purple mash

CRASH COURSE

Computing Scheme of Work

Year 4 Coding Crash course

For children in Year 4 who have not used 2Code previously.

Contents

Year 4 Crash Course – Introduction

The crash course aims to prepare year 4 children for using the Computing Scheme of Work Coding unit 5.1 in year 5.

Differentiation

The Gibbon 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 timers 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.

To enhance children's ability to code and understand the process of coding and design, children should have had as many of the following experiences as possible:

Challenges

When using the guided activities, children should have attempted the challenges at the end of the guided lessons in 2Code and come up with solutions to these either individually or using shared coding as a group or class.

Free coding

Children will benefit from spending some time using:

- Y1-2 Free code Chimp (or Free code scenes)
- Y3-4 Free code Gibbon
- Y5-6 Free code Gorilla

To create their own programs.

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 build up their vocabulary of coding words.

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 **P**redictions to what they think will happen when the code is run. You'll then **R**un the code and give them time to discuss what happens and relate it back to their predictions. You'll spend time with them **I**nvestigating the code, looking at how different parts work and helping them to understand how. Once children have an understanding of how the code works, they will be encouraged to **M**odify 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 **M**ake their own program from scratch.

Levels of Scaffolded coding tasks

You can support children' 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.computingatschool.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.

Year 4 Crash Course – Medium Term Plan

Lesson	Title	Success Criteria
1	Intro to Coding: Objects, Actions and Events	• Children can explain what **coding** is. • Children know that for the computer to make something happen, it needs to follow clear instructions. • Children can create a program using **event**, **object** and **action** code blocks. • Children can explain what **events**, **objects** and **actions** do in a program.
2	Algorithms	• Children can explain that an **algorithm** is a set of instructions. • Children can describe the **algorithms** they created. • Children can explain that for the computer to make something happen, it needs to follow clear instructions.
3	Different Object Types and Buttons	• Children can create a computer program that includes different **object** types. • Children can create a computer program that includes a **button object**. • Children can modify the **properties** of an **object** and a **button** to fit their program design. • Children can explain what a **button** does in their program.
4	Using Timers	• Children can create a program that uses a **timer-after command.** • Children can create a program that uses a **timer-every command.** • Children understand there can be different ways to solve a problem.
5	IF and IF/ELSE Statements	• Children can create a program that includes an **IF** and **IF/ ELSE statement**. • Children can interpret a flowchart that depicts an **IF** and an **IF/ ELSE statement**. • Children can read code that includes **repeat until** and **IF/ ELSE** and explain how it works.
6	Number Variables	• Children can explain what a **variable** is in programming. • Children can create and use **variables** when programming.
7&8	Design and Make an Interactive Scene (Recommended Optional Lessons)	• Children can use the **properties** table to set the **properties** of **objects**. • Children can plan their **scene** and code before they create their program. • Children can confidently make several different things happen in a program.

Lesson 1 - Introduction to Coding: Objects, Actions and Events

Aims

- To understand what **coding** means in computing.
- To use code to make a computer program.
- To understand what **objects**, **actions** and **events** are.
- To use an **event** to control an **object**.

Success Criteria

- Children can explain what **coding** is.
- Children know that for the computer to make something happen, it needs to follow clear instructions.
- Children can create a program using **event**, **object** and **action** code blocks.
- Children can explain what **events, objects** and **actions** do in a program.

Resources

Unless otherwise stated, all resources can be found on the unit main page. From here, they can be set as 2Dos by clicking on the icon. To preview resources linked to here, right-click and '**open in new tab'** so you do not navigate away from this page.

- Code block cards.
- Fun with Fish Activity. This is on the main 2Code page in the Chimp section.
- Optional: Exercise books to be used as 2Code workbooks for recording coding exercises and designs.
- A pot of bubbles and a bubble wand (usually part of the lid!)
- Bubble Coding. This is on the main 2Code page in the Chimp section.
- Example Code

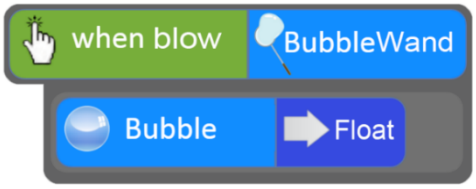
Preparation

- Set Fun with Fish as a 2Do.
- Set Bubble Coding as a 2Do.
- NB: This lesson introduces quite a few fundamental coding concepts in one session. Depending on the previous coding experience and ability levels of your children, this lesson may need to be split over two sessions.

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Introducing Programming	Use **slide 4** to explain to the children that they are going to learn about Computer Programming, which is sometimes also known as **Coding**. Ask them if they know what this is. Discuss briefly that it is the way that computer programmers input instructions into computers to create programs. Can they give any examples of computer programs that they have used?
Activity 1: Teacher is the Programmer	Use **slide 5** to explain to the children that you are now going to be the programmer and they are all the robots. Reveal the instructions on the board as symbols. Get children to 'act' out / follow the instructions you have displayed as symbols – a twirl, a hand next to a toe and a hand next to an ear - the children should twirl, touch their toes then touch their ears.
Activity 2: Using Symbols	Use **slide 6** to display a hand next to an up arrow and see if the children can see that this would be 'hand up'! Ask children to use small whiteboards draw symbols for 'hand down', what about touch nose?
Computers Follow Instructions	Display **slide 7**. Now that children have practised receiving instructions in code represented as symbols, reiterate the introduction using this slide. Explain that a coder writes instructions in code for the computer to follow, this is called the input. These instructions make our programs work, our programs are the output.
Fish in the Sea	Display **slide 8** Fish in the Sea and discuss what you can see – 3 fish in the sea. Ask children what they think those fish could be programmed to do.
Objects, Actions and Algorithm	Display **slide 9.** Explain to the children that the light blue code blocks represent **objects**, and the dark blue code blocks represent **actions.** This set of instructions can be called an **algorithm**.
Command	Display **slide 10** to introduce the term c**ommand**. Explain that a single instruction is called a **command**.
Demonstrating: Fun with Fish	Display **slide 11**. Open Purple Mash and go to 2Dos, click on Preview within the Fun with Fish 2Do to show children the Fun with Fish lesson. Open stage 1 and click on OK to close the instruction screen. Click on 'Design' (in the top right-hand corner) and discuss what can be seen – a fish in the sea. Explain to children that they will use 2Code to program the object (fish) to do an action (move right).
	Display **slide 12**. Click  to go back to the code view. Open the instruction screen by clicking on the  . Watch the video for stage 1.

	Display **slide 13**. Complete stage 1 as a class; emphasise the need to give the computer clear instructions for moving the fish. The available **actions** for the fish **object** pop-up as soon as the fish is dragged into the code window. Show the children what to do if they click on the wrong direction - click on the direction again and select the correct one. Show the children where the Play button is to **run** the code and emphasise that the code has programmed the **object** to do an **action.** Show them how to move to the next stage of the activity or stop the code running to make changes. Complete stage 2 together as a class.
Activity 3: Fun with Fish	Display **slide 14**. Ask children to log in to Purple Mash, go to their 2Dos and click on 'Start' on the Fun with Fish 2Do. Challenge them to complete stages 1 and 2. Ask them to use the **code** blocks to make their Tuna move, and then move onto the next challenge to make the Crab move.
	Use **slide 15**. Load stage 3 and explain that this is a stage where you must fix the code that the monkey has got wrong. We call this **debugging**. Complete this stage as a class (show children that if you want to change an action you can click on it). Children to then complete stage 3 in Purple Mash independently. Review progress together - did they get lots of code monkey stars? The maximum is 5; they lose stars for using hints.
	Display **slide 16**. Look at stage 4 together – this is the challenge stage. All the guided activities have this challenge stage, and this is where children deepen their understanding of the **code** that they have been working on. Take a few suggestions from the class about how to improve the fish tank by adding new **objects** – fish / crabs – add one new **object** then switch to the code screen to notice it then appears as a blue object code piece, show how to program it to move and test it out using the play button. Add one or two more **objects** and show children how to use the **event**. Ask children to complete the challenge stage and then save their work before they exit. Review children work together against the lesson aims – this could be done by sharing some good examples from the 2Dos folder. Did any children try using the '**when clicked event**'? What did that do?
Events	Display **slide 17**. Start by telling children that the **when clicked** code block is an example of what is called an **event** in coding. Ask the children to describe how the **when clicked event** worked in the last stage of Fun with Fish.

Activity 4: Bubbles	Use **slide 18.** This physical activity will help children understand what events are and how they make things happen: Get the bubbles out! Blow bubbles. Ask children: What is the **event**? (What do you do to make something happen?) What are the **objects**? (bubble wand, bubble) What is the **action**? (float) Show the children what the code might look like for blowing bubbles. This command block has three parts: the **event (when blow);** the **objects** – the bubble wand and the bubble and the **action** - float. Show the children what the code could look like for blowing bubbles. 
Activity 5: Other Events	Use **slide 19** to discuss other **event** – **object** – **action** examples children might be familiar with (e.g. push – swing – swing - forward, kick – football - football – roll). Which part of the code is the **event?** Which is the **object?** What **action** does the **object** do? Rearrange the **code** for the football example (this could be done physically using Example Code printed on paper or as a drag into place activity on the board using slide 19). 
Optional extension activity Activity 6: Bubbles Coding	Use **slide 20** to re-iterate how to use click events before they have a go (slide 21). Use the slide to open the Bubbles Coding activity. Look at the available code blocks available and see if children can tell you what they might see when you click to 'Design'. Referring to **event**, **object**, **action**, add code that makes a bubble move up when it is clicked on, add another bubble and make it pop when it is clicked on. Remind children of the play button to **run** the code. Test the **code** you have just added in together, discuss what other **code** they could add to make the other bubbles move.
	Use **slide 21** to set children off independently on Bubbles Coding from their 2Dos.

	Use **slide 22** to review progress together as a class – using terminology **event**, **object**, **action**. Have any of the children noticed or tried to use the **sound** button? Demonstrate how it might be used so the bubble makes a **sound** when it pops. Challenge children to improve their program by adding **sounds** and then saving before exit.
Review Success Criteria	Display **slide 23**. Review the success criteria from slide 3. Children could rate how well they achieved this using a show of hands.

Lesson 2 - Algorithms

Aims

- To understand what an **algorithm** is.
- To create a computer program using an **algorithm**.

Success Criteria

- Children can explain that an **algorithm** is a set of instructions.
- Children can describe the **algorithms** they created.
- Children can explain that for the computer to make something happen, it needs to follow clear instructions.

Resources

Unless otherwise stated, all resources can be found on the main unit 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
- Two identical sets of any construction toy
- Air Traffic Control. This is on the main 2Code page in the Chimp section.
- (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 Air Traffic Control as a 2Do for your class.
- Build two models using two identical sets of any construction toy - one that follows the instructions and one that does not. An example would be using Lego Duplo to build a bird, one that follows these instructions and one that uses the same bricks but not the instructions. Download the instructions for your model so that you can display them on the board.
- (Optional) Print storyboard templates for program design.

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Vocabulary	Use the Coding Vocabulary Quiz on **slide 4** as a class to help review the coding vocabulary learnt in lesson 1. It is set up so that you attempt all questions and then click the hand in button to check the answers. Run

	through the answers to the questions together. You could use the vocabulary cards to find the answers and display in the classroom.
Algorithms	Use **slides 5 and 6** to introduce today's lesson. Read and discuss the definition of an algorithm.
Activity 1: Lego Models	Display **slide 7**. Show the children the two models you built using identical construction toys - without displaying the instructions. Ask them which is correct. The answer you are looking for is that they are both correct; there is no such thing as 'correct' or 'incorrect' when building creatively. They might prefer one over the other, but both are correct.
	Display **slide 8**. Display the instructions on how to build the model and ask the questions on the slide. If you have enough building materials for the class, they could attempt to follow the **algorithm** to create the model themselves.
Making a Computer Program	With **slide 9**, discuss the process of making a computer program. Look at the plan for the airport program. This time you want children to concentrate on implementing this **algorithm**. Discuss what the **objects** in this program are (the planes and the helicopter); what are the **actions**? (the planes go up and right); what **events** are used to make these actions happen? (a click **event** is needed to make the objects move).
Activity 2: Air Traffic Control	Display **slide 10.** Use it to open Air Traffic Control. Watch the video for stage 1 together as a class. Remind children that there are hints if they need them, and that once they've clicked on OK they can get back to the hint by clicking on the instruction at the top. Ask children to come up to the front of the class and use the code blocks to make the plane take off when it is clicked on:  Click on **run**  to test the code and see if it works as they were expecting. When you do this, notice with them that the **code** highlights orange when it **executes** (you may need to click on the stop button and re-run the program to point it out). Explain that if you click on the plane before the **code executes** it won't take off, they need to make sure the code **executes** first. Get children to watch the code and see what happens when you click on the planes, which bit of code **executes** when?
	Use **slide 11** to set children off on completing stages 1 and 2 of Air Traffic Control themselves.
	Use **slide 12** to support looking at stage 3 together as a class. Discuss what **debugging** is – detecting and fixing any errors in the code. Demonstrate

	how you can drag code around to move it into a different place and click on actions to change them. Fix the code together and click on Next Challenge. Set children off to complete the **debugging** stage.
Air Traffic Control Final Stage	Display **slide 13**. Look at the slide together and revise the main elements of the code view.
	Display **slide 14**. Talk about the design of Air Traffic Control Final Stage together and ask children to point out the **background** and **objects** in the design. Refer back to the program on **slide 9**. Compare it to the design for our Airport Program.
Algorithm -› Code	Follow **slide 15**; use it to open the final stage of Air Traffic Control in 2Code and click on 'Exit Design' and ask children to help you add in code for the first step of the algorithm (click event to enable a plane to take off). Click on the play button and run the program. Watch what happens when the code is running: why does some of the code go orange? It is when that bit of code executes.
Activity 2: Air Traffic Control	Display **slide 16**. Ask children to return to 2Code and continue working through Air Traffic Control until they reach the final stage, then challenge them to start adding the code to make the first 3 steps of the **algorithm** work.
Saving Your Code	Use **slide 17** to remind children how to save their work and discuss why it is important to save their coding regularly so that they have a working version to go back to.
Air Traffic Control	When the majority of the class have programmed the **click events**, draw their attention back to **slide 18** and work together to program the last step of the **algorithm** - to make a **sound** play if the helicopter crashes into the yellow plane. When you have talked through the questions on the slide, as an extension ask: could we make a different sound play if the helicopter collides with the purple plane? (You would need two **collision detection** events – one for each plane)
Extension: Air Traffic Control	Use **slide 19** to challenge children to see if they can add **code** for the **collision detection** and complete making the program work. For the optional final step of the **algorithm** children will need to add 2 **actions** to their **collision detection** event. Children can now get creative by adding other planes and runways onto their design and program them. Note: Although you can add more landing strips into the design, you would not normally program them to do actions!
How Did You Get On?	Display **slide 20**. Ask children to work with a partner to read through each other's code and **predict** what will happen when they **run** the program.

Review Success Criteria	Display **slide 21**. Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

Lesson 3 – Different Object Types and Buttons

Aims

- To understand that different **objects** have different properties.
- To create a program using a given **design**.
- To understand the function of **buttons** in a program.

Success Criteria

- Children can create a computer program that includes different **object** types.
- Children can create a computer program that includes a **button object**.
- Children can modify the **properties** of an **object** and a **button** to fit their program design.
- Children can explain what a **button** does in their program.

Resources

Unless otherwise stated, all resources can be found on the main unit 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 Chimp (this is found on the main 2Code page).
- Snail Race. This is on the main 2Code page in the Chimp section.
- Turtle and Character.
- Road Scene.

Preparation

- Set Free Code Chimp as a 2Do.
- Set Road Scene as a 2Do for less confident children.

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Objects and Actions	Display **slide 4** and introduce today's topic of objects and actions.
Snail Race	Display **slide 5.** Open Snail Race and work through stages 1-3 together as a class. Focus on the **actions** available for the snail **object** in stage 1 – have they seen these before? Up until now children have been programming **objects** to move left, right, up, down and stop – but this program works differently. Discuss

	with them how it is different – snails are a different type of **object** to ones they have used before and have different options for **actions**.
Different Actions	Look at the scene on **slide 6**. Ask children to predict what will happen when the program is run. Run the program Turtle and Character, interact with it and see what happens.
Designing a Scene	Talk through **slide 7** and then open Free Code Chimp in front of the class. Follow the instructions on the slide to set the scene.
Choosing Objects	Talk through **slide 8.** Return to your design in Free Code Chimp and look at the **object** types to choose from on the left. Add a turtle and 3 other **objects** that would move (tell children that in this part of the lesson we are using any object type apart from the button – we are going to look at the button object type later).
Changing Objects	Display **slide 9**. Talk through how to change the objects and the size of the objects. Recap how to move the objects around. This is the first-time children have used Free Code Chimp in coding lessons so spend a bit of time in 2Code browsing the clipart galleries – pointing out the categories and search option.
Activity 1: Create the Scene	Use **slide 10** to challenge children to create a **scene** like this by setting the background and adding **objects** - they could choose different clipart so they all have different **objects** on their scenes. (You could set Road Scene as a 2Do for less confident children so they just have to add objects to the scene rather than create it).
Making Objects Move	Once the majority of children have made their scene, draw their attention back to the board and click on 'Exit design' to start adding some code. Talk through **slide 11**, thinking about the **events** and the **when key**. The **when key** is an event command. It makes code run when you press the specified key on the keyboard. In the example on **slide 11,** the **when key** event will run when you press the up arrow on the keyboard.
Actions for Objects	Use **slide 12** to re-cap how to program objects to do actions. Point out that the actions for a turtle are different to the other ones. When adding code for the turtle, in this lesson we are going to program it just to move forwards. (Programming a turtle to turn involves some understanding of degrees of a turn – e.g. a quarter turn = 90 degrees, a half turn = 180 degrees). In 2Code open the design you created earlier in the lesson and, with the children, add code to program some of the objects to move. Ask them to predict what will happen, then run the code.
Adding to Your Code	Display **slide 13**. Challenge the children to add their own code to their programs. To give them more support you could return to your program in 2Code, delete the code you had in there and ask children to help you use timers to make the **objects** start at different times. (They will learn more about timers in the next lesson, so you may wish to leave it out for now.) Could they also add some food **objects** in to be eaten along the way? (This will involve using **collision detection**!).

	NB: If children are coding on tablets, the **when key event** is not available. Instead, they could try using **when clicked** or **when swiped events**.
Adding Buttons to Your Code	Display **slide 14** to introduce what a button is and what it does.
Activity 2: Add a Button	Display **slide 15**. In 2Code, add a button to your scene.
Button Properties	Display **slide 16** and look at the button properties. Name the button and set the text for the button. You can also change the **text size**, **text colour** and **background** colour of the button.
Coding Your Button	Display **slide 17**. Look at the code blocks and ask children to help you add code to make one of your objects move when the button is clicked on.
Activity 3: Program Your Scene	Display **slide 18**. Set the children back to their own designs to add code to make their **objects** move – challenge them to try using different **events** and add in **collision detection** when a key is pressed. They could also add in more **buttons** to make **objects** move. **NB:** If children are coding on tablets, the **when key event** is not available. Instead, they could try using **when clicked** or **when swiped events**.
How Did You Get On?	Display **slide 19** and ask children to save their designs. Share great examples with the class, discussing the code that has been used to make them work.
Review Success Criteria	Display **slide 20**. Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

Lesson 4 – Using Timers

Aims

- To understand that there are different types of **timers**.
- To be able to select the right type of **timer** for a purpose.

Success Criteria

- Children can create a program that uses a **timer-after command.**
- Children can create a program that uses a **timer-every command.**
- Children understand there can be different ways to solve a problem.

Resources

Unless otherwise stated, all resources can be found on the main unit 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.

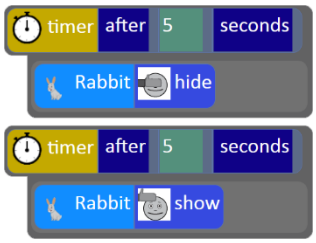
- Magician. This is on the main 2Code page in the Chimp section.
- Night and Day. This is on the main 2Code page in the Chimp section.
- Tick Tock Challenge. This is on the main 2Code page in the Chimp section.

Preparation

- Set Magician as a 2Do.
- Set Night and Day as a 2Do.
- Set Tick Tock Challenge as a 2Do.

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Vocabulary	Display **slides 4** and **5.** Use **slides 4 and 5** to discuss the key vocabulary: **timer**.
	Use **slide 6** to introduce the key vocabulary: **sequence**.
Instructions With Delays	Display **slide 7.** Look at the flowchart together and then ask the children to follow the instructions it gives. Ask children to draw their own version on an individual whiteboard or piece of paper. Ask children to swap their flowchart with a partner and have a go at following each other's instructions.
Magician	Work through **slide 8** and watch the video for stage 1 of the Magician guided lesson.

Magician - Stage 1	Display **slide 9.** Use the slide to talk through stage 1. This is a bit like using an **event** code block – it sets a **timer** and after the specified time the **object** (rabbit) will hide. Open Magician and work through the first stage as a whole class. Watch the videos and remind them that they can unlock a hint if they get stuck. Make mistakes as you add the code and get the children to help you **debug** and fix the problems.
Magician - Stage 2	Display **slide 10.** Work through stage 2 together. Add code incorrectly first (see below), then **test** it and ask the children to help you **debug:**  This code doesn't work because when you **run** the program both **timers** start at the same time (if you click stop and play again you could notice they both highlight orange at the same time) and the code to 'hide' *and* 'show' the rabbit **executes** at the same time – after 5 seconds. Point out that the **timer** for the rabbit to 'show' needs to start **AFTER** the rabbit has hidden. You need to add the second **timer** *inside* the first **timer** OR work out that the rabbit 'shows' 10 seconds after the start (5 + 5) and alter the second **timer** to reflect that, so either of the solutions shown on the slide would work.
Activity 1: Magician	Display **slide 11**. Ask children to start Magician from their 2Dos and work through stages 1-4 independently.
Activity 2: Night and Day	Display **slide 12.** Ask children to start the Night and Day activity from their 2Dos and see if they can complete it. This works in a very similar way to Magician.
Activity 3: Tick Tock Challenge	Display **slide 13.** Once they have completed Night and Day and they have recapped using **timer-after**, tell children that there is another kind of **timer**, and they are going to learn about it by working through the Tick Tock Challenge. Set them to start and complete this challenge from their 2Dos. Review how they have got on – what have they learnt? Ask children: What is the difference between **timer-after** and a **timer-every?**
Activity 4: Extension	Display **slide 14.** Ask children to look at the scene and read the code, then **predict** what would happen when the code is **run**. Discuss with children how they could use a **timer-every** command to develop this program.
Review Success Criteria	Display **slide 15.** Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

Lesson 5 – IF and IF/ ELSE Statements

Aims

- To begin to understand **selection** in computer programming.
- To understand how **IF** and **IF/ELSE statements** works.

Success Criteria

- Children can create a program that includes an **IF** and **IF/ ELSE statement**.
- Children can interpret a flowchart that depicts an **IF** and an **IF/ ELSE statement**.
- Children can read code that includes an **IF** and an **IF/ ELSE** and explain how they work.

Resources

Unless otherwise stated, all resources can be found on the main unit 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.

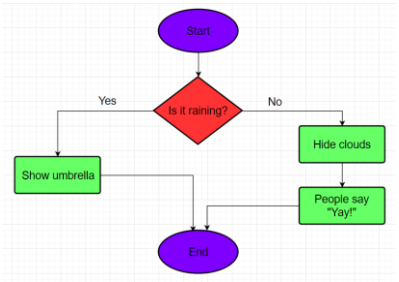
- Is It Raining? 2Code activity.
- 'Rain IF' flowchart example.
- 'Lost' 2Code Example.
- Selection video.
- Is it Raining? IF ELSE Flowchart.
- Free Code Gibbon (this is found on the main 2Code page).
- IF/ELSE Flowchart template.
- Small whiteboards

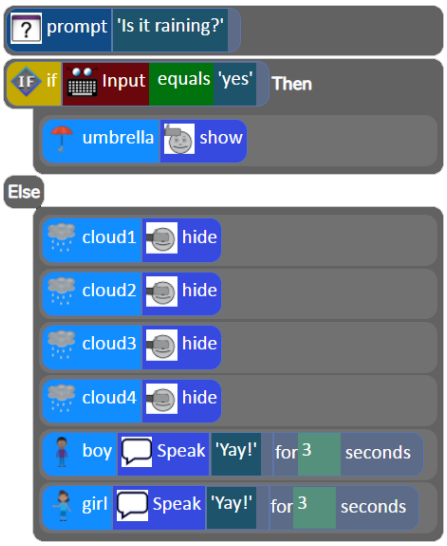
Preparation

- Set 'Lost' 2Code Example as a 2Do.
- Set Free Code Gibbon as a 2Do.
- Print copies of the IF/ELSE Flowchart Template

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Vocabulary	Use **slide 4** to introduce the terms **selection** and **IF statement**. Ask children what they think **selection** means in computer programming.
IF Statement	Display **slide 5**. 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.
Selection Video	Display **slide 6**. Play Selection video to children (Video should play from slide).
Is It Raining? IF Statement	Use **slide 7** to 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 **properties** 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.
Comparing IF and IF/ELSE statements	Use **slide 8** to introduce an **IF/ ELSE statement using the 'Is it Raining?' example:** In this **flowchart,** if the answer to 'Is it raining?' is **yes**, then the umbrella shows. If the answer was ***no,*** then nothing changes. However, we now want 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. There is a suggestion on the slide - if it is not raining, the clouds will disappear and everyone will be happy! Ask the children to discuss how the **flowchart** could be adapted for this example.
IF/ ELSE	Use **slide 9** to show children what the Is it Raining IF ELSE Flowchart could look like: 

	Talk it through with the children, then go back to the Is It Raining? 2Code activity and ask them to help you develop the program to reflect the changes to the flowchart (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 could 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.
Lost	Follow **slide 10:** Look at the design for the 'Lost' program together and notice that there is a background and 2 objects. Look at the code and see if children can 'read' the code and predict what will happen when the program is run. Use the link on the slide to open' Lost' in 2Code. Run the program twice, putting in different inputs to see what happens. 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. Click on 'Design' and remind children how to change the backgrounds and objects – remind them to change the name of an object in the properties table if they change it so the name matches what it is.
Activity 1: Lost IF/ELSE	Use **slide 11** to ask children what could happen if the fish says no, they don't want to go to the sea. What could happen instead? Tell children that in this lesson they will be making their own 'Lost' program in 2Code. Ask them to think about how this example could be

	developed to include an IF/ ELSE statement, then challenge them to draw the flowchart for their program on the IF/ ELSE flowchart template – either using the template on the front or by drawing their own on the back. Ask children to open the 'Lost' code activity from their 2Do area and work independently or with a partner to modify the 'Lost' code – changing the **IF statement** into an **IF/ELSE statement** that matches their planned ideas.
Extension: Your own IF/ELSE program	Display **slide 12**. Explain to the children that they are going to create their own **IF/ ELSE** program. They should start by using the IF/ ELSE flowchart template to draw the flowchart for their planned program. Once children have finished their flow chart, they have a go at making their program by going to their 2Dos and starting Free Code Gibbon.
How did you get on?	Display **slide 13.** Review children's work together against the lesson aims – this could be done by sharing some good examples from the 2Dos folder.
Review Success Criteria	Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

Remember to close your 2Dos when you have finished the lesson.

Lesson 6 – 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 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. This is on the main 2Code page in the Gibbon section.
- Night and Day. This is on the main 2Code page in the Gibbon section.
- 2 copies of Number Cards 0-23.
- Variable Game Cards.
- 2 boxes (transparent if possible, e.g. ice-cream or take away containers)

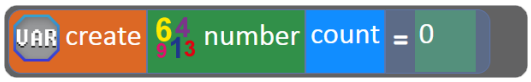
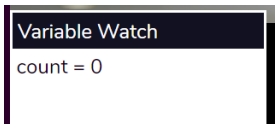
Preparation

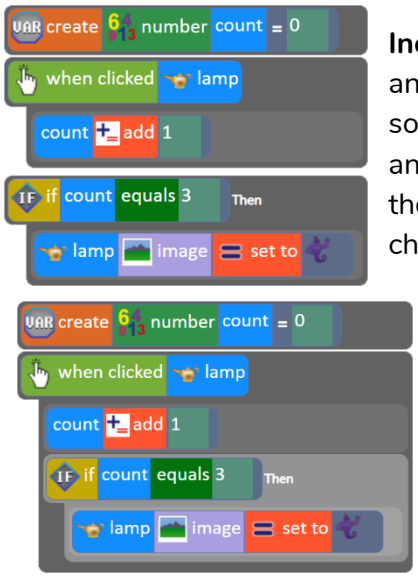
- Set Genie as a 2Do.
- Set Night and Day as a 2Do (if using extension)
- Print and cut up 2 copies of number cards 0-23.
- Print and stick 4 Variable Game Cards under 4 children's chairs.

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Vocabulary	Display **slide 4**. Explain that today we will be working with **variables**. Go through the definition.

	Display **slide 5**. 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 6**. 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 7** 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 *replaced* with the new value each time – a variable holds only 1 value.
Look under your chair	Display **slide 8**. 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. NB. At this point your **variable value** might increase to a number higher than you have number cards for, if this happens create the relevant extra values out of post-it notes or scrap paper! Who won? Discuss how the answers impacted the value of the **variables** and emphasise the importance of naming **variables** sensibly.
2Code Genie	Use **slide 9** 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 play 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 10**. 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 play 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 play (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 11**. Children have a go at working through Night and Day Gibbon that you have set as a 2Do.
Review Success Criteria	Display **slide 12**. Ask children to 'hand in' tasks with an honest review of how they got on. Review the lesson together against the success criteria.

Remember to close your 2Dos when you have finished the lesson.

Lessons 7 and 8 – Design and Make Interactive Scene

Lessons 7 and 8 are recommended optional lessons that will further embed the children's coding understanding. They are designed to ensure children have enough creative coding experience in order to be ready for Coding unit 5.1 in Year 5.

Aims

- To design and create an interactive scene.

Success Criteria

- Children can use the **properties** table to set the **properties** of **objects**.
- Children can plan their **scene** and **algorithms** before they create their program.
- Children can confidently make several different things happen in a program.

Resources

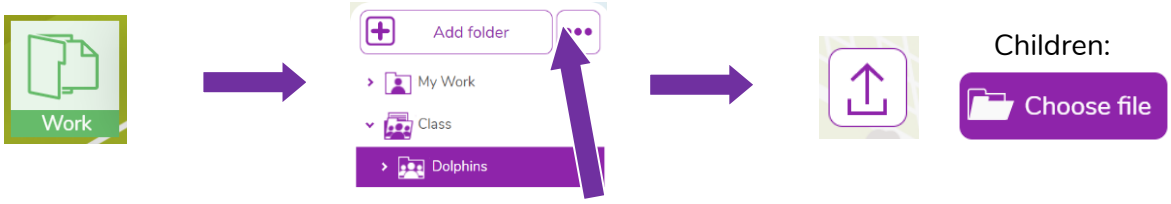
Unless otherwise stated, all resources can be found on the main unit page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you do not lose this page.

- Lightning Scene.
- Moon Phases.
- Solar System.
- Viking Discovery.
- Unicorn Dog Seagull.
- Free Code Chimp (this is found on the main 2Code page).
- Storyboard Planner.
- Scene and Code Planner.
- Sketch Plan Example.

Preparation

- Print and copy a range of planning documents for children to use in Lesson 7.
- Open the example programs: Lightning Scene, Moon Phases, Solar System and Viking Discovery in 4 browser tab for easy access.
- Set Free Code Chimp for children to refer to in Lesson 7 and use in Lesson 8.

Children might want to be able to use images that are photographs or not part of Purple Mash when creating their program. You could create a folder of topic-related images in the class folder for them to choose from when they are coding. You can upload these **in the work area** and then children can select 'choose file' from the galleries in 2Code to access them:



Children could alternatively source them, save them and upload them to their own folder or import them directly from their device.

Lesson 7 Activities

Introduction	Display **slide 2** and outline the lesson aim. Display **slide 3** and outline the success criteria.
Design and Make an Interactive Scene	Display **slide 4** and introduce the main activity which will be spread over two lessons. Explain that the main aim of this lesson is to create a plan that would be good enough for someone else to follow to make your program if you didn't.
1. Lightning Scene	Display **slide 5**. Open the program, look at the design and the code and then run the program and discover how it works. Give children time to discuss what they like or don't like about each program, and how they might **develop** it – if they think it needs **developing**. Leave the program open in a tab.
2. Moon Phases	Display **slide 6**. Open the program, look at the design and the code and then run the program and discover how it works. Give children time to discuss what they like or don't like about each program, and how they might **develop** it – if they think it needs **developing**. Leave the program open in a tab.
3. Solar System	Display **slide 7**. Open the program, look at the design and the code and then run the program and discover how it works. Give children time to discuss what they like or don't like about each program, and how they might **develop** it – if they think it needs **developing**. Leave the program open in a tab.

4. Viking Discovery	Display **slide 8**. Open the program, look at the design and the code and then run the program and discover how it works. Give children time to discuss what they like or don't like about each program, and how they might **develop** it – if they think it needs **developing**. Leave the program open in a tab.
Object Properties	Display **slide 9** and talk through the object properties for the animal object. Open up the program Unicorn Dog Seagull. Click on an object and look at the options in the **properties** table with the children. How does changing each **property** affect the **object**? In the 'Lightning Scene' most of the **objects** in the design were hidden, and they were programmed to show when they were needed the **sequence**. Set a sensible name for the objects in this scene. If 'allow off screen' is set to 'yes', the **object** will be allowed to move off the screen in the direction it's moving and it will disappear. If it is set to 'no' it will move off the screen in the direction it's moving, but come back on the other side, moving in the same direction. You could add a simple 'animal right' command to demonstrate this.
Different Objects	Display **slide 10**. Remind children that if there isn't an **object** type that matches what they want to add (e.g. they want to add a car) they can choose one that is there (e.g. animal) and then change the image and the name. If you return to 'Lightning Scene' and click on the fire engine you'll see in the properties table that the fire engine is actually an **animal object** with the image and name changing it to a fire engine.
Show/Hide – Don't Forget Nesting!	Display **slide 11** and ask children to read each example of code and **predict** what would happen when the program is **run**. Go back to Unicorn Dog Seagull and program the second example: set the **show/ hide property** for the seagull to hide and then add code to make it show after 2 seconds and say 'Hi!' when it shows. Emphasise the importance of **nesting** the **speak command** into the **after timer** or the seagull will speak but you won't be able to see it!
Button Properties	Display **slide 12** and remind children of the button properties. In Unicorn Dog Seagull, drag a button onto the **scene** and remind children how the **properties** options are different.
Alert!	Display **slide 13** and use it to remind children of the 'Moon Phases' program - notice how there are instructions at the start – explain that these were programmed using an **alert**.
Planning	Using **slide 14**, show the children the planning frameworks you've printed and copied and encourage them to choose a method of

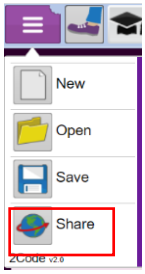
	planning that suits them or that they think suits their program. They might favour a scene sketch, the Storyboard Planner or the Scene and Code Planner.
	Display **slide 15**. Give children time to make their plan, recommend that they have Free Code Chimp open in front of them so they can explore backgrounds and clipart/ images available to them as they plan. If they don't finish this lesson they will be able to carry on in the next, and if they finish they could start making their programs in Free Code Chimp.

Lesson 8 Activities

Introduction	Remind children of the aims and success criteria from the last lesson: Display **slide 2** and outline the lesson aim. Display **slide 3** and outline the success criteria.
Planning	Display **slide 15**. Ask children to get out their program plans from last lesson and complete them if they haven't already.
Create Your Program	Display **slide 16**. When their plan is complete, ask the children to open Free Code Chimp from their 2Dos and create their programs using their plans.
How Did You Get On?	Display **slide 17**. Remind children to save their work when they have completed it, and hand in their 2Do. Share children's work to a Displayboard (see Appendix 1) and allow them some time to view and interact with each other's interactive scenes. Review their work and celebrate achievements.
Review Success Criteria	Display **slide 18.** Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

If you want to share the programs children have created you can create a QR code or web link to them. This can be inserted into a school blog or webpage:

How to Create a QR Code

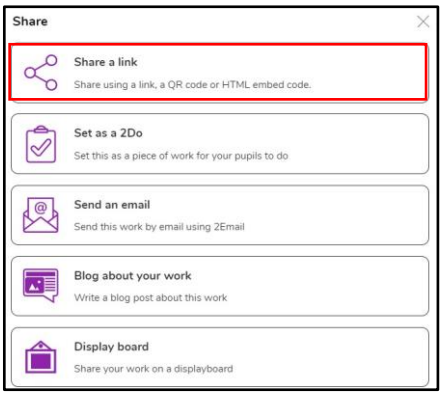


- Save the file.
- From within the menu, click on 'Share':

10.

11.

- Next, select Share, then Link and QR code



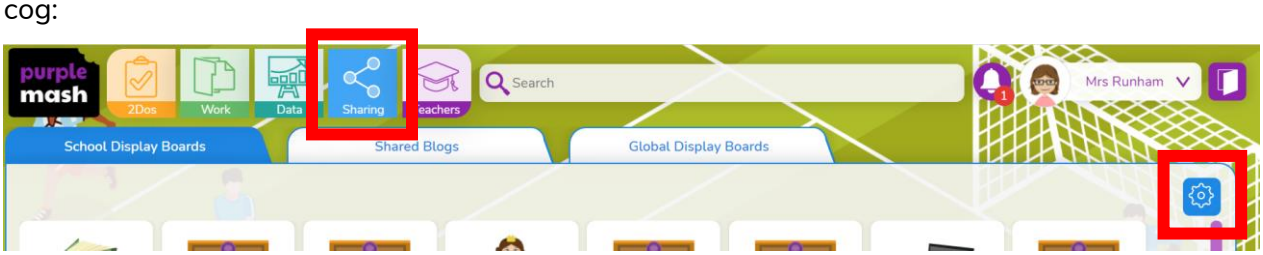
- The link and QR code can be copied and pasted into documents. Clicking on the QR code will show a large image that can be saved into the computer (right-click on it, choose Save As).

Appendix 1: Display Boards

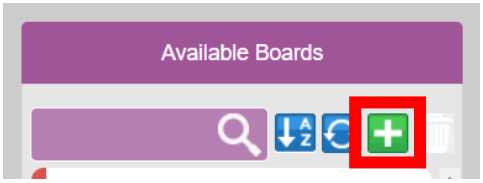
Create the Display Board

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

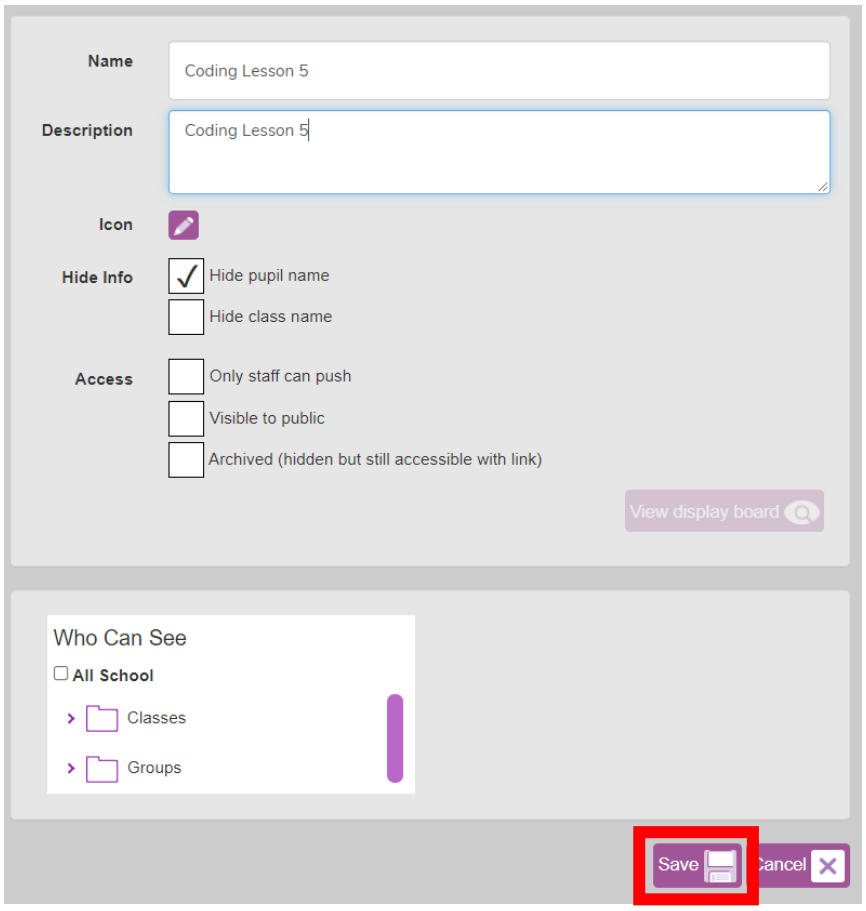
1. Click on the 'Sharing' button to find the Display Board tab, and then click on the settings cog:



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':

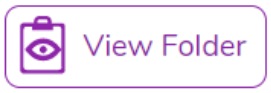


4. Exit Display Board settings:

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:

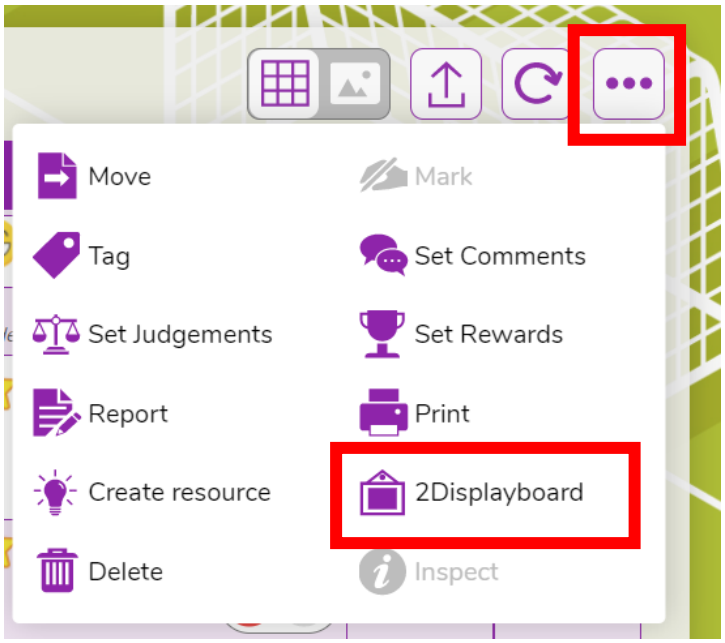
1. Click on 'View Folder' from the 2Do:



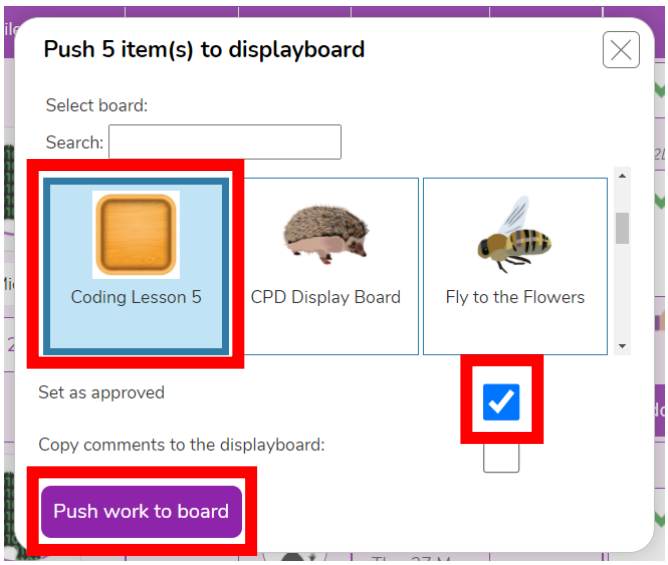
(or navigate to the work you want to share in the Work area).

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.

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':

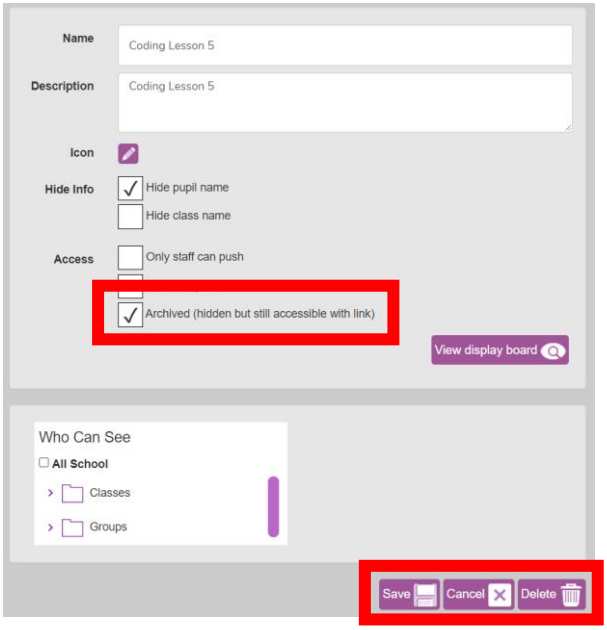


5. Click on 'Sharing' button and then on the display board, you should see the work you've added. It can be deleted by clicking on 'Edit' at the top of the board, then clicking work and then delete. This will remove it from the display board, it won't delete it from Purple Mash.

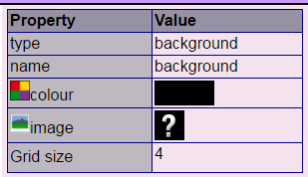
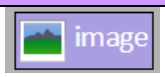
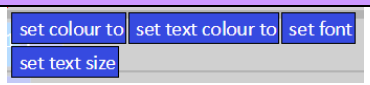
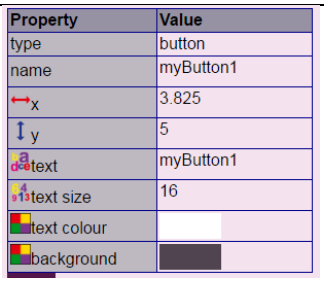
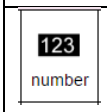
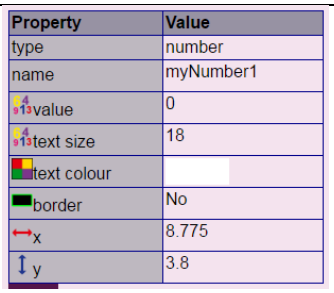
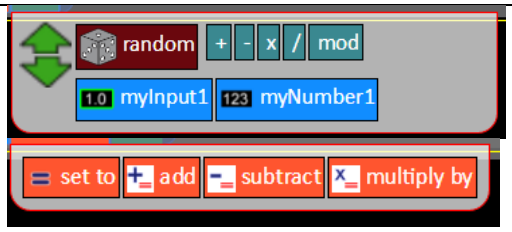
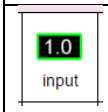
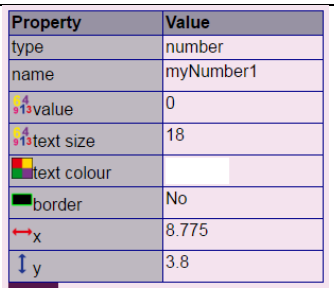
Deleting or Archiving a Display Board:

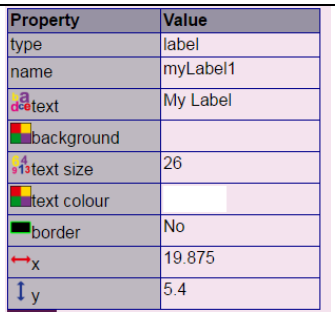
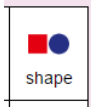
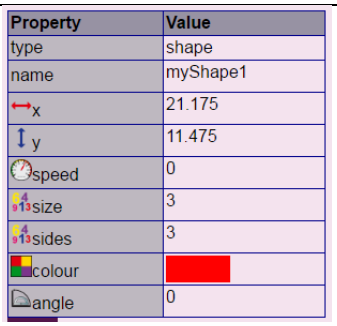
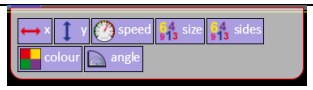
When you've finished the lesson, you can return to the Display board settings and either delete it or archive it to stop it appearing under the 'Sharing' button.

1. Click on 'Sharing' and then on the settings cog.
2. 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.

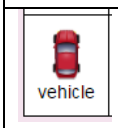
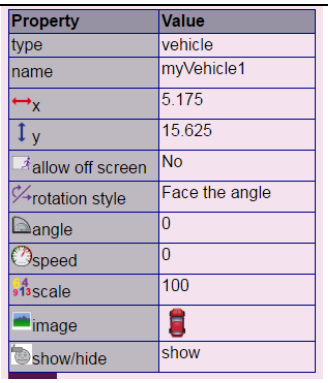
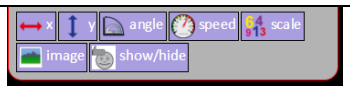


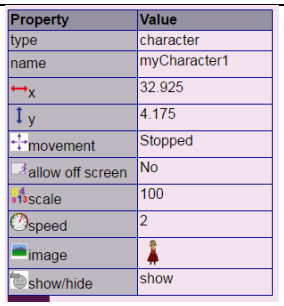
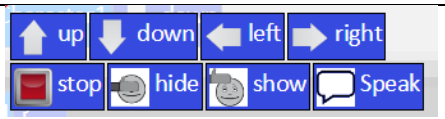
Appendix 2: Actions for Gibbon objects

Object	Properties in Design View	Properties in Code View	Actions in code view
			 These set the background colour and the properties of any text that is printed to the screen.
		None	None
			 This displays a number on the screen which can be set to different values and/or have calculations performed on it. In the image above, myInput1 and myNumber1 are objects that also have a number value and the value of these can be set to affect the value of the number object.
			This displays an input box on the screen into which a number can by typed. In the code, input can be set to different values and/or have calculations performed on it.

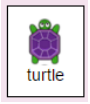
<table>
<tr>
<td> 
</td>
<td> 
</td>
<td>None</td>
<td>None. The text for the label is set in design view and cannot be changed</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
<td> 

The options offered will depend upon the property selected.</td>
<td> 

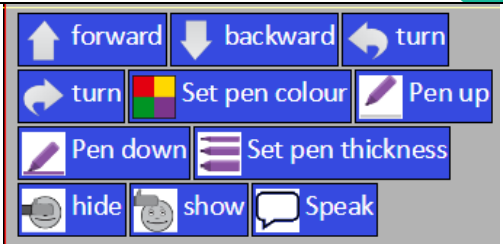
These make the object move in different directions.
Stop, hide, or show the object.
Make the object speak by displaying a speech bubble.</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
<td> 

The options offered will depend upon the property selected.</td>
<td> 
</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
</td>
<td> 

These make the object move in different directions.
Stop, hide or show the object.
Make the object speak by displaying a speech bubble.</td>
</tr>
</table>



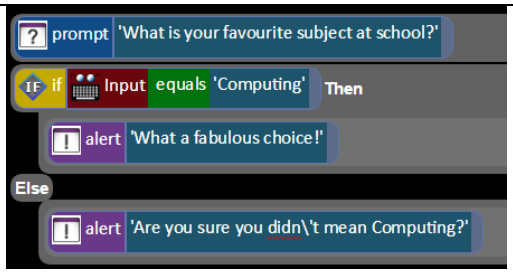
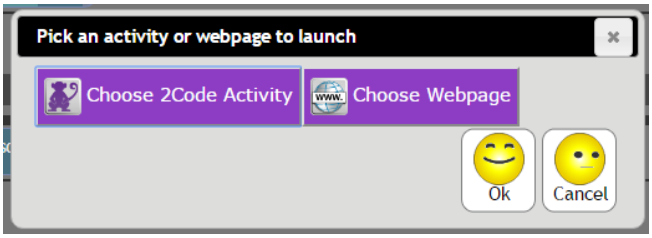
Property	Value
type	turtle
name	myTurtle1
x	10.825
y	15.95
angle	0
scale	100
image	
show/hide	show

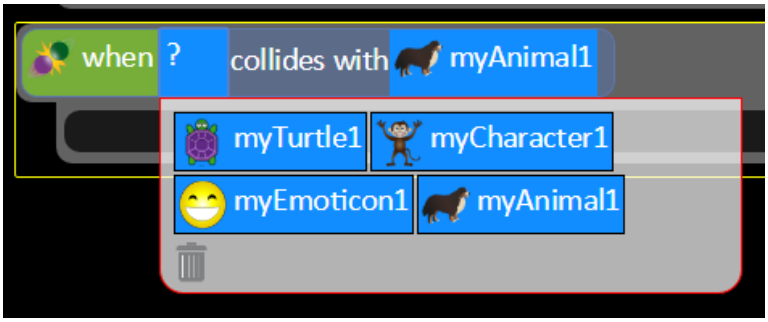


A turtle moves in a similar way to a floor turtle using Logo type actions. Turn is by a number of degrees.

Appendix 3: Commands for Gibbon objects

Command	Explanation
print to screen	Prints some text specified by the coder to the screen.
sound	Causes a sound to play. the sound picker will open for the coder to select a sound, when this code block is added to the code window.
alert	Creates a pop-up window with a message for the user and an OK button to click.
get input	This command will put a cursor in the top left of the screen and get the input typed onto the screen. For example if you have an alert that asks the user to type their name, you can use this to print their name back to them: 
prompt for input	This combines the alert and get input functions, a pop-up screen will ask the user to enter something and they type it into a text box on the pop-up screen.
timer	Create a timer. The coder can select whether this time should run after a certain length of time or every x length of time. The time length is measured in seconds or quarter seconds. 
if	This runs the code inside if a certain condition is met. The condition could depend upon something entered by the user or upon the value of a variable.
if/else	This runs the code inside the first block if a certain condition is met, otherwise it runs the code inside the second block.

	
repeat	Repeats the code inside it either forever or every x (or quarter seconds). 
repeat until	This command repeats the code inside until a certain condition is met. The condition could depend upon something entered by the user or upon the value of a variable.
restart	This will restart the program from the beginning. Useful if you want to include a restart button in your program.
launch	This will launch another 2Code program or open a web page. The following screen will allow the coder to select which. You might want buttons in your program to link to other programs that you have made or to the Internet. The launch command can be useful when you write much bigger programs as you can split them into smaller chunks that launch each other. 
when key	Runs the code inside it when the specified key is pressed. The code chooses which key (including arrow keys and space bar)
when clicked	Runs the code inside it when the object is clicked. The coder is given a choice of all the available objects.
when swiped	Useful for tablets. This command runs the code inside it when an object is swiped. The coder chooses the object and the direction of the swipe. You can use the swipe speed and swipe angle in the code. This works especially well when applied to objects that can have both their angle and speed set, such as vehicles. Remember that you can change the image of a vehicle to anything else such as a person if you want to be able to do this with objects that don't look like vehicles.

	
collision detection	Runs the code inside it when two objects collide. The coder selects the two objects. 

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 have a basic understanding that coding involves writing instructions that a computer can follow. They are developing their understanding that these instructions must be precise and carefully structured through their work in Free Code Gibbon making simple one and two step programs for example in the lesson 1 bubble program or making an object move when clicked on in (lesson 1). Children know that an algorithm is related to giving instructions. With support, children can manipulate how their program looks using the 2Code design mode, by adding and changing backgrounds, characters, sounds (lesson 1) and objects. They can create a program that controls a character. They can make a character move when clicked but might not be able to plan how to make a character move when a different character (or the background) is clicked. Children are beginning to understand that they can correct unexpected outcomes by changing the code and they make attempts to identify the source of bugs. With support, children can explain the possible actions of objects including movement, clicking on them and collision. When looking at a simple program they can 'read' the code one line at a t ime but might not be able to envision the bigger picture of the overall effect of the program. Children will be able to suggest that an object might move when clicked but might not be able to suggest that an object might move when the background is clicked. Children can design and code a program that follows a simple sequence (lesson 1). Children attempt to introduce repetition and selection into their code using timers and simple 'if statements' (lessons 2, 3 & 4). Children' 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 4 they can follow the examples with support but will struggle when applying this with their own ideas. Children can use the 'get input' command (lesson 2, step 14) to work with user input. Children have a basic understanding that coding involves writing instructions that a computer can follow.

Assessment Guidance	
	They are developing their understanding that these instructions must be precise and carefully structured through their work in Free Code Gibbon making simple one and two step programs for example in the bubble program or making an object move when you click it (lesson 1). Children know that an algorithm is related to giving instructions. They can relate a simple one-step algorithm to the outcome of code in Free code Gorilla. For example, in Lesson 2 they have been able to make a program that follows the algorithm e.g. 'when the helicopter is clicked it takes off'. With support, children can create a simple one step program that achieves a specific purpose. With support, children can identify and correct errors (Lesson 3). With support, children can identify the parts of an algorithm that control and initiate specific actions. Based on this, children can make good attempts to 'read' code and predict what will happen in a program which can help them to correct errors (Lessons 3 and 4). Children's designs for their programs, show that they are thinking of the structure of a simple program in logical, achievable steps (lessons 7 and 8). With support, children can turn a real-life situation into an algorithm for a program that has cause and effect (Lesson 6) and use their algorithm to write simple programs using 2Code (Lesson 6). 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** (Lesson 5). 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 6 they can follow the examples but might struggle when applying this with their own ideas.
Expected	Children have an understanding that coding involves writing instructions that a computer can follow.Children can explain that an algorithm is a set of instructions to complete a task. They have turned algorithms of more than one step into code using freecode Gibbon. 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 (Lesson 1). Children can 'read' others' code and predict what will happen in a program which helps them to correct errors. They can also make good attempts to fix their own bugs as their coding becomes more complex (Lessons 7 and 8).

Assessment Guidance	
	Children experiment with the use of timers to achieve delay effects in their programs – they understand the difference between timer-after and timer-every commands (Lesson 4). 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 (Lesson 6). They make use of user input (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 (Lesson 6) and make attempts to use and manipulate the value of variables. Most children can integrate multimedia components such as sounds, animation and images into their coding. They can apply specific actions to these objects to animate them as part of the overall process of creating their own program (Lessons 7 and 8). Children can interpret the flowcharts used to represent IF and IF/ELSE statements (Lesson 5) and create their own when planning their programs.
Exceeding	Children have a clear understanding that coding involves writing instructions that a computer can follow. Children can explain and give examples that an algorithm is a set of instructions to complete a specific task. They can create complex and logical algorithms of several steps that accomplish the aim of the task that can be easily utilized to create executable code. Children show an awareness of the need to be precise in their designs so that algorithms can be successfully translated into code (Lesson 2). Children can identify the parts of a program that respond to specific events and initiate specific actions. Based on this, children can adopt a systematic approach for predicting the behaviour of programs. Furthermore, using cause-and-effect language, Children can reason in detail about what will happen in a program (Lesson 3). Children can identify an error within a program that prevents it following the desired algorithm and then fix it. Children make intuitive attempts to debug their own programs as they increase in complexity (Lesson 3). Children have a good understanding of timers within timers in a program (Lesson 4) and this is evidenced in their program designs (Lessons 7 & 8). Children's designs show that they are thinking of the required task and how to accomplish this in code (Lessons 7 & 8) using new knowledge of coding structures such as IF/ELSE statements and variables. Children make intuitive attempts to debug their own programs as they increase in complexity (Lessons 4, 5 and 6).