Animation

Computing Science Concepts

-Various inputs
-Sequence
-Repetition
-Counted loops
-Continuous loops
-Algorithm
-PRIMM

National Curriculum Programs of Study

Pupils should be taught to:

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

Research Focus

This planning uses **PRIMM**[1] methodology where the code is provided and pupils are encouraged to predict what it will do before investigating it, modifying it and creating their own version.

Computer scientists have **four levels of abstraction**[2]. The ideas level, Planning level (which includes the algorithm), code level, and execution level (testing the code).

Animation Planning Version C [PRIMM]

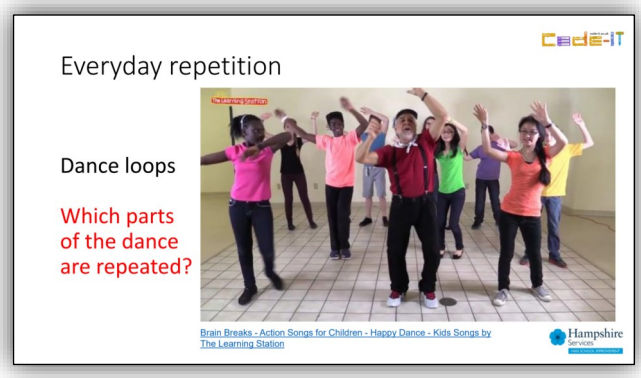
Overview

1. Introduce the concept of repetition through everyday examples
2. Extend the concept of repetition through role play
3. Introduce the challenge by sharing the idea, algorithm and example code for pupils to PRIMM [Predict, Run, Investigate, Modify, Make]
4. Learn about all the different animation techniques and how they were created
5. Pupils plan their own animations to go with either their monologues or dialogues
6. Pupils turn their algorithmic planning into their own programmed animations
7. Pupils fill in the summative assessment form

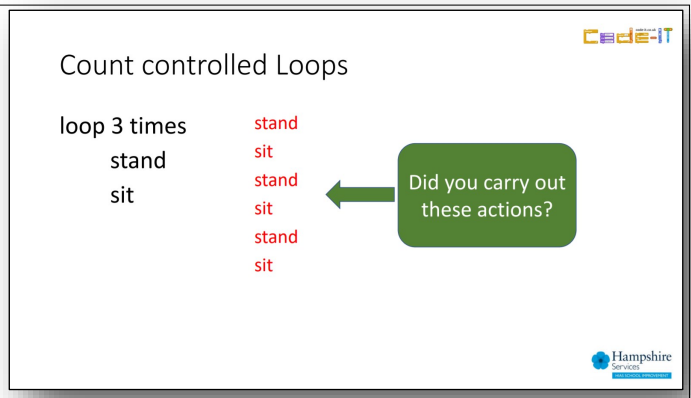
1. Introduce the concept of repetition through everyday examples

Download **Everyday computing concepts PDF** from http://code-it.co.uk/knowledge or directly from http://code-it.co.uk/wp-content/uploads/2019/04/everydaycomputingconcepts.pdf

Use the ideas in the repetition section to introduce the idea of repetition in our everyday lives. There is a You tube and Vimeo version to the dance activity. By linking the concept to its everyday use you are linking to known knowledge which means pupils are more likely to assimilate the idea.



2. Extend the concept of repetition through role play



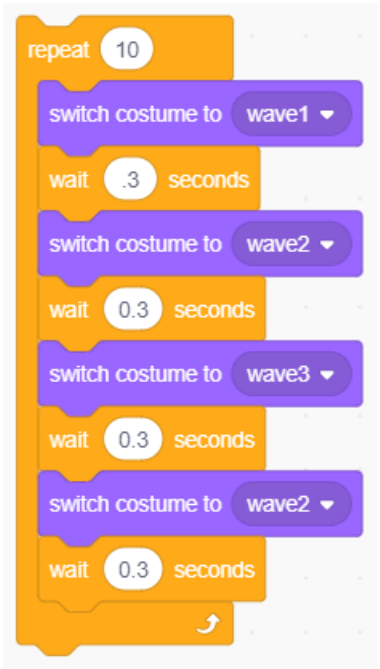
Download Concepts before coding PDF from http://code-it.co.uk/knowledge or directly from http://code-it.co.uk/wp-content/uploads/2019/04/conceptbeforecoding.pdf Follow the links in the menu to count controlled loops. Use those slides to roleplay and write simple fun repetition algorithms.

Move on to roleplay and write continuous loop algorithms before moving on.

Formative assessment opportunity

While pupils are writing their own repletion algorithms go round and check them all. Is anyone struggling? Have they copied the one on the board exactly? This is often an indication that they are not sure how to create their own or that spelling is an issue. A good supportive activity is to get them to tell you about their own sequence that you scribe for them.

3. Introduce the challenge by sharing the idea, algorithm and example code for pupils to PRIMM [Predict, Run, Investigate, Modify, Make]



Introducing the challenge

Explain that we have a well designed example that we are going to study and learn from before we make our own project.

Predict

But before we go into what it is about or run the code we want you to predict what you think each code section does from these offline code print outs. Give out **animation_vCPRIMM** page 1 and give pupils time to complete these. They could work in equal ability pairs to facilitate discussion.

Opening template file

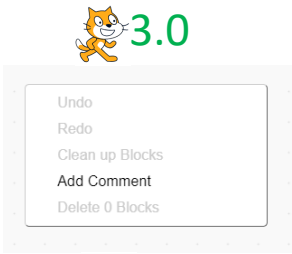
Direct pupils to open the Scratch **PRIMM example animation** file This can be found either on the Scratch website at https://scratch.mit.edu/projects/305130323/

or as a link on http://code-it.co.uk/gold/

You can also find downloadable Scratch 2 and Scratch 3 files on the same page as
animationexamplescratch2 ZIP
animationexamplescratch3 ZIP

Animation as a stand alone project

This project works well on its own. Create animations without linking them to the monologue and animation projects.



Example algorithm plan

Idea

Wave hand

Algorithm

Loop 3 x

Hand to right

Wait

Hand to left

Wait

A3 Extended planner last column

Animation
Start with stand 1
14 seconds
Hand wave loop
12 seconds
Nod head loop

Run the code

Was your predictions right? What do the other code sections (animal, wave, talk, nod) do?

Investigate

Give out **animation_vCPRIMM** page 2. There are two sections to the activity a find the code features and convert the code into an algorithm. The answers are on **animation_vCPRIMM_answers.** If pupils struggle with the algorithm activity remind them of the algorithms they wrote in part two.

Modify

Give out **animation_vCPRIMM** page 3. There are a set of things to modify and record what happens. The answers are on **animation_vCPRIMM_answers.** Alternatively you could ask pupils to modify code and record the changes using the comments.

4. Learn about all the different animation techniques and how they were created

See the double page printable pull out at the end of the module for animation techniques. Demonstrate these yourself or you can also use the videos found at

https://youtu.be/b0igJEQteTA for Scratch 2.0
https://youtu.be/mLzYpXcbv84 for Scratch 3.0

5. Pupils plan their own animations to go with either their monologues or dialogues

There are two parts to this plan. The first is for pupils to decide what they want to animate and how they want to do it. The second is to decide where it will fit in on their existing monologue or dialogue plans.

For the first part there is a idea and algorithm planner and for the second part they need to use the A3 extended planner last column. Where will their algorithms fit into the conversation flow. This is the part that we haven't shown, modelled or discussed so it may be worth briefly sharing a good one similar to that shown on the left.

Formative assessment

Check pupils ideas and algorithms as they are creating. Do their ideas make sense? Have they chosen a loop type? Have they indented what is going to be repeated by their loop? Do they refer to the type of pictures they are going to create? Have they included timing blocks to slow the loop down?

6. Pupils turn their algorithmic planning into their own programmed monologue

Give pupils time to do this and to test their creations. Does it fit in with the dialogue or monologue?

7, Pupils fill in the summative assessment form

You can find a summative assessment quick **Kahoot Quiz** linked at http://code-it.co.uk/gold/

Whilst Kahoot is a limited assessment tool it is free and it is easy for teachers to pass the results back to code-it via phil.bagge@code-it.me so I can look at which method provides best short test results. Not conclusive but useful.

If you pass the results back please

1, Anonymise the results by removing the names

2, Ask the head teacher for permission

3, In the email title say which module you are doing (ie Animation D)

Research References

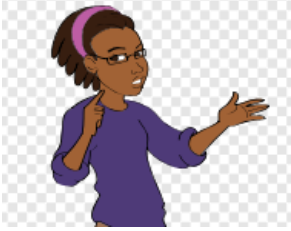
[1] PRIMM Sentence

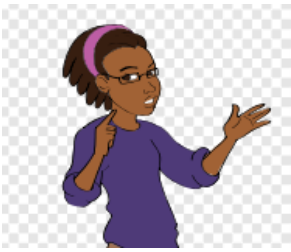
https://blogs.kcl.ac.uk/cser/2017/09/01/primm-a-structured-approach-to-teaching-programming/

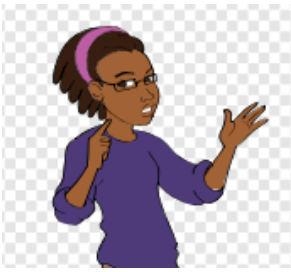
[2] Four levels of abstraction

This article includes an example of the four levels of abstraction and sign posts the work of Waite and Armoni in using them with school level pupils.

http://code-it.co.uk/algprogdiff/



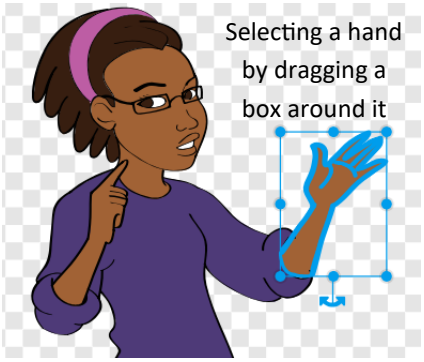




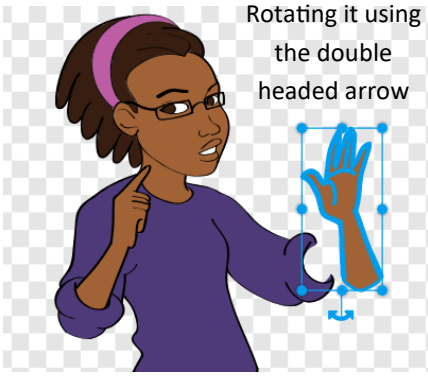
New to Scratch 3 is the ability to deconstruct parts of the pictures by left clicking and dragging a box around the parts that you want to use.

These also have a double headed arrow at the bottom so you can rotate the part you have selected.

You can move the part by holding it in the middle and resize it from the corner manipulation points.



Selecting a hand by dragging a box around it

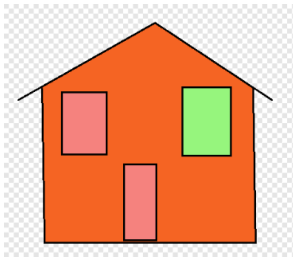


Rotating it using the double headed arrow

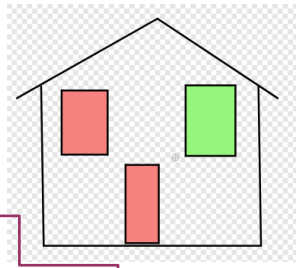
The bear and the ballet dancer and great examples of sprites with multiple costumes already created

3.0

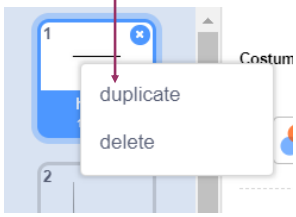
The house is made by starting with an empty sprite and renaming the first one as house.



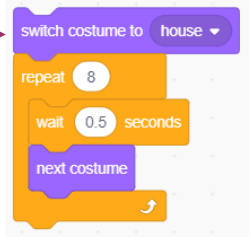
Drawing the first part of the picture in this case the base line.

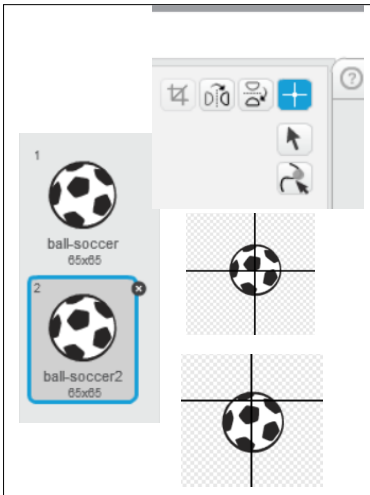


Then duplicating the picture and adding more details before duplicating and repeating the previous steps.



Novice programmers often exclude the initialisation of the sprite costumes and the need to start at the first picture.





Making a ball bounce in Scratch 2.0

Import a ball sprite

Duplicate the sprite

On the second sprite only select the centre tool.

Move it up or down

2.0

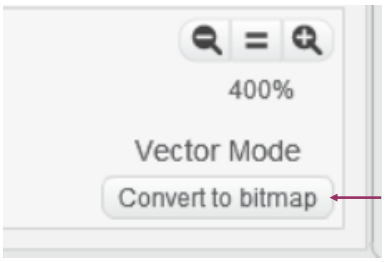
Making a ball bounce in Scratch 3.0

Import a ball sprite

Duplicate the sprite

On the second sprite only drag a box around the whole sprite.

Move it up or down from the middle.



Scratch 2.0 doesn't have the same ability to deconstruct pictures as Scratch 3.0

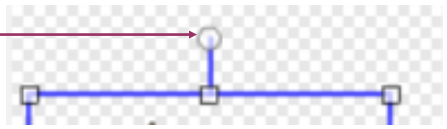
You will need to convert a sprite to bitmap which will reduce the quality of the picture which is noticeable in Scratch 2.0.

The animal, house and ballerina work the same as Scratch 3.0

Adjusting pictures needs a different method as shown.

Then select the select tool highlighted blue.

Drag a box around the area of the picture you wish to adjust.

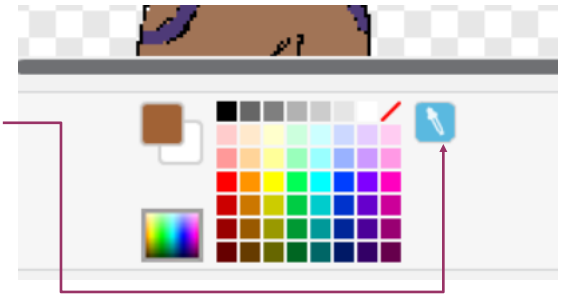


Use the rotate button above the select box to rotate your selection.

Once you have repositioned the picture you will need to use the colour choosing tool to find the right colour to fill in any damage to the picture.

You may need to zoom in to adjust the picture at pixel level.

Pixels are the tiny dots that makeup every screen. The more dots there are the greater the picture quality.



Once you have selected the colour chooser you need to left click on the colour you need and you can then paint using it.

Animation double page pull out