technology

Build and debug the core features

Lesson at a glance

Pairs reopen their saved MakeCode Arcade game and choose one core feature: scoring, a repeating loop, or an event reaction. After a short teach on feature, bug and debug, the teacher live-models Path A scoring with a planned miss, then the run–notice–change-one-thing cycle. Pairs build from the three board checklists, run often, and log the bug (or a genuine near-miss) on the Investigation Journal page before a brisk class share of fixes.

Learning objectives

- Build a platform-style Arcade game with left and right movement and gravity
- Find and fix at least one bug while building, and describe what was wrong
- Explain how one core feature of the game is coded

Before the bell — prep

Have tablets/laptops ready with MakeCode Arcade and each pair's Part 1 save open before the bell. Load a clean demo project (movable player sprite) on the IWB plus the three path checklists left fully visible. Spot-check logins; have a neighbour-pair or Part 1 starter ready for lost saves.

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

Shared devices: short turns so both partners drive; sensible volume; log out and return devices to the trolley at pack-away. No high-voltage kit—MakeCode Arcade only.

Lesson 24

Teaching moves

- **Getting Started:** Devices already open on the saved Part 1 project. Pairs name one thing their sprite can already do, then pick only Path A, B or C. If a save is missing, pair them with a neighbour or reopen the starter on the IWB—do not invite free-form features outside the three paths.
- **Bugs, features and fixing:** Keep the board tight: feature, bug, debug, plus the score-stays-at-zero example. Name aloud that professional programmers debug constantly—an unexpected result is data, not proof they are 'bad at coding'.
- **Model the next coding steps:** Seven-minute live model of Path A only: set score to 0, add a coin, plant a wrong-kind overlap bug, run so the score stays at zero. Take two pupil predictions on where the bug hides, fix only the overlap pair, run again, and voice 'I knew it worked because the score went 0 to 1'. Point briefly at Path B and C checklist blocks; leave all three checklists plus 'run often · change one thing' visible.
- **Build and debug on device:** Coach the process, not their code: run early, change one thing, say expected vs actual before fixing. If stuck beyond a couple of minutes, compare the same path with a neighbour then return to their project. Success is a visible feature plus a real bug story or optional genuine near-miss; both partners drive the device.
- **Log the bug you fixed:** Paper Investigation Journal only—not on the device. Circulates checking they separate what they saw from what they think it meant. Accept a real bug fix alone; near-miss only if the build worked first time—never push invented bugs.
- **Share bugs and fixes:** Display-only talk: three or four pairs across different paths. Revoice into expected vs actual, one change, evidence after re-run. Draw out that bugs often sit in the trigger, not the action. Key question: what made a fix trustworthy, not a lucky guess?

What it should show

Path A: on-screen score rises by 1 on one clean coin touch and the coin is destroyed or moved. Path B: second sprite keeps moving with no button press and stays on screen. Path C: every player–hazard touch triggers the same clear reaction. Odd fails usually mean wrong sprite kinds on the overlap, score change left outside the event, or a heavy block freezing a forever loop—re-run after one fix.

Misconceptions & interventions

- **'If there's a bug, I'm bad at coding.'** — Revoice from the teach beat: programmers spend much of their time debugging; an unexpected result is useful data. Ask expected vs actual before any block change.
- **Pupils change several blocks at once, then cannot tell which fix worked.** — Point at the IWB rule 'change one thing · then run again'. Have them undo extras, re-run, and name the single change that made the feature visible.
- **Feature blocks are added but never fire—score stays 0 or the hazard does nothing.** — Ask: 'When should this happen? Which block makes that moment?' Check the overlap kinds match player and coin/hazard, and that change-score or the reaction sits inside the event.

Differentiation

Emerging	Developing	Proficient
• Steer the pair to Path A beside the IWB checklist; aim only for one successful score change, copying the overlap structure from the demo then returning to their own save.	• When the core feature works, add one small extra from the same path (second coin, faster loop, or second hazard) or help a neighbour with debugging questions only.	• Critique a fix for trustworthiness: would one re-run be enough evidence, or should they test a second touch/edge case? Lead a neighbour with questions, not by taking the mouse.

Cross-curricular hook

English oral language: pairs must state expected vs actual and evidence the fix worked—clear cause-and-effect talk.