technology

Build, debug and share a fuller game

Lesson at a glance

Open by naming the designer habit: build a piece, run it, fix what breaks. On the IWB, model a fresh MakeCode Arcade project and walk Stage 4 pairs through Shark Attack step by step — player sprite with joystick move and stay-on-screen, then enemy sharks on a 5-second timed loop that spawn top-left, chase the player, and end the game on overlap. Pairs build and debug beside you. Close with a display-only code talk on one bug they fixed and one change they would add next.

Learning objectives

- Extend a saved game with a loop or an event
- Find and fix bugs and simplify the program by removing unused blocks
- Demo the finished game and explain what was changed

Before the bell — prep

Book devices and test arcade.makecode.com on the class network before the lesson — logins and blocked sites are the usual delay. Open a blank Arcade project on the IWB ready to model. Pair pupils so every pair has one working device.

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 shared keyboards so bags and chairs stay clear of cables; remind pairs to save before closing the browser tab.

Lesson 26

Teaching moves

- **What we're building:** Devices on, Arcade open. Tell pairs the routine: watch the board, build the piece, run it. Ask 'What does a designer do after adding one new piece?' (test it). Reassure pairs who feel behind that every step is modelled together.
- **Create a new Arcade project:** Model arcade.makecode.com → New Project on the IWB. Every pair starts empty — if anyone opens an old file, get them onto a fresh one. Watch for pairs stuck at a login or school-filter screen.
- **Create your player sprite:** Model sprites.create and click the grey image box to pick from the gallery. Ask 'Which character will you control?' Common slip: leaving the grey box blank so nothing appears — point pairs to the gallery button before moving on.
- **Move the sprite:** Add the controller/move block, run it, and let every pair drive their sprite with joystick or arrow keys before the next step. Ask 'What makes it move now that didn't before?'
- **Make it stay on the screen:** Run first so the class sees the sprite slide off the edge — name it as a bug. Then model the stay-in-screen block. Ask 'What changed after we added this?' Watch for the block dropped outside the right stack.
- **Create the enemy sprites:** Model onUpdateInterval(5000) and count aloud so pupils see a shark appear every few seconds. Stop the room and check every pair set SpriteKind to Enemy, not Player — that is the big check of this step.
- **Set the enemy position:** Model setting spawn to the top-left. Ask 'Why is it better sharks don't appear on top of the player?' Re-run any pair still spawning in the centre so they can compare.
- **Make them chase the player sprite:** Add the follow/chase block at speed 10 and run so sharks track the player. Ask 'What does the speed number change?' Invite confident pairs to predict a higher speed before they test it.
- **Game over:** Model the overlap event on player + enemy and stress selecting LOSE, not WIN. Run together and celebrate the first Game Over. Ask 'What event ends our game?'
- **Code talk — what worked and what we'd change:** Talk only — no typing. Surface a real bug a pair hit (wrong sprite kind, or WIN instead of LOSE) and have them say how they found and fixed it. Revoice so the whole class hears the debug steps. Prompt one change they would add next time.

What it should show

A working Shark Attack: player moves and stays on screen; a new Enemy shark spawns about every 5 seconds at top-left, chases the player, and overlap ends the game with Game Over (LOSE). If sharks never appear, the interval or Enemy kind is wrong. If overlap does nothing or shows a win, the event kind or LOSE setting is the usual fault — re-check those two blocks.

Misconceptions & interventions

- **Pupils leave the new shark as SpriteKind.Player, so it never acts as an enemy.** — Freeze the class at the spawn step and have every pair read the kind aloud. Point to Enemy on the IWB and re-run until a chasing shark appears.
- **Pupils pick WIN instead of LOSE on the game-over block, so touching a shark 'wins'.** — Open their overlap event on the board, switch it to LOSE, and re-run so the class sees Game Over. Ask 'Which event should end the chase game?'
- **Pupils forget to click the grey image box, so the sprite code runs but nothing is visible.** — Point to the gallery button on the IWB, have them pick any sprite, and run once so the character appears before adding movement.

Differentiation

Emerging	Developing	Proficient
• Keep the pair at the teacher IWB table and build each block in lock-step with your model; they run after every single add. • Give a simple checklist on paper: player → move → stay on screen → enemy → chase → overlap LOSE.	• After the game works, ask them to change chase speed once and say what they notice. • Prompt: name the bug you fixed and which block cured it.	• Challenge: predict and test what happens if the interval is shorter, then explain the trade-off to another pair. • Fair-code stretch: strip any unused blocks and demo a cleaner version, saying what each remaining block does.

Cross-curricular hook

Tie to Maths measures — the 5000 ms interval and chase speed are quantities pupils can compare and reason about.