**Year 3 / 4 / 5 / 6 Possible Apps**

**Knowledge objectives (from CAS: A curriculum for schools)**

* Algorithms can be represented symbolically [flowcharts] or using instructions in a clearly defined language [turtle graphics].
* Algorithms can include selection (if) and repetition (loops).
* Algorithms should be stated without ambiguity and care and precision are necessary to avoid errors.
* Algorithms are developed according to a plan and then tested. Algorithms are corrected if they fail these tests.
* A computer program is a sequence of instructions written to perform a specified task with a computer.
* Programs can be created using visual tools.

Apps

Hopscotch Page 2

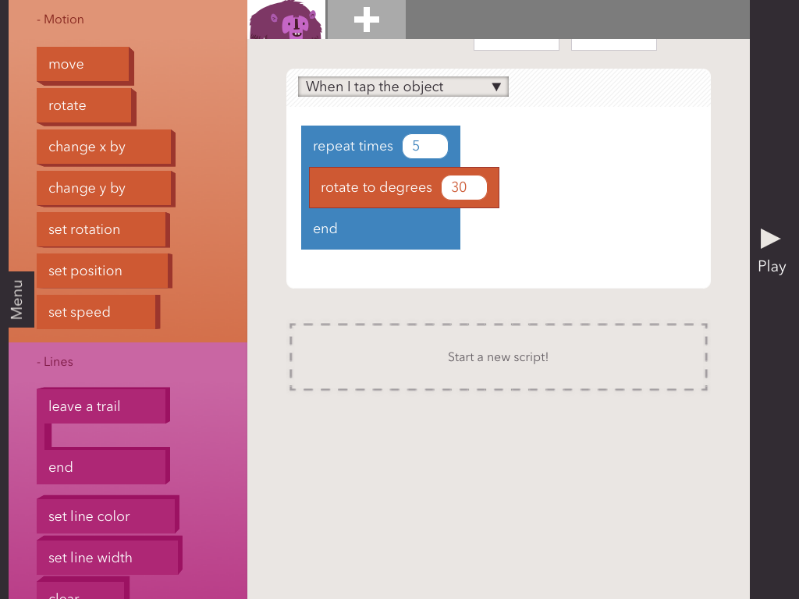
Logo Draw Page 5

Cargo Bot Page 10

**Hopscotch**

* Use a variety of inputs
* Use the ‘repeat’ (loop) command within a series of instructions.
* Use the ‘if… then’ (conditional statement) command within a series of instructions

Hopscotch is made by the same developers as Daisy the Dinosaur. It is a visual programming environment as algorithms and programs are constructed by dragging and dropping command blocks into the scripting area. In this way it is very similar to MIT’s Scratch, which pupils will come onto in Year 4. A screen shot of Hopscotch appears below.



Objects for which scripts are being programmed

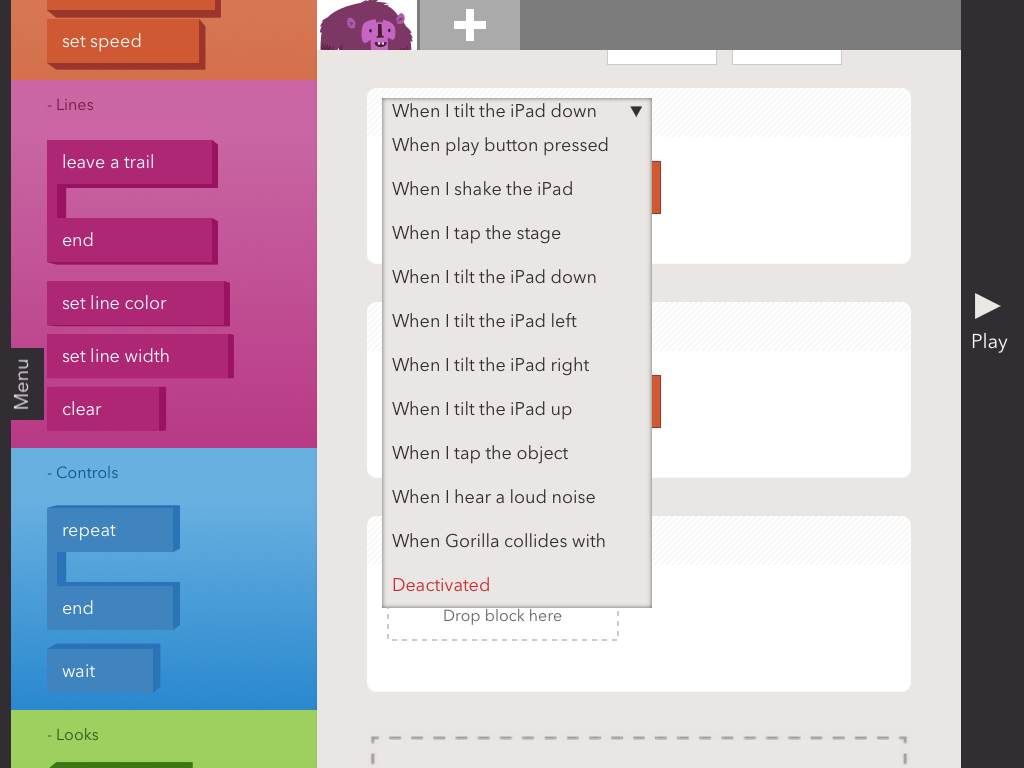
Scripting area

Commands available

Menu tab

In the screenshot above by clicking the ‘menu’ tab you will access options to create, save, share or browse programs made in Hopscotch. It can be seen that the commands available are colour coded (much like Scratch) and are grouped under: Motion; Lines; Controls; Looks; and Operators.

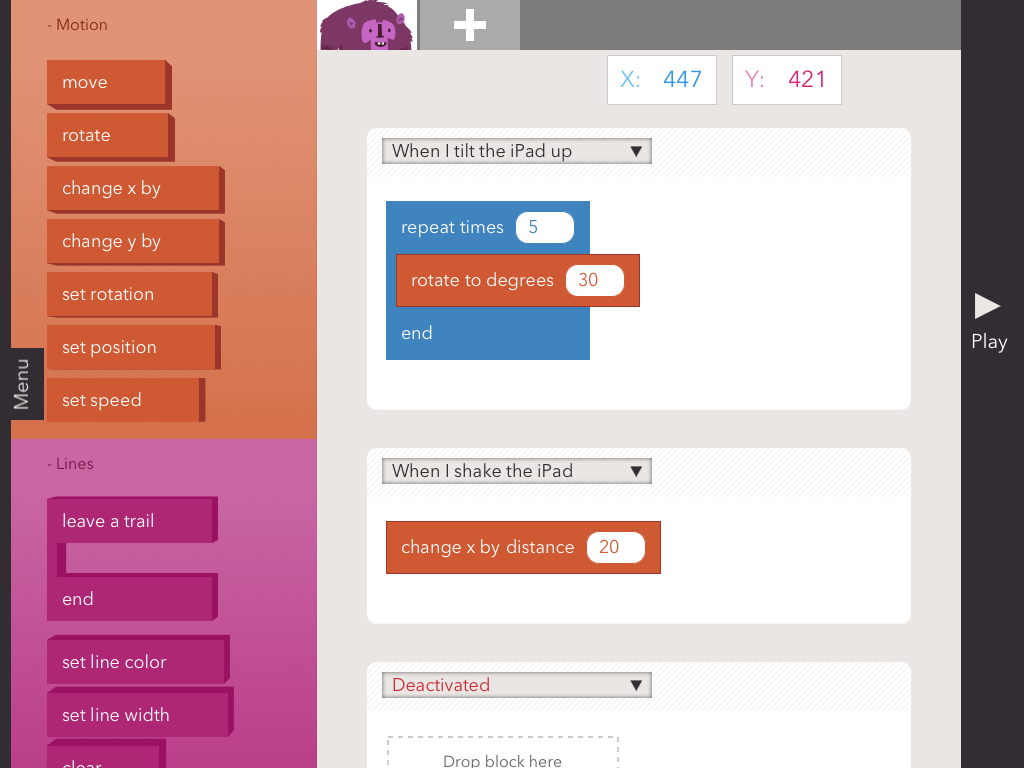
**Inputs within Hopscotch:** A focus of this unit is to use a variety of inputs. The screenshot below shows the range of inputs that Hopscotch recognises. Each of these inputs can be used to action a command within Hopscotch.



Inputs available

within Hopscotch

Once an input is selected, the commands that this input should action may be dragged into the scripting area beneath the input. For example, in the screenshot below, we can see that when the iPad is ‘tilted up’ the object (the Gorilla) rotates by 30 degrees 5 times (notice the use of a repeat command) and when the iPad is shaken, the Gorilla changes its X-Position by a value of 20 (Note by scrolling up the scripting area you can set the starting position of your object).



To add an object click here

In this unit pupils should be given the opportunity to explore the use of inputs within Hopscotch to control the program they code. They may try adding additional objects, achieved by touching the add sign at the top of the screen – highlighted above. Pupils should experiment using the loop function and may also wish to use the line commands, which cause the object to leave a path to draw out shapes.

Coding Challenge: **Can you program a simple game in hopscotch?**

Pupils should be challenged to code a simple game in Hopscotch. Two example include:

1. A game in which they have to move their character around and avoid other characters on the Hopscotch stage
2. A game which is an electronic recreation of Etch-A-Sketch

Both games above would require use of the various tilt inputs to alter the X and Y position of a character so you can move your character by tilting the iPad in various directions. In addition, for the first game pupils may add a second character and use the ‘When … touches…..’ input so a consequence occurs if you touch the other character - such as turning transparent using the opacity command! For the Etch-A-Sketch game pupils should use the line function for their character to leave a trail.

**Unit 2: Logo (Logo Draw app)**

* Write a simple program in Logo to produce a line drawing.
* Use more advanced Logo programming, including pen up, pen down etc.
* Write a program to reproduce a defined problem, e.g. geometric shape/pattern.

Logo is a word based programming language which was designed for educational purposes in 1967. This unit uses 2 different Logo programming apps both running on the iPads: Logo Draw & i-Logo. The purpose of using 2 programs is to gradually introduce pupils to writing the logo commands themselves, since the first app, Logo Draw, provides blocks from which pupils select a command. Let’s look at this app first:

**Logo Draw** 

When Logo draw is opened, you are asked to ‘Tap to Start’ and you will enter the app on the home screen as shown below.

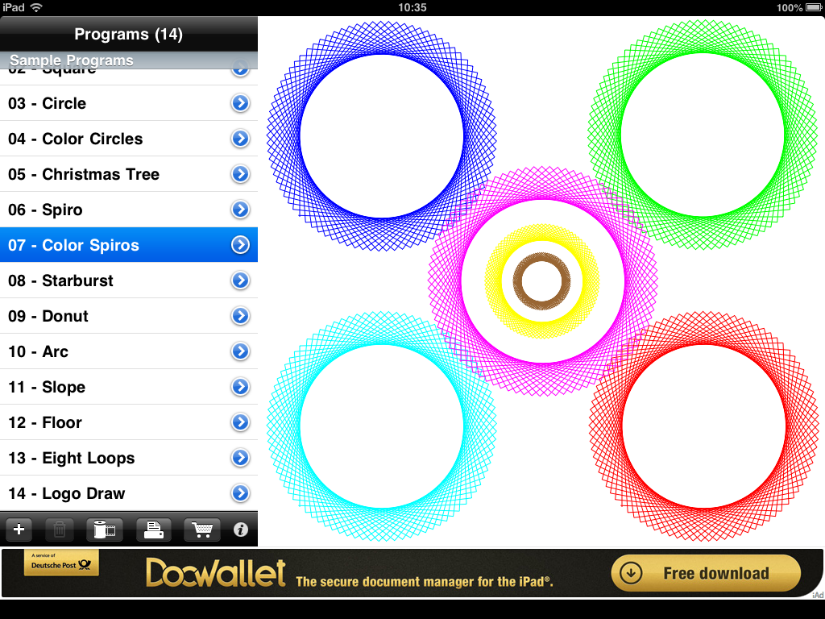


Turtle drawing area

To run a program click the blue arrow

Sample programs

The turtle and drawing area appears to the right of the screen with the programs that can be run on the left. As you can see, the program comes with a series of programs pre-loaded into the app, to run any of these programs simply touch the screen on the blue arrow next to the program name and the program commands will be displayed and the turtle will execute the commands. Below is the colour spiros program.



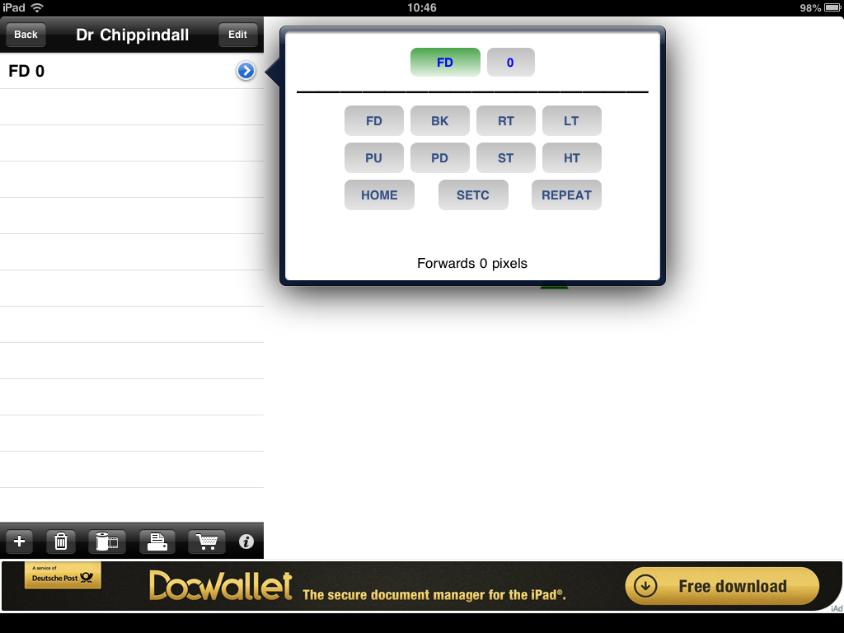
Click this + to add a new program

To add a new program, click the + button on the menu bar beneath the programs selection, indicated on the screen shot above. You will be asked to enter the name of your program and then the following screen will appear. In this screen we can construct a program which, when run, will be executed by the turtle. To start our program, click the + button again to add a command.



Click the + to add a command to your program

When the ‘+’ is clicked a new command will be added to the command list and, as a default, this will be a forward command (FD). The screen will look like the screen below:



Commands available to programmer

As you can see, a default command of forward (FD) has been added of the command list available. Since a key feature of Logo is knowing and using the command abbreviations (code) let us review all of the options that can be seen as being available to us:

**FD** – Forward. A value (amount forward) needs to be specified with this command which is currently 0 in the scree shot above.

**BK** – Backward. A value (amount backward) needs to be specified with this command.

**RT** – Right Turn. A value (degrees of turn) needs to be specified with this command.

**LT** – Left Turn. A value (degrees of turn) needs to be specified with this command.

***For both of the above turning commands, a lesson on right angle turns linking to degrees is required. Pupils could experiment by entering different degree values to explore how this changes the angle the turtle changes by.***

**PU** – Pen Up

**PD** – Pen Down

**ST** – Show Turtle

**HT** – Hide Turtle

Home – Returns turtle to the starting position

**SETC** – Sets the colour of the turtles line. A value needs to be given for this corresponding with the colour as shown below:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Colour | Blue | Green | Light Blue | Red | Pink | Yellow | White | Brown | Orange | Black |
| Value | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |

**Repeat ­**– Used to repeat any number of times the commands within square brackets (see separate screen shot below)

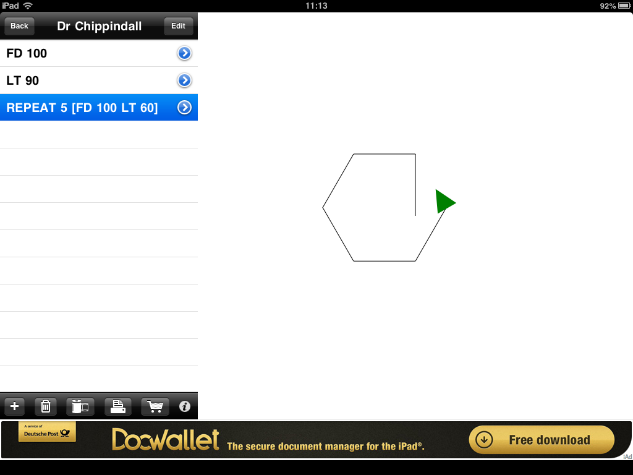


This number specifies the number of repeats

The commands with the [ ] brackets are repeated. The commands below the line are available to be repeated

**Building/editing programs**

A program is built up in the command sequence window by clicking ‘+’ as discussed above and choosing the command for the turtle. As the commands are added the turtle completes the command. To remove a command, simply click on it until it is highlighted blue and click on the bin icon, as shown below.



Click on bin icon to remove the commands

Click line of commands to remove

**Suggested sequence of learning:** Logo is a programming language and if pupils learn the language they can use any Logo programming environment, not just the two apps used here. As such, it is envisaged that this unit will start with an introduction to the language. This could be done, initially, entirely without ICT in a similar style to a MFL lesson perhaps? The commands (language) could be introduced and children could be the turtle decoding the commands given. This could be completed as a Kagan group activity in which a pair of pupils generate a sentence of code and the remaining members of the group interpret the code by moving themselves, or by tracking a pen over square paper.

The Logo Draw app could then be introduced and a lesson could focus on familiarising the pupils with the app – as has been done above. If a lesson has already been completed on introducing the language, then the command abbreviations that appear when the ‘+’ button is pressed will already be familiar to pupils. At this stage, pupils could work through an example with the teacher of writing the code for the turtle to trace a multi-coloured square – this provides scope for cross-curricular numeracy learning. It is suggested that a period of exploration is then given in which pupils can experiment with writing code in Logo and discuss the outcomes with partners and the class.

Pupils may then progress to use the app more independently to write code so that their turtle program draws a pre-defined shape – i.e. coding a solution to a defined problem. The teacher can define the complexity of the shape –differentiated to ability – or, again in Kagan groups, could be generated by the children. Using squared paper for pupils to generate shapes that they wish their partner to try and reproduce ensures that right angles are used (linking to age related numeracy objectives).

Coding Challenge: **Can you produce your own piece of artwork in Logo?**

****

The Coding Challenge in this unit could focus on pupils programming the turtle to achieved pre-defined outcomes using the Logo language they have become familiar with.

**Cargo bot app:**

* *use the repeat command within a series of instructions*

Given the power of the **repeat command** – or loops more generally - within code to refine command sequences, an additional part of this unit uses the app Cargo Bot to stretch pupils’ ability to think about the use of repeat loops. A screen shot of Cargo Bot appears below.



Cargo arm which can be programmed – the aim is to program it to move the blocks so they appear as arranged in the goal window.

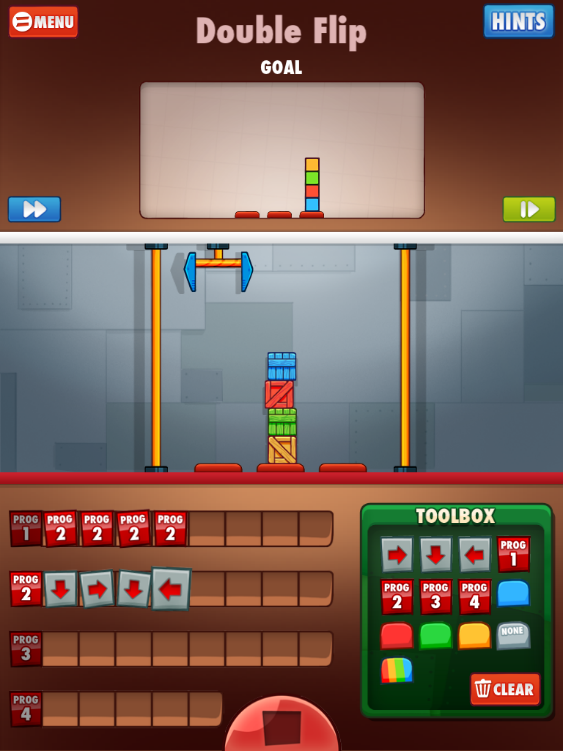
Commands need to be dragged to the program boxes and the program is run by clicking the green arrow

These are the commands which can be used to program the cargo arm

The Cargo Bot app main screen can be seen above. The game is simple, in that the cargo arm can be programmed with commands from the toolbox by placing the commands into the program boxes to the bottom left of the screen. The program is run (and the cargo arm moves following the program) when the green play button is pressed. The aim of the application is to write a program so the cargo arm rearranges the blocks on screen from the start position to the goal position as shown in the box at the top of the screen.

To familiarise yourself with the application further please have a go at completing the tutorial by clicking tutorial when the app opens. The tutorial will guide you through the process of programming the cargo arm and introduce you to a key factor of the game – **that the shorter/more efficient the code you write, the more points you get!** And the key to writing shorter code is, you guessed it, the use of repeat/loops!

This can be seen in the example below, whereby Program 2 has been coded so that the arm: moves down, moves right, moves down, moves left. ‘Program 2’ has then been run 4 times in Program 1 to move the blocks into the arrangement you can see. Can you think of the rest of the program to be written to complete the movement of the blocks so they are arranged as is shown in the goal window here?



It is envisaged that a session modelling how the cargo arm can be programmed and then providing pupils with time to attempt the challenges on this app, will help to refine their critical thinking and problem solving skills within the context of programming.