Year 4 – Programming A – Repetition in shapes

Unit introduction

Learners will create programs by planning, modifying, and testing commands to create shapes and patterns. They will use Logo, a text-based programming language. This unit is the first of the two programming units in Year 4 and looks at repetition and loops within programming.

Software and Hardware requirements

The unit has been written with screenshots from FMSLogo, which is a free downloadable piece of software for desktops or laptops. If using a Chromebook or tablet, you can access the alternative software Turtle Academy, which has a near identical interface but does contain adverts. Alternatively, J2Logo Level 3 can be used, but be aware this also allows pupils to select pieces of code by clicking.

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 Programming a screen turtle	This lesson will introduce pupils to programming in Logo. Logo is a text-based programming language where pupils type commands that are then drawn on screen. Pupils will learn the basic Logo commands, and will use their knowledge of them to read and write code.	To identify that accuracy in programming is important ● I can program a computer by typing commands ● I can explain the effect of changing a value of a command

		• I can create a code snippet for a given purpose
2 Programming letters	In this lesson, pupils will create algorithms (a precise set of ordered instructions, which can be turned into code) for their initials. They will then implement these algorithms by writing them in Logo commands to draw the letter. They will debug their code by finding and fixing any errors that they spot.	To create a program in a text-based language • I can use a template to draw what I want my program to do • I can write an algorithm to produce a given outcome • I can test my algorithm in a text-based language
3 Patterns and repeats	In this lesson, pupils will first look at examples of patterns in everyday life. They will recognise where numbers, shapes, and symbols are repeated, and how many times repeats occur. They will create algorithms for drawing a square, using the same annotated diagram as in Lesson 2. They will use this algorithm to program a square the ‘long’ way, and recognise the repeated pattern within a square. Once they know the repeated pattern, they will use the `repeat` command within Logo to program squares the ‘short’ way.	To explain what ‘repeat’ means • I can identify repetition in everyday tasks • I can identify patterns in a sequence • I can use a count-controlled loop to produce a given outcome
4 Using loops to create shapes	In this lesson, pupils will work with count-controlled loops in a range of contexts. First, they will think about a real-life example, then they will move on to using count-controlled loops in regular 2D shapes. They will trace code to predict which shapes will be drawn, and they will modify existing code by changing values within the code snippet.	To modify a count-controlled loop to produce a given outcome • I can identify the effect of changing the number of times a task is repeated • I can predict the outcome of a program containing a count-controlled loop • I can choose which values to change in a loop

5 Breaking things down	In this lesson, pupils will focus on decomposition. They will break down everyday tasks into smaller parts and think about how code snippets can be broken down to make them easier to plan and work with. They will learn to create, name, and call procedures in Logo, which are code snippets that can be reused in their programming.	To decompose a task into small steps ● I can identify 'chunks' of actions in the real world ● I can use a procedure in a program ● I can explain that a computer can repeatedly call a procedure
6 Creating a program	In the final lesson, pupils will apply the skills that they have learnt in this unit to create a program containing a count-controlled loop. Over the course of the lesson, they will design wrapping paper using more than one shape, which they will create with a program that uses count-controlled loops. They will begin by creating the algorithm, either as an annotated sketch, or as a sketch and algorithm, and then implement it as code. They will debug their work throughout, and evaluate their programs against the original brief.	To create a program that uses count-controlled loops to produce a given outcome ● I can design a program that includes count-controlled loops ● I can make use of my design to write a program ● I can develop my program by debugging it

Subject knowledge and CPD opportunities

You will need to be able to access and demonstrate the version of Logo that you are using. You will also need to be aware of the Logo commands used in this unit. You can find these in the glossary which is part of Lesson 3 of this unit.

This unit focuses on repetition, where actions or commands in programming are repeated. The repeating commands can also be placed into a loop. Loops can be repeated indefinitely, or a set number of times — the latter are called 'count-controlled loops'.

Different pedagogies are used in this programming unit. For example, pupils will encounter Parson's Problems, which are programming puzzles where the pupil is given the correct code, but the commands have been split and mixed up. Pupils will also carry out code tracing, where they will read through the code line by line and say exactly what each command will make happen when it runs.

In Lesson 5, pupils will look at decomposition and procedures. They will decompose code snippets, breaking them down to make them easier to plan and work with. They will use these broken down chunks to help recognise patterns in their programming.

Pupils will create and call procedures in Logo. Procedures are code snippets that are named and can be reused in their programming. When creating a procedure, the word ‘TO’ is typed, followed by the procedure name, eg `TO SQUARE`.

Note: A subroutine is a sequence of commands to perform a specific task with an identifiable name. In programming there are a number of constructs which are variations of a subroutine; these include subprogram, **procedures** and functions.

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
- Teaching programming to 5- to 11-year-olds

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 progresses students’ knowledge and understanding of programming. Within the Year 3 units, Programming A- Sequencing Sounds and Programming B- Events and Actions in programs, learners will have an awareness of the sequence of commands in a program. This unit progresses on to using count-controlled loops in those sequences. Pupils will create algorithms and then implement those algorithms as code.

Common Misconceptions

Learners may have the misconception that repetition is just about making things faster or easier, however it is a fundamental programming concept that allows for efficient and scalable code.

When using FMS Logo, learners may believe that slight variations in syntax or structure do not matter, and that the computer will understand what is meant. Even small errors such as misspelling (FWD instead of FD) or omitting a space (FD50 instead of FD 50), can cause a program to fail. They should learn that precision is important in coding. Learners may also not realise that the turtle's position and orientations are relative to its current state. They may believe that the right and left code moves the turtle in that direction, however in reality it just turns the turtle on the spot the number of degrees stated.

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 summative assessment document of multiple-choice questions for this unit. This can be downloaded as a paper copy, with answers, or in a digital format to be shared.

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