purple mash

Computing
Scheme of Work

CRASH COURSE

Coding Unit for Children in Year 6
Who haven't used 2Code in Years 1 - 5

when clicked

Print print to screen

Year Group: 6
Number of Lessons: 6

From 2simple

Contents

Introduction

The crash course is intended to replace unit 6.1 – Coding for classes who have not completed the previous Purple Mash Coding units from the scheme of work.

Differentiation

The 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 variables and If/else statements.

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

The crash-course aims to give children a good grounding in coding using 2Code. It covers the Y1-Y5 material but not the Y6 coding unit (6.1). 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.

Program Design

To master coding skills, children need to have the opportunity to explore program design and put computational thinking into practice. The crash course suggests some designing before coding in the plans. Children could do this through:

- Storyboarding their ideas for programs. For example, creating a storyboard when planning a program that will retell part of a story.
- Creating annotated diagrams. For example, creating an annotated diagram to plan a journey animation that tells the story of an historical event they have been studying.
- Creating a timeline of events in the program. For example, creating a game program against the computer, what are all the actions needed from the objects?

During the design process, children should be encouraged to clarify:

- the characters (objects and their properties)
- what they will do (actions and events)
- what order things will happen (the algorithm)
- rate their confidence at being able to code the different parts of their design and either refine the design or review possible solutions as a class or group.

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 practice 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	Covered by guided activities in 2Code	How to provide additional opportunities
Most scaffolded	Copying code	✓	
	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 →	• 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	Guided activities ✓ Guided challenges at the end of each guided activity ✓ Free code ✓ In Literacy, some teachers follow a progression that scaffolds learning to write texts. At first pupils 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.	**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. • 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/

Year 6 Crash Course – Medium Term Plan

Lesson	Aims	Success Criteria
1	To explain what coding is. Introduction to the 2Code interface including the possible actions of character, car and animal objects. Tinkering with 2Code	• Children can explain that coding is how computer programs are created. • Children can navigate around the 2Code interface, dragging and dropping code blocks and running code.
2	To create a program with an object that repeats actions indefinitely. To use a timer to make characters repeat actions. To explore the use of the repeat command and how this differs from the timer.	• Children can show how their character repeats an action and explain how they caused it to do so. • Children are beginning to understand how the use of the timer differs from the repeat command and can experiment with the different methods of repeating blocks of code. • Children can explain how they made objects repeat actions.
3	To introduce If statements to allow selection in a program.	• Children can create an 'if' statement in their program. • Children can use a timer and 'if' statement to respond to the actions of a character and change their actions.
4	To understand what a variable is in programming. To use a variable to create a visual timer. To explore number and string variables.	• Children can explain what a variable is in programming. • Children can explain why variables need to be named. • Children can create a variable in a program. • Children can set/change the variable values appropriately
5 & 6	To go through the design, code, execute, refine process. To use the coding skills that they have encountered creatively in their own program. To create a program that controls or simulates a physical system, i.e. changing the speed and angle of moving objects.	• Children have an idea about the design process and its benefits. • Children have turned a design into a functioning program. • Children can explain how their program simulates a physical system, i.e. objects move at different speeds and angles, what they did to make their vehicle change angle, show that their vehicles move at different speeds.

Lesson 1 – Introduction to coding with 2Code

Aims

- To explain what coding is.
- Introduction to the 2Code interface including the possible actions of character objects.

Success criteria

- Children can explain which commands they included in their program and what they achieve.
- Children can explain what Object, Action, Output, Control and Event are in computer programming.

Resources

Unless otherwise stated, all resources can be found on the main unit 6.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you don't lose this page.

- Bubble program design – to be displayed on the whiteboard or printed for display.
- (Optional) Exercise books to be used as 2Code workbooks for recording coding exercises, if desired.
- **Note**: Coding vocabulary is highlighted in bold in this lesson plan, this vocabulary should be used so that children pick up the appropriate coding vocabulary in context.

Activities

1. Explain to the children that we are going to be coding. Ask them if they know what coding 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?

2. Start off by doing some activities where the children must follow or give clear instructions.

3. Choose two children; one is a robot and the other is a **coder**. The coder needs to direct the robot to walk from one place in the classroom to another. How can they give the instructions so that the robot does not crash into objects in the way? Repeat a few times in different locations.

4. Tell the children that you are now going to be the robot and they are the coders. Stand by the whiteboard and ask the children to give you clear instructions, one step at a time, for drawing a smiley face.

5. Once you have a set of clear instructions, introduce the term **algorithm**.

   > A precise step by step set of instructions used to solve a problem or achieve an objective.

6. The children have created an algorithm to draw a smiley face.

7. Discuss the way that coding languages use symbols rather than whole sentences; they turn the **algorithms** into code. Can they come up with their own symbols for the instructions for drawing a smiley face?

For example, holding their fingers in a circle shape for the face and combining this with a symbol for 'small' for the eyes. These symbols could be written on the whiteboard to create a program.

This can become more complex, depending upon the understanding and interest level of the children. Some children will enjoy working out how their 'code' could be adapted to draw a sad or sleeping face. Some children will want to be precise about the placement of the shapes, e.g. a symbol to show putting the smile *below* the eyes.

8. Show the children the design diagram and explain that, together, you are going to make a program using 2Code that follows this design. Talk through the diagram so that children understand what the program will do.

9. Put Free Code Gorilla on the board and point out the **code blocks** on the left-hand side. The first thing to do is to set up the look of the program in deign view. Click on the ![Design] button. The program is created by adding **objects** and then writing code for what these **objects** do. The different object types in 2Code Gorilla are on the left-hand panel:

Background

10. To add an **object**, you drag it onto the screen. Drag a character into Design View.

11. Explain that the background object is always added. You could click on ![screen] to see its **properties:**

Property	Value
name	background
colour	
image	?
Tiling	stretch
Grid size	3
Grid origin	Top Left
Show grid	No

12. Change the background by double-clicking on the question mark, this opens the Clipart picker where you can select an existing background, upload an image from your device or paint an image. Choose the underwater background from the existing backgrounds.
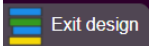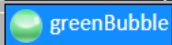
13. Click on the character **object** that you added and change the name **property** to 'greenBubble':
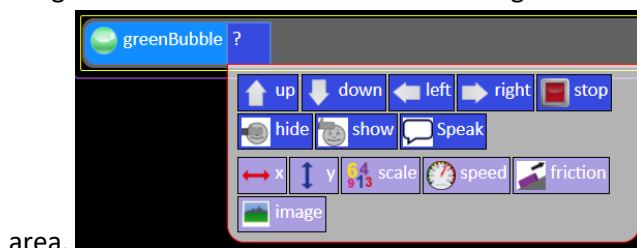
Property	Value
type	character
name	greenBubble
x	16
y	7
movement	Stopped
allow off screen	No
scale	100
speed	2
friction	0
image	
show/hide	show

14. Next, double-click on the image **property** to open the clipart picker. You will find the green bubble image in the bubble section of the clipart picker (use the drop-down box at the top). The bubble will probably be too big, so look for the scale **property** and click on it to reduce the scale of the bubble. Add the other required objects and name them appropriately referring to the design diagram. Note: the fish should be an animal object type and the image can be found in the sea section of the clipart picker.

15. Show children how to save their work into their work folders and remind them of the need to save regularly with sensible names to avoid losing their work.

16. Return to Code View [Exit design] and point out that the added **objects** now appear on the left of the screen below the existing code blocks.
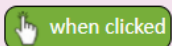
17. Drag a [greenBubble] block into the coding area of the screen; this is the grey bar at the top of the main area.
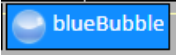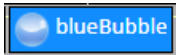
18. Choose an **action** for it e.g. [up].

19. Now click on the [▶] button to run the code and you should see the green bubble moving up. Ask the children why the other objects are not performing any **action**? (a program will only do what you code it to do – you haven't written any code for the other objects yet). Click [■] to return to the coding view

20. Drag a [when clicked] block below the line of code that you just wrote. This is called an **event** block; when an **object** is clicked it triggers an **event** (the click event) which will have an **action**
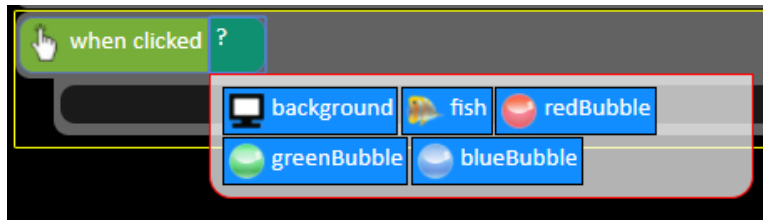
21. Look at how this **event** works:

22. You select the **object** that will be clicked – refer to the design to see that when the blue bubble is clicked it should go up. Select

23. You then drag in the **object** that will perform an **action** when the first object is clicked into the indented part of the code window – emphasise the importance of this to the children - select again.

24. You then select an **action** for that object – select

25. What do children think will happen when the code is run now? Save the code and see if they predicted correctly; the green bubble will move up, when the blue bubble is clicked, the blue bubble will move up.

26. See whether children can return to their own devices and create the rest of the design themselves.

27. Once they have done this, they should have some time to have a **tinker** with 2Code and see what they can discover. This tinkering could be completely free, or you could give some directives as suggested below:

    - Add different types of objects and see how they have different actions e.g. compare character, vehicle and turtle objects – see Appendix 2 for details of what these are.

    - Work out what each of the Commands does (the code blocks at the top above the objects) – see Appendix 3 for details.

    - Try adding sound when objects are clicked

    - Try to work out how a timer or a repeat block works

28. If the children have workbooks, they can print their code and write about what they have discovered and what they would like to try doing next.

Lesson 2   Repetition Commands

Aims

- To create a program with an object that repeats actions indefinitely.
- To use a timer to make characters repeat actions.
- To explore the use of the repeat command and how this differs from the timer.

Success criteria

- Children can show how their character repeats an action and explain how they caused it to do so.
- Children are beginning to understand how the use of the timer differs from the repeat command and can experiment with the different methods of repeating blocks of code.
- Children can explain how they made objects repeat actions.

Resources

Unless otherwise stated, all resources can be found on the main unit 6.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you don't lose this page.

- Vocabulary flash cards.
- Example programs from Crash Course Examples
  - character repeat using timer
  - character repeat forever using timer
  - timer dance forever
  - character repeats using repeat command.
  - turtle repeat dance forever
- Worksheet 'What does it do?' You could set this as a 2Do or print in colour.
- 2Code Freecode **Gorilla** tool.

Activities

1. In this lesson, we will be learning some new vocabulary relating to programming. On the board, go through the terms: Sequence, Repeat, Input and Output.

    Sequence - When a computer program repeats a sequence of commands. In 2Code this could be done using "REPEAT", "REPEAT UNTIL" or using a "Timer"

    Repeat – In 2Code a "repeat" command can be used to make a block of commands run a set number of times or to repeat a block of commands forever.

    Input – Information going into the computer. An input could be user the moving or clicking the mouse, or the user entering characters on the keyboard. On tablets, there are other forms of input such as finger swipes, touch gestures and tilting the device.

    Output - Output is information that comes out of the computer. This could be items that appear on the screen or sound that comes out of the speakers. Examples of output are "Print to screen" and "Sound".

2. Firstly, we are looking at making objects repeat actions using a timer. Open the example program character repeat using timer on the whiteboard. The character repeats the action of going up and down twice. Have a look at how this is done in the code.

How could we make the character go up and down forever? It would be impossible to keep adding the lines …..
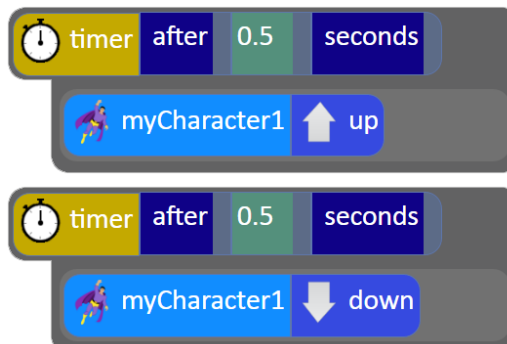
…….inside each other forever.

3. To make the timer repeat forever, you first need to work out how often to repeat the block of code. In the example case, every 1 seconds should be the right amount because he goes up for 0.5 seconds then down for 0.5 seconds then we want him to repeat this again.

Open the example character repeat forever using timer. Look at the first line of code
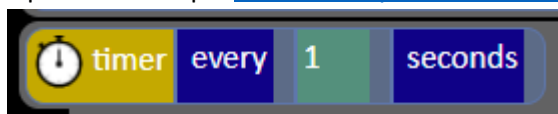
Note that the timer says '**every**' rather than '**after**':

Inside this timer is the code to repeat:
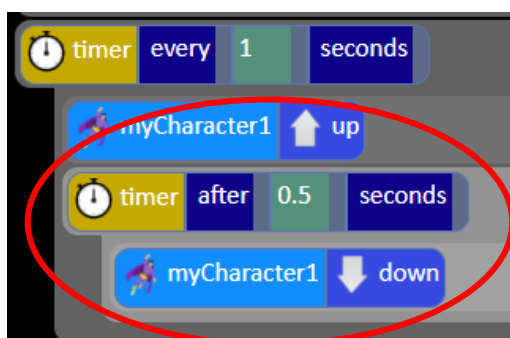
4. Another example of a character repeating actions forever is in the example file timer dance forever. Look at this with the children, can they 'read' the code to see how the actions are repeated?

5. Give children time to experiment with making a small program using the timer to repeat actions. They should briefly plan their program by drawing a labelled diagram or a storyboard before coding it.

6. Once they are ready, bring the class back together to discuss the following:

7. Another command that is used to repeat blocks of code is the [repeat] command. It might seem that you could just use this instead of the timer. However, the repeat command is designed to perform an action (run a block of code) many times as quickly as possible. This is one of the big advantages of using a computer to do tasks; it can be programmed to perform complex calculations much faster than a human can. Therefore, the repeat command might not always be the best choice.

8. Look at the example code 'character repeats using repeat command'.

9. When you run it, the code will be performed so fast that you can't actually see the actions! You can try running the code in slow mode and you'll see that the actions do happen.

   Use the slider bar at the bottom right to alter this before you press 'OK' on the message at the start:

   So, in this case, the Repeat command isn't really the best one to use.

10. If the children have done any work with Turtle objects and Logo, you might want to show them the additional example of using the repeat command in the way that they would with a turtle object. It does work, just too fast to use in a visual program. An example of this is in turtle repeat dance forever.

11. The 'What does it do?' sheet asks children to predict what code will do when run, run the code to check their predictions and then use the code to make a better program. All code snippets use repeat; they are reproduced here:

Sentence generator using repeat command.

This instantly generates sentences by combining random adjectives, animals and verbs.

Welcome you
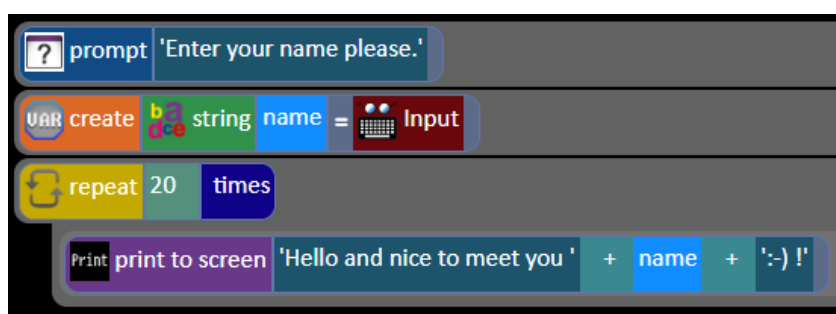
To recreate this example, children will need to use the [create variable] block. For the time being, do not focus on exactly what a variable is (they will cover this in lesson 4 as it would be too much to squeeze in here). You might want to walk them through the process of creating a variable like the example, this is detailed in Appendix 4 where the process for combining text and variables in 'print to screen' responses is also covered.

Number sequence maker using user input
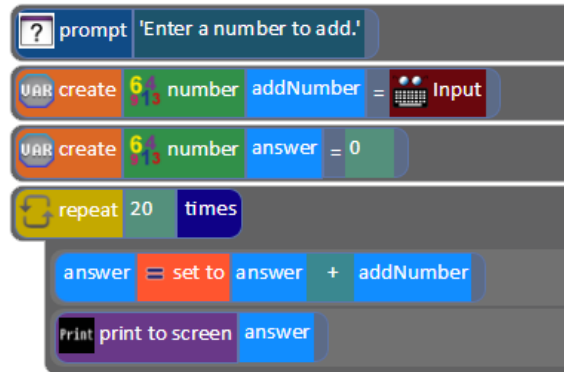
12. Give children time to experiment with programs using the repeat command to repeat actions. Could they write a program that asks the user for a number and then lists the first 50 numbers in that times table?

Lesson 3 – 'If' Statements for selection

Aims

- To create a program that responds to the 'if' command or the 'if/else' command
- To use selection within a program.

Success criteria

- Children can create an 'if' statement in their program.
- Children can create an 'if/else' statement in their program.
- Children can use a timer and 'if' statement to respond to the actions of a character and change their actions.

Resources

Unless otherwise stated, all resources can be found on the main unit 6.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you don't lose this page.

- Vocabulary flash cards.
- Blank printable storyboards for designing.
- Program Design Examples as a reference to possible design style examples.

Activities

1. Review the vocabulary that children have been introduced to so far, specifically, this includes:

- algorithm
- object
- property
- event
- action
- timer
- repeat

Though they will be familiar with many other terms now such as 'block', 'coder', 'bug' etc

Explain that so far, the children's programs have all followed a **sequence** of events and some have included **repetition** (using repeats and/or timers). Show the **Sequence** vocabulary card. Now we are going to introduce some **selection** into their coding.

2. Show the **selection** vocabulary card. Then introduce the new vocabulary that they will be learning about today: **'if' statements** using the vocabulary card. Compose some 'if statements' as a class e.g. 'if you listen well then I will be happy', 'if there is rain then it will be wet play' etc.

3. Open the activity 'Guard the Castle 2' in the **Gibbon** activities and do Step 1 together.
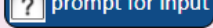
4. In Step 2, you must create a timer which checks the X position of the knight every second; **if** the knight's position is greater than 15, **then** he should change direction. Switch to design view and point out the x
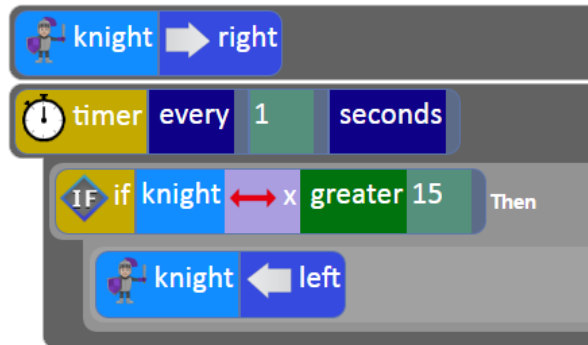
and y positions displayed in the bottom left corner. Drag the knight around and see how they change. This is to help the children gain an understanding of how measuring the X position can decide whether the knight should change direction.

5. Do Step 3 together, adding another 'if' statement. Talk about how this should be indented at the same level as the first 'if' statement so it is also inside the timer.

6. Now open free code Gorilla and drag a block into the coding window. Type a question into the input box such as 'Do you like computing?'

7. Drag in an **IF if** as follows:

```
? prompt 'Do you like computing?'

IF if  Input equals 'yes'  Then

    Print print to screen 'That\'s great!'
```
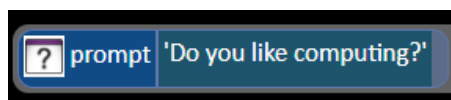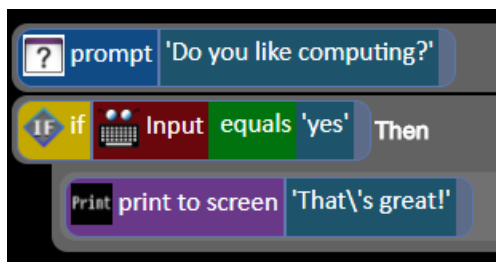
8. Run the code and answer 'yes' then run again and answer 'no'; nothing happens. Discuss that this is not a good response. How could we change it so the computer does something if the person enters 'no'. If children do not notice the **IF if/else**, point it out to them. Show them the definition of if/else using the vocabulary cards.

9. Replace the **IF if** with **IF if/else**:

```
? prompt 'Do you like computing?'

IF if  Input equals 'yes'  Then

    Print print to screen 'That\'s great!'

Else

    Print print to screen 'Oh dear!'
```
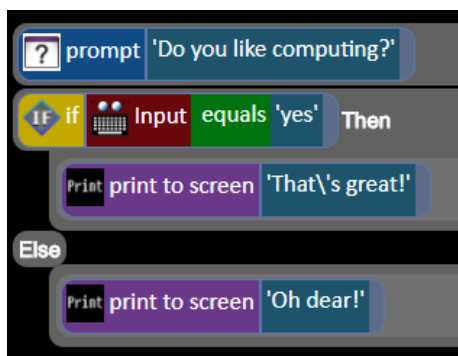
10. Run the program again and on the third time enter gibberish to the question. The answer won't be appropriate, what could we do? Here is a possible solution; talk this through with the children.

```
? prompt 'Do you like computing?'

IF if  Input equals 'yes'  Then

    Print print to screen 'That\'s great!'

Else

    IF if  Input equals 'no'  Then

        Print print to screen 'Oh dear!'

    Else

        Print print to screen 'Sorry, I don\'t understand. Please enter \'yes\' or \'no\'.'
```
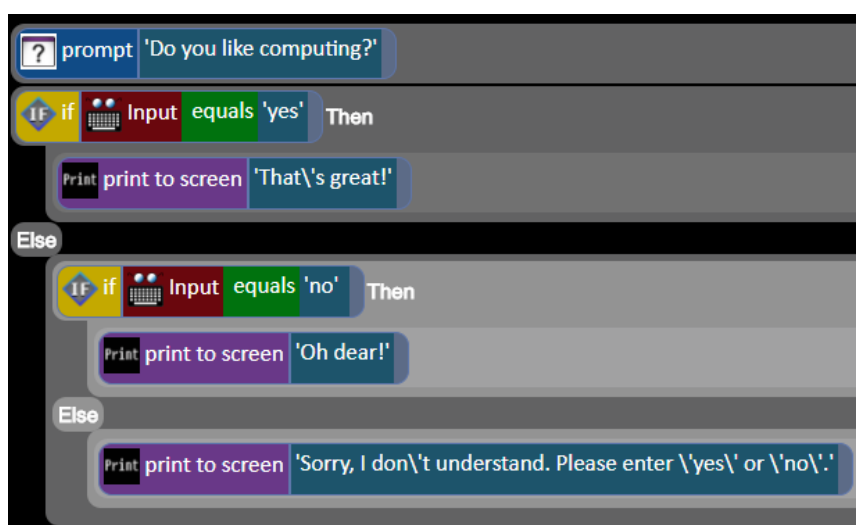
11. Children should **plan** a program that uses 'if' or 'if/else' blocks. They could use user input or a more visual program like Guard the Castle. They could even make a simple chat robot using a character object who speaks:
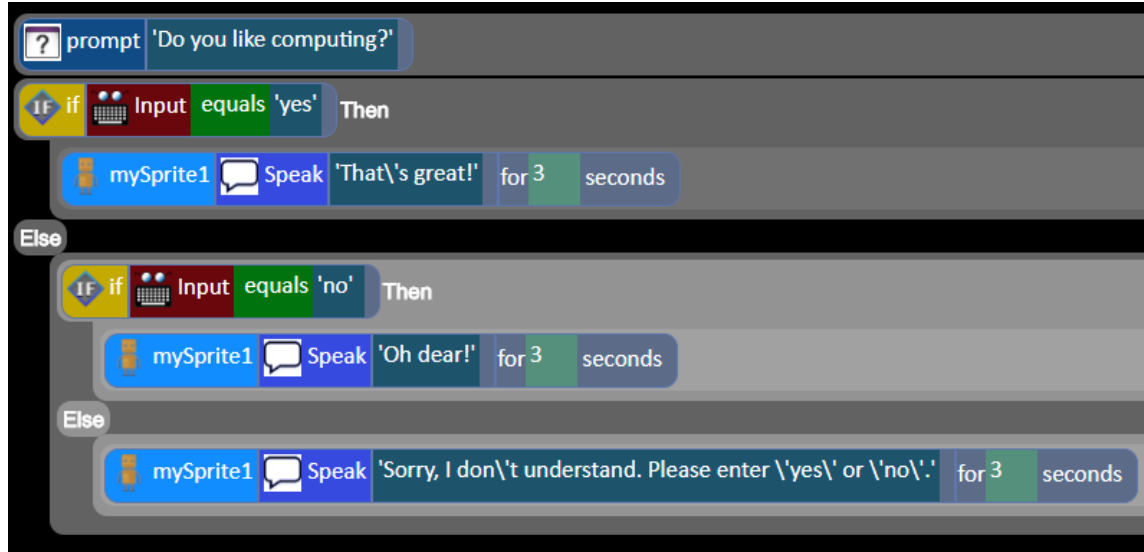
A note about design: encourage children to think through their designs and annotate them including their confidence in coding what they have designed (red, amber, green), this will give you feedback on areas that children need help with and help to ensure that children create realistic designs and successful programs for their skill level. Use the Program Design Examples as a reference to possible design style examples.
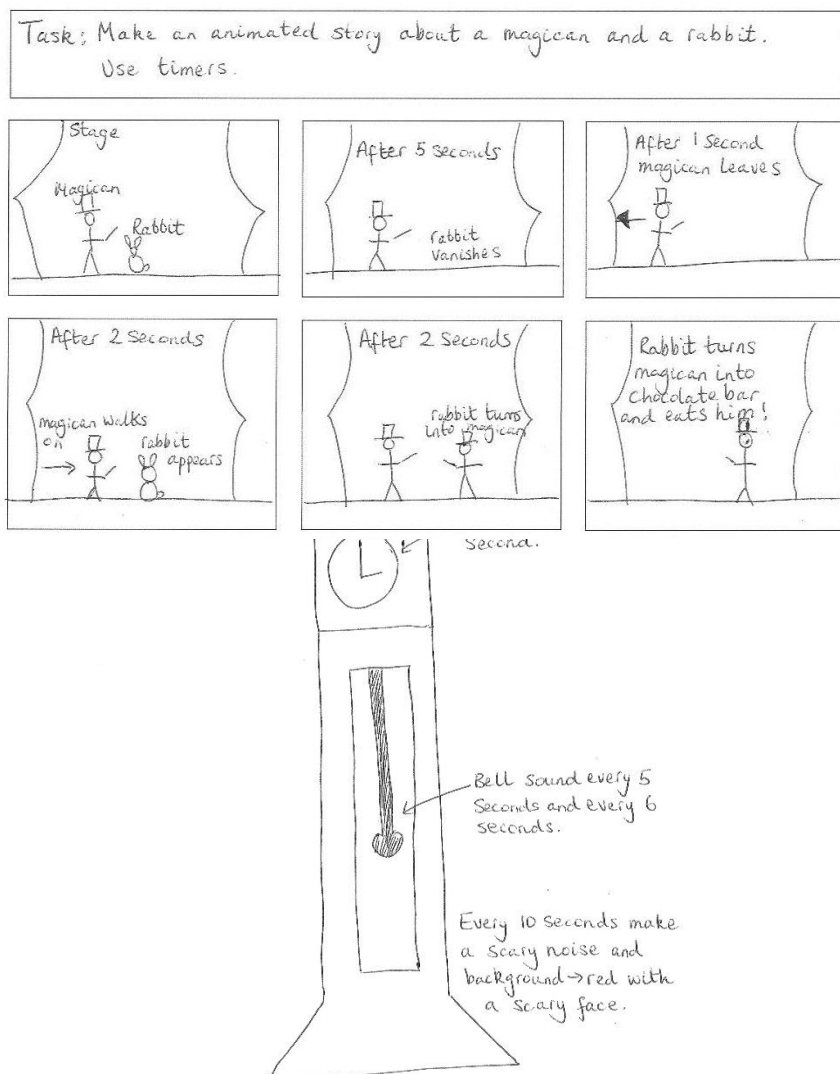
Lesson 4 - Introducing Variables

Aims

- To understand what a variable is in programming.

- To use a variable to create a visual timer.
- To explore number and string variables.

Success criteria

- Children can explain what a variable is in programming.
- Children can explain why variables need to be named.
- Children can create a variable in a program.
- Children can set/change the variable values appropriately

Resources

Unless otherwise stated, all resources can be found on the main unit 6.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you don't lose this page.
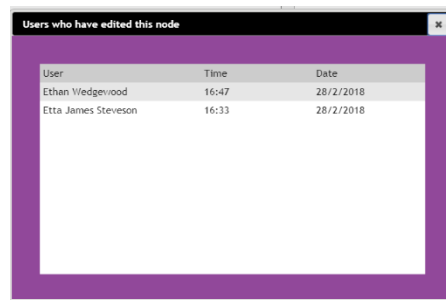
- Vocabulary flash cards.
- Example code snippet

Activities

1. Today we will be working with variables. Use the vocabulary flash card to give a definition of the word

2. Explain the following:

> Variables are like boxes in which the computer can store one piece of information.
>
> To find the information from the right box, each box should be labelled. This reserves some space in the computer memory for this variable.
>
> Therefore, each variable (each of our boxes) needs to have a name. The name should be something that helps you remember what it is.
>
> The information inside the box is called the **Variable Value**. The user or the program can change this Variable Value.
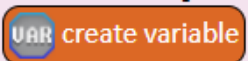
3. In 2Code, variables can either be numbers, strings (words, phrases or even whole sentences) or functions.

4. Open Free Code Gorilla on the IWB and look at the orange variable buttons in the menu on the left-hand side.

5. Drag a **create variable** block into the code window. The drop-down menu gives three options; **number** **string** **function**. For today, we are going to be looking at number and string variables.

6. Choose 'Number'. Give the variable the name 'counter'. Ask children why we are naming the variable?
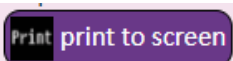
7. We can either define the variable as a specific number or set it to Random. For this example, we will set the variable to 0 as the counter will start from 0.
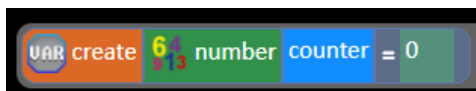
8. Drag in a timer and change 'After' to 'Every'

9. Drag **change variable** into the timer. We are using the timer as this is an easy way for us to get the variable to continue changing all the time. Select your variable from the drop-down menu. Set it to Add 1 every second.

10. Drag a **print to screen** block into the timer as well and select your variable.

11. Run the code and show children that it is printing every number that it changes

12. Have a look at the bottom left-hand corner of the screen as the timer counts. This is the Variable Watch window. It tells us what variables we have in our program and what they are doing. Variable Watches are common in programming environments to help programmers understand what the computer is doing and to debug their code if necessary.

13. Depending upon ability, children should now try to create a program that either replicates the above or uses a number variable to print their choice of times table to the screen. Some children might be able to work out how the user could select the times table to print. Here are some possible solutions:

14. Bring the class back together to look at string variables

15. . Display the example code snippet. But before running the code, can the children suggest what the code will do?

16. Things to note with the children:

- Use of the Random function – what does random mean?
- What does the + sign do to text? What would 'Hello + World' produce?
- The way 2Code can select a random animal, noun, verb or adjective in order to build sentences.
- The importance of spaces, the spaces are not correct on the example, can you debug the code as a class? Solution is:

- Try setting a string variable to '1' (the quote marks are required) and + 2 (see snippet below); can children guess what this will make? How would you code this so it adds 1+2?
Snippet:

Solution:

17. Give children some time to explore text and number variables. What ideas can they come up with to make funny messages?

18. Here are some examples of variables in use in the guided activities, you could look at these as a class or ask children to try the activities:

- the on/off state of a switch (see Switching Background Gibbon lesson)
- counting the number of swipes before changing a lamp into a genie (see Gibbon Genie lesson)
- the numbers changing in a timer (see Night and Day Gibbon lesson).
- Send the rocket into space (see Gorilla lessons)
- Catching Game (Gorilla)
- Driving Game (Gorilla)
- the counting machines in Coding Principles (on the 2Code main page in the section called Coding Principles).

Lesson 5 and 6   Designing a game that simulates a Physical System

Aim

- To go through the design, code, execute, refine process.
- To use the coding skills that they have encountered creatively in their own program.
- To create a program that controls or simulates a physical system, i.e. changing the speed and angle of moving objects.

Success criteria

- Children have an idea about the design process and its benefits.
- Children have turned a design into a functioning program.
- Children can explain how their program simulates a physical system, i.e. objects move at different speeds and angles, what they did to make their vehicle change angle, show that their vehicles move at different speeds.

Resources

Unless otherwise stated, all resources can be found on the main unit 6.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you don't lose this page.

- Flashcards as before.
- Gorilla guided 2Code activity Football – this is on the main 2Code page (Tools → 2Code).
- Exercise books to be used as 2Code workbooks for recording coding exercises, if desired.
- Blank printable storyboards for designing.
- 2Connect file saved in collaborative mode: The user guide for 2Connect can be found at https://www.purplemash.com/site#app/guides/2Connect_Guide . To make a 2Connect file collaborative, first save it in a shared folder – the image below shows saving a 2Connect file called Physical system coding in the Ants class folder.
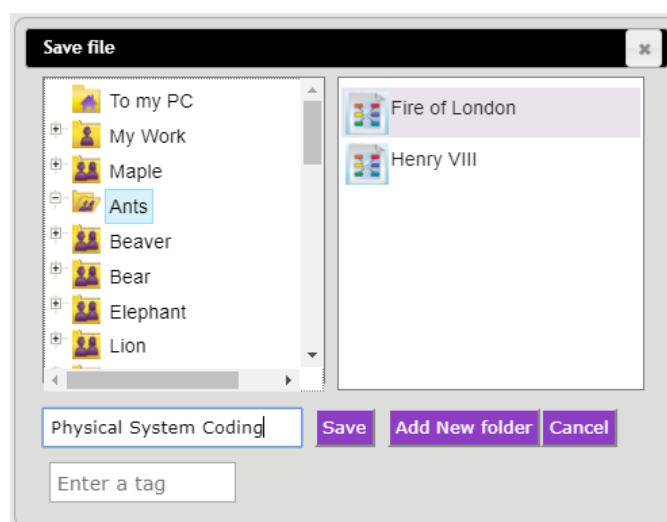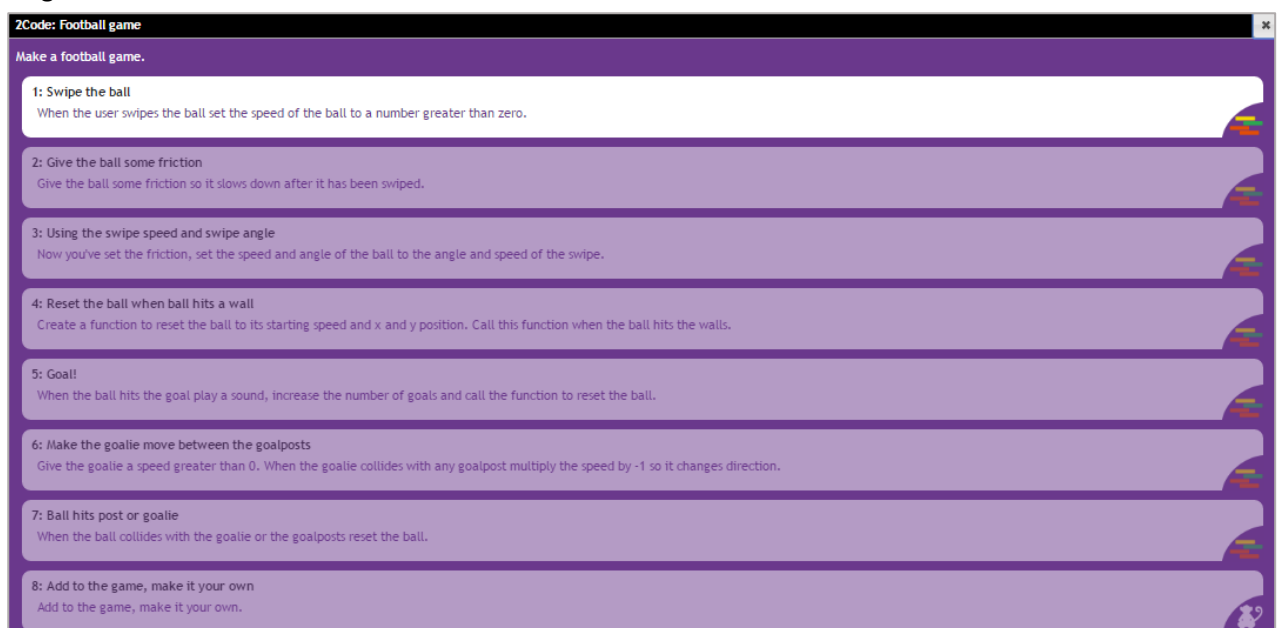
Next, click [icon] in the top menu bar and the image will change to [icon] (though you might be prompted to save it again first) this means that all children from the Ants class will have access to it (in their work area they can navigate to the class folder) and can collaborate on the file. When you open a node (bubble with text in) by clicking on the pencil symbol on the node, you can click on the [icon] button to see who has edited the node and contributed to the work.

Activities

1. Today we will be creating a program that simulates a physical event. Who can guess what that means? Children discuss in pairs for a minute. Explain that for our program, simulating a physical system means using angles or speed to change the way an object moves, for example, the way someone kicks a football determines how far and how high the ball goes.

2. Look at the stages of the Football game in Gorilla activities (do not do the activity yet). This game simulates kicking the ball and the movement of the ball. The stages show how this is achieved in small steps, each one adding to the physical simulation, e.g. simulating kicking the ball, simulating the effect of friction, simulating the direction and speed of the ball, the effect of the ball hitting a wall, scoring a goal and the movement of the goalie.

3. Work through the first three stages of the football game to demonstrate how to set the speed and angle of a vehicle object.

4. On the whiteboard, open Free Code Gorilla and go to Design Mode. Drag in a vehicle, double click on it and show children that they can change it into something else if they like by altering the image. Look at the choices of images using the drop-down box of categories.

5. This should spark the children's imaginations about what their program can do.
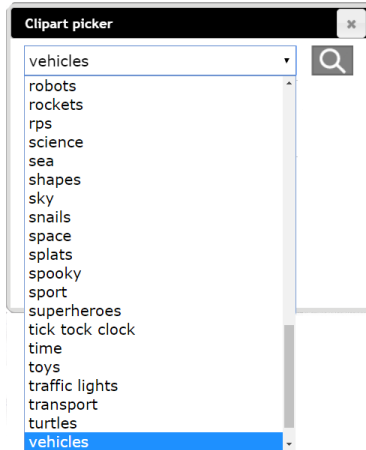
6. Open the 2Connect file and explain to children how to add/edit nodes and join nodes.

7. Show the children how to open the same file from their devices.

8. Spend a few minutes brainstorming ideas for possible programs – encourage children to enhance one another's ideas. If children are struggling, you could use the following starters and ask children to elaborate on what could be coded on these themes:

   - other ball games
   - airport simulations
   - Space or rocket simulations
   - racing simulations
   - snail races
   - science related simulations of objects moving on different surfaces
   - anything else where children will be able to experiment with changing angle, speed and friction.

9. You might want to display the coding vocabulary that they have encountered to help them plan

10. Children should then use the storyboards to plan and annotate their designs. See the following notes on design:

A note about design: encourage children to think through their designs and annotate them including their confidence in coding what they have designed (red, amber, green), this will give you feedback on areas that children need help with and help to ensure that children create realistic designs and successful programs for their skill level

11. Over the two sessions children should have time to code and refine their design. You could spend additional time on this or ask children to continue at home. Children could also blog about their coding and/or add finished programs to a displayboard.

Task: Make an animated story about a magican and a rabbit. Use timers.

Stage
Magican
Rabbit

After 5 seconds
rabbit vanishes

After 1 Second magican leaves

After 2 Seconds
magican walks on   rabbit appears

After 2 Seconds
rabbit turns into magican

Rabbit turns magican into chocolate bar and eats him!

Task: To make a ticking clock with Sound effects using a timer

Tick sound every second.

Bell sound every 5 seconds and every 6 seconds.

Every 10 seconds make a scary noise and background → red with a scary face.
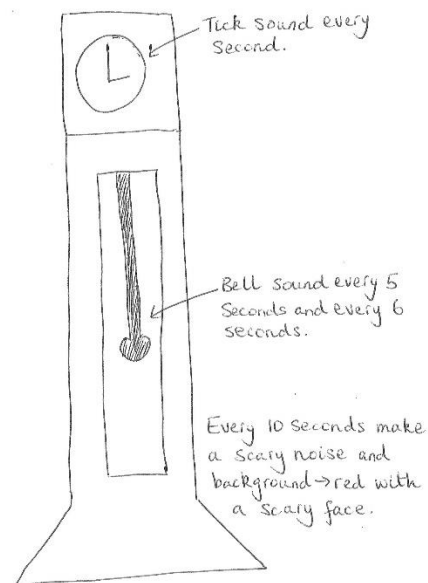
Appendix I: Other features of 2Code

Real Code Mode:
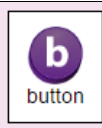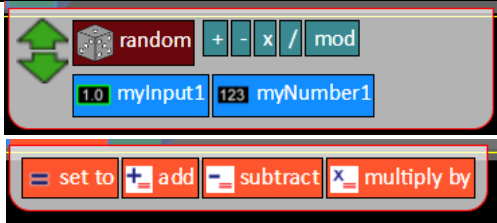
On the more advanced lessons and Free Code modes it is possible to click the real code button and see the code in a simple subset of Javascript. The code can be edited in "real code" mode and clicking the "edit blocks" button will bring the user back to the usual graphical representation. If the user types code into real code window that is syntactically incorrect the real code window will flash red. Changing back to "edit blocks" will restore the code to the last state that was syntactically valid.

Sharing:

Once a 2Code program has been saved into Purple Mash, it can be shared by clicking on the globe icon in the toolbar. This will display a window with a hyperlink and an embed code. Clicking the hyperlink will launch the 2Code program in a web browser in fullscreen mode and the embed code can be used to embed a 2Code program into a blog or a website. A Purple Mash login is not required

to run a shared 2Code program.

Appendix 2: Actions for Gorilla objects

Object	Properties in Design View	Properties in Code View	Actions in code view
(monitor icon)	**Property** / **Value** type — background name — background colour — (black) 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.
button	**Property** / **Value** type — button name — myButton1 x — 3.825 y — 5 text — myButton1 text size — 16 text colour — (white) background — (grey)	None	None
123 number	**Property** / **Value** type — number name — myNumber1 value — 0 text size — 18 text colour — (white) border — No x — 8.775 y — 3.8		random + - x / mod 1.0 myInput1 123 myNumber1 = set to + add - subtract x multiply by 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.
1.0 input	**Property** / **Value** type — number name — myNumber1 value — 0 text size — 18 text colour — (white) 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.

Property	Value
type	text
name	myText1
background	
border width	0
border colour	
text size	26
text colour	
text align	left
font	sans-Serif
x	8.822
y	6.381
show/hide	

Show/Hide

Property	Value
type	shape
name	myShape1
x	21.175
y	11.475
speed	0
size	3
sides	3
colour	
angle	0

The options offered will depend upon the property selected.

These make the object move in different directions.

Stop, hide or show the object.

Make the object speak by displaying a speech bubble.

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

The options offered will depend upon the property selected.

Speak

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

These make the object move in different directions.

Stop, hide or show the object.

Make the object speak by displaying a speech bubble.

Property	Value
type	character
name	myCharacter1
↔x	22.362
↕y	3.549
movement	Stopped
allow off screen	No
scale	100
speed	2
friction	0
image	
show/hide	show

| ↔x | ↕y | angle | speed | scale |
| image | friction | show/hide | | |

Speak

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

| ↔x | ↕y | angle | scale | image |

forward   backward   turn

turn   Set pen colour   Pen up

Pen down   Set pen thickness

hide   show   Speak

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
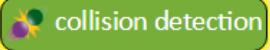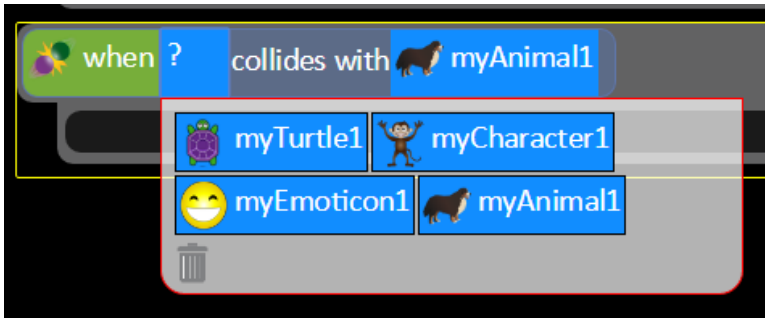
Command	Explanation
Print print to screen	Prints some text specified by the coder to the screen.
sound	Causes a sound to play. the sound picker will open for the coder to select a sound, when this code block is added to the code window.
! alert	Creates a pop-up window with a message for the user and 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: alert 'Please enter your name' get input Print print to screen 'Hi there ' + Input + '!'
? 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. timer after 1 seconds after every
IF 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 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.

	prompt 'What is your favourite subject at school?'
	
	 IF if Input equals 'Computing' Then
	
	 alert 'What a fabulous choice!'
	
	 Else
	
	 alert 'Are you sure you didn\'t mean Computing?'

repeat	Repeats the code inside it either forever or every x (or quarter seconds). repeat 1 times times forever
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. **Pick an activity or webpage to launch** Choose 2Code Activity Choose Webpage Ok Cancel
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.

Appendix 4 – Creating variables

If children are trying to recreate the examples in lesson 2, they will come across the lines:
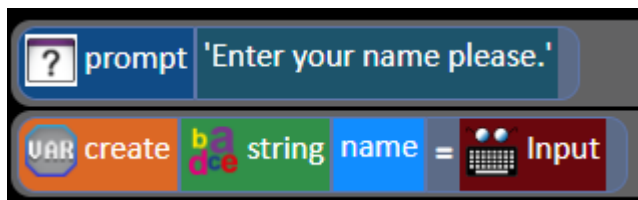
They will not have encountered these blocks before so might need some help to recreate them.

The process they are following is to create variables and these are covered in more detail in a future lesson.

1) In the above example they firstly need to drag in the block .
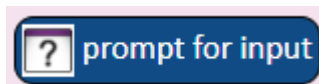
    This block asks the user of the program a question - children type the question in. It then stores the result as .

2) Then they should drag in the block

     . 2Code will ask whether this should be a
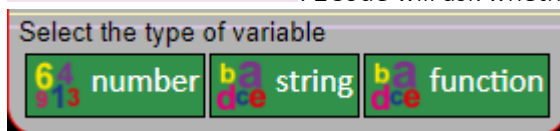
     . For the example above select .

3) 2Code will name the variable myString1 because all variables must have a name. The name can be changed by deleting it and typing in a different name.

4) Click in the final box to see the variable to .
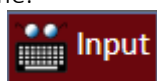
5) In some examples the command is used to give the user information about the variables, sometimes this is combined with text. Use the options that pop-up to help you. Any variables that have been created will pop-up here, or you can type into the box. You can combine typing and variables by
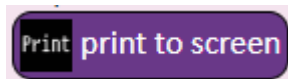
    using the sign. To get these options to reappear, click on the end of the line of code:

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.

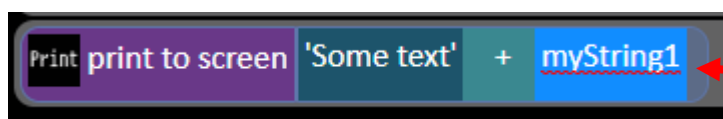| Assessment Guidance	
Emerging	Children have a basic understanding that coding involves writing instructions that a computer can follow.
	They are developing their understanding that these instructions must be precise and carefully structured through their work in Free Code Gorilla making simple one 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.
	Pupils can design and code a program that follows a simple sequence (lesson 1).
	Pupils attempt to introduce repetition and selection into their code using timers and simple 'if statements' (lessons 2 and 3). 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 4 they can follow the examples with support but will struggle when applying this with their own ideas.
	Pupils 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 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's designs for their programs, show that they are thinking of the structure of a

Assessment Guidance

	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's 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.

Pupils 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 make use of variables in their programs and combine these with timers to creative effect (lesson 4).

Pupils 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). |