Year 6 Programming 3 My Logo Web and Function Machine (Choice)

<table>
<tr><td><b>Timing</b>

6 sessions of approximately 45 minutes<b>.</b></td><td><b>Children will</b>

• Review knowledge of Logo and create shapes of different colours
• Meet challenges to recreate unusual shapes
• Make shapes grow and create a spider web
• Make numbers and function machines</td></tr>
<tr><td><b>e-safety links</b>

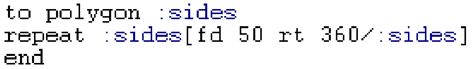
I can explain the consequences to myself and others of not communicating kindly and respectfully.</td><td><b>Objectives</b>

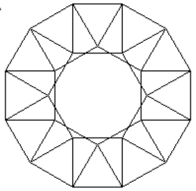
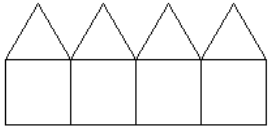
<b>Programming</b>
• I can deconstruct a problem into smaller steps and can see how these are similar to solutions I have used before.
• I can explain and program each of the steps in my algorithm.
• I can design an efficient program through the use of repeat procedures.
• I can use different inputs to control an onscreen action and predict what will happen.
• I can use a variable to achieve a required output including stopping a program.
• I can recognise errors in a program and link errors to a problem in the algorithm on which it is based.</td></tr>
<tr><td colspan="2"><b>Links to other learning</b>

<b>Mathematics:</b> using and applying maths, simple introduction to algebra and shape and space</td></tr>
<tr><td><b>Resources</b>

FMS free logo download
or LOGO in www.j2e.com/j2code</td><td>Preparation

Ask technician to install FMS free logo download or use Logo within www.j2e.com/j2code which can be accessed on android and iPads as well as laptops and PCs – even mobile phones!
Check YouTube access for https://www.youtube.com/watch?v=_X-xDmY1puA and https://www.youtube.com/watch?v=9qaaYlpVmJM</td></tr>
</table>

1	**Programming**
	I can deconstruct a problem into smaller steps, recognising similarities to solutions used before.
		Reviewing basic knowledge of Logo (For tablets use LOGO in www.j2e.com/j2code)
		• *Note for teacher: most children will have gained this from earlier learning experiences in previous years (Year 3 Term 3 Logo / Year 5 Term 2), if not begin with those experiences)*
		• This sequence of learning experiences is planned using FMS free logo download. Those working with tablet devices may choose to use i-Logo or Hopscotch but only limited access to some of the experiences will be possible.
		• Some children will be ready to move straight to session 8. However, the challenges in 2 are useful assessment activities to see the confidence of different learners.
		• **Children working on individual devices but encouraged to talk to those around about their tasks OR in pairs sharing a device.**
		• **Teach the computer to draw a shape**
		○ Remind children of creating a procedure (see Logo Basics support sheet on eLIM website)
		○ Can you teach the computer to draw a triangle?
		○ Can you teach the computer to draw a rectangle?
		○ Can you teach the computer to draw a circle? repeat 360[fd 2 rt 1]
		○ Can you teach the computer to draw any regular polgyon? 
		`to polygon :sides`
		`repeat :sides[fd 50 rt 360/:sides]`
		`end`
		• **Move onto changing pen colour**
		○ What does setpc 3 do? (May need to prompt them to make the turtle move to discover. If the computer tells them it doesn't know what to do they may have forgotten to put a space before number.)
		○ Try changing the number to any number between 1 and 16.
			Gold: I can use an efficient procedure to create regular polygons.
			Silver: I can use repeat commands to create regular polygons.
			Bronze: I can write a program to create regular polygons.

		• Can you make a prompt poster to help other people to know what to do?	
2	**Programming** I can deconstruct a problem into smaller steps and can see how these are similar to solutions I have used before. I can explain and program each of the steps in my algorithm.	**Logo challenges** • Children become proficient with the basic skills needed to perform operations on logo and begin to create more complex patterns. They will consider what the possibilities with logo could be. • Challenge the children to use what you have learnt to meet the challenge questions below. • Ask them to say the algorithm they will use first. Talk about the need to deconstruct the challenge into smaller steps that are similar to the programming they have used before. Talk about the need to continually test the program and to debug the program as they go along. • Can you draw a house? (Depending on the kind of triangle you make for the roof, you will need to rotate 30 or 90 – or perhaps anotherr amount depending on your programming - to orientate the roof of the house) • Can you draw a circle of houses? • This could also be a wheel. What could you create if you can create a wheel?  • Can you draw a row of houses? Can you draw a row of houses with spaces in between? Can you add windows, doors etc?  • Big challenge: Can you draw an outline that would be recognisable as buildings in your town / village? You will need to break the challenge down into manageable steps. Not all the buildings will be the same shape. What procedures will you need to create with the town procedure?	Gold: I can break a problem into smaller steps, recognising their similarity to things I have done before. I can program each step of my algorithm Silver: I can break down a problem into smaller steps and program each step. Bronze: I can create a shape, for example a house.

		• Can you make a windmill? Deconstruct the challenge into smaller steps – make a flag first. • Can you use the procedures for other shapes (such as those in 1.) to create patterns such as the wheel and the windmill? • Talk about whether they needed to change their algorithm while they were creating the image. Who had to debug their program? • What else can you create? Watch this video to see what is possible: https://www.youtube.com/watch?v=_X-xDmY1puA	
3	**Programming** I can deconstruct a problem into smaller steps and can see how these are similar to solutions I have used before. I can use a variable to achieve a required output *including stopping a program.*	**More Logo challenges** • Introduce the idea of some shapes being allowed to 'grow'. • Watch a video of spider building a web. https://www.youtube.com/watch?v=9qaaYlpVmJM or watch minute 1:00 – 2:00 of http://www.bbc.co.uk/education/clips/zbqd7ty • **Challenge: Can you create a spider web?** • Encourage the children to consider the shape they will need to create. What will the algorithm be? Can you break it down into smaller parts. • You might want to start children with creating a spiral or move them directly to creating a spider web. The support sheets from these links introduce the idea of needing to stop a procedure. They also introduce the possibility of using a procedure within its own procedure. This is the idea of recursion which they will meet in KS3. Here they only need to understand it as a way of making the same thing happen again and again but at different sizes. • Both the spiral and the spider web use either < or > to set the condition which means the program is finished. Needing to create a condition to stop a program is an important idea for programming. • Review the algorithms and the programming. Reinforce the need to keep trying	Gold: I can use a variable to tell a program when to stop. Silver: I can use a variable to change the size of a shape. Bronze: I can use a procedure to make shapes of different sizes.

		out the programming and that mistakes are an important part of discovering a solution. • **Extension:** This could be combined with music for some additional ***multimedia*** work. Can the children create a video similar to the one at the end of activity 1?	
4	**Programming** I can use different inputs to control an onscreen action and predict what will happen.	**Exploring possibilities with Logo** • Children use Logo to explore different possibilities with numbers. Give the children to following bits of program to try. Each time, review - with a mini plenary, before moving on. make "number 30 print :number • Try using these commands (will print out 30): • Now try: make "number :number + :number print :number This will double 30 which is the number that was set for the variable number. • What does the " in front of number do? What is the difference between this and :number? • What does this big number procedure do? Allow time for experimentation. to bignumber make "number 30 print :number repeat 10 [make "number :number + :number print :number] end • **Challenge task.** Edit the procedure to input any number. (You will need to use :number) • Use + - * and / to play with numbers. • Plenary: consider the application of this lesson to real world contexts.	Gold: I can edit procedures to input any numbers, making observations about what happens. Silver: I can experiment to see what happens when I try out different sets of numbers, and comment on the observations. Bronze: I can follow simple procedures to manipulate numbers on logo.

5	**Programming** I can link errors in a program to a problem in the algorithm on which it is based.	**Making function machines with Logo** • Children use Logo to explore the possibility of using logo to make function machines. • Tell the children that they will be using Logo to create instructions to do mathematical problems such as finding the area of a quadrilateral. First of all we will get the computer to talk to us. • Try using these commands: to greet print [What's your name?] make "name readword print [Hello] print :name end Variables: number1 number2 answer • What do the square brackets do? What does readword do? Why do you need : in front of name? Now we are going to get the computer to do a calculation. • What will these commands do? make "number :number1 * :number2 print :number • Think through what you do when you calculate the area of a regular quadrilateral. This will be your algorithm. It may look like the one below but there are many ways you could write the algorithm. The important thing is that it makes sense to you and it provides the steps you can use to make your program for the function machine. I also found it useful to list the variables that I would be using. • One algorithm could be: Look at a regular quadrilateral. Tell me the length of one of the equal sides. Tell me the length of the other equal side. The area is XX • You could add a person's name to the algorithm so that you can 'tell' the person	Gold: I can create a function machine to do a calculation. As I work I can link errors in the program to problems in the algorithm on which it is based. Silver: I can make a function machine for a simple calculation. Bronze: I can make a simple function machine to add two numbers.

		the answer. • Don’t forget to keep testing your program to check it is doing what you have described in the algorithm. If it doesn’t work as you want, is the error in the algorithm or is it in your programming? • You can see one way to create a function machine to calculate the area of a regular quadrilateral if you look at the support sheet. • **Challenge:** What other function machines can you create? • You may want to add an additional session if you would like to use the function machine idea to support development of mathematical thinking for different calculations.	
6	**Programming** I can use logical reasoning to detect and correct errors in a program.	**Recognising errors in a Logo program** • Children will have been debugging problems with their programming. Ask the class what has helped them sort out difficulties while they have been programming. Show them a bit of programming / coding which contains an error. This can be a syntax error (the code is written correctly) e.g. when telling the computer to say a name: to greet print [What's your name?] make "name readword print [Hello] print name ○ The colon is missing before name so the computer can’t retrieve name to print on the screen. • OR a logic error (the command chosen doesn’t do what is required, you have used an incorrect operator in a calculation, or you have used an input in a command which is not enough or is too much e.g. rt 60 in an equilateral triangle instead of rt 120) eg when calculating an area:	Gold: I can recognise errors in a program and correct them. Silver: I can recognise errors in a program. Bronze: I can recognise when a program does not do what I expect.

```
to input_sides
print [What is the length of one of the equal sides?]
make "number1 readword
print [What is the length of the other equal sides?]
make "number2 readword
make "answer :number1 + :number2
print :answer
end
```

- The two numbers are being added instead of multiplied.

- What are the checks we need to do when debugging a program?
- How do these compare to checks when debugging Scratch programs? What are the similarities and differences to programming with Scratch?