Lesson Plan

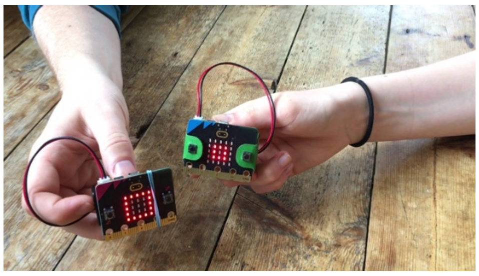
Lesson 6: Rock, paper, scissors

Ages: 7 – 11

Programming language: MakeCode blocks

Topics: Selection/conditionals (Programming), Input/output, Sensors (Computer systems), Randomisation (Programming)

Outcomes: Students build on their prior use of input sensors, variables, and logic to make a computer simulation of a popular game of chance: rock, paper, scissors.



Key learning in this lesson

- Use the accelerometer via the 'on shake' block to start the code running.
- Code the creation of random numbers in a fixed range.
- Use variables so they can be tested using logic.
- Make use of more complex logical 'if… then… else if…' conditional instructions.
- Apply these concepts to make a computer simulation of a real-world game.
- Evaluate the fairness of computer simulations.

Learning objectives

- I can code a micro:bit Rock, paper, scissors game using inputs, random numbers, variables, and logic.
- I can explain how combining inputs, random numbers, variables, and logic can make a computer simulation of a real-world game.

Preparation: before the lesson

What you need

- BBC micro:bits and micro-USB cables – at least one for every two students
- A computer (laptop or desktop) with internet access to the Microsoft MakeCode editor https://makecode.microbit.org/ - at least one for every two students
    - Alternatively, you can use iPads with the micro:bit app installed. See our guide: https://mbit.io/lessons-mobile
- micro:bit battery packs (optional) – one per micro:bit
- PowerPoint presentation – whole class teaching slides
- Code blocks student handout (optional)

The lesson download also includes an optional '.hex' program file of the completed project, which may be useful if you have limited internet access. You can drag and drop this direct onto the MICROBIT drive when you connect a micro:bit to your computer. You can also drag '.hex' files into the MakeCode editor to examine the code and test it in the simulator.

Differentiation ideas

- If this is one of your first coding lessons with the micro:bit, it may be hard to know which students will need more support. You can use the extension ideas in the teaching section below for students who complete the task more quickly than others.

Decide how to deliver the 'Create' coding activity

You'll share the completed code on screen with your whole class from the slides. Additional options include:

- You (or selected students) model building and testing the code yourself on a large screen. The completed code is in the lesson plan and slide deck.
- Give students printed code blocks handouts to follow or cut out and assemble.
- Share a step-by-step YouTube coding video with the whole class, or individual students.
- If YouTube is blocked in your school, we also provide an animation in the slides showing how to assemble the code.

- Students can individually follow an online step-by-step tutorial.
- You can also choose to manage the whole class coding activity and save every student's code using **micro:bit classroom**. Find out more at https://mbit.io/lessons-classroom

Decide how to deliver the 'Evaluate' activity

Students download their code to real micro:bits and test the project.
You may want your students to answer the evaluation questions:

- on paper
- verbally with partners
- as part of a whole-class discussion.

Glossary

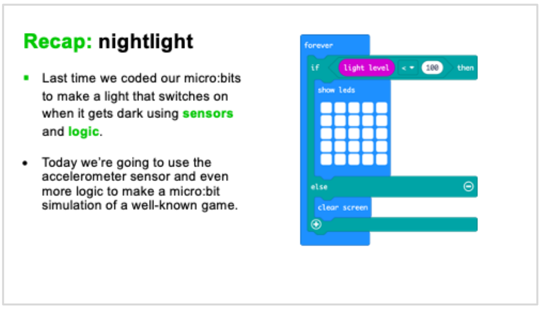
accelerometer:	a sensor that detects movement
conditionals:	see 'selection'
input:	data sent to a computer for processing such as button presses and sensor readings
LED:	light emitting diode - the micro:bit display is made of 25 LEDs
logic:	how computers make decisions based on whether things are true or false
output:	data sent from a computer such as words shown on the display
selection:	making different things happen based on different conditions
sensor:	an input that senses things in the real world, such as movement, temperature, and light levels
simulation:	a computer model of something in the real world

Teaching: during the lesson

Recap prior learning (slide 2)

Ask your students what they discovered last time, for example:

- Last time we coded our micro:bits to make a light that switches on when it gets dark using **sensors** and **logic**.
- Today we're going to use the accelerometer sensor and even more logic to make a micro:bit simulation of a well-known game. A simulation is a computer version, or model, of something in the real world. The MakeCode editor has a simulator of a real micro:bit, for example.


Recap: nightlight

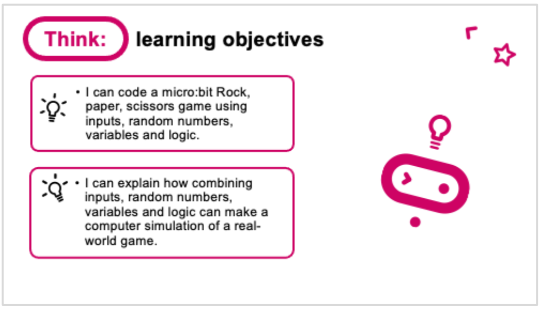
- Last time we coded our micro:bits to make a light that switches on when it gets dark using **sensors** and **logic**.
- Today we're going to use the accelerometer sensor and even more logic to make a micro:bit simulation of a well-known game.

```
forever
  if light level < 100 then
    show leds
  else
    clear screen
```

Think: starter activity

Learning objective (slide 3)

- I can code a micro:bit Rock, paper, scissors game using inputs, random numbers, variables, and logic.
- I can explain how combining inputs, random numbers, variables, and logic can make a computer simulation of a real-world game.


Think: learning objectives

- I can code a micro:bit Rock, paper, scissors game using inputs, random numbers, variables and logic.
- I can explain how combining inputs, random numbers, variables and logic can make a computer simulation of a real-world game.

This final lesson brings together many concepts from the last five lessons. You could ask any of these questions:

- What is an input? (Data going into a computer.)
- What input would be useful to use in this game? (The accelerometer is a good input to use because it can sense when we shake the micro:bit which fits the game.)
- What are random numbers? (Talk about examples like dice. Is a dice more random than a human playing rock, paper, scissors? Do your students think a computer will be truly random?)

- What are variables? (We used variables in the step counter project to keep track of our steps. Today we're going to use them so we can test a random number and show different pictures depending on its value.)
- What is logic? (We used logic in the nightlight project to turn the light on if it was dark. Here we'll use logic in the form of if-then-else blocks to make different pictures appear to represent rock, paper, or scissors depending on the random number generated by the micro:bit.)
- What is a simulation? (A computer version, or model, of something in the real world. The MakeCode editor has a simulator of a real micro:bit, for example.)

Rock, paper, scissors introduction video (slide 4)

Optionally play the project introduction video, which explains how to play the game and which icons will be used: https://mbit.io/lessons-rock-intro-video

Think: rock, paper, scissors introduction video

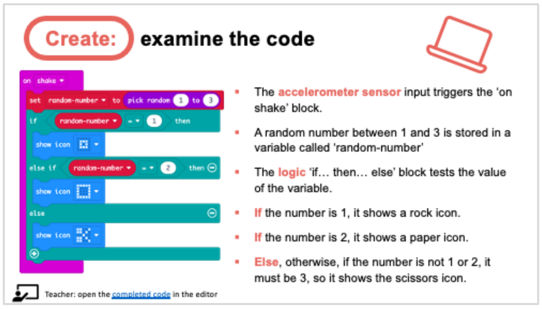
Optionally play video: https://youtu.be/8wFIZLLBbuk

Create: coding activity

Examine code with students (slide 5)

Explain:

- The **accelerometer sensor** input triggers the 'on shake' block.
- A random number between 1 and 3 is stored in a variable called 'random-number'
- The **logic** 'if… then… else' block tests the value of the variable.
- **If** the number is 1, it shows a rock icon.
- If the number is 2, it shows a paper icon.
- **Else**, otherwise, if the number is not 1 or 2, it must be 3, so it shows the scissors icon.



Create: examine the code

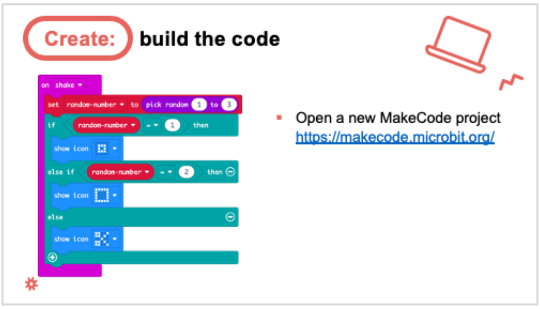
```
on shake
  set random-number to pick random 1 to 3
  if random-number = 1 then
    show icon
  else if random-number = 2 then
    show icon
  else
    show icon
```

- The accelerometer sensor input triggers the 'on shake' block.
- A random number between 1 and 3 is stored in a variable called 'random-number'
- The logic 'if… then… else' block tests the value of the variable.
- If the number is 1, it shows a rock icon.
- If the number is 2, it shows a paper icon.
- Else, otherwise, if the number is not 1 or 2, it must be 3, so it shows the scissors icon.

Teacher: open the completed code in the editor

You can also follow the link in slide 5 to open the completed code in the editor to model testing it in the simulator: https://mbit.io/lessons-rock-code

Model building the code (slides 6-9)


Create: build the code

```
on shake
  set random-number to pick random 1 to 3
  if random-number = 1 then
    show icon
  else if random-number = 2 then
    show icon
  else
    show icon
```

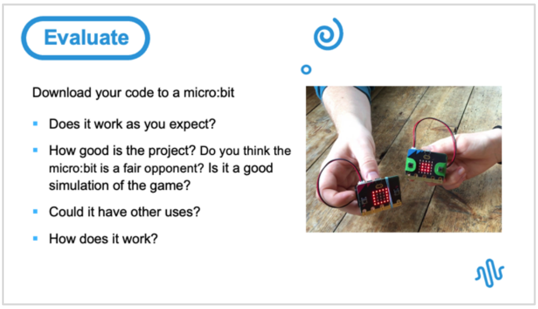
- Open a new MakeCode project https://makecode.microbit.org/

- You can open a new MakeCode project from slide 6 and model building the code.
- Optionally share the YouTube coding video on slide 7 with your class: https://mbit.io/lessons-rock-code-video
- Or share the coding animation on slide 8 if YouTube is blocked in your school.

Students recreate the code, testing it in the simulator. They can either:

- Copy the code from slide 5.
- Follow printed code blocks handouts.
- Individually follow a step-by-step online tutorial: https://mbit.io/tutorial-rps - you can share the link from slide 9.
- If you are using [micro:bit classroom](#), start a new session, add your own starter code and ask your students to join your lesson.

Evaluate: (slide 10)


Evaluate

Download your code to a micro:bit

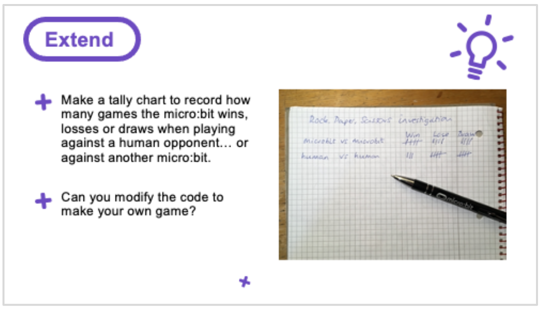
- Does it work as you expect?
- How good is the project? Do you think the micro:bit is a fair opponent? Is it a good simulation of the game?
- Could it have other uses?
- How does it work?

Questions:

- Does it work as you expect?
  - If not, do you need to debug the code and download it again?
- How good is the project?
  - Do you think the micro:bit is a fair opponent?
  - Is it a good simulation of the game?
- Could it have other uses?
- How does it work?

- Encourage students to think about how it works when holding it in their hands.

Extend: (optional, slide 11)


Extend

Make a tally chart to record how many games the micro:bit wins, losses or draws when playing against a human opponent… or against another micro:bit.

Can you modify the code to make your own game?

- Investigate how random the computer (micro:bit) is compared with humans. Make a tally chart to record how many games the micro:bit wins, losses or draws when playing against a human opponent… or against another micro:bit.
- Can you modify the code to make your own game? (Students might design their own icons and make new rules about which objects beat other objects.)

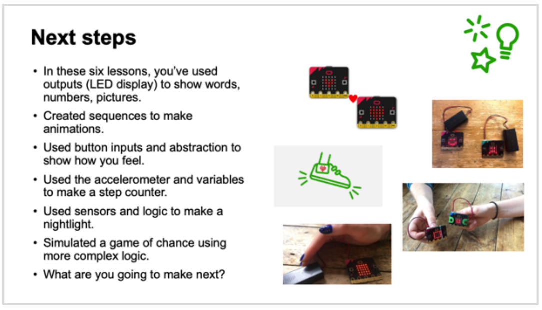
Share: revisit learning objectives (slide 12)

- I can code a micro:bit Rock, paper, scissors game using inputs, random numbers, variables, and logic.
- I can explain how combining inputs, random numbers, variables, and logic can make a computer simulation of a real-world game.

Ask:

- How random were the numbers? How fair is the micro:bit game compared with a human opponent? (Students may have found that the micro:bit is fairer than a human because it can't cheat. The normal game is only fair if both players show their hands at exactly the same time!).
- Why did we use a variable? (So, we could set its value to a random number and test it to see which picture to show. If we didn't use a variable, we would not be able to do this.)
- Why did we not test if the variable value is 3? (Using logical thinking, we made the code more compact. We know the variable has a value of 1, 2 or 3. If the variable is not 1 or 2, it must be 3.)

Next steps (slide 13)


Next steps

- In these six lessons, you've used outputs (LED display) to show words, numbers, pictures.
- Created sequences to make animations.
- Used button inputs and abstraction to show how you feel.
- Used the accelerometer and variables to make a step counter.
- Used sensors and logic to make a nightlight.
- Simulated a game of chance using more complex logic.
- What are you going to make next?

An opportunity to look back at the learning in the last six lessons and think about what you might want to do with your micro:bits next.

- In these six lessons, you've used outputs (LED display) to show words, numbers, pictures.
- Created sequences to make animations.
- Used button inputs and abstraction to show how you feel.
- Used the accelerometer and variables to make a step counter.
- Used sensors and logic to make a nightlight.
- Simulated a game of chance using more complex logic.
- What are you going to make next?

Assessment: after the lesson

When assessing students' work you may find it helpful to ask these questions:

How well did the student follow instructions to code and test a micro:bit rock, paper, scissors game?

What is their understanding of how random numbers, variables and logic combine to make a simulation of a real-world game?

Here are some guiding criteria you might want to include when assessing your students' work:

WORKING TOWARDS the learning objective

The student, with assistance or with code provided, tested the game using the simulator in the editor.

The student may know what individual concepts like random numbers are but not explain how they work together to make a model or simulation of a game.

MEETING the learning objective

The student independently coded a working model of a rock, paper, scissors game and transferred it to a real micro:bit to test.

The student can describe what random numbers, variables and logic are and how they are combined to simulate a real-world game.

EXCEEDING the learning objective

The student independently coded a working model of a rock, paper, scissors game and transferred it to a real micro:bit. They may also have modified the code to make their own game with new images and rules.

The student can describe what random numbers, variables and logic are and how they are combined to simulate a real-world game. They also can describe how they tested and made assessments of the computer simulation’s fairness or randomness by comparing it with human players.