Computing Science Concepts

-Sequence
-Some sequences have a definite order
-Algorithm
-Use modify create
-Four levels of abstraction

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).

Stage, Sound & Movement Planning Version B [Use, modify, create]

Overview

1. Introduce the challenge of adding movement, sound and different backgrounds to an existing monologue or dialogue
2. Pupils use an example idea, algorithm and code [Use, modify, create] ⇔
3. Pupils modify an example idea [Use modify, create] ⇔
4. Pupils plan their own stage directions, music & background changes to go with a previously designed and created monologue or dialogue
5. Pupils code their stage directions, music & background changes
6. Pupils test their creations
7. Peer assessment of projects
8. Refinement of projects responding to peer assessment

1. Introduce the challenge of adding movement, sound and different backgrounds to an existing monologue or dialogue

Remind pupils of the monologues or dialogues that they created. Ask pupils what would improve the program? Backgrounds, sound, movement etc.

2. Pupils use an example idea, algorithm and code [Use, modify, create]

Give pupils either the bat example extended planning ⇔ or the Churchill example extended planning. Can they spot what was the original planning and what ideas have been added? Scene changes, costume changes, direction facings and wait periods from the start. Can they describe what these things do?

Changes are on the right of the A3 planners.

3. Pupils modify an example idea [Use modify, create] ⟺

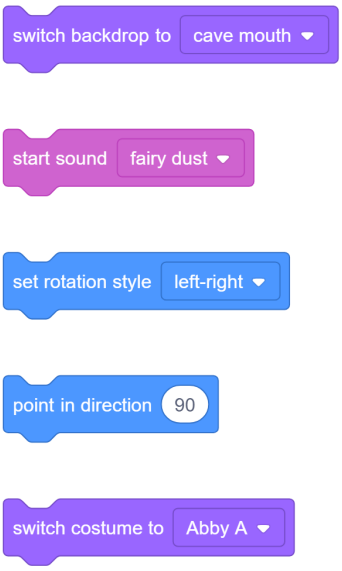
Pupils load either the **batSScompleteScratch file** or the **ChurchillSScompleteScratch file** for their version of Scratch.

Scratch Files

BatSScompleteScratch S2
BatSScompleteScratch S3
ChurchillSScompleteScratch S2
ChurchillSScompleteScratch S3

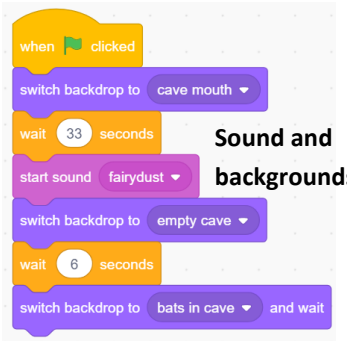
Use

Pupils run the code and discuss with their partners what it does. Remind them to look in the stage as well as sprites.



Modify

Pupils must modify three things that they have not used in either the monologue or dialogue parts of the project. This could be switching backdrops, starting or stopping sounds, facing right or left, pointing towards right or left or switching costumes.



Sound and backgrounds

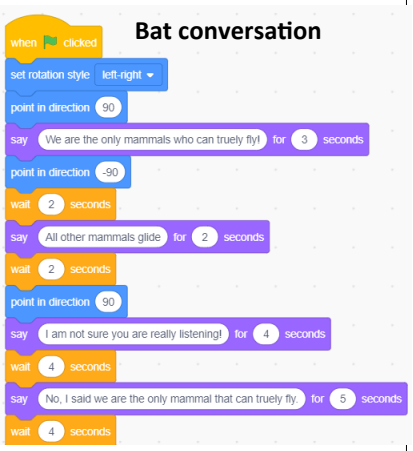
A3 Dialogue planner extended

Name	Class	Idea
		A conversation between a bat and a human where a few facts about bats are shared.

Character 1 Bat	Time secs	Character 2	Time secs	Total Time	Character 1 costume	Character 2 costume	Sounds	Backgrounds	Character 1 Facing	Character 2 Facing	Animation
					Bat a	Abby-a		Cave mouth	Bat face left right Bat face right		
We are the only mammals who can truly fly!	3	Wait	3			Abby c			Bat face left		
Wait	2	Woah a talking bat!	2								

Once complete

Once most have finished ask a couple of pupils to tell you about the aspects they changed and what happened then get everyone to tell someone sitting next to them what they changed and what effect it had.



4. Pupils plan their own stage directions, music & background changes to go with a previously designed and created monologue or dialogue

Pupils need their original monologue or dialogue planning which should be complete on the left hand side. If you completed the A4 planner by mistake then just stick it on the left hand side of the A3 planner.

It can help to give pupils access to Scratch so they can see what sounds are available but don't let them start making until they have completed the planner.

5, Pupils code their stage directions, music & background changes

Give pupils time to make and test their changes.

Formative assessment opportunity

Give pupils time to get started and then ask them to show you one change that they planned which they have carried out. Did they do what they said? Is it in the right place?

6, Pupils test their creations

Remind pupils to test their creations regularly and adapt what doesn't work. It helps to remind pupils to do this regularly.

7, Peer assessment of projects

A part way through the project give out post it notes and instruct everyone to set up what they have made so far. Everyone is to move around one place and assess projects using two stars (two good things about the project) and a wish (something that needs improving). Move pupils around a couple of times until pupils have written two or more sets of comments.

Ask pupils to read the comments carefully. If there are any comments that are rude or unacceptable then it is worth training your class in polite criticism.

8, Refinement of projects responding to peer assessment

Now give pupils time to adjust their projects taking into account the comments.

Summative assessment opportunities

You could assess the final finished project.

You could assess the planning.

You could assess how far the planning has been fulfilled in the final project.

You could ask pupils to tell you what a good version looks like (WAGOLL), collect their ideas and then get them to assess their own projects against the criteria.

Pupils could also assess their computational attitudes using some of the boxed statements over the page.

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

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/

2, Ask the head teacher for permission

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

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