Plan and program a dialogue

Computing Science Concepts

-Sequence
-Order is important for some sequences
-Inputs
-Algorithm
-Four levels of abstraction
-Use modify create

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 **use, modify, create**[1] which involves using premade code before modifying it and finally building your own program. A process that takes the user from non ownership to ownership.

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).

Dialogue Planning Version B Use, Modify, Create

Overview

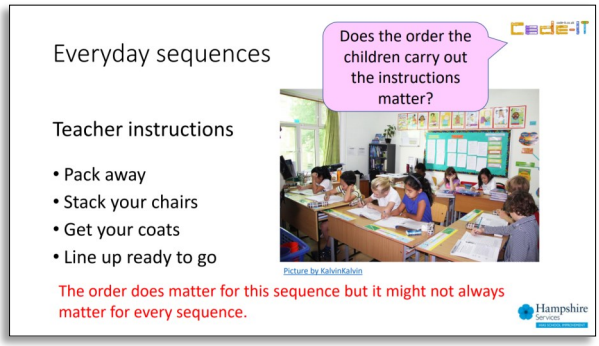
1. Introduce the concept of sequence through everyday examples
2. Extend the concept of sequence through role play
3. Introduce the challenge by sharing the idea, algorithm and finished code [use, modify, create]
4. Pupils plan their own short dialogue around a theme of yours or their choosing
5. Pupils turn their algorithmic planning into their own programmed dialogue
6. 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

If pupil used this material when creating a monologue skip this step.

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.

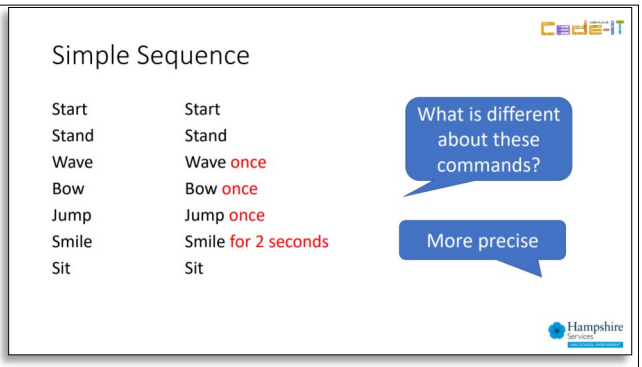


2. Extend the concept of sequence through role play

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.

If pupils used this material as part of their monologue planning then skip to the sequence dance challenge which was not included in the monologue planning.



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 finished code [use, modify, create] ⟺

Scratch basics

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.

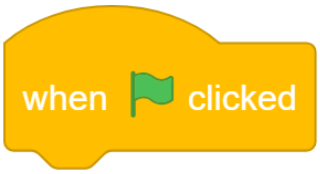
Repeat until basic Scratch introduced

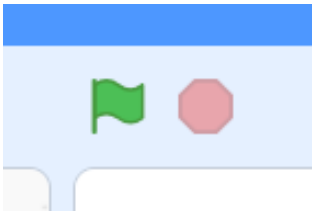
Show pupils basic feature

Pupils have time to try it out

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.

Starting blocks have a curved top and can be found in the events section. Drag out the when green flag clicked starting block and show pupils where this can be triggered on the Scratch display.





Which challenges to use

I recommend covering one say and wait dialogue and one timed say block dialogue one at a time. You can leave the other one for those who finish early. There is a description of the premade ones on the right hand side. Make sure pupils look at the idea and algorithm before they use and modify the code.

What challenge to use

This module comes with three pre-made challenges.

Evacuation Conversation

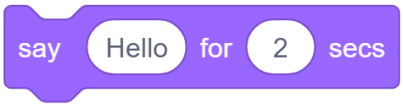
Version 1 (Timed say blocks)

Version 2 (Non timed say blocks)

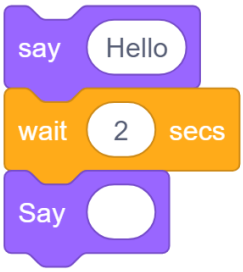
Bat

Timed say blocks

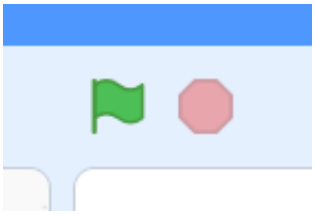
You can of course make your own really easily. Scratch supports copy and paste. Make sure you muddle the blocks up.



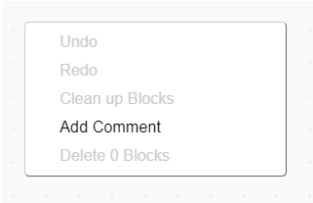
timed say block



say and wait



Run the code using the green flag button



Idea and algorithm planners

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

If pupils are going to complete the stage and sound module then use the left half of the A3 dialogue planner.

Opening the completed code

Direct pupils to open the template file. This can be found either on the Bat Scratch website at
https://scratch.mit.edu/projects/304421178/ Bat timed say blocks
https://scratch.mit.edu/projects/304424498/ Evac non timed say blocks
https://scratch.mit.edu/projects/304501931/ Evac timed say blocks

Or as a downloadable files to put on your network

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

Introducing the challenge

Show pupils the idea and algorithm from the one you have chosen to do first. Point out that each character has a column and that when one character is talking the other one is waiting without talking. It can be good to model this physically with the class by having a simple conversation.

Model how to switch between the code attached to both characters and show pupils that there is no code in the stage area. Don't run the code.

Use

Instruct pupils to run the code and see what it does. Can they describe to their partner how it works? Is there any part of the code that they don't understand?

Modify

Give pupils plenty of time to experiment with the code and modify it. Some pupils can be afraid to change things so make sure they know that you expect them to adapt things and they can always go and open a fresh template if they accidentally delete something. Can they tell you how they changed things and what those changes did? They could record their changes using comments in Scratch (right click on the scrips area, add comment)

An extra challenge can be had improving the evacuation program as it has original grammar and punctuation issues as well as times which are all the same.

Summative assessment

Why not have a class list and a list of the challenges that pupils can tick off as they complete them. Include using and modifying each example and creating an idea and algorithm for their own dialogue code.

Character 1 Modern Girl	Time secs	Character 2 Viking Boy	Time secs
Do your parents know you have that sharp knife?	3	Wait	3
Wait Zig Zag Pattern	3 Total Time	All the children in our village carry knives apart from the slaves.	3 Total Time
You have slaves, that is so wrong!	3 Total Time	Wait	3 Total Time

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

4. Pupils plan their own short dialogues

Give out the idea and algorithm planner to each pupil. You can simplify the design stage

by giving pupils an idea. This could have any cross curricular theme. You might want to model creating dialogue similar to the model shown on the left although they have already seen finished planning in the examples.

Now give pupils time to plan.

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? Does the character that is waiting time match the character speaking time? Is there mostly a zig zag pattern?

5, Pupils turn their algorithmic planning into their own programmed dialogue

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

6, 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] Use, modify, create Irene Lee et al Computational thinking for Youth in practice (2011)

[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/

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

I look for how a project can be extended

I can evaluate my solutions against a set criteria

Handles Ambiguity

Open Ended Problem Solver

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

Copes with Complexity

I can identify patterns in problems & solutions

I can contribute useful ideas to a partner or group

I can encourage others to share their ideas

Communicates

Adapts

I can adapt existing ideas to solve new problems

I can develop, test and debug until a product is refined

I lead using all the people talent in my group

Investigates

I make predictions about what will happen

I learn from setbacks and don't let them put me off

Perseveres

I repeatedly experiment through predicting, making, testing & debugging

I can persevere even if the solution is not obvious

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

@baggiepr