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

Topic – Events and actions in programs	Year 3 Summer 2	Strand – Programming

Prior Learning

In **Year 3 – Spring 1 - Sequence in Music** learners explored the concept of sequencing in programming through Scratch. It began with an introduction to the programming environment, which was new to most learners. They were introduced to a selection of motion, sound, and event blocks which they used to create their own programs, featuring sequences. The final project was to make a representation of a piano. The unit was paced to focus on all aspects of sequences, and make sure that knowledge was built in a structured manner. Learners also applied stages of program design through this unit.

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.

I can use event and action command blocks in order to make sprites carry out acts when certain prompts take place.

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

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.

How I will show what I have learned

To explain how a sprite moves in an existing project	- I can explain the relationship between an event and an action - I can choose which keys to use for actions and explain my choices - I can identify a way to improve a program
To create a program to move a sprite in four directions	- I can choose a character for my project - I can choose a suitable size for a character in a maze - I can program movement
To adapt a program to a new context	- I can use a programming extension - I can consider the real world when making design choices - I can choose blocks to set up my program
To develop my program by adding features	- I can identify additional features (from a given set of blocks) - I can choose suitable keys to turn on additional features - I can build more sequences of commands to make my design work
To identify and fix bugs in a program	- I can test a program against a given design - I can match a piece of code to an outcome - I can modify a program using a design
To design and create a maze-based challenge	- I can make design choices and justify them - I can implement my design - I can evaluate my project

What vocabulary I need to know

Motion, event, sprite, algorithm, logic, move, resize, Extension block, pen up, set up, pen, design, event, action, debugging, errors, setup, code, test

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

What's next

In **Year 4 – Spring 1 - Repetition in Shapes** learners will create programs by planning, modifying, and testing commands to create shapes and patterns. They will use Logo, a **text-based programming language** as well as the knowledge they have gained throughout all previous programming units.

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 highlighted in each of the lesson plan documents. The learning objective and success criteria will be introduced at the beginning of each lesson and then reviewed at the end. Learners should assess how well they feel they have met the learning objective using the teacher's chosen method.

Summative assessment document included - multiple choice questions. This should be used, alongside teacher judgement, to complete summative assessment on ScholarPackhttps://teachcomputing.org/curriculum/key-stage-2/programming-b-events-and-actions

Teacher Subject Knowledge

The use of teacher and learner accounts in Scratch is recommended, to help with assessment throughout this unit. For guidance on setting up teacher accounts, visit scratch.mit.edu/educators/faq. A teacher account enables you to manage learners' accounts and organise projects into studios. If you are unable to use teacher and learner accounts, work can be saved offline to local devices.

This unit focuses on the links between 'events' and 'actions' in programming, while also developing learners' understanding of sequencing. It highlights that events cause actions, and that the order of those actions can have an impact on the outcome of a program. This unit also further develops learners' understanding of design in programming, using the approach outlined below.
When programming, there are four levels that help to describe the stages of 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 code writing aids learners in assessing the 'do-ability' of their programs and reduces a learner's cognitive load during programming.