technology

Scratch: loops and events

Lesson at a glance

Open on the IWB with the finished Penalty Shootout so pairs know the goal, then model each build beat: empty project, goal backdrop, Soccer Ball start code, stage click that broadcasts shoot, the ball's repeat loop, Casey the keeper, and the save/post/goal if checks. Pairs build and run after each model. Close with a display-only code talk on bugs fixed, then the Investigation Journal debug record.

Learning objectives

- Use a loop to repeat instructions in a Scratch program
- Use an event, the green flag, a click on the goal, or a message being received, to control when a script starts
- Debug a program that uses loops or events and record a fix or a change you tried

Before the bell — prep

Book devices and confirm Scratch.mit.edu access before the lesson. Have the goal backdrop image ready to download and a starter-project link on the board for pairs who cannot upload. Pre-open Scratch on the IWB so modelling starts immediately.

Materials

Item	Qty	Per	Source	Low-cost substitute
tablet or laptop with Scratch	1	pair	school devices	run the lesson on the IWB with pupils taking turns

Safety watch-point

Devices stay on scratch.mit.edu only. Supervise fair turns at shared machines and sensible volume; no account password sharing between pairs.

Lesson 27

Teaching moves

- **What we're building:** Show the finished game once on the IWB, then set the routine: watch the model, build in pairs, run it. Ask what jobs the code will need to do. Remind pairs to share the device and stay on Scratch only.
- **Create a new Scratch project:** Model a fresh project and delete the cat. Check every pair is on a genuinely new empty project before moving on — building inside an old project is the common slip. Pair slow logins with a ready pair.
- **Download and add the goal backdrop:** Show the upload path for the goal image. Keep the starter-project link visible for anyone stuck. Ask how a backdrop differs from a sprite, then move on quickly so coding stays the focus.
- **Add the Soccer Ball sprite:** Model the Soccer Ball and its green-flag start code. Ask which event fires this script. Point out x:0 y:-130 places the ball bottom-middle. Check the correct sprite is selected before pairs build.
- **Choose your shot:** Model adding when-stage-clicked on the backdrop, not a sprite. Explain broadcast as a signal other sprites can hear. Ask who will listen for shoot. Spot-check that the code sits on the stage.
- **Move the ball:** Model the receive-shoot script and the repeat loop. Have pairs predict the ball's path before running. Stress that broadcast and receive names must match exactly. Ask what changing the repeat count would do.
- **Add the Casey sprite:** Model Casey's green-flag setup and the left-right rotation style so the keeper stays upright. Confirm the code is on Casey, not the ball. List block names on the board for pairs who need a faster find.
- **Make the goalkeeper dive:** Model the dive and highlight pick-random so each save attempt differs — ask why that keeps the game fair. If nothing happens, check the receive block matches shoot. Run several trials to show the randomness.
- **Detect if the keeper makes a save:** Model the if-touching-Casey block inside the loop and stress stop-this-script sits inside the if. If the ball slides on after a save, the stop is outside — fix it on the spot. Ask why the ball must halt here.
- **Detect if it hits the post:** Model the colour-touching if and use the colour picker on the actual post, not a typed value. Have pairs aim deliberately at the post to test. Do the colour pick together on the IWB for anyone stuck.
- **Score a goal:** Explain aloud: if not saved and not post, it is a goal — so the say block goes after the loop, never inside. Have each pair play a full game checking saves, posts and goals. Offer a score variable as stretch.
- **Code talk — what worked and what we'd change:** Invite one pair to name a bug and the fix, then revoice their debugging for the class. Frame debugging as normal programming work. Send pairs to the Investigation Journal debug page — talk first, paper record second.

What it should show

A working game: green flag resets ball and keeper; a stage click broadcasts shoot; the ball loops toward the pointer while shrinking; Casey dives to a random spot; touching Casey says saved and stops; white post colour says post; finishing the loop without either says goal. If the ball never reacts, broadcast and receive names usually do not match.

Misconceptions & interventions

- **Pupils put the when-stage-clicked broadcast on a sprite instead of the backdrop, so clicks do nothing.** — Click the stage thumbnail on the IWB and rebuild the two blocks there. Have the pair click empty pitch and watch shoot fire.
- **Pupils think the ball should move on green flag alone, without receiving the shoot message.** — Trace the chain: stage click → broadcast shoot → ball receives shoot → loop runs. Run once with the receive detached so they see silence, then reattach.
- **The stop-this-script or goal say sits outside the if or inside the loop, so saves fail or goals spam every step.** — On the IWB, drag the blocks to the correct nest: stop inside the save if; goal say after the repeat ends. Replay one save and one goal to confirm.

Differentiation

Emerging	Developing	Proficient
• List the next block names on the board and sit at the teacher table to place green-flag and receive scripts on the correct sprite. • Offer the starter project with backdrop already loaded so the pair begins at the ball code.	• Ask what changing the repeat count or move steps does to the shot speed, then re-run once. • Prompt a second deliberate post aim to confirm the colour picker sample.	• Add a score variable that counts goals across several shots. • Critique fairness: does pick-random diving make saves too easy or too hard, and what would they change?

Cross-curricular hook

Link to Maths coordinates and chance — ball start at (0, −130) and pick-random dive directions make the save unpredictable.