LESSON 23

Microbit Pet

60 minutes | Microbit Masterclass

WHAT THIS LESSON DOES

Follow this guide to transform your Microbit into a virtual pet. You'll program it to show emotions like happiness, sadness, and boredom, respond to touch and movement, and even ask for food or playtime.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Develop skills in creating and using functions in Microbit programming.
- Understand and apply the use of different sensors on the Microbit device.
- Gain knowledge in programming interactive responses using sound and visual cues.
- Learn to use random time intervals in programming for unpredictable outcomes.
- Enhance problem-solving skills by debugging and testing code in a simulator and on a physical device.

BEFORE THE LESSON

- Open and test makecode.microbit.org on your own device beforehand and confirm the school filter does not block it.
- One device per pupil or pair, logged in and ready to start a new project.
- Sound on and headphones available, since the pet uses sound expressions throughout.
- If using physical Microbits, have them charged and USB cables ready for flashing code.

Microbit Pet

HOW THE LESSON RUNS

Introduction
Create a new Microbit project
Create the 'happy' function
Make it sad
Make it giggle
Create the 'feedme' function
Feed the pet
Create the 'play' function
Make it sleep
Play with the pet
Send the code

TEACHING MOVES

- 1. Introduction. Walk through the five behaviours the pet will have before any coding starts. Ask pupils which real pet actions each one mimics so the plan feels concrete.
- 3. Create the 'happy' function. Stress why we make a function: we reuse 'happy' many times, so writing it once saves repeating blocks. Show it called inside on start.
- 4. Make it sad. Demonstrate shaking in the simulator to trigger the gesture. Point out this block sits separate from on start and runs only on the shake.
- 5. Make it giggle. Show the logo sensor on screen and explain touching it counts as a tickle. Note the custom LED face is drawn dot by dot.
- 6. Create the 'feedme' function. Explain the milliseconds input and the pick random block. Convert 10000 to 30000 to seconds aloud so pupils see it is a 10 to 30 second wait.
- 7. Feed the pet. Match the east arrow to the B button on the right. Pressing B calls happy again, so pupils see one function reused.
- 8. Create the 'play' function. Point out this function is written now but called later. Building it in advance is normal and does nothing on its own yet.
- 9. Make it sleep. Show flipping the Microbit face down in the simulator. Note the pause makes a 10 second sleep, after which play is called with a random time.
- 10. Play with the pet. Match the west arrow to the A button on the left. Have pupils predict what pressing A does before you run it.
- 11. Send the code. Test every behaviour in the simulator first: shake, tickle, flip, A and B. Only flash to a physical Microbit once each response works.

WHAT TO WATCH FOR

- Pupils think a function they have created runs by itself once written. Show that 'play' and 'feedme' do nothing until they are called. Point to where each is called and remove the call to prove the pet stops responding.
- Pupils put a response block inside on start instead of inside its own event block, so it only fires once. Explain that gesture and button blocks are separate 'listeners'. Check each pupil's shake, touch and flip code sits in its own event block, not in on start.
- Pupils expect the random hungry or bored prompt immediately and think it is broken during the wait. Remind them the pause can be up to 30 seconds. Suggest testing with a shorter number first, then restoring the random range.

DIFFERENTIATION

- Emerging: Give them the on start and 'happy' function fully modelled on the board and let them copy it before adding the next behaviour.
- Developing: Have them build one new response at a time and test it in the simulator before moving on, so a bug is easy to locate.