technology

Build, debug and share a fuller game

Lesson at a glance

Open by asking what a game designer does after adding one new piece — test it. Pupils reopen their saved Shark Attack project in MakeCode Arcade and, modelling each step on the board first, build up the game: a player sprite that moves and stays on screen, sharks that spawn every 5 seconds in the top-left and chase the player, and a game-over event when they overlap. They run after each block, hunt bugs and tidy unused code, then close with a code talk on what broke and what they'd change.

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

One device per pair with MakeCode Arcade (arcade.makecode.com) open and pupils' saved Shark Attack projects locatable — check logins work before the bell, as login snags eat time. Have a fresh blank project ready as a fallback for any pair who can't find or open their save. Test the IWB link and your own project so you can model each block live.

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

Lesson 26

Teaching moves

- **Create a new Arcade project:** Model opening the project on the IWB. Most pupils are reopening a save, not starting fresh — guide them to their file, and hand any locked-out pair the blank fallback project so nobody stalls at the login screen.
- **Make it stay on the screen:** Run the game BEFORE adding the block so pupils see the sprite slide off the edge — name it as a bug. Then model the stay-on-screen block and re-run. Ask 'What changed after we added this?' Watch for the block placed in the wrong spot.
- **Create the enemy sprites:** Model the timed spawn and count aloud so pupils see a shark appear every five seconds — this is the timed-loop concept live. Stop and check every pair selected Enemy, not Player, as the sprite kind; that's the classic slip here.
- **Make them chase the player sprite:** Model the chase block and run it so pupils watch sharks follow the player. Ask 'What does the speed number change?' Invite confident pairs to predict what a higher speed does before they test it.
- **Game over:** Model the overlap event and stress selecting LOSE in the game-over block — this is the most common bug. Explain the overlap block is a watchman for the player and shark touching. Run it together and celebrate the first 'Game Over'.
- **Code talk — what worked and what we'd change:** Ask a pair who hit a real bug (wrong sprite kind, or WIN instead of LOSE) to explain how they found and fixed it. Revoice their debugging so the whole class hears the process.

What it should show

A working game: the player sprite moves with the joystick and stays on screen, a shark spawns from the top-left every five seconds and chases the player, and touching a shark triggers 'Game Over'. If a pair's game never ends when a shark catches them, they've usually picked WIN instead of LOSE in the game-over block, or left the enemy as Player kind so the overlap never fires.

Misconceptions & interventions

- **Pupils think the new shark sprite works fine even though they left it as the Player kind.** — Open the create-sprite block on the IWB and show the kind is Player — then the overlap event can't tell player from shark. Have them switch it to Enemy and re-run to see the game-over trigger.
- **'The game should end when a shark catches me, but nothing happens.'** — Check the game-over block on their device — if it reads WIN, the catch counts as a win and play continues. Switch it to LOSE, re-run, and watch the correct 'Game Over' appear.
- **Pupils think code is finished once it's typed, without running it.** — Press play together after each block and name the try-it-and-fix-it habit — model spotting the sprite sliding off the edge as a bug you only catch by running the program.

Differentiation

Emerging	Developing	Proficient
• Pair with the teacher device and build one block at a time, running after each so success is visible before the next step. • Provide the saved project already opened so no time is lost finding the file.	• After the chase block, change the speed number and predict then test what happens to the sharks. • Add a second enemy spawn and re-run to see the game get harder.	• Remove any unused blocks and explain why the tidied code still works the same. • In the code talk, critique one bug they hit and describe exactly how they'd stop it happening next time.

Cross-curricular hook

Link to English oral language — the code talk has pupils explain and sequence their debugging steps clearly for the class.