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 it to control components (including output devices — LEDs and motors). Learners will be introduced to conditions as a means of controlling the flow of actions in a program. 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 demonstrate 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, learners 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, your learners will become familiar with the Crumble controller and the programming environment used to control it. Learners will connect a Sparkle to a Crumble and then program the	To control a simple circuit connected to a computer

	Crumble to make the Sparkle flash different colour patterns. Learners will also use infinite loops, which were introduced to the learners in the previous school year.	• I can create a simple circuit and connect it to a microcontroller • I can program a microcontroller to make an LED switch on • I can explain what an infinite loop does
2 Combining output components	In this lesson, learners will connect a Sparkle and a motor to the Crumble controller. Learners will design sequences of actions for these components. 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 component to a microcontroller • I can use a count-controlled loop to control outputs • I can design sequences that use count-controlled loops
3 Controlling with conditions	In this lesson, learners will be introduced to conditions, and how they can be used in programs to control their flow. They will identify conditions in statements, stating if they are true or false. 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.	To explain that a loop can stop when a condition is met • I can explain that a condition is either true or false • I can design a conditional 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 then represent conditions and actions using the 'if…then…' structure. Learners will create algorithms that include selection. They will use their algorithms to guide their program writing. Learners will see that infinite repetition is required to repeatedly check if a condition has been met.	To explain 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 and selection when designing a project to meet the requirements of a given task. To support their understanding, learners will identify how selection might be used in real-world situations, then they will consider how they can apply this knowledge to design their project. Learners will produce design sketches to show how their model will be made and how they will connect the microcontroller to its components.	To design a physical project that includes selection • I can identify a real-world example of a condition starting an action • I can describe what my project will do • I can create a detailed drawing of my project
6 Writing and testing algorithms	In this final lesson of the unit, learners will develop Crumble programs to control the model of a fairground ride they built in Lesson 5. First, learners will identify how they are going to use selection before writing an algorithm to meet the requirements of the given task. They will then implement their algorithms as code. Learners will run their programs to identify any bugs, and then return to the code or	To create a program that controls a physical computing project • I can write an algorithm that describes what my model will do • I can use selection to produce an intended outcome • I can test and debug my project

	algorithm to debug it where necessary. Finally, to conclude the unit, learners will evaluate their designs.	

Progression

This unit assumes that learners will have prior experience of programming using a block-based language (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

Design and Technology (Key stage 2)

Design

- Generate, develop, model, and communicate their ideas through discussion, annotated sketches, cross-sectional and exploded diagrams, prototypes, pattern pieces, and computer-aided design

Make

- Select from and use a wider range of tools and equipment to perform practical tasks [for example, cutting, shaping, joining, and finishing], accurately
- Select from and use a wider range of materials and components, including construction materials, textiles, and ingredients, according to their functional properties and aesthetic qualities

Evaluate

- Evaluate their ideas and products against their own design criteria and consider the views of others to improve their work

Technical knowledge

- Understand and use electrical systems in their products [for example, series circuits incorporating switches, bulbs, buzzers, and motors]
- Apply their understanding of computing to program, monitor, and control their products

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.

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 a 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 geared motors and wheels (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), macOS (10.6 and above), and ChromeOS on Chromebooks. Download the software from redfernelectronics.co.uk/crumble-software.

Subject knowledge

This unit focuses on physical computing, which allows learners to control real-life projects 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 show a screencast animation on a slide. 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 animations to see what to demonstrate, then show learners with a live demonstration, however, animations 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 components. Additionally, you will connect the Crumble controller to 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 physical computing project or standalone program and how it works:

- Task — this is what is needed
- Design — this is what it should do
- Build — 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 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.

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

Selection is "part of a program where, if a condition is met, then a set of commands are run".

Selection is implemented in programming using **if...then...** 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) often have to be used to instruct the device to check the condition repeatedly. Without using loops, the condition would only be checked once. It's important to understand that each loop cycle will complete before the condition is checked

again. 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