purple mash

CRASH COURSE

Computing Scheme of Work

Year 6 Coding Crash course

For children in Year 6 who have not used 2Code previously.

Contents

Introduction

The crash-course is a catch up course designed for children in Year 6.

Differentiation

The Gibbon and Gorilla 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' 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'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	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.
Least scaffolded	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/

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	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.
3	Using Repeat	• Children understand how the **turtle object** moves. • Children can use the **repeat command** with an **object**. • Children can create a computer program that includes use of the **repeat command**.
4	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.
5	Number Variables	• Children can explain what a **variable** is in programming. • Children can create and use **variables** when programming.
6	Friction and Functions	• Children can create a program which represents a physical system. • Children can create and use functions in their code to make their programming more efficient.
7 Optional recommended lesson	Coding Efficiently	• Children can use **simplified** code to make their programming more efficient. • Children can use **variables** in their code. • Children can create a simple playable game.
8 Optional recommended lesson	Making 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**.

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

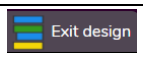
Unless otherwise stated, all resources can be found on the unit main page. From here, they can be set as 2Dos by clicking on the icon. To preview resources linked to here, right-click and '**open in new tab'** so you do not navigate away from this page.

- 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

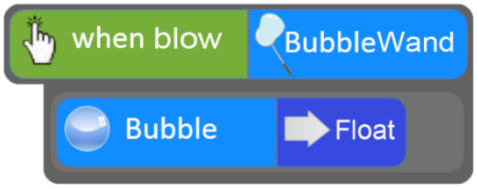
Preparation

- 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 **Command**. 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 reiterate 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 and see if children can tell you what they might see when you click on '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 – 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.
- 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 and see what happens.
Designing a Scene	Display **slide 7** and open Free Code Chimp in front of the class. Follow the instructions on the slide to set the scene.
Choosing Objects	Display **slide 8.** Look at the **object** types to choose from on the left and add a turtle and 3 other **objects** that would move (tell children that in this lesson we are using any object apart from the button – we are going to look at the button later in the lesson).
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	With **slide 10**, 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. 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**. Add a button to your scene.
Button Properties	Display **slide 16** and look at the button properties. In 2Code, name the button and set the text for the button. You can also change the **text size**, **text colour** and **background** colour on 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 add and use a button. **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?	With **slide 19**, ask children 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 3 – Using Repeat

Aim

- To understand how to use the **repeat command**.

Success Criteria

- Children understand how the **turtle object** moves.
- Children can use the **repeat command** with an object.
- Children can create a computer program that includes use of the **repeat command**.

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 Gorilla this is found on the main 2Code page).

Preparation

- Set Free Code Gorilla as a 2Do, call it 'Repeat Command'.

Activities

Introduction	Display **slide 2** and outline the lesson aim. Display **slide 3** and outline the success criteria.
Vocabulary	Display **slide 4**. Introduce the new vocabulary and discuss the piece of coding shown.
The Repeat Command	Display **slide 5**. Explain to the children that they will be creating a program that uses the **repeat** command. Follow the instructions on the slide to start the scene.
Button Properties	Use **slide 6** to discuss and change the properties of the button, including text, size, text colour and background colour.
Draw a Square	Display **slide 7** and refer back to the aim of the task. Children to draw and think about the process of drawing a square.
	Display **slide 8** and ask a child to walk in a square. Discuss the questions on the slide.
Program the Turtle	Display **slide 9.** Talk through the slide and then open Free Code Gorilla and drag across the **repeat command** to start your code.

	NB: If the children want to start with the button or when clicked, this is not incorrect, add that in and then add the repeat command inside the when clicked button event and continue.
	Display **slide 10**. Once the **repeat command** is set, we need to add inside it the **block of commands** that will repeat. Talk through the slide and then return to Free Code Gorilla and drag in the **object code block** for your turtle object (mySuperTurtle1 if you haven't renamed it) and look at the possible **actions** for that **object**. Look at the available actions and ask children what they want the turtle to do 4 times. They are likely to say 'forward', and then 'turn'. That is fine to start with.
	Display **slide 11**. Talk through the slide and then return to Free Code Gorilla and add in the code similar to that shown and then **test** it (if children have asked you to add the button in already the code for the repeat command will look like this, but it will be nested in a **when clicked button event**). You'll see when you **test** this code, that the turtle moves in a square but doesn't draw anything. Ask children to help you fix the problem – by adding in turtle 'pen down' and turtle 'pen up' **commands** at the beginning and end of your **block of commands. Test** the code, watch the turtle draw a square, continue to **test** and **debug** if needed.
	Display **slide 12**. Return to the aim of the task – ask children: Does our **algorithm** achieve this aim? We need to program all this to happen when the **button** is clicked on (unless the children have already told you to do program the button). Return to your code and ask children to help you add the **button** in and move (drag) the **repeat command** into the **when clicked** event for the **button**. **Test** the code, click on the button, watch the turtle draw a square- or continue to **test** and **debug** if needed.
Develop the Program	Display **slide 13**. Talk through the slide and then demonstrate to children how you could develop the program by altering the image and the background. Remind children that they can alter the image of the **turtle object** by double-clicking on it, or by clicking on the image in the **properties** table to open the clipart picker. Remind children how to rename **objects**.
Activity 1: Create your Program	Introduce the activity with **slide 14**. Ask children to start 'Repeat Command' from their 2Dos and create a program that uses a **turtle object** and the **repeat command**. They can choose any **background** and change their **turtle object** into something else from the clipart gallery.
Activity 2: Extension	Display the extension activity on **slide 15**. Ask children to see if they can develop their program by adding more objects and buttons, and explore what happens if they change the number of times the commands repeat or the angles the objects turn? Can they find out the number of sides and angles they'd need to set to make different

	shapes? – You could ask them to look it up or display them on the board for them to refer to.
Review Success Criteria	Display **slide 16.** 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 Gorilla (this is found on the main 2Code page).
- IF/ELSE Flowchart template
- Small whiteboards

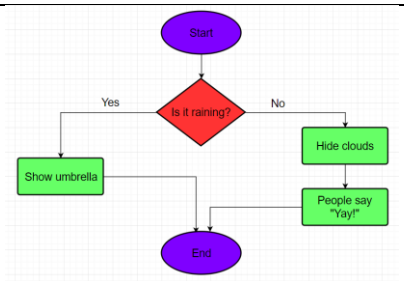
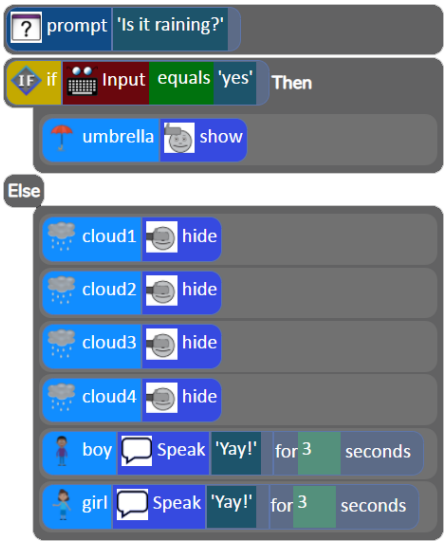
Preparation

- Set 'Lost' 2Code Example as a 2Do.
- Set Free Code Gorilla 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: 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 flowchart, they have a go at making their program by going to their 2Dos and starting Free Code Gorilla.
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 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 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 Gorilla. 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)

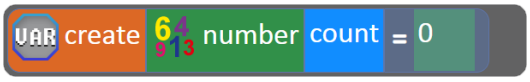
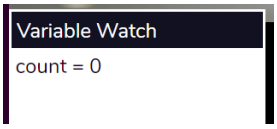
Preparation

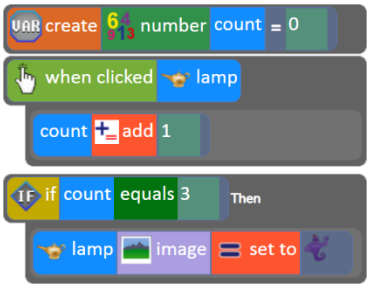
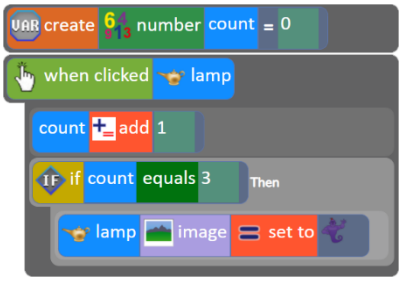
- Set Genie as a 2Do.
- Set Night and Day as a 2Do (if using extension)
- Print and cut up 2 copies of number cards 0-23.
- Print and stick 4 Variable Game Cards under 4 children's chairs.

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, 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 stay the same. . 'If a Team 2 answer is correct, the value of team2score will increase by 1, else the value of team2score 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 **slide 7** <u>The Quiz</u> on the board 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 choose a 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** but for 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.

Lesson 6 – Friction and Functions

Aims

- To understand how to use **friction** in code.
- To begin to understand what a **function** is and how **functions** work in code.

Success Criteria

- Children can create a program which represents a **physical system**.
- Children can create and use **functions** in their code to make their programming more efficient.

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 don't lose this page.

- Football Game. This is on the main 2Code Page (scroll down to the Gorilla activities).
- Friction Example.
- Physical Football (Optional)

Preparation

- Set Football Game as a 2Do.
- Create a display board for the class to share their programs to. Details of how to do this are given in Appendix 1.
- Physical Football (Optional)

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria. Explain that often when coding we want to **simulate physical systems** – program objects to behave in a realistic way.

	Gently kick a football across a space in front of the class. Watch it roll, then come to a stop. Ask children what would happen if you'd kicked it harder? And what if you were on a wooden floor and not carpet?
Football	Display **slide 4**. Explore this slide together as a class. Discuss the code and questions.
Football Friction	Display **slide 5**. Open the Friction Example. Go through the slide together as class. You might choose to let the children explore the Friction Example in small groups and feed back to the class.
	Display **slide 6**. Discuss with the children the code on the slide. Explore how friction has been set for each surface.
Football Game	Display **slide 7**. Children to open Football Game from their 2Dos. Explain the stages and what they are simulating. Display **slide 8** and work from stages 1 to 3 as a class, model each one and then give the children time to complete it. • **Stage 1:** Choose 'any' so the ball can be swiped in any direction. A speed of 1-4 is sensible. • **Stage 2:** Add **friction** to the ball, friction of between 1-4 is about right, test and change it and see what children want to set. • **Stage 3:** Involves altering existing code so click on what you want to change – the speed, click on the value for the current speed and select swipe speed so the speed of the ball will match the speed of the swipe. Drag in football and set the angle in the same way. Talk through **slide 9** with the children and use it to help them understand how co-ordinates can be used in coding. Use **slide 10** to introduce children to **functions**. Watch the video for **stage 4** together. At the start of the program the ball is at X=3, Y=8 and the speed = 0. Creating this **function** will apply those properties to the ball whenever it is used (**called**). Talk through the code on **slide 11**, and then demonstrate it in 2Code. We use 'create variable' to create the **function** – a **function** is a type of **variable** (this can be a difficult concept to comprehend until you are creating more complex code (e.g. Java or Python coding) when the reasoning becomes clearer). Then use 'change variable' to add your **function** into your collision detection **event.**

Activity 1: Football Game	Display **slide 12**. Use this to set the children off on completing the Football Game from their 2Dos finish. Children to carry on until they have made the football game. Review children's progress and ask them how many times they called their **function**. The **function** contained 3 pieces of code: - The X Co-ordinate - The Y Co-ordinate - The speed Notice together how calling the **function** saved them coding all this each time. **Functions** also help us **simplify** code and make our programming more efficient.
How did you get on?	Display **slide 13**. Share children's work on a Display Board (see appendix 1) and give them some time to play each other's games.
Review Success Criteria	Display **slide 14.** 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 7 – Coding Efficiently

Lesson 7 is a recommended optional lesson that will further embed Year 6 children's coding understanding. It is designed to ensure children have enough creative coding experience in order to be ready for Coding unit 6.1 in Year 6.

Aims

- To review existing coding knowledge.
- To begin to **simplify** code.
- To create a playable game.

Success Criteria

- Children can use **simplified** code to make their programming more efficient.
- Children can use **variables** in their code.
- Children can create a simple playable game.

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 don't lose this page.

- Coding Vocabulary Quiz.
- Animal Race 1 (example 2Code program).
- Animal Race 2 (example 2Code program).
- Catching Game. This is on the main 2Code Page (scroll down to the Gorilla activities).
- Free Code Gorilla. This is on the main 2Code Page (scroll down to the Gorilla activities).
- 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 Catching Game as a 2Do.
- Print/ copy When Picked Blank Code enough for one per child.
- Create a display board for the class to share their programs to. Details of how to do this are given in Appendix 1

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.
Vocabulary	Use **slide 4** to open the Coding Vocabulary Quiz and complete as a class to help refresh coding knowledge from previous lessons. 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. You could use the vocabulary cards to find the answers and display in the classroom.
Activity 1: When Picked, Stand Up	Display **slide 5.** Ask the children to look at the design and read the code, can they **predict** what will happen when the program is **run**? Use the slide to open Animal Race 1, click on play to **run** the program and click on the animals to see if their predictions were correct. Recap **event** – **object** – **action**, identifying each in this code.
	Display **slide 6**. The design in this program is the same, but the code is different. Can children **predict** what will happen when this program is **run**? Use the slide to open Animal Race 2, click on play to **run** the program and click on the animals to see if their predictions were correct. Explain to children that in this lesson they will revise some of the vocabulary and concepts they have learnt in Year 6, and start being able to **simplify** code to make their programming more **efficient**. Discuss what it might mean to make things more efficient. Return to **slide 6** and begin to look at how this code works.
	Display **slides 7-8**. Click through the slides and use them to help you explain how the code in Animal Race 2 works. In Year 6 children will use Free Code Gorilla. At the end of **slide 8** you could open Free Code Gorilla at this point and use it to recap children's understanding of different object types.
Catching Game	Use **slide 9**. Display Catching Game on the board. The following slides go through and explain points about this activity. Complete it on the board together or get children to open it as well and complete it with you one step at a time using the following slides.

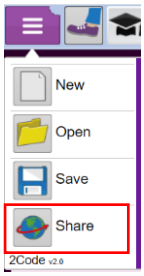
	Refer to key vocabulary and concepts as you go through it with the children including **selection**, **IF/ELSE Statements**, **prompt**, **number variable**, **timer** (**after/ every**), **event**, **object**, **action**, **co-ordinates**.
Catching Game: Stage 1	Use **slide 10** to work through **Catching Game Stage 1** Start by clicking on 'Design' and looking at the **scene** together, there is a score**,** a catcher and some food **objects**. Click on the food **objects** and set a speed in the Property Value box for each one– around 1-3 is sensible!
Catching Game: Stages 2 and 3	Use **slide 11** to work through **Catching Game Stages 2 and 3** • Stage 2: Create arrow key press events so the player can control the catcher. • Stage 3: Add a collision detection event so that when the catcher collides with a weight, a sound plays and the game starts again. Explain that because there are 2 objects of the same **object type** (weights), 2Code gives you the option of using 'Any Weight' to save you having to program a collision detection for each one. It has automatically generated a variable for this object type. This puts into practise simplifying the code. NB: weight is a custom object type that is not available in Free Code Gorilla – but children can achieve a similar outcome by adding a group of the same type of objects from the options they have – identified in **slide 8**. When you test this code children might notice that the sound doesn't play because the program restarts straight away (re-run and watch when the code highlights orange to notice this) – discuss with them how that might be overcome later when we are able to fix it.
Catching Game: Stages 4 and 5	Use **slide 12** to work through **Catching Game Stages 4 to 6** • Stage 4: Write code that increases the score by a value of 1 and plays a sound when the catcher collides with the food. • Stage 5: Write code so that the food hides when the catcher collides with it. The individual food object that needs to hide is the value of the computer generated variable 'Collided food'. The value of 'Collided food' could be any of the food items – whichever is clicked on. This variable value will be set by the collision detection event, so they will

	need to use 'change variable' for this. Once they drag 'change variable' into the collision detection event they can select 'Collided food'. • Stage 6: Ask the children for ideas for how the game can be improved and together fix the problem of the sound not playing when the catcher bumps into a weight – add in a timer so the game restarts after a second.
Activity: Improve Catching Game	**EITHER (Slide 13)** Ask children to open Purple Mash and work through the Catching Game 2Do. They should work through all the steps you've been through and then try and improve the game in the final stage – challenge them to include x and y **co-ordinates** and an **IF statement** in their code. Challenge them to fix the problem where the sound doesn't play before the game restarts when the catcher collides with a weight. Remind children they can click on the instruction to bring the video back up. **OR (Slide 14)** Ask children to use Free Code Gorilla to have a go at creating their own game. OR both! You could refer children back to lesson 2 slides 7-10 for a reminder of how to add and change backgrounds and objects in Free Code Gorilla.
How did you get on?	Display **slide 15**. Share children's work 2Displayboard (see Appendix 1) and allow them some time to play each other's games. Review their work and celebrate achievements.
Review Success Criteria	Display **slide 16**. Review the success criteria from **slide 3.** Children could rate how well they achieved this using a show of hands.

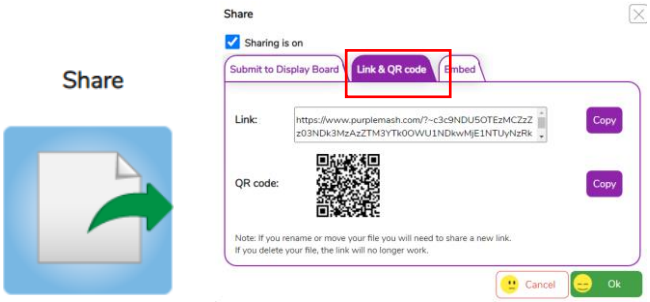
***If you want to share games you can create a QR code or web link to them. This can be inserted into a school blog or webpage:**

How to Create a QR Code

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



- 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).

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

Lesson 8 – Design and Make a Game with a Score

Lesson 8 is a recommended optional lesson that will further embed the children's coding understanding. It is designed to ensure children have enough creative coding experience in order to be ready for Coding unit 6.1 in Year 6.

Aims

- To review vocabulary and concepts learnt in Year 6 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 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.

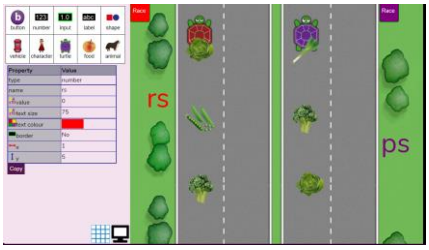
- Free Code Gorilla (this is found on the main 2Code page).
- Storyboard Planner.
- Turtle Race game.

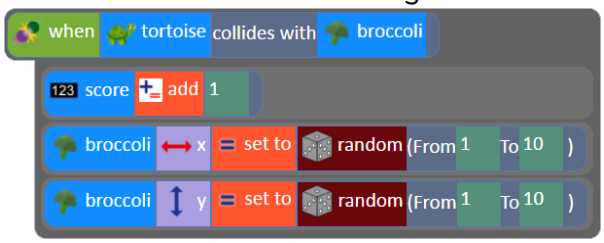
Preparation

- Set Free Code Gorilla as a 2Do.
- 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

Activities

Introduction	Display **slide 2** and outline the lesson aims. Display **slide 3** and outline the success criteria.

Vocabulary	Use **slide 4** to review key vocabulary and concepts used or learnt in Year 6 coding lessons: **Selection**, **IF**, **IF/ ELSE**, **timers** (after, every), **repeat**, **prompt**, **input**, **variable.**
Turtle Race	Use **slide 5** to display Turtle Race game on the board, look at the design with the children. Click on 'rs' and 'ps' and look across to the **properties** table – notice they are **number objects** with a **value** set to 0. Objects of the number type **can display a variable value.** Open Turtle Race and click on play 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 **properties** table. Ask the children to speculate as to how this game might be played.
	Display **slide 6**. Look at the code and 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 two number objects? When the turtles collide with the food the value property of the object changes. Click on play 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. 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** to open Free Code Gorilla. Click on 'Design' and add a **background** and **objects**, use the **properties** table to name them appropriately. Add a **number object type** called 'score'. Your scene might look something like this:  Click on 'Exit Design' 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 Gorilla 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 Gorilla. 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 **objects** to make a 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 • Alert boxes (these could be good for instructions)

How did you get on?	Display **slide 11**. Share children's work to a Display Board (see Appendix 1) and play a few of the games together to share children's work and celebrate achievements.
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.

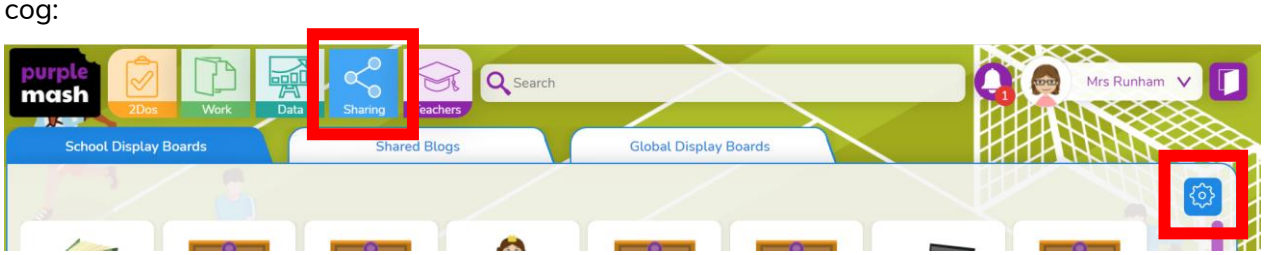
Remember to close your 2Dos when you have finished the lesson.

Appendix 1: Display Boards

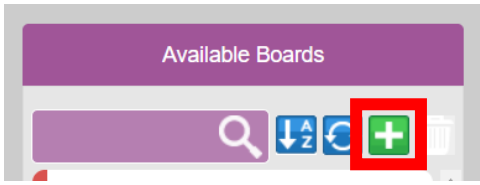
Create the Display Board

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

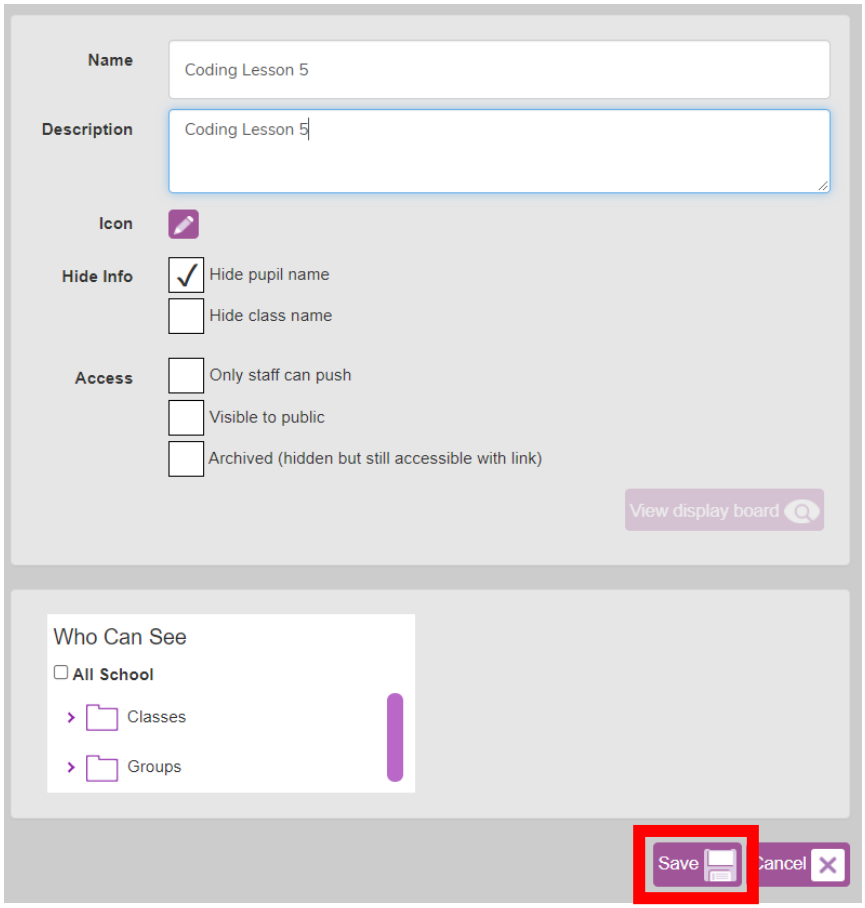
1. Click on the 'Sharing' button to find the Display Board tab, and then click on the settings cog:



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':

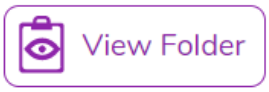


4. Exit Display Board settings:

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:

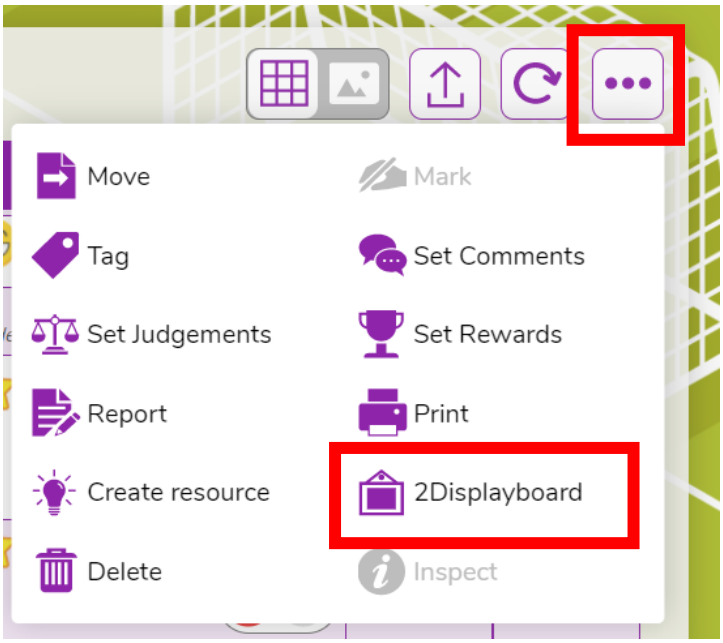
1. Click on 'View Folder' from the 2Do:



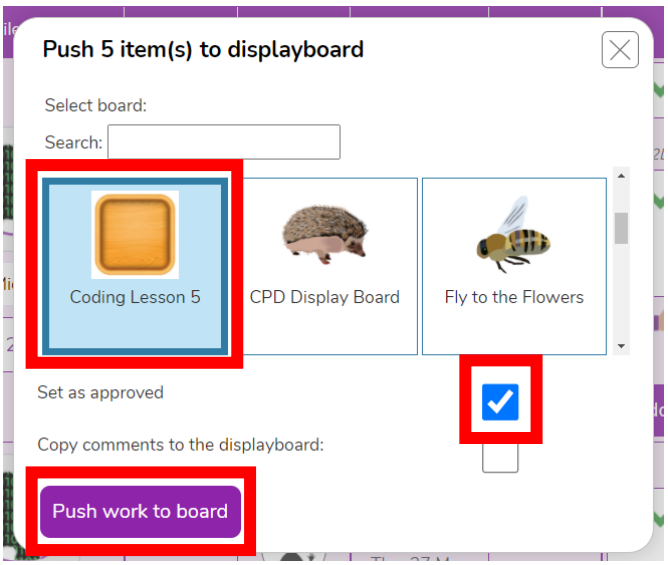
(or navigate to the work you want to share in the Work area).

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':

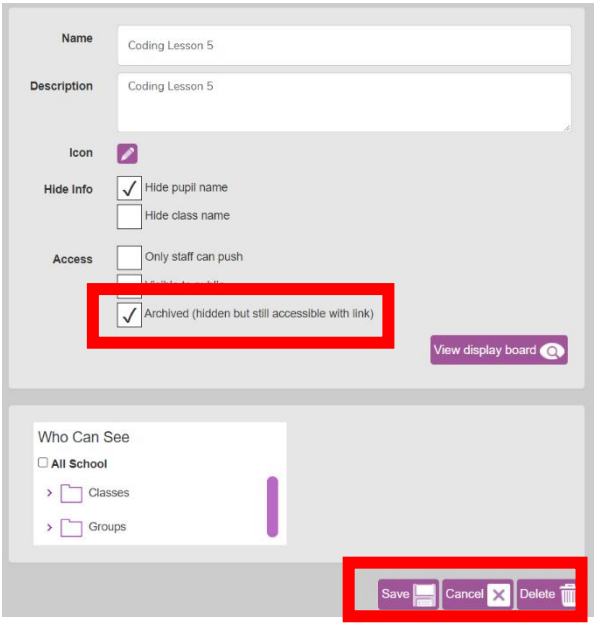


5. Click on 'Sharing' button and then on the display board, you should see the work you've added. It can be deleted by clicking on 'Edit' at the top of the board, then clicking work and then delete. This will remove it from the display board, it won't delete it from Purple Mash.

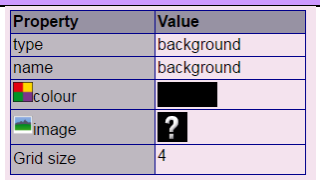
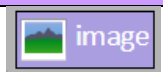
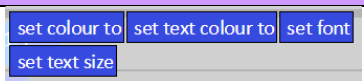
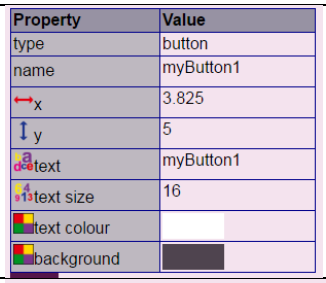
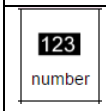
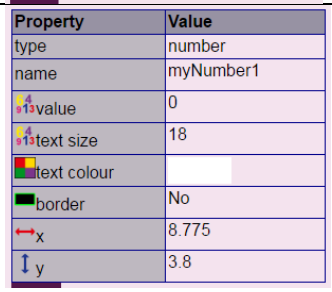
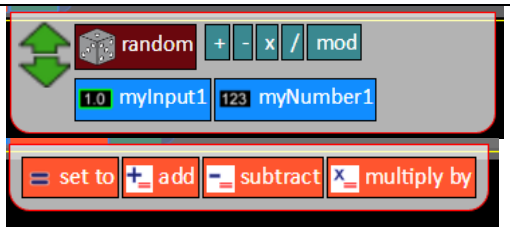
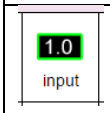
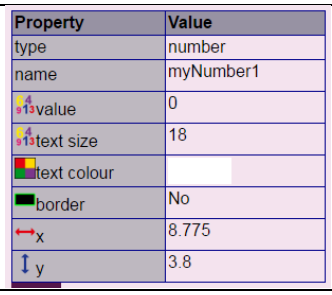
Deleting or Archiving a Display Board:

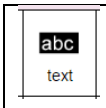
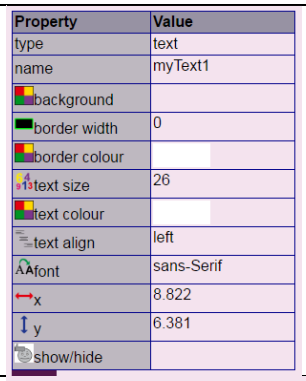
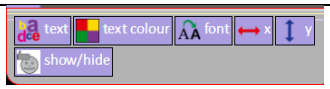
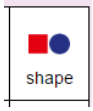
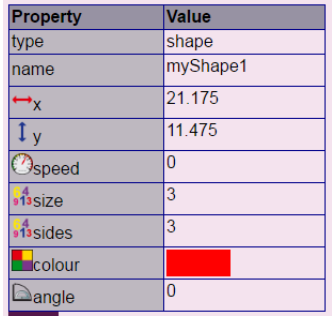
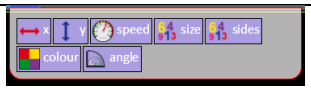
When you've finished the lesson, you can return to the Display board settings and either delete it or archive it to stop it appearing under the 'Sharing' button.

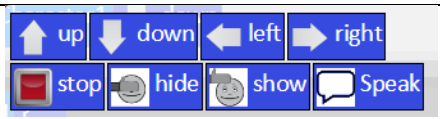
1. Click on 'Sharing' and then on the settings cog.
2. Tick 'Archive', and then 'Save' OR 'Delete'
   Clicking on 'Delete' will delete the display board but the work will still be available in the work area, it doesn't not delete the files.

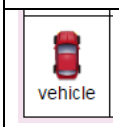
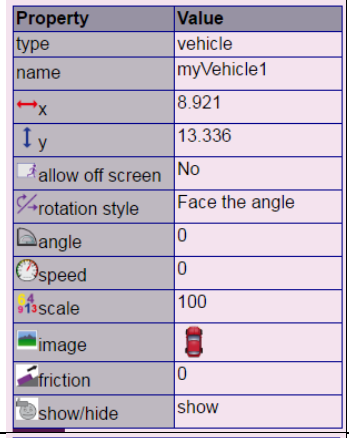
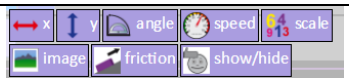


Appendix 2: Actions for Gorilla objects

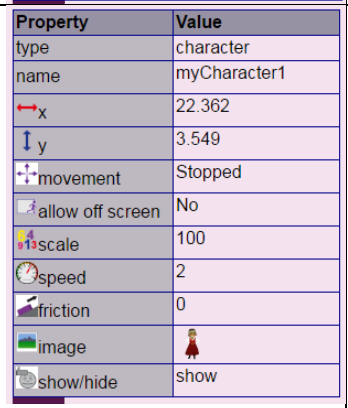
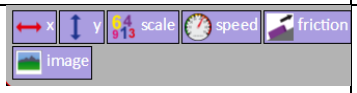
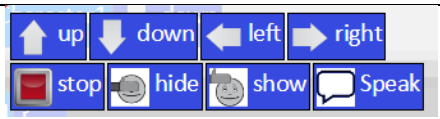
Object	Properties in Design View	Properties in Code View	Actions in code view
	 Property / Value type: background name: background colour: image: ? Grid size: 4	 image	 set colour to / set text colour to / set font / set text size These set the background colour and the properties of any text that is printed to the screen.
	 Property / Value type: button name: myButton1 x: 3.825 y: 5 text: myButton1 text size: 16 text colour: background:	None	None
	 Property / Value type: number name: myNumber1 value: 0 text size: 18 text colour: border: No x: 8.775 y: 3.8		 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.
	 Property / Value type: number name: myNumber1 value: 0 text size: 18 text colour: border: No x: 8.775 y: 3.8		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> 
<table>
<tr><th>Property</th><th>Value</th></tr>
<tr><td>type</td><td>text</td></tr>
<tr><td>name</td><td>myText1</td></tr>
<tr><td>background</td><td></td></tr>
<tr><td>border width</td><td>0</td></tr>
<tr><td>border colour</td><td></td></tr>
<tr><td>text size</td><td>26</td></tr>
<tr><td>text colour</td><td></td></tr>
<tr><td>text align</td><td>left</td></tr>
<tr><td>font</td><td>sans-Serif</td></tr>
<tr><td>x</td><td>8.822</td></tr>
<tr><td>y</td><td>6.381</td></tr>
<tr><td>show/hide</td><td></td></tr>
</table></td>
<td> 
</td>
<td>Show/Hide</td>
</tr>
<tr>
<td> 
</td>
<td> 
<table>
<tr><th>Property</th><th>Value</th></tr>
<tr><td>type</td><td>shape</td></tr>
<tr><td>name</td><td>myShape1</td></tr>
<tr><td>x</td><td>21.175</td></tr>
<tr><td>y</td><td>11.475</td></tr>
<tr><td>speed</td><td>0</td></tr>
<tr><td>size</td><td>3</td></tr>
<tr><td>sides</td><td>3</td></tr>
<tr><td>colour</td><td></td></tr>
<tr><td>angle</td><td>0</td></tr>
</table></td>
<td> 

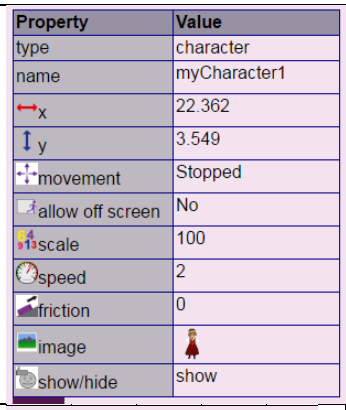
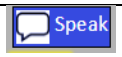
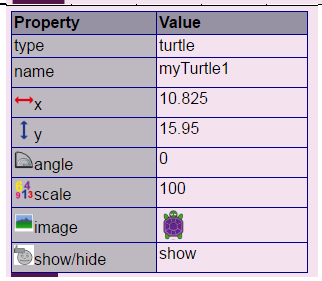
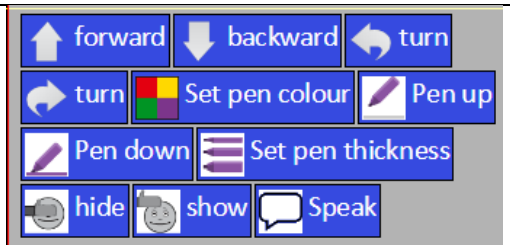
The options offered will depend upon the property selected.</td>
<td> 

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> 
<table>
<tr><th>Property</th><th>Value</th></tr>
<tr><td>type</td><td>vehicle</td></tr>
<tr><td>name</td><td>myVehicle1</td></tr>
<tr><td>x</td><td>8.921</td></tr>
<tr><td>y</td><td>13.336</td></tr>
<tr><td>allow off screen</td><td>No</td></tr>
<tr><td>rotation style</td><td>Face the angle</td></tr>
<tr><td>angle</td><td>0</td></tr>
<tr><td>speed</td><td>0</td></tr>
<tr><td>scale</td><td>100</td></tr>
<tr><td>image</td><td></td></tr>
<tr><td>friction</td><td>0</td></tr>
<tr><td>show/hide</td><td>show</td></tr>
</table></td>
<td> 

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

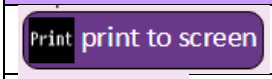
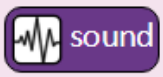
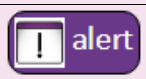
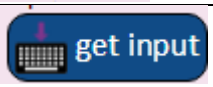
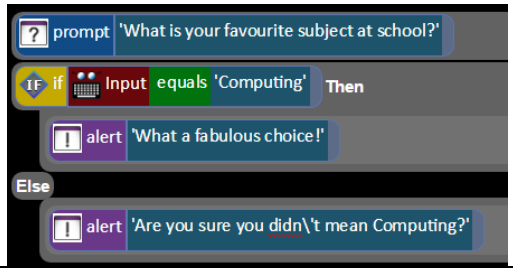

</td>
<td> 
<table>
<tr><th>Property</th><th>Value</th></tr>
<tr><td>type</td><td>character</td></tr>
<tr><td>name</td><td>myCharacter1</td></tr>
<tr><td>x</td><td>22.362</td></tr>
<tr><td>y</td><td>3.549</td></tr>
<tr><td>movement</td><td>Stopped</td></tr>
<tr><td>allow off screen</td><td>No</td></tr>
<tr><td>scale</td><td>100</td></tr>
<tr><td>speed</td><td>2</td></tr>
<tr><td>friction</td><td>0</td></tr>
<tr><td>image</td><td></td></tr>
<tr><td>show/hide</td><td>show</td></tr>
</table></td>
<td> 
</td>
<td> 

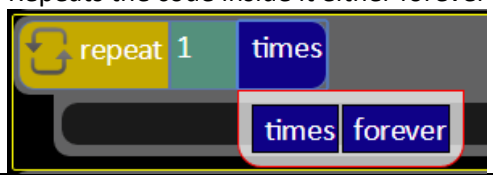
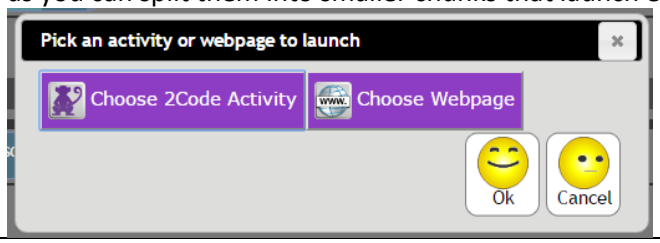
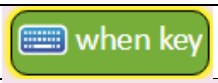
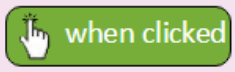
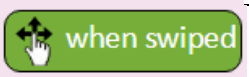
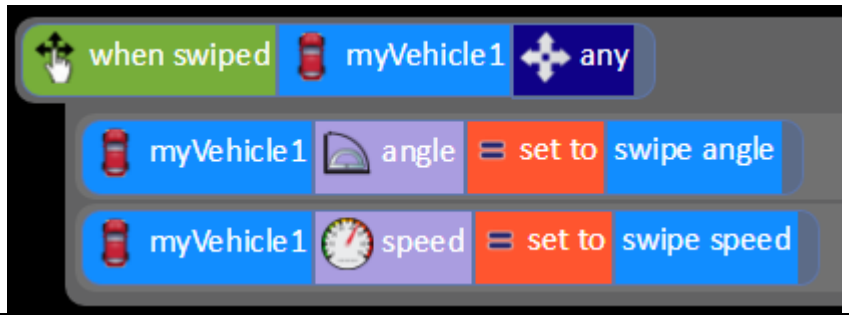
These make the object move in different directions.
Stop, hide or show the object.
Make the object speak by displaying a speech bubble.</td>
</tr>
</table>

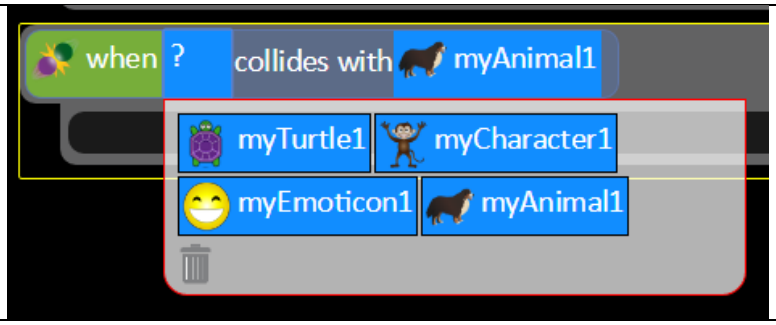
<table>
<tr>
<td> 
</td>
<td> 
<table>
<tr><th>Property</th><th>Value</th></tr>
<tr><td>type</td><td>character</td></tr>
<tr><td>name</td><td>myCharacter1</td></tr>
<tr><td>x</td><td>22.362</td></tr>
<tr><td>y</td><td>3.549</td></tr>
<tr><td>movement</td><td>Stopped</td></tr>
<tr><td>allow off screen</td><td>No</td></tr>
<tr><td>scale</td><td>100</td></tr>
<tr><td>speed</td><td>2</td></tr>
<tr><td>friction</td><td>0</td></tr>
<tr><td>image</td><td></td></tr>
<tr><td>show/hide</td><td>show</td></tr>
</table></td>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td> 
</td>
<td> 
<table>
<tr><th>Property</th><th>Value</th></tr>
<tr><td>type</td><td>turtle</td></tr>
<tr><td>name</td><td>myTurtle1</td></tr>
<tr><td>x</td><td>10.825</td></tr>
<tr><td>y</td><td>15.95</td></tr>
<tr><td>angle</td><td>0</td></tr>
<tr><td>scale</td><td>100</td></tr>
<tr><td>image</td><td></td></tr>
<tr><td>show/hide</td><td>show</td></tr>
</table></td>
<td> 
</td>
<td> 

A turtle moves in a similar way to a floor turtle using Logo type actions. Turn is by a number of degrees.</td>
</tr>
</table>

Appendix 3: Commands for Gorilla objects

Command	Explanation
	Prints some text specified by the coder to the screen.
	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.
	Creates a pop-up window with a message for the user and an OK button to click.
	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: 
	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.
	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. 
	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.
	This runs the code inside the first block if a certain condition is met, otherwise it runs the code inside the second block. 

	Repeats the code inside it either forever or every x (or quarter seconds). 
	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.
	This will restart the program from the beginning. Useful if you want to include a restart button in your program.
	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. 
	Runs the code inside it when the specified key is pressed. The code chooses which key (including arrow keys and space bar)
	Runs the code inside it when the object is clicked. The coder is given a choice of all the available objects.
	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. 
	Runs the code inside it when two objects collide. The coder selects the two objects.

	 when ? collides with myAnimal1 myTurtle1 myCharacter1 myEmoticon1 myAnimal1	

Assessment Guidance

The unit overview for Year 6 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 Gorilla 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 time 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 and 3). 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 will make good attempts to explain how programs simulate physical systems and can create their own program to meet a design brief relating to a physical system (Lesson 5/6).
Expected	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 Gorilla. For example, in Lesson 1, they have been able to make a program that follows their algorithm e.g. 'when the bubble is clicked it hides'. Children show an awareness of the

Assessment Guidance	
	need to be precise in their designs so that algorithms can be successfully translated into code. In lesson 3, children used a planning format on paper before implementing on screen within 2Code. Children' 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. They can use the Design Mode within 2Code to carefully see how their planned program will look and are able to switch into Code Mode to apply actions to objects. They confidently include objects, actions, events and outputs successfully within their 2Code programs. Children experiment with the use of timers to achieve repetition effects in their programs – they can determine whether a timer should be called every x seconds or after x seconds and the difference between the two (lesson 2). They are beginning to understand the difference in effect of using a timer command rather than a repeat command when creating repetition effects in their coding (lesson 2). Children can use 'if' statements to bring selection into their own coding (lesson 3). They understand how variables can be used to store information while a program is executing (lesson 4) 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. Children can predict program outcomes and attempt to debug. Children can identify the parts of a program that respond to specific events and initiate specific actions. Based on this, children can predict and describe, using a cause and effect sentence, what will happen in a program. They make use of user input and outputs such as 'print to screen' (lesson 4) as well as sound and movement of objects. Children can explain how programs simulate physical systems and can successfully create their own program to meet a design brief relating to a physical system (Lesson 5/6).
Exceeding	Children are attempting to turn increasingly complex real-life situations into algorithms for a program by deconstructing the situation into manageable parts. Children' design shows that they are thinking of the required task and how to accomplish this in code. Children can identify an error within a program that prevents it following the desired algorithm and then fix it (all lessons). Children make intuitive attempts to debug their own programs as they increase in complexity. Children realise the constraints of creating purely sequential programs and intuitively grasp the concepts of selection (lesson 3) and repetition (lesson 2). Children have a good understanding of when to use a timer in a program rather than a 'repeat' command to for repetition and this is evidenced in their program designs. Children

Assessment Guidance	
	make use of variables in their programs and combine these with timers to creative effect (lesson 4). Children like to challenge themselves to combine these with other coding structures to achieve the effects that they design to personalise and to improve their programs (lessons 4, 5 & 6).