ACET Junior Academies'

Scheme of Work for Computing
Scratch KS1/KS2

About this unit:
This scheme of work is designed to build on and develop programming skills using Scratch 3.0 throughout KS2.

Assessment note: it is worth printing and annotating computing work to show understanding of programmes and how goals have been accomplished.
Teaching note: it is worth recapping previous learning / pre-requisite skills as a warmup task before teaching a new skill

Unit structure

Pre-Unit – Programmable Toys (Y1)
Unit 1 – Sprites and backgrounds (Y1)
Units 2 – Programming sprites to respond (Y2)
Unit 3 – Moving and interacting (Y3)
Unit 4 – Adding timers and scoring systems (Y4)
Unit 5 – Creating a game with a purpose (Y5)
Unit 6 – Debugging (Y6)

All units should include a recap of terminology and incorporate previously learned Scratch skills for revision.

Links to previous and future National Curriculum units

design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts (KS2)

use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs (KS2)

use sequence, selection, and repetition in programs; work with variables and various forms of input and output (KS2)

understand what algorithms are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions (KS1)

use logical reasoning to predict the behaviour of simple programs (KS1)

create and debug simple programs (KS1)

Pre-Unit: Programmable Toys (KS1)

Knowledge and concepts	Computing skills:	Assessment criteria:	Curricular links:
Children should be aware of what instructions are	Knowledge that a set of instructions can be given to a toy or programme	I can give instructions to a programmable toy I can use instructions logically to reach a goal	This can work linked or unlinked to a topic. This has an ideal opportunity to be linked to a literacy unit on instructions. Maths – left / right and directions.

Suggested activities:	Resources:	Useful links/ideas:
Using a large area, like the hall, children can give their partner (blindfolded) directions. Using a programmable toy and a squared mat to reach a goal. It is worth modelling how the toy functions and how it is instructions clearly at the start of the session. If no squared mat is available, then a large piece of paper and marker pens can be used. The goal should be for the toy to move from the start point to an end point. Writing down the instructions will enable the children to see their code and then alter is based on testing. A similar algorithm task can be done in an analogue way for a set of instructions e.g. making a jam sandwich - working in pairs to develop a clear set of instructions with one child acting as a robot that can only perform actions it is clearly instructed to do.	Programmable toy Squared mat or large piece of paper	https://curriculum.code.org/csf-19/coursea/3/ - happy maps could be used as an analogue way to do the same task with recording https://curriculum.code.org/csf-19/courseb/6/ - this is another way that an algorithm could be learned with a focus on repetition - children could develop their own algorithm of actions for another child to act out using the code

Unit 1 – Sprites and backgrounds (KS1)

Links to previous learning	Knowledge and concepts	Computing skills:	Assessment criteria:	Curricular links:
Children should be familiar with basic key computing skills such as opening programmes and using a cursor	Sprites are images that can be given their own instructions. A background is an image that the sprites act in front of.	Opening programmes Adding components Moving components	I can add a sprite I can add a background I can change the size of a sprite I can change the location of a sprite	This session can and should be linked to your current topic learning (e.g. creating a farmyard scene or a space scene)

Suggested activities:	Resources:	Useful links:
Children should be introduced to what scratch is, what it is used for and how it works by showing creation of a simple project demonstrating clicking together blocks to make them work together. Explain how to do this clearly down to the level of which mouse click, if to hold it etc. This is more of a skills based lesson with a small focus on scratch to introduce the children to the programme. Children should use scratch to create a topic-linked scene with 3-5 sprites with a focus on how to add sprites and backgrounds, change size etc.	Scratch 3.0 (free online) A laptop	https://projects.raspberrypi.org/en/projects/rock-band - this project has instructions needed to generate a scene with interactive sprites – this can easily be tweaked to fit a specific topic

Unit 2: Programming sprites to respond (KS1)				
Links to previous learning	**Knowledge and concepts**	**Computing skills:**	**Assessment criteria:**	**Curricular links:**
Children should be familiar with adding sprites and background to a project.	Scratch creates an algorithm using blocks of code which clip together. There are lots of different types of code blocks that have different functions. Sprites are images that can be given their own instructions.	Opening programmes Creating an algorithm Testing an algorithm Debugging	I can create a sequence of code I can test a sequence of code	This session can and should be linked to your current topic
Suggested activities:		**Resources:**	**Useful links:**	
Revise: what scratch is, why it is used, how to use the blocks of code. Children should use scratch to create a topic-linked scene with 3-5 interactable sprites. These sprites should respond to button presses or key clicks with a sound or colour change.		Scratch 3.0 (free online) A laptop	https://projects.raspberrypi.org/en/projects/rock-band - this project has instructions needed to generate a scene with interactive sprites – this can easily be tweaked to fit a specific topic	

Unit 3: Moving and interacting

Links to previous learning	Knowledge and concepts	Computing skills:	Assessment criteria:	Curricular links:
Children should be familiar with adding sprites and background to a project including how to create a simple algorithm.	A sprite can be given many different algorithms which respond to different inputs. Changing values in a code affects the output in logical and predictable ways.	Creating an algorithm Making predictions about an output Testing an algorithm Debugging	I can create a sequence of code I can make predictions about values in an algorithm I can test a sequence of code	This session can and should be linked to your current topic

Suggested activities:	Resources:	Useful links:
Children should create a topic-linked scene where a sprite can travel in 4 different directions using the arrow keys. This is an excellent time to investigate and make predictions about how the steps number will affect the sprite's movement. The same could be done with changing colours, making sounds, enlarging and shrinking etc. Children could undertake an open-ended investigation to find out different ways to make sprites respond to inputs. EX: Creating a scene with two sprites which can move using different key combinations (arrows for one sprite, WASD for another sprite)	Scratch 3.0 (free online) A laptop	https://scratch.mit.edu/projects/10128431/ orange ball responding to movement inputs for arrow keys (to SWOT up on the code)

<table>
<tr><th colspan="6">Unit 4: Adding timers and scoring systems</th></tr>
<tr><th>Links to previous learning</th><th>Knowledge and concepts</th><th>Computing skills:</th><th>Assessment criteria:</th><th colspan="2">Curricular links:</th></tr>
<tr><td>Children should be familiar with adding sprites and background to a project including how to create a simple algorithm.</td><td>Each sprite can be given a separate algorithm and operate independently.
Cursor movement and clicking can be an input.
Changing values in a code affects the output in logical and predictable ways.
Unique variables can be created to track outputs (score)</td><td>Creating an algorithm
Making predictions about an output
Altering an algorithm to achieve a given purpose (challenge)
Adding a tracked variable (score)
Testing an algorithm
Debugging</td><td>I can create a sequence of code
I can make predictions about values in an algorithm
I can test a sequence of code</td><td colspan="2">This session can and should be linked to your current topic</td></tr>
<tr><th colspan="2">Suggested activities:</th><th>Resources:</th><th colspan="3">Useful links:</th></tr>
<tr><td colspan="2">Children could create a simple game which keeps track of a score. The ghostbusters project idea can be adapted for any topic. This involves 3 sprites which appear and disappear using a randomised time period and respond to being clicked by the cursor to score points.
This lends itself to discussion around what values in the code do and how they can be manipulated to make the game harder or easier (appearing and disappearing more quickly) or that sprite size could also be a factor in this.
EX: give the children an open-ended task of making their version more challenging after discussion</td><td>Scratch 3.0 (free online)
A laptop</td><td colspan="3">https://projects.raspberrypi.org/en/projects/ghostbusters - a simple code project that includes a score and randomised sprite behaviour
https://scratch.mit.edu/projects/10128368/ - a similar concept to ghostbusters
https://projects.raspberrypi.org/en/projects/flappy-parrot - more difficult conceptually but could be given a partial code (requires understanding of X and Y axes)</td></tr>
</table>

<table>
<tr><th colspan="5">Unit 5: Creating a game with a purpose</th></tr>
<tr><th>Links to previous learning</th><th>Knowledge and concepts</th><th>Computing skills:</th><th>Assessment criteria:</th><th>Curricular links:</th></tr>
<tr><td>Children should be aware of how to make sprites move, respond to inputs, create a scoring variable and have the skills to make predictions and educated choices around values in an algorithm based on testing.</td><td>A game could consist of some sort of challenge to make it competitive.

Two different tracked variables can operate at the same time.

A timer is just a variable that increases or decreases by one point every second.</td><td>Creating an algorithm

Making predictions about an output

Altering an algorithm to achieve a given purpose (challenge)

Adding a tracked variable (score)

Testing an algorithm

Debugging</td><td>I can create a sequence of code

I can make predictions about values in an algorithm

I can test a sequence of code

I can debug their own code based on their testing</td><td>This session can and should be linked to your current topic</td></tr>
<tr><th colspan="2">Suggested activities:</th><th>Resources:</th><th colspan="2">Useful links:</th></tr>
<tr><td colspan="2">Children to create a game where sprites have to interact to score points. This could include a timed element. Using knowledge from previous units it would be suggested to make a controlled sprite that responds to arrow key inputs have to reach and touch objects which disappear and reappear randomly, scoring points as it does so.

EX: Children should have the skills now to test and re-evaluate their algorithms and to some extent, debug their code based on this. Challenge them to make the game as difficult as possible.</td><td>Scratch 3.0 (free online)
A laptop</td><td colspan="2">https://projects.raspberrypi.org/en/projects/boat-race - sprite following cursor using a timer to complete a maze – sprite interactions for game over</td></tr>
</table>

<table>
<tr><th colspan="5">Unit 6: Debugging</th></tr>
<tr><th>Links to previous learning</th><th>Knowledge and concepts</th><th>Computing skills:</th><th>Assessment criteria:</th><th>Curricular links:</th></tr>
<tr><td>Children should be aware of almost all of the basics of an algorithm and the effect of values on the effects of a code block.</td><td>A code is not always perfect and often needs testing / debugging.

A code normally always be improved in some way.</td><td>Testing an algorithm

Debugging</td><td>I can alter an algorithm to make it work as intended using knowledge of coding</td><td>This session could, but doesn’t necessarily, lend itself to being topic linked.</td></tr>
<tr><th colspan="2">Suggested activities:</th><th>Resources:</th><th colspan="2">Useful links:</th></tr>
<tr><td colspan="2">Children could be given an altered project where some variables or outputs have been changed. They should be shown the intended output at the start of the session and be challenged with debugging an algorithm so that it achieves the intended purpose e.g. all the paint colours are wrong, the width slider works in the opposite way to intended etc.

This could also be done with any project similar to those created previously in the scheme.

EX: Children could study the code to figure out how it works and begin to expand on the code after it has been properly debugged to make a better end product (e.g. adding a new, sparkly pen option in paintbox)</td><td>Scratch 3.0 (free online)
A laptop</td><td colspan="2">https://projects.raspberrypi.org/en/projects/paint-box - a good project to make simple changes to the code for the children to attempt to debug</td></tr>
</table>