technology

Start and build a MakeCode Arcade game

Lesson at a glance

Open by naming the steps a simple catching game needs — hero, left–right move, stay on screen — so the build has a purpose. On the IWB, model a fresh MakeCode Arcade project, then guide pairs through creating a sprite, painting it in the Sprite Editor, adding arrow-key control, locking it on screen, and trying an effect. Pairs run the simulator after every new block and say what each block does. Close with a display-only code talk on one bug they fixed and one idea for next time.

Learning objectives

- Create a MakeCode Arcade project and add a character sprite
- Program the character to move under player control and keep it on the screen
- Run the game as it is built and say what each piece of code makes the character do

Before the bell — prep

Book one device per pair with a modern browser; pin arcade.makecode.com and open a blank project on the IWB before the bell. Check network access to MakeCode. No handheld (BrainPad/GameGo) needed — the browser simulator is enough. Have a spare device ready for pairs whose tab freezes.

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

Standard classroom device use only — posture, shared-screen turns, and no unapproved sites. No electrical or tool hazard in this browser-only build.

Lesson 25

Teaching moves

- **What we're building:** Hold devices closed. Ask what steps Coin Catch needs before anyone can play it. Gather a spoken plan — hero, left–right move, stay on screen — then open the board demo so the build has a clear purpose.
- **The Code Editor:** Point to simulator, toolbox and code area on the IWB as you name each. Ask which part shows if the code works — lock the answer on the simulator before anyone builds.
- **Create a new Arcade project:** Model New Project live so every pair lands in the editor, not the play page. Let them name freely; circulate and confirm each pair has a fresh empty project open.
- **Create a sprite:** Place the sprites.create block on the board first, then run so the empty sprite appears. Stop anyone who jumps straight to drawing without the block in the code area.
- **Design your character:** Demo a quick simple character in the Sprite Editor and set a soft time limit. Check every pair hits the green Done button so the sprite shows in the simulator before moving on.
- **Control your character:** Add the controller block carefully, run, and move with the arrow keys on the IWB. If nothing moves, the block is missing or snapped wrong — re-attach live and ask what left and right now do.
- **Make them stay on the screen:** Use the sprite sliding off as the reason for this block. Add it, run, and deliberately try to push the sprite off-screen so the class sees it stop at the edge.
- **Add an effect:** Drop the effect block on the board and run so the visual change is the reward for the build so far. Pair anyone still behind with a neighbour and rebuild alongside the IWB.
- **Change the effect:** Invite pairs to swap the effect and re-run each one. Prompt which effect suits their character and why — no wrong answer, just safe try-and-see.
- **Code talk — what worked and what we'd change:** Pupils talk only — no typing. Invite one pair to name a bug and exactly how they fixed it; revoice that debugging is normal. End with one change each pair would add next session.

What it should show

A successful build shows a painted sprite that moves with the arrow keys, cannot leave the screen edge, and carries a visible effect. If the sprite never appears, the create block is missing or Done was not clicked. If it appears but will not move, the controller block is absent or snapped outside the main stack — re-attach and re-run.

Misconceptions & interventions

- **"I can just draw the character — I don't need the sprites.create block first."** — Show the empty code area on the IWB, place the create block, then open the grey box. The editor only stores the picture inside a sprite the program already made.
- **"The sprite vanished off the edge, so the game is broken."** — Replay pushing it off, then add the stay-on-screen block and push again. Name it as a rule games use so the player can always see and control the hero.
- **"If nothing moves, MakeCode itself is broken."** — Trace the stack with a finger: create, then controller, snapped together. Re-add the controller block live and ask them to press left and right so they see the fix.

Differentiation

Emerging	Developing	Proficient
• Sit the pair at the IWB table and have them copy each block immediately after you place it; confirm the simulator runs before the next step. • Offer a pre-started project that already has sprites.create in place so they only paint and add movement.	• After the effect works, ask them to say aloud what each block makes the character do before changing anything else. • Try two different effects and decide which suits the character, with a one-sentence reason.	• Sketch on paper how a second sprite (a coin) and a score bump might work for next session — plan only, no new kit. • Critique a neighbour's stack: spot one block that could snap in a clearer order and explain why.

Cross-curricular hook

Links to Maths algorithm thinking — ordered steps, predict–run–fix — the same sequence used in number procedures.