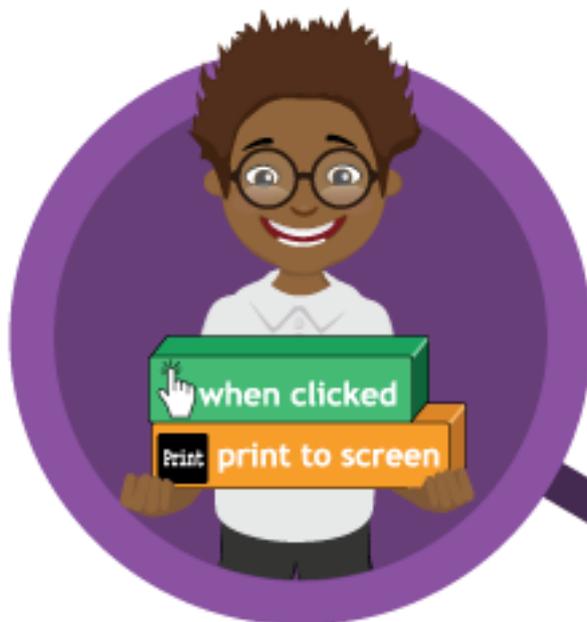


Unit 6.1 - Coding



Year Group: 6
Number of
Lessons: 6

From **2**simple



Contents

Introduction	3
Program Design	3
Levels of Scaffolded coding tasks	4
Year 6 – Medium Term Plan	5
Lessons 1 and 2 - Designing and Writing a More Complex Program.....	6
Aims	6
Success criteria	6
Resources	6
Activities	6
Lesson 3 - Introducing Functions	8
Aims	8
Success criteria	8
Resources	8
Activities	8
Lesson 4 – Using user input	11
Aims	11
Success criteria	11
Resources	11
Activities	11
Lesson 5 – Flowcharts and Control Simulations	13
Aim	13
Success criteria	13
Resources	13
Activities	13
Lesson 6 - Using 2Code to make a text-based adventure	14
Aim	14
Success criteria	14
Resources	14
Activities	14
Appendix 1 - Creating a Displayboard	19
Appendix 2 - Approving work on a displayboard.....	20
Appendix 3 – The Avatar tool	21
Assessment Guidance.....	22



Introduction

This unit consists of six lessons that assume children have followed the Coding Scheme of Work in Years 1 to 5. If most of the class have not, use the Coding Catch-Up unit instead of this unit.

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

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

Children will often be able to solve their own problems when they get stuck, either by reading through their code again or by asking their peers; this models the way that coding work is really done. More able 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?
- Using the 2Chart tool to create flowcharts of the processes.

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  Least scaffolded	Copying code	By giving children examples of code to copy.
	Targeted tasks	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<p>Projects (imitate, innovate, invent, remix)</p> <p>There are different ways to scaffold learning in projects. This process can be applied to programming projects;</p> <ul style="list-style-type: none"> • Using example projects e.g. the Guided 2Code activities. • Completing the challenges at the end of each guided activity. • Free code ✓ • Create a project that imitates a high-quality exemplar. • Remixing ideas. • Independently creating a brand-new program.
	Tinkering	<p>Use Free code Gorilla to access the full suite of 2Code objects and commands ✓</p> <p>Use Free code to play and explore freely.</p>

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

In Literacy, some teachers follow a progression that scaffolds learning to write texts. At first 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.



Year 6 – Medium Term Plan

Lesson	Aims	Success Criteria
Lessons 1 & 2 - Designing and Writing a More Complex Program	<ul style="list-style-type: none"> To review good planning skills. To design programs using their choice of objects, attributing specific actions to each using their new programming knowledge. To use variables within a game to keep track of the properties of objects. 	<ul style="list-style-type: none"> Children can plan a program before coding to anticipate the variables that will be required to achieve the desired effect. Children can follow through plans to create the program. Children can debug when things do not run as expected.
Lesson 3 - Introducing Functions	<ul style="list-style-type: none"> To use functions and understand why they are useful in 2Code. To debug a program and organise the code into tabs. To organise code into functions and Call functions to eliminate surplus code in the program. 	<ul style="list-style-type: none"> Children can explain what functions are and how they can be created and labelled in 2Code. Children can explain how to move code from one tab to another in 2Code. Children can explain how they organised code in a program into functions to make it easier to read.
Lesson 4 – Using user input	<ul style="list-style-type: none"> To explore the options for getting text input from the user in 2Code. How to include interactivity in programming. 	<ul style="list-style-type: none"> Children can code programs that take text input from the user and use this in the program. Children can attribute variables to user input. Children are aware of the need to code for all possibilities when using user input.
Lesson 5 – Flowcharts and Control Simulations	<ul style="list-style-type: none"> To use flowcharts to test and debug a program. To create a simulation of a room in which devices can be controlled. 	<ul style="list-style-type: none"> Children can follow flowcharts to create and debug code. Children can create flowcharts for algorithms using 2Chart. Children can be creative with the way they code to generate novel visual effects.
Lesson 6 - Using 2Code to make a text-based adventure	<ul style="list-style-type: none"> To explore how 2Code can be used to make a text-based adventure game. 	<ul style="list-style-type: none"> Children can follow through the code of how a text adventure can be programmed in 2Code. Children can adapt an existing text adventure to make it unique to their requirements.



Lessons 1 and 2 - Designing and Writing a More Complex Program

Aims

- To review good planning skills.
- To design programs using their choice of objects, attributing specific actions to each using their new programming knowledge.
- To use variables within a game to keep track of the properties of objects.

Success criteria

- Children can plan a program before coding to anticipate the variables that will be required to achieve the desired effect.
- Children can follow through plans to create the program.
- Children can debug when things do not run as expected.

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.

- Exercise books to be used as 2Code workbooks for recording coding exercises, if desired.
- [2Code Game Planner](#) or [Leaflet - Coding Planner](#) this can be set as a 2do or printed for each child/pair.
- [Making a Timer and Score Pad guide](#). This can be set as a 2do for children to refer to or printed.
- [Spider Catcher game](#). This is a game that they will have made in year 5, it is used as an example today.
- [Introduction to Tabs video](#).
- Free code Gorilla found on the main 2Code screen in the Gorilla section.

Activities

- Discuss how the children's coding skills are becoming more sophisticated and emphasise how important the planning stage for their programs is becoming as a result. Over two lessons, children will explore good planning skills using a pro forma and then consider how this helps them create their programs more efficiently with fewer bugs.
- Put [Free Code Gorilla](#) on the board. Review how to add objects (characters) in 2Code by going in to Design Mode.
- From last year, can children recall using a number variable to create a timer and to control the time left in a game? Review what they did in the Spider Catcher game.
- The aim for today is to design a game that includes timing and scoring. Can the children suggest any ideas from a game they might have played? They will need to start with a simple version of the game that they can then enhance. Remind them that this is called a high level of **abstraction**.
- Show the children the planning proforma and explain the types of things that should be included in each section, for example, the characters, the events that will occur and any variables required to keep a track of things.



Purple Mash Computing Scheme of Work Unit 6.1 – Coding – Lessons 1 & 2

- If they have been following the Scheme of Work in previous years, they should be able to incorporate *some* of the following aspects into their program and refresh their knowledge:
 - Characters that respond to being clicked.
 - Characters that respond to collisions.
 - Using timers to repeat actions.
 - Using Repeat Until.
 - Using If and If/else statements.
 - Using variables (numbers and strings).
 - Adding a timer and/or score pad to games.
 - Adding buttons that take the user to places outside the program.
- Give children time to think about what they want their games to do and plan in detail before they start coding. Children may want to look over their previous programs to refresh their memory on some aspects if they have not used 2Code for a while. The Gorilla activities might provide further reminders.
- Before the children start coding, revise the use of tabs, used to keep code organised. Now that the children are making more complex programs it is important for them to organise their code so that they can understand what it does, and it makes debugging easier. Show the video [Introduction to Tabs video](#). The Spider Catcher game also uses tabs.
- If they wish, they can finish their game at home and create a QR code or web link to it. This can be inserted into a school blog or webpage.

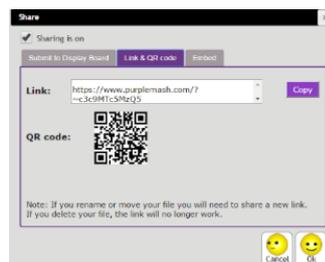
How to Create a QR Code.

- Show the children how they can create a QR code for their program. They can print the QR code for their books or save the image to be scanned by a QR reader to open the program.
- First, save the file in an online folder (not on the computer), then click on the Share icon at the top of the



screen.

- Next, select Share, then Link and QR code



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





Lesson 3 - Introducing Functions

Aims

- To use functions and understand why they are useful in 2Code.
- To debug a program and organise the code into tabs.
- To organise code into functions and Call functions to eliminate surplus code in the program.

Success criteria

- Children can explain what functions are and how they can be created and labelled in 2Code.
- Children can explain how to move code from one tab to another in 2Code.
- Children can explain how they organised code in a program into functions to make it easier to read.

Resources

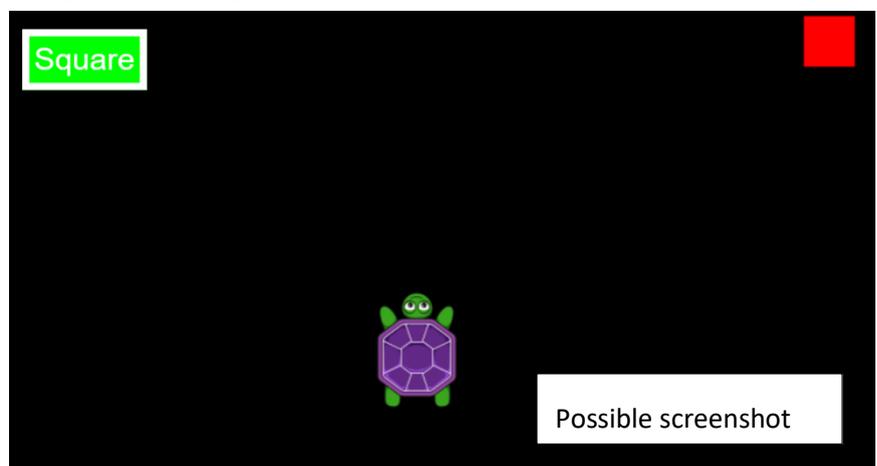
- set the Functions guided lesson in **Gibbon** as a 2do. This is found in the main 2Code area in Tools.

Activities

1. In this lesson we will be learning about functions. Has anyone seen this word in 2Code? They might have seen it when creating a variable as the type 'function' is available as well as string and number.

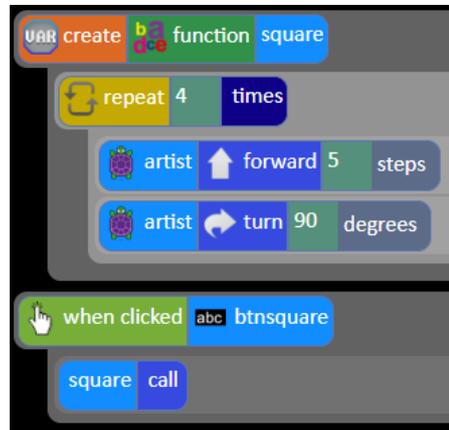
A **function** is a block of code that you can access when you need it, so you don't have to rewrite the same block repeatedly. Instead, you simply **Call** the function each time you want it.

- Children should work on the **FUNCTIONS** guided lesson, found in the **Gibbon** activities section of 2Code.
- This is a brief activity that shows children how to create and call functions.
- Once children have completed this, bring the class back together and open Free Code Gorilla on the board.
- Add a turtle object, a shape object and a text object.
- Call the turtle object 'artist'.
- Call the shape 'red'. Set the number of sides property to 4, the size to 1 and the angle to 45.
- Call the text object 'btnsquare', double-click on it and change the text to 'Square'. Show the children how to edit the look of the text object using the following properties:

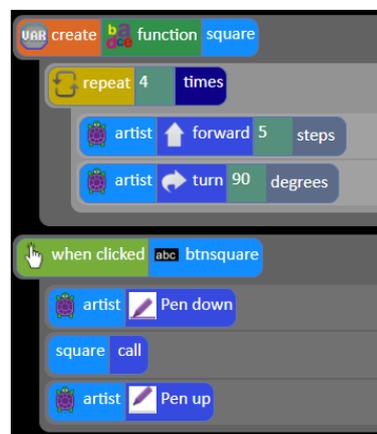




- Save the work and switch to code view. Ask the children to guide you in making a function named Square (they will need to guess the number of steps and then debug this code, if necessary, once they have tested it). The function needs to be **Called** for it to run. **Call** the function when the btnsquare (the text object) is clicked.



- Save and test the code. The turtle should move in a square shape.
- Can children suggest a way to make the turtle draw the square? As a class, add to the code so that the turtle puts the pen down before drawing the square and then picks the pen up at the end. Should this code go in the click event of the button or in the function? It should go in the function if you want it to happen every time a square is drawn. Either would work for now, but say you wanted to make patterns where squares were drawn repeatedly, you might just want one pen down and pen up in the flow of the code.





Purple Mash Computing Scheme of Work Unit 6.1 – Coding – Lesson 3

- Can you now add code so that when the red square is clicked, it changes the pen colour to red?
- Switch to design view and show children how to copy an object by clicking on it and then clicking the Copy button at the bottom of the property window. This makes a copy on top of the original, they can drag it to a different location and change its properties instead of creating a new object from scratch.
- Children should now create additional functions and buttons to draw other shapes. They can also add more colour choices. They could use a separate tab for the colour changes to make their code easier to read. They could also add arrow buttons to move the turtle between shapes, putting the code for this in a separate tab as well.

Here are some examples of a possible screen and code for teacher reference.

The image shows a screenshot of the Purple Mash software interface. At the top, there are four tabs: 'Square' (green), 'Triangle' (blue), 'Hexagon' (magenta), and 'Pattern' (orange). To the right, there is a vertical color palette with six colored squares: red, blue, yellow, green, and pink. The main canvas features a large blue geometric pattern on the left, a yellow triangle, a purple turtle, and a pink hexagon on the right. Below the canvas are two code blocks.

Code Block 1 (Left): This block is titled 'My Code' and has a sub-tab 'colours'. It contains five 'when clicked' blocks, each followed by an 'artist' block and a 'Set pen colour' block with a corresponding color swatch: red, yellow, green, blue, and pink.

Code Block 2 (Right): This block is titled 'My Code' and has sub-tabs 'colours' and 'move'. It contains five 'when clicked' blocks, each followed by an 'artist' block and a movement block: 'left' (x subtract 1), 'right' (x add 1), 'up' (y subtract 1), and 'down' (y add 1).



Lesson 4 – Using user input

Aims

- To explore the options for getting text input from the user in 2Code.
- How to include interactivity in programming.

Success criteria

- Children can code programs that take text input from the user and use this in the program.
- Children can attribute variables to user input.
- Children are aware of the need to code for all possibilities when using user input.

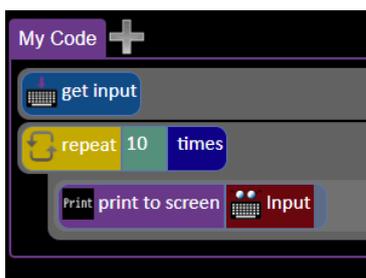
Resources

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.

- [Alien cards](#) – each child/pair will need a copy of the cards, they will need to cut them up into separate aliens.
- [Guess Which Alien partially completed program example](#), set this as a 2do for the class.
- Create a display board for the class to share their programs to. Details of how to do this are given in [Appendix 1](#)

Activities

1. Can children suggest ways that they can program 2Code to get input from the user of the program? When the user clicks on an object, when the user presses keys or swipes the screen, the get Input and Prompt for input commands.
2. Ask the children to spend 5 minutes using 2Code Gorilla to explore the get Input command and the Prompt for input command.
3. Bring the class back together to see what they have discovered. Some might have made programs that illustrate what they do. Use their examples to demonstrate or use the following examples:





4. Notice how the string is built up in this code. Discuss why the parts , and are needed.

```

VAR create 6 number year = 0
prompt 'What year were you born?'
year = set to Input
alert 'In ' + year + ' 100 ' + ', you will be 100!'
    
```

5. Discuss setting the input to a variable (number or string) and then manipulating it. What happens if you give a word answer for this last example?
6. Explain that today, children will be making an interactive game using input from the user. Open the example on the board and show the children the alien cards. Play the game with the alien called Zinky. Zinky should work but none of the other answers have been programmed yet.

```

prompt 'Please enter a name'
repeat 10 times
  print print to screen Input + ' the ' + random Adjective + ' ' + random Animal + '!'
  alert Input + ' is many things!'
    
```

7. Look at the code. Can children read through the existing code, explaining what happens?
8. What do they think the next steps to completing the program are?
- Code for the red alien with 2 eyes
 - Code for the other colours of aliens; after colour, the questions do not have to be about number of eyes, but they do have to be about appearance.
9. Explain that the information about the aliens can be found in 2Investigate (in Tools) in the example Aliens database. They could use this to add detail and personality to their game.
10. **Extension/Alternative:** Some children might want to attempt to make a different game using the same principles but other objects such as bugs or musical instruments. They might want to use their classmates as objects – stress the need to be polite and considerate in the questioning. You might prefer them to create some avatar pictures to use instead of actual classmates. They can do this very easily using the Avatar tool (see [Appendix 3](#) for details).
11. Children should share their finished games to the displayboard (see [Appendix 2](#) for details of how to do this). Then children can play each other’s games.



Lesson 5 – Flowcharts and Control Simulations

Aim

- To use flowcharts to test and debug a program.
- To create a simulation of a room in which devices can be controlled.

Success criteria

- Children can follow flowcharts to create and debug code.
- Children can create flowcharts for algorithms using 2Chart.
- Children can be creative with the way they code to generate novel visual effects.

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.

- Example program: [Billy's bedroom Simulation](#). Set this as a 2do
- [Billy's bedroom flowcharts](#). Set these as a 2do or print a copy for children to refer to.

Activities

1. Open Billy's bedroom on the board and explain that this is a simulation of his room. There are several devices that perform actions either automatically or when buttons are clicked.
2. Open the flowchart for Billy's room and look at the Day/Night chart.
 - What should happen?
 - What are the pink boxes (these indicate an external procedure runs (you will see the fairy and rocket procedures elsewhere on the sheet).
 - Run the code to see if this works in the code. Day/Night background does work, and the rocket does work but the fairies do not.
3. Look at the flowchart for the rocking horse:
 - What effect is this algorithm designed to achieve? (Rocking, slowing and stopping).
 - Does this look correct in the code – how do we test it?
4. Children's task today is to interpret the flowcharts, debug and get the rest of the devices in Billy's room working.
 - There are some bugs in the code, so they need to check whether each device works according to the flowchart and fix it if not.
 - There is no flowchart for the car, robot and computer.
 - There are some suggested ideas, but children will need to decide what they should do and how.
 - They should create flowcharts in a new 2Chart file.
 - They might need to add more objects e.g. there is no remote control for the car, they will need to add buttons to look like a remote control. They can look at the buttons on the drawer knobs to get an idea for this.
 - There are functions to make the fairies fly, but these do not have any code in them.

They should start with the things that look simplest and then try the harder ones. They could add more devices and create flowcharts for them if they have time.



Lesson 6 - Using 2Code to make a text-based adventure

Aim

- To explore how 2Code can be used to make a text-based adventure game.

Success criteria

- Children can follow through the code of how a text adventure can be programmed in 2Code.
- Children can adapt an existing text adventure to make it unique to my requirements.

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.

- Example [Y6 Text Adventure](#). Set this as a 2do for your class to open.
- Pencils and paper for children to sketch plans and ideas on.
- [Text Adventure Functions](#) children might need to refer to these.

Activities

1. What is a text adventure?

Explain that a text adventure is a computer game that uses text instead of graphics. The games simulate environments in which players use text commands to control characters and influence the environment. A bit like computer versions of the 'Choose your own adventure' books but often with puzzles to solve.

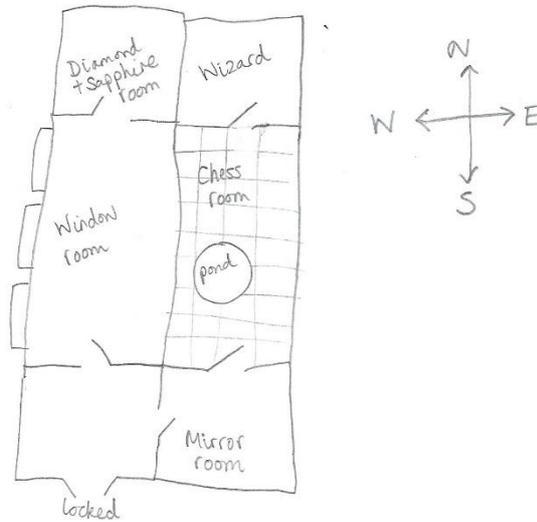
They were very popular before graphics based games were developed. Despite of the lack of graphics, they are still fun to play and usually require you to think and solve puzzles. Modern games like 'Escape the room' adventures are very like text adventures but with the addition of visual clues. Text adventures are also known as 'Interactive fiction' where they are more of a story with choices than puzzles to solve.

2. We are going to begin today by playing a simple text adventure written using 2Code. Then you will have a chance to change the code to make the adventure your own.
3. To solve this adventure, it is very useful to sketch a map of the places that you go and write some notes. It will also help you to adapt this adventure yourself.
4. Show children how to open the adventure game and press play to get started. For the sake of simplicity of programming, answers can only be typed in lower case. Directions should be entered as n, e, s or w.
5. Give the children some time to play the adventure, see who is first to solve it.

The adventure has only 6 rooms but let children explore freely without clues. Children should go to the room with the diamond on the floor and take it to the wizard. He gives you the key which you can use to exit the game.



- Once children have had time to play, compare the maps that they have drawn and discuss how having a map is helpful for solving the game.
- It is useful for everyone to have a simple map when discussing the code. Something like this is perfect:



- The next stage is to look at the code. Children will be making changes to their code, so it is worth talking about the need to save their files frequently after testing that they work. It can be easy to remove some vital code and then not know how to get the code back to how it was before so by saving correct code after each successful change you can always go back to how it was by reopening your file. Also, highlight the undo button which appears when you have changed something so that you can undo; often this will fix the problem.
- In the code, each room has been given a number. Tell the children that you will look through the code together and try to write the numbers on the map.
- Look at the first 5 lines of the code; this is where various variables are set. Do the names of the variables provide clues as to what they are for? They should do and it's the reason that you should always try to name variables well. Once your coding gets more complex it is very confusing if the variables are all called things like 'MyNumber1' 'MyNumber2'.

See the following table of variables.

The variables are:

Variable name	Type	Purpose
haveDiamond	number	0 if the player doesn't have the diamond 1 if the player does
haveKey	number	0 if the player doesn't have the key 1 if the player does
room	number	The number of the room that the player is in
finished	number	0 if the player has not escaped 1 if the player has escaped
answer	string	This gets set to whatever the player types in.



- Functions:** The next lot of code calls functions for each room, these are called rooms0, rooms1 etc. The functions are only run when they get called by the main code. Inside each function is the code that should be run when the player is in this room. This includes the text about where the doors are and code to test if the player has collected the diamond or has the key etc.

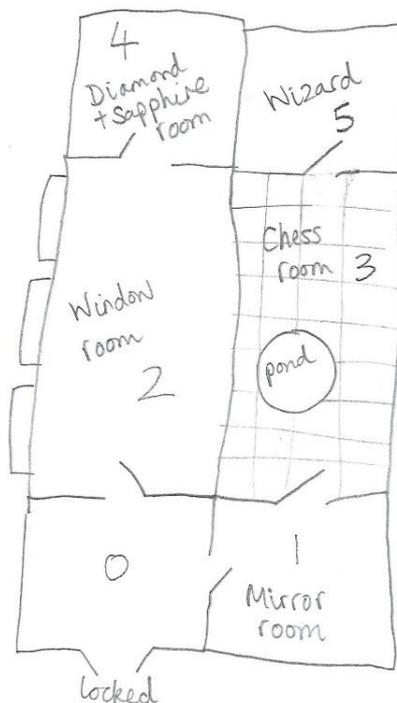
The Text Adventure functions pictures are for your use if you wish to use them. Some children might find it helpful to have printed versions that they can refer to when changing code.

Here is the example for rooms1

```

VAR create function rooms1
  ? prompt
  'You are in a room where the walls are all mirrored. There are doors to the north and west. Which way would you like to go?'
  variable name? = set to Input
  IF if answer equals 'n' Then
    variable name? = set to 3
  Else
    IF if answer equals 'w' Then
      variable name? = set to 0
    Else
      ! alert 'Sorry I don\'t understand.'
    
```

- Can children use this information to label the rooms on their map with the room numbers? In the example above the room number is 1 and you can tell from the text that this is the room with mirrored walls.
- Children’s maps should end up like this:





- **Repeat:** After the functions is the main code that runs when the program starts. The first line of this says the following code should keep being repeated until **finished = 1**. Looking at the variables above, this means that it continues until the player has escaped.
- This code uses if/else statements to establish which room the player is in and then run the appropriate function for that room.

```

repeat until finished equals 1
  IF room equals 0 Then
    rooms0 call
  Else
    IF room equals 1 Then
      rooms1 call
    Else
      IF room equals 2 Then
        rooms2 call
      IF room equals 3 Then
        rooms3 call
      Else
        IF room equals 4 Then
          rooms4 call
        Else
          IF room equals 5 Then
            rooms5 call

```



- **Conclusion:** This is the code that runs when finished = 1. This is when the player has escaped and finished the adventure. Can children explain what this does?



- Ask children to change the room descriptions to their own ideas and then play the game again. How much can they make the game feel different just by changing the text?

NOTE: Sometimes when text is changed, there is a bug which reverts the text to what it was before when they go to change another part of the code. If this happens, children should drag

another  **command below the existing one, enter the new text and then delete the original text by dragging it to the trash bin in the bottom right of the screen.**

- The diamond, wizard and key could become something else. Perhaps the player is Alice lost in Wonderland and she needs to give the wizard something, so he can give her a shrinking potion? Perhaps the player is a master spy who needs to find a vital clue and report it back to the wizard\Q\Prime Minister?

- Play the resulting games as a class and enjoy!

- **Extension**

In Unit 6.5 children will have the opportunity to change the rooms in this adventure. Some children might be eager to try this out now. Encourage them to map out their rooms and start with small changes then build upon them. Save working code often so that if it all goes wrong, they don't lose everything.



Appendix 1 - 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.

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.

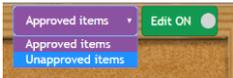


Appendix 2 - Approving work on a displayboard

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  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.

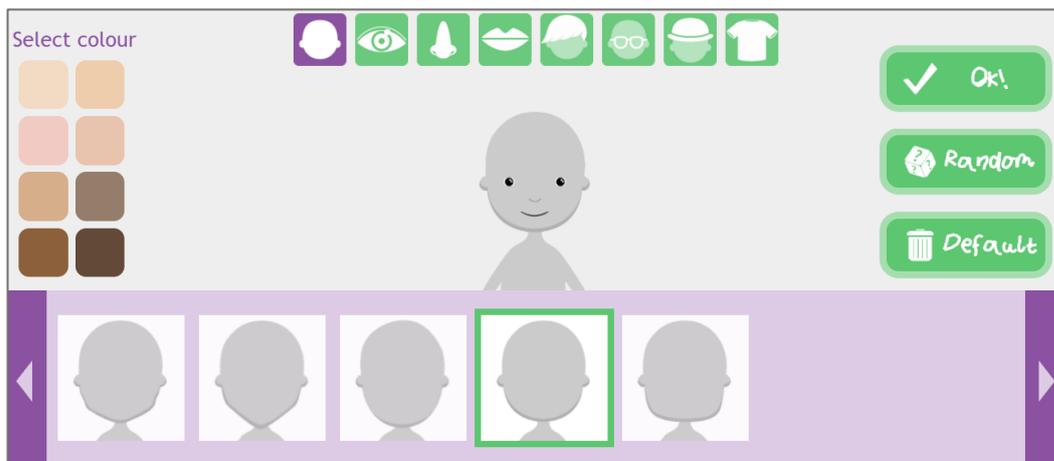


Appendix 3 – The Avatar tool

Purple Mash Avatar creator; this is opened by clicking on the user portrait near the top right of the screen next to the Logout button.



1. Click on the image (which will just be a grey outline image if an avatar has not yet been created).
2. Create an avatar or click the Random button to make a random avatar; this is a quick way to make several different avatars.



3. Click the **download** button to save an image file of this avatar to your device. Create enough of these to use in your game. The Aliens version has 20 aliens.
4. To make game cards, print the image files to the desired size.



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	<p>Children are beginning to be able to turn a more complex programming task into an algorithm by identifying the important aspects of the task (abstraction) and then decomposing them in a logical way with support (lessons 1/2 & 5).</p> <p>They can then use this design to write a program using 2Code.</p> <p>Pupils understand sequence, selection and repetition in programs and can use them in their simplest forms. They will require support when combining these aspects e.g. using selection within a repeat in a game (lesson 1/2 and lesson 6).</p> <p>With support, children can plan, design and create a simple program that includes a single variable relating to timing. They can also include a button which will launch another program (Unit 6.1 Lesson 1 and 2).</p> <p>They will usually require support to make use of variables and manipulate variables in their code and in understanding the way that functions are beneficial (lesson 1/2, lesson 3 & lesson 6).</p> <p>As their coding becomes more complex, they will require support to tackle debugging in a logical rather than a trial-and-error method.</p> <p>Children can make good attempts to ‘read’ code and predict what will happen in a program (lessons 3 & 5). They can usually interpret a program in parts but will need support to put the separate parts of a complex algorithm or program together to explain the program as a whole (lesson 6).</p>
Expected	<p>Children are beginning to be able to turn a more complex programming task into an algorithm by identifying the important aspects of the task (abstraction) and then decomposing them in a logical way using their knowledge of possible coding structures and applying skills from previous programs.</p> <p>They can then use this design to write a program using 2Code (lessons 1/2, 3 & 5).</p> <p>Children can translate algorithms that include sequence, selection and repetition into code and their own designs show that they are thinking of how to accomplish the set task in code utilizing such structures including nesting structures within each other (lesson 1/2 and lesson 6).</p> <p>Children can plan, design and create a program that includes variables relating to timing and scoring along with buttons which launch other programs (Unit 6.1 Lesson 1 and 2). Furthermore, children will consider how to organise their code using multiple tabs (Unit 6.1 Lesson 2).</p> <p>They use functions within their code to eradicate unnecessary code such as shape creation (Unit 6.1 Lesson 3).</p>



Assessment Guidance

	<p>Their coding displays an understanding of the function of variables in coding (lesson 1/2 and lesson 6), outputs such as sound and movement (lesson 1/2), inputs from the user of the program such as button clicks (lessons 3, 4 & 5) and the value of Functions (lesson 3).</p> <p>Children can make good attempts to ‘read’ code and predict what will happen in a program (lessons 3 & 5). They can usually interpret a program in parts and can make logical attempts to put the separate parts of a complex algorithm or program together to explain the program as a whole (lesson 6).</p> <p>Children test and debug their program as they go and can use logical methods to identify the approximate cause of any bugs but might need support to identify the specific line of code that is causing the problem as the complexity of the programs increases. They try to improve and debug their own programs (all lessons).</p> <p>Within their programs, they can use features such as interactivity with the end users with the desired effect of adding greater impact. (Unit 6.1. Lesson 4).</p> <p>Most children demonstrate a secure understanding of the impact of changing the position of instructions within 2Code. With this knowledge, they can demonstrate use of the tabs feature to carefully section code for the intention of easier debugging and less code error, as their coding becomes more complex. They know that instructions can be shortened by using the ‘function’ feature and are aware of the impact on their desired outcomes if they haven’t thought this through fully (Unit 6.1 Lesson 3).</p>
Exceeding	<p>Children can turn a more complex programming task into an algorithm by identifying the important aspects of the task (abstraction) and then decomposing them in a logical way using their knowledge of possible coding structures and applying skills from previous programs. They can then use this design to write a program using 2Code (lessons 1/2). Children’s design shows that they are thinking both, of the required task, and of how to accomplish this in code. Children test and debug their program as they go and can use logical methods to identify the approximate cause of any bugs then test systematically to identify the specific line of code that is causing the problem.</p> <p>Pupils intuitively grasp the concepts of selection, repetition and variables and make use of the various commands to use input from users and produce output including sound and movement. 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 (lesson 1/2 and lesson 6). They are also thinking about good structure to their code with a view to debugging such as the use of tabs and functions to organise code and the naming of variables (lessons 1/2 & 3).</p>