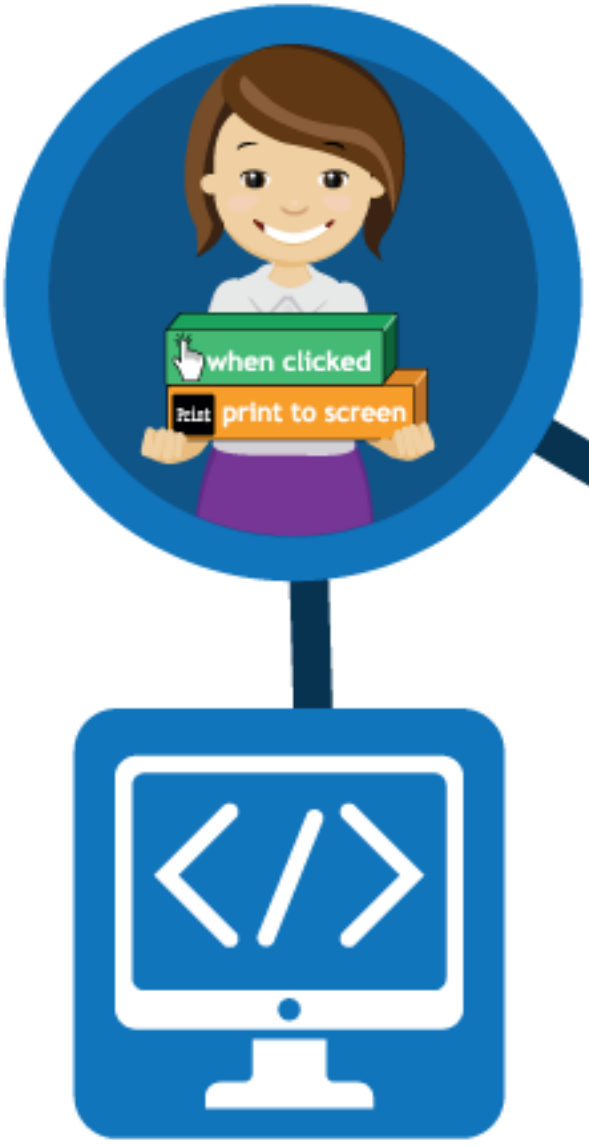
purple mash

Computing

Scheme of Work

Unit 5.1 - Coding



From 2simple

Contents

Introduction

This unit consists of six lessons that assume children have followed the Coding Scheme of Work in Years 1 to 4. 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	• Read and understand code • Remix code to achieve a particular outcome. • Debugging. • Use printed code snippets so that children can't run the code but must read it. • Include unplugged activities and 'explaining' tasks e.g. 'how do variables work?'
	Shared coding	• Sharing Challenge activities as a class or group on the whiteboard. • Complete guided activity challenges as a class. • After completing challenges; share methods to create a class version of the challenge. • Free coding as a class
	Guided exploration	• Exploring a limited repertoire of commands • Remixing code • Explore commands in free code before being taught what they do. • Use questioning to support children's learning. • PRIMM approach; Predict – Run – Investigate – Modify - Make
	Project design and code	**Projects (imitate, innovate, invent, remix)** There are different ways to scaffold learning in projects. This process can be applied to programming projects; • Using example projects e.g. the Guided 2Code activities. • Completing the challenges at the end of each guided activity. • Free code✓ • Create a project that imitates a high-quality exemplar. • Remixing ideas. • Independently creating a brand-new program.
	Tinkering	Use Free code Gorilla to access the full suite of 2Code objects and commands ✓ Use Free code to play and explore freely.

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

In Literacy, some teachers follow a progression that scaffolds learning to write texts. At first pupils read lots of examples of the genre of text they are going to create. Then they create an ***imitation*** of an example text. Next, they create a variation of the text (***remix and innovate***). Finally, they get to ***inventing*** a brand-new version.

Note: To force links within this document to open in a new tab, right-click on the link then select 'Open link in new tab'.

Year 5 – Medium Term Plan

Lesson	Aims	Success Criteria
Lesson 1 – Review	• To review coding vocabulary. • To use a sketch or storyboard to represent a program design and algorithm. • To use the design to create a program.	• Children can use sketching to design a program and reflect upon their design. • Children can create code that conforms to their design.
Lesson 2 - Simulating a Physical System	• To design and write a program that simulates a physical system.	• Children can explain how their program simulates a physical system. • Children can select the relevant features of a situation to incorporate into their simulation by using decomposition and abstraction. • Children can reflect upon the effectiveness of their simulation.
Lesson 3 - Text Variables	• To review the use of number variables in 2Code. • To explore text variables.	• Children can explain what a variable is in programming. • Children can set/change the variable values appropriately. • Children know some ways that text variables can be used in coding.
Lessons 4 and 5 - A Competitive Game	• To create a playable, competitive game. • To combine the use of variables, If/else statements and Repeats to achieve the desired effect in code. • To read code so that it can be adapted, personalised and improved.	• Children can create a game which has a timer and score pad. • Children can use variables to control the objects in the game. • Children can create loops using the timer and If/else statements.
Lesson 6 - The Launch Command	• To explore the launch command and use buttons within a program that launch other programs or open websites. • To create a program to inform others.	• Children can include buttons and objects that launch windows to websites and programs. • Children can code a program that informs others.

Lesson 1 – Review Prior Coding

Aims

- To review coding vocabulary.
- To use a sketch or storyboard to represent a program design and algorithm.
- To use the design to create a program.

Success criteria

- Children can use sketching to design a program and reflect upon their design.
- Children can create code that conforms to their design.

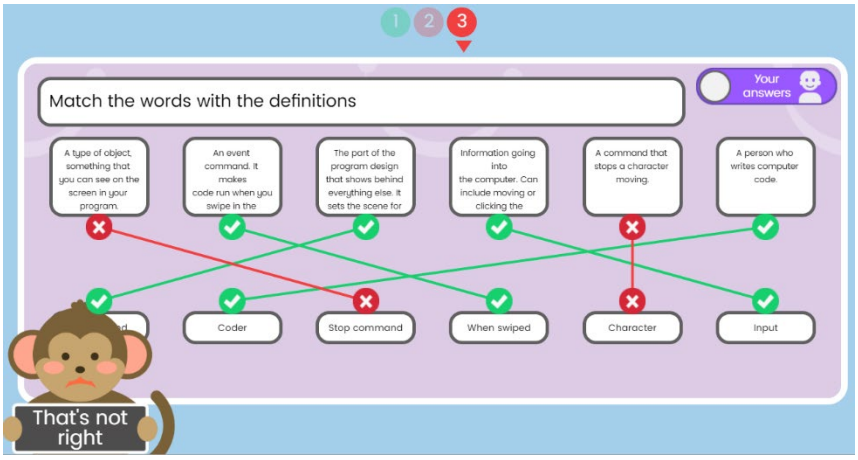
Resources

Unless otherwise stated, all resources can be found on the main unit 5.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you don't lose this page.

- Coding Vocabulary Quiz 4
- Program Design examples to be displayed on the board.
- Plain paper for sketching a design or printed storyboard templates from the Printable Storyboards on the main unit 5.2 page.
- Using TABS video. To show on the board.
- (Optional) Vocabulary flash cards. The Teacher flash cards have been created in such a way that you can print them on A4 paper, cut them to size, fold them in half and glue them together.
- (Optional) Exercise books to be used as 2Code workbooks for recording coding exercises, if desired.
- Lesson 1 storytelling example.
- Free code Gorilla – this is accessed from the main 2Code area (go to Tools and then 2Code, then scroll down to the Gorilla level).

Activities

1. Use the quiz as a class. 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:



2. You can use the vocabulary cards to find the answers and display in the classroom.
3. In this lesson the children will be learning/reviewing programming terms and creating their own programs that tell a story.
4. Put Free Code Gorilla on the board. Review how to add objects (characters) in 2Code by going in to Design Mode. Drag a car and an animal into the Design View.

5. Refresh the children's knowledge by asking them to create a short, simple program in which two characters perform actions that tell a story. Look at the Example program; can they tell which story it is from? Ask children to sketch a simple design for the program. Use the design examples document to remind children of the types of designs they have created in the past. They should design their program before beginning coding. They should detail:

   - What story they want to tell?
   - How many objects they will need and the types of objects.
   - How their objects will tell their story, i.e. what events will cause actions to occur?
   - What their objects will do to tell their story, i.e. what actions the object will perform or output.

6. 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 who 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.

   Children may want to look over their programs from the past to refresh their memory on some aspects if they have not used 2Code for a while. The Gibbon activities might provide further reminders.

7. Once children have had some time to code, save, test and debug their program, bring the class back together.

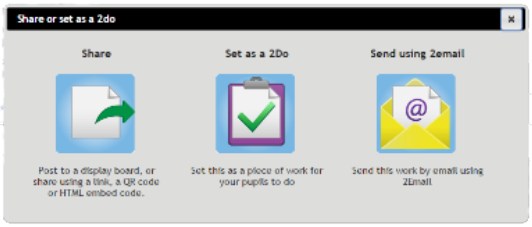
8. Introduce the children to Tabs in 2Code Gorilla. These are used to keep code organised. Now that the children are making more complex programs, it is important for them to organise their code so that they can understand what it does, and it also makes debugging easier. Show the Introduction to TABS video.

9. Ask the children to reorganise their code so that each character has its own tab.

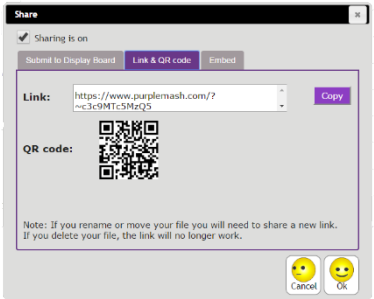
10. Show the children how they can create a quick response (QR) code for their program. They can print the QR code for their books or save the image to be scanned by a QR reader to open their program.

11. First, save the file in an online folder (not on the computer), then click on the Share icon at the top of the screen.

12. Select Share



13. Then Link and QR code.



14. 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 and printed.

Lesson 2 - Simulating a Physical System

Aim

- To design and write a program that simulates a physical system.

Success criteria

- Children can explain how their program simulates a physical system.
- Children can select the relevant features of a situation to incorporate into their simulation by using decomposition and abstraction.
- Children can reflect upon the effectiveness of their simulation.

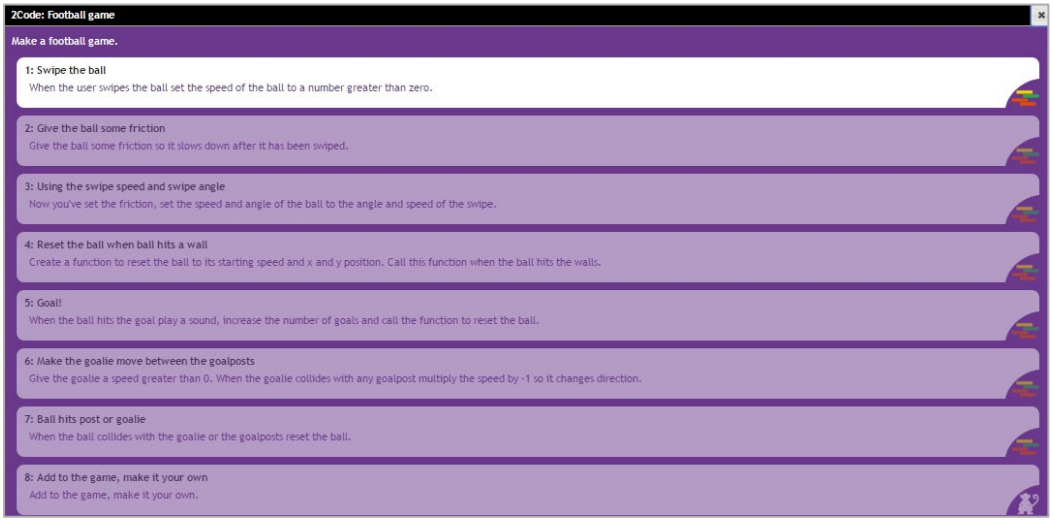
Resources

Unless otherwise stated, all resources can be found on the main unit 5.1 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.

- (Optional) print storyboard templates for program design from the Printable Storyboards section.
- (Optional) Exercise books to be used as 2Code workbooks for recording coding exercises, if desired.
- (Optional) Self-Assessment templates:
  - Pdf for printing
  - Writing template
- Create a coding blog for the class. There are instructions of how to do this in Appendix 1.

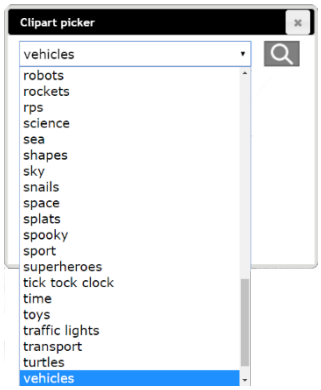
Activities

1. Today we will be creating a program that simulates a physical system. Ask the children if they know what this means? Take some suggestions.
2. It means creating a program where the **objects** behave as they would in the real world. For our program, simulating a physical system means using angles or speed to change the way an object moves, for example, the way someone kicks a football determines how far and how high the ball goes.
3. When you are simulating a physical system, you first must break the system down into parts that can be coded. The different parts will come together to make the full simulation. This is called **decomposition**.
4. Open the Gorilla guided activity Football from the main 2Code page. Do not start the activity yet. This game simulates kicking the ball and the movement of the ball. The stages show how this is achieved in



small steps, each one adding to the physical simulation, e.g. simulating kicking the ball, simulating the effect of friction, simulating the direction and speed of the ball, the effect of the ball hitting a wall, scoring a goal and the movement of the goalie.

5. You also need to remove aspects of the simulation that are not relevant. This is called **abstraction**. In this simulation, there are no spectators, the weather is not part of the simulation, the quality of the pitch is



not included. Some computer games have hyper-realistic simulations, can children suggest some games where the level of abstraction is very detailed down to expressions on faces and details of an in-game world that is not directly relevant to the game play?

6. Open Free Code Gorilla in a separate tab of your Internet browser and go into Design Mode.
7. Drag in a vehicle, double click on it and show children that they can change it into something else if they like by altering the image. Look at the choices of images using the drop-down box of categories. Hopefully, this will spark the children's imaginations about what their program can do. Ask for some ideas.
8. Return to the football activity and work through the first three stages of the football game to demonstrate how to set the speed and angle of a vehicle object.
9. Children should now plan and design their program briefly before getting to work on coding it. Plans should include:
    a. What physical system their program is simulating?
    b. The parts that they will need to code (**decomposition**).
    c. Things that are not relevant (**abstraction**).
    d. How many vehicles they will include in their program and what their images will be?
    e. What their vehicles will do? These are the steps of the **algorithm** – they should think about the steps for each part they have **decomposed**.
    f. What can they do to make their simulation even more realistic? – They should initially focus on a high level of **abstraction** and get the program working before adding additional details.
    g. Children should add an alert command as the first block in their code. In the alert they should enter some text that explains what the program is simulating and how.
10. Once children have coded, tested and debugged, they should share their work to the class blog. For details of how to do this, see Appendix 2. Children should include their own self-evaluation of what they think went well and what they could improve and any questions that they have about how to do this.
11. Once you approve the postings, children will be able to give comments on each other's efforts and make suggestions for how others can solve their issues.

Lesson 3 - Introducing Text Variables

Aims

- To review the use of number variables in 2Code.
- To explore text variables.

Success criteria

- Children can explain what a variable is in programming.
- Children can set/change the variable values appropriately.
- Children know some ways that text variables can be used in coding.

Resources

Unless otherwise stated, all resources can be found on the main unit 5.1 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.
- Y5 Lesson 3 Example

Activities

1. Today we will be working with variables. Can anyone remember what a variable is in coding?
2. Remind children of the definition of variables:

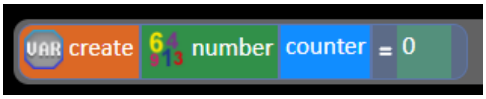
> **Variables** are like boxes in which the computer can store information. To find the information in the box, each box should be labelled. Therefore, each variable (each of our boxes) needs to have a **name**. The **name** should be **something that helps you remember what it is**. The information inside the box is called the **Variable Value**. The user, the program or another variable can change this Variable Value.

3. Examples can be found in the Guided lessons; the on/off state of a switch (see SWITCHING BACKGROUND lesson), to count the number of swipes before changing a lamp into a genie (see GENIE lesson) or the numbers changing in a timer (see NIGHT and DAY Gibbon lesson). (Guided lessons are all on the main 2Code page).
4. In 2Code, variables can either be numbers or strings (words, phrases or even whole sentences) or functions.
5. Can children recall using a number variable to create a timer last year?
6. To review what they did, open Free Code Gorilla on the board and look at the orange variable buttons in the left-hand menu.
7. Drag a 

block into the code window. The drop-down menu gives three options; 

. For today, we are going to be looking at number and string variables.
8. Choose 'Number'. Give the variable the name 'counter'. Ask children why we are naming the variable?

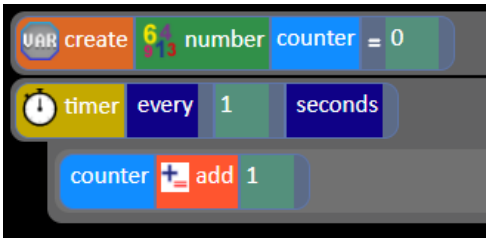
9. We can either define the variable as a specific number or set it to Random. For this example, we will set the variable to 0 as the counter will start from 0.



10. Drag in a timer and change 'After' to 'Every'

11. Drag 

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



12. Add a print to screen command so that the **value** of counter gets printed. Run the code to demonstrate the counter.

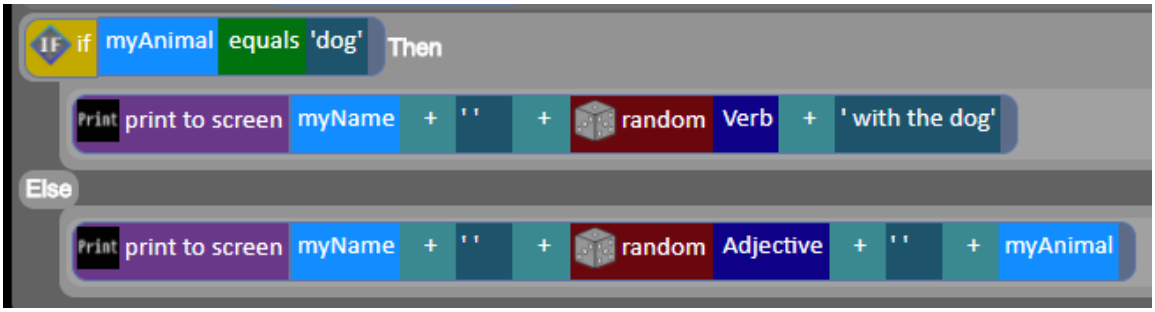
13. Try **initializing** the counter value to 30 and subtracting 1 from the value of counter each second. Can children predict what the result will be? How could this code be used in a game? (e.g. as a countdown).

14. In this lesson, we are going to create text variables. Display the code snippet example. But before running the code, can the children suggest what the code will do?

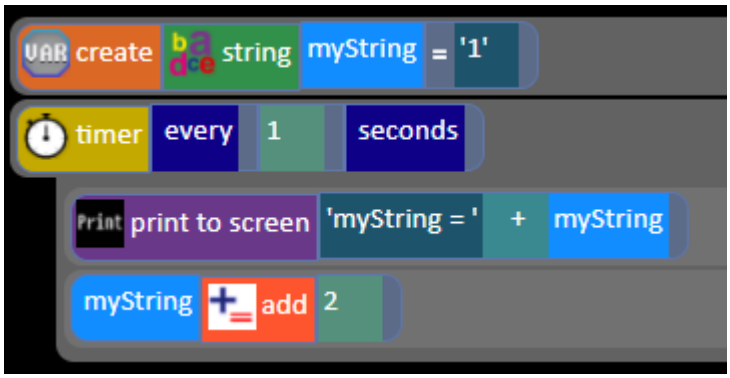
15. Things to note with the children:

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

    The solution is:



17. Try setting a string variable to '1' (the quote marks are required) and + 2 to it (see snippet below); can children guess what this will make?



18. How would you code this so it adds 1+2?

    Solution:

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

20. Last year the children did some of the guided Coding principle activities that also explore this area, they could complete challenges that they have not yet done.

Lessons 4 and 5 - Creating a Game with Score and Timer

This lesson is designed to take more than one session so that children have enough time to make a basic game and then improve and build upon it.

Aims

- To create a playable, competitive game.
- To combine the use of variables, If/else statements and Repeats to achieve the desired effect in code.
- To read code so that it can be adapted, personalised and improved.

Success criteria

- Children can create a game which has a timer and score pad.
- Children can use variables to control the objects in the game.
- Children can create loops using the timer and If/else statements.

Resources

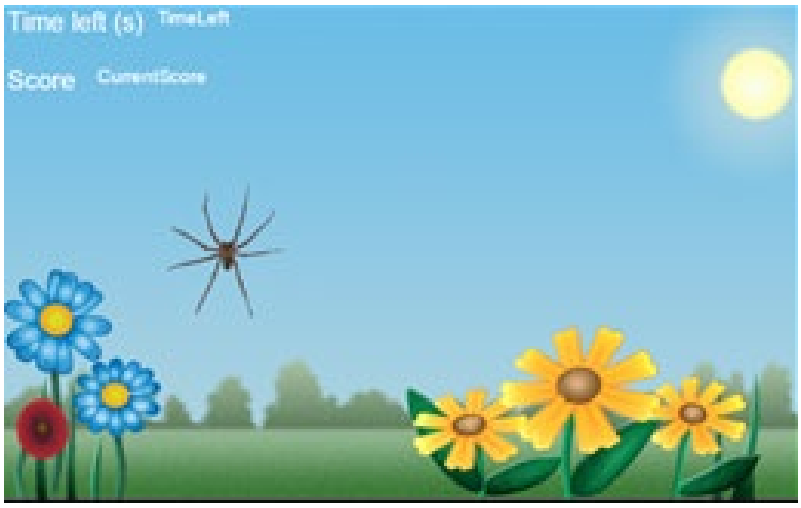
Unless otherwise stated, all resources can be found on the main unit 5.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you don't lose this page.

- Making a timer and scoreboard guide. Children will need to refer to this, so it can be set as a 2Do for the class or printed for the children.
- Step-by-step instructions for pupils: This is a step by step guide for how to complete step 1 which can be found in the download folder. Children will have had a lot of practice doing these steps if they have completed the previous 2Code lessons but some might need the extra guidance especially if they have not.
- Example Spider Game: Use this premade game during the first part of the lesson to unpick the code required to make a game.

Activities

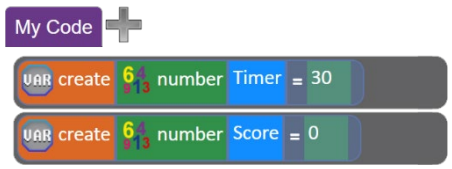
1. In this lesson, we will be aiming to create a fun game that uses a timer and scoring to improve the challenge. The example game is called Spider Catcher. The player must catch as many spiders as possible in the available time, and one point is awarded for each spider caught (clicked on). The spiders disappear and appear at random.
2. Work through the main stages of this example game as a class first, then you will get the opportunity to adapt the program to make your own game.
3. The stages of this example game are:
4. First, in free code Gorilla, add a background and spider character (more spiders will be added later), timer and score pad objects and labels.

   Here is an example background and character. Name the number objects appropriately as it will make



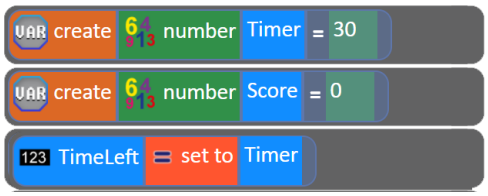
the coding easier, e.g. TimeLeft and CurrentScore. The titles for TimeLeft and CurrentScore are text objects; the actual score and time are number objects.

5. Next, create variables to store the score and time left. Should these be text or numbers? Can children

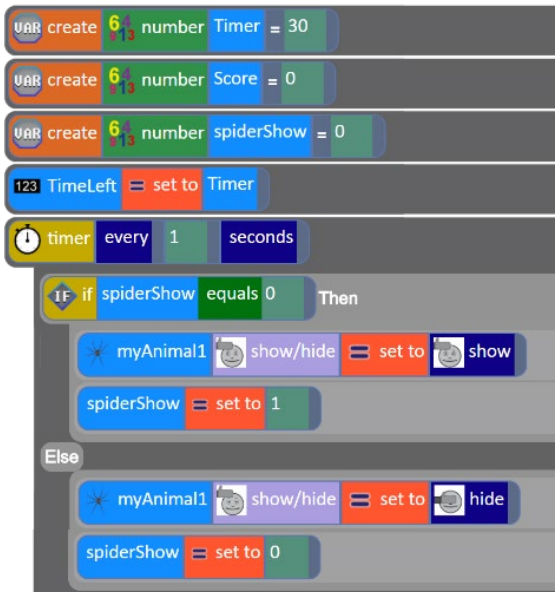


decide what events will cause the variable to change and how they will change? The time for this game is going to be 30 seconds so, as it will count down, set the timer variable to 30. The score should start at 0.

6. Then, to display it on the screen, set the TimeLeft number on the screen to the Timer variable.

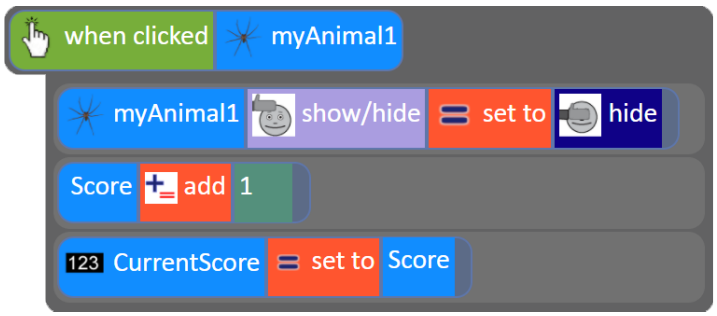


7. After that, the code needs to be added to make the spider appear or disappear each second. This is done in the same way as the Day and Night program that they looked at previously (Y4 lesson 2). A number variable should be created (spiderShow) and is set to 1 if the spider should be shown and 0 if it should hide. A timer is used to repeat every second and change this variable between Show and Hide.



8. Save and play the program; the spider should appear and disappear each second.

9. Now add the code to change the timer variable each second and display the time left in the TimeLeft number object. Also, add code that tells the player the final score when the timer reaches 0.

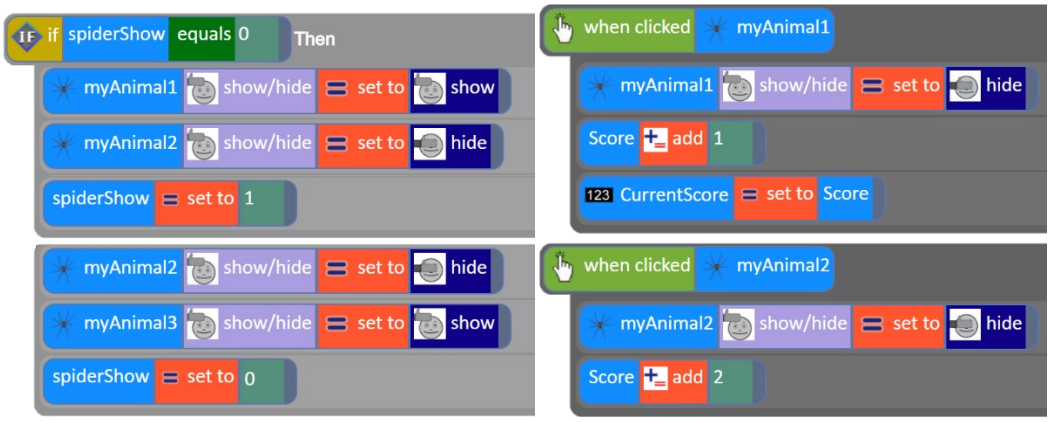


10. To score a point, the player needs to click on the spider. Write some code to make the spider hide and the score increase by one when the player does this:

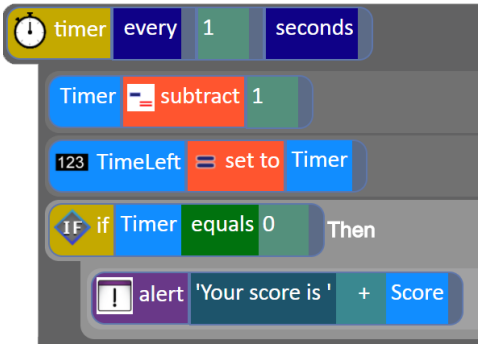
11. Finally, save and test the game. It should work now … but it's not that exciting, is it?

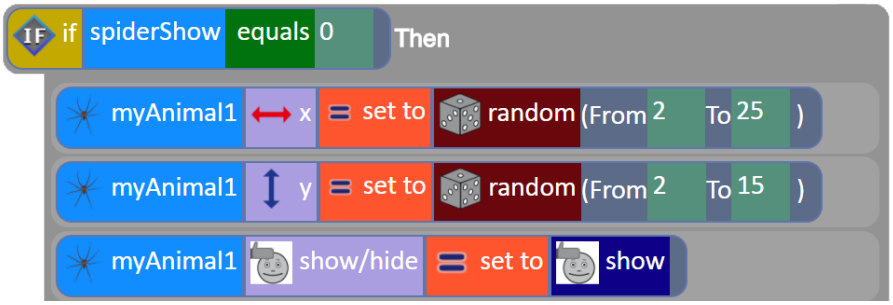
12. Can you improve it? Here are some suggestions:

13. Add more spiders; try making them smaller (use the scale in Design View). You will need to add code for the new spider to hide/show and to score a point when it is clicked on. Smaller spiders are harder to click; maybe they should get two points for a small spider?



14. The spiders appear and disappear in the same place; can you alter the code so that the X and Y positions make the spider appear somewhere random? Here's a hint:





15. Remember this is YOUR game – if you don't want spiders, you could use different characters. Make it fun!

NB This game contains a bug in that once the user clicks on OK at the end of the game, the timer and scoring continue. This has not been fixed in this lesson for the sake of simplicity, but higher ability children might notice it. Challenge them to find a solution!

Lesson 6 - The Launch Command

Aims

- To explore the launch command and use buttons within a program that launch other programs or open websites.
- To create a program to inform others.

Success criteria

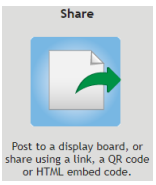
- Children can include buttons and objects that launch windows to websites and programs.
- Children can code a program that informs others.

Resources

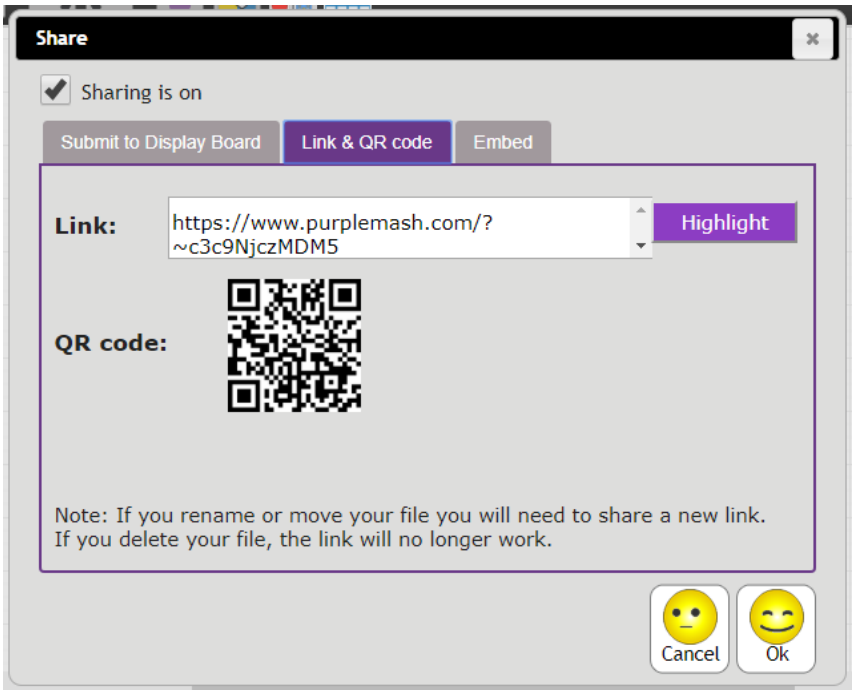
Unless otherwise stated, all resources can be found on the main unit 5.1 page. From here, click on the icon to set a resource as a 2do for your class. Use the links below to preview the resources; right-click on the link and 'open in new tab' so you don't lose this page.

- Any work that children have created about Internet Safety when completing the Purple Mash Online Safety units in previous years. This should be in their work folders.
- Access to the Purple Mash Online Safety resources.
- Examples of suggested Online safety websites
- Video Adding buttons to open webpages.

Activities

1. Today we will be using our knowledge of coding to create a program that launches activities about staying safe online.
2. Watch the video Adding buttons to open webpages. The video uses pictures of animals as the buttons, but button objects can be used and labelled. Use Free Code Gorilla to show children the difference and how they can use the launch command after the When Clicked event.
3. **Linking to a website:** Demonstrate how children can have internet tabs open with the relevant websites to copy and paste web addresses into the 2Code launch window.
4. **Linking to a Purple Mash Activity:** On a different tab, go to the Purple Mash homepage, click on 'Computing' and then open any of the online safety activities – make a note of what it is about.
5. In 2Code drag in a button and label it appropriately to link to the Purple Mash activity. Then demonstrate how to write the code so this activity will launch when the button is clicked.
6. **Linking to a 2Code activity:** Demonstrate how to use the Launch button to link to another 2Code activity – some children might be able to make a few 2Code programs each on different aspects of online safety and then have one main program with buttons or other objects that lauch the other programs.
7. **Linking to some work on Purple Mash:** Demonstrate opening a piece of completed work from your my work folder (maybe a written piece about online safety or an online safety game) in Purple Mash. Click on the globe button to share the work  . Click on 

and then on the Link or QR code tab:



8. Copy the Link from this window (highlight, right-click and select copy or use the best method for your device) and paste it into the launch command.
9. Children design and plan what the program should do, how it will explain what online safety is and why it is important to stay safe on the Internet. One example would be two characters, moving on command, with explanatory text that prints to the screen.
10. Pupils should include at least two buttons that launch two separate websites providing more information on why people should stay safe on the Internet. Pupils should think about how they can use their coding knowledge to create this program.
11. Programs could be shared to a blog or displayboard to inform others.

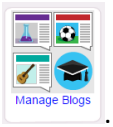
Appendix 1: Creating a class blog

The following appendix will show you how to create a blog using 2Blog for lesson 2. A further User Guide is available in the Teacher Guides and Planning area.

You can turn off the approvals process for a class blog should you wish to improve the flow of a blogging lesson but it is highly advised to turn the approvals process back on afterwards.

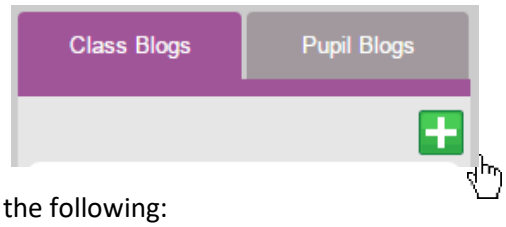
1. Open 2Blog in the Tools area 

or click on 

then 

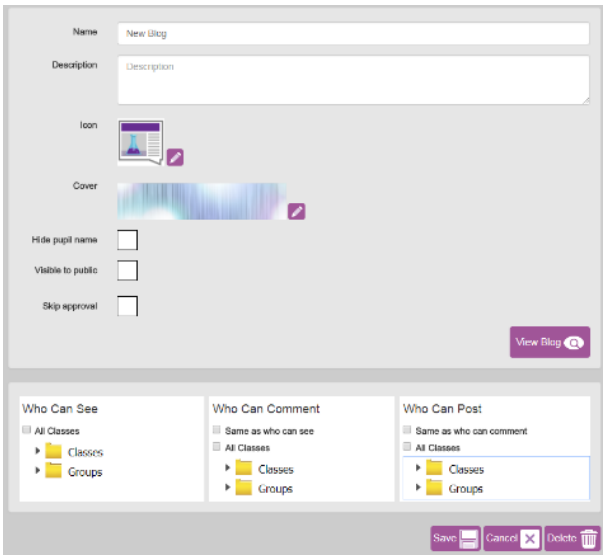
.

2. Create a blog by clicking on the Add Blog button in the My Blogs sidebar. Pupils will only see the My Blogs option in this sidebar. Teachers see two tabs: Class Blogs and Pupil Blogs.



3. On the next screen, fill in the following:

   - Blog name (e.g. Year 5 Coding examples)
   - Description for the blog
   - Select an icon
   - Select a background cover.
   - You could tick skip approval, **but it is strongly recommended that this is ticked again once the lesson has finished as children will be able to post and comment live with no need for your approval.**
   - In the bottom of the screen, select your own class as the users who can see the blog and then tick 'same as who can see/comment' for the other options.



4. Once you have selected options and clicked 'Save', you can view the blog using the 'View Blog' button. The blog title will also appear in the My Blogs sidebar.

5. Click on View Blog. This will show you the blog page. To add a blog post, click the Add Post button.

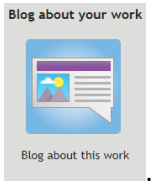
Appendix 2: Commenting and Posting to a blog

Commenting on a blog post

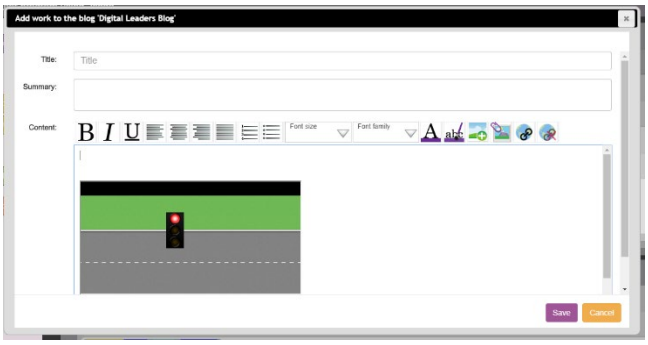
1. Children access the blogs by clicking on the Sharing tab.

2. Then click on Shared Blogs. Find the coding blog that you have prepared and click on it.
3. Click on any blog post to view it. Children should be able to click on each other's programs in the blog posts and play the code.
4. They can add comments below the post. If approvals are set to be on, you will get notifications in Purple Mash that there are comments to approve and pupils will not be able to see the comments until you have approved them.

Adding work to a blog

1. First save the work in Purple Mash and then click on the  button.
2. Click on 

.
3. Find the coding blog that you have prepared and click on it. This will create a blog post with the file embedded in it.
4. To add text before the file, click on the post content box and use the arrow keys to move to the beginning of the post and press enter a couple of times.



5. Add some text to the content.
6. Write in the title and summary of the post then click save.
7. If approvals are set to be on, you will get notifications in Purple Mash that there are posts to approve and pupils will not be able to see the post until you have approved it.

Assessment Guidance

The unit overview for year 5 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	With support, children can turn a real-life situation or story into an algorithm for a program that has cause and effect (lessons 1 & 2) and use their algorithm to write simple programs using 2Code. Pupils experiment with repeat and selection structures in their designs but may need support to write a logical algorithm to tackle a more complex coding task objective. As their coding becomes more complex, they will require support to tackle debugging in a logical rather than a trial-and-error method. They will usually require support to make use of variables and manipulate variables in their code (lesson 3 & lesson 4/5). Children's designs for their programs, show that they are thinking of the structure of a simple program in logical, achievable steps (lessons 1 & 2). Children can make good attempts to 'read' code and predict what will happen in a program (lessons 2 & 3) which can help them to correct errors. Pupils will make good attempts to explain how programs simulate physical systems and can create their own program to meet a design brief relating to a physical system (Lesson 2. Point 2).
Expected	Children can turn a specified real-life situation into an algorithm for a program by deconstructing it into manageable parts (lesson 2). Children's designs for their programs, show that they are thinking of the structure of a simple program in logical, achievable steps with attention to specific events that initiate specific actions (lessons 1 & 2). Children's design shows that they are thinking of the required task and how to accomplish this in code utilizing the coding structures for selection and repetition that they have learnt about (lesson 1, lesson 2) and displays an understanding of the function of variables in coding (lesson 3). 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 (lessons 1 & lesson 2). Their coding displays an understanding of the function of variables in coding (lesson 3) and outputs such as print to screen and sounds (lessons 3 & 4/5). 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. They try to improve and debug their own programs (lessons 5 &6). Children can 'read' others' code and predict what will happen in a program which helps them to correct errors (lessons 2 and 3). They can also make good attempts to fix their own bugs as their coding becomes more complex (lesson 4/5 & 6). When they code, children are beginning to think about their code structure in terms of the ability to debug and interpret the code later, e.g. the use of tabs to organise code and the naming of variables.

Assessment Guidance	
	Children can explain how programs simulate physical systems and can successfully create their own program to meet a design brief relating to a physical system (Lesson 2. Point 2). Throughout this unit, children will demonstrate that they are open to feedback from both the teacher and fellow peers on their programs, specifically during lessons 4 and 5 where they are expected to create and improve a game.
Exceeding	Children are attempting to turn increasingly complex real-life situations into algorithms for a program by deconstructing the situation into manageable parts (lessons 1 & 2). Children's design shows that they are thinking of the required task and how to accomplish this in code using coding structures for selection and repetition and variables (lessons 1,2 & 3). Their designs are ambitious, but the algorithms are logical and achievable. Pupils intuitively grasp the concepts of selection, repetition and variables. Pupils like to challenge themselves to combine these with other coding structures to achieve the effects that they design to personalise and to improve their programs (lessons 4, 5 & 6). They are also thinking about good structure to their code with a view to debugging such as the use of tabs to organise code and the naming of variables 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. Children can 'read' others' code and predict what will happen in a program which helps them to correct errors (lessons 2, 3 & 4/5).