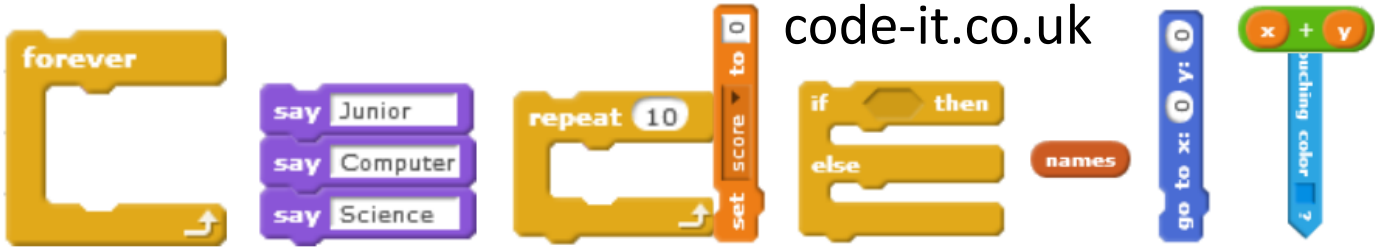


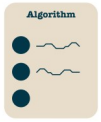
code-it.co.uk

Program Aim: Two characters take part in a programmed conversation (determined by teacher or students)

Conversation

Programming Concepts

-Sequence



Computational Thinking

-Algorithmic Thinking

Formative Assessment

For many activities we shall be defining what constitutes success and how we can assess this. Pupils that achieve this quickly can, in some circumstances, be moved on to the enrichment activity. Pupils that display a lack of understanding or partial understanding can be directed towards the corrective activity.

1, Hook	2, Algorithm	3, Algorithm to Code	4, Extending Algorithm	5, Extending Code

Learning Path →

Problem Solving Skills

Investigating

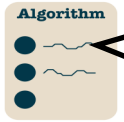
Open Ended Problem Solver

Evaluating

Perseveres

Where in KS2

-In my scheme of work I would put this in Year 3 (7-8 years old) but it could be used with older pupils as a quick introductory challenge



A precise set of instructions or rules to solve a problem

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**

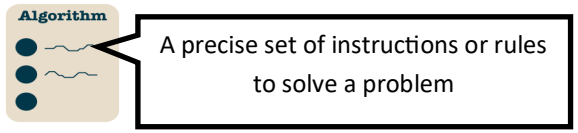
-**use logical reasoning** to explain how some simple algorithms work and **to detect and correct errors in algorithms and programs**

1, Share the Hook

Take a few moments to excite and enthuse your pupils about the conversation hook you or they have chosen.

Cross Curricular Focus

The basic premise can be adapted for use across all areas of the curriculum. It could be a dentist explaining dental hygiene techniques or a pair of stone age characters talking about their new tools. A powerful farewell at a train station as part of a WW2 evacuation scene or a campaigner trying to persuade a shopkeeper to stock Fairtrade produce.

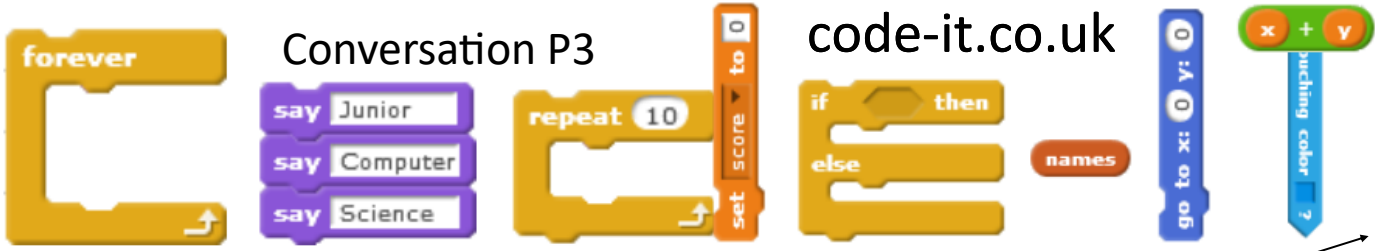


2, Conversation Algorithm Write **WAIT** in large letters on two whiteboards. Hand one to a child or adult helper and keep the other one yourself. Have a very simple conversation about something you both know about. When you are speaking the child should hold up wait. When they are speaking you hold up wait. Explain that they are waiting for the same time that you are talking and visa versa. Now open up an interactive whiteboard and import the first three lines of the conversation algorithm planner. Model a conversation opener that links to your theme. It could end up looking something like this.

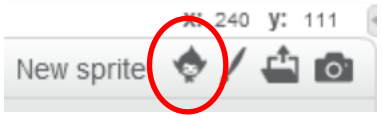
Character 1 Modern Girl	Time secs	Character 2 Viking Boy	Time secs
Do your parents know you have that sharp knife?	3	Wait	3
Wait	3 Total Time	All the children in our village carry knives apart from the slaves.	3 Total Time
You have slaves, that is so wrong!	3 Total Time	Wait	3 Total Time

Zig Zag Pattern

1, Shape Algorithm	**Success?** **Teacher Observation**	**Corrective Activity**	**Enrichment Activity**
Pupils have time to complete their conversation algorithm on the sheet provided. Give everyone at least 5 minutes before you start checking their work	Start with your learners who struggle with writing the most as they are the most likely to make errors in this algorithm. On the first scan of the classes work look for zig zag pattern as the conversation moves from side to side down the page. On the second scan look for times that are the same on each line as the wait of one character matches the speech of another. On the third scan look for good English that you can praise and others can magpie or adapt.	Take your wait white boards with you and model their speech and waits to point out where two people are talking at the same time for the same length of time. Be sure to emphasise where one character is out of synch with the others due to different timings. Check back to make sure they correct this. With very slow writers be willing to scribe the latter part of their conversation as long as they tell you exactly where to write it and how long to say it for.	A particularly literate child could be encouraged to adapt their conversation algorithm to include three people. Can they bring in a third character and continue their conversation on the three character sheet?



Scratch 1.4



Scratch 2.0

3. Modelling Turning Algorithm into Code

Import two sprites using the choose new sprite from file/library button.

Add a when green flag clicked starting block to each sprite and ask the pupils where the green flag is? (Top right Scratch 1.4 & Top middle Scratch 2.0)

Drag out a say hello for two seconds block and a wait block and start to turn your algorithm into code. Frequently refer back to your conversation and make a great play over matching the times and order of the conversation to the algorithm.

Character 1

Character 2

I can evaluate my solution against set criteria

I can design criteria to evaluate my creations

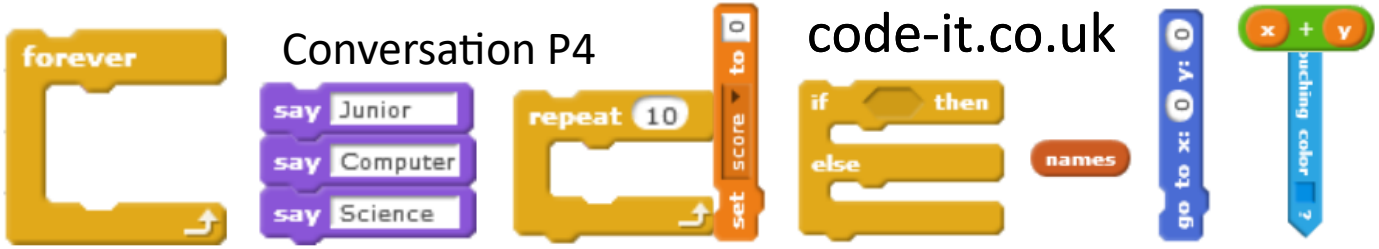
I can develop. Test and debug until a product is refined

Investigates

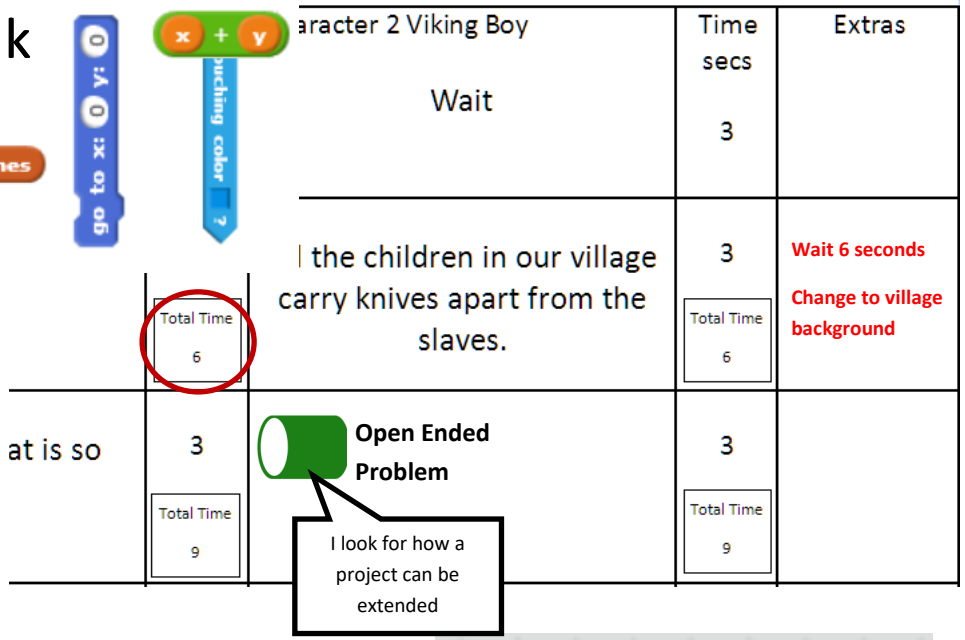
3, Turning Algorithm into Code Give pupils plenty of time to turn their algorithms into code.

Evaluating Programming 10/15 minutes into the session ask pupils to talk to their neighbours and discuss what criteria they will use to determine if their program works. Draw out that the conversation will make sense. That one character will talk while another will listen. That there will be enough time to read the speech bubbles but not so much time that the reader gets bored. That the subject matter will be interesting and engage the reader.

Evaluation debugging and refinement	**Success? Teacher/Pupil Work Observation**	**Corrective Activity**	**Enrichment Activity**
Pupils test their work and evaluate it against the class criteria decided earlier	Do the characters talk at different times? Does the speech make sense? Is the speech interesting and engaging?	Instruct pupils to go back to their algorithm line by line working their way through one character first and check their code is the same as the algorithm. Use literacy knowledge of the pupil to appropriately challenge/support them to improve their writing.	Don't forget to provide extra literacy challenge to those pupils who are fulfilling all evaluation criteria. You will know what areas you have been working on that can be developed through this writing opportunity.



4 & 5, Extending the Algorithm Extending the Code In these sections you introduce pupils to a few ideas and encourage them to experiment to discover new ideas for themselves. Before you start you get pupils to complete the total time boxes on their algorithm sheet. You also explain that the final column can be used to plan where different effects will happen. When you demonstrate how backgrounds and facing can be changed make sure you plan this on your algorithm sheet extras column first.



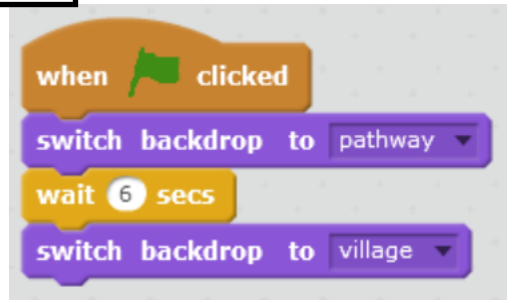
Change Backgrounds

Show pupils how they can import a background.

You may also wish to demonstrate how a background can be changed after a period of time through the use of a script.

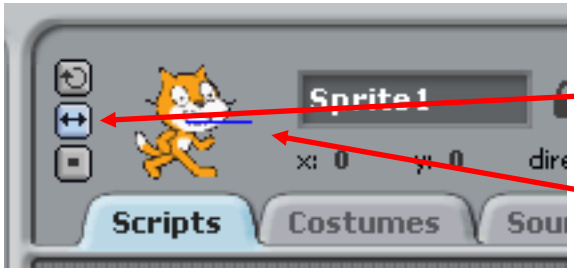
Scratch 1.4
-Stage
-Backgrounds
-Import

Scratch 2.0
-Stage
-Backdrops
-Choose backdrops from library



Facing

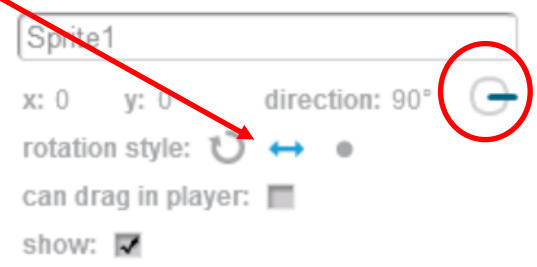
In a natural conversation characters look away occasionally. Scratch can program that as well.

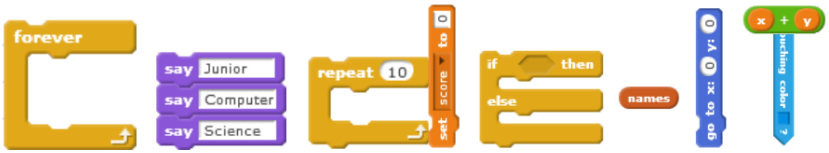


First set the character to left and right only. (Note you can flip the character manually by moving the blue direction of travel line to left or right.

Now you can build up a script using waits and point in direction blocks.

Select i in Scratch 2.0 first





Where Next?

Explore other programming planning at code-it.co.uk/csplanning

Design your own programming planning using computational thinking & problem solving skills

You can follow me on Twitter

@baggiepr

You can find lots of free resources at code-it.co.uk

I enjoy sharing my journey with fellow educators around the world

Develop Pupils into Computational Thinkers

- Algorithmic thinkers — Think through steps or rules to achieve something
- Algorithmic evaluators — Which algorithm is best, most efficient?
- Decomposers — Break problem up into parts and solve parts separately
- Abstractors — Find the most important part of problem
- Generalisers — Adapt solution from one problem to solve something else

Summative Assessment Criteria

- Conversation will make sense.
- That one character will talk while another will listen.
- That there will be enough time to read the speech bubbles but not so much time that the reader gets bored.
- That the subject matter will be interesting and engage the reader.

Pupils could assess each others work using these criteria

Complexity	I can break complex problems into parts	I can discover / concentrate on the most important part of a problem	I can explain how I used decomposition & abstraction
Ambiguity	I recognise there is more than one way to solve a problem	I recognise there is more than one way to describe a problem	I can explain how I managed ambiguity
Open Ended	I look for a range of solution to the same problem	I don't just accept the first solution	I can describe how a project can be extended
Adapt	I can adapt existing ideas to solve new problems	I can identify patterns in problems & solutions	I can explain how I adapted a solution to solve a new problem
Evaluate	I can evaluate my solutions against a set criteria	I can design criteria to evaluate my creations	I can explain how evaluation helped me improve a project
Experiment & Debug	I can develop, test and debug until a product is refined	I repeatedly experiment through making, testing & debugging	I can explain how using the iterative cycle improves my work
Persist	I can persevere even if the solution is not obvious	I learn from setbacks and don't let them put me off	I can describe how I overcame problems
Communicate	I can contribute useful ideas to a partner or group	I can encourage others to share their ideas	I can lead using all the people talent in my group

Decomposition | Abstraction | Generalisation | Algorithmic Evaluation | Algorithm

Adapted from Behaviour Rubric created with @MarkDorling and linked at http://code-it.co.uk/attitudes/ Adapted with permission