**Computing Medium Term Planning**

|  |  |  |
| --- | --- | --- |
| **Term:** Summer 2 | **Year:** 5/6 | **Topic/Unit:** 6.1 Coding and 5.1 Coding |
| **Key Vocabulary**  **Lesson 1 &2- algorithm, action, command, co-ordinates, debug/debugging, decomposition, event, input, object, output, attributes, repeat, repeat until, selection, string, timer, variable, launch command, x and y attributes**  **Lesson 3- simulation, object, variable, decomposition, abstraction**  **Lesson 4- command, attributes, run/execute, object, function, text object, tabs**  **Lesson 5- program, algorithm, variable, timer, loop, debug**  **Lesson 6-** **program, algorithm, variable, timer, loop, debug** | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **National Curriculum** | **Week** | **NC Coverage** | **Skills taught** | **Knowledge** | **Activity Outline** |
| •design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts  • use sequence, selection, and repetition in programs; work with variables and various forms of input and output  • use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs  •understand computer networks including the internet; how they can provide multiple services, such as the world wide web; and the opportunities they offer for communication and collaboration  • use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content.  •Select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information  • use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact. | 1 & 2  Unit 6.1 Lessons 1&2 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts  use sequence, selection, and repetition in programs; work with variables and various forms of input and output  use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs | 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. | Children can plan a program which includes a timer and a score.  • Children can follow their plans to create a program.  • Children can debug when things do not run as expected. | *Preparation:*  *Set Free Code Gorilla as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:*  *• 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*](https://static.purplemash.com/mashcontent/applications/schemes_of_work/computing_schemes_of_work/computing_sow_year6_unit_6_1_2021/Unit%206.1%20Coding.pdf)  Lesson 1:  Slide 4: Display slide 4. 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  button to check the answers. Click ‘OK’ to see which are correct and incorrect:  Run through the answers to the questions together. Use slide 5 which has 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.  Slides 5 and 6: Discuss new vocabulary with the children on vocab slide.  Go through PM slides.  We do: Display ‘Splatty bug’ and complete it together as a class.  Slide 8 to remind them 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.  Then add the code:    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.  Slide 9:  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.    This section of code will now set the playable time of the game to 10 seconds, alert the player at the end and restart the game. Review selection – IF/ ELSE statements and discuss how they work.  Slide 10:  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 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.  Stage 4:  Add code to show the splat and set its X and Y values to equal those of ‘Clicked bug’.    Review use of co-ordinates and how they help position objects in a scene.  Stage 5: Read Slide 5 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.  Free code gorilla: Display slide 13. 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.  Slide 14: 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. Use the slide to go through the things they could include. Give children time to think about what they want their games to do and plan in detail before they start coding. **At the end of this lesson, let children swap their plans and review each other’s.**  Lesson 2:  Make Your Computer Game: Display slide 15. Ask children to get out their program plans from last lesson and complete them if they haven’t already.  Children to open Free Code Gorilla from their 2Dos to create programs using their plans – their games should feature at least a timer and a score.  Adaptions: Children can be given copies of the Making a Timer and Score Pad guide to help them implement their designs if needed.  Ask children to save their program, then use the display board (see Appendix 1) 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. What challenges did they come across?  **Share positive examples on the board. (Good time to add comment to children’s work for marking)**  Plenary:  Review meaning of vocabulary (click on words to reveal the definitions) |
| 3  Unit 5.1 Lessons 3 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts | 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 use decomposition to make a plan of a real-life situation. | • Children can make good attempts to break down their task into smaller achievable steps.  • Children recognise the need to start coding at a basic level of abstraction to remove superfluous details from their program that do not contribute to the aim of the task. | *Preparation:*  *Set Decomposition and Abstraction writing frame as a 2Do for your class OR*  *• Print out the Decomposition and Abstraction writing frame, one for each child or pair. You can select the following computing objectives when setting the 2Dos to make future assessment easier:*    Lesson outline:  Discuss new vocabulary with the children on vocab slide.  Go through PM slides.  Activity 1: Children to plan a process (meal, board game etc) use the writing frame: [Decomposition and Abstraction.](https://www.purplemash.com/app/pup/coding_decomposition_abstraction)  **Share positive examples on the board. (Good time to add comment to children’s work for marking)**  Plenary:  Review meaning of vocabulary (click on words to reveal the definitions) |
| 4  Unit 6.1 Lessons 3 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts  use sequence, selection, and repetition in programs; work with variables and various forms of input and output.  use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs. | To use functions and understand why they are useful.  To understand how functions are created and called. | • Children can create a program that makes use of functions.  • Children can create a program that uses multiple functions with the code arranged in tabs.  • Children can explain how their code executes when their program is run. | *Preparation:*  *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.*  Lesson outline:  Go through PM slides.  Activity 1: . Ask children to log into Purple Mash and have a go at the Functions guided lesson that you’ve set as a 2Do.  Review children’s progress and discuss how they’ve got on using functions.  Slide 7: Use this 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 Free Code Gorilla by clicking on the ‘Preview’ button in the 2Do. Click on ‘Design’, then add a turtle object, a shape object, and a text object.  Slide 8: 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 (some of the attributes you could change are show below).  Your scene may look something like the this:    Save the work and click on ‘see code’.  Slide 9: 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.    Slide 10: 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 shown on the 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  Activity 2: Ask children to open Free Code Gorilla from their 2Dos area and have a go at creating a turtle program using functions.  1. What other shapes can they create functions for? Show them the images to illustrate the type of program you are expecting them to create.  2. Tell children to arrange their code into tabs so it is easier to debug – remind them about decomposition  **Share positive examples on the board of Game plans. (Good time to add comment to children’s work for marking)**  Plenary: Recap vocabulary definitions |
|  | 5  Unit 6.1 Lessons 4 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts  use sequence, selection, and repetition in programs; work with variables and various forms of input and output.  use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs. | To use flowcharts to test and debug a program.  To create a simulation of a room in which devices can be controlled. | Children can follow flowcharts to create and debug code.  Children can create flowcharts for procedures.  Children can be creative with the way they code to generate novel visual effects | *Preparation:*  *Set Billy’s Bedroom Simulation as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:*    *Print/ copy Billy’s Bedroom Flowchart enough for one each for the children.*  Lesson outline:  Go through PM slides. Introduce vocabulary.  Billy’s bedroom flowcharts: Slide 6: Open the Billy’s Bedroom Flowchart – note that there are several flowcharts on one page here, and each flowchart represents a procedure for something that happens in Billy’s bedroom.  Give each child a printed copy of Billy’s Bedroom Flowchart. 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?  Billy’s bedroom simulation: Open Billy’s Bedroom Simulation in 2Code and share with the children.  Does it contain all the same things as your picture? Probably, but quite likely in different places.  Click on the Run button to run the code. Watch it for a while and notice it switching between day and night.  Ask children: What are the differences?  What might happen at night that doesn’t happen during the day? How long is the day/ night?  Give children a couple of minutes in talking partners to look at their printed flowcharts and predict what will happen when different objects are clicked on.  Slide 8: Run the simulation. 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/ debugging.  Activity 1: Set the children the task of interpreting the flowcharts, debugging existing code and getting the rest of the devices in Billy’s room working.  - There are no flowcharts for some objects (car, robot, computer) they can plan the procedures for these in flowcharts on paper or in 2Chart in another tab.  - 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. |
|  | 6 & 7  Unit 6.1 Lessons 6 | design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts.  use sequence, selection, and repetition in programs, work with variables and various forms of input and output.  use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs. | To understand how 2Code can be used to make a text-based adventure game. | Children can follow through the code of how a text adventure can be programmed in 2Code.  Children can design their own text-based adventure game based on one they have played.  Children can adapt an existing text adventure, so it reflects their own ideas. | *Preparation:*  *Set Y6 Text Adventure as a 2Do. You can select the following objectives when setting the 2Dos to make future assessment easier:*  *• Print/ copy Text Adventure Functions to hand out as a supporting document.*  *• Create a display board for the class to share their programs to. Details of how to do this are given in Appendix 1*  Lesson outline:  This lesson will last for 2 weeks.  Go through PM slides, introduce vocab etc.  Use slide 5 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.  Activity 1 (10 mins): Ask children to open the Y6 Text Adventure from their 2Dos and press Run to get started.  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.  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.  Slide 7: 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. If children’s maps are varied, let them refer to the image on the slide.  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.  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.  Remind children of the undo button which appears when you have changed something so that you can undo; often this will fix the problem.  Slide 8: In the code, each room has been given a number. Players start in room 0. Look through the code together and try to write the numbers on the map.  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.  Variables: 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 to children:  • 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: Use the slide to explore the next section of code – Functions.  Articulate the following points to children:  • 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. • 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.  \*In the resources, you’ll find Text Adventure Functions Code. Some children might find it helpful to have printed versions that they can refer to when changing code.  Repeat until: 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 ‘roomls0’. • 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  Display slide 12 and explore the code that runs when the ‘finished’ variable =1.  Can the children explain what this does?  Lesson 7:  Activity 2: slide 13. 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.  Display slide 14. 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.  Plenary: Recap vocabulary definitions |
|  |  |  |  |  |  |

