Merchant Venturers School of Engineering

Outreach Programme

Micro:bit Space Invaders

Using the Block Editor

Created by

Ed Nutting

Organised by

Caroline.Higgins@bristol.ac.uk

Published on September 13, 2016

Notes to Teachers & Helpers

- This workshop is intended to last 1 to 2 hours.
- This workshop is intended for ages 8^+ (years 4^+).
- The content is intended to be learnt through self-directed, individual work, using this worksheet as a guide.
- The learning platform is the BBC Micro:bit (and the online Block Editor - very similar to Scratch).
- Students can use their own Micro:bit or we can provide a Micro:bit for the session.
- Students should be comfortable using the computer, including knowing how to click/drag blocks in the editor to join them together.
- This workshop teaches the following skills:

  Items marked with an asterisk are directly relatable to the National Curriculum.

  - Basic operations: Turning on/off LEDs, showing numbers
  - Use of sensors/input: Buttons, Tilt-sensor
  * Conditional blocks (if/else-if/else)
  * Basic logic: AND, OR, Greater-than, Less-than, Equal-to
  * Basic mathematical programming: Addition, Subtraction, Division, Max/Min
  * Basic game programming: Events, Control Logic

1 Introduction

Hi! In this short workshop we're going to try to introduce some of the concepts that Computer Scientists use every day to design everything from your games on your mobile to controllers for nuclear power plants.

Let's get started. Each section is made up of four parts:

Actions Stuff for you to do. They are highlighted in blue.

Notes Notes about important stuff you need to be aware of (and possibly remember!). They are highlighted in red.

Questions Questions you should try to answer. Sometimes you'll need to write things down; other times you'll need to build something in the game. They are highlighted in yellow.

Ask a helper or the teacher to check your answers.

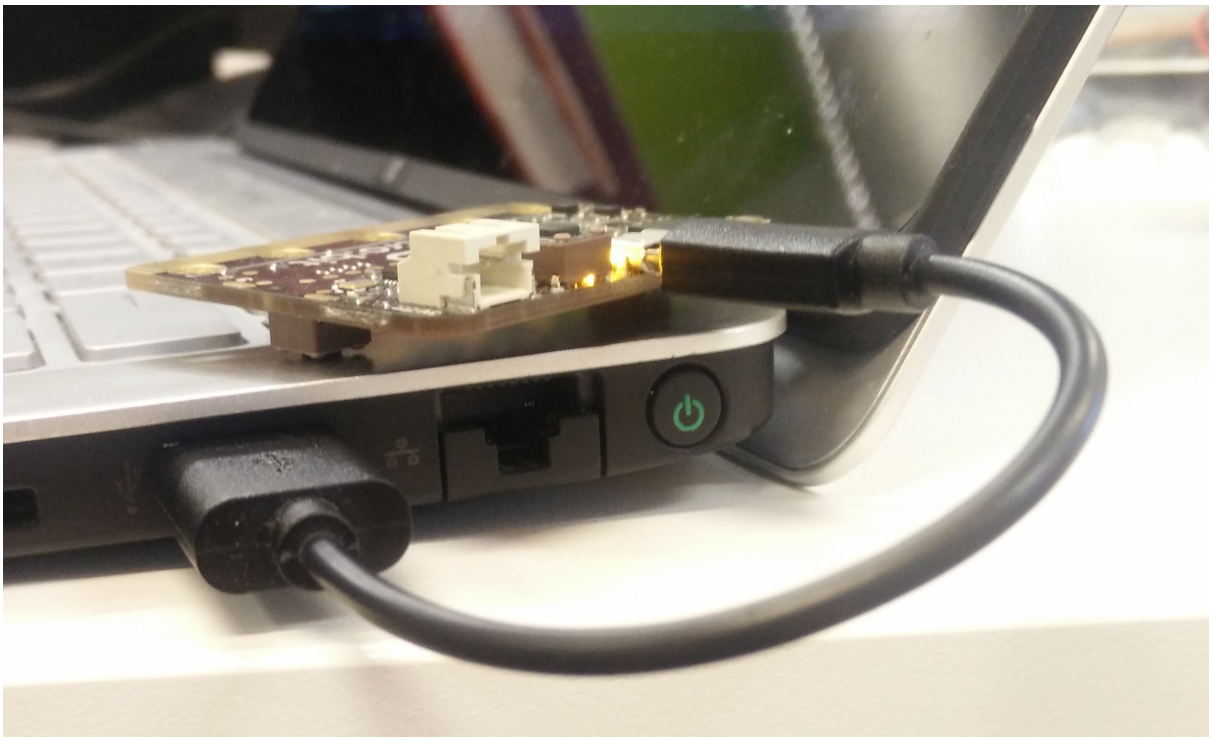
Goals Stuff you should have completed at the end of each section. They are highlighted in green.

We'll also write some information between parts and include plenty of screenshots to help you out.

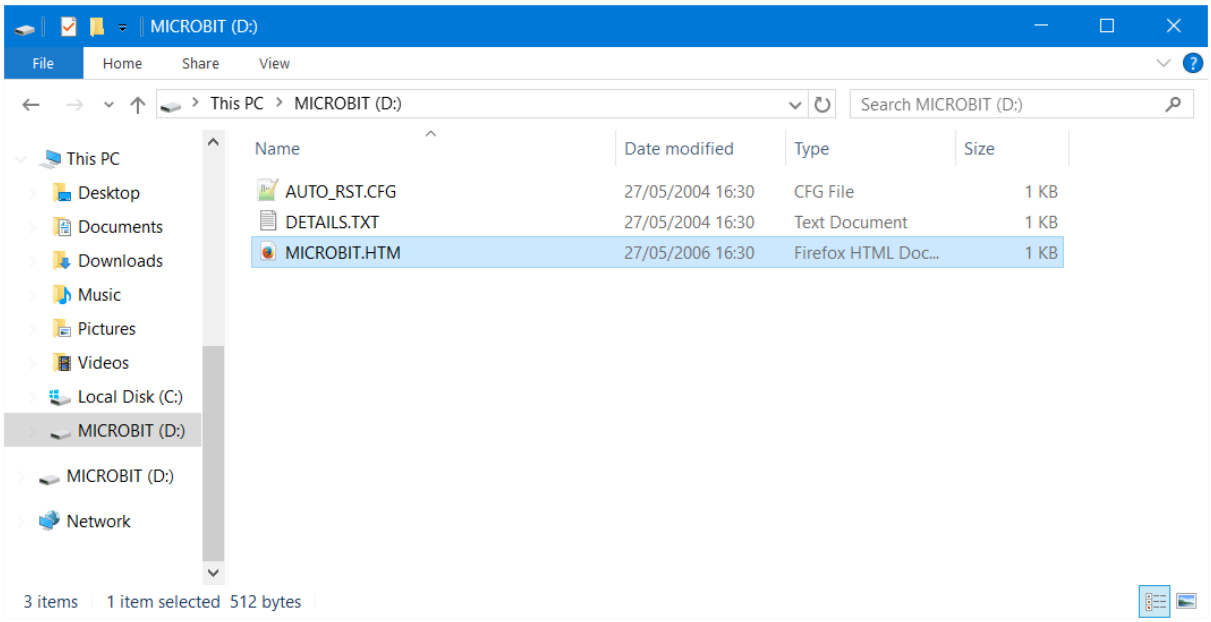
2 Getting Started

Actions

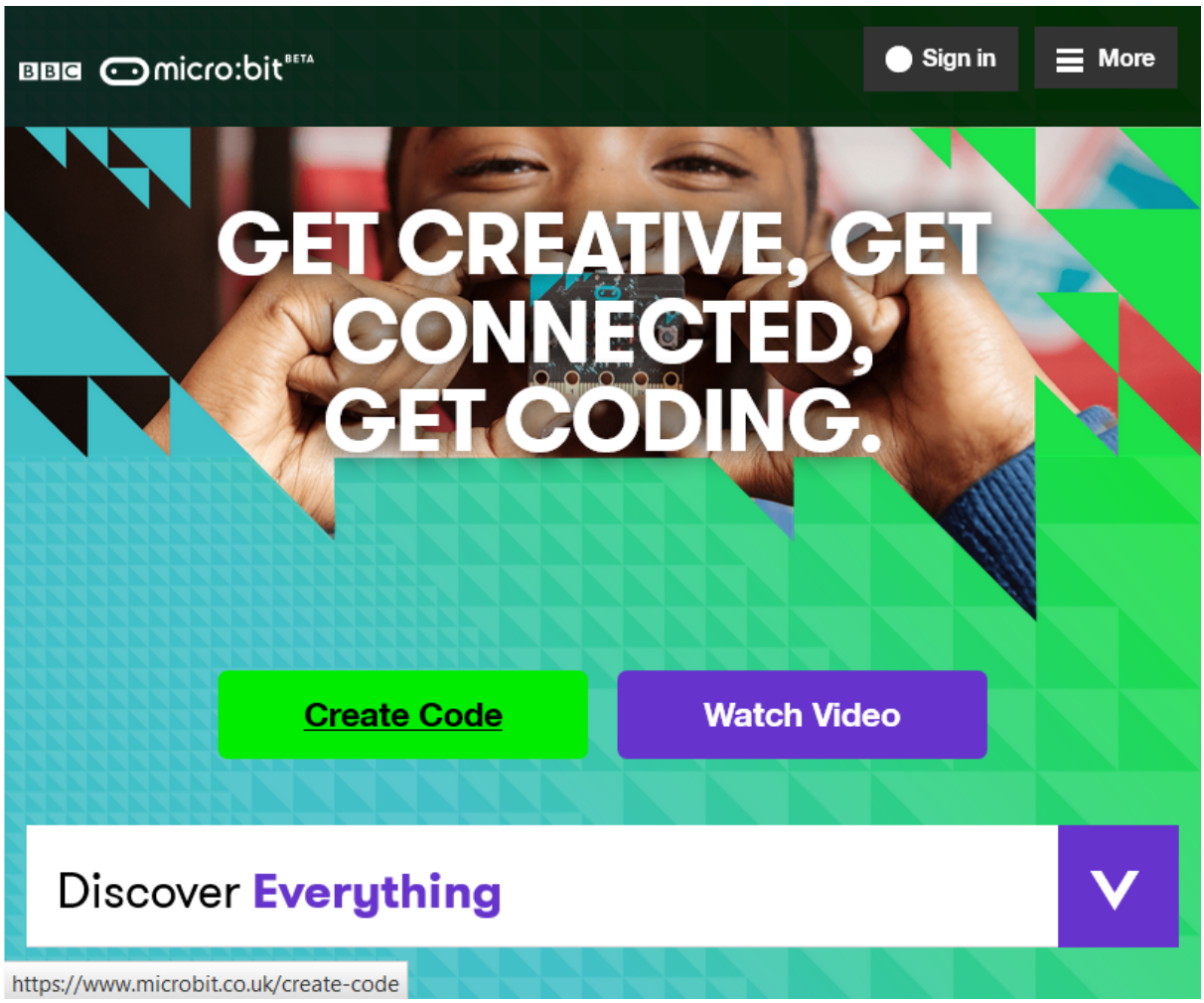
1. Plug your Micro:bit into the computer using the USB cable
2. Look at the Micro:bit files in the file explorer
3. Open the "Microbit.htm" file (e.g. by double clicking) which should take you to the Micro:bit website
4. Click the "Create Code" button
5. In the "Microsoft Block Editor" box, click the "New Project" link
6. At the top of the page, rename your project to "Space Invaders"



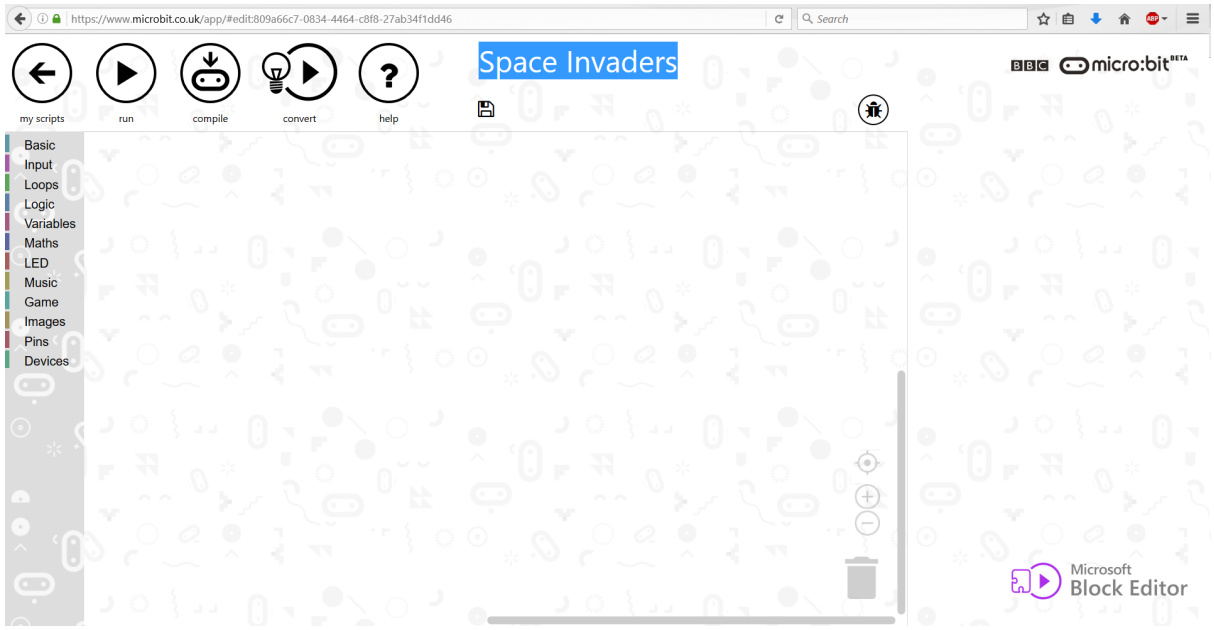
Micro:bit plugged into laptop using the supplied USB cable



Files on the Micro:bit - double click the .htm file to open the website



Create Code on Micro:bit website



Rename your new project to Space Invaders

Goals

Great! You should now have a new Block Editor project open and ready to start creating your game.

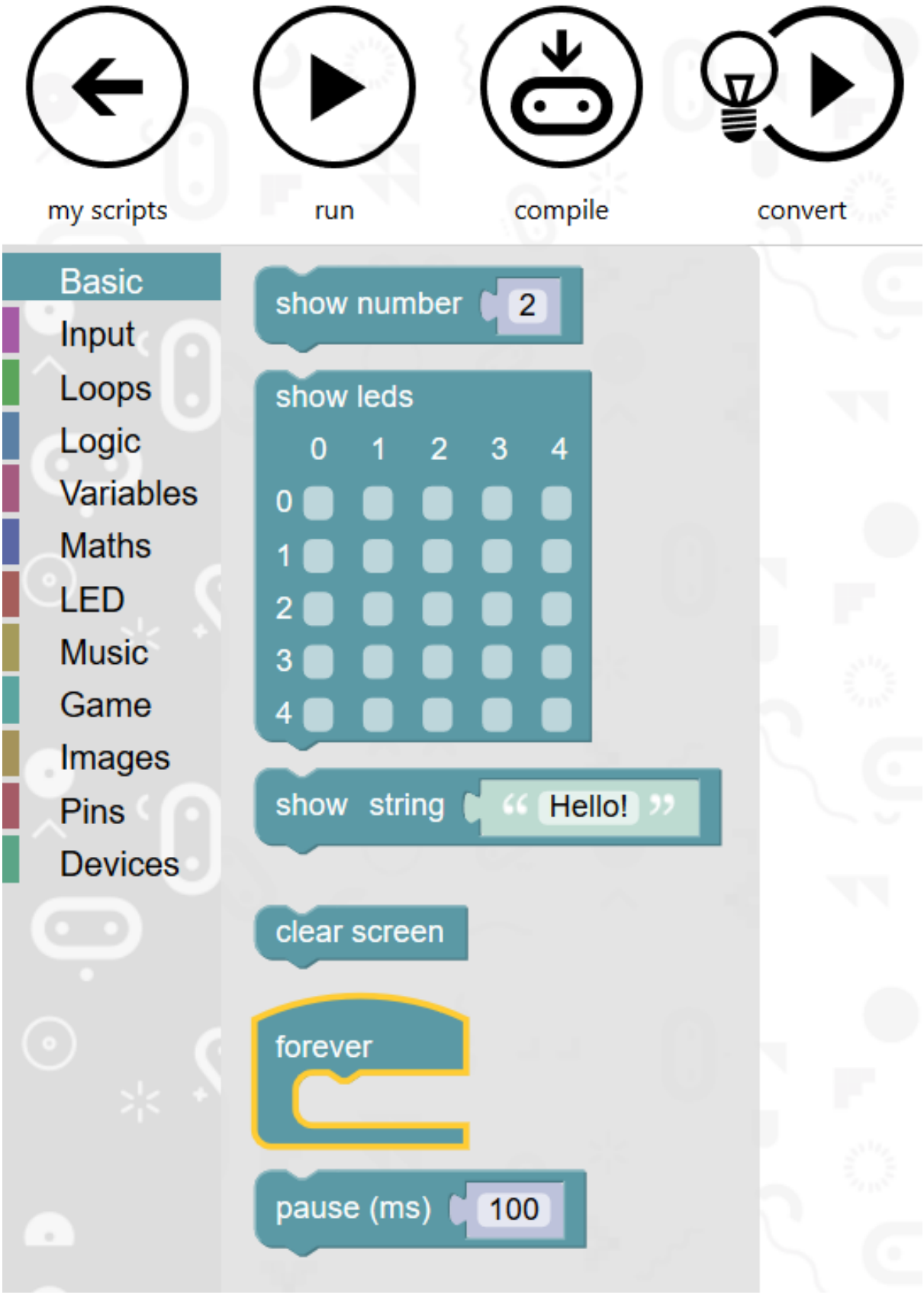
3 Lighting the LEDs

We're going to start simple, by just getting an LED to display on the screen. Then we'll see how we can make a pattern of LEDs blink.

First, we need a loop. A loop is a block of code which repeats. Some loops repeat forever, others repeat for only a certain number of times. We want a "forever" loop.

Actions

1. Click the "Basic" tab on the left
2. Click and drag a "forever" block into the editing area

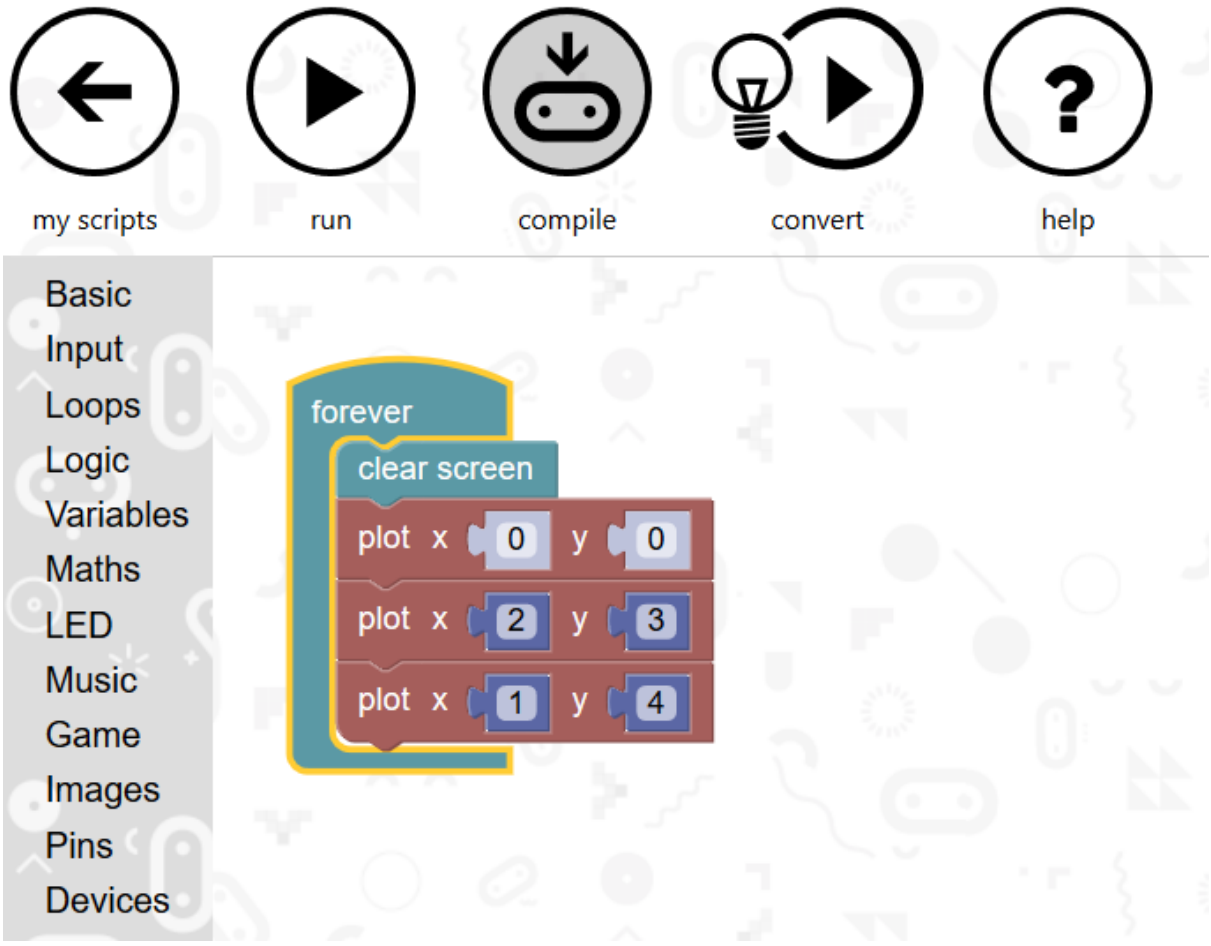


Forever block in the Basic tab

Each time around the loop, we're going to clear the screen and then light up some LEDs again. Later, we'll see how this allows us to move the defender and the space invaders around the display.

Actions

3. Click and drag a "Clear screen" block from the "Basic" tab to the inside of your "forever" block
4. Click and drag a "Plot x/y" block from the "LED" tab to just after the "Clear screen" block
5. Add more "Plot x/y" blocks and change the x/y values to plot more pixels. You can choose any number between 0 and 4 (including 4)!
6. Click "compile" at the top of the screen and download the file to your Microbit.



Goals

You should now see your pattern of LEDs on the display of your Micro:bit!

4 Defender Position

Now we know how to make the LEDs light up, we want to be able to use the bottom row of LEDs to display the defender. The defender will be represented by a single LED in the column the defender can shoot in.

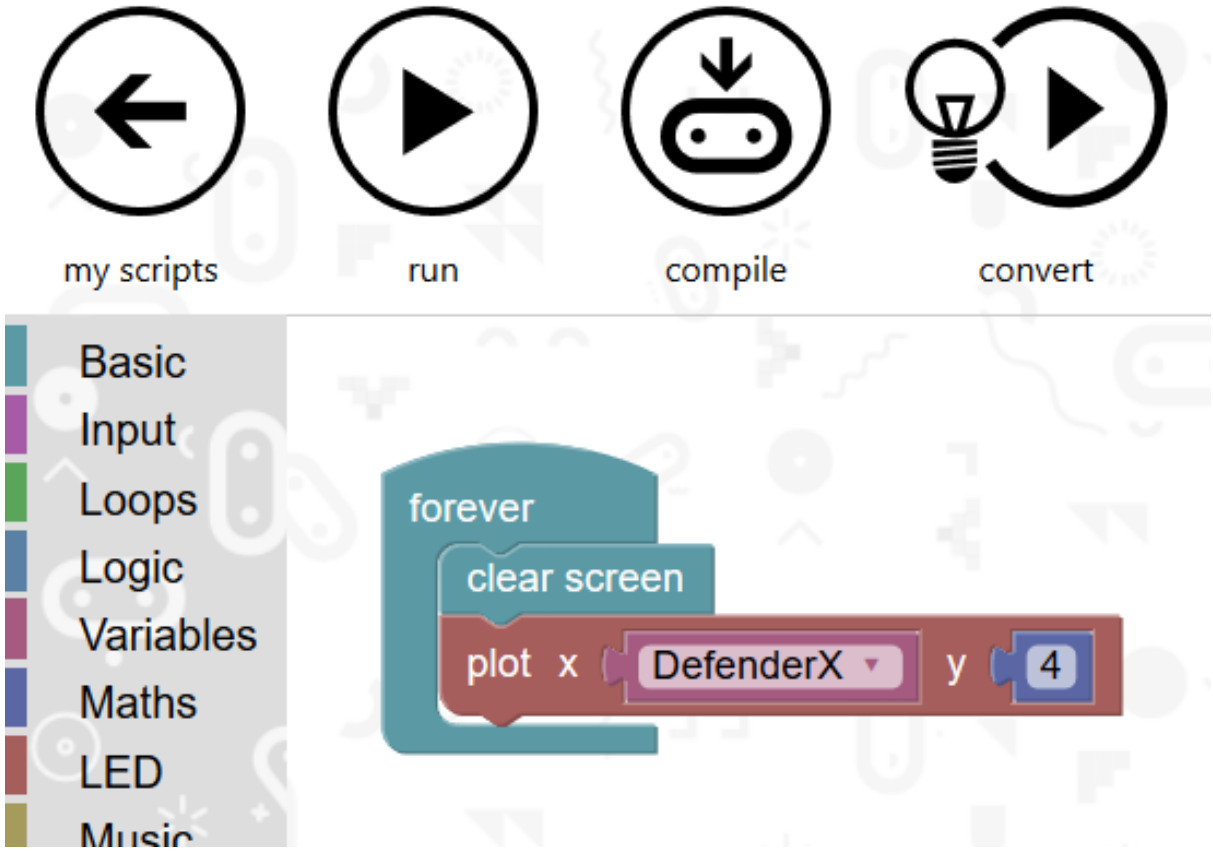
We need to track the defender's position - 0 for left-most column, 4 for right-most and 1,2,3 for the columns in between. To keep track of which column the defender is in, we can use a "variable".

Notes

A variable named thing which can be set to a whole number (0,1,2,3,.....) or a logic value (true/false). We can change the value of the variable at different points in the code.

Actions

1. Click and drag an "Item" block from the "Variables" tab into the "x" value of a "plot" block
2. Click the down arrow next to the word "item" in the block
3. Click "Rename variable..."
4. Rename the variable to "DefenderX" (without the quotes)
5. Set the "y" value of the "plot" block to 4
6. Remove any other "plot" blocks from your "forever" loop



Plotting the Defender

Goals

Click run to see a simulation of your program in the editor. You should see the bottom-left LED light up.

5 Moving the defender

We now know how to use a variable to track the defender's position and make the LED light up. But the value of the variable is never changed - our defender always stays in the bottom left!

We're going to use the tilt sensor to move our defender around. When the board is tilted left, the defender will move towards the left hand side of the display. When the board is tilted right, the defender will move towards the right hand side of the display.

Actions

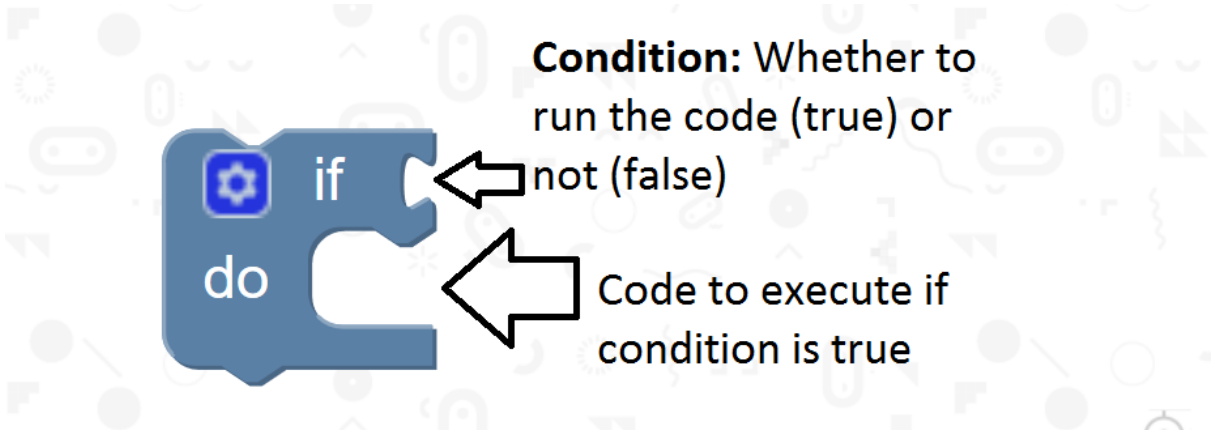
1. Click and drag another "forever" block into the editing area (from the "Basic" tab)
2. Click and drag an "if" block from the "Logic" tab into the "forever" block

Notes

An "if" block is a decision block (often called a Conditional block). You test something, and if that test passes, then the code inside the block executes. Otherwise, the code inside the block doesn't execute and the programs goes to the block following the if block.

Notes

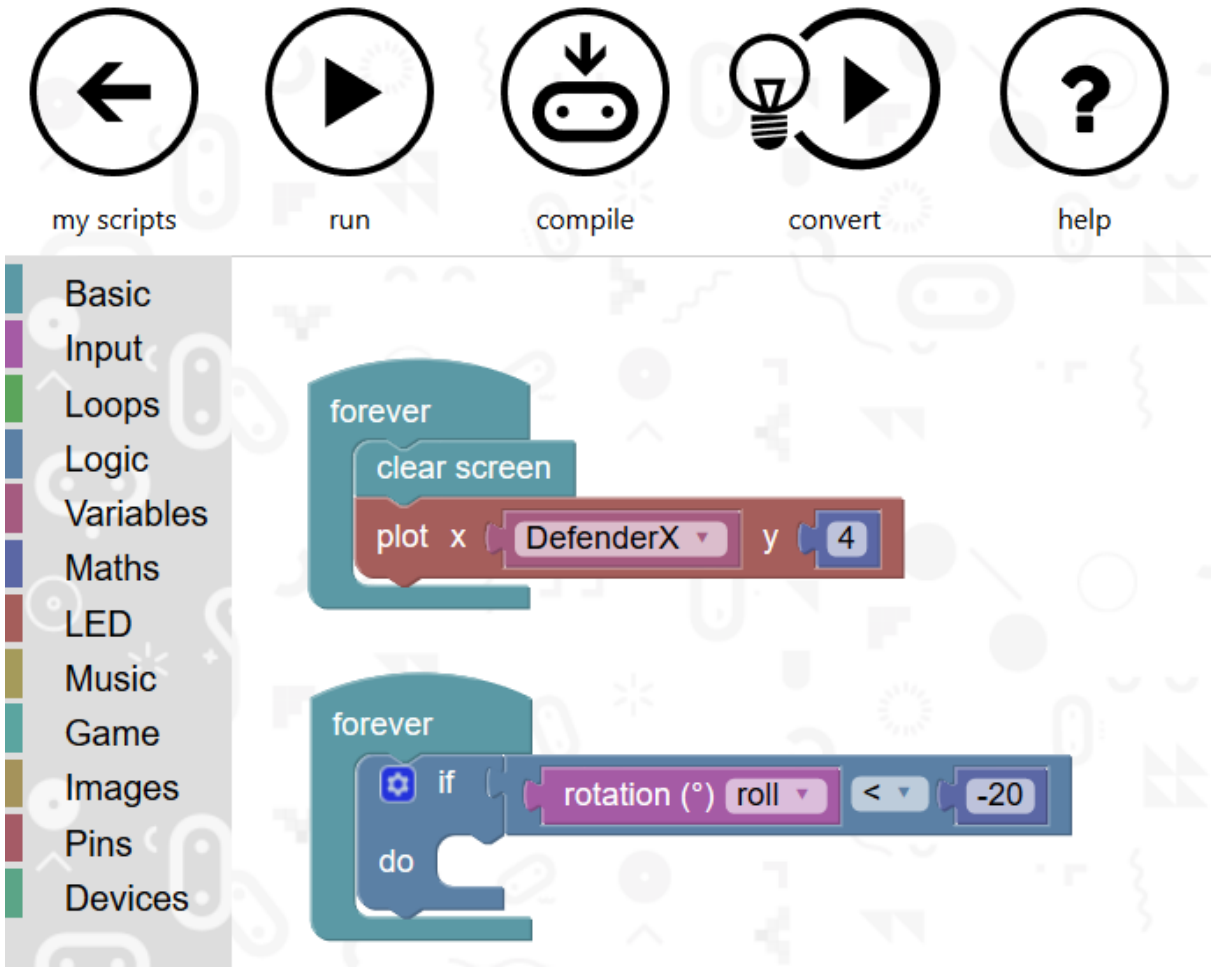
The "test" the if-block uses is called the "condition". It is a logic equation. For example, "is five greater than four" is a condition (to which the answer is always "yes" - usually called "true"). If the condition is "true" the code inside the if block executes, otherwise the condition is "false" so the code inside doesn't execute.



An if-block: Condition determines if the code inside runs.

Actions

3. Click and drag a "less-than" block ("¡" block) from the "Logic" tab to the connector on the right of the word "if"
4. Click and drag a "rotation" block from the "Input" tab to the left-hand "0" of the "less-than" block
5. Click the drop down and select "roll" instead of "pitch"
6. Set the "0" on the right-hand side to "-20"



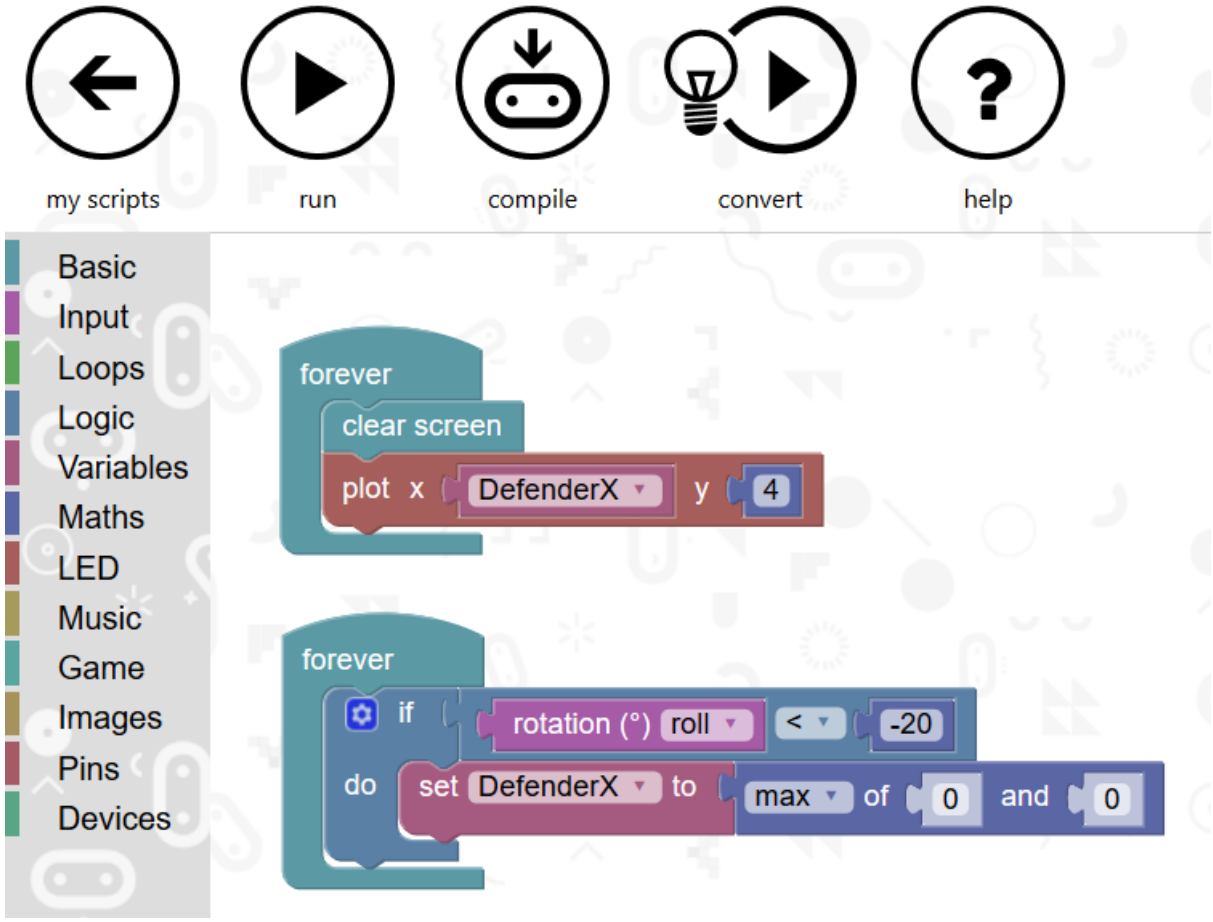
First condition for rotation of the Micro:bit

Notes

This code continuously loops, checking to see if the tilt of the Micro:bit is 20 degrees or more to the left. If it is, it will execute the (empty) code inside the if block.

Actions

7. Click and drag a "set item to" block from the "Variables" tab to inside the "if" block
8. From the drop down, change "item" to "DefenderX"
9. Click and drag a "max" block from the "Maths" tab to the right-hand connector of the "set" block



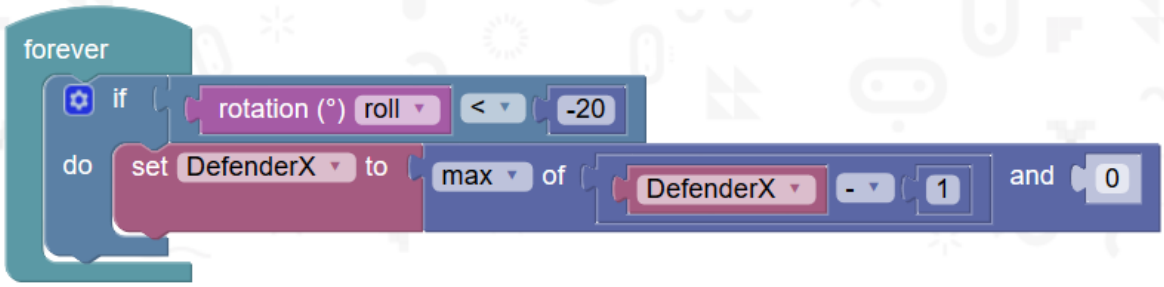
First use of a max-block

Notes

"Max" is a clever maths operation which picks the larger of the two numbers it is given.

Actions

10. Click and drag a "subtract" block ("-" block) from the Maths tab to inside the left-hand value of the "max" block
11. Click and drag a "DefenderX" item block to inside the left value of the "subtract" block
12. Set the right side of the "subtract" block to 1



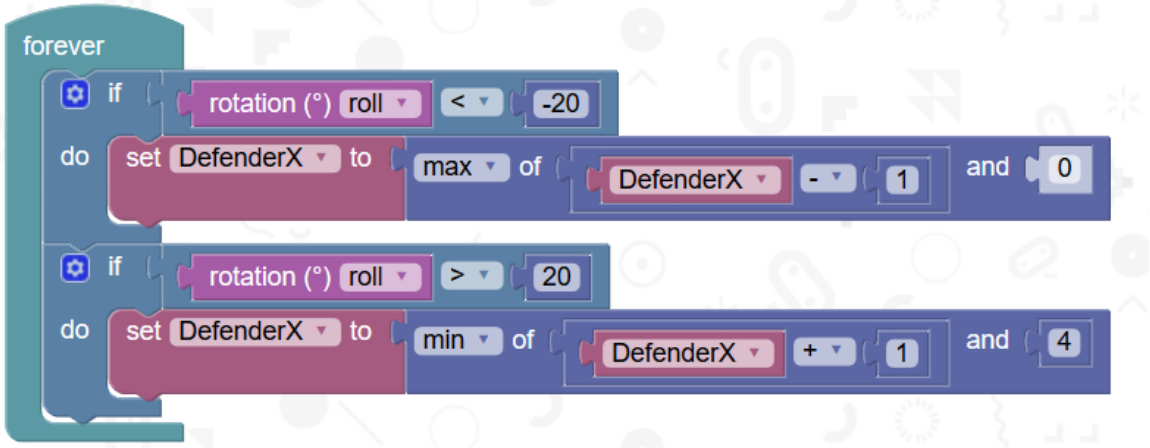
Code for moving the defender left

Notes

The code inside the if block will now set the "DefenderX" variable to the maximum of "DefenderX - 1" and "0". In other words, if the defender is in the left-most column (column 0) it will stay there. If it is not in the left-most column, the defender will move one column to the left. E.g. if it is in column 3, it will move to column 2.

Actions

13. Copy and paste the "if" block to after itself (use Ctrl+C to copy, Ctrl+V to paste)
14. Set the condition to use ">" (greater-than) instead of "<" (less-than)
15. Set the right hand comparison value to "20" instead of "-20"
16. Set the maths operation to "min" instead of "max"
17. Change the "subtract" operation to "addition"
18. Set the right hand "and" value of the "min" block to "4"



Code for moving the defender left and right

Notes

The tilt direction in the simulator in the editor is the opposite to the real Micro:bit.

Goals

Compile and save your code to your Micro:bit - you should now see the Defender move left and right on the bottom of the display when you tilt the Micro:bit left and right.

Notes

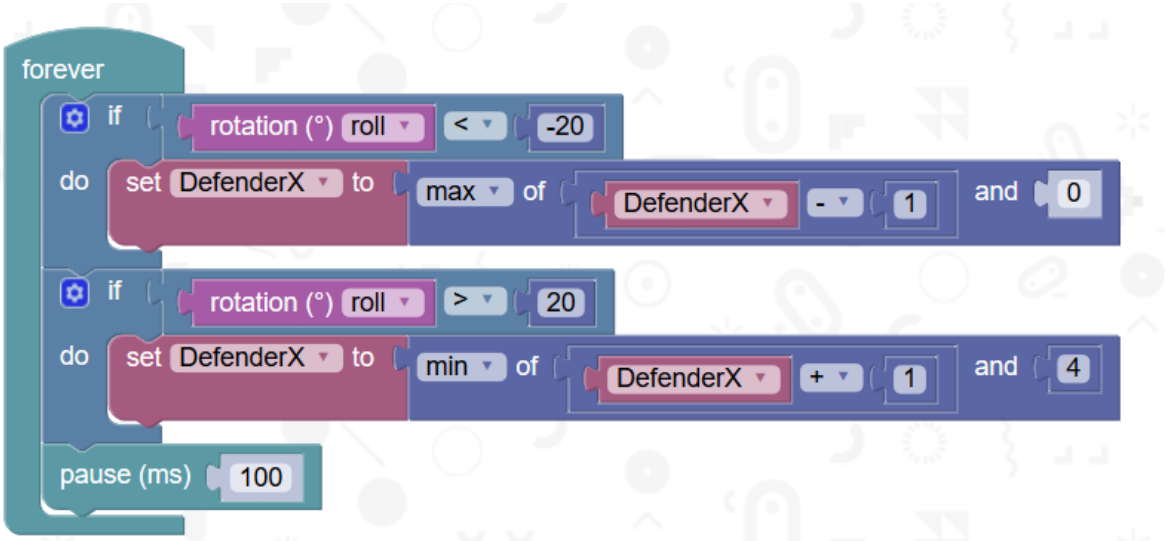
You may notice the defender moves very quickly - too quickly! We can fix that by adding a delay to the loop.

Notes

A delay pauses the execution of the blocks for a length of time. The length of time is in milliseconds. There are 1000 milliseconds to 1 second.

Actions

19. Click and drag a "pause" block from the "Basic" tab to the end of the "forever" loop after your "if" blocks
20. Set the value to "100" (if it is not already 100)



Improved code for moving the defender left and right

Goals

Compile and save your game to your Micro:bit. You should now be able to control the defender position so that you can put it in a particular column by tilting left and right.

6 Starting the game

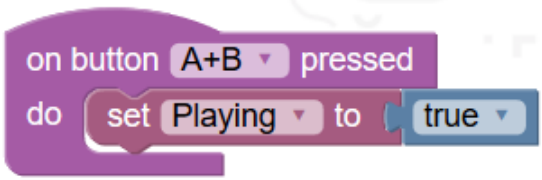
At the moment, as soon as the Micro:bit switches on, the game will begin. This is a problem because you may want to wait until the player is ready. We can achieve this by using another variable and waiting for the A and B buttons to be pressed before starting the game.

When both the A and B buttons are pressed together, we will set a new variable called "Playing" to "true" to indicate the start of the game. When the player dies during a level, it will reduce their number of lives by 1. When they have no lives left, we will set "Playing" to false to signal the end of the game.

Let's start by doing the game start signal - buttons A and B being pressed together.

Actions

1. Click and drag an "on button A pressed" block from the "Input" tab onto the editing area
2. From the drop down, change "A" to "A+B"
3. Add a "set item to" block (from the "Variables" tab) into the "do" section of the button press block
4. From the "item" drop down, select "New variable..."
5. Call the new variable "Playing" (without quotes)
6. From the "Logic" tab, click and drag a "true" block to the right hand connector of the "set Playing to" block.

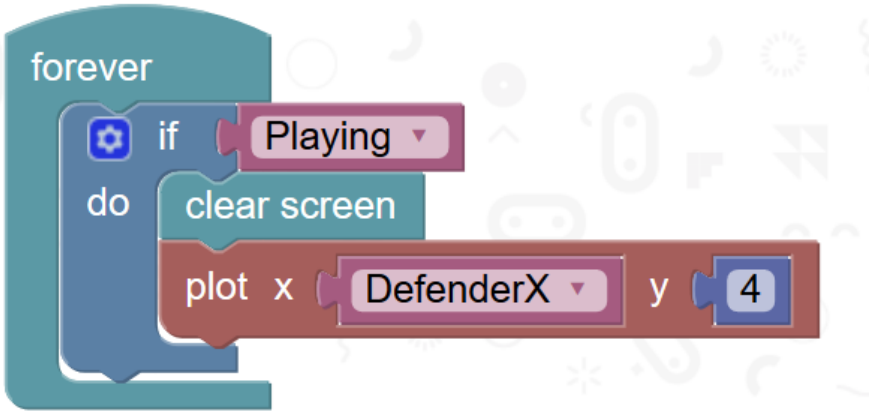


Code to detect user pressing buttons A and B to start the game.

We also only want to draw the player and invaders on the screen when the game is being played. We can make sure they are only drawn when the game is being played by putting an if-block around our draw code. The condition of the if block will be our variable - "Playing".

Actions

7. Click and drag an "if" block from the "Logic" tab into the editing area
8. Click and drag the "clear screen" block (which should drag the blocks after it too) from the "forever" loop into the "if" block
9. Click and drag a "Playing" variable block from the "Variables" tab into the right-hand connector of the "if" block condition
10. Click and drag the "if" block (and all its contents) back into the empty "forever" loop



Only draw the player / invaders when playing the game

Goals

Compile and save your game to your Micro:bit. When the Micro:bit powers on, the screen should be blank. Press the A and B buttons together - you should now see the defender at bottom of the display.

7 Invaders Positions

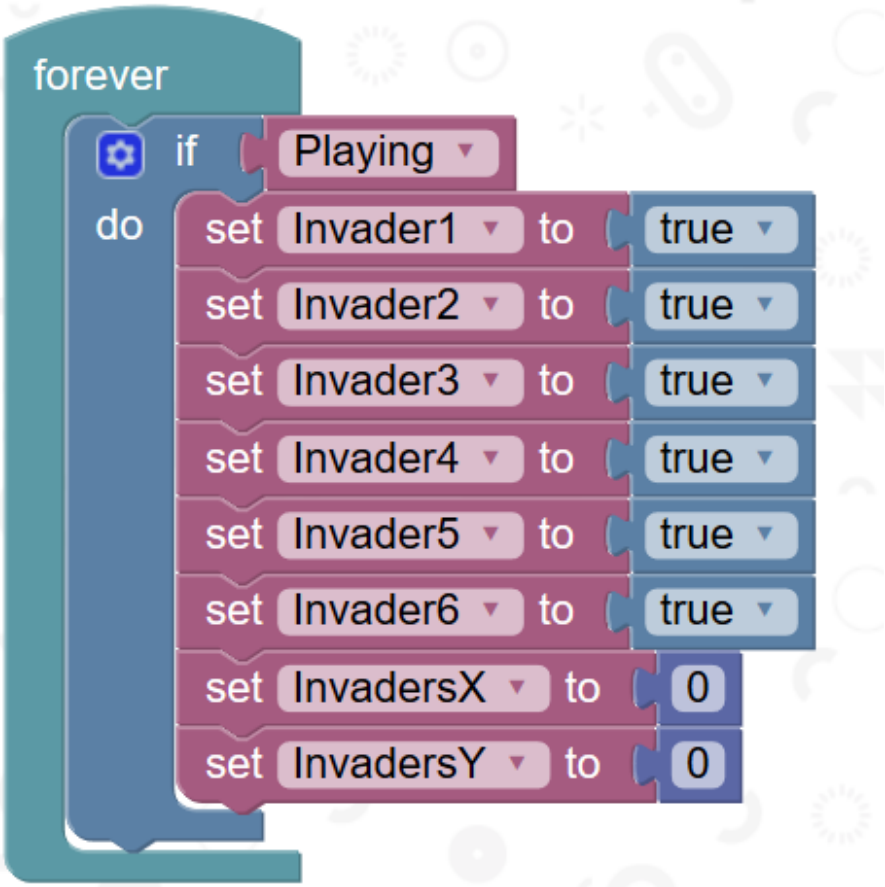
There will be six invaders, that will start at the top-left and work their way across as a block from left to right. When they reach the right hand side, they will jump back to the left again.

We will need to keep track of which invaders are alive and which are dead. We can do this using six variables named "Invader1", "Invader2" and so on up to "Invader6". If an invader variable is "true", it will mean the invader is alive. If it is "false", the invader is dead.

We will also need to know where the invaders are. We will use two variables for this: one for the X-location (column number) that the left-most invader is in, and one for the Y-location (row number) that the top-most invader is in.

Actions

1. Add blocks to your game's code so that it matches the screenshot below. You will need to add a new "forever" block.

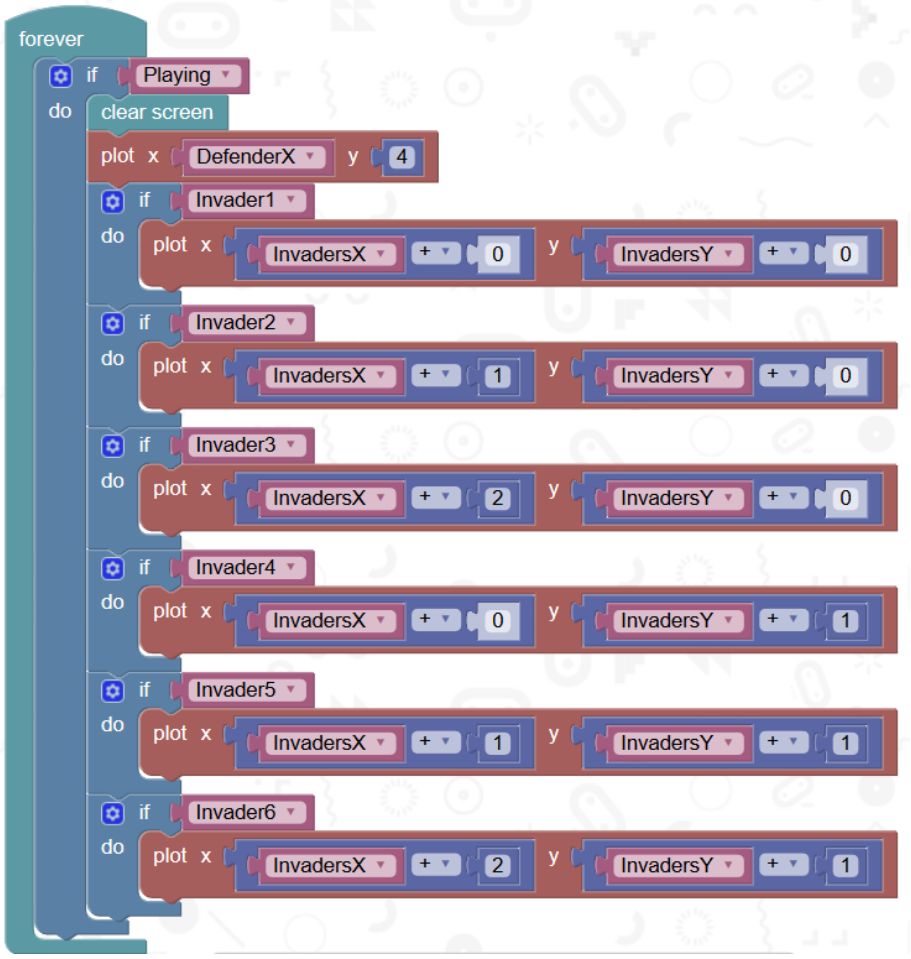


Initialising variables for tracking the invaders

Now we need to plot the invaders - i.e. to display them on the LEDs. We can do this by using the current location of the invaders and whether each invader is alive. If the invader is alive, we turn on the relevant LED. Otherwise, we leave the LED off.

Actions

2. Add blocks to your game's code (extending the section we made earlier for displaying the defender) so that it matches the screenshot below.



Code for displaying the invaders and the defender

Goals

Compile and save your game to your Micro:bit game to your Microbit. When the device powers on, press the A and B buttons together. You should now see the defender moving as before and the six invaders (staying still) in the top-left corner. (We'll make them move in the next section!)

8 Controlling the game

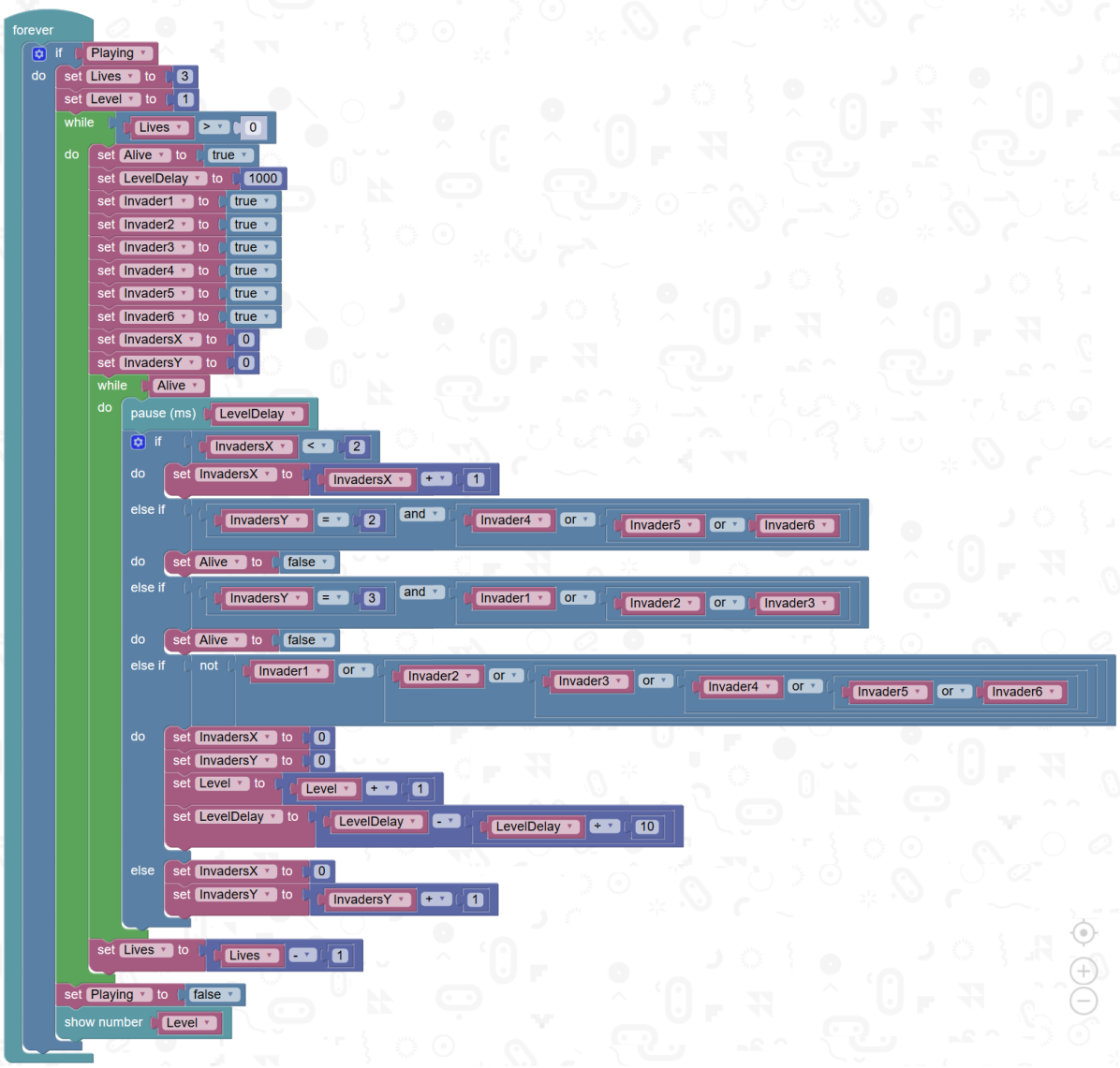
Our game is almost complete - we just need to add the final two sections of code. The first one is what we call the "Game Logic". The second is the section to enable the defender to shoot players.

The Game Logic is the main section of control code which determines how the game proceeds (as opposed to how its drawn or how the player moves). Game Logic also includes the number of lives a player has and what determines when a player dies.

Our game logic breaks down into two loops: An outer loop that allows the player to keep playing while they have lives left. And an inner loop: Which progresses through harder/faster levels until the player dies.

Actions

1. Add blocks to your game's code so that it matches the screenshot below.



Code for the game logic

Questions

1. Can you work out what is going on in this code?
2. Before you run the code, from your understanding of the code, do you think the space invaders will move from left to right or from right to left?
3. Before you run the code, from your understanding of the code, how many lives will the player get?
4. Before you run the code, from your understanding of the code, will the speed of each level increase? If so, by how much each time?

Goals

Compile and save your game to your Micro:bit. Press the A and B buttons together to start the game. You should see the invaders move around the display and the defender move at the bottom of the display.

Questions

5. Now you've tried running the code, which direction do the space invaders move?

6. How many lives did the player get?

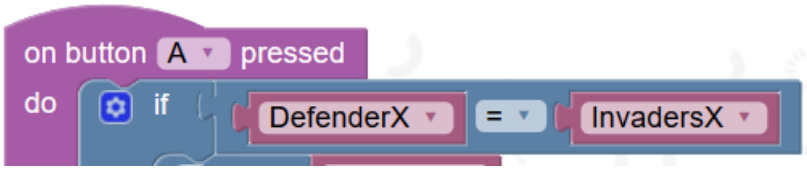
The speed of the current level is reset when the player dies, so you won't have seen the level-speed increase yet.

9 Shooting invaders

This is the last part to our code - shooting the invaders! We're going to program our game so that, after the game has started, when button A is pressed, the bottom-most invader that is in the same column as the defender will be shot.

Actions

1. Click and drag an "on button A pressed" block from the "Input" tab onto the editing area.
2. See if you can work out what if-blocks and conditions you will need to add to the "on button A pressed" block, to determine which invader the defender is underneath (if any)



See if you can work out the last section of code!

Notes

Hint: Think about the position of the defender (DefenderX) and the position of the invaders (InvadersX). You will want to compare them somehow!

Notes

Hint: The bottom-most invader should be killed first. Once it is dead, the top row invader can be shot.

Goals

Test your new code. Does it work? If not, keep trying - code, test, repeat - that's how real computer scientists and software engineers work.

Actions

3. Try to get a working version of shooting the invaders. If you're having problems, ask a workshop helper.

Goals

Congratulations! You've finished making the game!

Questions

7. Are there any improvements to the game you can think of? Have a go at programming them. Remember: code-test-repeat.

10 Wrap-up

We hope you enjoyed this workshop! There's another workshop called Microbit Python Space Invaders that will teach you how to create a more advanced version of this game in Python. Ask your teacher about it!