Plan and program a monologue

Computing Science Concepts

-Sequence

-Order can be important in sequences

-Algorithm

-Parsons problem

-Four levels of abstraction

-Inputs

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 a **parsons problem**[1] where the code is provided but pupils need to assemble it in the right order. This one has a twist as they are provided with the idea and algorithm to help them construct the programs.

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).

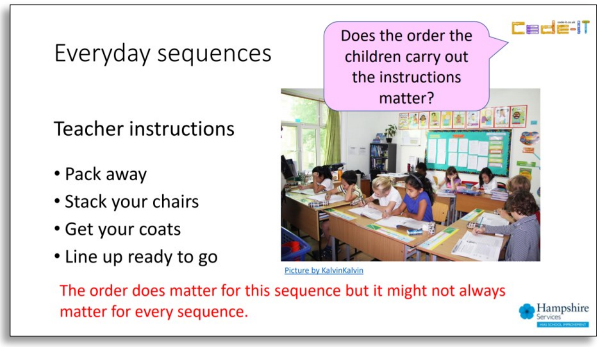
Monologue Planning Version A Parsons Problem

Overview

1. Introduce the concept of sequence through everyday examples
2. Extend the concept of sequence through role play
3. Introduce the challenge by sharing the idea, algorithm and unfinished code [Parsons problem]
4. Introduce new challenges to those who complete the earlier ones
5. Pupils plan their own short monologue
6. Pupils turn their algorithmic planning into their own programmed monologue
7. Pupils fill in the summative assessment form

1. Introduce the concept of sequence 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



Everyday sequences

Does the order the children carry out the instructions matter?

Teacher instructions

- Pack away
- Stack your chairs
- Get your coats
- Line up ready to go

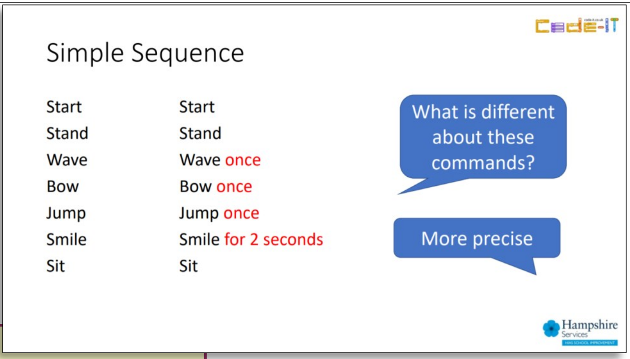
Picture by KalvinKalvin

The order does matter for this sequence but it might not always matter for every sequence.

Use the first eight slides to introduce the idea of sequence in our everyday lives. The answer mostly appear in red text. There are some sequences where the order is less important and some where the order is paramount. The same will be true for programming sequences. 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 sequence 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 simple sequence. Use those six slides to roleplay and write simple fun sequences. These slides introduce the idea that the more precise a sequence is the more useful it is. Stop when you get to the dance slide.



Simple Sequence

Start	Start
Stand	Stand
Wave	Wave once
Bow	Bow once
Jump	Jump once
Smile	Smile for 2 seconds
Sit	Sit

What is different about these commands?

More precise

Formative assessment opportunity

While pupils are writing their own sequences 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 unfinished code [Parsons problem]

Choosing a challenge

I recommend choosing one say and wait challenge and one timed say block challenge. You can leave the other for those who finish early. There is a description of the premade ones on the right hand side.

Introducing Scratch

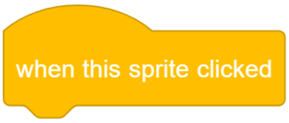
If pupils have never used Scratch before it is worth going over the basics. I have included some videos here Scratch 2.0 https://youtu.be/bNoyArexVns or Scratch 3.0 https://youtu.be/gtqMauyKE_w but I recommend that you watch them but introduce it yourself in short bursts.

Introducing Scratch algorithm

```
Repeat until basic Scratch introduced
    Show pupils basic feature
    Pupils have time to try it out
```

Sprites are pictures that we can give instructions too through dragging blocks and snapping them together in the scripts area.

There are lots of different types of blocks and they are colour coded with similar blocks. Many of these blocks will contain ideas that we won't know how to use until upper KS2 or KS3.



```
when this sprite clicked
```

Starting blocks have a curved top and can be found in the events section. Drag out the when this sprite is clicked starting block to show pupils.

Opening template file

Direct pupils to open the template file. This can be found either on the

What challenge to use

This module comes with four pre-made challenges.

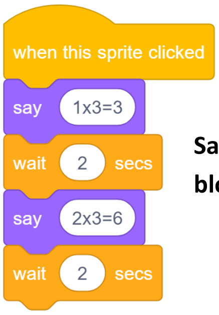
Tables –3 times tables (say and wait blocks)

Victorian –Victorian working conditions (say and wait blocks)

Julius C –His account of the Britons when he invaded (timed say blocks)

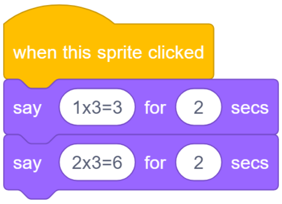
Churchill –His speech in 1940 (Timed say blocks)

You can of course make your own really easily and there is a blank word template to create the idea and algorithm. Scratch supports copy and paste. Make sure you muddle the blocks up.



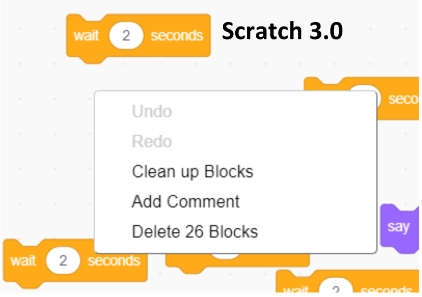
```
when this sprite clicked
say 1x3=3
wait 2 secs
say 2x3=6
wait 2 secs
```

Say and wait blocks



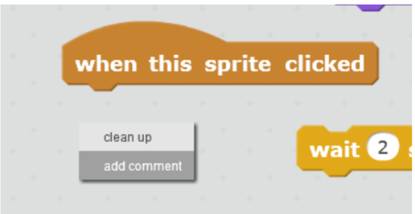
```
when this sprite clicked
say 1x3=3 for 2 secs
say 2x3=6 for 2 secs
```

Timed say blocks



Scratch 3.0

At some point you may wish to show pupils how they can clean up their code



Scratch 2.0

Idea and algorithm planner

If pupils are only writing a monologue use the A4 monologue planner.

monologuealgorithmplannerA4 PDF

If pupils are going to improve their plan with sound, stage, animation etch then use the A3 planner.

monologuealgorithmplannerA3 PDF

Scratch website at https://scratch.mit.edu/projects/303980204/

Or as a downloadable file to put on your network

Scratch 2.0 parsons_monoA_scratch2 ZIP

Scratch 3.0 parsons_monoA_scratch3 ZIP

You could also adapt this to make it read out the code or translate it into another language but that would increate the complexity of the project.

Introducing the challenge

Show pupils the idea and algorithm from the one you have chosen to do first. Explain that the author got as far as creating their planning and they have created the blocks but the blocks are not in the right order.

If you haven't shown them already show them how to snap a few blocks to together and warn them not to drop blocks on the blocks section as they will delete them.

4. Introduce new challenges to those who complete the earlier ones

Tell pupils that if they finish one they can test it by left clicking on the sprite. Once they have tested it they can ask for another idea and algorithm from you.

When pupils say they have finished ask them if they gave adjusted the times to match the algorithm? Very few pupils will notice this.

Summative assessment ✓

Why not have a class list and a list of the challenges that pupils can tick off as they complete them including creating an algorithm and their own monologue.

After the majority of pupils have created and tested a say and wait and timed say blocks version move on.

5. Pupils plan their own short monologue

Print out the idea and algorithm planner. You can simplify this by giving pupils an idea. This could have any cross curricular theme.

Formative assessment

Check pupils algorithms as they are creating. Do they make sense? Have they kept to their idea? Have they included punctuation? Have they included timings?

Check pupils algorithms before they turn them into code. Do they know what type of blocks their algorithm can easily convert into? Although it is easy to adjust an algorithm to output in either say and wait or timed say blocks.

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

Give pupils time to do this. Those that finish earlier than others can create a different type of 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)

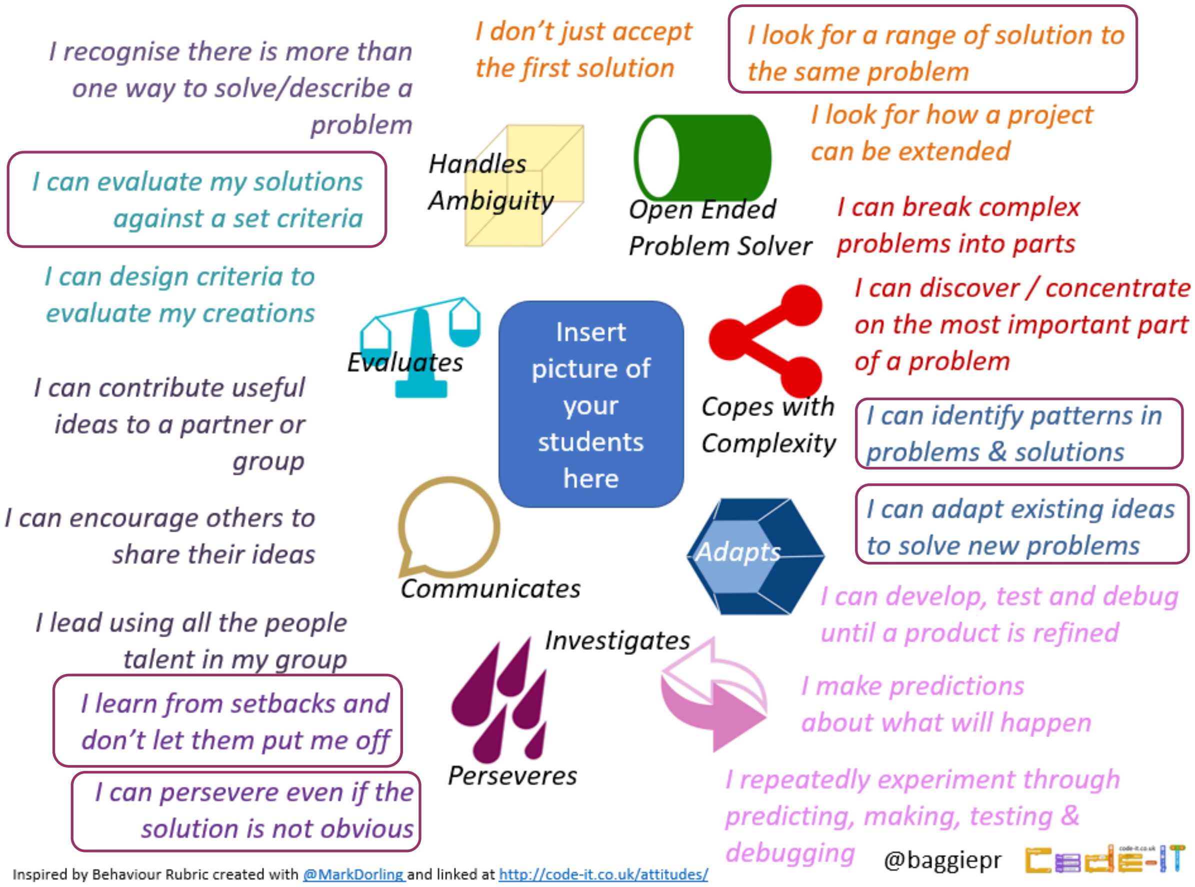
Research References

[1] **Parsons problems** https://www.computingatschool.org.uk/news_items/365

[2] **Four levels of abstraction**

This article includes an example of the four levels of abstraction and sign posts the work of Armoni and Waite.

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



I recognise there is more than one way to solve/describe a problem
I don't just accept the first solution
I look for a range of solution to the same problem
Handles Ambiguity
Open Ended Problem Solver
I look for how a project can be extended
I can evaluate my solutions against a set criteria
I can break complex problems into parts
I can design criteria to evaluate my creations
Evaluates
Insert picture of your students here
I can discover / concentrate on the most important part of a problem
I can contribute useful ideas to a partner or group
Copes with Complexity
I can identify patterns in problems & solutions
I can encourage others to share their ideas
Communicates
Adapts
I can adapt existing ideas to solve new problems
I lead using all the people talent in my group
Investigates
I can develop, test and debug until a product is refined
I learn from setbacks and don't let them put me off
I make predictions about what will happen
I can persevere even if the solution is not obvious
Perseveres
I repeatedly experiment through predicting, making, testing & debugging
@baggiepr
Code-IT

Inspired by Behaviour Rubric created with @MarkDorling and linked at http://code-it.co.uk/attitudes/