Coding (Unit 1.7) for Year 2 Catch up Medium Term Plan

Note: This MT plan has been used to support catch up in coding units for Year 2 where the Year 1 unit of coding has not been fully taught. The planning follows the Unit 1.7 Coding plans from the Purple Mash scheme of work, but has been amended to meet the Year 2 Learning Ladders Objectives and ensure the NC targets are being met.

In 2024-2025, the Year 1 cohort completed half of this unit as the intended coding unit for Year 1 Coding (Unit 1.7). In 2025-2026, the now current Year 2 will be continuing this catch up unit from lesson 4 before moving on to Unit 2.1 Coding. To ensure learning from this catch up MT plan supports a smooth build up and progression into Coding Unit 2.1, lessons 4- 6 have been further amended.

Year 2. Coding (Unit 1.7) This unit builds on children's understanding that an algorithm is a precise, step by step set of instructions to complete a task (building on knowledge gained in Unit 1.4 and 1.5) and continue to develop the understanding that this is at the core of coding. In this unit, children revisit the knowledge that a program is an algorithm for a computer (as previously learnt in Unit 1.4 and 1.5) and are introduced to the term 'computer programming' and 'coding,' and begin to understand how a programmer uses code to make a computer program. This builds on from experience gained in unit 1.5 where they began to use directional arrows and numbers (rather than whole sentences) to make screen sprites move, and is developed further as children are introduced to code blocks from the 2Code programming app. They explore how these code blocks can be joined together to create commands and further joined to create algorithms, whilst being exposed to vocabulary associated with programming in 2Code such as 'objects, actions, event, commands, design view, code view, run, execute...' Pupils begin to develop skills in reading lines of code and are encouraged to make guesses of what might happen when the code is run. When running a code does not produce the desired outcome (as with the Code Monkey Activities throughout this unit), children build on prior knowledge gained in Unit 1.5 to recognise that the code must be wrong. In these instances, they try to fix their code and are reminded of the term 'debugging' and 'bugs' as introduced in units 1.4 and 1.5 previously. By the end of the unit, children will have designed and programmed their own simple program.

Knowledge covered:

NC	Ladders Objectives
Understand what algorithms are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions.	~~Y1- CS1 I understand that an algorithm is a precise, step-by-step set of instructions to complete a task~~ ~~Y1- CS2 I know that an algorithm written for a computer is called a program~~ Y2-CS1 I can explain an algorithm is a precise, step by step set of instructions to complete a task Y2- CS2 I know I need to be precise when I plan my algorithm so it will work when I make it into a code
To create and debug simple code	~~**Y1- CS3 (KPI) I can design a simple program to control the movements of a screen sprite**~~ ~~Y1-CS4 I can say that if something does not work how it should, it is because my code is incorrect~~ ~~Y1-CS5 I can try and fix my code if it isn't working properly~~ Y2 – CS3 I can design a simple program that achieves a purpose Y2- CS4 I can find and correct some errors in my program (debugging)
Use logical reasoning to predict the behaviour of simple programs.	~~Y1 – CS6 I can make good guesses of what is going to happen in a program. For example, where the sprite might go~~ **Y2 – CS5 (KPI) I can say what will happen in a program** Y2 – CS6 I can spot something in a program that has an action or effect (does something)
Use technology purposefully to create, organise, store, manipulate and retrieve digital content	~~**Y1- CS3 (KPI) I can design a simple program to control the movements of a screen sprite**~~ (Y2-CS3) I can design a simple program that achieves a purpose

Coding (Unit 1.7) for Year 2 Catch up Medium Term Plan

Medium Term Lesson Plan:

Lesson 1 Unplugged (completed in Year 1 24-25)	L.I: I understand that computer programs work by following instructions called code *Carry out pre-assessment learning* (Follow unit plan lesson 1)	S.C: Understand how symbols can represent instructions Arrange code blocks to create an algorithm	<u>Key Question (from KWL)</u> *Prior knowledge* What is a program? *Prior knowledge* What is an algorithm? (Y2-CS1) (Y2-CS2)
Lesson 2 (completed in Year 1 24-25)	L.I: I can design a simple program that completes a task (Y2-CS3) (Follow unit plan lesson 2)	S.C: Use object and action blocks (Y2-CS6) Debug coding (Y2-CS4)	<u>Key Question (from KWL)</u> *Prior knowledge* What is a program? *Prior knowledge* What is an algorithm? (Y2-CS1) (Y2-CS2) Why might a program not work correctly? (Y2- CS2) (Y2-CS4) What should you do if a program does not work correctly? (Y2-CS4)
Lesson 3 (completed in Year 1 24-25)	L.I: I can design a simple program that completes a task (Y2-CS3) (Follow unit plan - lesson 3)	S.C: Use event, object and action code blocks (Y2-CS6) Say what will happen in a program (Y2-CS5 KPI)	What do you think will happen in this program? (Y2-CS5 KPI)
Lesson 4 (stages 1-2 completed in Year 1 24-25) **Amended** **Lesson 1 for Year 2 2025-2026**	L.I: I can design a simple program that completes a task (Y2-CS3) (follow unit plan – lesson 4) **LI: I understand that computer programs work by following algorithms written in code** ***Carry out pre-assessment learning***	SC: Find and correct errors (debug) (Y2-CS4) Say what will happen in a program (Y2-CS5 KPI) **SC: Follow an algorithm** **Vocabulary reminder**	<u>Key Question (from KWL)</u> Can you predict where the sprite will go? (Y2-CS5 KPI) *Prior knowledge* Why might a sprite not move where you wanted it to? (Y2-CS2) *Prior knowledge* What should you do if a sprite does not move where you wanted it to? (Y2-CS4) (Y2-CS6) Can you spot an event? Action? **<u>Key Question (from KWL)</u>** ***Prior knowledge* What is an algorithm? (Y2-CS1) (Y2-CS2)** ***Prior knowledge* What is a program?**

	(recap learning from Lego Builders 1.4 when following an algorithm on Paint Projects. **Introduce new learning of the steps to make a computer program - from lesson 1 unit 2.1** **Remind of the coding language we used in Coding Unit 1.7 and remind of object and action blocks)**		
Lesson 5 **Amended** **Lesson 2 for Year 2 2025-2026**	L.I: I can edit a scene by adding, deleting and moving objects **LI and SC remain the same.** (**Follow lesson plan 5 from Unit 1.7 coding but amend to use the 'steps to make a computer program' from lesson 1 Unit 2.1 as introduced last week, and link this lesson to step 3 – creating the design on the computer.)**	SC: Understand what backgrounds and objects are Use properties table to scale Use bin to delete Drag mouse to move	
Lesson 6 *Carry out end of unit assessment* **Amended** **Lesson 3 for Year 2 2025-2026**	L.I: I can design a simple program that completes a task (Y2-CS3) **LI: I can design a simple program that achieves a purpose (Y2-CS3)** **(Follow lesson plan 6 from Unit 1.7 coding but amend to use the 'steps**	SC: Be precise (Y2-CS2) Debug (Y2- CS4) Say what will happen in the program (Y2-CS5 KPI) **S.C: Look at my plan for the design and function of a program** **Write the algorithm (Y2-CS2)**	Why might a program not work correctly? (Y2-CS2; Y2-CS4) What should you do if a program does not work correctly? (Y2-CS4) Can you predict where the sprite will go? (Y2-CS5 KPI) (Y2-CS6) Can you use an event? Action? **<u>Key Question (from KWL)</u>** ***Prior knowledge* What is a program?** ***Prior knowledge* What is an algorithm? (Y2-CS1) (Y2-CS2)**

	to make a computer program' from lesson 1 Unit 2.1 as introduced 2 weeks ago, and link this lesson to steps 2 – writing the algorithm, and step 4 – adding code so the computer can follow the algorithm to make the program work **Recap use of the event block – when clicked - from Coding Unit 1.7)**	**Use object, action and event blocks to code the algorithm** **Debug code**	**Why might a program not work correctly? (Y2- CS2) (Y2-CS4)** **What should you do if a program does not work correctly? (Y2-CS4)**