this we will need to add a Pick Random block from the Math panel. Add this block to the x value for both the lunar_module and the landing_site.



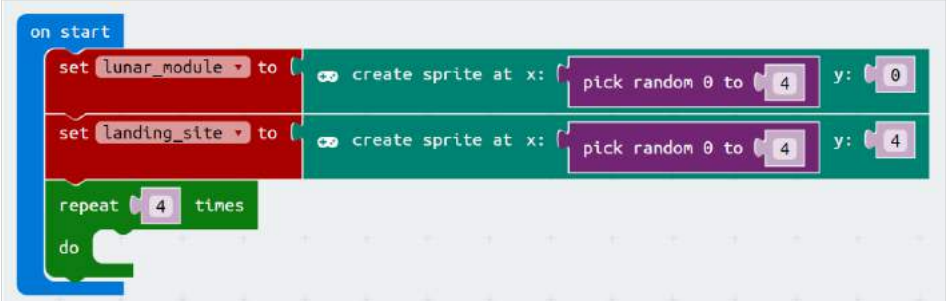
- [ ] Test your code by pressing the refresh button on the simulator several times. Do the two dots appear randomly on the top and bottom row? If so, then you have correctly positioned both the lunar_module and the landing_site!

STEP 2 - MAKE THE LUNAR MODULE FALL

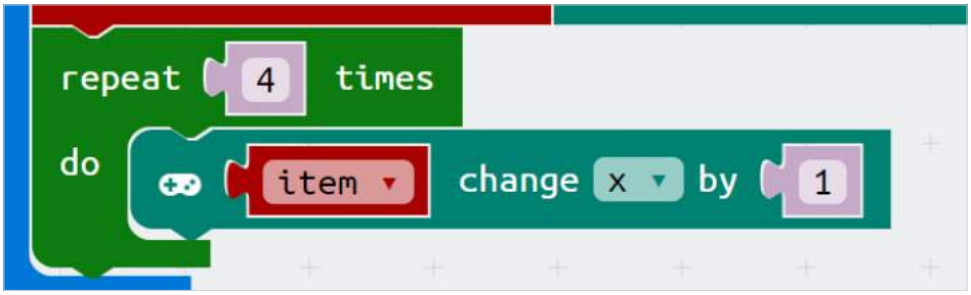
In this step, we will make the Lunar Module fall towards the surface of the Moon.

Activity Checklist

- [ ] We want the lunar_module to fall towards to surface of the Moon. There are four rows between the lunar_module and the landing_site, so we want to move the lunar_module down a row four times. To do this we will create a repeat loop and set it to repeat four times.

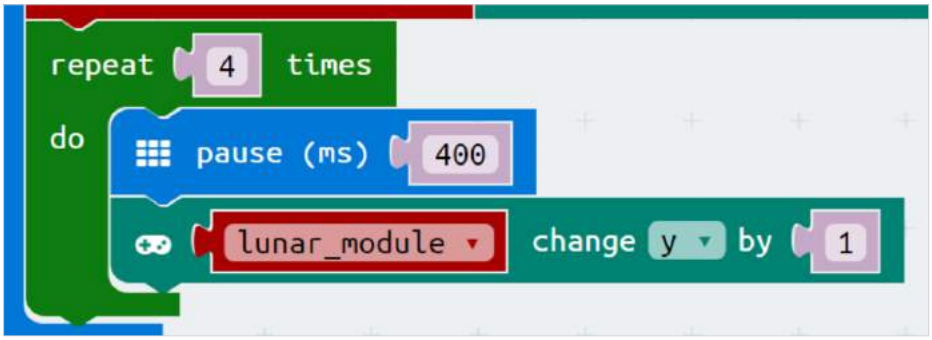


- [ ] Next we want to make the lunar_module drop down one row at a time. We can do this by going to the game panel and selecting item change x by 1 and putting it in the repeat loop.



- [ ] Now we need to update two things, we want to use the sprite assigned to lunar_module, not item, and we want to change y, not x. Use the drop down arrows to change the values to the correct values.

- [ ] Run the simulation. What happens? Both dots are appearing on the bottom row. This is because the lunar_module is falling so fast that you can't see it move! To fix this, we need to add a pause block before our lunar_module starts falling. We'll set the pause time to 400ms (which is just under half a second).



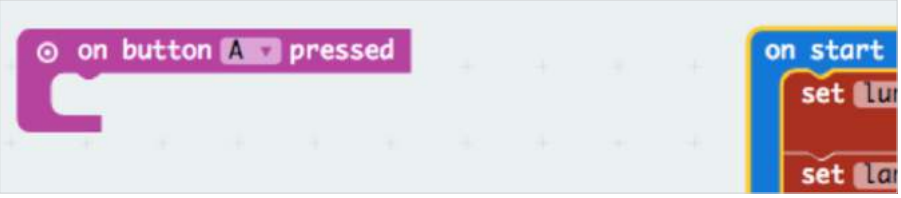
- [ ] Run the simulation again.
You should now see the lunar_module fall.

STEP 3 - CONTROL THE LUNAR MODULE

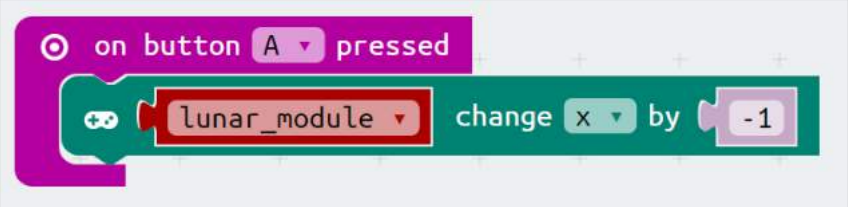
Now that we have the lunar_module falling, we want to be able to control it. We will do this by using the A and B buttons on the micro:bit to control the x value of the lunar_module.

Activity Checklist

- [ ] We need to add the commands that recognise a button press into the program. In the Input panel, drag the on button A pressed block to the script area.



- [ ] Now we need to add code to make the A button move the lunar_module to the left. We will do this by going to the game panel, and adding the item change x by 1 block. We want to change item to lunar_module, and 1 to -1, because left is in the negative x direction.



- [ ] Now we want to make the lunar_module move to the right when we press the B button. Create a similar set of instructions for on button B pressed.
Be sure that we change x by 1, because we want to go to the right, which is the positive x direction.

- [ ] Try it out on the simulation. You can now control the left and right movement of your lunar_module.

code club

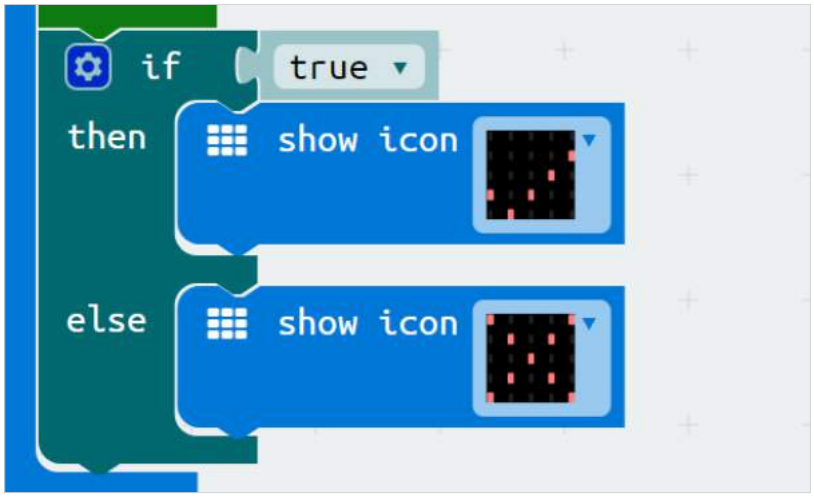
STEP 4 - WIN OR LOSE

Finally we will add the code to see if the user has landed the lunar_module on the landing_site.

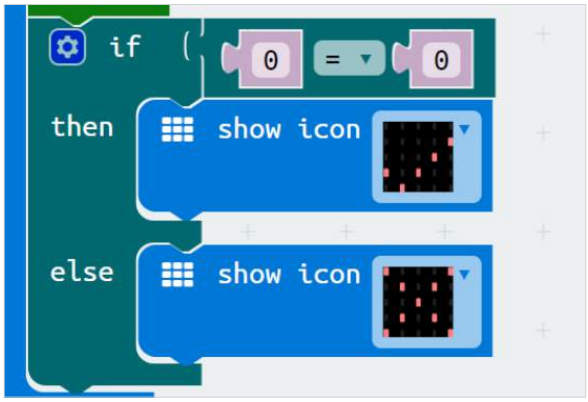
Activity Checklist

- [ ] We want the program to make a decision between whether the user has won or lost the game. To do this, we need to add an if-then-else block from the Logic panel to the end of our "on start" block.

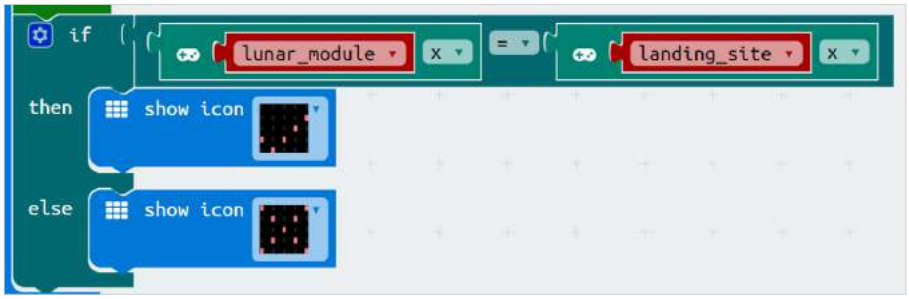
- [ ] Next we want to tell the user whether they have won or lost. We will do this using the micro:bit's built in icons. Add a show icon block to both the then and the else parts of the if-then-else block. Change the icon under then to a tick, and under else to a cross.



- [ ] Finally we need to tell the program how to recognise that a player has won. A player has won if the lunar_module lands on the landing_site. We can tell if this is the case if the x coordinates of both the lunar_module and the landing_site are the same after the lunar_module has finished falling. To check this, we will use the `0 = 0` block in the `Logic` panel. Put this next to the `if` statement (replacing "true").



- [ ] Finally we will need to determine the x positions of both the lunar_module and the landing_site using the `item x` block. Add this to both sides of your equals sign, using the drop down arrows to change one to lunar_module and the other to landing_site.



- [ ] Congratulations! You have completed the 2018 Moonhack micro:bit project. Celebrate by testing your Lunar Module landing skills with your friends!

CONGRATULATIONS!

You have completed the Moonhack Micro:bit project of 2018! High-five your teacher, friends and family!

Now that this project is done there are so many other things you can do.

Extra Activities

- Have a go at the Moonhack Scratch or Python project.
- Create your own "Moon" themed project in Scratch, Python or HTML/CSS!
- Improve this project by completing the challenges for this activity;

Challenge - Difficulty

Is this game too easy or too hard for you? Can you change the speed that the Lunar Module falls so that you can change the difficulty of this project?

Challenge - See the Lunar Module on the Bottom Row

The player can't see the lander on the final row before the tick or cross appears. Can you add a pause block in the program to give the micro:bit time to display the sprite on the bottom row before showing the icon?

Challenge - Download the Code to Your Micro:bit

If you have a micro:bit, can you download the code onto it and play it on your device?
Need help? Check out the micro:bit quick start guide.