

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**



**Subject:** Purple Mash Coding

L1 6.1

L2 6.1

L3 5.1

L3 6.1

L4 6.1

L6 6.1

**Key Concept/ Theme:** • To design a playable game with a timer and a score. • To plan and use selection and variables. • To understand how the launch command works. • To use functions and understand why they are useful. • To understand how functions are created and called. • To use flowcharts to create and debug code. • To create a simulation of a room in which devices can be controlled. • To understand how user input can be used in a program. • To understand how 2Code can be used to make a text-adventure game.

• To begin to simplify code. • To create a playable game. • To understand what a simulation is. • To program a simulation using 2Code. • To know what decomposition and abstraction are in computer science. • To take a real-life situation, decompose it and think about the level of abstraction. • To understand how to use friction in code. • To begin to understand what a function is and how functions work in code. • To understand what the different variables types are and how they are used differently. • To understand how to create a string. • To understand what concatenation is and how it works.

**Prior Learning links:**

	Cycle A	Cycle B
Year 1/2	Unit 1.1 Coding • Introducing block coding • Objects and actions • Events (Click event, sound output) • Executing a program • Design view: Planning  Unit 1.4 Lego Builders • Algorithms • Logical decision making • Sequencing instructions • Following instructions  Unit 2.1 Coding • Algorithms • Collision detection • Timers • Object types • Buttons • Debugging	Unit 1.5 Maze Explorers • Coding a 'turtle' • Creating programs using sequencing and repeat. • Visual use of the Logo programming language. • Program logic and structure.  Unit 2.4 Questioning • Logical decision processing. • Forward planning to achieve a solution

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

<p style="text-align: center;">Year 3/4</p>	<p>Unit 3.6 Branching databases  <ul style="list-style-type: none"> <li>• Logical decision processing • Modelling selection on a binary model</li> </ul> <p>Unit 4.1 Coding  <ul style="list-style-type: none"> <li>• Code, test, debug process • IF statements • Repeat Until and IF/ ELSE Statements • Number Variables</li> </ul> <p>Unit 5.1 Coding  <ul style="list-style-type: none"> <li>• Efficient Coding • Simulating a Physical System • Decomposition and Abstraction • Friction and Functions • Introducing Strings • Text Variables and Concatenation</li> </ul> </p></p></p>	<p>Unit 4.5 Logo  <ul style="list-style-type: none"> <li>• Text-based coding • Utilize understanding of coding structures</li> </ul> <p>Unit 4.6 Animation  <ul style="list-style-type: none"> <li>• Sequencing and animation in logical steps.</li> </ul> <p>Unit 4.1 Coding  <ul style="list-style-type: none"> <li>• Code, test, debug process • IF statements • Repeat Until and IF/ ELSE Statements • Number Variables</li> </ul> <p>Unit 5.1 Coding  <ul style="list-style-type: none"> <li>• Efficient Coding • Simulating a Physical System • Decomposition and Abstraction • Friction and Functions • Introducing Strings • Text Variables and Concatenation</li> </ul> </p></p></p></p>
<p style="text-align: center;">Year 5/6</p>	<p>Unit 6.5 Text Adventures  <ul style="list-style-type: none"> <li>• Development from textbased coding • Maintaining a mental map. • Debugging skills</li> </ul> <p>Unit 6.1 Coding</p> </p>	<p>Unit 6.1 Coding</p>

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

**Key Vocabulary**

**Action**  
 The way that objects change when programmed to do so. For example, move or change a property.

**Co-ordinates**  
 Numbers which determine the position of a point, shape or object in a particular space.

**Execute\Run**  
 Clicking the Play button to make the computer respond to the code. Execute is the technical word for when you run the code. We say, 'the program (or code) executes.'

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

**Event**  
 An occurrence that causes a block of code to be run. The event could be the result of user action such as the user pressing a key (**when Key**) or clicking or swiping the screen (**when Clicked, when Swiped**) or when objects interact (**collision**). In 2Code, the event commands are used to create blocks of code that are run when events happen.

**Command**  
 A single instruction in a computer program.

**Decomposition**  
 A method of breaking down a task into manageable components. This makes coding easier as the components can then be coded separately and then brought back together in the program.

**Debug/Debugging**  
 Fixing code that has errors so that the code will run the way it was designed to.

**Flowchart**  
 A diagram that uses specifically shaped, labelled boxes and arrows to represent an algorithm as a diagram.

**Function**  
 A block or sequence of code that you can access when you need it, so you don't have to rewrite the code repeatedly. Instead, you simply **call** the function each time you want it.

**Object**  
 Items in a program that can be given instructions to move or change in some way (action). In 2Code Gorilla, the **object types** are button number, input, text, shape turtle, character, object, vehicle, animal.

**Procedure**  
 An independent code module that fulfils a task and is referenced within a larger body of code. In 2Code a procedure might be coded as a function.

**Selection**  
 Selection is a decision command. When selection is used, a program will choose which bit of code to run depending on a condition. In 2Code selection is accomplished using **'if'** or **'if/else'** statements.

**Tab**  
 In 2Code, this is a way to organise a program into separate pages (tabs) of code.

**Input**  
 Information going into the computer. This could be the user moving or clicking the mouse, or the user entering characters on the keyboard. On tablets there are other forms such as finger swipes, touch gestures and tilting the device. In 2Code the commands **prompt for input** and **get input** are used to prompt the user to enter typed input and then use this input.

**Properties**  
 These determine the look and size of an object. Each object has properties such as the image, scale and position of the object.

**Sequence**  
 This is when a computer program runs commands in order.

**Simulation**  
 A model that represents a real or imaginary situation. Simulations can be used to explore options and to test predictions.

**Timer**  
 Use this command to run a block of commands after a timed delay or at regular intervals.

**Launch Command**  
 This command will open another Purple Mash file or an external website that you specify when it is called.

**Output**  
 Information that comes out of the computer e.g. **sound, prompt, alert** or **print to screen**.

**Predict**  
 Use your understanding of a situation to say what will happen in the future or will be a consequence of something

**Repeat**  
 This command can be used to make a block of commands run a set number of times or forever.

**Repeat Until**  
 In 2Code this command will repeat a block of commands until a condition is met.

**Variable**  
 A named area in computer memory. A variable has a **name** and a **value**. The program can change this variable value. Variables are used in programming to keep track of things that can change while a program is running.

**Vocabulary:**

https://www.purplemash.com/

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

**Key Vocabulary**

**Abstraction**

A way of de-cluttering and removing unnecessary details to get a program functioning.

**Concatenation**

The action of linking a mixture of strings, variable values and numbers together in a series.

**Efficient**

In coding, simplified code runs faster and uses less processing memory, it is said to be more efficient.

**Action**

The way that objects change when programmed to do so. For example, move or change a property.

**Debug\ Debugging** Fixing code that has errors so that the code will run the way it was designed.

**Flowchart**

A diagram that uses specifically shaped, labelled boxes and arrows to represent an algorithm as a diagram.

**Algorithm**

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

**Decomposition**

A method of breaking down a task into manageable components. This makes coding easier as the components can then be coded separately and then brought back together in the program.

**Event**

An occurrence that causes a block of code to be run. The event could be the result of user action such as the user pressing a key (**when Key**) or clicking or swiping the screen (**when Clicked, when Swiped**) or when objects interact (collision). In 2Code, the event commands are used to create blocks of code that are run when events happen.

**Nesting**

When coding commands are put inside other commands. These commands only run when the outer command runs.

**Physical System**

In this context, this is any object or situation that can be analysed and modelled. For example modelling the function of a traffic light, modelling friction of cars moving down surfaces or modelling the functions of a home's security system.

**Function**

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

**Object**

Items in a program that can be given instructions to move or change in some way (action). In 2Code Gorilla, the **object types** are button number, input, text, shape turtle, character, object, vehicle, animal.

**Properties**

These determine the look and size of an object. Each object has properties such as the image, scale and position of the object.

**Selection**

A conditional decision command. When selection is used, a program will choose which bit of code to run depending on a condition. In 2Code selection is accomplished using **'if'** or **'if/else'** statements.

**Input**

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

**Output**

Information that comes out of the computer e.g. **sound, prompt, alert** or **print to screen**.

**Repeat**

This command can be used to make a block of commands run a set number of times, until a condition is met or forever.

**Sequence**

This is when a computer program runs commands in order.

**Simplify**

In coding this is used to describe modifying the code to complete the same process with less lines of code.

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

**Timer**

Use this command to run a block of commands after a timed delay or at regular intervals.

**Variable**

A named area in computer memory. A variable has a **name** and a **value**. The program can change this variable value. Variables are used in programming to keep track of things that can change while a program is running. In 2Code, variables can be **strings**, **numbers** or **computer-generated** variables to control objects of a type.

---

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

**Key Images**

Design  
Open design mode in 2Code.

Exit Design  
Switch to code mode in 2Code.

My Code +  
Add a new Tab to your code

change variable  
A change variable block.

print print to screen 'MyName' + random word Verb + 'with the dog.'  
Example of combining variables and strings to print to the screen

create number myNumber1 =  
Select the type of variable  
number string clone function  
Creating a variable in 2Code

create function myFunction1  
myCar1 angle = set to 90  
Creating a function in 2Code

fn call myFunction1  
Calling a function in 2Code

**Key Images**

Design  
Open design mode in 2Code.

Exit Design  
Switch to code mode in 2Code.

change variable  
A change variable block.

print print to screen 'MyName' + random word Verb + 'with the dog.'  
Example of combining variables and strings to print to the screen

create number myNumber1 =  
Select the type of variable  
number string clone function  
Creating a variable in 2Code

Character Vehicle Collision +  
Add a new Tab to your code or move code blocks between tabs

create function Square  
repeat 4 times  
Artist forward 5 steps  
Artist turn 90 degrees  
when clicked Button  
call Square  
A function called 'square' that is called by clicking on a button called btnSquare.

Key Images:

**Resources needed for each lesson – 2dos to set.**

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

- Coding Vocabulary Quiz Y6
- 2Code Game Planner, Leaflet - Coding Planner, Storyboard Template
- Making a Timer and Score Pad guide.
- Splatty Bug. • Free Code Gorilla found on the main 2Code screen in the Gorilla section.
- (Optional) Vocabulary flash cards. The teacher flash cards have been created so you can print them on A4 paper, cut them to size, fold them in half and glue them together. You can display and use these throughout coding lessons to support use of vocabulary. Preparation
- Set Free Code Gorilla as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:

Year:	Y6	▼
Subject:	Computing	▼
Strand:	Computer Science	▼
Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts.		<input checked="" type="checkbox"/>
Use sequence, selection and repetition in programs; work with variables and various forms of input and output.		<input checked="" type="checkbox"/>
Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs.		<input checked="" type="checkbox"/>

- Print/ copy your choice or a mixture of 2Code Game Planner, Leaflet - Coding Planner, Storyboard template for children to use to plan their programs.
- Open 2 Tabs, Splatty Bug in one and Free Code Gorilla in another.
- Create a display board for the class to share their programs to. Details of how to do this are given in Appendix 1

**L3 5.1**

**L3 6.1**

- Functions.
- Free Code Gorilla found on the main 2Code screen in the Gorilla section.
- For further guidance on Functions (otherwise known as Procedures) you could use Year 4 Unit 4.5 Using Logo – particularly Lesson 4.

Year:	Y6	▼
Subject:	Computing	▼
Strand:	Computer Science	▼
Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts.		<input checked="" type="checkbox"/>
Use sequence, selection and repetition in programs; work with variables and various forms of input and output.		<input checked="" type="checkbox"/>
Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs.		<input checked="" type="checkbox"/>

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

- Set Functions as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:
- Set Free Code Gorilla as a 2Do.

**L4 6.1**

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.

- Billy's Bedroom Simulation.
- Billy's Bedroom Flowchart.
- A piece of plain paper and a pencil for each child.
- Set Billy's Bedroom Simulation as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:

Year:	Y6	▼
Subject:	Computing	▼
Strand:	Computer Science	▼
Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts.		<input checked="" type="checkbox"/>
Use sequence, selection and repetition in programs; work with variables and various forms of input and output.		<input checked="" type="checkbox"/>
Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs.		<input checked="" type="checkbox"/>

- Print/ copy Billy's Bedroom Flowchart enough for one each for the children.

**L6 6.1**

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.
- Aliens Database.
- Guess the Alien.
- Is It Raining and Reginald Rocket Examples from Y4.
- User Input Example 1.
- User Input Example 2.



**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

- Free Code Gorilla found on the main 2Code screen in the Gorilla section.
- Set Guess the Alien as a 2Do. Add in Aliens Database and Alien Cards as supporting documents – you will need to save these documents to your work folder before you set the 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:

Year:	Y6	▼
Subject:	Computing	▼
Strand:	Computer Science	▼
Design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts.		<input checked="" type="checkbox"/>
Use sequence, selection and repetition in programs; work with variables and various forms of input and output.		<input checked="" type="checkbox"/>
Use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs.		<input checked="" type="checkbox"/>

- Print/ copy Alien cards enough for one each for the children
- Open 4 tabs: User Input Example 1, User Input Example 2, Is It Raining and Reginald Rocket.
- Create a display board for the class to share their programs to. Details of how to do this are given in Appendix 1.

1. **Deeper learning questions: Why is coding useful? Why do we need to be able to decompose code?**

Unit 6.1 **Reconnection:** assess children knowledge – crash course if needed. – go over previous vocab

Lesson 1 & 2 **LO:** To design a playable game with a timer and a score.  
To plan and use **selection** and **variables**.  
To understand how the **launch command** works.

**Activity:**  
Go over new vocab for lesson

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

Use the [Coding Vocabulary Quiz Y6](#) as a class to review prior learning. It is set up so that you attempt all questions and then click the 'Hand In' button to check the answers. Click 'OK' to see which are correct and incorrect:

Run through the answers to the questions together: Use slides 5 and 6 which have definitions.

Explain to children that now they are in Year 6, their coding skills are becoming more sophisticated. Emphasise how important the planning stage for their programs is becoming as a result. Explain that over the first two lessons they will plan and make a game that reminds them of lots of the skills they have learnt in primary school.

This lesson will focus on the planning but you will start by making a game together to remind children of some of the skills they have learnt. Start the practical part of the lesson by creating a game with them using one of the guided lessons in 2Code.

Display [Splatty Bug](#) on the board and complete it together as a class. You could complete it on the board or get children to open it as well and complete it with you one step at a time.

Refer to key vocabulary and concepts as you go through it with the children including **selection, IF/ELSE statements, prompt, number variable, variable, timer (after/ every), event, object, action, co-ordinates**

Use the slide to remind children about decomposition and explain that Splatty Bug asks us to add code for one part of the program at a time.

**Stage 1:**

Start by looking at the design together and ask children to predict how the game will work.

Identify myTime as a variable – in Design View you can see the value of myTime indicates the 'Time left'. Review number variables – what they are and how they work.

Use the slide to go through stage 2 with the children.

**Stage 2:**

Add in a **timer** that decreases the myTime **variable** by 1 every second and triggers an **alert** when the time has run out.

Add a restart block at the end.

Review **selection – IF/ ELSE statements** and discuss how they work.

**Splatty Bug Stage 3**

- Add in a click event that hides the bug that has been clicked on and increases the score.
- Once you have added 'When clicked Any Bug' you will need to drag across and then select 'change variable' to include the Clicked bug **variable** – the value of which will be set by the click **event** when it is run.
- Discuss **simplifying** the code and how sometimes **variables** are used to reduce the amount of code needed.

**Splatty Bug Stage 4**

- Add code to show the splat and set it's X and Y values to equal those of the Clicked bug **variable**.
- Review use of **co-ordinates** and how they help position objects in a **scene**.

Use the slide to go through stage 5.

**Stage 5:**

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

Read the information and click on the 'Info' button to see all the code that you would have needed to add if you hadn't used the 'Any bug' feature and 'Clicked bug' variable in stage 3.

Take suggestions from the children as to how this game could be further developed. Can anybody suggest why computer programmers are often called **developers**? Save the final version of the game you've made.

Open [Free Code Gorilla](#) in a new **tab** and remind children how to add **backgrounds** and **objects** – reminding them that different **object** types have different options in the **attributes** table and different possible actions in the code.

Ask children to open [Free Code Gorilla](#) from their 2Dos area then browse the different **backgrounds** and **objects** for a short while to get ideas for the game they will design.

Introduce children to the **Launch command** and demonstrate how it can be used in 2Code to Launch another activity – show how this works for a Purple Mash activity and/ or an external website.

- 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?
- Explain to them that you want them to start with a simple version of the game that they can then develop and enhance once it's working.
- 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 **objects**, the **events** that will occur and any **variables** required to keep a track of things.
- They should have [Free Code Gorilla](#) in front of them so they can explore **background** and clipart libraries etc to help them plan.
- 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, display this on the board to remind them of what they can do:
  - 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, functions and strings).
  - Adding a timer and/or score pad to games.
  - Adding buttons that take the user to places outside the program.
  - Using friction, angles and speed attributes of objects.
  - Using co-ordinates to help control placement of objects in the scene.
- Give children time to think about what they want their games to do and plan in detail before they start coding – tell children that at the end of this lesson they will swap plans with a partner who will read them through and see if they can explain how their game would work.

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

	<p>Ask children to get out their program plans from last lesson and complete them if they haven't already.</p> <p>Children to open <a href="#">Free Code Gorilla</a> from their 2Dos to create programs using their plans – their games should feature at least a <b>timer</b> and a <b>score</b>.</p> <p>Differentiation: Pupils can be given copies of the <a href="#">Making a Timer and Score Pad guide</a> to help them implement their designs if needed.</p> <p>Ask children to save their program, then use the display board (see <a href="#">Appendix 1</a>) to share great examples with the class, discussing the code that has been used to make them work. Emphasise the importance of the design, code, test and debug process.</p> <p>What challenges did they come across?</p> <p>Go over vocab and success criteria.</p> <p><b>Extension: How could you improve if you did it again?</b></p>
3. Unit 5.1 Lesson 3	<p><b>Deeper learning questions: can you deconstruct a code to say what happens? Have you removed all the unnecessary detail?</b></p> <ul style="list-style-type: none"><li>② <b>Is your plan broken down into manageable parts?</b></li><li>② <b>Could you follow your plans to create your meal or board game?</b></li></ul> <p><b>Reconnection:</b> go over vocab and prior learning</p> <p><b>LO:</b> To know what decomposition and abstraction are in Computer Science. To take a real-life situation, <b>decompose</b> it and think about the level of <b>abstraction</b>. To use <b>decomposition</b> to make a plan of a real-life situation</p> <p><b>Activity:</b> Go over new vocab for lesson</p> <p>Reveal slide to go through vocabulary and definitions of decomposition and abstraction. discuss decomposition and the example withing it As a class, open the <a href="#">Football example</a> and explore slide prompts together.</p>

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

	<p>Explore the Abstraction slide together. Explain to the class that, today, they are going to try these two processes – <b>decomposition</b> and <b>abstraction</b> – to design and write <b>algorithms</b> for either a simple board game (perhaps share some of the examples you have), or the scenario of planning and making a meal. Explore the ‘pie’ example on the slide. Children could work in pairs to discuss and feedback to the class. Use the slide to introduce the activity. Children to plan a process (meal, board game etc) use the writing frame: <a href="#">Decomposition and Abstraction</a>.</p> <p>Go over vocab and success criteria.</p> <p><b>Extension: Can you write a code for a friend to decompose?</b></p>
4.  Unit 6.1  Lesson 3	<p><b>Deeper learning questions: Did your code work? Why or why not?</b></p> <p><b>Reconnection:</b> Go over previous vocab.</p> <p><b>LO:</b> To use functions and understand why they are useful. To understand how <b>functions</b> are created and <b>called</b>.</p> <p><b>Activity:</b> Go over new vocab for lesson</p> <p>Explain that in this lesson we will be learning more about <b>functions</b>. Ask children to tell you where they have used <b>functions</b> before – they used them in Year 5 Lesson 4 ‘Friction and Functions’ when they made a Football Game using the freecode. They also practised using them in the Year 5 lesson introducing them to <b>strings</b>. Ask children to log into Purple Mash and have a go at the <a href="#">Functions</a> guided lesson that you’ve set as a 2Do. Review children’s progress and discuss how they’ve got on using <b>functions</b>. Use this slide as a guided and shared teaching opportunity before children make their own Turtle Game later in the lesson. Children could open their own version of Free Code Gorilla. Open <a href="#">Free Code Gorilla</a> by clicking on the ‘Preview’ button in the 2Do.</p>

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

Click on 'Design', then add a **turtle object**, a **shape object**, and a **text object**.  
Name the **turtle object** 'artist' in the **attributes** table that appears to the left when you click on it.  
Click on the shape and name it 'red'. Set the number of sides to 4, the size to 1 and the angle to 45.  
Name 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 **attributes** table.

Save the work and click on 'See Code'.

- Ask the children to help you make a **function** named 'Square' which will program the **turtle object** to draw a square.
- Children 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.
- **Note:** it is possible to use the 'create variable' codeblock to create a function, and also possible to use the 'change variable' codeblock to call a function although using the 'create function' and 'call function' codeblocks are recommended.

With the code on the previous slide, the turtle should move in a square shape, but it won't draw a square because you haven't programmed a 'Pen down' command.

Ask children to help you look at the code to see if it can be developed in a way that makes the turtle draw the square.

Your code might now look like either of the examples on this slide.

Whichever it looks like, run the program, click on the turtle object and watch it draw a square. Stop running the program and look back at the code, see if the children can spot the alternative place to put the pen down/ pen up commands to achieve the same outcome.

Can they help you further develop the code so that when the red square is clicked, it changes the pen colour to red?

Ask children to open [Free Code Gorilla](#) from their 2Dos area and have a go at creating a turtle program using functions.

What other shapes can they create functions for? Show them the images to illustrate the type of program you are expecting them to create.

Tell children to arrange their code into tabs so it is easier to debug – remind them about decomposition.

Go over vocab and success criteria.

**Extension: Use your friends – can they think of ways to improve each other or figure out the code they used.**

5. **Deeper learning questions: What's in the room? How do we know? Why do we need to debug**

Unit  
6.1

**Reconnection:** Go over previous vocab and key learning.

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

Lesson 4	<p><b>LO:</b> To use flowcharts to test and debug a program. To create a <b>simulation</b> of a room in which devices can be controlled.</p> <p><b>Activity:</b> Go over new vocab for lesson</p> <p>Explain that in this lesson we are going to work on a <b>simulation</b>, ask children if they can remember what a <b>simulation</b> is – they covered it in Year 5. Reveal definition on slide.</p> <p>Open the <a href="#">Billy's Bedroom Flowchart</a> – note that there are several <b>flowcharts</b> on one page here, and each <b>flowchart</b> represents a <b>procedure</b> for something that happens in Billy's bedroom.</p> <p>Give each child a printed copy of <a href="#">Billy's Bedroom Flowchart</a>. Give children 6 minutes to draw a bedroom that could be Billy's – emphasise that it doesn't need to be a work of art, it's just a quick sketch. What is in the room?</p> <p>Display <b>slide 6</b>. Open <a href="#">Billy's Bedroom Simulation</a> in 2Code and share with the children.</p> <p>Does it contain all the same things as your picture? Probably, but quite likely in different places.</p> <p>Click on the Run button to <b>run</b> the code. Watch it for a while, and notice it switching between day and night.</p> <p>Ask children: What are the differences?</p> <p>What might happen at night that doesn't happen during the day? How long is the day/ night?</p> <p>Give children a couple of minutes in talking partners to look at their printed <b>flowcharts</b> and <b>predict</b> what will happen when different <b>objects</b> are clicked on.</p> <p>Display <b>slide 7</b>. <b>Run</b> the <b>simulation</b>. Ask children to suggest objects for you to click on, does the simulation run like they were expecting? After you click on each, discuss if they think it has been coded correctly, or if they think it needs correcting/ <b>debugging</b>.</p> <p>Display <b>slide 8</b>. Set the children the task of interpreting the <b>flowcharts</b>, <b>debugging</b> existing code and getting the rest of the devices in Billy's room working.</p> <ul style="list-style-type: none"><li>- There are no <b>flowcharts</b> for some <b>objects</b> (car, robot, computer) they can plan the <b>procedures</b> for these in flowcharts on paper or in 2Chart in another <b>tab</b>.</li><li>- They might need to add more <b>objects</b> e.g. there is no remote control for the car, they will need to add <b>buttons</b> to look like a remote control. They can look at the <b>buttons</b> on the drawer knobs to get an idea for this.</li><li>- There are <b>functions</b> to make the fairies fly, but these do not have any code in them.</li></ul> <p>They should start with the things that look simplest and then try the harder ones. They could add more devices and create <b>flowcharts</b> for them if they have time.</p>
-------------	---

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

	<p>Go over vocab and success criteria.</p> <p><b>Extension: Make one for a friend to debug</b></p>
6. Unit 6.1 Lesson 6	<p><b>Deeper learning questions: Why is having a map helpful for solving a game like this? Can you explain how the code works?</b></p> <p><b>Reconnection:</b> Remind children of text adventure learning from previously.</p> <p><b>LO:</b> To understand how 2Code can be used to make a text-based adventure game.</p> <p><b>Activity:</b> Go over new vocab for lesson</p> <p>Use the slide to introduce 'Text Adventure'. Explain to children that they are going to begin by playing a simple text adventure written using 2Code, and then they will have a chance to change the code to make the adventure their own.</p> <p>Ask children to open the <a href="#">Y6 Text Adventure</a> from their 2Dos and press play to get started.</p> <p>Give the children some time to play the adventure, see who is first to solve it. Ask them to sketch a map as they are playing it.</p> <p>See if they can work out that the aim of the game is to collect the diamond and give it to the wizard, who will then provide them with a key to escape.</p> <p>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.</p> <p>It is useful for everyone to have a simple map when discussing the code. If children's maps are varied, let them refer to the image on the slide.</p> <p>The next stage is to look at the code. Children will be making changes to the code, so it is worth talking about the need to save their files frequently after testing that they work.</p> <p>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 'exit without saving' and then click on 'continue' in the 2Do to open your last saved file.</p> <p>Remind children of the undo button which appears when you have changed something so that you can undo; often this will fix the problem.</p> <p>In the code, each room has been given a number.</p> <p>Players start in room 0. Look through the code together and try to write the numbers on the map.</p>



**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

Look at your map, Slide 8 has help for rooms 0, 1 and 2

Now that the rooms are labelled with the numbers, look through the code in even more detail – starting by looking at the variables, then by looking at functions, then by looking at how the repeat command is used followed by how the game concludes. This will help children understand how the game works and check the numbers on their rooms are correct.

Use the slide to explain to the children that the first 5 lines of the code are where various variables are set.

Articulate the following points:

- 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'.

**Functions:**

- The next section of code sets a **function** for each room, these are called 'rooms0', 'rooms1' etc.
- The functions are only run when they get **called** by the main code.
- Each **function** starts with a **prompt for input** – because when you get into each room you are given some information and asked a question, and what happens next depends on your answer.
- In the resources you'll find Text Adventure Functions Code. These are for you to use if you wish. Some children might find it helpful to have printed versions that they can refer to when changing code.
- By looking at the prompt we can confirm that room 1 is the mirror room – you could use the information in the functions to check the room numbers on your map.

Use the slide to explore the next section of code – Repeat.

Articulate the following points to children:

- This code is after the functions but will run when the program is started.
- At the start of the code the variable 'room' was set to equal 0, so when the program is run and the first IF statement is checked, it will be true and the first function to be called will be 'rooms0'.
- When the player moves rooms the 'room' variable will change. This repeat until command will run through the IF/ ELSE statements until it finds a statement that is true and then it will call the function for that room.
- There is an IF/ ELSE statement for each room.

**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 draw a blank plan of the building and plan their own room layouts and features. Leave the doors in the same place.

Challenge them to plan to change the code for the text adventure they have been playing and exploring and make their own text adventure:

Reveal ideas on the slide.

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

Challenge children to change the code for the text adventure they have been playing and exploring and make their own text adventure.  
NOTE: Sometimes when text is changed it reverts the text to what it was before when they go to change another part of the code. When children change text, tell them to make sure they click out and check it has changed before going on to change something else. If they are having problems they could drag in a new prompt box enter their text into that, and delete the original one.

- Share children's work to a displayboard (see [Appendix 1](#)) and give them some time to play each other's games.

Share and celebrate achievements. Refer back to the lesson aims and discuss their understanding.

Go over vocab and success criteria.

**Extension: can you use others ideas to make your game better.**

**End of unit quiz & reflect on gaps from the unit:**

Unit 4.1 Quiz – found on unit page on PM (year 4)

Questions:

**End of unit quiz & reflect on gaps from the unit:**

Unit 6.1 Quiz – found on unit page on PM (year 6)

What name is given to a block or sequence of code that you can access when you need it, so that you do not have to rewrite the code repeatedly?

Which command would run a block of code after a set delay, or at a regular intervals?

Which of these icons would you use to create a new coding window tab?

What does the variable 'Anna' do in this program?

In this program, which tab contains the functions that allow specific procedures to run?

Which of these code blocks is used to obtain user input without a pop up window appearing on screen?

In this program, how is the value of the created variable 'pet' attributed?

In the program shown in the last question, what name is given to the linking together of things as seen in the three 'print to screen' command blocks?

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

How would you debug this example of 'Billy's Bedroom simulation' to ensure that the fairies could be seen flying at night time?

Match the code block colour to the command type.

Look at this flowchart. What do you think this program will do?

Look at this extract from a text adventure game. What causes the variable 'Door' to change in this program?

End of unit vocabulary check,

Questions:

**End Points:**

How can you use Tabs in 2Code Gorilla?

Tabs are used to organise you code and make it more readable. This also makes it easier to debug. Give the Tabs useful names to help with this.

What is a function in coding? Give an example that you have used in 2Code Gorilla.

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. You call the function each time you want it. In a turtle program you could have a button that will make the turtle draw a square each time you click it. In the text adventure, there were functions for each room that were called when the user navigated to the room.

In 2Code Gorilla, how can a program receive user input?

When the user clicks on an object, when the user presses keys or swipes the screen with the mouse, the 'Get Input' and 'Prompt for input' commands. On a touchscreen: when the screen is touched or swiped.

**Evaluation:** What have the end of unit quizzes, pupil self-reflections and termly work told you about what the children can remember and recall? What are the gaps? Ensure that the areas that need further reinforcement are documented in the next subject unit MTP. **Plan in time to revisit gaps within units, determined by the quizzes.**

•

**Pioneer Federation**  
**Medium term plan**  
**Cycle B, Term 6**  
**ICT**

--