# 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

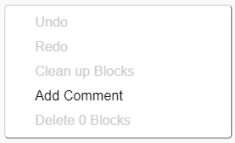
Pupils should be taught to:

**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**

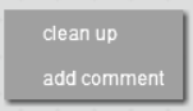
**use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs**

3.0




**To add comment Right mouse click in the code script area**

2.0




# Stage, Sound & Movement Planning Version C [PRIMM]

## Overview

1. Introduce the challenge of adding movement, sound and different backgrounds to an existing monologue or dialogue
2. Pupils explore an existing example using the [PRIMM] method ⟺
3. Pupils plan their own stage directions, music & background changes to go with a previously designed and created monologue or dialogue
4. Pupils code their stage directions, music & background changes
5. Pupils test their creations
6. Peer assessment of projects
7. 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 explore an existing example using the [PRIMM] method ⟺

Give pupils either the **bat example code** or the **Churchill example code.**

**Predict**

Before pupils are allowed to run the code can they work through the new scripts and identify what is new?

Can they then predict what the new additional code will do. Ask pupils to leave a comment by each new section of code or code addition explaining what they think it will do. This is a great activity to do with a partner of similar ability as it encourages dialogue about the code.

**Run**

Once pupils have run the code a few times ask them to identify any effects that they didn't identify. What did these do?

**Investigate**

Now give pupils the idea and planner which goes with the example that they are inspecting. The items on the right hand side are new. Can pupils find these in the code?

*What is different about the code and algorithm?*

The algorithm is simpler and doesn't have the detail the code needs to execute. The algorithm is set out for a human to read.

*Could the project author have chosen to put some code ideas in a different place?*

Yes the sounds and backdrop could be inside a character scripts instead of in the backdrop area.

*Is the order of these sequences important?*

Yes for the conversation which relies on a specific order to make sense.
No for some of the stage directions which could be in very different places in both examples.

**Modify**

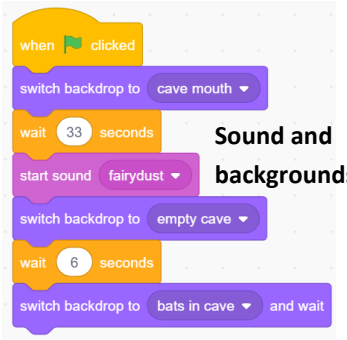
Give pupils time to modify the code.
*Can they add a sound?*
*Can they make a character face right and left at different times?*
*Can they remove the set rotation style left right block and run the code?*
What happens?

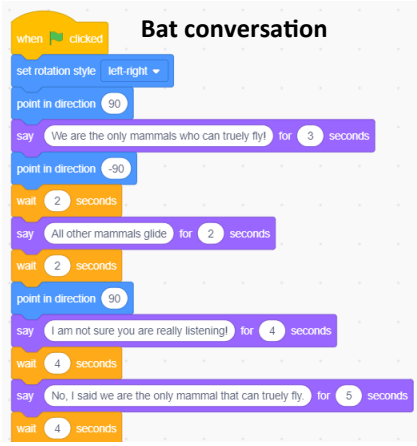
## 3. Pupils plan their own stage directions, music & background changes to go with a previously designed and created monologue or dialogue

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.

**Scratch Files**

BatSScompleteScratch S2
BatSScompleteScratch S3
ChurchillSScompleteScratch S2
ChurchillSScompleteScratch S3




**A3 Dialogue planner extended**

| Name Class | Idea<br>A conversation between a bat and a human where a few facts about bats are shared. |
|---|---|

| Character 1<br>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<br>right<br>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 | | | | | | | | |

## 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).


## 4, 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?

## 5, 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.

## 6, 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.

## 7, 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

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/

**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/