Year 2 : Algorithmic Evaluation through Bee-bots

Program Aim: There may be many algorithmic solutions to a problem, evaluation asks which one was best and why?

- Assessing that an algorithm is fit for purpose
- Assessing whether an algorithm does the right thing (functional correctness)
- Designing and running test plans and interpreting the results (testing)
- Assessment whether the performance of an algorithm is good enough
- Comparing the performance of algorithms that do the same thing

(Adapted from Computational Thinking framework)

Programming Concepts
-Sequence

Computational Thinking
-Algorithmic Thinking

Handles Ambiguity

Copes with Complexity

Open Ended Problem Solver

Investigates

Persists

Communicates

Evaluates

Adapts

National Curriculum Programs of Study

Pupils should be taught to:

- understand what **algorithms** are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions
- **create** and **debug** simple programs
- use **logical reasoning** to predict the behaviour of simple programs

Computational Thinking
Algorithm is a precise set of instructions or rules to achieve an outcome or solve a problem.
Algorithmic Evaluation is how we look at algorithms and determine how useful they are, how adaptable, how efficient, how correct.

Learning path:

Session 1: The Hook

Session 2-4: Routes

Session 5: Algorithm to code

Session 6: Building complexity

Previous learning suggested for Reception: http://code-it.co.uk/wp-content/uploads/2016/06/guidedplayreception.pdf
Previous learning suggested for Year 1: http://code-it.co.uk/wp-content/uploads/2016/06/y1beebotplan.pdf

Session 1: The Hook

Decide on a context which links to a project you are doing. Geography or local history are popular but it could be anything. If pupils have not had opportunities to create algorithms and convert these into Bee-bot code before, start with the year 1 Bee-bot planning. Children should have the opportunity to create algorithms using the symbol cards prior to this module.

Drawing Bee-Bot World: Draw a grid of 15cm2 squares across a large sheet of sugar paper before the lesson. Use the Bee-Bot World slides to help pupils design and create their own world. Use your own theme locations, not the ones of the slides. Pupils work in pairs to design Bee-bot world.

Session 2-4: Routes

Evaluation

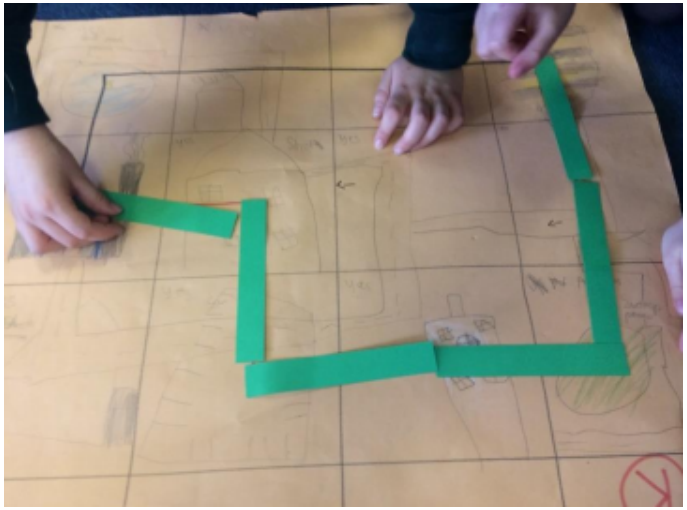
As pupils engage with this task they are evaluating by comparing different algorithms that do the same thing. They are making value judgements that may involve the number of instructions (less is better) and the distance it might need to go (shorter is better) or the time it might take (less is best). Some pupils may suggest factors that are not obvious such as the route that visits the most landmarks. All are valid and can be investigated.

The best route

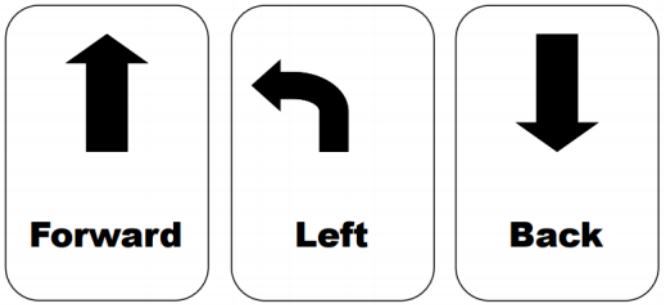
Preparation before the lesson Draw two routes from A to B on the Bee-bot world maps. Draw them in different colours (red & Black). Make sure each map route is different.

Ask pupils which route is best out of the two routes.
Why is it best? Allow them time to discuss it in pairs. Choose suggestions to investigate.
E.g. least/most distance covered, time taken

It is easier to investigate one factor with all of the class. Pupils rotate round each map, testing it for distance, time, number of instructions or whatever other criteria they thought was best and you selected to test.



Cross-curricula link:
Strips of 15cm long card are great for calculating distance.
Stopwatches or whiteboard timers can be used to investigate time.



The algorithm symbol cards are useful for investigating the number of instructions.
Recording sheets for these activities are here:
http://code-it.co.uk/beebot

Session 5: Algorithm to code

Session 6: Building complexity

Collect the most amount of counters with a limited number of cards	**Success? Teacher observation** Collect the highest number of with a limited number of algorithm symbol cards
The task here is to design an algorithm to pick up the most counters with a set number of cards starting from the same starting point and same Bee-bot position. Complexity could be added by having squares that Bee-bot could enter but would cost a pause card or by increasing or decreasing the number of cards. 	 A variation of the counter task is to write numbers on all grid squares and challenge pupils to reach the highest total using a limited number of symbol algorithm cards. They can only visit a square once.

Where next?

Explore other programming planning at code-it.co.uk/csplanning
Design your own programming planning using computational thinking & problem solving skills.

Author: Philip Bagge
Find more resources at code-it.co.uk
Follow me on twitter @baggiepr

Designer: Amanda de Fraine
adefraine@hotmail.com
Follow me on LinkedIn

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

This work is licensed under the Creative Commons Attribution-NonCommercial 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA