technology

Coding sequences and debugging in Scratch

Lesson at a glance

The class builds Banana Jump in Scratch, step by step, on their own devices: a cat that runs on the spot at the bottom of the stage, bananas that appear on the right and slide across, a jump on the space key, and a game that ends if a banana touches the cat. The learning is sequence and debugging, so the order the blocks go in, and what pupils do when the program does not behave as expected, matter more than finishing the game. Model each piece on the board first, let them build it and run it, and close by talking about the bugs they found.

Learning objectives

- Build a Scratch program by putting the blocks in the right order so each sprite does what you want
- Find and fix a bug in a program and describe what was wrong

Before the bell — prep

One tablet or laptop with Scratch per pupil or pair, and the board linked to your device. Build the game yourself once beforehand. It takes about fifteen minutes and it is the only reliable way to recognise their bugs quickly. Decide before the lesson where pupils will save, so the work survives to the next session.

Materials

Item	Qty	Per	Source	Low-cost substitute
tablet or laptop with Scratch	1	pair	school devices	run the lesson on the IWB with pupils taking turns

Safety watch-point

Normal care with devices, and supervise packing away at the end. Nothing else on this lesson, the work is all on screen.

Lesson 26

Teaching moves

- **What we're building:** Play your finished Banana Jump once on the board, then close it. Ask what the cat does, what the bananas do, and what ends the game. Say plainly that almost no program works first time and that finding the fault is the job, not a sign of failure. Keep the block palette closed until the modelling starts.
- **Animating the Cat:** Model this stack on the board and run it before they copy it. The cat should sit at the bottom left and look like it is running on the spot. Draw out that the forever block is what keeps the costume changing, and the wait is what makes the run readable instead of a blur. A cat in the wrong place is nearly always the go to x y block missing, or placed after the forever instead of before it.
- **Adding Bananas:** Warn the class before they run this one: the banana will vanish, and that is correct. The hide block is deliberate, and the clones being made are invisible until the next step shows them. Without that warning, half the room reports a bug that is not there and starts undoing working code.
- **Making Bananas Move:** This is the step where order matters most. The clone code has to start at the right edge, then show, then repeat the small move, then delete itself. Ask a pair to read their stack aloud in order before they run it. If bananas pile up on the right, the delete is missing or sits outside the repeat. If nothing appears at all, the show block is missing.
- **Jumping Cat:** Ask what makes the jump start, and take the answer that it is the space key and not the green flag. That is the difference between one script and another, in plain sight. The two repeats are the way up and the way back down, so a cat that goes up and stays up has lost the second repeat or the wait between them.
- **Detecting Collisions:** Read the forever and the if together as a question the program keeps asking: is the cat touching a banana yet? Run it, lose on purpose, and let them watch the game stop. A game that stops the instant it starts usually has the cat sitting on top of a banana at its start position.
- **Code talk:** Talk only, nothing typed and nothing saved. Take two or three pairs on where the program did something they did not expect, and how they worked out which block was at fault. Revoice the habit you want: change one block, run it again, see what changed. Finish by asking one pupil what they would add if they built it again.

What it should show

Most pupils finish with a cat that runs on the spot and hops on the space key, bananas that appear on the right and slide left, and a game that stops when the two touch. A pupil with a running cat and a working jump but no collision check has still met the objectives. A pupil who found and fixed one real bug has met them well. Unfinished code with a clear note of the bug counts.

Misconceptions & interventions

- **Pupils think the banana disappearing at the hide step means they have broken something, and start deleting blocks that were working.** — Say before they run it that the banana is meant to vanish. Show that the copies are being made all along, and that the show block in the next step is what makes them visible.
- **Pupils treat a clone as the same thing as the original sprite, so they expect the original banana to move across the stage.** — Point at the two separate stacks on the board. One makes copies, the other is the instructions each copy follows once it exists. The original stays hidden for the whole game.
- **Pupils drag the same change y block ten times instead of using the repeat, and believe it is the same thing.** — Put the two side by side on the board. Change the number once inside the repeat and every hop changes. Change it in the stack of ten and there are nine more to edit. That is what a loop is for.
- **When the program misbehaves, pupils delete it all and start again, so they never find out what was wrong.** — Stop the pair and ask them to change one block and run it again. When it works, name that as debugging and have them say which block it was.

Differentiation

Emerging	Developing	Proficient
• Aim for the running cat and the working jump. Sit these pupils where they can see the board stacks, and let them copy one block at a time, running it after each one. • Pair them with a pupil who is one step ahead rather than one who has finished, so the help arrives as narration rather than as a completed screen.	• Build all four scripts. Ask them to point at the block that starts each one and say what it is waiting for, so the green flag and the space key are named as two different starts. • When something misbehaves, have them predict which block is at fault before they touch it.	• Once the game runs, change one number and predict what it will do before testing: the jump height, the banana speed, or the gap between bananas. Keep the change if it makes a better game, and note what it did. • Ask for a second way to end the game, built only from blocks they have already used.

Cross-curricular hook

English oral language. Explaining a bug is a sequencing task in words: what you expected, what happened instead, what you changed, and what happened then. Ask for it in that order at the code talk.