

LESSON 20

Avoid the Enemy

60 minutes (2 periods)

Introduction to Game Design

WHAT THIS LESSON DOES

This lesson guides students through creating an 'Avoid the Enemy' game using MakeCode Arcade. Students will learn how to create and control sprites, make them interact, and use the game update loop for dynamic gameplay. No prior coding experience is necessary.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Develop understanding of coding principles through creation of a game.
- Gain proficiency in creating and controlling sprites in a game environment.
- Learn to implement game dynamics using update loops.
- Understand and apply the concept of sprite interaction within a game.
- Enhance problem-solving skills by debugging and improving game functionality.

BEFORE THE LESSON

- Open <https://arcade.makecode.com> on a class device and confirm it loads through the school filter.
- Have one device per pupil or pair, charged, with keyboard or on-screen controls ready.
- Be ready to project the site so you can model each step as pupils build.

Avoid the Enemy

HOW THE LESSON RUNS

Introduction to Avoid the Enemy Game

Create a new Arcade project

Create your player sprite

Move the sprite

Make it stay on the screen

Create the enemy sprites

Set the enemy position

Make it chase the player sprite

Game Over, Well Done!

TEACHING MOVES

1. Introduction to Avoid the Enemy Game. State the goal in one line: build a game where a player sprite dodges enemies that appear and chase them. Then move straight to MakeCode.
2. Create a new Arcade project. Send everyone to arcade.makecode.com and wait until every device has a new blank project before you continue.
3. Create your player sprite. Model `sprites.create`, open the gallery, and set the kind to Player. Check the room that nobody left the kind on the default.
4. Move the sprite. Add the controller movement code, then have pupils test with arrow keys or the joystick before adding anything else.
5. Make it stay on the screen. Add the stay-on-screen code and ask pupils to drive into an edge so they see the sprite stop instead of disappearing.
6. Create the enemy sprites. Build the `onUpdateInterval` at 5000 ms together. Stress `SpriteKind.Enemy` and a gallery image so a new enemy appears every five seconds.
7. Set the enemy position. Set spawn to the top left and run the game once so pupils see enemies no longer appear on top of the player.
8. Make it chase the player sprite. Add follow at speed 10, then give quiet play-test time. Ask: can you still dodge when more than one enemy is on screen?
9. Game Over, Well Done! Celebrate a working game. Invite two or three pupils to show one bug they fixed or one choice they made for their sprites.

WHAT TO WATCH FOR

- Pupils leave the enemy on the wrong sprite kind, so it is not treated as an Enemy in the game. Pause at the gallery step and have them read the kind aloud: it must say Enemy, not Player or the default.
- Pupils think the enemy should appear where the player is, so both sprites start stacked in the centre. Run their project once, point at the overlap, then add the top-left position and compare.
- Pupils expect one enemy only, or think the update interval runs once rather than every five seconds. Watch together for ten seconds and count how many enemies appear. Link that count back to the 5000 ms interval.

DIFFERENTIATION

- Emerging: Sit with them for the first `sprites.create` and `gallery pick` so they have a visible player before working on movement. Pair them with a partner who names each block while they snap it in.
- Developing: After each run, ask them to tell you what the newest code changed before they add the next part. If the chase fails, have them compare their follow code and sprite names to your projected example.
- Proficient: Ask them to explain to a neighbour how the five-second interval and the follow speed work together. Challenge them to change the follow speed in the same block and say whether the game got easier or harder.