Unit 2.2: We are games testers

Working out the rules for games

Software: Scratch, FixTheFactory (alternative: Blockly Games)

Hardware: iPads/Android tablets, laptops/desktop/Chromebook computers for Scratch

Overview

In this unit, pupils play some **Scratch** games, trying to work out the rules of the game, i.e. the **algorithms** the programmers have used. They also play a simple coding-based game and discuss game playing. In:

- **Session 1** pupils work out the rules (algorithms) for a simple arithmetic game
- **Session 2** pupils work out the rules (algorithms) for a chase game
- **Session 3** pupils work out the rules (algorithms) for a two-player sports game
- **Session 4** pupils work out the rules (algorithms) used in a shooting game
- **Session 5** pupils play a professionally produced coding-based game
- **Session 6** pupils play a turn-based two-player game, working together to identify winning strategies.

Alternatives

The unit sessions give step-by-step guidance on carrying out this unit using Scratch. An alternative to FixTheFactory is Blockly Games.

Knowledge, skills and concepts

In this unit, pupils will learn to:

- observe and describe carefully what happens in computer games
- use logical reasoning to make predictions of what a program will do and test these
- think critically about computer games
- create sequences of instructions for a virtual robot to solve a problem
- work out strategies for playing a game well
- be aware of how to use games safely and in balance with other activities.

Progression

In Key Stage 1:

- Pupils learned about **input**, **output** and **repetition** in **Unit 2.1: We are astronauts**.

In Key Stage 2:

- In **Unit 3.2: We are bug fixers**, pupils use computational thinking to debug Scratch programs.
- In **Unit 4.1: We are software developers**, pupils develop another arithmetic game.
- In **Unit 5.1: We are game developers**, pupils develop their own games in Scratch.

Differentiation

See each session (pages 23–28) for ways to increase support and add challenge to this unit.

Pupils learning EAL can view the **source code** of the **Scratch** games in their first language using the globe icon in the Scratch editor.

Cross-curricular opportunities

English: Pupils' own explanations of how algorithms work will link with their work in English.

Maths: The addition race game supports pupils' learning in maths. Thinking ahead to solve problems is an important aspect of mathematical reasoning.

Science: When testing their predictions, pupils are using a simple version of the scientific method.

PE: Pupils can compare games that simulate sports (such as the Scratch tennis game) to the real thing.

Other: Computer games offer engaging interactive simulations of curriculum topics. Pupils can also consider the artwork, music, sound and stories, which are essential components of many games.

Assessment – by the end of the unit:

All pupils can:

- understand that computer games are made up of precise instructions for the computer to follow
- understand that programmers implement many **algorithms** when making computer games
- use logical reasoning to make predictions about what happens next in a game
- suggest improvements to simple computer games
- be aware of and observe age restrictions on commercial games
- solve problems by working out short sequences of simple instructions
- follow the rules for playing a game.

Most pupils can:

- describe clearly what happens in a computer game
- conduct tests to check their predictions
- notice common features in several game algorithms
- create longer sequences of instructions to solve problems
- identify some winning strategies in a game.

Some pupils can:

- understand how Scratch **source code** determines the behaviour of a game they play
- make changes to the Scratch source code for simple computer games
- think one or two moves ahead when playing a strategy game
- predict what their opponent will do to win when playing a strategy game.

Background information

- Gaming has much in common with programming. There is:
  - a clear sense of what can be achieved
  - a set of formal rules governing what happens
  - a varying of scope between detail and 'big picture' thinking.
- This unit helps pupils develop their **computational thinking** through 'reverse engineering' some simple computer games – the pupils try to work out how the game operates, i.e. what **algorithms** the programmer used when making the game.
- Pupils should then look at the **source code** and compare it to what they noticed while playing the game. Just as it is important that pupils read books as well as writing their own stories, it is important that they spend time reading programs written by others, as well as writing their own.
- Session 5 helps pupils to develop their programming skills further by creating sequences of instructions to solve problems for a virtual robot.
- Session 6 helps pupils with computational thinking as they look for strategies for playing Nim, a two-player game where players share a set of counters. Nim might be credited as the first ever computer game – a version was automated at the New York World's Fair of 1939, a few years before early digital computers.

Key vocabulary

Abstraction: computational thinking approach to managing complexity by simplifying things through identifying what is important, and what detail can be hidden

Algorithm: a sequence of precise instructions or steps (sometimes a set of rules) to achieve an objective

Computational thinking: a way of looking at problems so that the solution can be automated using a computer

Input: data supplied to a computer – in this case, it is a mouse click, keyboard press or tapping on a tablet

Output: information produced by a computer – in this case, it is moving sprites on a screen

Parallel processing: when programs run (or appear to run) simultaneously

Pattern recognition: computational thinking approach in which common aspects of how a system behaves are used to simplify implementing solutions

Remix: to take a project and make changes to its source code

Repetition: programming construct which allows a group of instructions to be repeated a number of times, or until a certain condition is met

Scratch: simple, block-based programming language in which programs for characters are built by snapping together code blocks

Source code: the code that a particular program follows; the instructions or rules that determine what happens in a game or other application

Sprite: a graphical character in a program that can be given its own sequence of instructions

Preparation for teaching the unit

- Check you have access to Scratch, and the Scratch games (see *Resources needed*).
- Install FixTheFactory on the iPads/Android tablets (or check access to Blockly Games).
- Read pages 20–21 to get an overview of the unit.
- Read the steps in the unit sessions (pages 23–28) and look at the associated online resources, printing out the worksheets as required.
- Watch the videos for this unit.
- Work through the unit yourself so you know what is expected of the pupils.
- Review the **source code** for the four Scratch games used in Sessions 1–4, so that you can help pupils make connections.

Resources needed

- **Software:** Scratch, FixTheFactory
- **Hardware:** iPads/Android tablets, laptop/desktop/Chromebook computers
- See *Alternatives* on page 20

Online resources provided

Session resources

- Worksheet 2.2a: Addition race game
- Worksheet 2.2b: Fish game
- Worksheet 2.2c: Tennis game
- Worksheet 2.2d: Duck shooting
- Worksheet 2.2e: A coding game
- Worksheet 2.2f: End-of-unit quiz
- Worksheet 2.2g: Pupil self-assessment
- Teaching slides: 2.2a–2.2f
- Scratch games:
  - Addition race: **scratch.mit.edu/projects/15905989/**
  - Fish: **scratch.mit.edu/projects/15906446/**
  - Tennis game: **scratch.mit.edu/projects/15906870/**
  - Duck Shoot: **scratch.mit.edu/projects/15907506/**
  - Nim: **scratch.mit.edu/projects/330713349/**
- Walkthrough video: 2.2a
- Interactive end-of-unit quiz 2.2

Additional resources

- Software in 60 Seconds: Introduction to Scratch
- Software in 60 Seconds: Scratch 1–3
- CPD video: Instructions for sprites

Online safety

- Pupils will be using Scratch online in an Internet browser. Ensure that the appropriate filters and monitoring software is in place.
- The Scratch online community is generally a safe, well-moderated space but if pupils encounter inappropriate content, they should turn off their screen or turn over their tablet, and alert an adult. Scratch moderators can also be informed.
- Pupils need individual accounts to participate in the Scratch community, which requires parental approval. It would be wise to leave this until pupils are older. Pupils with accounts need to know how to contribute positively and what to do if they encounter inappropriate behaviour.
- If you become aware of pupils playing age-inappropriate games at home, you should inform your designated safeguarding lead, in accordance with your school's safeguarding policy.

Collaboration

Pupils work together to record their ideas when looking at games, and later collaborate to identify winning Nim strategies.

Useful links

Software and tools

- Scratch: **www.scratch.mit.edu**
- FixTheFactory: App Store/Google Play Store
- Blockly Games: **blockly.games**
- Pong: **www.ponggame.org**
- Codergeeks space invader game: **scratch.mit.edu/projects/330698413/**
- Cargo-Bot: **www.altermanchess.wixsite.com/cargobot**
- Lightbot Code Hour: **www.lightbot.com/hour-of-code.html**
- Bee-Bot emulator: **scratch.mit.edu/projects/11074524/editor/**

Online tutorials

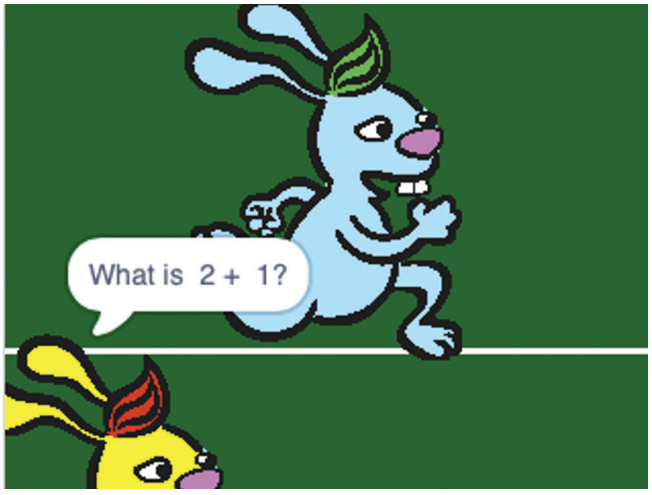
- Tutorials for Scratch are built in to the editor
- Nim: **www.youtube.com/watch?v=sfVvdsfdV2g**
- FixTheFactory: **www.youtube.com/watch?v=GPqpq09qQqc**

Information and ideas

- Scratch creative computing curriculum: **www.creativecomputing.gse.harvard.edu/guide**
- Nim: **www.cs4fn.org/binary/nim/nim.php**
- Pong: **en.wikipedia.org/wiki/Pong**

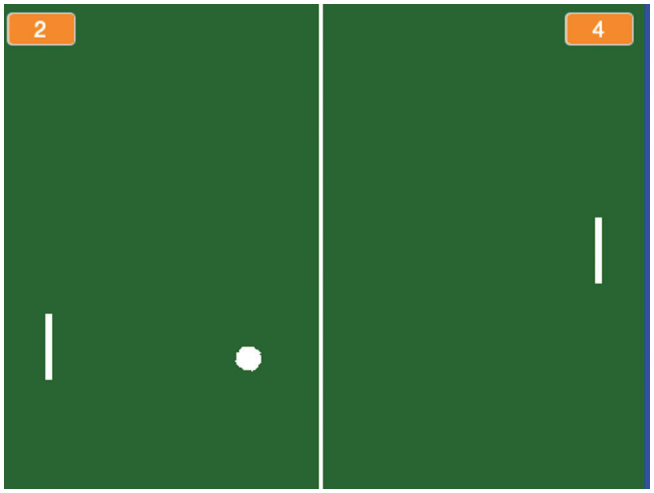
Unit games

Below are the Scratch games used in this unit.



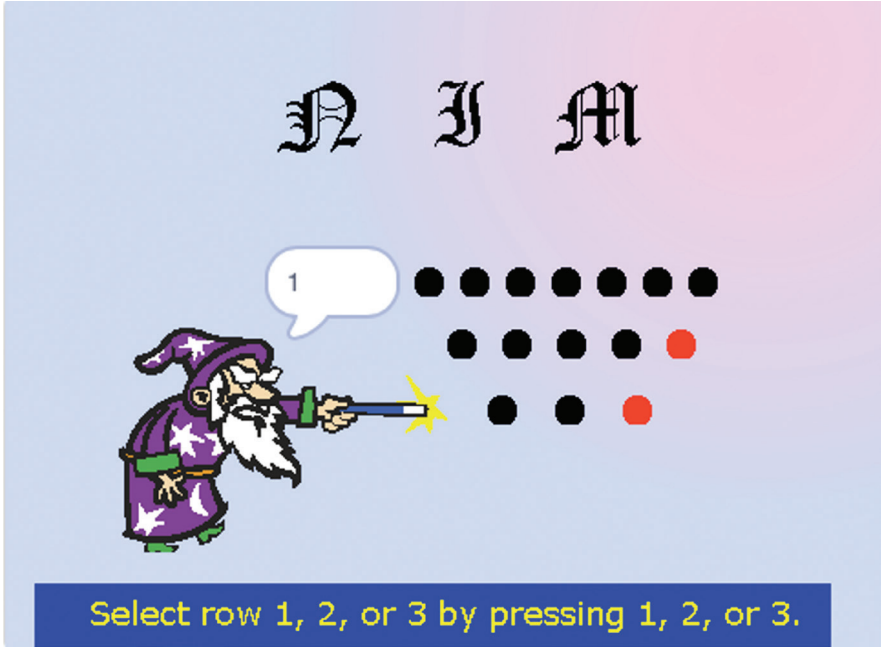
Session 1: Addition race game

Session 2: Fish game



Session 3: Tennis game

Session 4: Duck shooting game



Session 6: Nim