technology

Finish, test and share

Lesson at a glance

Pupils reopen their saved MakeCode Arcade project and the Design Brief from earlier in this project. They use a timed finish window to close gaps on scoring, win/lose and movement, then run two short play trials marking pass or fail against each plan criterion. Results, one design or code change, and one next improvement go on the Investigation Journal page. Board samples and a peer device carousel share the games; a final partner talk focuses on the build and debug process, not only the product.

Learning objectives

- Build a complete Arcade game with a player, targets and scoring
- Test the finished game and explain one change made during the build
- Reflect on the process of building and debugging, not only the finished game

Before the bell — prep

Before the bell, confirm every pair can open their cloud- or device-saved MakeCode Arcade project and has their Part 1 Design Brief on the desk. Set a visible IWB or classroom countdown for the finish window. Investigation Journal pages stay aside until after the two test trials. Fallback: one IWB Arcade session with rotating short turns while others plan fixes on paper.

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

Confirm cloud or device saves before the peer carousel and before tidy-away so work is not lost when devices are swapped or returned.

Lesson 25

Teaching moves

- **Getting Started:** Have pairs open the saved project and place the Design Brief on the desk — no Journal page yet. Ask: if we only judged by looks, what might we miss? Draw out that a polished screen can still fail the plan (no score, stuck sprite, missing win).
- **What does finished mean?:** Model once aloud: run against the Brief ticks, score rises but nothing at ten points — win condition still missing. Stress finished means the core plan works well enough to share, not bug-free forever. Thirty seconds partner talk on one thing that must work, then take two quick ideas.
- **Finish the game:** Start the shared countdown. Circulate only with prompts — which plan part is still missing? What is the smallest fix? — and do not put a shared block recipe on the board. Watch that pairs run after each change and finish the Brief rather than invent new features.
- **Test against the plan:** Hard stop on new features. One pupil plays ~1 minute; the other marks pass/fail on the Design Brief only, with a few observation words on any fail. Swap roles for trial two. Fold in two pairs: one pass, one fail — is that a plan problem or a code problem?
- **Record the results:** Hand out the Investigation Journal page now. Pupils write whether the plan was met, one change from the build, and one evidence-backed improvement. Offer stems aloud: The plan said... The tests showed... One change we made was... Next we would...
- **Share the game:** Board sample three or four pairs only: 20-second run, met plan yes/partly/not yet, one change — timed. Confirm cloud saves, then peer carousel: swap devices 30–40 seconds each way, no block edits; listeners name one change spotted, spoken only.
- **Talk about the process:** Protect this display-only block: 90s silent think, 90s pair talk, then two minutes for three or four volunteers. Prompts: hardest to get working, and what you would change with one more lesson. Revoice bug, test, change, plan. Partner talk already counts — class share is a sample.

What it should show

A successful finish meets the Design Brief core (movement, scoring, win/lose) well enough for two honest trials — not a perfect game. Expect mixed pass/fail marks; clear fails with observation notes are valid evidence. Common setup issues: score block outside the overlap event, win check never reached, or controller blocks on the wrong buttons. Pairs should name one deliberate design or code change and one next improvement tied to a failed criterion.

Misconceptions & interventions

- **Finished means the game looks good and is bug-free forever.** — Point back to the Design Brief ticks: finished means the planned goals work well enough to test and share, and you can name one change you made — not perfect art or zero bugs.
- **A fail on the test means we failed, so we should hide it or keep polishing instead of recording.** — Stop the polish urge. Say a clear fail is useful evidence: write what you saw (score stayed at zero), not what you hoped, and carry it to the Journal and the share.
- **Sharing is only a demo of the cool finished screen — the bug story does not count.** — Structure the board share as run → met the plan? → one change. Revoice that programmers present what they changed and why; that is the Nature of STEM review habit.

Differentiation

Emerging	Developing	Proficient
• Aim for one working goal only (move + score); strip back to the last version that ran and restate the bug in one sentence before editing. • Offer the spoken stems for Journal sentences and accept short phrases tied to Design Brief marks.	• After two trials, add a second improvement idea linked to a failed criterion, or note one difference watched in a neighbouring pair's test.	• Polish a second rule already on their plan (lives, levels, or a clearer win message) — not a brand-new game idea — and explain in the share whether a fail was a plan problem or a code problem.

Cross-curricular hook

Links to English oral language: structured peer share with a timed claim (met the plan / one change) and listening for a clear design decision.