



Computing Scheme of Work

Unit 4.1 –

**Coding - New
From 2023**



Contents

Introduction	4
PRIMM	4
Levels of Scaffolded coding tasks.....	5
Year 4 – Medium-Term Plan.....	6
Lesson 1 – Design, Code, Test and Debug.....	7
Aims	7
Success Criteria	7
Resources	7
Preparation	7
Activities	8
Lesson 2 – IF Statements	10
Aims	10
Success Criteria	10
Resources	10
Preparation	10
Activities	10
Lesson 3 – Co-ordinates.....	13
Aims	13
Success criteria	13
Resources	13
Preparation	13
Activities	13
Lesson 4 – Repeat Until and IF/ ELSE Statements.....	17
Aims	17
Success criteria	17
Resources	17
Preparation	17
Activities	18
Lesson 5 – Number Variables	22
Aims	22
Success criteria	22
Resources	22
Preparation	22
Activities	23

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: @2simplesoftware



Lesson 6 – Design and Make a Game with a Score	26
Aims	26
Success criteria	26
Resources	26
Preparation	26
Activities	27
Appendix 1: Display Boards	30
Assessment Guidance	34



Introduction

This unit consists of six lessons that assume children have followed the Coding Scheme of Work in Years 1 to 3. If most of the class have not, use the Coding Catch-Up unit instead of this unit.

Key coding vocabulary is shown in **bold** within the lesson plans, use these new words in context to help children understand the meaning of them and start to build up, their vocabulary of coding words.

The Gibbon guided activities provide further practice of the concepts that the children will be learning and can be used as extension activities. More able children can be encouraged to explore other things that they can change in their programs and experiment with the options available, such as **variables** and **If statements**.

Children will often be able to solve their own problems when they get stuck, either by reading through their code again or by asking their peers; this models the way that coding work is really done. More able children can be encouraged to support their peers, if necessary, helping them to understand but without doing the work for them.

Note: To force links within this document to open in a new tab, right-click on the link then select 'Open link in new tab'.

PRIMM

The coding lessons in these units are structured around the **PRIMM** approach. The whole approach may take place during a lesson or series of lessons.

Predict... what this code will do

Run... the code to check your prediction

Investigate... trace through the code to see if you were correct

Modify... the code to add detail, change **actions**/outcome

Make... a new program that uses the same ideas in a different way. Get creative!

Often lessons will start by looking at existing code, asking the children to 'read' it and make **Predictions** to what they think will happen when the code is run. You'll then **Run** the code and give them time to discuss what happens with them and relate it back to their predictions. You'll spend time with them **Investigating** the code, looking at how different parts work and helping them to understand how. Once children understand how the code works, they will be encouraged to **Modify** it - changing and adding code and re-running the program to view the impact of their changes. And once confident with this, they are encouraged to try and **make** their own program from scratch.

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: [@2simplesoftware](https://twitter.com/2simplesoftware)



Levels of Scaffolded coding tasks

You can support children's learning and understanding by using different degrees of scaffolding when teaching children to code. The lessons provide many of these levels of scaffolding within them and using Free Code Chimp, Gibbon and Gorilla enables children to clarify their thinking and practise their skills. These are not progressive levels; children can benefit from all the levels of activities at whatever coding skill level they are:

Scaffolding	Task type	Examples of how to provide these opportunities
Most scaffolded  Least scaffolded	Copying code	By giving children examples of code to copy.
	Targeted tasks	• Read and understand code • Remix code to achieve a particular outcome. • Debugging. • Use printed code snippets so that children can't run the code but must read it. • Include unplugged activities and 'explaining' tasks e.g. 'how do variables work?'
	Shared coding	• Sharing Challenge activities as a class or group on the whiteboard. • Complete guided activity challenges as a class. • After completing challenges; share methods to create a class version of the challenge. • Free coding as a class
	Guided exploration	• Exploring a limited repertoire of commands • Remixing code • Explore commands in free code before being taught what they do. • Use questioning to support children's learning. • PRIMM approach; Predict – Run – Investigate – Modify - Make
	Project design and code	Projects (imitate, innovate, invent, remix) There are different ways to scaffold learning in projects. This process can be applied to programming projects; • Using example projects e.g. the Guided 2Code activities. • Completing the challenges at the end of each guided activity. • Free code✓ • Create a project that imitates a high-quality exemplar. • Remixing ideas. • Independently creating a brand-new program.
	Tinkering	Use Free code Gorilla to access the full suite of 2Code objects and commands ✓ Use Free code to play and explore freely.

Adapted from work by Jane Waite - Computing at Schools <https://www.computingschool.org.uk/>

In Literacy, some teachers follow a progression that scaffolds learning to write texts. At first children read lots of examples of the genre of text they are going to create. Then they create an **imitation** of an example text. Next, they create a variation of the text (**remix and innovate**). Finally, they get to **inventing** a brand-new version.

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: @2simplesoftware



Year 4 – Medium-Term Plan

Lesson	Title	Aims (Objectives)	Success Criteria
1	Design, Code, Test and Debug	- To review coding vocabulary and knowledge. - To create a simple computer program.	- Children can explore different object types in 2Code. - Children can use a background and objects to create a scene. - Children can plan an algorithm for their scene and use 2Code to program it.
2	IF Statements	- To begin to understand selection in computer programming. - To understand how an IF statement works.	- Children can create a program that includes an IF statement. - Children can interpret a flowchart that depicts an IF statement.
3	Co-ordinates	- To understand how to use co-ordinates in computer programming. - To understand how an IF statement works.	- Children can make use of the X and Y attributes (properties) of objects in their coding. - Children can create a program that includes an IF statement.
4	Repeat Until and IF/ELSE Statements	- To understand the Repeat until command. - To begin to understand selection in computer programming. - To understand how an IF/ELSE statement works.	- Children can read code that includes repeat until and IF/ ELSE and explain how it works. - Children can create a program that includes an IF/ ELSE statement. - Children can interpret a flowchart that depicts an IF/ ELSE statement.
5	Number Variables	- To understand what a variable is in programming. - To use a number variable.	- Children can explain what a variable is in programming. - Children can create and use variables when programming.
6	Making a Playable Game	- To review vocabulary and concepts learnt in Year 4 Coding. - To create a playable game.	- Children can read code that includes repeat until and IF/ ELSE and explain how it works. - Children can create a program that includes and IF/ ELSE statement. - Children can interpret a flowchart that depicts an IF/ ELSE statement.

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: [@2simplesoftware](https://twitter.com/2simplesoftware)





Lesson 1 – Design, Code, Test and Debug

Aims

- To review coding vocabulary and knowledge.
- To create a simple computer program.

Success Criteria

- Children can explore different **object** types in 2Code.
- Children can use a background and **objects** to create a scene.
- Children can plan an algorithm for their scene and use 2Code to program it.

Resources

Unless otherwise stated, all resources can be found on the [main unit 4.1 page](#). From here, click on the icon to set a resource as a 2Do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- [Coding Vocabulary Quiz Y4](#)
- [Free Code Gibbon](#) (this is found on the [main 2Code page](#)).
- [Storyboard Planner](#) OR individual whiteboards.
- (Optional) [Vocabulary flash cards](#). The Teacher flash cards have been created so you can print them on A4 paper, cut them to size, fold them in half and glue them together. You can display and use these throughout coding lessons to support use of vocabulary.

Preparation

- Set [Free Code Gibbon](#) as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:

Year:	Y4	▼
Subject:	Computing	▼
Strand:	Computer Science	▼
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.		✓

- Print copies of the [Storyboard Planner](#) for children to use if you are using it (see main plan).

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: [@2simplesoftware](https://twitter.com/2simplesoftware)



Activities

Introduction	Display slide 2 and outline the lesson aims. Display slide 3 and outline the success criteria.
Vocabulary	Display slide 4. Use the Coding Vocabulary Quiz Y4 as a class. It is set up so that you attempt all questions and then click the  button to check the answers. Click 'OK' to see which are correct and incorrect: Run through the answers to the questions together. You could use the vocabulary cards to find the answers and display in the classroom or use slide 5 which has definitions. Slide 5 can be used to review previous vocabulary. The use of this vocabulary is recapped during the lesson. The vocabulary is repeated at the end of the lesson where it can be used to review new vocabulary.
Free Code Gibbon	Display slide 6. Put Free Code Gibbon on the board. Review how to add objects in 2Code by going into Design View. Talk about creating a scene using a background and some objects. Then run through the design steps shown on the slide ending with 'Running' the program and testing the code. Display slide 7. Stop the program, click on 'Design' and look in more detail at the object attributes. Ask children to predict what would happen if you edited the animal object attributes to change the speed or allow off screen. Look at the attributes for the button (click on it to display them). Ask the children to help you to make the car move when you click on the button: Change the text on the button to e.g., 'Car Go' and the name of it to e.g., 'CarButton' (it is one object so the name can only be one word – no spaces). Add a click event that makes the car move at a set speed when CarButton is clicked on (a speed between 3 and 6ish is sensible, try the children's suggestions and correct if the speed is too fast).



	Display slide 8. Notice the car goes up – you didn’t have up, down, left or right action options like you did with the animal object. Ask children to suggest how you could make the car go along to the right? Go back to design view and look through the attributes of a vehicle – notice and adjust the angle to 90 degrees. Run the program and test the code. Stop the program and return to Design View, discuss how you’ve seen that different object types have different attributes and different actions available when you use their code blocks. Show children how to delete an object (click on it, then click on the bin).
	Display slide 9. Exit Design View using the ‘See Code’ button and look at the different code blocks available – inputs, outputs, timers etc. Ask the children which ones they recognise and to explain what they might do – it doesn’t matter if they don’t know them all yet, they’ll be learning more in this unit!
Activity: Create a Computer Program	Use slide 10 to Explain to children that today they are going to explore Design view in Free Code Gibbon and make their own designs by adding background and objects. Ask them to log into Purple Mash and open Free Code Gibbon from their 2Dos, then work through the following: Set a background. - Experiment with adding different object types and exploring their attributes and actions. - Use a whiteboard or Storyboard Planner to plan what will happen in their program. - Use code to implement their plan - running, testing and debugging as they go.
How did you get on?	Display slide 11. Ask children to save their program, then share great examples with the class, discussing the code that has been used to make them work. Emphasise the importance of the design, code, test and debug process. What challenges did they come across?
Vocabulary Overview	Slide 12 can be used to review lesson vocabulary. Click on the words to reveal the definitions.
Review Success Criteria	Review the success criteria from slide 3. Children could rate how well they achieved this using a show of hands



Lesson 2 – IF Statements

Aims

- To begin to understand **selection** in computer programming.
- To understand how an **IF statement** works.

Success Criteria

- Children can create a program that includes an **IF statement**.
- Children can interpret a flowchart that depicts an **IF statement**.

Resources

Unless otherwise stated, all resources can be found on the [main unit 4.1 page](#). From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- [‘Is It Raining?’ 2Code Example](#).
- [‘Rain IF’ flowchart example](#).
- [‘Lost’ 2Code Example](#).
- [Storyboard planner](#).
- [Selection video](#).

Preparation

- Set [‘Lost’ 2Code Example](#) as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:

Year:	Y4	▼
Subject:	Computing	▼
Strand:	Computer Science	▼
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.		✓

- Print copies of the [Storyboard Planner](#).

Activities

Introduction	Display slide 2 and outline the lesson aims. Display slide 3 and outline the success criteria.
Vocabulary	Slide 4 can be used to review previous vocabulary. The use of this vocabulary is recapped during the lesson.

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: [@2simplesoftware](https://twitter.com/2simplesoftware)



	The vocabulary is repeated at the end of the lesson where it can be used to review new vocabulary. Use slide 5 to introduce the term – ‘Selection’ in relation to computer programming. Reveal the slide.
IF Statement	Display slide 6. Say to the children ‘IF my class is quiet for 30 seconds, then I will [insert action/ activity here!!]’ Start a timer and then check IF statement is true. If it is, carry out stated action/ activity. In pairs, get children to write an IF statement on their boards, then check if it’s true and run the action if it is, or not if it isn’t. Discuss as a class: When tested, were any not true? Explain that in code we can use IF statements to help our programs work – for example, IF the countdown has reached 0 the game is over, or IF the score equals 10 the ‘amazing’ sound plays.
Selection Video	Display slide 7. Play Selection video to children (Video should play from slide).
Is It Raining?	Display slide 8. Display Is It Raining 2Code activity – show how the chart in the video looks in a program – look at the design together; two people under some rain clouds and a hidden umbrella (you can hide objects at the start using the attributes table). Talk through the code – it starts with a prompt for input. If the user notices the rain clouds and puts ‘yes’ into the input, the IF statement runs and shows the umbrella.
Lost	Display slide 9. Open Lost from your 2Dos by clicking on ‘Preview’. Look at the design together and discover that there is a background and 2 objects: Click on ‘See Code’ and see if children can ‘read’ the code and predict what will happen when the program is run. Run the program twice, putting in different inputs to see what happens. You could ask children to help you draw a flowchart for this program. Delete the code and see if the children can help you put it back in again – you may need to emphasise the difference between alert and prompt for input. Ask children – what could happen after they get to the sea? Click on ‘Design’ and remind children how to change the backgrounds and objects – remind them to change the name in the attributes table if they change the object so it matches what it is.



Activity: Lost	Display slide 10. Explain to the children that they are going to create their own 'Lost' program which should include a timer and an IF statement. They will start with a background and two objects but they can add more if they wish – but be careful not to get distracted by adding too many. Give children the Storyboard Planner– tell them to sketch inside each box and make notes including timings. You could challenge them to draw the flowchart for the IF statement (or one IF statement if they have more than one) on the back of their plan. Once children have finished their designs, they have a go at making them by going to their 2Dos and starting 'Lost'.
How did you get on?	Review children's work together against the lesson aims – this could be done by sharing some good examples from the 2Dos folder.
Vocabulary Overview	Slide 12 can be used to review lesson vocabulary. Click on the words to reveal the definitions.
Review Success Criteria	Review the success criteria from slide 3 . Children could rate how well they achieved this using a show of hands.



Lesson 3 – Co-ordinates

Aims

- To understand how to use **co-ordinates** in computer programming.
- To understand how an **IF statement** works.

Success criteria

- Children can make use of the X and Y **attributes** of **objects** in their coding.
- Children can create a program that includes an **IF statement**.

Resources

Unless otherwise stated, all resources can be found on the [main unit 4.1 page](#). From here, click on the icon to set a resource as a 2Do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- [Knights Castle flowchart](#).
- [Guard the Castle \(Gibbon\)](#).
- Have printed storyboard templates available for program design.
- [Football Goal](#) 2Code activity.

Preparation

- Set [Guard the Castle \(Gibbon\)](#) as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:

Year:	Y4	▼
Subject:	Computing	▼
Strand:	Computer Science	▼
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.		✓

- Set [Football Goal](#) as a 2Do (if planning to include extension).

Activities

Introduction	Display slide 2 and outline the lesson aims. Display slide 3 and outline the success criteria.
Vocabulary	Slide 4 can be used to review previous vocabulary. The use of this vocabulary is recapped during the lesson.

Need more support? Contact us:

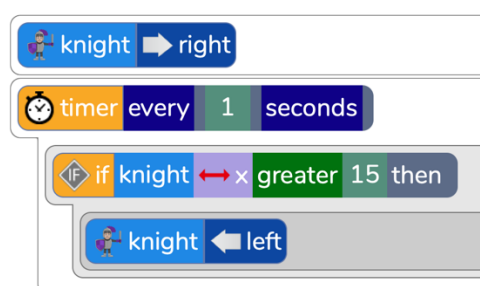
Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: @2simplesoftware



	Use slide 5 to introduce the new vocabulary the children will be learning today, co-ordinates .
Co-ordinates in 2Code	Display slide 6. Reveal the questions and then Open Free Code Gibbon. Go into Design View and click on the grid button in the bottom left. This makes a grid visible. Drag in a vehicle and look at the attribute table for it. You will see it has an X and Y position with a little icon showing which is which. Drag the vehicle to a different position and you will see that the attributes change. Work out where 0,0 is and the maximum X and Y by dragging the vehicle around. Give children X and Y positions and see whether they can make a good estimate as to where the vehicle should go. Relate this to the context of co-ordinates and graphs – notice that 0,0 on a computer is top left. Click on the background button (bottom left) and, in the background attributes table, change the grid size to a different size. Note that an objects X and Y values actually remain the same, but the position of an object on screen is altered due to the change in the grid size. NB: If you change the grid size after you have set up the screen design, it can mess things up so do this before you start coding. Briefly review how to make a character respond to a user's input on the keyboard.
Guard the Castle (Gibbon)	Display slide 7. Open the guided lesson Guard the Castle from the Gibbon activities and do stage 1 together. In stage 2, you must create a timer which checks the X position of the knight every second; if the knight's position is greater than 15, he should change direction. Enter the following code (left), run it, and notice how the code executes (right - it highlights orange when it executes):



Can children spot a problem? – The **IF statement executes** and the computer checks the knight's X position when the program is run, not every second – the **IF statement** needs to **nest** within the **timer** to trigger the computer to check the knight's X position every second:



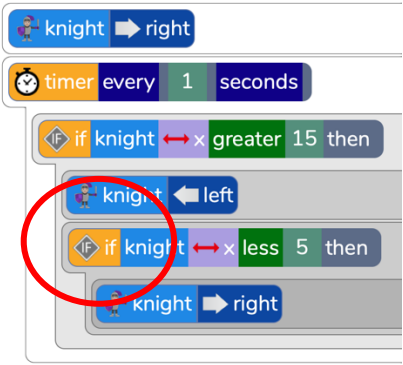
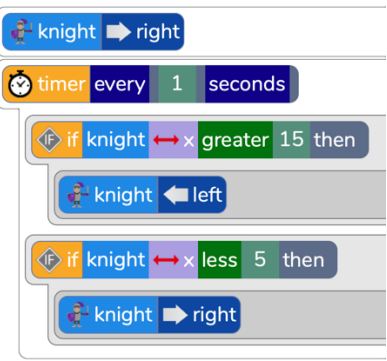
Run the program with the corrected code and notice with children when the code **executes** and what effect it has.

Activity: Guard the Castle

Display **slide 8**. Ask children to log into Purple Mash and open [Guard the Castle](#) from their 2Dos area, then work through stages 1-4.

At stage 3, children commonly add the second **IF statement** inside the first **IF statement** so it looks like the following:



	 If this happens, run their code with them and notice when the statements execute. Fix the problem together, the code should look like this (the second IF statement nested within the timer, not within the first IF statement – so both IF statements are triggered by the timer – every second): 
Flowchart: Guard the Castle	Display slide 9. Open the Knights Castle Flowchart for this activity and read it together, relating it to what they have done in the lesson. They could then either do the challenge activity (last stage) or the extension or both depending on what time you have.
Extension: Football Goal	Display slide 10. Football Goal Challenge. Can children program the goalie to defend the goal using an if statement and co-ordinates? Can they add code for the football so a player can shoot at the goal and if the football collides with the goalie it is saved? They could try to draw a flow chart on a piece of paper that they could use to explain their code.
Vocabulary Overview	Slide 11 can be used to review lesson vocabulary. Click on the words to reveal the definitions.
Review Success Criteria	Review the success criteria from slide 3. Children could rate how well they achieved this using a show of hands.

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: [@2simplesoftware](#)



Lesson 4 – Repeat Until and IF/ ELSE Statements

Aims

- To understand the **repeat until** command.
- To begin to understand **selection** in computer programming.
- To understand how an **IF/ ELSE statement** works.

Success Criteria

- Children can interpret a flowchart that depicts an **IF/ ELSE statement**.
- Children can read code that includes **Repeat Until** and **IF/ ELSE** and explain how it works.
- Children can create a program that includes an **IF/ ELSE statement**.

Resources

Unless otherwise stated, all resources can be found on the [main unit 4.1 page](#). From here, click on the icon to set a resource as a 2Do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- [Is It Raining 2Code Example](#).
- [Is It Raining IF Flowchart](#).
- [Is it Raining IF ELSE Flowchart](#).
- [Reginald Rocket 2Code Example](#).
- [Reginald Rocket IF ELSE Flowchart](#).
- [Free Code Gibbon](#) (this is found on the [main 2Code page](#)).
- [Storyboard Planner](#).

Preparation

- Set [Free Code Gibbon](#) as a 2Do.
- Set [Reginald Rocket 2Code Example](#) as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:

Year:	Y4	▼
Subject:	Computing	▼
Strand:	Computer Science	▼
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.		✓

- Print copies of the [Storyboard Planner](#) for children to use if you are using it (see step 7)

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: [@2simplesoftware](https://twitter.com/2simplesoftware)





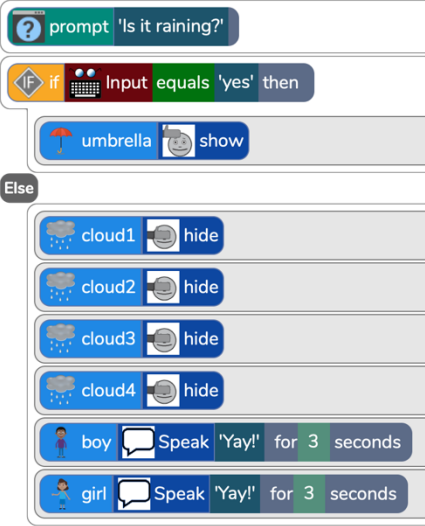
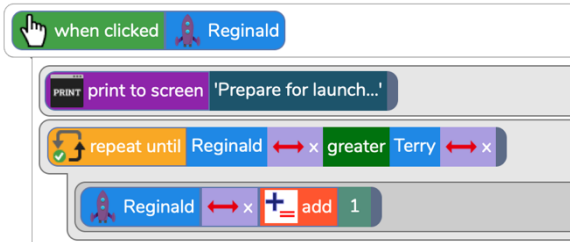
Activities

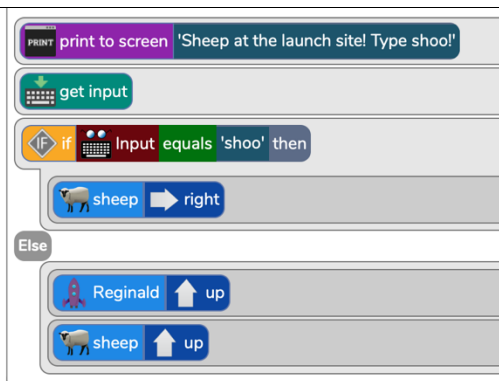
Introduction	Display slide 2 and outline the lesson aims. Display slide 3 and outline the success criteria.
Vocabulary	Slide 4 can be used to review previous vocabulary. The use of this vocabulary is recapped during the lesson.
IF	Start this lesson by returning to the Is It Raining 2Code example from lesson 2. Display the design view in one tab, and the Is It Raining IF Flowchart in another (or display slide 5). <pre> graph TD Start([Start]) --> Decision{Is it raining?} Decision -- Yes --> ShowUmbrella[Show umbrella] ShowUmbrella --> End([End]) Decision -- No --> End </pre> Ask the children to remind you what selection is and explain how an IF statement works. Explain that today we are going to start by looking at how to program something to happen if the condition is not met e.g. program something to happen in our scene if it is not raining.
IF/ ELSE	Display slide 6. Say to the children 'IF my class is quiet for 30 seconds, then I will [insert action/ activity here!!] Start a timer and then check IF statement is true. If it is, carry out stated action/ activity. In pairs, ask one child to write an IF statement on their small whiteboard, then the other to check if it's true and run the action if it is, or not if it isn't. Discuss as a class: When tested, were any not true? Explain that in code we can use IF statements to help our programs work – for example, IF the countdown has reached 0 the game is over, or IF the score equals 10 the 'amazing' sound plays. Show children the Is It Raining IF ELSE flowchart: <pre> graph TD Start([Start]) --> Decision{Is it raining?} Decision -- Yes --> ShowUmbrella[Show umbrella] ShowUmbrella --> End([End]) Decision -- No --> HideClouds[Hide clouds] HideClouds --> PeopleSay[People say "Yay!"] PeopleSay --> End </pre>

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: [@2simplesoftware](https://twitter.com/2simplesoftware)



	Talk it through with the children and then go back to the Is It Raining 2Code activity and ask them to help you add code to program it (ask them to look at the code blocks and see if they can pick out IF/ ELSE as a sensible one to use, if not direct them to it). The code should look like this:  Run the program a couple of times, once typing 'yes' in the prompt for input box, once typing 'no'. Notice how the code executes and what happens each time.
Reginald Rocket	Display slide 7. Load Reginald Rocket 2Code example. Look at the design first, and then the code. Look at the first part of the code and pick out the repeat until command.  Can children 'read' the code to see what this command is doing? When the user clicks on Reginald (the rocket), a message is printed to the screen – Prepare for Launch – then Reginald will move right (adding 1 to the X attribute) this repeats until the X is greater than the X position of Terry. Ask children what/who is Terry? - Terry is the launch pad (you can work this out by looking for the object called Terry in Design View). And what is the purpose of this first section of code? - This first section of code moves the rocket onto the launch pad. Now look at the next part of the code:



What do children think will happen when this program is run?

When the children click on Reginald, he will move along to the launchpad.

If the input is 'shoo' the sheep will run out of the way, if not (else) Reginald will take off with the sheep!

Display **slide 8**. Show children the [Reginald If Else Flowchart](#) – if they look on the 'statement is true' side Reginald doesn't take off. What code would they need to add for Reginald to take off 3 seconds after the sheep was shooed away? Can we add a blast off sound?



Activity: Create a Program

Display **slide 9**. Ask children to make a written plan with the following task specification:

Task: Create a short program that uses **Repeat Until** and **IF/ ELSE** commands.

Ask children to use the [Storyboard Planner](#) to plan their program.

Challenge them to plan how their code will work using a **flowchart** on the back of their storyboard.

Remind children not to be too ambitious, and to think about the knowledge they have when making their plans, so they know they will be able to create them in 2Code.

Children could have [Free Code Gibbon](#) in front of them as they plan so they can look at available **backgrounds** and **objects**.



	Children open the Free Code Gibbon task from their 2Dos area and start to make their plan into a working computer program.
How did you get on?	Display slide 10 . Review children's work together against the lesson aims – this could be done by sharing some good examples from the 2Dos folder.
Vocabulary Overview	Slide 11 can be used to review lesson vocabulary. Click on the words to reveal the definitions.
Review Success Criteria	Review the success criteria from slide 3 . Children could rate how well they achieved this using a show of hands.



Lesson 5 – Number Variables

Aims

- To understand what a **variable** is in programming.
- To use a number **variable**.

Success Criteria

- Children can explain what a **variable** is in programming.
- Children can create and use **variables** when programming.

Resources

Unless otherwise stated, all resources can be found on the [main unit 4.1 page](#). From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- [Free Code Gibbon](#) This is found on the [main 2Code page](#).
- [Genie](#).
- [Night and Day](#).
- [Variable Game Cards](#).
- 2 copies of [Number Cards 0-23](#).
- 2 boxes (transparent, if possible, e.g. ice-cream or take away containers).

Preparation

- Set [Genie](#) as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:

Year:	Y4	▼
Subject:	Computing	▼
Strand:	Computer Science	▼

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.	✓

- Set [Night and Day](#) as a 2Do (if using extension)
- Print and cut up 2 copies of [number cards 0-23](#).
- Print and stick 4 [Variable Game Cards](#) under 4 children's chairs.

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: [@2simplesoftware](https://twitter.com/2simplesoftware)



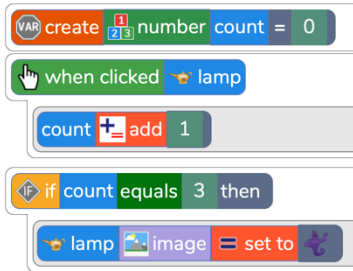
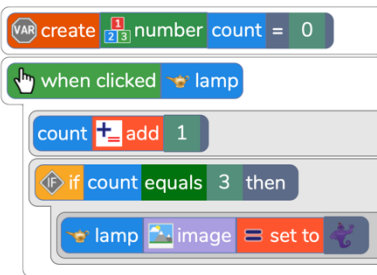
Activities

Introduction	Display slide 2 and outline the lesson aims. Display slide 3 and outline the success criteria.
Vocabulary	Slide 4 can be used to review previous vocabulary. The use of this vocabulary is recapped during the lesson. Display slide 5. Explain that today we will be working with variables. Go through the definition.
	Display slide 6. Use this slide to help you explain what a variable is. On this slide each box is a variable. They both have names - the first variable is called 'team1score' and the second variable is called 'team2score'. The variable values are determined by how well (or not) each team does in a quiz. Display slide 7. Put two actual boxes on a table so that all the children can see them, and label them 'team1score' and 'team2score' as in the slide. On the board, write 'team1score=' and 'team2score='. Read the IF/ ELSE statements that will have an impact on these variables – 'If a Team 1 answer is correct, the value of team1score will increase by 1, else the value of team1score will stay the same'. 'If a Team 2 answer is correct, the value of team2score will increase by 1, else the value of team2score will stay the same'. Tell children that they will be taking part in a class quiz that will have an impact on these variables. Split the class into Team 1 and Team 2. Display the quiz on slide 8 and play it: Team 1 play the first question. When they answer, check if it is correct and then react in the way the IF/ ELSE statement directs – use a number card to show the variable value in the team1score variable box (put the number card in the box) and add a value to 'team1score =' on the board. Ask team 2 the next question – repeat as above until all the questions have been answered – changing the number cards in the boxes, and the values of the relevant variables each time an answer is given. Emphasise that the variable value is <i>replaced</i> with the new value each time – a variable holds only 1 value.



Look under your chair	Display slide 9. Ask children to look under their chairs – four should find the Variable Game Cards you stuck underneath them. The cards say something like the following: Bonus points: add 4 to your team's score. Bonus points: double your team's score. Disaster card: subtract 2 from your team's score. Disaster card: halve your team's score. Who won? Discuss how the answers impacted the value of the variables and emphasise the importance of naming variables sensibly.
2Code Genie	Use slide 10 to open the Genie activity. Stage 1: Complete together – when you create the variable point out that there are different types of variables and in this lesson, we are choosing 'number'. Creating the variable is a bit like making the box, our box in the classroom was named score, this one will be named 'count' as it keeps a count.  When you click on Run to run the program point out the variable watch box:  Explain that you can't see this variable in the scene as it's part of the code. Display slide 11. Ask children to open Genie from their 2Dos and try and complete it independently. Remind them that they can click on the instruction to return to the video or unlock hints if they get stuck. Notes to support children with task. Stage 2: When children have added in the click event, they will need to add the  code block and then select count because they need to change the count variable when the lamp is clicked on. When they click on Run to run the code encourage them to look at the variable watch box and notice how the variable changes each time they click on the lamp.



	Stage 3: At stage 3 the children may make the following error -  Incorrect Code - the IF statement runs and checks to see if the count=3 as soon as you press Run (so only once), and it needs to be triggered to check if the count=3 every time the variable changes value.  Correct Code - the IF statement is checked every time the lamp is clicked on (the click event triggers the IF statement to run), so if/when it's true, the lamp can turn into a genie!
Extension: Night and Day (Gibbon)	Display slide 12 . Children have a go at working through Night and Day Gibbon that you have set as a 2Do.
Vocabulary Overview	Slide 13 can be used to review lesson vocabulary. Click on the words to reveal the definitions.
Review Success Criteria	Display slide 14 . Ask children to 'hand in' tasks with an honest review of how they got on. Review the lesson together against the success criteria.



Lesson 6 – Design and Make a Game with a Score

Aims

- To review vocabulary and concepts learnt in Year 4 Coding.
- To create a playable game.

Success criteria

- Children can use the correct code to make their game work.
- Children can explain how their code makes their game work.

Resources

Unless otherwise stated, all resources can be found on the [main unit 4.1 page](#). From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- [Free Code Gibbon](#) (this is found on the [main 2Code page](#)).
- [Storyboard Planner](#).
- [Turtle Race game](#).

Preparation

- Set [Free Code Gibbon](#) as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:

Year:	Y4	▼
Subject:	Computing	▼
Strand:	Computer Science	▼

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.	✓

- Print copies of the [Storyboard Planner](#) for children to use.
- Create a display board for the class to share their programs to. Details of how to do this are given in [Appendix 1](#)

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: [@2simplesoftware](#)



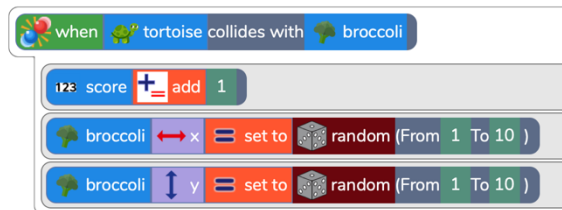
Activities

Introduction	Display slide 2 and outline the lesson aims. Display slide 3 and outline the success criteria.
Vocabulary	Slide 4 can be used to review previous vocabulary. The use of this vocabulary is recapped during the lesson.
Turtle Race	Use slide 5. Display Turtle Race game on the board, look at the design with the children. Click on 'rs' and 'ps' and look across to the attributes table – notice they are number objects with a value set to 0. These can display a variable value. Click on Run at the top to run the program and see that they display as 0 at the start.  Click on the stop button and return to Design view. Click on the food, turtles and button and see how they are names in the attributes table. Ask the children to speculate as to how this game might be played. Display slide 6. Exit the design using the 'See Code' button and look at the code, give the children a chance to 'read' it – you might want to split it into two parts as suggested in slide 6. Give them time with a talking partner to discuss what the code will do. Relate this to the teaching from the previous lesson on variables – what is going to change the values of the 2 number objects? In the previous lesson the value of the variables was altered when the lamp was clicked on. Rather than variables, this program uses number objects, when the turtles collide with the food the value attribute of the object changes. Click on Run to run the code and click on the red and purple race buttons in turn until one turtle has eaten all its food. Which turtle won? The number objects keep a count of how much food the turtles have eaten, like a score.

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: @2simplesoftware



	Give children 2 minutes with their talking partner to discuss how this game could be improved. Ask them to feedback to the class and encourage them to consider if they think these are features, they could add in 2Code.
Improve Turtle Race	Reference slide 7 and 8. Use either children's feedback or slides 7 and 8 to make some changes/ improvements to the game (or both!).
Free Code Gibbon	Use slide 9. Open Free Code Gibbon, click on 'Design' and add a background and objects, use the attributes table to name them appropriately. Add a number variable called 'score'. Add a number variable called 'score'. Last lesson we made created a variable in the code, this lesson we are going to add a number variable in design mode so the variable value (the score) is visible in our design. Your scene might look something like this:  Click on 'See Code' and add code that programs one object to collide with another and a score to increase. Add code so that the object collided into jumps to random X/ Y co-ordinates. Ask the children to predict what this will do. Your code could look something like this:  Ask children to consider how this game might start and finish – how will the user know what to do?
Design and Make YOUR Game!	Use slide 10. Explain to the children that they will be designing and making their own game with a score. Children use the Storyboard Planner to design their game and make notes on how it will work. They might like to be in front of Free Code Gibbon while they do this so they can explore the galleries and see what backgrounds and objects are available to them. Once children have finished their designs, they have a go at creating them in Free Code Gibbon. Remind them of the design – code – test – debug process.



	These are some of the things they have learned so far – they might want to consider using some of them: • Number variables to keep score • Selection – IF and IF/ELSE statements • Co-ordinates (using X and Y) • Timers – after and every (could be handy to set a time limit) • Repeat and Repeat until • Alert boxes (these could be good for instructions)
How did you get on?	Display slide 11 . Share children's work 2Displayboard (see Appendix 1) and play a few of the games together to share children's work and celebrate achievements.
Vocabulary Overview	Slide 12 can be used to review lesson vocabulary. Click on the words to reveal the definitions.
Review Success Criteria	Display slide 13 . Ask children to 'hand in' tasks with an honest review of how they got on. Review the lesson together against the success criteria.

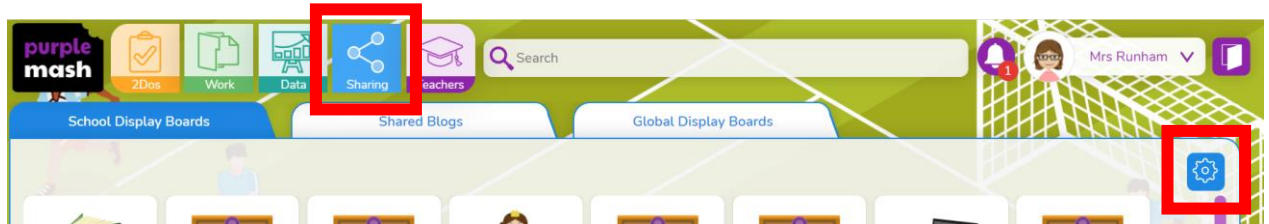


Appendix 1: Display Boards

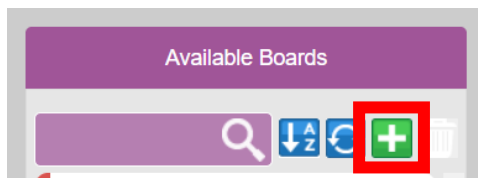
Create the Display Board

Creating the display board is usually something you do before the lesson.

1. Click on the 'Sharing' **button** to find the Display Board tab, and then click on the settings cog:



2. Click on the '+' in the menu on the left:



3. Edit the settings (don't forget to add an icon by clicking on the ) , select the class and then click on 'Save':



Name Coding Lesson 5

Description Coding Lesson 5

Icon

Hide Info ☒ Hide pupil name
☐ Hide class name

Access ☐ Only staff can push
☐ Visible to public
☐ Archived (hidden but still accessible with link)

Who Can See
☐ All School
 > Classes
 > Groups

Buttons: View display board, Save, Cancel

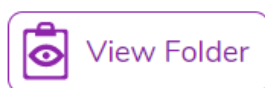
4. Exit Display Board settings:



The Display Board will now be visible under the 'Sharing' **button** to all those you've selected to have access to it.

Adding work to a Display Board:

1. Click on 'View Folder' from the 2Do:

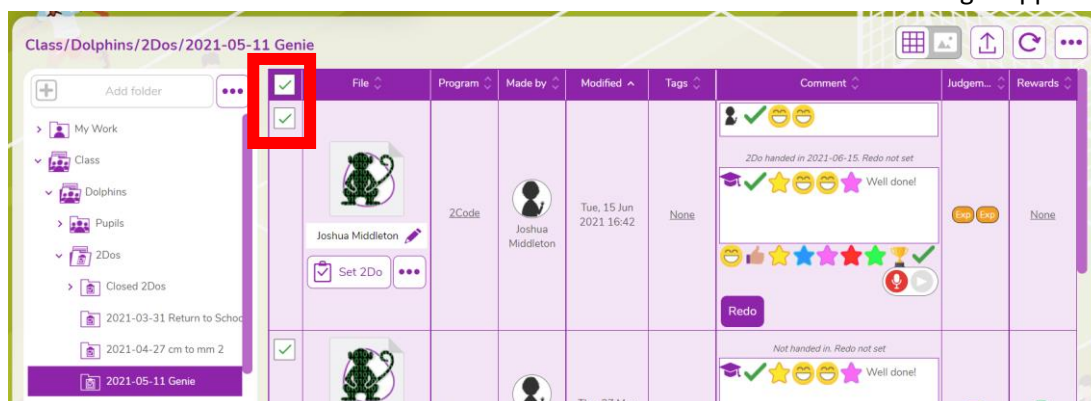


(or navigate to the work you want to share in the Work area).

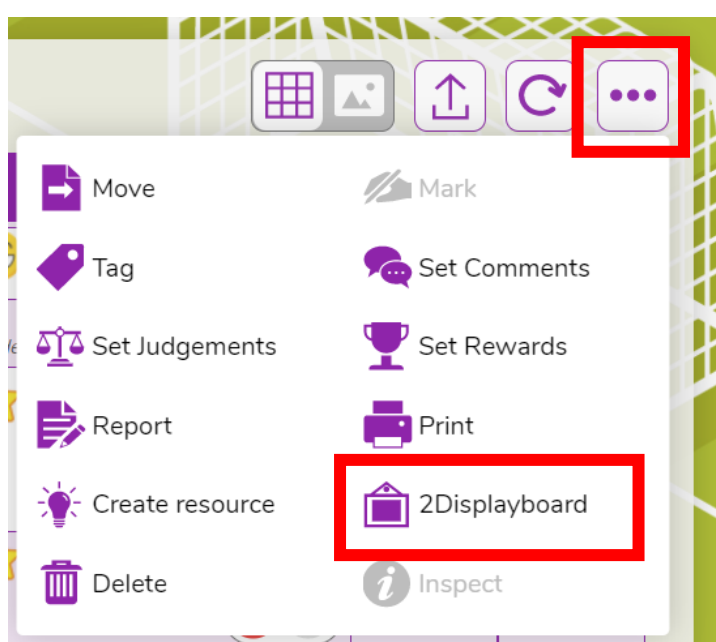
2. Select the files you want to add to the display board or select all files in the folder using the tick at the top.

Need more support? Contact us:

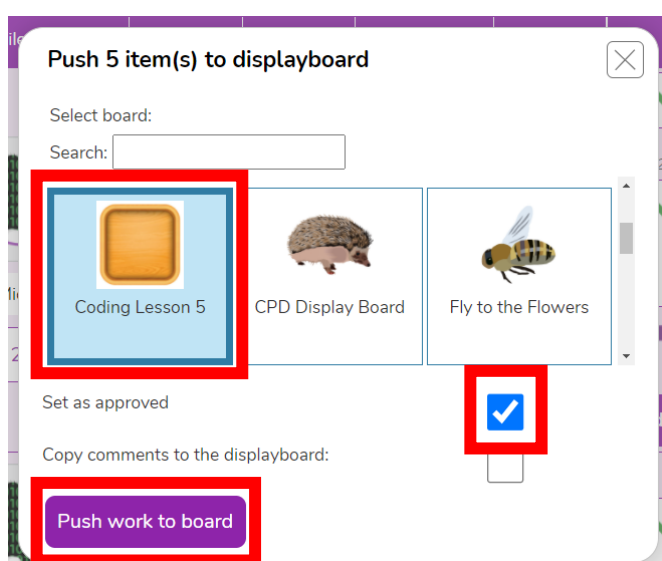
Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: @2simplesoftware



3. Click on the '...' menu **button** top right, then click on '2Displayboard':



4. Choose the display board you've made for the work, tick 'Set as approved' and 'Push work to board':



Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: @2simplesoftware



- Click on 'Sharing' **button** and then on the display board, you should see the work you've added. It can be deleted by clicking on 'Edit' at the top of the board, then clicking work and then delete. This will remove it from the display board, it won't delete it from Purple Mash.

Deleting or Archiving a Display Board:

When you've finished the lesson, you can return to the Displayboard settings and either delete it or archive it to stop it appearing under the 'Sharing' **button**.

- Click on 'Sharing' and then on the settings cog.

- Tick 'Archive', and then 'Save' OR 'Delete'

Clicking on 'Delete' will delete the display board but the work will still be available in the work area, it doesn't not delete the files.

Name: Coding Lesson 5

Description: Coding Lesson 5

Icon: [Icon]

Hide Info: ☒ Hide pupil name, ☐ Hide class name

Access: ☐ Only staff can push, ☒ Archived (hidden but still accessible with link)

View display board [Icon]

Who Can See: ☐ All School, > Classes, > Groups

Save, Cancel, Delete

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: @2simplesoftware



Assessment Guidance

The unit overview for year 4 contains details of national curricula mapped to the Purple Mash Units. The following information is an exemplar of what a child at an expected level would be able to demonstrate when completing this unit with additional exemplars to demonstrate how this would vary for a child with emerging or exceeding achievements.

Assessment Guidance	
Emerging	Children's designs for their programs show that they are thinking of the structure of a simple program in logical, achievable steps (Unit 4.1 Lesson 1). Children can make good attempts to 'read' code and predict what will happen in a program which can help them to correct errors in their code. With support, children can turn a real-life situation into an algorithm for a program that has cause and effect (Unit 4.1 Lesson 2) and use their algorithm to write simple programs using 2Code (Unit 4.1 Lesson 2). Furthermore, they can identify errors within their programs and make logical attempts to fix it (Unit 4.1). Children attempt to introduce selection into their code using simple 'if statements' (Unit 4.1 Lesson 2). Children's use of these structures is experimental; they cannot always predict the outcome accurately or anticipate the structures required when planning their code. They have a developing idea that a variable can be used to store information in a program, in lesson 5 they can follow the examples but might struggle when applying this with their own ideas.
Expected	Children's design shows that they are thinking of the required task and how to accomplish this in code using coding structures for selection and repetition (Unit 4.1 Lessons 1 and 6). Children can identify an error within a program that prevents it following the desired algorithm and then fix it (Unit 4.1), they apply these techniques to their own code to fix bugs. Children understand IF and IF/ ELSE statements for selection and combine these with other coding structures including variables to achieve the effects that they design in their programs (Unit 4.1 Lesson 4). Their design demonstrates their growing understanding of when a coded solution will require repetition e.g. in Lesson 4 'Reginal Rocket' children can see that the position of the rocket is changed repeatedly until it is in line with the rocket launch pad. They can explain the new command 'Repeat Until'. They make use of user input (Unit 4.1 Lesson 2) and outputs such as 'print to screen' (Unit 4.1 Lesson 4) as well as sound and movement of objects. They understand how variables can be used to store information while a program is executing (Unit 4.1 Lesson 5) and make attempts to use and manipulate the value of variables.

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | Twitter: @2simplesoftware



Assessment Guidance

	Children's designs for their programs show that they are thinking of the structure of a simple program in logical, achievable steps with attention to specific events that initiate specific actions (Unit 4.1 Lessons 1 and 6). Children can 'read' others' code and predict what will happen in a program which helps them to correct errors (Unit 4.1). They can also make good attempts to fix their own bugs as their coding becomes more complex (Unit 4.1 Lesson 6). Most children can create programs which accomplish a specific goal utilising a variety of media such as images, sounds and animation effects. (Unit 4.1 Lessons 1 and 6). Children can interpret the flowcharts used to represent IF/ELSE statements (Unit 4.1 Lesson 4) and create their own when planning their programs.
Exceeding	Children's design shows that they are thinking of the required task and how to accomplish this in code using coding structures for selection and repetition and variables (Unit 4.1 Lessons 1, 4 and 5). Children can identify an error within a program that prevents it following the desired algorithm and then fix it (Unit 4.1). Children make intuitive attempts to debug their own programs as they increase in complexity (Unit 4.1 Lesson 6). Children realise the constraints of creating purely sequential programs and intuitively grasp the concepts of selection (Unit 4.1 Lessons 2, 3 and 4), repetition (Unit 4.1 Lesson 4) and variables (Unit 4.1 Lesson 5). Children like to challenge themselves to combine these with other coding structures to achieve the effects that they design in all their programs (Unit 4.1). Their designs are ambitious but logical and achievable. Children's designs for their programs show that they are absorbing new knowledge of coding structures such as IF statements, repetition and variables. Children can 'read' others' code and predict what will happen in a program which helps them to correct errors (Unit 4.1). They can also make good attempts to fix their own bugs as their coding becomes more complex (Unit 4.1 Lesson 6).