Primary Computer Science

KS1 Turtle Progression By Phil Bagge @baggiepr v1.2 revised 13-12-14

KS1 Programs of study covered
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.
Evaluation is how we look at algorithms and determine how useful they are, how adaptable, how efficient, how correct. 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)

Using Bee-Bots or Roamer Too with simple interface

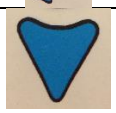
These are **not** pupil ready activities. You will need to give them exciting relevant cross curricular context and decide when and which pupils are ready to move on. I recommend pupils working in pairs or small groups when using turtles (Bee-Bots, Roamer Too etc). Knowing their right and left is also important and there are some ideas you could adapt here.

Play

Pupils need lots of opportunities to play and explore using the turtles for their own reasons before you move on with any of these activities. Playing helps pupils really understand the logical rules the device is constrained by. Too little play and they will struggle later on. You can tell children that the turtles turn without moving forward or backwards but they won't understand that until they have explored it and played with it and worked it out by themselves. It is good to have a range of structured and unstructured play opportunities. The structured ones can be facilitated using sound blocks or cards to record challenges.

Do you know what every command does?

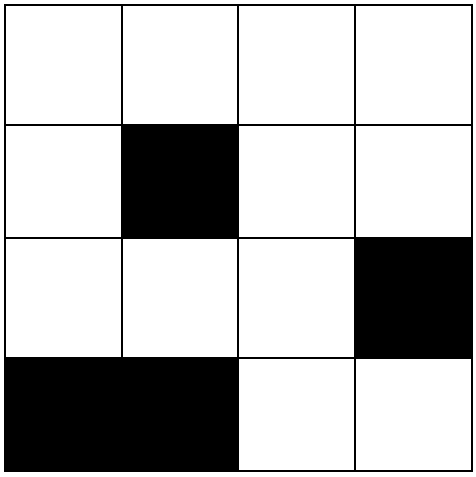
Can you describe it to a partner?
Could your students create a video to explain what they do?

Roamer Too	Bee-Bot	What do they do
		
		
		
		
		
		
		
		

(As a teacher did you include how far they went? Did you try the turns on different surfaces?)

Turtle Marbles

Have a quick round of turtle marbles. Use a premade mat or mark out a 4x4 or 5x5 grid. Place some obstacles into some of the squares. The first person programs their turtle onto the grid in a place that will be hard to hit with the second turtle. The next person tries to program their turtle to bump into the other one. Each player takes it in turns until there is contact. Turtles cannot be turned or moved by hand.



From A to B

Can you program the turtle to move from A to B? This could be on a mat with interesting cross curricular starting and finish points, or student designed starting and finishing points, or inside masking tape marked out squares. Pupils are creating an algorithm in their heads before programming the digital device.

From A to B creating symbol algorithm first

Symbol Algorithms

Use ready bought cards or print and cut out these.
Pupils could also draw arrow symbols on whiteboards.

Decide on challenges that fits your curriculum. Populate the mat or marked out grid with destinations. List the A to B challenges in order of complexity. Can pupils create a sequence of cards (algorithm) to go from A to B. Get your partner to test your algorithm using the turtle. Did these need debugging (fixing)? Working through these you are evaluating your algorithm and asking whether it does the right thing.

From A to B to C creating symbol algorithm first

Same as above but with more destinations

From A to C avoiding B

There are lots of variations on this theme

Reading out the algorithm cards alongside the bee-bot program

A really good strategy for making the link between algorithm and programming and to aid debugging is to encourage pupils to read out their cards as the turtle performs its instructions. This makes it much easier for pupils to spot errors called bugs in programming. This is similar to programmers tracing their more complex code to work out what it does before testing it on a digital device.

Cards on the mat

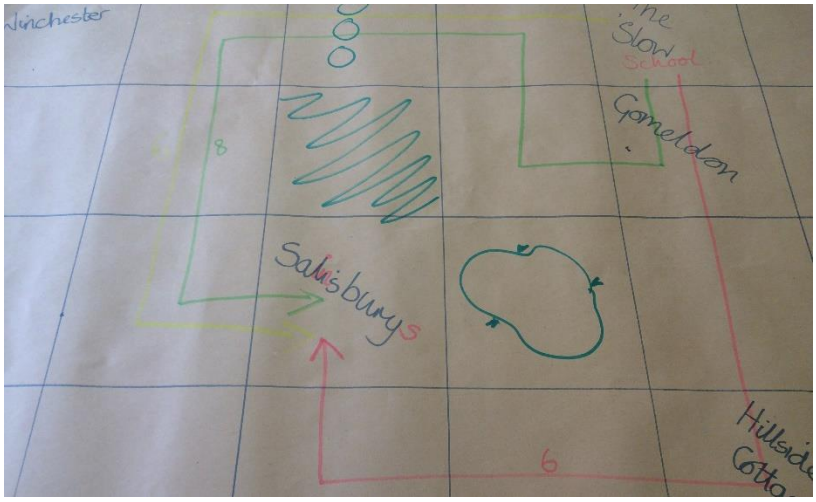
Some pupils find it very helpful to place the cards on the route the turtle will take instead of alongside the mat

The best route

Set up an interesting grid that avoids predefined paths such as roads or rivers. Give pupils a symbol algorithm printed on paper to get from A to B. Make sure this is not the quickest, shortest or easiest route. Ask pupils to test it and say if it works or not? If they volunteer that it is not the best route ask them what would be and why would this be best? If they don't volunteer any improvements ask them if it is the best route. Could they work out a better route? Why would it be better? As pupils engage with this task they are evaluating their algorithm 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).

Adaptations of the best route created by teachers at Wickham C of E Primary School during a computing inset day

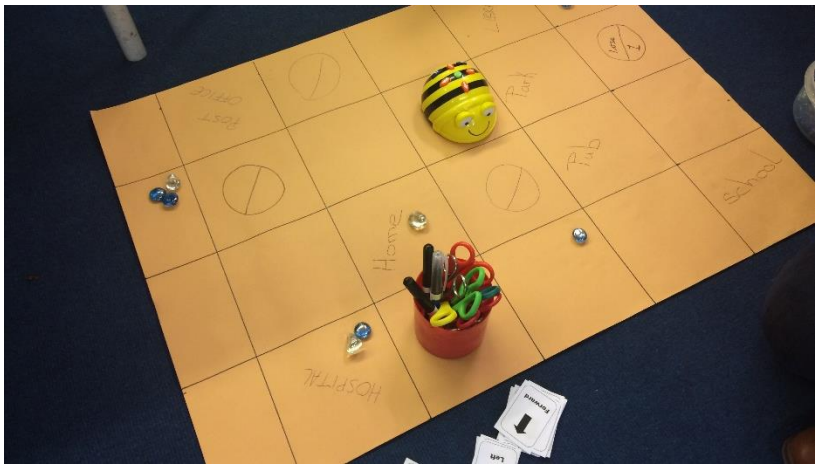
The best route out of a selection of marked routes



The task here is compare routes that start and finish in the same place with each other. Looking at distance travelled by the number of 15cm* strips needed for each route. Time taken by comparing the number of seconds needed for each route. Number of instruction symbol cards needed for each route.

*Bee-bot

Collect the most amount of counters with a limited number of 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 turtle facing. Complexity could be added by having squares that turtle could enter but would cost a pause card or by increasing or decreasing the number of cards.

Collect the highest number with a limited number of algorithm symbol cards

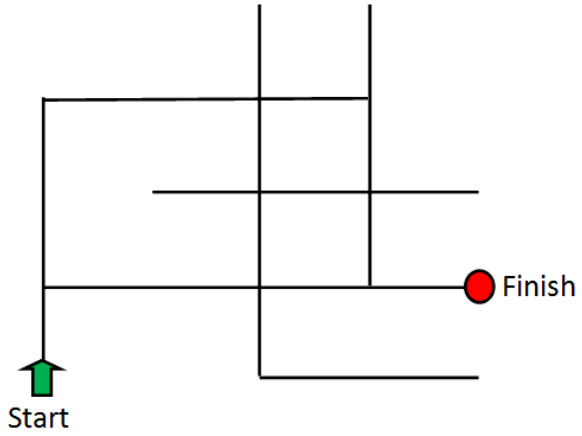
A variation of the idea above is to place number cards on some grid squares and challenge pupils to reach the highest total using a limited number of symbol algorithm cards.

Unfinished Algorithm

Set up an interesting grid. Give pupils an unfinished symbol algorithm printed on paper to get from A to B. Miss out the go button for the first one. Get them to test it. Did they spot the problem? Could they fix it? They could draw in the missing step? As pupils do this they are evaluating if the algorithm is fit for purpose. They are also debugging the code. Some pupils might even get onto creating their own unfinished algorithm problems for other to solve.

Turtle Maze

Set up a maze using masking tape. Don't use walls but just do lines down the middle. You will need to measure the distances depending on your device. Make sure there is more than one solution. Mark the start and finish point. Can pupils create symbol algorithms for all the different routes? Which is the fastest? Which is the slowest? Which takes the longest distance? They could measure it using cm or a piece of string or the number of roamer steps (make card rulers the length of the turtle step) depending on their maths ability. If you have simple timers or stopwatches they can use these. You can get a lot of commands inside a minute without having to convert seconds to minutes.



As they consider these issues they are evaluating the performance of algorithms that do the same thing. Could they make a video to explain why one route was better than another?

Acknowledgement

With thanks to Emma Goto who checked the work for KS1 errors and suggested refinements.

Thanks also to Prof. Paul Curzon, Mark Dorling, Thomas Ng, Dr. Cynthia Selby & Dr. John Woollard for their excellent document *"Developing computational thinking in the classroom: a framework"* one of the most useful documents I have read in the last year.

You can download a copy here

http://community.computingatschool.org.uk/resources/2324

All errors are my own

Phil Bagge