technology

Plan and start the build

Lesson at a glance

Open with five minutes on the Investigation Journal game plan — goal, player and how it moves — then share two or three aloud. Model a blank MakeCode Arcade project on the IWB and walk pairs through sprite, movement, tile-map border with walls, camera-follow, timed projectiles, overlap and lives, running the simulator after each block. Close with a display-only code talk on bugs and one change, then finish the Journal page with what the game does so far and one feature to add next.

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

Book devices with browser access and open arcade.makecode.com on the IWB before the bell. Investigation Journal game-plan pages ready. Optional BrainPad/GameGo handhelds only if already in school — not required. Pair devices so every pair can see the simulator.

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

Manage device leads and posture at screens; optional handhelds stay on desks until download is modelled. No other hazard.

Lesson 23

Teaching moves

- **What we're building:** Hold devices closed for five minutes. Pupils fill only three short lines on the Journal — goal, player, movement — then take two or three aloud so the class hears different games before anyone builds.
- **The Code Editor:** On the IWB, point and name simulator, toolbox and code area. Leave the three words on the board all lesson so pupils can point rather than describe. Ask: Where will we watch our game run?
- **Create a new Arcade project:** Model a fresh blank project — nothing saved to open. Watch for pairs landing in a tutorial and steer them back to blank. Sensible project names only.
- **Add a sprite:** Explain a variable as a labelled box that can hold a sprite. Model dragging the create-sprite block; stop anyone who deletes the whole block instead of editing it. Ask: What is our sprite stored in?
- **Choose a sprite from the gallery:** Model gallery pick and set a short choose-time so browsing does not eat the build. Reassure them they can paint their own later.
- **Move your sprite:** Model the move-sprite block and steer with arrow keys in the simulator. If a sprite sits still, check the block is present and press Run before hunting deeper bugs.
- **Draw the tile map:** Model opening the map editor and drawing only a border, not a filled block. Ask: What is the difference between the map and a wall? Emphasise this step is the map only.
- **Draw the walls:** Model switching to the Draw Walls tool and tracing walls over the border tiles. On the spot, reopen the editor if anyone used the wrong tool. Test by trying to walk the sprite through the edge.
- **Camera follow sprite:** Explain the map is bigger than the screen. Model camera-follow; if the sprite vanishes off-screen, the block is missing or misplaced. Ask: Why did our sprite vanish before we added this?
- **Add some projectiles:** Model the on-update-interval projectile block and let two or three firings run before pairs copy. If they ask why every shot looks the same, tell them random direction and speed come next.
- **Set their direction and speed:** Use the vx/vy direction table on the board. Model the values; if a projectile fires the wrong way or not at all, check vx and vy match what was modelled.
- **Detect overlap:** Model the on-overlap block and run a deliberate hit so the class hears the sound and sees the spray. If nothing happens, check the block is in and the sprite kinds match Player and Projectile.
- **Lose a life:** Model set-lives to three, lose one on hit, and destroy the projectile. Quick test run after adding. If lives stay stuck, the life-change or set-life block is missing.
- **Code talk — what worked and what we'd change:** Talk only — no typing. Invite one pair to name a bug and how they tracked it, then revoice the debugging process for the class. Finish the Journal: what the game does so far and one feature to add next.

What it should show

A working first version: gallery sprite steered with arrows, held inside wall tiles, camera following on the larger map, projectiles crossing every two seconds, overlap playing sound/effect and dropping a life from three. Sprite freezes usually mean the move block is missing; walking through the border means walls were drawn with the map tool, not Draw Walls; no life drop means the overlap or life blocks are absent.

Misconceptions & interventions

- **I drew the border tiles so they must already be walls.** — Reopen the map editor and switch to Draw Walls — tiles and walls are separate tools. Have them trace walls over the border and test walking into the edge.
- **My sprite vanished — the game is broken.** — The map is bigger than the screen. Check the camera-follow block is present and attached to their sprite, then move again in the simulator.
- **One wire of code should be enough — I deleted the sprite block to start clean and now nothing works.** — Show the variable as a labelled box that still needs the create-sprite block inside it. Restore the block as modelled rather than rebuilding from scratch.

Differentiation

Emerging	Developing	Proficient
• Teacher-table pair: gallery sprite and move block only first, with the three editor words to point at; add map border together before walls. • Partly-built starter project on the IWB they can copy block-by-block rather than find each category alone.	• After lives work, predict which side a given vx/vy pair fires from, then test in the simulator. • Journal stretch: name the block that fixed their biggest bug today.	• Critique fairness of always-the-same projectiles and suggest one change to direction or interval using blocks already in the project. • Design one next-feature plan on the Journal that reuses overlap or lives rather than brand-new blocks.

Cross-curricular hook

Link vx/vy to Maths coordinates — positive and negative values set direction and speed on a grid.