technology

Plan and start the build

Lesson at a glance

The class builds their first MakeCode Arcade game, step by step, on their own devices: a sprite chosen from the gallery, a tile map with walls drawn in the editor, a camera that follows the sprite, projectiles that fire across the map every two seconds, and three lives that go down when one hits you. The hour is a code-along, so the learning is in running it after every block and being able to say what each one did. Model each piece on the board first, then let them build and play it.

Learning objectives

- Create a MakeCode Arcade project and add a character sprite
- Program movement and keep the character inside the play area
- Run the game as it is built and say what each block does

Before the bell — prep

One tablet or laptop per pupil or pair, on arcade.makecode.com, and the board linked to your device. Build the game yourself once beforehand, particularly the tile map and the walls, because drawing those in the editor is the part pupils get stuck on and it is much easier to help once you have done it. If you have Arcade handhelds, have the cables out for the last step; if not, the game runs perfectly well in the browser and nothing is lost.

Materials

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

Safety watch-point

Normal care with devices, and supervise cables and packing away if handhelds are used. Nothing else on this lesson, the work is all on screen.

Lesson 23

Teaching moves

- **What we're building:** Play the finished game on the board for a minute, then close it. Ask what the player controls, what they have to avoid, and what happens when they get hit. Do not open the editor yet.
- **Move your sprite:** Run it as soon as the movement block is in and let them drive the sprite around an empty screen. It is the first moment the thing on screen answers to them, and it is worth thirty seconds of noise.
- **Draw the tile map:** Slow down here and draw yours on the board while they watch. The tile map editor is a different kind of screen from the block editor and pupils who miss the switch sit looking at the wrong panel. Keep the map small, a big map takes the rest of the lesson.
- **Draw the walls:** Walls are drawn on a separate layer from the tiles, which is the single most common place to get stuck: pupils paint a wall on the tile layer and then walk straight through it. Show the layer switch explicitly, then have them test by walking into a wall before moving on.
- **Camera follow sprite:** Ask first why the sprite disappears off the bottom of the screen. Take the answer that the map is bigger than the screen, then add the camera block so the fix lands as an answer to a problem they have already met.
- **Detect overlap:** Read the overlap block aloud as a question the game keeps asking: has a projectile touched my sprite yet? Run it and get hit on purpose so they see the sound and the effect fire.
- **Lose a life:** This is the block that turns it from a demo into a game, so leave time to play it properly. Point out that the lives are set once at the start and taken away inside the overlap, which is why the order of those two pieces matters.
- **Code talk:** Talk only, nothing typed and nothing saved. Take two or three pupils on where the game did something they did not expect and how they found the block responsible. Ask one to say what a single block does in their own words, and finish on what they would add next.

What it should show

Most pupils finish with a sprite they can drive around a walled map, a camera that follows it, projectiles firing across, and three lives that go down on a hit. A pupil with a moving sprite on a map they drew themselves has met the objectives, even without projectiles. Expect the tile map and the wall layer to take longer than the block work.

Misconceptions & interventions

- **Pupils draw walls on the tile layer and cannot understand why the sprite walks through them.** — Show the layer switch on the board and have them redraw one wall on the wall layer, then walk into it to prove it. This costs a minute and saves the rest of the lesson.
- **Pupils think the sprite has vanished when it moves past the edge of the screen, and start deleting movement code.** — Ask where the sprite actually is. It is still on the map, the camera is not following it. That is exactly what the next block fixes.
- **Pupils expect a score, because most games they play have one.** — Say plainly that this game counts lives rather than points. Ask what would have to happen for a score to go up, and keep it as an idea for the code talk rather than building it now.
- **When something misbehaves, pupils start a new project rather than find the block at fault.** — Stop the pupil and ask which block they changed last. Change one thing, run it again, and name that as the way to find a bug.

Differentiation

Emerging	Developing	Proficient
• Aim for a sprite that moves on a small map with walls. Sit these pupils where they can see your screen and let them run the game after every block. • Keep their tile map to a few tiles. A large map is slower to draw and no better to play.	• Build through the projectiles and the lives. Ask them to say what each new block made happen before they move to the next one. • When something misbehaves, have them name the block they think is responsible before changing anything.	• Once it plays, change one number and predict the effect first: how often projectiles fire, how fast they travel, or how many lives you start with. • Ask how they would add a second projectile coming from a different direction, using only blocks already on screen.

Cross-curricular hook

Maths, through the map. The tile map is a grid, the sprite has a position on it, and the projectiles have a speed in two directions. Ask where the sprite is on the grid and what would change if a velocity were negative.