LESSON 16

# Strand 1 challenge build

40 minutes | Strand 1: Introducing computer science | Outcome 1.5 | Outcome 1.6 | Outcome 1.7 | Outcome 1.8 | Outcome 1.9

## WHAT THIS LESSON DOES

Combine planning, coding and testing skills into one small build. Choose a project, plan it with pseudo-code or a flow chart, build it in Scratch or MakeCode Arcade, test as you go, then share for feedback and improve it.

## CURRICULUM OUTCOMES

| Outcome | What the specification asks for |
|---|---|
| **1.5** | develop appropriate algorithms using pseudo-code and/or flow charts |
| **1.6** | construct code to implement algorithms |
| **1.7** | discuss and implement core features of structured programming languages, including variables, operators, loops, decisions, assignment and modules |
| **1.8** | test the code to determine its functionality and to identify and manage errors |
| **1.9** | evaluate the results in groups of two or three and discuss how the code worked and potential modifications to help improve it |

## WHAT STUDENTS SHOULD BE ABLE TO DO

- Plan a small build of your choice with pseudo-code or a flow chart, make it in Scratch or MakeCode Arcade, and test and debug it as you go (LOs 1.5, 1.6, 1.7, 1.8)
- Show your build to a group of two or three and act on one piece of feedback (LO 1.9)

### BEFORE THE BELL

- Check that Scratch and MakeCode Arcade both open on the room's machines, and know where the class saves work, since students save twice during the lesson.
- Optional: print or share a one-page starter scaffold (pre-labelled sprite and score variable) for students who need support.
- Student devices with Scratch and/or MakeCode Arcade ready; class way of saving projects available
- Student devices with Scratch and/or MakeCode Arcade; paper or on-screen space for pseudo-code or flow charts; board milestones ready; optional one-page starter scaffold (sprite + score) for support

# Strand 1 challenge build

### HOW THE LESSON RUNS

| | | |
|---|---|---|
| **4 min** | introduction | Start |
| **2 min** | content | The challenge |
| **2 min** | content | What done looks like, with an example |
| **8 min** | content | Plan it, then open Scratch |
| **12 min** | content | Build, test and save |
| **6 min** | content | Hands-on: Show, feedback, improve |
| **3 min** | content | Make sense |
| **2 min** | content | Reflect |
| **1 min** | summary | What you covered |

### TEACHING MOVES

- The challenge order, the done-looks-like list and the mini-collector example are on the pupil screens, so keep the three build milestones on the board instead: plan by minute 5, first run by minute 15 or cut scope, save by minute 29.
- During the build, circulate to check that students have written plans before they start coding.
- Call a one-minute freeze after five minutes so that plans are finished before any more blocks go in.
- At minute 15 call the checkpoint aloud: either the first run has happened, or the scope is cut now to one goal and one decision.
- Before a first run, ask the student what they expect to see, then let them run it and compare.
- Treat a plan plus one working path as enough to enter the share round, so LO 1.9 stays reachable for everyone.
- Default share groups to pairs and use threes only when numbers require it. Cap each run-through at about 60 seconds and hold three to four minutes at the end for making the change and saving again.
- Model how to give constructive feedback before the sharing starts.

### WHAT TO WATCH FOR

- Students begin coding without a plan: pause them and ask for the written plan first.
- Overly ambitious scope: point at the mini-collector example and remind students that one screen or one feature is enough. Hold the minute-15 cut-scope checkpoint.
- Feedback heard but not acted on: check that each student picks one specific improvement and makes it.

**DIFFERENTIATION**

- Support: point students at the plan frame on screen (Goal / Step 1 / Step 2 / Step 3 / Decision), the four starter ideas, and the mini-collector example.
- Support: offer a one-page starter scaffold with a pre-labelled sprite and score variable so they can focus on one decision or feature.
- Support: pair a student who is stuck on planning with a peer who has finished their plan.
- Extension: encourage adding a second feature once the first version is working.