Fellside Community Primary School Computing Curriculum

Year 5 – Programming A – Selection in physical computing

Unit introduction

In this unit, learners will use physical computing to explore the concept of selection in programming through the use of the Crumble programming environment. Learners will be introduced to a microcontroller (Crumble controller) and learn how to connect and program components (including output devices — LEDs and motors) through the application of their existing programming knowledge. Learners will be introduced to conditions as a means of controlling the flow of actions, and explore how these can be used in algorithms and programs through the use of an input device (push switch). Learners will make use of their knowledge of repetition and conditions when introduced to the concept of selection (through the 'if... then...' structure) and write algorithms and programs that utilise this concept. To conclude the unit, learners will design and make a working model of a fairground carousel that will incorporate their understanding of how the microcontroller and its components are connected, and how selection can be used to control the operation of the model. Throughout this unit, pupils will apply the stages of programming design.

There are two Year 5 programming units:

- Programming A – Selection in physical computing
- Programming B – Selection in quizzes

This is unit A, which should be delivered before unit B.

Overview of lessons

Lesson	**Brief overview**	**Learning objectives**

1 Connecting Crumbles	In this lesson, learners will become familiar with the Crumble controller, some of its associated components, and the programming environment used to control it. They will explore how the items connect together to create a complete circuit, and how to construct programs that turn an LED on and off and set its colour. Learners will apply their understanding of repetition by identifying how their programs can be modified to make an LED flash continuously.	To control a simple circuit connected to a computer ● I can build a simple circuit to connect a microcontroller to a computer ● I can program a microcontroller to light an LED ● I can explain why I used an infinite loop
2 Combining output devices	In this lesson, learners will develop their knowledge of a Crumble controller further by connecting additional devices (another Sparkle and a motor) to the controller, and they will construct programs to control more than one of these. They will design sequences of actions for these output devices. They will then apply their understanding of repetition by using count-controlled loops when implementing their design as a program.	To write a program that includes count-controlled loops ● I can connect more than one output device to a microcontroller ● I can design sequences for given output devices ● I can decide which output devices I control with a count-controlled loop
3 Controlling with conditions	In this lesson, learners will be introduced to conditions, and how they can be used in algorithms and programs to control their flow. They will identify conditions in statements, stating if they are true or false, and learn how they can be used to	To explain that a loop can stop when a condition is met, eg number of times

	start and stop a set of actions. Learners will be introduced to a Crumble switch, and learn how it can provide the Crumble controller with an input that can be used as a condition. They will explore how to write programs that use an input as a condition, and use this knowledge to write a program that uses a condition to stop a repeating light pattern.	● I can explain that a condition is something that can be either true or false (eg whether a value is more than 10, or whether a button has been pressed) ● I can experiment with a 'do until' loop ● I can program a microcontroller to respond to an input
4 Starting with selection	In this lesson, learners will develop their understanding of how the flow of actions in algorithms and programs can be controlled by conditions. They will be introduced to selection, and learn to represent conditions and actions using the 'if… then…' structure. They will apply their understanding by using selection in an algorithm created to meet the requirements of a task. They will discover that infinite repetition is required when programming input devices to repeatedly check if a condition has been met.	To conclude that a loop can be used to repeatedly check whether a condition has been met ● I can explain that a condition being met can start an action ● I can identify a condition and an action in my project ● I can use selection (an 'if… then…' statement) to direct the flow of a program
5 Drawing designs	In this lesson, learners will apply their understanding of microcontrollers, output devices, and selection when designing a project to meet the requirements of a	To design a physical project that includes selection

	given task. To ensure their understanding, they will identify how selection might be used in real-world situations, then they will consider how they can apply this knowledge when designing their project. They will produce detailed drawings to show how their model will be made and how they will connect the microcontroller to its components.	● I can identify a condition to start an action (real world) ● I can describe what my project will do (the task) ● I can create a detailed drawing of my project
6 Writing and testing algorithms	In this final lesson of the unit, learners will build on the designs that they developed in Lesson 5 by creating an algorithm to meet the requirements of the given task. They will identify how they are going to use selection before writing their algorithm. They will then move into the code level to test their algorithm by implementing it as a program, running it, identifying any bugs, and returning to the algorithm to debug it where necessary. Finally, to conclude the unit, they will evaluate their algorithms and other areas of their designs.	To create a controllable system that includes selection ● I can write an algorithm to control lights and a motor ● I can use selection to produce an intended outcome ● I can test and debug my project

Progression

This unit assumes that learners will have prior experience of programming using block-based construction (eg Scratch) and understand the concepts of sequence and repetition. The National Centre for Computing Education key stage 1 units focus on floor robots and ScratchJr, however, experience of other languages or environments may also be useful.

See the learning graph for this unit for more information about progression.

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

Science – Electricity (Year 4)

- construct a simple series electrical circuit, identifying and naming its basic parts, including cells, wires, bulbs, switches and buzzers

Assessment

Formative assessment

Assessment opportunities are detailed in each lesson plan. The learning objective and success criteria are introduced in the slide deck 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

See the assessment rubric to support summative assessment for this unit.

Resources

The unit has been designed to make use of the components provided in the Crumble starter kit, which are as follows:

- 1 Crumble controller
- 12 crocodile leads
- 2 Sparkles (A Sparkle is an RGB LED — red, green, blue light-emitting diode. The D connector allows the Crumble to use an electronic signal to control the Sparkle. The signal sets the colour and brightness of the LED.)
- 1 push switch suitable for Crumble
- 1 light sensor suitable for Crumble
- 1 buzzer suitable for Crumble
- 1 micro USB cable
- 1 switched battery box suitable for Crumble

Unless stated otherwise in the individual lesson plan, learners will need access to these resources (preferably one kit per pair) in each lesson. In Lessons 2, 5, and 6, learners will also need to use motors (this is also indicated in the individual lesson plans).

Learners will also need access to devices capable of running the Crumble software. This is currently available for Microsoft Windows (XP SP3 or newer) and macOS (10.6 and above). Download the software from redfernelectronics.co.uk/crumble-software.

Subject knowledge

This unit focuses on physical computing that allows learners to control real-life events through the construction of programs. When learners undertake physical computing, they write programs that control real-world objects, like LEDs and motors, using a computer. The tangible effect of seeing the commands that they entered into a computer being carried out on a physical item, rather than on screen, can be highly motivational for learners. Physical computing also offers the opportunity to take a more project-based approach to learning, and allows learners to make choices about the purpose, design, and program of their product.

Throughout this unit, there are opportunities to demonstrate a concept within the Crumble programming software or play a video. Pedagogically, it is more beneficial to demonstrate the concepts to learners, as it allows for easier questioning and understanding. We recommend that you use the videos to see what to demonstrate, then show learners with a live demonstration, however, videos are provided on the slides if you wish to use them instead.

For this unit, you will need experience of constructing programs using the Crumble programming software (see the 'Resources' section at the end of this document). It uses the same drag-and-drop style as Scratch. You will need to write programs that turn LEDs (Sparkles) on and off, change LED colours, spin motors, use push switches as inputs, and combine a number of these peripherals. Additionally, you will be connecting the Crumble controller with battery packs, Sparkles, motors, and push switches. For further support on using Crumbles, see the Crumble 'Getting Started' guide at redfernelectronics.co.uk/crumble-getting-started.

Levels of abstraction
When programming, there are four levels that can help describe a project (known as 'levels of abstraction'). Research suggests that this structure can support learners in understanding how to create a program and how it works:

- Task — this is what is needed
- Design — this is what it should do
- Code — this is how it is done
- Running the code — this is what it does

Spending time at the 'Task' and 'Design' levels before engaging in writing code aids learners in assessing the 'do-ability' of their programs and reduces a learner's cognitive load during programming. Learners will move between the different levels throughout the unit, and this is highlighted within each lesson plan.

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. These 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 need 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 ‘score’ in a game reaches a certain value to when a key on a keyboard has been pressed.

Conditions

Conditions are statements that need to be met for a set of actions to be carried out. They can be used in algorithms and programs to control the flow of actions. When a condition is met, it is referred to as ‘true’ and when it is not met, it is referred to as ‘false’. You will need to be able to identify and use conditions in algorithms in the form of statements to both start and stop sets of action. Additionally, you will need to understand that conditions can be used in loops, and when they are, that the set of actions in the loop will be carried out repeatedly until the condition is true, for example, ‘until button A is pressed’.

Selection

When designing programs, there are often points where a decision must be made. BBC Bitesize defines selection as:

> Selection – a decision within a computer program when the program decides to move on based on the results of an event (source: BBC Bitesize)

These decisions are known as selection, and are implemented in programming using if statements. Selection is used to control the flow of actions in algorithms and programs by checking if a condition (see above) has been met. If it has been met, the identified actions will be carried out. When selection is used in programs, loops (see above) have to be used to instruct the device to check the condition repeatedly. Without using loops, the condition would only be checked once. In the Crumble programming software, selection is implemented through the if... then... command block.

In addition to the above, you will also need to understand that programs are an implementation of an algorithm, and that when the program does not produce the required output, the algorithm should be debugged. This should then be implemented in the program.

Enhance your subject knowledge to teach this unit through the following training opportunities:

Online training courses

- Raspberry Pi Foundation online training courses

Face-to-face courses

- National Centre for Computing Education face-to-face training courses