LESSON 1

First Arcade Project

60 minutes | Advanced Game Development

WHAT THIS LESSON DOES

Get started with creating your own arcade game using MakeCode Arcade. Follow step-by-step instructions to build a project, add sprites, design maps, and programme movement and interactions. Create, test, and play your game in a browser or on a handheld device.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand and utilise MakeCode Arcade for creating games.
- Manipulate the Code Editor to build and modify game elements.
- Create and customise sprites for use in a game.
- Develop a tile map and implement walls for game navigation.
- Implement game mechanics such as projectiles, sprite movement, and life count.

BEFORE THE LESSON

- Open and test arcade.makecode.com on the class network beforehand: it is a large site and school filters or slow wifi can block or stall it.
- One device per pupil, or per pair if devices are short. The game builds up across steps so pupils need to stay on the same project throughout.
- Turn classroom sound on and have headphones ready for the projectile hit sound in the later steps.
- Optional: if you have BrainPad or GameGo handhelds, have them charged and connected so pupils can download their finished game.

First Arcade Project

HOW THE LESSON RUNS

- What is MakeCode Arcade?
- The Code Editor
- Create a new Arcade project
- Add a sprite
- Choose a sprite from the gallery
- Move your sprite
- Draw the tile map
- Draw the walls
- Camera follow sprite
- Add some projectiles
- Set their direction and speed
- Detect overlap
- Lose a life
- Send the code to your handheld

TEACHING MOVES

- 1. What is MakeCode Arcade? Link it to Scratch or micro:bit blocks pupils have used before, so the block-snapping feels familiar rather than new. Show the arcade.makecode.com site on the board.
- 2. The Code Editor. Point to each of the three parts on screen as you name them: simulator, toolbox, code area. Have pupils find each one on their own device before moving on.
- 3. Create a new Arcade project. Watch that everyone reaches a blank project and names it. This is the step where login or site-access problems surface, so sort them here before pupils are behind.
- 4. Add a sprite. Explain that a sprite is being stored in a variable, the same box that usually holds numbers or text. Pull the block from the toolbox rather than typing it.
- 5. Choose a sprite from the gallery. Steer pupils to pick a gallery sprite for now. Reassure the keen designers they can paint their own once the game works.
- 6. Move your sprite. Name the eight buttons, then check the simulator: pupils should be able to drive their sprite up, down, left and right before continuing.
- 7. Draw the tile map. Keep the first map simple: just a border around the edge. Demonstrate opening the tile editor once on the board so everyone sees where to draw.
- 8. Draw the walls. Stress that walls are drawn with a separate tool over the tiles. Have pupils test straight away that the sprite cannot pass through the wall.
- 9. Camera follow sprite. Show first why the sprite disappears off the edge, then add the camera block so the fix is meaningful rather than copied blindly.
- 10. Add some projectiles. Explain the two-second interval before pupils add the block, so they understand why a projectile keeps appearing on its own.
- 11. Set their direction and speed. Use the vx and vy table on the board. Let pupils change one value at a time and watch which side of the screen the projectile starts from.
- 12. Detect overlap. Explain overlap as the game noticing two sprites touch. Point out the sound and spray effect fire on contact, and check sound is on.
- 13. Lose a life. Walk through the three added actions: start with three lives, lose one on a hit, destroy the projectile. Test that lives count down in the info bar.
- 14. Send the code to your handheld. Have pupils play their finished game and try adding one new thing. If handhelds are available, help pupils download their code onto them.

WHAT TO WATCH FOR

- Pupils think the projectile appears once when they add the block, rather than every two seconds on a repeating interval. Point at the onUpdateInterval block and count aloud with the simulator so they see it fire again and again on a timer.
- Pupils confuse the sprite variable with the number or text variables they have used before, and expect it to hold a value. Show that this variable holds a whole sprite, the character itself, and that everything after refers back to that same stored sprite.
- Pupils draw the tile map but forget the walls are a separate tool, then wonder why the sprite still walks off the edge. Reopen the editor together and show the Draw Walls tool sits apart from tile drawing. Both layers are needed.

DIFFERENTIATION

- Emerging: Pair the pupil with a partner and let them build on one device together, taking turns to drag blocks. Get the sprite moving and the walls working first. A game that moves is a success even without projectiles.
- Developing: Have them explain back to you what the vx and vy values do before they change them, so the projectile direction is understood not guessed. Prompt them to test after each block rather than building several at once.
- Proficient: Invite them to paint their own sprite in the editor instead of using the gallery. Challenge them to add one new feature the lesson does not name, such as more projectiles or a different sound on a hit.