

## Stage, Sound & Movement

### Computing Science Concepts

- Unified start
- Sequence
- Some sequences have a definite order
- There is more than one way to accomplish the same goal
- Screen output in pictures and text

### 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 different type of completion problem that I am choosing to call a **reverse completion** where all of the code is provided but pupils need to complete the algorithm.

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

## Stage, Sound & Movement Planning Version A [Reverse Completion]

### Overview

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

2, Pupils examine example code in detail

3, Pupils complete the partially finished algorithm using the code to help them [Reverse completion] 

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 examine example code in detail

Give pupils either the **bat example code** or the **Churchill example code**. Can they spot what was the original code and what code have been added? Scene changes, costume changes, direction facings and wait periods from the start.

3, Pupils complete the partially finished algorithm using the code to help them [Reverse completion]

Pupils examine either the **bat incomplete example stage sound planning file** or the **Churchill incomplete example stage sound planning file**. They are instructed to use the code to complete the algorithm. There is a completed one that teachers can use to give pupils hints.

If pupils get stuck refer them back to the code or a specific part of the code.

### Once complete

Once most have finished, show them the original planning. If pupils have written their own algorithmic commands rather than use your helpful list then it is perfectly alright for them to be different. As long as an algorithm is readable and makes sense and is not ambiguous it is fine to use varying instructions

### Feedback

As you go through the stage area did they get the costume changes, sounds in the right place, backgrounds in the right order with the correct waits, Are the facings written into the correct characters?

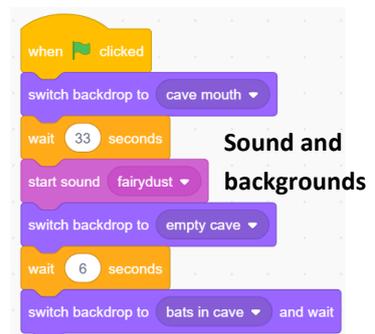
### 4. Pupils plan their own stage directions, music & background changes to go with a previously designed and created monologue or dialogue

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.

### Files

- BatSScompleteScratch S2
- BatSScompleteScratch S3
- ChurchillSScompleteScratch S2
- ChurchillSScompleteScratch S3



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	
We are the only mammals who can truly fly!	3	Wait	3		Bat a	Abby-a		Cave mouth	Bat face left right Bat face right			
Wait	2	Woah a talking bat!	2			Abby c			Bat face left			

### 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.





## 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 their projects using two stars (two good things about the project) and a wish (something that needs improving). Move around a couple of times until pupils have 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.

### Possible 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
- 2, Ask the head teacher for permission
- 3, In the email title say which module you are doing (ie Animation D)

### Research References

<sup>1</sup> Completion problems are suggested by Sweller in his work on cognitive load theory.

You can read about it [here](#).

<sup>2</sup> 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 can evaluate my solutions against a set criteria*

Handles Ambiguity



Open Ended Problem Solver

*I look for how a project can be extended*

*I can break complex problems into parts*

*I can design criteria to evaluate my creations*



Evaluates

Insert picture of your students here



Copes with Complexity

*I can discover / concentrate on the most important part of a problem*

*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*



Perseveres

Investigates



*I make predictions about what will happen*

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

*I can persevere even if the solution is not obvious*

*I repeatedly experiment through predicting, making, testing & debugging*

@baggiepr