purple mash

Computing Scheme of Work

Unit 2.1- Coding

Year Group: 2
Number of Lessons: 5

From 2simple

Contents

Introduction

This unit consists of six lessons that assume children have completed Unit 1.7 in year 1. If children have not completed this unit, then you might wish to teach the Coding Catch-Up Unit instead. Children will be coding using the 2Code tool.

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.

The Chimp 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 image and scale in 2Code.

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 pupils can be encouraged to support their peers, if necessary, helping them to understand but without doing the work for them.

Program Design

To master coding skills, children need to have the opportunity to explore program design and put computational thinking into practice. The lesson plans incorporate designing before coding in some lessons.

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	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/

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.

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'.

Year 2 – Medium Term Plan

Lesson	Aims	Success Criteria
Lesson 1 - Algorithms	• To understand what an algorithm is. • To create a computer program using simple algorithms.	• Children can explain that an algorithm is a set of instructions. • Children can describe the algorithms they created. • Children can explain that for the computer to make something happen, it needs to follow clear instructions.
Lesson 2 – Repeat and Timer	• To compare the Turtle and Character objects. • To use the button object. • To understand how use the Repeat command. • To understand how to use the Timer command.	• Children know that the Turtle and Character objects have different properties and move in different ways. They can begin to make choices about which object type to use. • Children are beginning to understand that the Repeat and Timer commands both make objects repeat actions but function differently and the type of object can affect which is the best command to use. • Children can include a button in their programs.
Lesson 3 - Debugging	• To know what debugging means. • To understand the need to test and debug a program repeatedly. • To debug simple programs.	• Children can explain what debug (debugging) means. • Children have a clear idea of how to use a design document to start debugging a program. • Children can debug simple programs. • Children can explain why it is important to save their work after each functioning iteration of the program they are making.
Lesson 4 – Different Object Types	• To create programs using different kinds of objects whose behaviours are limited to specific actions. • To predict what the objects will do in other programs, based on their knowledge of what the object is capable of. • To discuss how logic helped them understand that they could only predict specific actions, as that is what the objects were limited to.	• Children can create a computer program using different objects. • Children can predict what the objects in classmates' programs will do, based on my knowledge of the objects' limitations, e.g. a turtle can only move in specific ways. • Children can explain how they know that certain objects can only move in certain ways
Lesson 5 – The design, code, test process	• To use all the coding knowledge, they have learned throughout their programming lessons to create a more complex program that tells a story.	• Children can plan and use algorithms in programs successfully to achieve a result. • Children can plan and use algorithms in programs successfully to achieve the desired a result. • Children can code a program using a variety of objects, actions, events and outputs successfully.

Lesson 1 - Algorithms

Aims

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

Success criteria

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

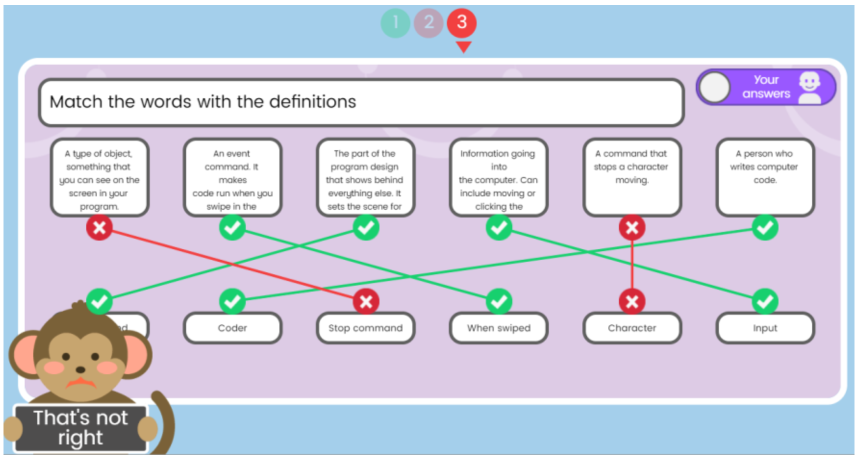
Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 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.

- Coding vocabulary quiz 1.
- Using two identical sets of any construction toy, build two models, one that follows the instructions and one that does not. An example would be using Lego Duplo to build a bird, one that follows these instructions and one that uses the same bricks but not the instructions. Download the instructions for your model so that you can display them on the board.
- Air Traffic Control guided lesson .
- Free code Chimp (found on the main 2Code page).
- (Optional) Print storyboard templates for program design.
- (Optional) Student and Teacher Flash vocabulary cards: The Teacher flash cards have been created in such a way that you can print them on A4 paper, cut them to size, fold them in half and glue them together.
- (Optional) Exercise books to be used as 2Code workbooks for recording coding exercises, if desired.

Activities

1. Firstly, we are going to review the coding vocabulary that the children learnt in year 1. Use the quiz as a class. It is set up so that you attempt all three questions and then click the Hand in button to check the answers. Click 'OK' to see which are correct and incorrect:



Match the words with the definitions

A type of object, something that you can see on the screen in your program.

An event command. It makes code run when you swipe in the

The part of the program design that shows behind everything else. It sets the scene for

Information going into the computer. Can include moving or clicking the

A command that stops a character moving.

A person who writes computer code.

Coder

Stop command

When swiped

Character

Input

2. You can use the vocabulary cards to find the answers and display in the classroom.

3. Explain that today, the class will be working on a new word: **algorithm**. Have a look at the definition together using the vocabulary cards:

   An ***algorithm*** is a precise step by step set of instructions used to solve a problem or achieve an objective.

4. Show the children your two models, without displaying the instructions. Ask them which is correct. Hopefully, they will say they are both correct; there is no such thing as correct when building creatively. They might prefer one over the other, but both are correct.

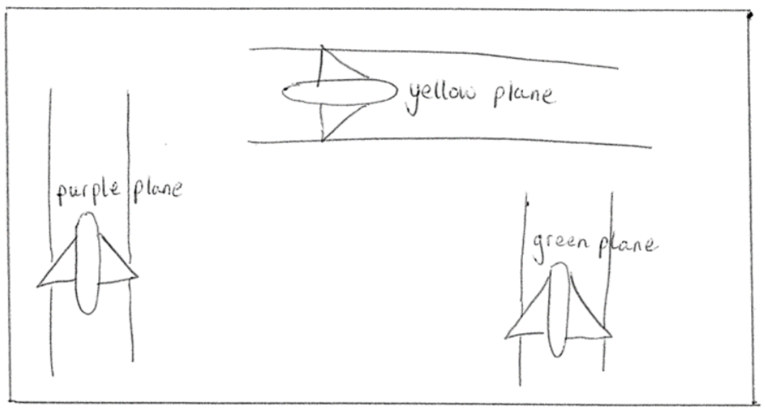
5. Now, show them the instructions and ask the question again. This time, the model made using the instructions is the correct one. The step-by-step instructions are the **algorithm** for building the bird.

6. Show children the Air Traffic Control lesson.

7. Work through challenges 1 and 2 with the children, reading through the blocks of code. Explain that we are following the instructions for what the code should do and then turning it into code.

8. When coding, we write algorithms for how the code should work then we turn the algorithms into computer code so that the computer can follow the algorithm. In the example, the algorithm is about making the planes move in different directions, so they look like they are taking off. Below is a diagram of the design with some numbered notes to show the algorithm.

<u>Task:</u> To make an airport program where the planes take off.



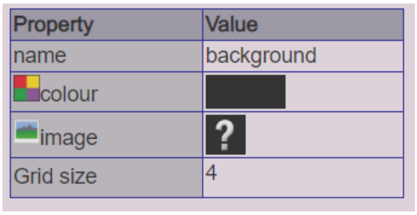
1) Click purple plane to make it take off.
2) Click yellow plane to make it take off.
3) Click green plane to make it take off.
4) If green plane crashes with yellow plane; make crashing sound.

9. We turn the algorithm into 2Code code that the computer can understand.

10. Next, review how to navigate to the 2Code page and find and open Free Code Chimp. Demonstrate how to go to **Design Mode** Design.

11. Drag in two turtle **objects** and change the image of one of them to a different coloured turtle (double-click on it).

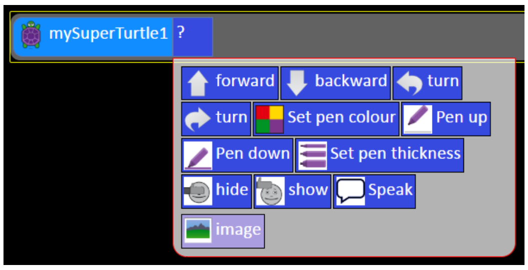
12. See whether children can remember how to add a background picture: click on the icon and then the image property question mark in the properties table:



Property	Value
name	background
colour	■
image	?
Grid size	4

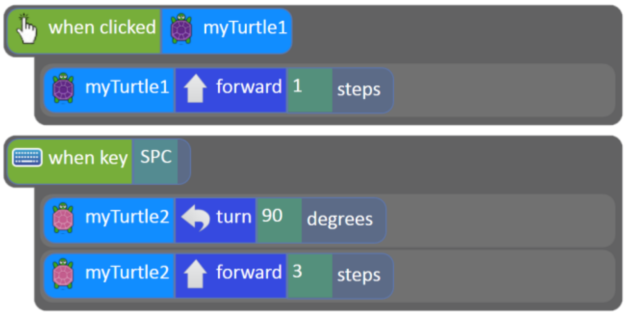
13. Remind children how to save their work and discuss why it is important to save their coding regularly so that they have a working version to go back to.

14. Go into Code Mode Exit design and show the children the **actions** that the turtle **object** can make by dragging one of the turtle **code blocks** into the coding window and looking at the pop-up **action** menu:



- forwards and backwards
- turn anticlockwise or clockwise by a set number of degrees
- Set pen colour, pen thickness, pen up and pen down
- hide, show and speak.

15. Children could work in pairs or individually. Children should create a design document similar to the plane example. They could use printed storyboard templates to show the sequence of the program (see resources section above). They should decide:
    - What their objects are (the turtles),
    - type of background (you might want to show the ready-made backgrounds) – **NB designs should always be rough drawings e.g. stick people, not a picture that takes time to draw.**
    - The steps of their algorithm (What their program should do?): Children should create a program in which one turtle uses an algorithm with one step, and the other an algorithm with two steps.



```
when clicked myTurtle1
    myTurtle1 forward 1 steps
when key SPC
    myTurtle2 turn 90 degrees
    myTurtle2 forward 3 steps
```

16. Once children have their programs planned out, they should create it in Free Code Gibbon. They should save and test their code.

17. Show children how to save their work in their folders and print it out for their workbooks, if required.

Lesson 2 – Repeat and Timer

Aims

- To compare the Turtle and Character objects.
- To use the button object.
- To understand how use the Repeat command.
- To understand how to use the Timer command.

Success criteria

- Children know that the Turtle and Character objects have different properties and move in different ways. They can begin to make choices about which object type to use.
- Children are beginning to understand that the Repeat and Timer commands both make objects repeat actions but function differently and the type of object can affect which is the best command to use.
- Children can include a button in their programs.

Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 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.

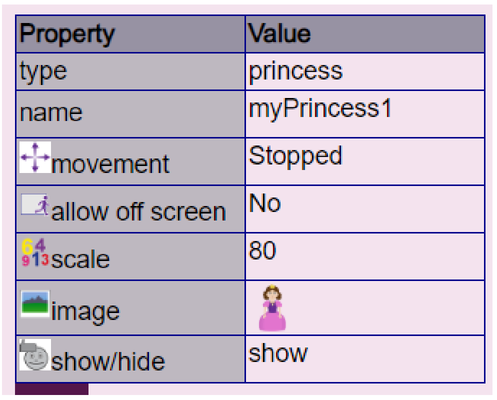
- Code-block cards: Children will need to use a few copies of each picture to create code away from the computer.
- Vocabulary cards as per Lesson 1. – specifically, the words 'Repeat' and 'Timer'.
- (optional) Exercise books to be used as 2Code workbooks for recording coding exercises, if desired.
- Repeat and Timer example. Set this as a 2do for the class.

Activities

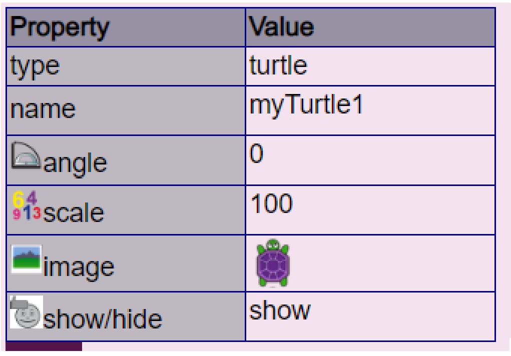
1. Begin by reviewing the code blocks used in year 1 to refresh the children's memories. Go through the code-block cards, discussing how they can be used to build up the code. Point out that each new instruction is written on a new line to make the code easier to understand. Give children some time in pairs to use the code-block cards to write their own code:
    - Child A should lay out a line of code by joining blocks.
    - Child B should read the line of code and explain what it does.
    - Repeat, swapping roles.
    - Can children include 'When Clicked' or 'When Swiped' in their code correctly?
2. Display the words, **Repeat** and **Timer**, on the board using the flash cards and ask the children to guess what they mean. Discuss each word with the children and explain their meaning.

    Repeat
    This command can be used to make a block of commands run a set number of times or forever.
    Timer
    Use this command to run a block of commands after a timed delay or at regular intervals.
3. Open Free Code Chimp on the board and look at the left-hand side toolbar. Ask children if they can see any of the new vocabulary to which they've just been introduced.

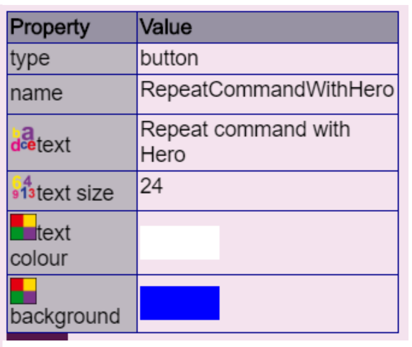
4. Open the example program. Look at the design view. There are 6 objects; a hero, a turtle and four buttons. These different objects move differently and have different properties. If you look at the properties of them in design view, you can see that they are different:



Property	Value
type	princess
name	myPrincess1
movement	Stopped
allow off screen	No
scale	80
image	
show/hide	show

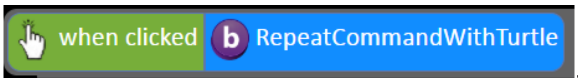


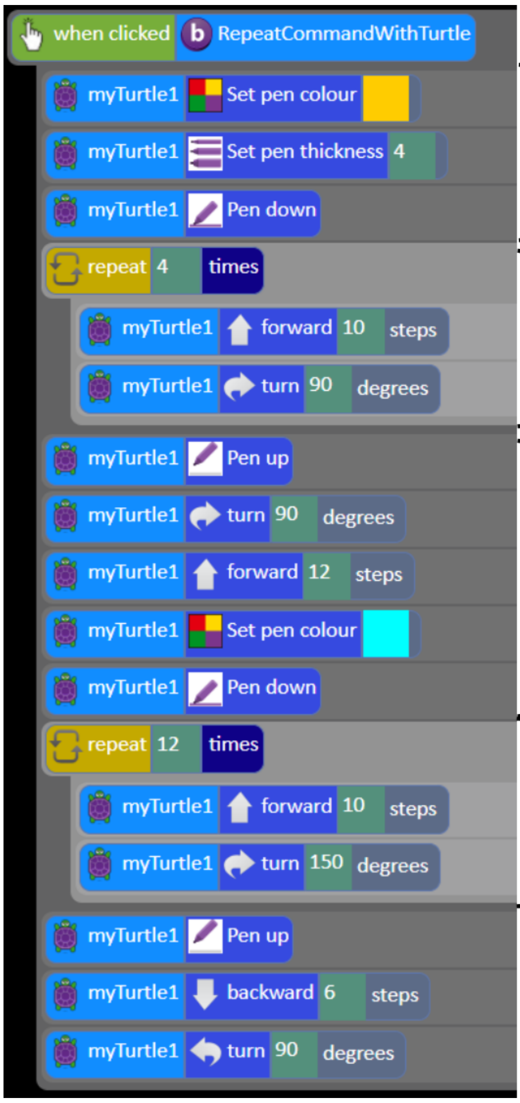
Property	Value
type	turtle
name	myTurtle1
angle	0
scale	100
image	
show/hide	show



Property	Value
type	button
name	RepeatCommandWithHero
text	Repeat command with Hero
text size	24
text colour	⬜
background	🟦

5. Switch to code mode and locate the code for the four buttons. You do not need to go through the code, just show children how to find the code for what happens when each button is clicked. The only **event** that buttons respond to is the **click event.** They cannot be coded to move or change.

6. Find the code for when  when clicked b RepeatCommandWithTurtle. What do the children think happens? You might be able to relate this to work that the children have done with floor robots.



```
when clicked  b RepeatCommandWithTurtle
    myTurtle1  Set pen colour 🟨
    myTurtle1  Set pen thickness 4
    myTurtle1  Pen down
    repeat 4 times
        myTurtle1  forward 10 steps
        myTurtle1  turn 90 degrees
    myTurtle1  Pen up
    myTurtle1  turn 90 degrees
    myTurtle1  forward 12 steps
    myTurtle1  Set pen colour 🟦
    myTurtle1  Pen down
    repeat 12 times
        myTurtle1  forward 10 steps
        myTurtle1  turn 150 degrees
    myTurtle1  Pen up
    myTurtle1  backward 6 steps
    myTurtle1  turn 90 degrees
```

Set the look of the line and put the pen down.

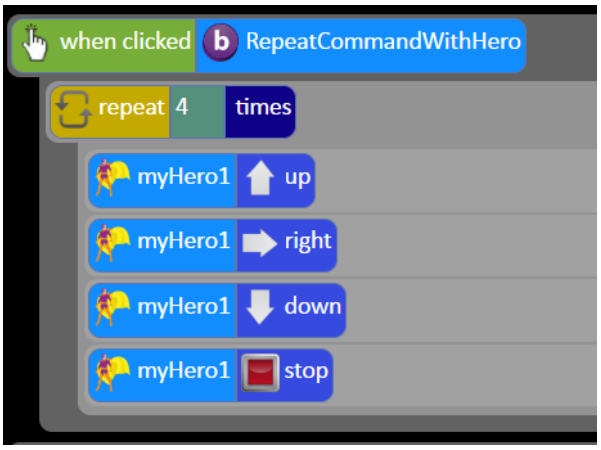
Can children work out what the Repeat does here? You will need to explain what 90 degrees means.

Pick the pen up and move it, set the look of the next lines.

Children are unlikely to be able to anticipate what the turtle draws here but can they explain what the turtle does and how many times it repeats?

7. Run the code and click on the button ‘Repeat command with Turtle’, concentrating on what the turtle does. You will see that the turtle moves very quickly.

8. Now look at the code for b RepeatCommandWithHero and discuss what they think happens.



```
when clicked  b RepeatCommandWithHero
    repeat 4 times
        myHero1 up
        myHero1 right
        myHero1 down
        myHero1 stop
```

9. Run the code and click on the button ‘Repeat command with Hero’, concentrating on what the hero does. It will probably not be what they are expecting. The reason it looks like the hero is doing very little is that the computer runs all the commands very quickly.

10. That’s why we use computers for things as they can process information extremely fast. We need code to slow the process down and that is where the **Timer command** comes in.

11. Click the Stop button to end the program.

12. Look at the code for the two timer buttons, can children make logical suggestions as to what will happen when the code is run?

13. Draw children’s attention to the difference between the timers for the turtle and the hero. The turtle timer says timer every. The Hero code uses timers that say timer after. What is the difference? Explain that the turtle timer will keep repeating until the program is stopped; there is no way to turn it off. The Hero timers will just cause an action to start or happen once.

14. Look at the final bit of code; when does this code run? When the Turtle collides with the Hero.

15. Run the code and compare this to their expectations. Notice that the turtle doesn’t stop whereas the Hero does. Run the program a few times, sometimes the hero will collide, sometimes it won’t – this means that the collision code doesn’t always get run.

16. Children should open the example from their 2dos tab and try changing some of the code to explore the function of the Repeat or timers. Can they reach a point where the code does exactly what they expect or are there still surprising effects? This is why code always needs testing to make sure that it functions as expected.

17. Further work on Timers (optional)

- In the Chimp activities, the use of the timer is introduced in a video in Step 5 of ‘**Princess and the Frog’**, and the **‘Tick Tock Clock’** and **‘Magician’** activities also use a timer. You can look at these with the children if they need more guidance or ideas.

Lesson 3 - Debugging

Aims

- To know what debugging means.
- To understand the need to test and debug a program repeatedly.
- To debug simple programs.

Success criteria

- Children can explain what debug (debugging) means.
- Children have a clear idea of how to use a design document to start debugging a program.
- Children can debug simple programs.
- Children can explain why it is important to save their work after each functioning iteration of the program they are making.

Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 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.

- Debug Challenges Chimp – this is on the main 2Code page - set this as a 2do for the class.
- Debugging process to display on the whiteboard.
- Debugging Record sheet – each child (or pair) will need a paper copy.
- Debugging examples (4); set these as 2dos.

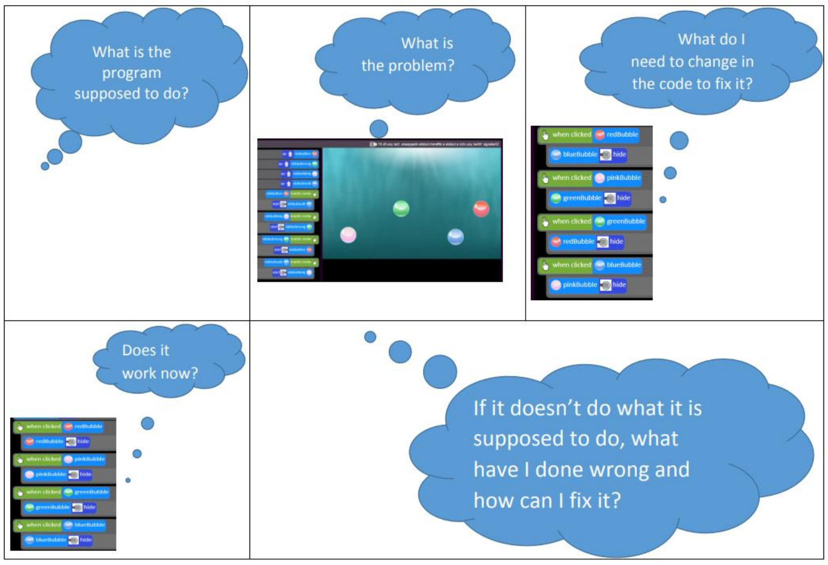
Activities

1. Today we are learning another new piece of vocabulary: **debugging**. Ask children if they know what **bug**, **debugging** or to **debug** means?

   Bug
   A problem in a computer program that stops it working the way it was designed.
   Debug/Debugging
   Looking for any problems in the code, fixing and testing them.

2. Last lesson we were looking at Repeat and Timer; these often don't work exactly as expected and can prevent the program doing what you design it to do. This shows how important it is to test the code, debug and save at each stage of creating a program, not just at the end.

3. Before you start to debug a program, you need to think through some steps. Put the following on the board (there is a link to a PDF of this table in the Resources section). Children could write/stick the steps into their 2Code workbooks.


What is the program supposed to do?
What is the problem?
What do I need to change in the code to fix it?
Does it work now?
If it doesn't do what it is supposed to do, what have I done wrong and how can I fix it?

```
when clicked redBubble
    blueBubble hide
when clicked pinkBubble
    greenBubble hide
when clicked greenBubble
    redBubble hide
when clicked blueBubble
    pinkBubble hide
```

```
when clicked redBubble
    redBubble hide
when clicked pinkBubble
    pinkBubble hide
when clicked greenBubble
    greenBubble hide
when clicked blueBubble
    blueBubble hide
```

4. Children should open the [Debug Challenges Chimp](#) on their device and complete the challenges.

5. Once children have completed the Debug Challenges, bring them back together and hand out the debugging record sheets. Explain that you have set four 2Code programs as 2dos and show the children how to find them.

6. The children need to work out which program matches each design and circle the letter on the sheet.

7. Then they need to use the algorithm and test the code to find the bugs (number of bugs shown in brackets), write down what the bugs are and fix them. Then save the fixed file. They can 'sign-off' to say that all the bugs are fixed 'like real programmer do'.

8. Review the examples as a class. How did they match the programs to the designs? How did they go about fixing the bugs?

Lesson 4 – Different Object Types

Aims

- To create programs using different kinds of objects whose behaviours are limited to specific actions.
- To predict what the objects will do in other programs, based on their knowledge of what the object is capable of.
- To discuss how logic helped them understand that they could only predict specific actions, as that is what the objects were limited to.

Success criteria

- Children can create a computer program using different objects.
- Children can predict what the objects in classmates' programs will do, based on my knowledge of the objects' limitations, e.g. a turtle can only move in specific ways.
- Children can explain how they know that certain objects can only move in certain ways

Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 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.

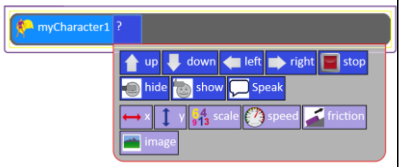
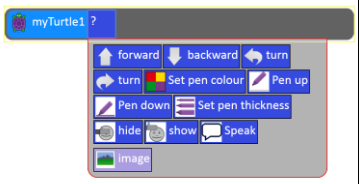
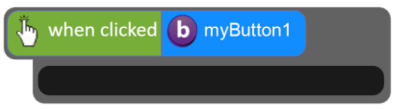
- Example program for Lesson 4.

Activities

1. Today we will be creating our own programs using different kinds of objects. Review what 'objects' are in coding. Remind children where objects can be found in Design Mode in Free Code Chimp.

2. Explain that in a computer program, objects can only do what we, the coders, tell them to do. Some objects can only do certain things.

3. Go through the different types of object found in Design Mode and explain that in this lesson we will only be using turtles and characters.

Characters	**Turtles**	**Buttons**
Food, animals, princesses, heroes and emoticons can go up, down, left, right, hide and show.  myCharacter1 ? up down left right stop hide show Speak x y scale speed friction image	Can go forwards, backwards, turn clockwise and anticlockwise.  myTurtle1 ? forward backward turn turn Set pen colour Pen up Pen down Set pen thickness hide show Speak image	Can be clicked. 

4. Tell children that the task is to create a program in Free Code Chimp using one turtle and the When Clicked command. The turtle should do three or four different things when clicked.

4. Children should create a brief design first to plan what they want their turtle to do.

5. Then children should code, save, test and debug their code.

6. Come back together as a class. Open one of the children's programs on the board in Design View and ask the children to predict what the turtle will do when they click on it.
7. The turtle is limited to going forwards, backwards, turning left or turning right, so any of those answers is potentially correct. Go through two to four programs, getting the children to guess what the turtles might do, based on their logical knowledge of the turtles' limitations.
8. Explain to children that now, they will be adding characters to their programs. Show on the whiteboard that characters can do more than turtles, so the children have more scope in terms of what to do with them. Example program for Lesson 4.
9. They should adapt their designs to add a character. The character should do four different things when different keys are pressed on the keyboard (using the When Key command). **NB** If the children are coding on tablets, this option is not available. Instead they should code the character to do different things when the background is swiped in different directions.
10. Send children to devices to work on their programs.
11. Once children are done, they should save their programs and come back together at the whiteboard.
12. Open a child's program on the board in Design View and ask the class to predict what the character will do when it is clicked. Characters can go up, down, left, right or stop, so any of those answers are potentially correct. Repeat with two to four more programs, asking children to predict both turtles' and characters' actions, using their knowledge of the character limitations of movement to guide them.

Lesson 5 – The design, code, test process

Aim

- To use all the coding knowledge, they have learned throughout their programming lessons to create a more complex program that tells a story.

It would be a good idea for the children to spend some time planning their story for this program in a Literacy lesson. The storyboards linked to below might be useful for this.

If you wish to have 6 lessons in this unit, this lesson could easily be stretched over 2 sessions.

Success criteria

- Children can plan and use algorithms in programs successfully to achieve the desired an end result.
- Children can code a program using a variety of objects, actions, events and outputs successfully.

Resources

Unless otherwise stated, all resources can be found on the main unit 2.1 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.

- Example program for Lesson 5 Example Story program.
- Blank printable storyboards for designing.
- Create a displayboard for children to share their work to. See Appendix 1 for details of how to do this.

Activities

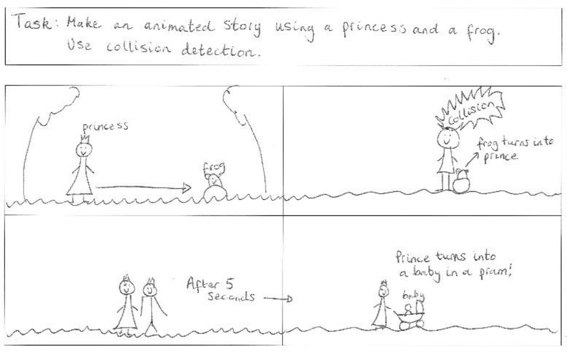
1. Today we will be using everything we have been learning about programming to create a computer program that tells a short story.

2. If you have used part of a Literacy lesson to plan the story, you can omit the planning stages here.

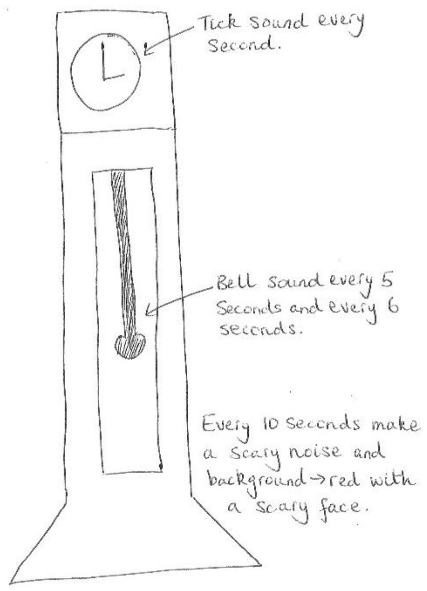
3. Children could work in pairs or individually to think of a short story that they could tell using their computer programming knowledge. For example, a character being eaten by a dragon and a hero coming to save them. The following is an example: Story Program Example.

4. Some of the Chimp activities can also provide ideas for actions in the story, such as 'Guard the Castle' and 'The Princess and the Frog'.

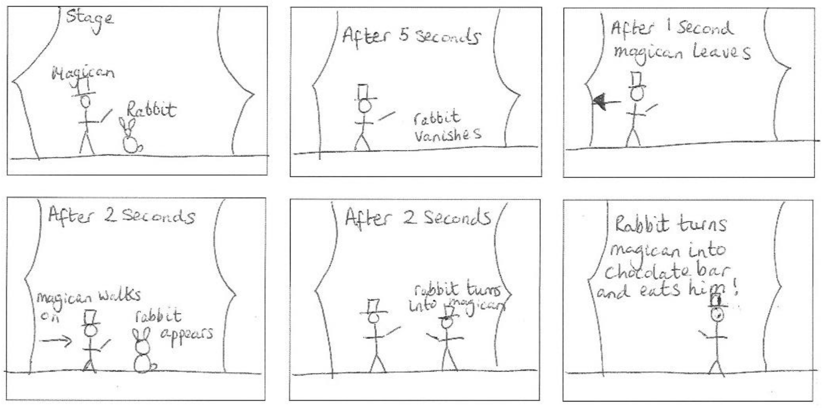
5. Encourage children to think through their designs and annotate them including their confidence in coding what they have designed (red, amber, green), this will give you feedback on areas that children need help with and help to ensure that children create realistic designs and successful programs for their skill level. Here are some sample designs:


Task: Make an animated story using a princess and a frog.
Use collision detection.
princess
frog
collision
frog turns into prince
After 5 seconds
Prince turns into a baby in a pram!
baby

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.

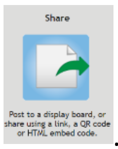
Task: Make an animated story about a magician and a rabbit. Use timers.


Stage
Magician
Rabbit
After 5 seconds
rabbit vanishes
After 1 second magician leaves
After 2 seconds
magician walks on
rabbit appears
After 2 seconds
rabbit turns into magician
Rabbit turns magician into chocolate bar and eats him!

6. Children should spend time designing, planning (annotating with the algorithms) and coding.

7. Remind children how to save and then show them how to share their work to a displayboard:

8. Click on the share button (they must have saved their work first).

9. Click 

Share

Post to a display board, or share using a link, a QR code or HTML embed code.

.

10. Find the displayboard that you created for them and click on it and then ‘ok’.

11. You (the teacher) will then need to bulk approve all the work so the children can see it. See Appendix 2 for a guide to how to do this.

12. Show some of the children’s work using the Displayboard. The child whose work it is, should read their own code to explain what the program should do, and you can try it out.

13. Peer review the programs, can others identify particularly good aspects from programs or suggest improvements in their programs?

Appendix I: Creating a Displayboard

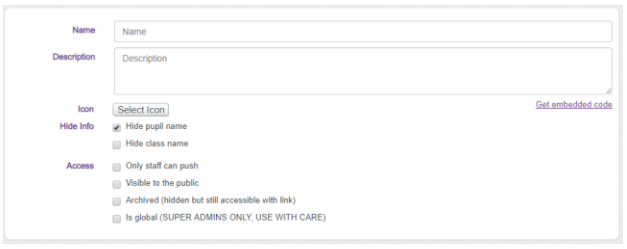
For detailed information about Purple Mash Displayboards, see the User Guide in the Teachers>User Guides section of Purple Mash. These instructions tell you only the steps needed for the displayboard this unit of work

Click on the Admin tab and select the Manage Display Boards icon to access the Display Boards control panel.

You will see a list of the existing boards. If this is your first board the list will be empty.

Under Available Boards click the icon and choose a name for your board.

You can optionally add a description for the board and choose an icon to represent your board.



Name	Name
Description	Description
Icon	Select Icon — Get embedded code
Hide Info	[✓] Hide pupil name
	[ ] Hide class name
Access	[ ] Only staff can push
	[ ] Visible to the public
	[ ] Archived (hidden but still accessible with link)
	[ ] Is global (SUPER ADMINS ONLY, USE WITH CARE)

Tick the Hide Info boxes if children access Purple Mash at home and you do not want names viewable.

Do not tick any of the Access tick boxes.

In the bottom section, locate your class by clicking on the arrow next to the Classes folder and ticking the box by the class name.

Click the Save button at the bottom of the screen to save your board.

You're done! Your board is ready to receive new projects from pupils.

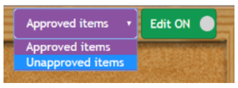
NB You will only be able to add classes if you are the allocated teacher for that class.

Appendix 2 - Approving work on a displayboard

For detailed information about Purple Mash Displayboards, see the User Guide in the Teachers>User Guides section of Purple Mash. The steps below assume that you are going to bulk approve all entries on the displayboard as part of lesson 4.

Open the displayboard from the home screen by clicking on the Sharing tab and then on the displayboard.

Turn on editing by clicking the Edit OFF slider. Then select 'unapproved items' from the drop-down list.



Click on the first item on the page, click and hold down, the Shift key, then click on the last item on the page.

Then click the icon to approve selected projects.

If there are more pages (grey arrow on the right), go to these and approve the work in the same way.

To return to the Display Board main screen, click the icon.

Assessment Guidance

The unit overview for year 2 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 know that an algorithm is related to giving instructions. They can relate a simple one step algorithm to the outcome of code in Free code Chimp. For example, in Lesson 1, steps 15-16 they have been able to make a program that follows their algorithm e.g. 'when the turtle is clicked it moves forwards'. With support, children can create a simple one step program that achieves a specific purpose. With support, children can identify and correct errors. For example, (Unit 2.1 Lesson 3). With support, children can identify the parts of an algorithm that control and initiate specific actions. Based on this, with support, children can predict what will happen in a program. For example, (Unit 2.1 Lesson 4).
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 Chimp. For example, in Lesson 1, step 16 they have been able to make a program that follows their algorithm e.g. 'when the turtle is clicked it moves forwards then turns right'. Children show an awareness of the need to be precise in their designs so that algorithms can be successfully translated into code. For example: lesson 5 – step 6. Children use a planning format on paper before implementing on screen within 2Code as they recognise this is the best approach for designing a solution efficiently (Unit 2.1. Lesson 5). 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 movements to turtle objects (Unit 2.1. Lesson 1 Point 16). They confidently include objects, actions, events and outputs successfully within their 2Code programs. Children can talk through code which contains repeat and timer commands, explaining where they are positioned and what will happen (Unit 2.1. Lesson 2 Points 6 & 8). Children can predict program outcomes and attempt to debug. For example, (Unit 2.1 Lesson 3).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. For example, (Unit 2.1 Lesson 4). Children can debug their own and other's programs using design documentation to test against (Unit 2.1. Lesson 3). Most children will be able to save their files using a memorable file name e.g. their name and a simple title etc. Specifically, they will be able to save their work in these lessons- (Unit 2.1 Lesson 1 Point 9, Unit 2.3 Lesson 1 Point 2 and Unit 2.6 Lesson 1 Point 8).

Assessment Guidance	
Exceeding	Children can explain and give examples that an algorithm is a set of instructions to complete a specific task. They can create complex and logical algorithms of several steps that accomplish the aim of the task that can be easily utilized to create executable code. Children show an awareness of the need to be precise in their designs so that algorithms can be successfully translated into code. For example: lesson 5 – step 6. Children can create more complex programs that utilize all the coding constructs that they have learnt about and extend their own learning by trying out different ways to code that achieve a specific purpose. Children can identify and correct errors. For example, (Unit 2.1 Lesson 3). An exceeding pupil will be able to apply their knowledge as a transferable skill across a range of debugging scenarios including making logical attempts to debug their own more complex code. Children can identify the parts of a program that respond to specific events and initiate specific actions. Based on this, children can adopt a systematic approach for predicting the behaviour of programs. Furthermore, using cause and affect language, children can reason in detail about what will happen in a program. For example, (Unit 2.1 Lesson 4).

14.