



Year 4 — Repetition in games

Unit introduction

Learners will explore the concept of repetition in programming using the Scratch environment. The unit begins with a Scratch activity similar to that carried out in Logo in Programming unit A, where learners can discover similarities between two environments. Learners look at the difference between count-controlled and infinite loops and use their knowledge to modify existing animations and games using repetition. Their final project is to design and create a game which uses repetition, applying stages of programming design throughout.

If learners are using the online version of Scratch, be aware this allows them to share and comment on projects. A simplified version of the Scratch's community guidelines can be found at the end of this unit guide. For the full guidelines, see the [Scratch website](#).

Software and Hardware requirements

Learners will need to have access to [Scratch](#) for this unit. The online version of Scratch runs via a web browser and can be accessed on desktops, laptops and tablets. You may want to consider setting up a [teacher account](#), to create logins for learners to save and access their projects. If internet connectivity is an issue in school, Scratch can be accessed offline via the [Scratch app](#).

If you've adapted this unit to better suit your school, please [share your adapted resources](#) with fellow teachers in the STEM community. Alternatively, if this unit isn't quite right for your school, why not see if an adapted version which better suits has already been shared?

Overview of lessons

Lesson	Brief overview	Learning objectives
--------	----------------	---------------------

1 Using loops to create shapes	In the first lesson, learners look at real-life examples of repetition, and identify which parts of instructions are repeated. Learners then use Scratch, a block-based programming environment, to create shapes using count-controlled loops. They consider what the different values in each loop signify, then use existing code to modify and create new code, and work on reading code and predicting what the output will be once the code is run.	To develop the use of count-controlled loops in a different programming environment • I can list an everyday task as a set of instructions including repetition • I can predict the outcome of a snippet of code • I can modify a snippet of code to create a given outcome
2 Different loops	In this lesson, learners look at different types of loops: infinite loops and count-controlled loops. They practise using these within Scratch and think about which might be more suitable for different purposes.	To explain that in programming there are infinite loops and count-controlled loops • I can modify loops to produce a given outcome • I can choose when to use a count-controlled and an infinite loop • I can recognise that some programming languages enable more than one process to be run at once
3 Animate your name	In this lesson, learners create designs for an animation of the letters in their names. The animation uses repetition to change the costume (appearance) of the sprite. The letter sprites will all animate together when the event block (green flag) is clicked. When they have designed their animations, the learners will program them in Scratch. After programming, learners then evaluate their work, considering how effectively they used repetition in their code.	To develop a design that includes two or more loops which run at the same time • I can choose which action will be repeated for each object • I can explain what the outcome of the repeated action should be • I can evaluate the effectiveness of the repeated sequences used in my program
4 Modifying a game	In this lesson, learners look at an existing game and match parts of the game with the design. They make changes to a sprite in the existing game to match the design. They then look at a completed	To modify an infinite loop in a given program • I can identify which parts of a loop can be changed

	design, and implement the remaining changes in the Scratch game. They add a sprite, re-use and modify code blocks within loops, and explain the changes made.	• I can explain the effect of my changes • I can re-use existing code snippets on new sprites
5 Designing a game	In this lesson, learners look at a model project that uses repetition. They then design their own games based on the model project, producing designs and algorithms for sprites in the game. They share these designs with a partner and have time to make any changes to their design as required.	To design a project that includes repetition • I can evaluate the use of repetition in a project • I can select key parts of a given project to use in my own design • I can develop my own design explaining what my project will do
6 Creating your games	In this lesson, learners build their games, using the designs they created in Lesson 5. They follow their algorithms, fix mistakes, and refine designs in their work as they build. They evaluate their work once it is completed, and showcase their games at the end.	To create a project that includes repetition • I can refine the algorithm in my design • I can build a program that follows my design • I can evaluate the steps I followed when building my project

Request a computing ambassador

This unit is ideal for linking to the world of careers, and a computing ambassador can support this. Through the [STEM ambassador platform](#), you can search for a computing ambassador. If you cannot find a computing ambassador with an offer to support this unit, then the following request will help to match you with the right person. You will need to edit the areas in red to ensure the request is right for your school.

Year 4 (ages 8-9) are learning about repetition in programming through the [Teach Computing Curriculum unit of six lessons](#). Within these lessons, pupils will learn the skills needed to design and create a game on Scratch.

*Our lessons are taking place from ***date*** to ***date*** and we would appreciate someone with skills in this area to offer some real-world experience to this unit. The unit uses the programming interface of [Scratch](#), with pupils coding via drag and drop block-based code, and focuses on the following areas:*

- *look at the difference between count-controlled and infinite loops in programming*

- *modify existing animations and games using repetition*
- *design a game, producing designs and algorithms for sprites*
- *build a game following my design algorithm*

*We require an ambassador who can support in any of these areas. We are hoping for an ambassador who would be willing to join us ***in the classroom/virtually*** to support our learning by ***providing some handy hints and tips for our projects/giving us constructive feedback on our final projects/discussing how programming is used within their profession and in the real-world of game design.****

Subject knowledge and CPD opportunities

This unit focuses on developing learners' understanding of repetition within the Scratch programming environment. Repetition is where actions or commands in programming are repeated. The repeating commands can also be referred to as a 'loop'. Loops can be repeated indefinitely (known as 'infinite loops'), or for a set number of times (known as 'count-controlled loops').

Repetition

You will need to know that repetition is used in programming to give the same instruction or set of instructions several times. Repetition uses loops as the means to give these instructions. This unit makes use of two types of loops: infinite and count-controlled, which have been defined below:

- **Infinite loop:** An infinite loop is a loop that commands the instruction/set of instructions to repeat forever. When an infinite loop is used in a program, there is no way of ending the program, as the command(s) within the loop will be repeated endlessly. For this reason, infinite loops should only be used when writing a program that is intended to run forever. The exception to this is when using selection in physical computing, as you will see throughout this unit.
- **Count-controlled loop:** A count-controlled loop is a form of repetition in which a set of commands are carried out a specific number of times. Count-controlled loops should only be used when it is known how many times a set of commands needs to be repeated.
- **Condition-controlled loop:** A condition-controlled loop is a form of repetition in which a set of commands stop being carried out when a condition is met. The condition could be anything from when the 'score' in a game reaches a certain value to when a key on a keyboard has been pressed.

Continual Professional Development

Enhance your subject knowledge to teach this unit through the following free CPD:

- [Getting started in Year 4 – short course](#)
- [Introduction to primary computing remote or face to face](#)
- [Introduction to Programming with Scratch](#)
- [Teaching programming using Scratch and Scratch Jr](#)

Teach primary computing certificate

To further enhance your subject knowledge, enrol on the [teach primary computing certificate](#). This will support you to develop your knowledge and skills in primary computing and gain the confidence to teach great lessons, all whilst earning a nationally recognised certificate!

Progression

This unit assumes that learners will have some prior experience of programming. The KS1 NCCE units cover floor robots and ScratchJr, and Scratch, and the skill of sequence, is introduced in the Year 3 programming units: [Sequencing Sounds](#) and [Events and actions in programs](#). However, experience of other languages or environments may also be useful.

Common Misconceptions

Learners may have misconceptions around the language used within this unit. They may believe that ‘repetition’ and ‘loop’ are different entities, but repeating commands can also be referred to as a ‘loop’. They may also get confused between infinite loops and count-controlled loops, as both have different outcomes. Infinite loops can be repeated without an end (forever loop), whereas count-controlled loops will run for a set number of times (repeat _ times). Learners may believe that infinite loops are the easiest to use, however you may want to share that programmers do not actually use infinite loops very often as they can get stuck, and the only way to stop them is to close the program and start again.

Curriculum links

[Computing](#)

- Design, write, and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts

- Use sequence, selection, and repetition in programs; work with variables and various forms of input and output
- Use logical reasoning to explain how some simple algorithms work, and to detect and correct errors in algorithms and programs
- Select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information

Assessment

Formative assessment

Assessment opportunities are detailed in each lesson plan. The learning objectives and success criteria are introduced in the slide decks at the beginning of each lesson and then reviewed at the end. Learners are invited to assess how well they feel they have met the learning objective using thumbs up, thumbs sideways, or thumbs down.

Summative assessment

Please see the assessment rubric document for this unit. The rubric can be used to assess student's work from lessons 5 and 6.

Scratch guidelines

- **Stay Safe Online:** Don't share personal info like your full name, address, or phone number. Also, don't share details about where you go to school or your social media accounts.
- **Be Kind and Helpful:** When you comment on someone's project, say something nice about it and offer suggestions in a friendly way. Don't be mean or spammy.
- **Share and Collaborate:** You can use other people's stuff on Scratch to make your own cool projects but remember to give credit. And when you share your work, others can use it too, as long as they give credit and make changes.
- **Be Honest:** Always tell the truth and be yourself when you're on Scratch. Don't pretend to be someone else.

- **Keep Scratch Friendly:** Make sure your creations and chats are friendly for everyone. If you see something mean or inappropriate, you can click the link that says "report" on any project, comment, discussion post, studio, or profile page. If you're unsure or it's a bit complicated, you can ask your teacher or a trusted adult to get in touch with us. The Scratch team will take care of it.

Resources are updated regularly — the latest version is available at: ncce.io/tcc.

Attribution statement

This resource was created by Raspberry Pi Foundation and updated by STEM Learning for the National Centre for Computing Education.

The contents of this resource are available for use under the [Open Government License](#) (OGL v3) meaning you can copy, adapt, distribute and publish the information. You must acknowledge the source of the Information in your product or application, by attributing Raspberry Pi Foundation and STEM Learning as stated here and are asked to provide a link to the [OGL v3](#).

The original version can be made available on request via info@teachcomputing.org.