'Do it, pass it' micro:bit game

Overview

This activity fulfills **Stage 2**, **Requirement 3** of the **Digital Maker Staged Activity Badge** ('Make a simple digital creation that uses code to interact with the wider world through inputs (such as buttons or typing on a keyboard) and outputs (such as a computer screen, sound, or lights)').

Young people will use a micro:bit to make a 'Do it, pass it' game. In the game, players will need to perform an action that will be shown on the micro:bit, then pass the micro:bit to the next person. When the timer runs out, the group will see their score and can keep playing to try to beat it.

30 – 60 minutes (plus time to play the game)

Flexible (ideally two young people per creation). Larger groups can play the game once it is built.

If you're running this activity without access to WiFi, you will need to download the software ahead of time. You may also wish to print handouts.

Wherever you have access to computers. The game can be played indoors or outdoors once it is built.

You will need:

- Computer or mobile device
- Activity handouts
- micro:bit MakeCode editor
- One micro:bit
- One USB A to micro USB B cable for programming (not needed for mobile devices)
- One battery pack and batteries
- One crocodile clip lead (optional)
- One torch (a mobile phone torch is fine)
- Additional input and output components, such as buttons and buzzers (optional)
- Craft materials, such as elastic bands, cardboard, sticky tape and scissors (optional)

If your meeting place has WiFi

Run the micro:bit MakeCode editor in a web browser using this link: **rpf.io/makecode**

If your meeting place doesn't have WiFi

Download the micro:bit MakeCode app beforehand (Windows only); refer to the micro:bit guide at **rpf.io/scouts-microbit** for instructions.

Key messages

- Computers talk to the world through inputs and outputs.
- The micro:bit is a small computer with built-in inputs and a display of 25 LED lights. You can connect more inputs and outputs to the micro:bit using its pins (labelled **0**, **1** and **2**).
- You can make your own electronic gadgets with a programmable device, code, electronic components, and other materials.

‘Do it, pass it’ micro:bit game

Leader instructions

- If they have not used them before, introduce the young people to the micro:bit. Explain that micro:bits have two buttons (for input) and a small LED display (for output), but they also have sensors, such as a light sensor and an accelerometer that can detect movement. Also, point out the pins (labelled **0**, **1** and **2**) that can be used to connect more inputs and outputs.
- Explain that young people will be creating a game. They will program the micro:bit to display images that correspond to actions, such as pressing a button or shining a light. Each player must perform an action and then pass the micro:bit to the next player. The goal is to score as many points as possible before the timer runs out.
- If the young people have not used the micro:bit with MakeCode before, demonstrate the editor and show them how to connect the micro:bit with a USB cable and download a program. Explain that blocks are colour-coded to make them easy to find.
- Ask the young people to open the MakeCode editor on their own computers and connect their micro:bits.

Alternatives

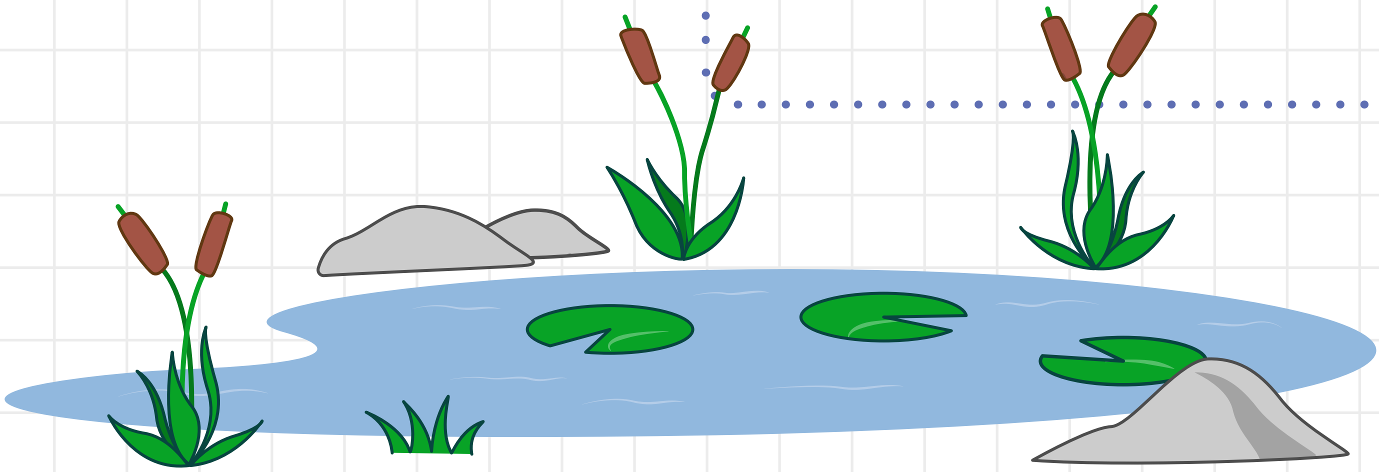
- If you don’t have access to micro:bit devices, young people can use the micro:bit simulation in the MakeCode editor to create the project.

Safety

Supervise young people when they’re online. Give safety advice, and remind them about staying safe online at home.

Adaptability

- Young people do not have to finish the full project to have a complete game.
- Depending on how much time is available, you can run a simple version of the project, or you can add more components and craft materials to extend the project.
- Young people with more advanced coding skills could create the same project with JavaScript in the MakeCode editor or with Python in the Mu editor.



‘Do it, pass it’ micro:bit game

Leader instructions (cont.)

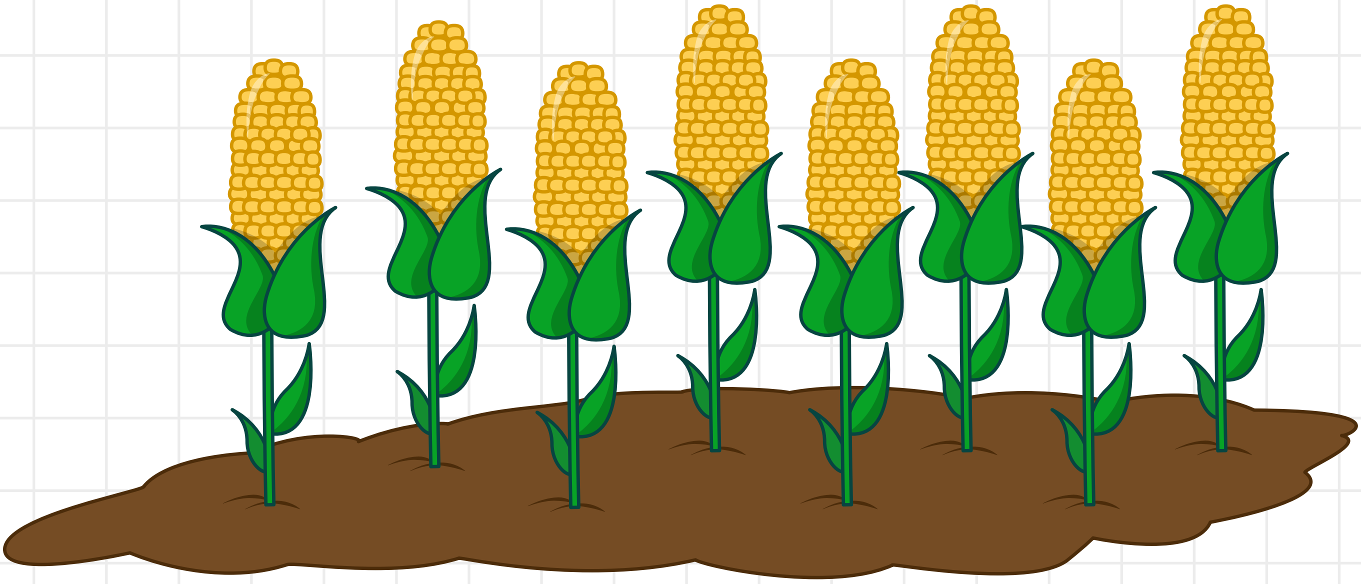
- Let them work through the rest of the instructions. Look out for the following common errors:
  - Check that the maximum in `pick random` matches the total number of actions, otherwise some actions won’t get chosen.
  - Make sure that the metal part of the crocodile clip touches the pin. If the pin touches the sleeve, it won’t work.
- When pairs finish their projects, they can try out their games with others in larger groups.
- If you have time, you can provide more electronic components and craft materials so that the young people can extend the game.
- Close with a discussion about how it felt to make their own electronic game instead of buying one. Can they think of other games that they could make that would fit in with their hobbies and interests?

Did you know

The Simon electronic memory game was first made in 1978. It has a microcontroller (a mini computer like the micro:bit), electronics components code, and a plastic case. The ‘Bop It!’ game introduced in 1996 uses similar technology to create a game with voice commands and a variety of inputs. Now, technology has reached the point where young people can make their own electronic games using programmable devices such as the micro:bit, Arduino, and Raspberry Pi.

Discuss

What electronic toys and games have you played with?



‘Do it, pass it’ micro:bit game

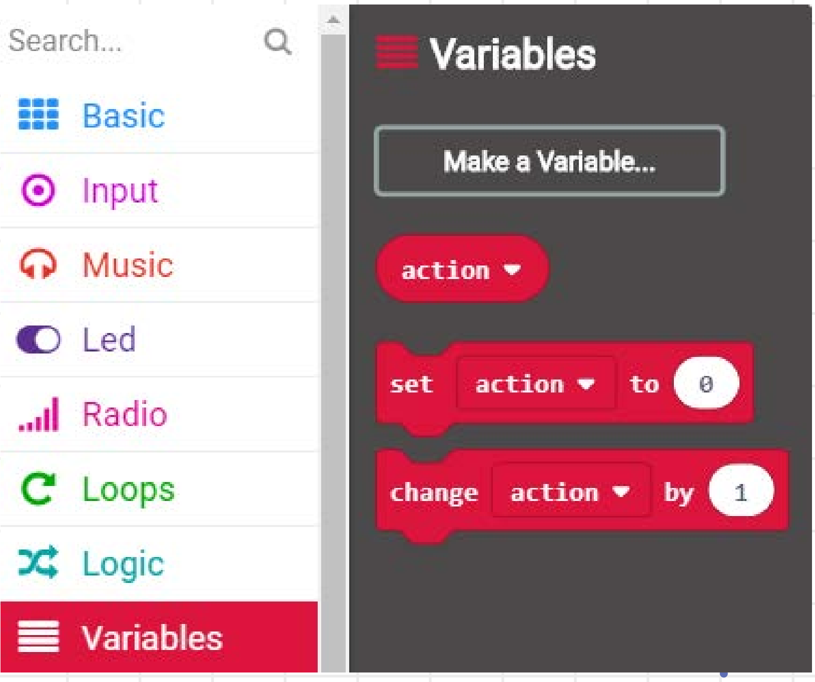
Display an action

The micro:bit has a display of 25 red LED lights that provide an output. You will program the display to show an action for the player to perform.

First, the display will show a letter A to tell the player to press button A on the micro:bit, or a maraca (shaker) to show that they need to shake the micro:bit.

1 Go the website **rpf.io/makecode** to open the MakeCode editor. If your device is not connected to the internet, open the MakeCode app.

2 Make a new variable called `action`. This variable will store the action that the player needs to perform:

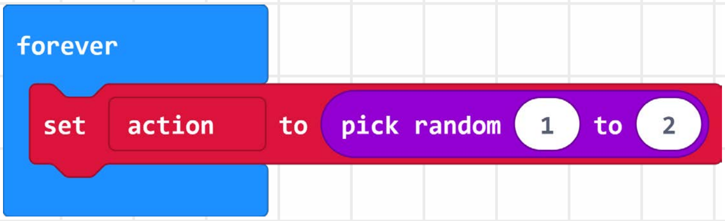


Later you will use new blocks in the `Variables` section to set and change the action variable.

Scouts Raspberry Pi

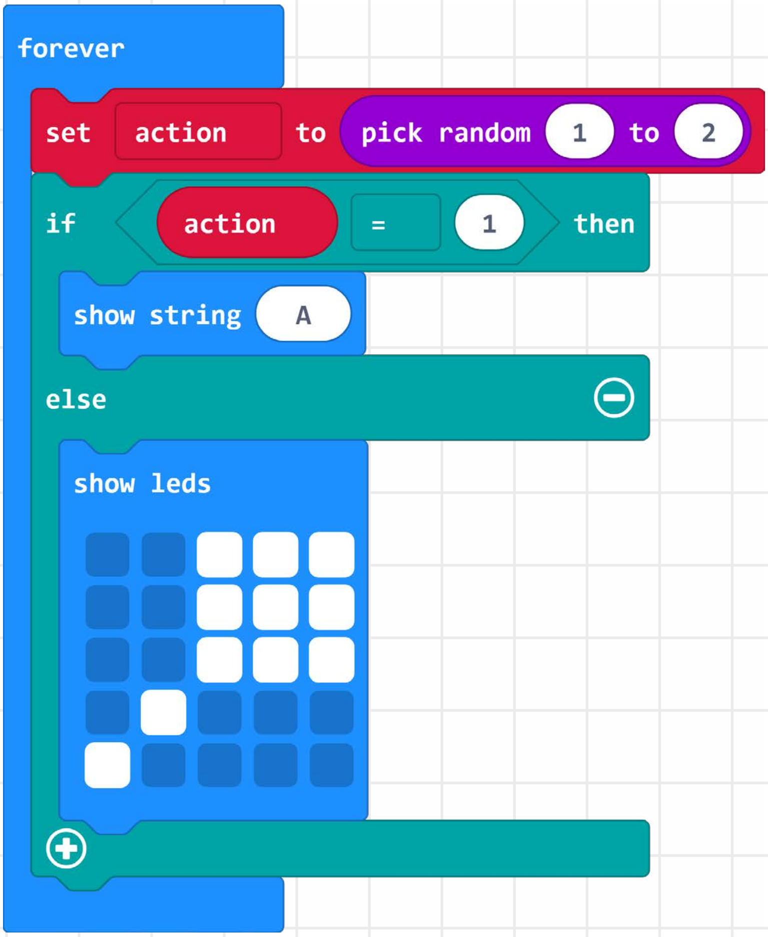
'Do it, pass it' micro:bit game

3 Add a forever block by dragging it into the workspace on the right. To randomly choose between two actions, add the set action block and a pick random block to your forever loop. Change the values in the pick random block to 1 and 2. In the next step, you will set up the two actions.



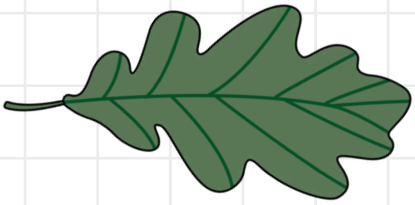
```
forever
    set action to pick random 1 to 2
```

4 In this step, you will add code to your forever loop to display an image that will represent the action that the player needs to perform. Add an if then else block, then add in an equals comparison block, a show string block after if and a Basic show leds block after else. Click on the squares in the show leds block to make an image of a maraca. Your code should look like this:



```
forever
    set action to pick random 1 to 2
    if action = 1 then
        show string A
    else
        show leds
            . . # # #
            . . # # #
            . . # # #
            . # . . .
            # . . . .
```

5 Your code will automatically run in the simulator. You should see either the letter A or an image of a maraca appear at random.

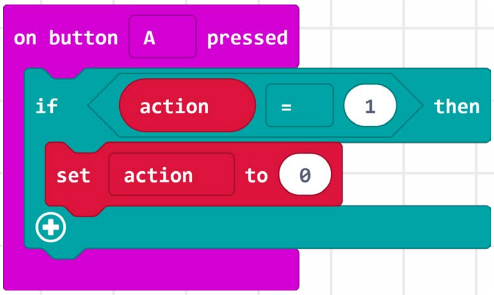


‘Do it, pass it’ micro:bit game

Get input

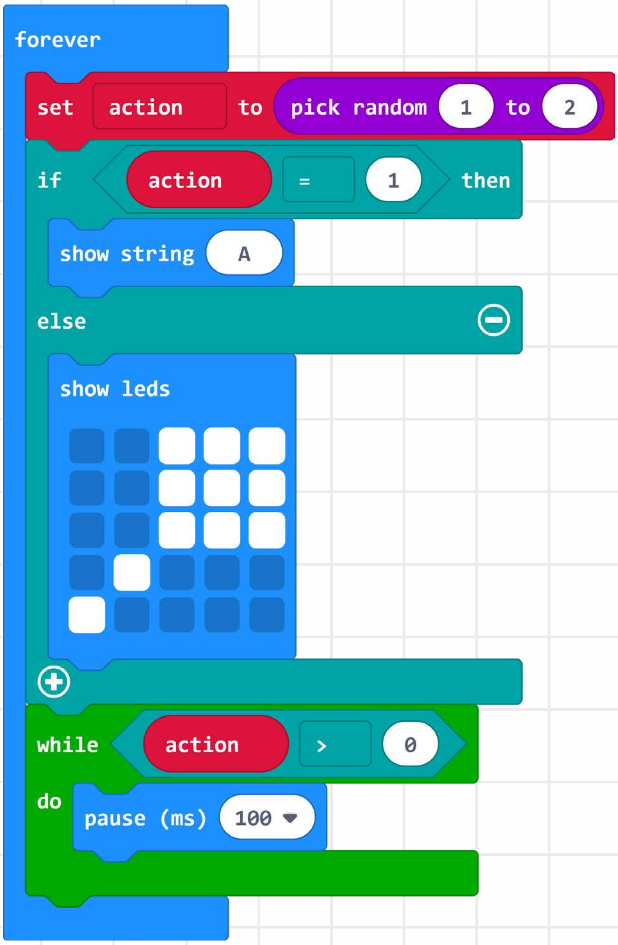
Next, you will add code to detect when the player provides input by pressing button A or shaking the micro:bit.

1 Add an `on button A pressed` block to detect when **button A** is pressed. Then, add an `if` block and set the `action` variable to 0 to show when the action has been completed:



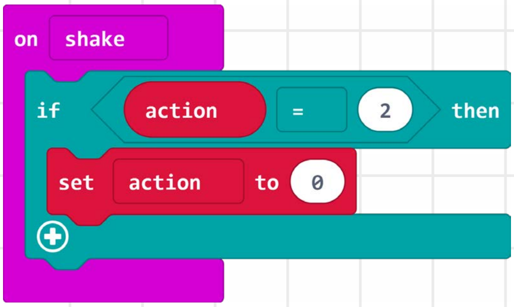
```
on button A pressed
    if action = 1 then
        set action to 0
    ⊕
```

2 Add a `while` block to the bottom of your `forever` loop. This loop will keep going while `action` is greater than 0. So the program will continue until the `action` variable gets set to 0. This will make sure that the player can only see the next action after they have completed the first action.



```
forever
    set action to pick random 1 to 2
    if action = 1 then
        show string A
    else ⊖
        show leds
            . . # # #
            . . # # #
            . . # # #
            . # . . .
            # . . . .
    ⊕
    while action > 0
    do
        pause (ms) 100 ▾
```

3 Add a block to detect the **shake** action (action 2). Then, add an `if` block and a `set action` block to show when the **shake** action has been completed.



```
on shake
    if action = 2 then
        set action to 0
    ⊕
```

4 Now, you can test your game in the simulator. If you see an A, press **button A**, and if you see a maraca, click on **Shake**.

Note that you can get the same action more than once, so if you see the same action again, you need to repeat it.

You can also try your game on your micro:bit:

1 Connect the micro:bit to your computer with a USB cable.

2 Click on **Settings**, then **Pair device**.

3 Click on **Download** to transfer your code to the micro:bit. From now on, you can just click on **Download** to transfer your code.

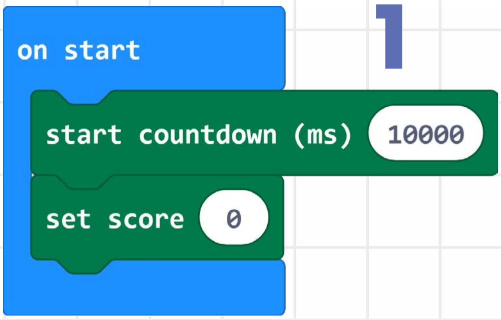
4 Test your code on the micro:bit.

Keep score

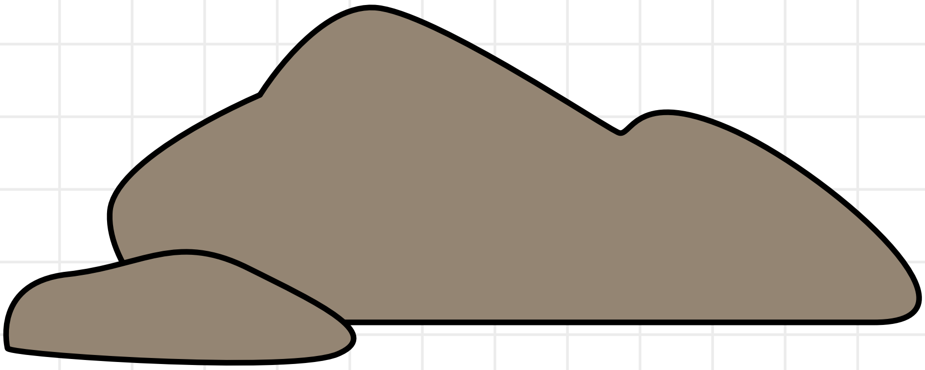
Next, you will turn your project into a game with a timer and a score. The MakeCode editor includes `Game` blocks (in the **`Advanced`** menu) to help you do this, such as a `countdown` block, which starts a timer, and a `score` variable.

1 Add `start countdown` and `set score` blocks to the `start` block. The default countdown is 10000 milliseconds, or 10 seconds. You can increase it later, if you like. The score should be set to 0 with a `set score` block from the `Game` menu.

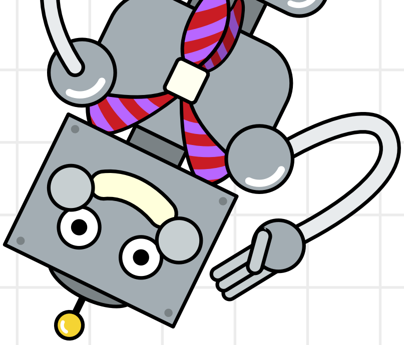
2 Your code will automatically run in the simulator. An animation will play at the beginning of the game, and after 10 seconds, the game will finish and show `GAME OVER` and your score.



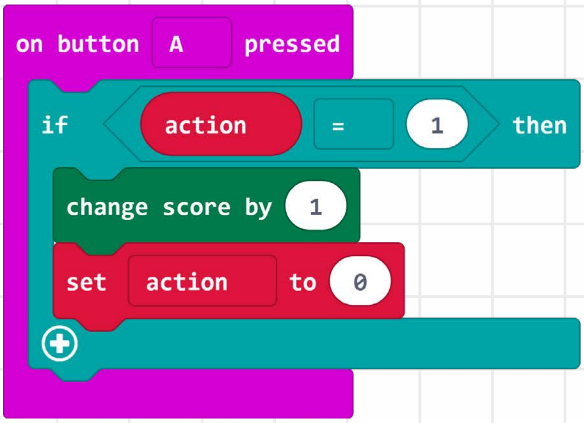
```
on start
    start countdown (ms) 10000
    set score 0
```



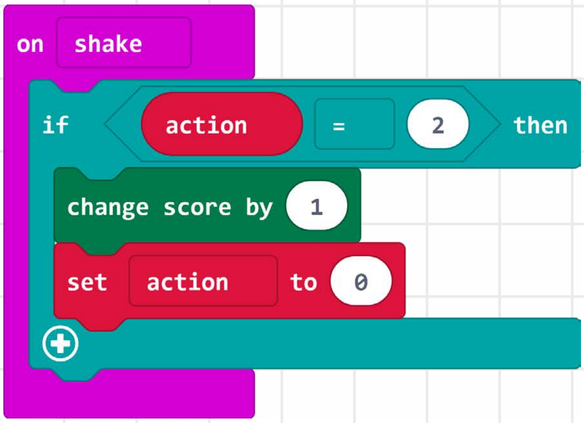
‘Do it, pass it’ micro:bit game



3 At the moment, the score will always be 0. To fix that, add a `change score` block to be executed if the player provides the correct input. You’ll need to do this for the **button A** and **shake** actions.



```
on button A pressed
  if action = 1 then
    change score by 1
    set action to 0
```



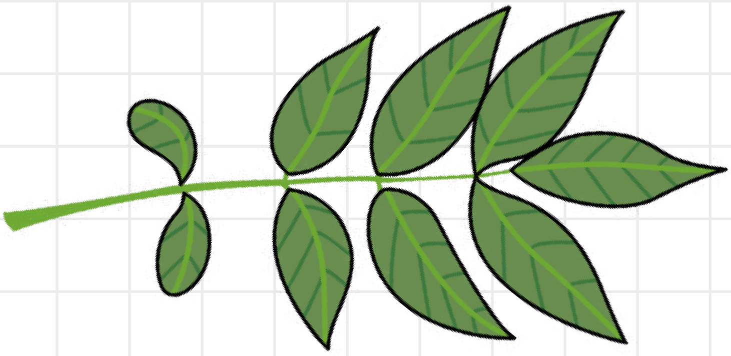
```
on shake
  if action = 2 then
    change score by 1
    set action to 0
```

4 Now, test your game and you will see a score at the end. You will also see a short animation when the score increases.

5 To play the game again, press the reset button on the back of the micro:bit or underneath the virtual micro:bit display.

6 If you like, you can change the timer to give yourself more time. 60000 milliseconds is one minute.

7 Now, play the game with two people. Pass the micro:bit to the other player when you complete an action. You can play in larger groups too.



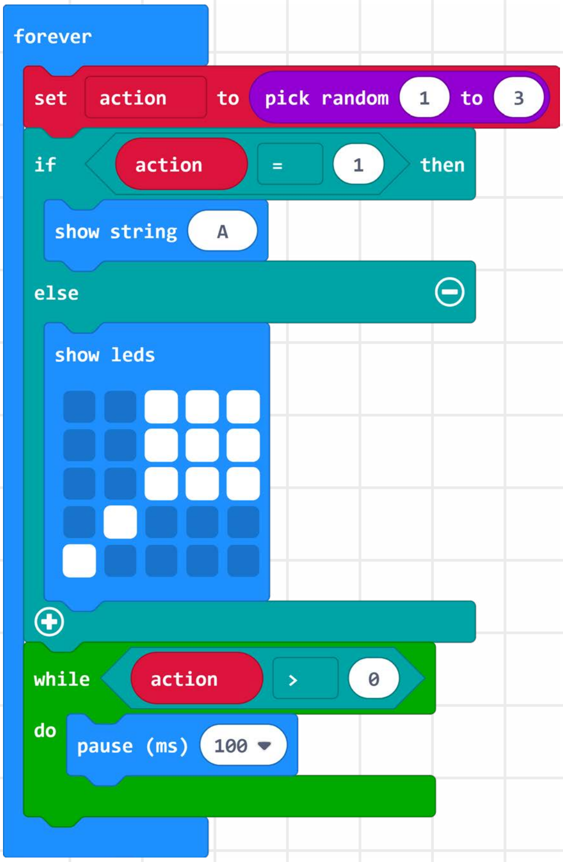
‘Do it, pass it’ micro:bit game

Add more input actions

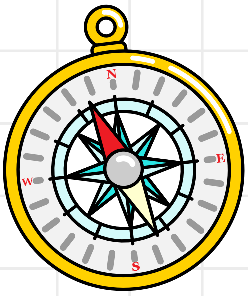
At this point, you have a complete game, but it only has two different actions. In this step, you’ll add another action: connecting and then releasing or ‘pressing’ the **0** pin. The player will provide this input by connecting a crocodile clip lead from the **GND** pin to the **0** pin.

Once you’ve done that, you can try adding your own actions.

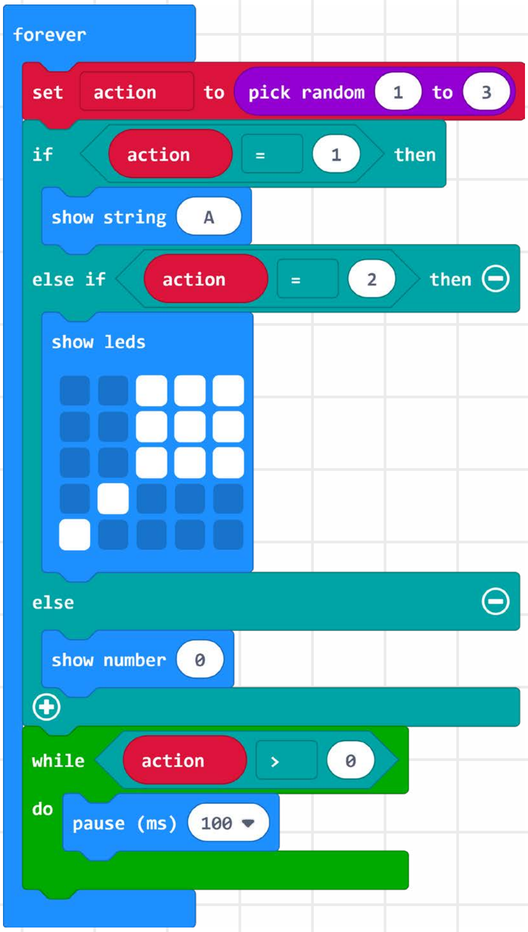
1 Change your `pick random` block so that it chooses a number between 1 and 3.



```
forever
    set action to pick random 1 to 3
    if action = 1 then
        show string A
    else
        show leds
            . . # # #
            . . # # #
            . . # # #
            . # . . .
            # . . . .
    while action > 0
    do
        pause (ms) 100
```



2 Click the plus sign button at the bottom of your action output code in your `forever` loop to add an `else if` section. Change your code so that it displays the maraca if the action is 2 by moving the `show leds` block up, and add a `show number` block to the `else` section below to display a 0.



```
forever
    set action to pick random 1 to 3
    if action = 1 then
        show string A
    else if action = 2 then
        show leds
            . . # # #
            . . # # #
            . . # # #
            . # . . .
            # . . . .
    else
        show number 0
    while action > 0
    do
        pause (ms) 100
```

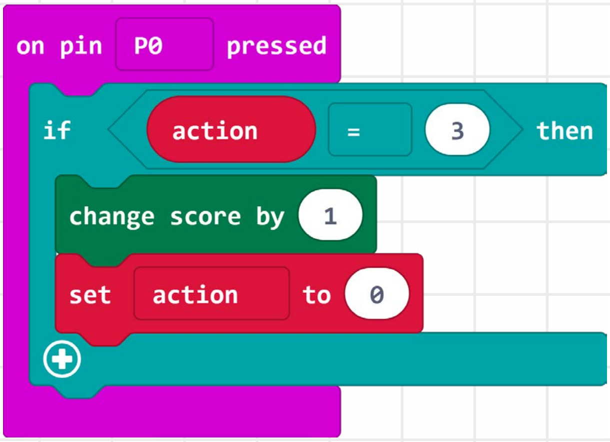
‘Do it, pass it’ micro:bit game

3 Now, add blocks of code to detect that the **0** pin has been connected and then released (pressed).

4 To test your code in the simulator, click on the **0** pin when a `0` is shown on the display.

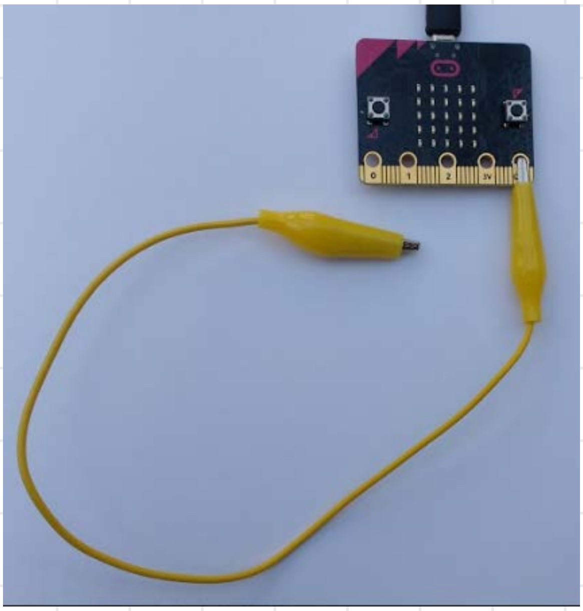
5 Download your code to your micro:bit.

6 Connect one end of a crocodile clip lead to **GND**.



```
on pin P0 pressed
    if action = 3 then
        change score by 1
        set action to 0
```

7 Test your game. When a 0 appears on the display, tap the other end of the crocodile clip lead on the **0** pin. If you don’t have a crocodile clip lead, you can hold the **GND** pin with one hand and then touch and release the **0** pin with the other hand. This works because your skin is conductive (if it doesn’t work then you might need to drink more!).



8 Now, try to add an action of your own. Each of the `input` blocks you have used has a drop-down menu with other options. For each action, you’ll need to:

- Increase the maximum in the `pick random` block by 1.
- Add another `else if` section to display an image for your action.
- Add an `input` block and code to detect when the action has been performed correctly.

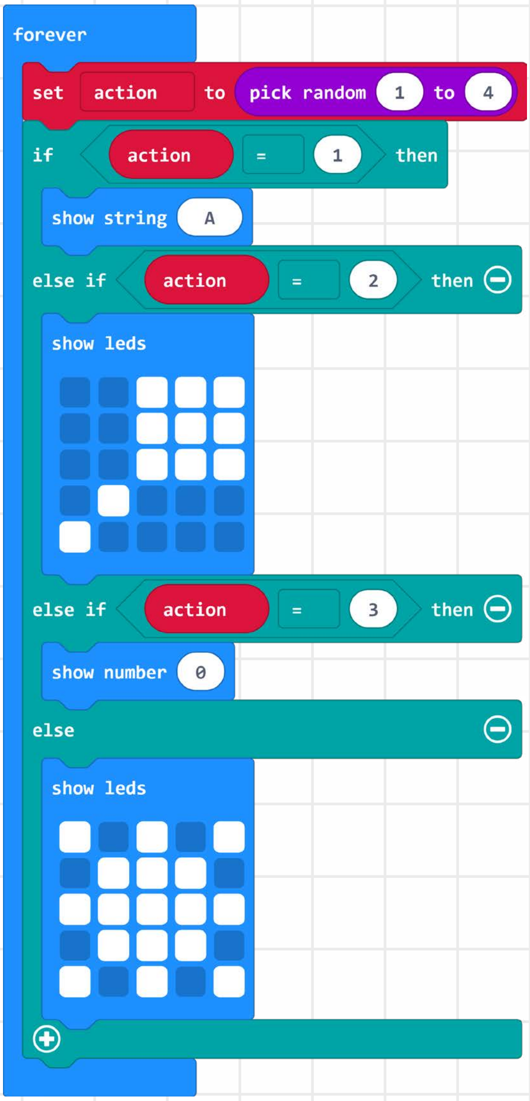
Lights

You can use readings from micro:bit sensors as inputs. In this step, you will add an action to detect when a bright light is shone on the micro:bit. You’ll need a torch (or a mobile phone with a torch option). The `light level` variable returns a number from 0 (dark) to 255 (bright).

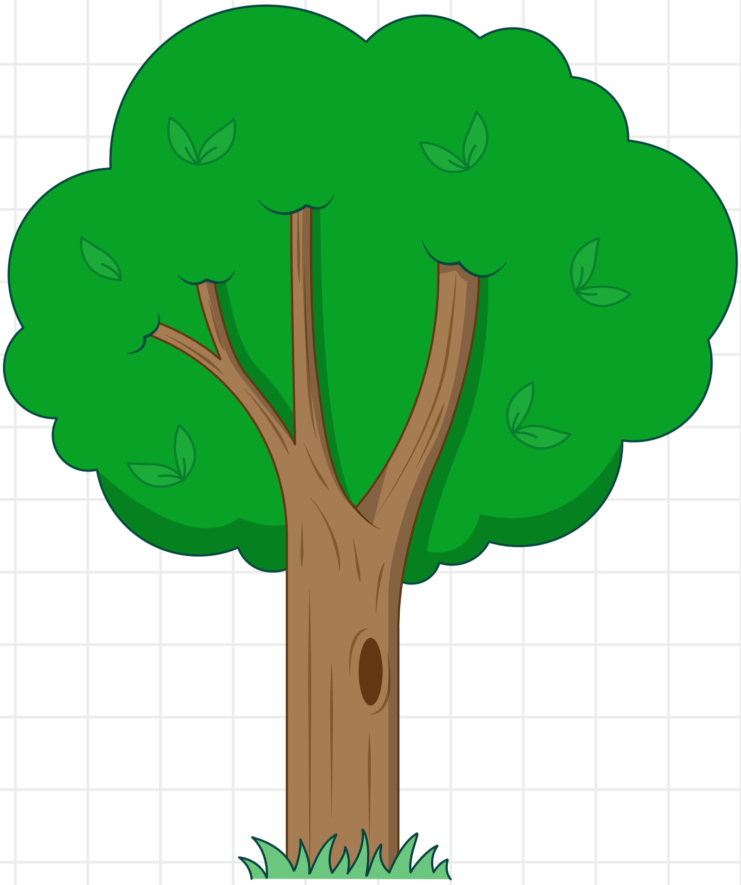
1 Increase the maximum in your `pick random` block by 1 to increase the number of possible actions. This number will depend on how many actions you have added.

2 Click the plus sign button to add an `else if` section to your action output code.

3 Add a `show leds` block and click on the squares to make an image of a light or torch (or something else, if you like).

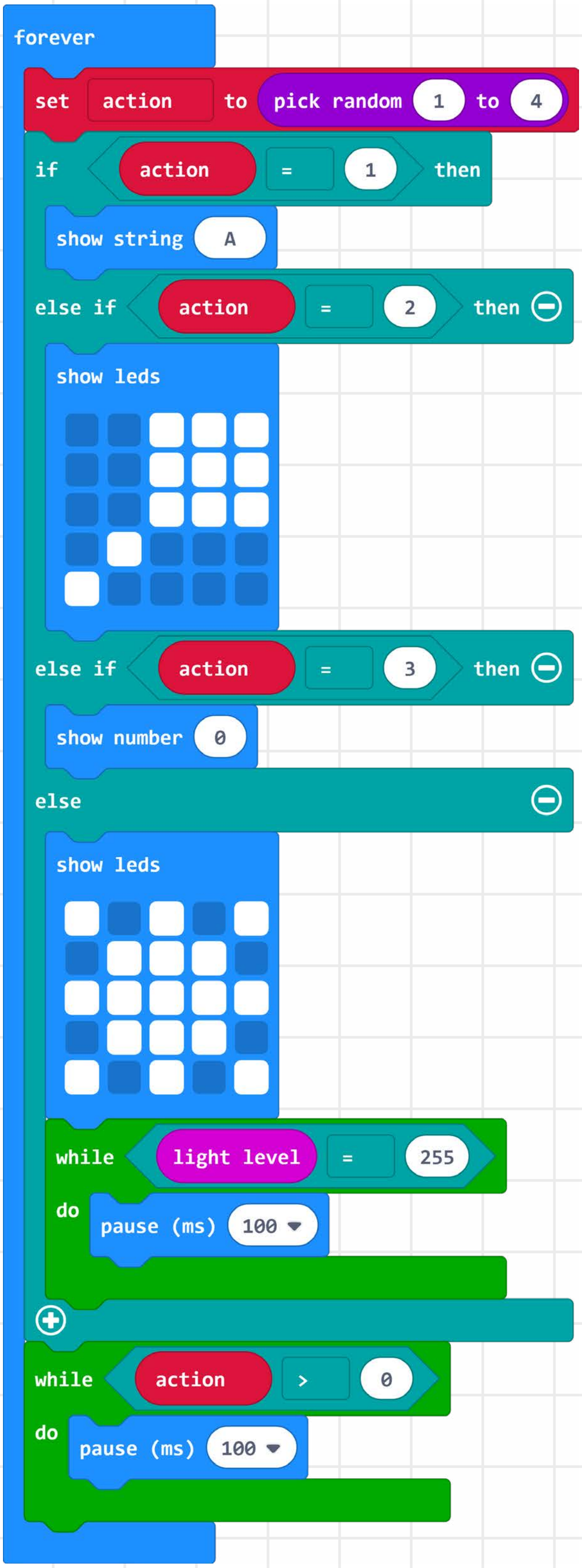


```
forever
    set action to pick random 1 to 4
    if action = 1 then
        show string A
    else if action = 2 then
        show leds
            . . # # #
            . . # # #
            . . # # #
            . # . . .
            # . . . .
    else if action = 3 then
        show number 0
    else
        show leds
            # . # . #
            . # # # .
            # # # # #
            . # # # .
            # . # . #
```

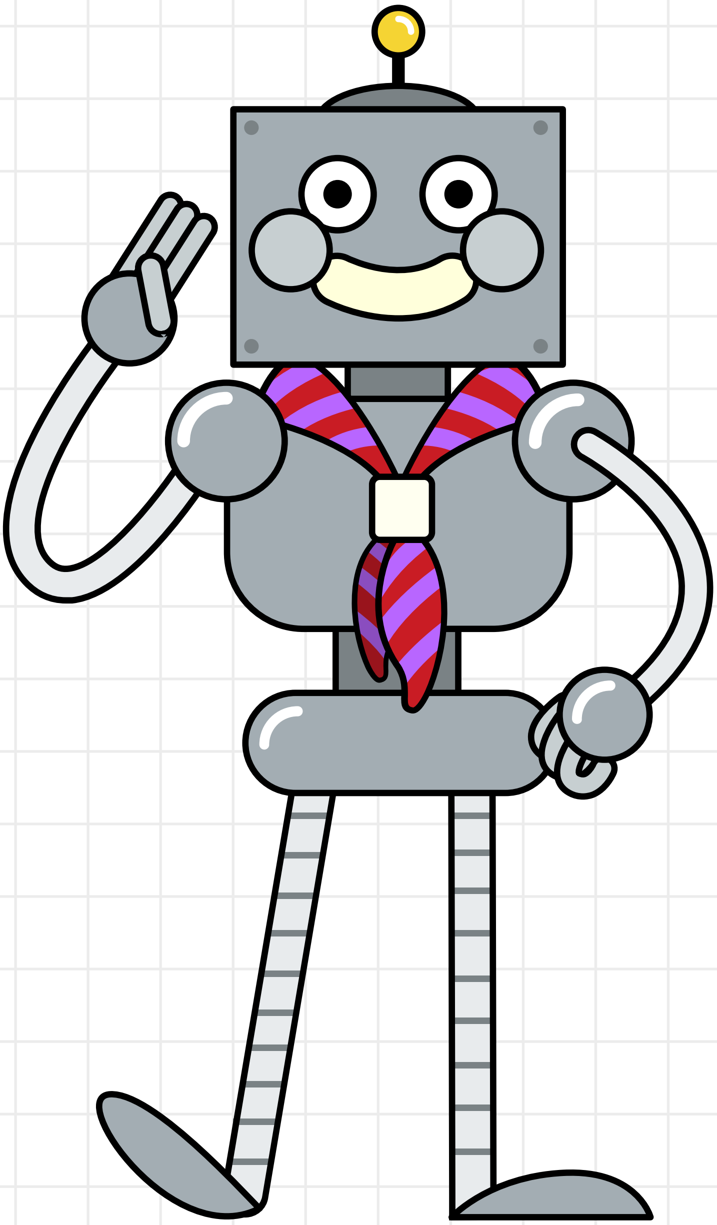


‘Do it, pass it’ micro:bit game

4 When you first use the light sensor on the micro:bit, it returns 255, so you’ll need to wait until the number drops. (This also means that you can’t just shine a light on the micro:bit all the time and cheat!) Add a **while** block to pause for 100 milliseconds when the sensor returns 255.



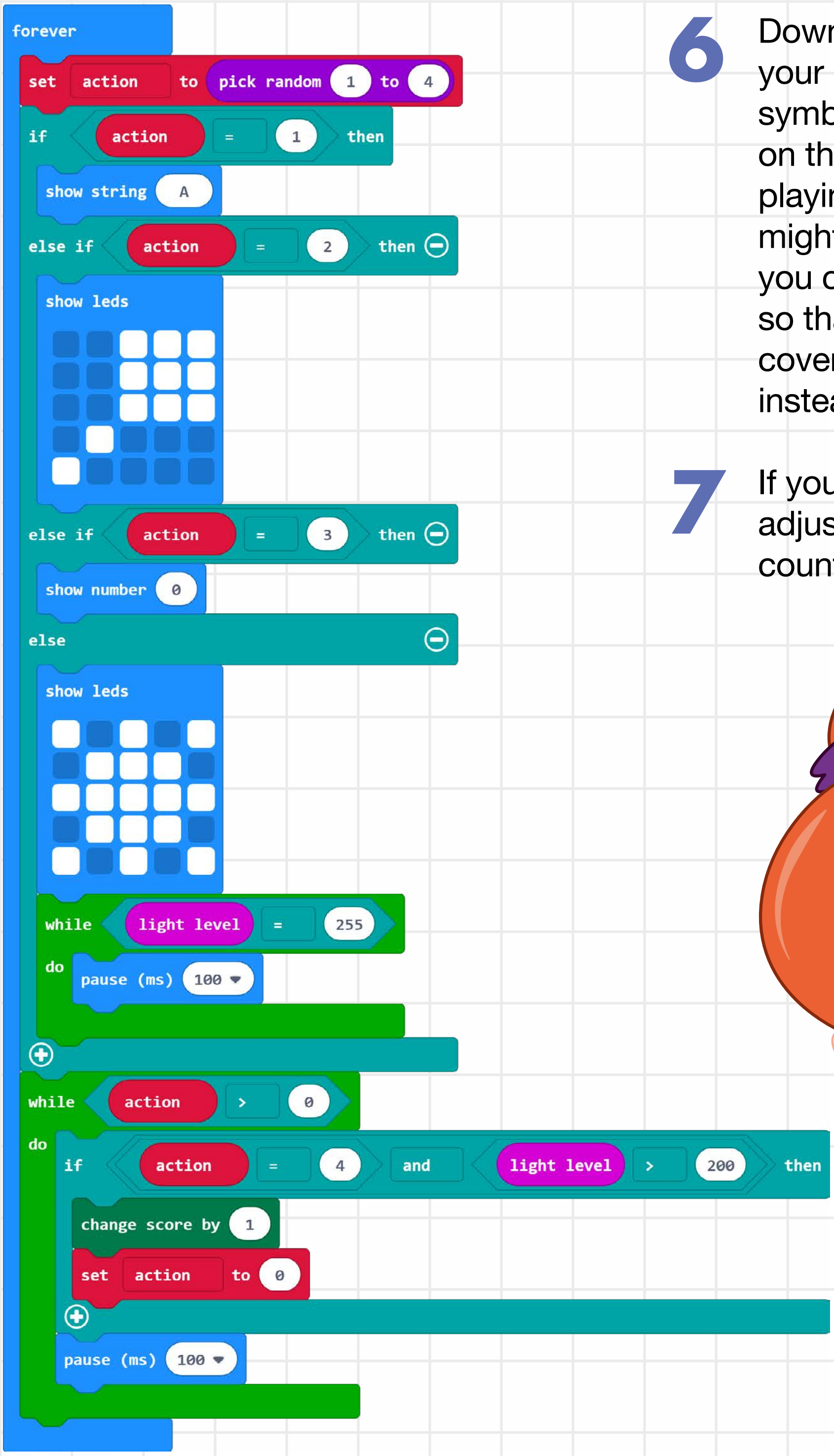
```
forever
    set action to pick random 1 to 4
    if action = 1 then
        show string A
    else if action = 2 then
        show leds
            . . # # #
            . . # # #
            . . # # #
            . # . . .
            # . . . .
    else if action = 3 then
        show number 0
    else
        show leds
            # . # . #
            . # # # .
            # # # # #
            . # # # .
            # . # . #
        while light level = 255
        do
            pause (ms) 100
    while action > 0
    do
        pause (ms) 100
```



Scouts Raspberry Pi

'Do it, pass it' micro:bit game

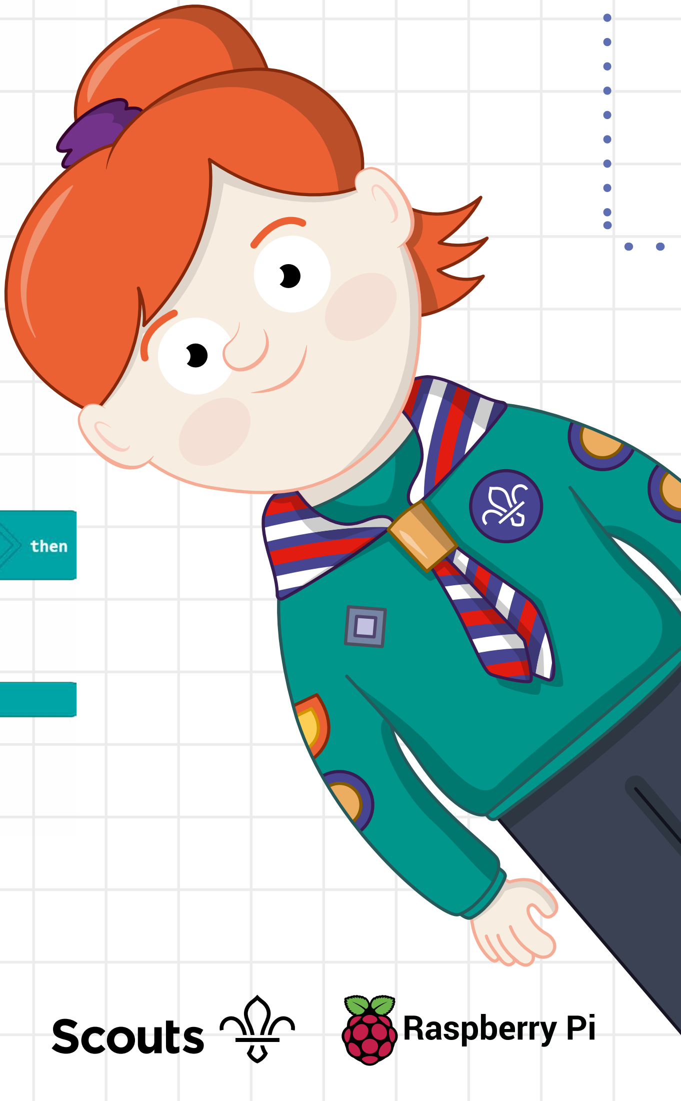
5 Edit the `while` block at the bottom of the code: add in blocks of code to detect that the `light level` is high when a player shines a torch on the sensor. Your code should look like this:



```
forever
  set action to pick random 1 to 4
  if action = 1 then
    show string A
  else if action = 2 then ⊖
    show leds
      . . # # #
      . . # # #
      . . # # #
      . # . . .
      # . . . .
  else if action = 3 then ⊖
    show number 0
  else ⊖
    show leds
      # . # . #
      . # # # .
      # # # # #
      . # # # .
      # . # . #
    while light level = 255
    do
      pause (ms) 100 ▾
  ⊕
  while action > 0
  do
    if action = 4 and light level > 200 then
      change score by 1
      set action to 0
    ⊕
    pause (ms) 100 ▾
```

6 Download your code and test your game. When the light symbol appears, shine a torch on the micro:bit. If you are playing in bright sunlight, you might want to adapt the game: you could change the code so that it detects darkness (e.g. covering the micro:bit display) instead of bright light.

7 If you need longer to play, adjust the amount of time in the countdown block.



What next?

If you like, you can add more input actions and more outputs to the game. You’ll need to design your own images and remember what they mean.

Ideas to try:

- Experiment with different gestures, such as **tilt** and **logo up**.
- Add actions for tapping different pins with the crocodile clip lead (you can leave the other end connected to **GND**).
- Use the magnetometer sensor to detect when a magnet or magnetic object comes near the micro:bit.
- Use the compass.
- Use the temperature sensor — could you move the micro:bit closer to the campfire? Be careful!
- Add different outputs — you could add a green LED to light up when you complete an action, or add a buzzer to produce sound effects.