Plan and program a dialogue

Computing Science Concepts

-Sequence
-Order is important for some sequences
-Inputs
-Algorithm
-Four levels of abstraction
-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).

Dialogue Planning Version C PRIMM

Overview

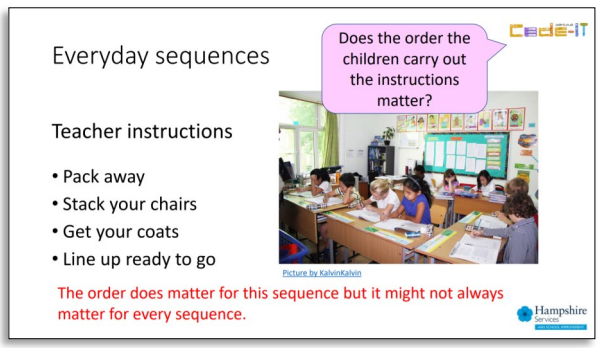
1. Introduce the concept of sequence through everyday examples
2. Extend the concept of sequence through role play
3. Introduce Scratch if it has not been used before
4. Introduce the challenge by sharing the idea, algorithm and example code for pupils to PRIMM [Predict, Run, Investigate, Modify, Make]
5. Pupils plan their own short dialogue
6. Pupils turn their algorithmic planning into their own programmed dialogue
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

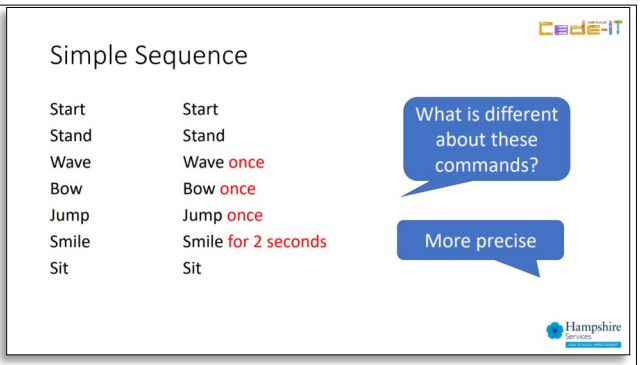
If pupil used this material when creating a dialogue skip this step.

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.

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

Formative assessment opportunity

While pupils are writing their own sequences go round and check them. 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 Scratch if it has not been used before

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.

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

Which templates 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.

What challenge to use

This module comes with three pre-made challenges.

Evacuation Conversation

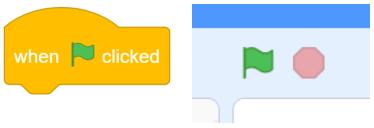
Version 1 (Timed say blocks)

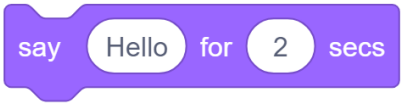
Version 2 (Non timed say blocks)

Bat

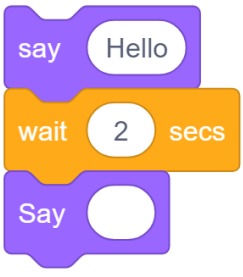
Timed say blocks

You can of course make your own really easily. Scratch supports copy and paste to create the idea and algorithm.

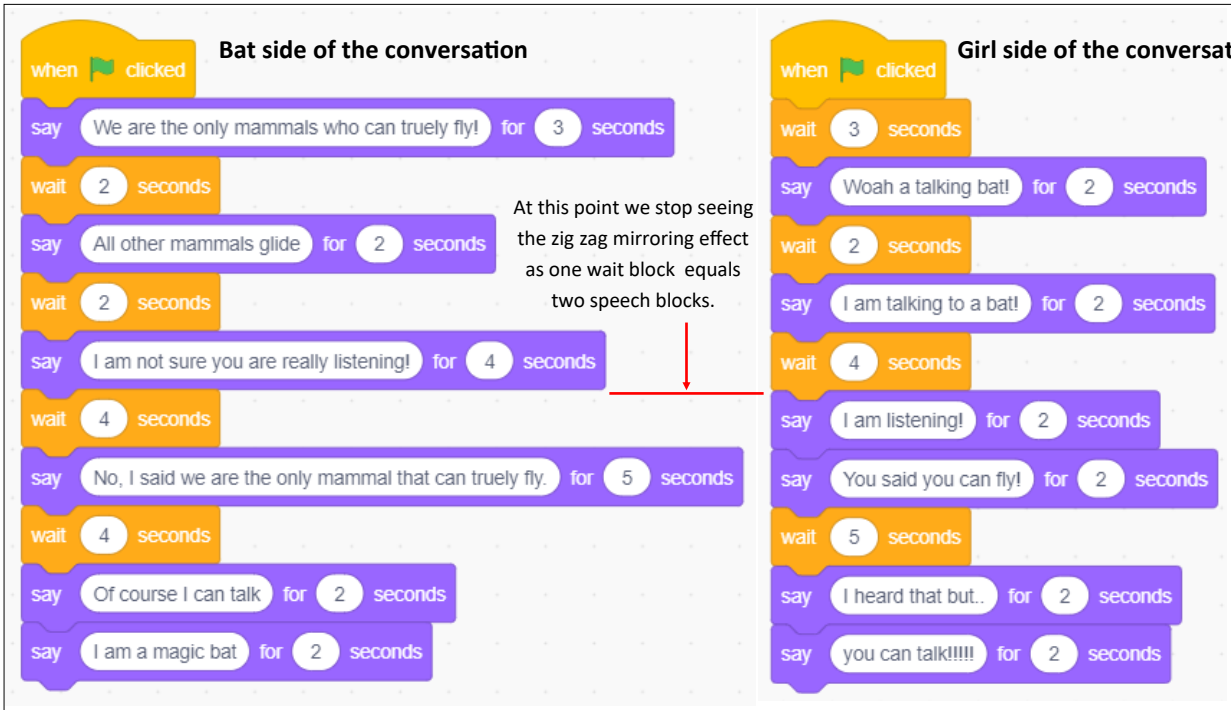




timed say block



say and wait



Open a template file

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/

Predict and run

Ask pupils to predict what they think the code will do before they run it. Once they have predicted to their programming partner and run the code give pupils the idea and algorithm that goes with their example. How close was their prediction to the authors idea?

Investigate & Modify

There are some slides **PRIMM Dialogue Module C investigates slides PDF** with some focus questions for pupils to answer with the right answers on the next slide. It can be good to work in similar ability pairs for this part of the lesson as it give space to discuss possible answers. Put up the first slide and give pupils a chance to explore and find answers before running through the answers with the class. Alternately you could give pupils the slides (with the answers removed) and they could work through them all before going through the answers with you as a class.

5. Pupils plan their own short dialogue

Give out the idea and algorithm planner. You can simplify this by giving pupils an idea or you could let them choose their own idea. This could have any cross curricular theme.

Grammar challenge

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.

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.**

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? Can you see one character talk while the other waits?

Check pupils algorithms before they turn them into code.

6, 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 dialogue.

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/