Seamer and Irton CP School – Computing (H.Griffiths)

Topic – Sequencing Sounds	Year 3 Spring 1	Strand – Programming

Prior Learning

In the following KS1 units
Year 1 – Moving a robot – Spring 1
Year 2 – Robot Algorithms – Summer 2
and
Year 1 – Introduction to Animation – Spring 1
Year 2 – An Introduction to quizzes – Summer 2
Pupils have had experience of floor robots and ScratchJr. Learners will have begun to understand that sequences of commands have an outcome and will have made predictions based on their learning. They used and modified designs to create their own quiz questions in ScratchJr and realise these designs in ScratchJr used blocks of code. Finally, learners evaluated their work and make improvements.

Key Knowledge I need to understand

I need to understand that:

Programming is when we make a set of instructions for computers to follow.

Scratch is a program that we can use in order to code our own stories and animations.

We use algorithms (a set of instructions to perform a task) to sequence movements, actions and sounds in order to program effective animations.

This unit explores the concept of sequencing in programming through Scratch. It begins with an introduction to the programming environment, which will be new to most learners. They will be introduced to a selection of motion, sound, and event blocks which they will use to create their own programs, featuring sequences. The final project is to make a representation of a piano. The unit is paced to focus on all aspects of sequences, and make sure that knowledge is built in a structured manner. Learners also apply stages of program design through this unit.

How I will show what I have learned

To explore a new programming environment	- I can identify the objects in a Scratch project (sprites, backdrops) - I can explain that objects in Scratch have attributes (linked to) - I can recognise that commands in Scratch are represented as blocks
To identify that commands have an outcome	- I can identify that each sprite is controlled by the commands I choose - I can choose a word which describes an on-screen action for my design - I can create a program following a design
To explain that a program has a start	- I can start a program in different ways - I can create a sequence of connected commands - I can explain that the objects in my project will respond exactly to the code
To recognise that a sequence of commands can have an order	- I can explain what a sequence is - I can combine sound commands - I can order notes into a sequence
To change the appearance of my project	- I can build a sequence of commands - I can decide the actions for each sprite in a program - I can make design choices for my artwork
To create a project from a task description	- I can identify and name the objects I will need for a project - I can relate a task description to a design - I can implement my algorithm as code

What vocabulary I need to know

Scratch, programming, blocks, commands, code, sprite, costume, stage, backdrop, sprites, motion, turn, point in direction, go to, glide, sequence, event, task, design, run the code, order, note, chord, design, algorithm, bug, debug

The following Glossary may be useful
https://icompute-uk.com/ewExternalFiles/iCompute-Glossary.pdf

What's next

In **Year 3 – Events and Actions in Programs – Summer 2**, learners will explore the links between events and actions, while consolidating prior learning relating to sequencing. Learners begin by moving a sprite in four directions (up, down, left, and right). They then explore movement within the context of a maze, using design to choose an appropriately sized sprite. This unit also introduces programming extensions, through the use of **Pen** blocks. Learners are given the opportunity to draw lines with sprites and change the size and colour of lines. The unit concludes with learners designing and coding their own maze-tracing program.

Assessment

National Curriculum Computing links

- 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 opportunities are provided throughout each of the lesson plan documents. The learning objectives and success criteria are introduced in the slide decks at the beginning of each lesson and then reviewed at the end.
The school recommends the use of teacher accounts in Scratch to help with assessment throughout this unit. For guidance on setting up teacher accounts, please visit the Scratch website.
(https://scratch.mit.edu/educators/faq)

Summative assessment – the assessment rubric document should be used to assess student's work from lesson 6. The rubric should be completed digitally and stored in individual pupil folders and then used alongside teacher judgement to complete ScholarPack.
https://teachcomputing.org/curriculum/key-stage-2/programming-a-sequence-in-music

Teacher Subject Knowledge

This unit focuses on developing learners' understanding of sequences in a new programming language. It highlights that the order of sequences is important. This unit also develops learners' understanding of design in programming, using the approach outlined below.
When programming, there are four levels which 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 - what is needed
- Design - what it should do
- Code - how it is done
- Running the code - what it does

Spending time at the task and design levels before engaging in code-writing can aid learners in assessing the 'do-ability' of their programs. It also 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.