A discovery of the simple rules to build termite mounds. A scaffolded implementation of the rules to illustrate new techniques and consideration of further pedagogy.

Preparation required:
Termite discovery activity resources, instructions and rules.
Termite investigation tasksheet and sample programs.

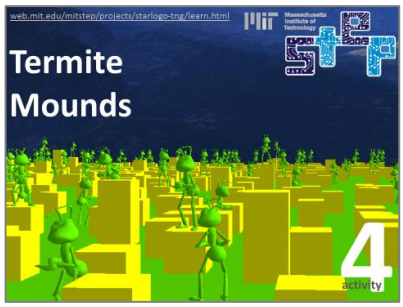
Termite Mounds

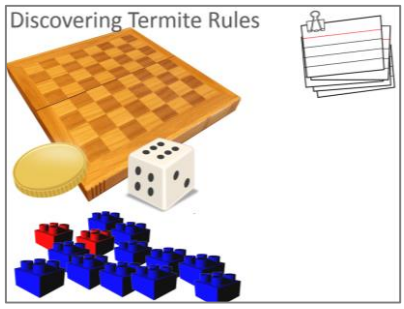
Many natural phenomena exhibit 'emergent' behaviour. Take the flocking of birds - it looks like a leader bird must be supervising and co-ordinating the flight. In actual fact, the behaviour emerges from the actions of individual birds. Moreover, the rules for individual birds are remarkably simple and involve only local communication with their neighbours. If time allows (8 min), show an excellent introduction to 'complex adaptive systems' and how they can be modelled: youtu.be/g5evD6AQeCQ. It makes links between different fields investigating the potential impact of emergence and includes interviews with many of the pioneers at Santa Fe developing StarLogo material for use in schools. If time is short, or for use in a classroom, youtu.be/CPHjsWSzOY0 is a 4 minute version produced by Project GUTS.

Let's continue our theme of exploring the natural world and stay on the African Plains. Our subject this time though is not the Zebras but the structure behind them in the picture on the slide. It's a termite mound and these incredible structures can be huge. They are a real wonder of nature, built by the collective efforts of tiny insects. Termite mounds are another good example of the sort of decentralised phenomena that occur in the natural world. They look like they must have been designed, or at least something must have supervised and directed their construction. In actual fact, the rules for individual termites are very simple – the complex structure emerges out of the simple interaction of many agents.

This exercises is based on materials produced by the Scheller Teacher Education Partnership at MIT. There are lots of other excellent materials hosted here, well worth exploring: http://goo.gl/fp0JcR.

Remember, we're interested in exploring and developing an 'ideas' model. The picture shows the start of our model. Scattered around our world are small yellow blocks which represent woodchips. The termites will wander around randomly, using the techniques we already know – our wiggle walk. As they do so, they'll follow some simple rules. The following slide is a video showing how simple 'termite mounds' emerge in this model. It starts with the model setup. We're looking at it using the 'agent's eye' initially so you can see the terrain is just a scattering of blocks. A click runs the video.

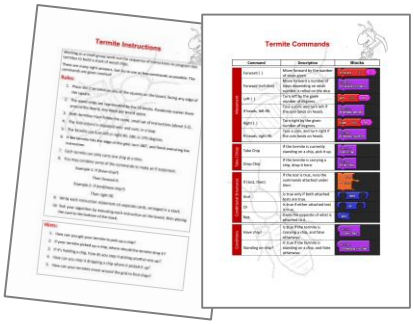




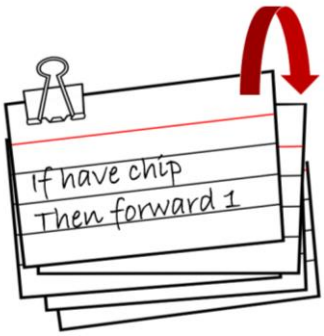
Discovering Termite Rules

The challenge for students is to work out the rules governing the Termites behaviour. This works well as a small group activity. Each group will need a grid to simulate a small termite world – chess boards are ideal. If these aren't available, photocopy the grid included and put 4 together to form and 8 x 8 grid. Some stackable blocks, like Lego can represent woodchips. Each group will need about 10, ideally the same colour. They should be scattered around the board, no more than one on a square. Two further blocks (different colour) can act as Termites. Alternatives could be counters or similar. Place anywhere on the board. Some way of identifying the head is required to indicate the direction they are facing. A coin to flip and a dice to roll are also required, and a stack of 5 or 6 cards to write instructions on. Software dice and coin flippers are available at goo.gl/HonXMb and goo.gl/yJh2QY.

Finally, each group requires a set of instructions and a list of commands to use. Give these out so you can refer to them in the following slides.



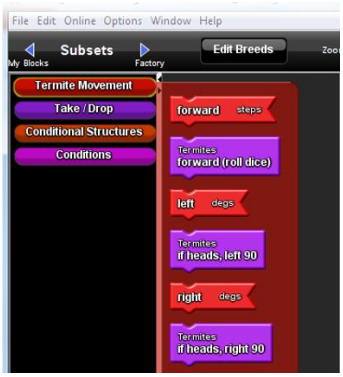
How might we best tackle this problem? Remember, the key to handling complexity is to break a task into smaller parts. What separate parts can students identify in this? We need to get Termites moving randomly, picking up and putting down chips. Note that the commands they must use are grouped together on the sheet to help. The random movement is the easiest to tackle, so suggest students start with this.



The groups should write one instruction statement on each card. They MUST be composed using the commands on the sheet. Use an example to demonstrate. There are many ways to build mounds, but try to restrict students to 5 or 6 cards maximum. The stack of instructions represents an infinite loop that applies to both Termites. They can test their loop by performing the action on the top card on both Termites, then placing it at the bottom of the stack - demonstrate this. If the sequence seems to works, scatter the blocks again and test the different layout. It is often surprising how lax children can be with testing. If they think they have a sequence of commands that work, ask them to demonstrate.

Implementing The Model

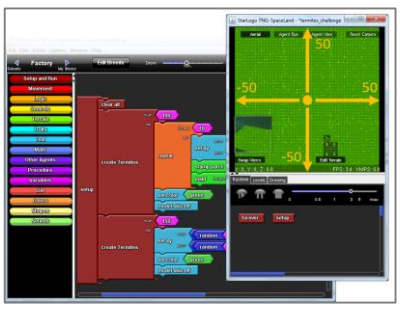
Once students have a set of rules we can try implementing them in StarLogo. Open termites_challenge.sltng. The setup procedure has already been defined, which we will look at in a moment. The challenge for students is to complete the forever loop which is executed at runtime.



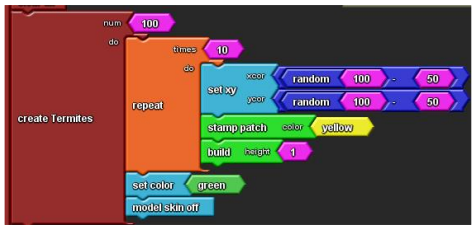
One of the barriers to mastering StarLogo is the number of drawers and commands available. By scaffolding this activity and insisting they use only the commands on the sheet in the preparation activity we can minimise this hurdle. So far we've highlighted the drawers available in the 'Factory' and 'My Blocks', but there is a third option 'Subsets' which is empty by default. Select it now. In this file it is configured to include four drawers that include only the commands available on the sheet. You can configure your own Subset drawers for any project using the tools available under Edit menu. It is straightforward, but a help sheet from MIT is available from goo.gl/ezSCGj, as are other quick start sheets. Restricting commands in this way allows you to set tightly focused exercises.

Before asking students to configure their forever loop, let's initiate the model through the Setup routine. The Setup button scatters blocks around SpaceLand and creates an army of Termites. Spend a moment looking at the code in the Setup space and invite the group to explain it. Reading code aloud is a good test of understanding. Can they explain what it does?

Start with the second batch of 'Create Termites' commands – what do they do? 150 Termites are created and scattered at random co-ordinates. We could have just scattered the Termites but this allows us to introduce the co-ordinates in SpaceLand. The SpaceLand world radiates out from the middle, the X co-ordinates extending to plus and minus 50. Similarly the Y co-ordinates. Once scattered, their colour is set to green and the 'skin' of the termite model is removed. This allows us to change the colour of the termites within the code. This is another useful technique as we shall see in later exercises.



The first batch of Termites do something else too. Can the group explain what? These termites are no different to the others but they create the original woodchips. Note how they are created – a patch is stamped yellow and its height raised by 1. They repeat this action 10 times, so will end standing on their final woodchip. The creation of woodchips should give an insight into how they can be 'picked up' and 'put down'. Can anyone suggest a way?



Now that we understand how the world is Setup, let's see if we can implement a solution. Remind everyone to tackle it in stages and test each stage. Random movement first, then pick up and put down.

There may be many solutions, but two key things are essential. Do not reveal these in advance!

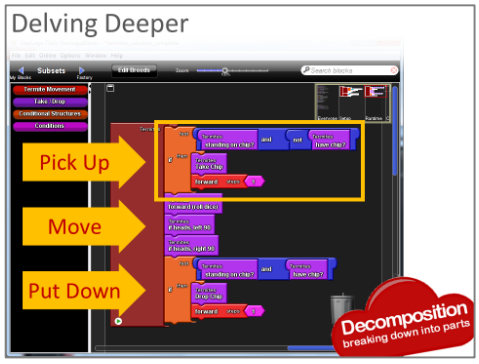
- There must be a command for dropping a chip onto another chip: "If standing on a chip? AND carrying a chip? Then drop chip.
- There must also be at least a forward or back command in between picking up and dropping a chip onto another chip. Without this a termite will just pick up a chip from a stack and drop it in the same stack again.

For students who finish quickly get them to consider the following extension. Although we are dealing with 'ideas' models, most Termite mounds are not a single tower. The picture shows a 'Cathedral Mound' with several spires. Can students modify their model so some Cathedral Mounds emerge? (A fairly simple modification dropping blocks next to ones they sense should result in piles being created, rather than towers. Some piles may well develop spires. You may need to alter the number of woodchips available at setup.)

Evaluating Solutions

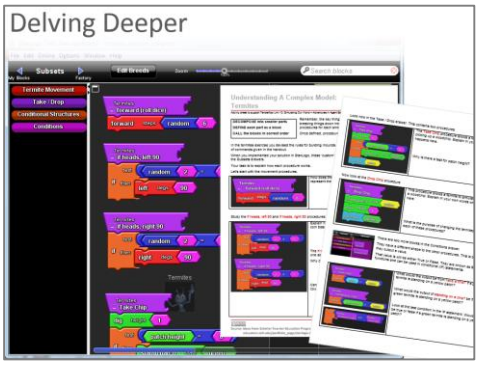
Heed the warning – answers on the next slide … and click to continue! When children have completed the task it is often worth reviewing and sharing different solutions. The slide shows one possible complete solution. It is included in the sample programs. Two partial solutions, showing just the movement, and the pick up, set down are also included.

Notice how easy it is to read. There are 3 clear sections. Well structured code is important so others can read it.



What else makes it easy to understand? The names of the procedures make the code clear. Take the pick up routine as an example. Ask a student to explain what it means – they should agree it is easy.

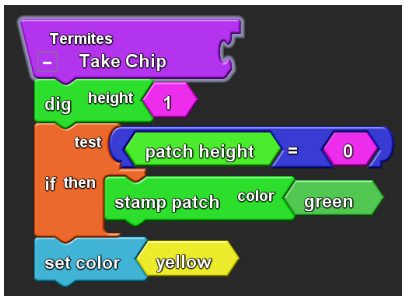
But where did these procedures come from? The next slide reveals the answer. They are procedures in their respective Subsets drawers.



By creating sensibly named procedures we remove the detail of how they work from the main code. We make it easy to read and understand. But where are these procedures? They are not part of the standard set of blocks provided in StarLogo. They were 'user defined', just like the ones we developed in Hungry Turtles. Notice how we have collapsed the Termites area so they aren't obvious. Once the space is restored all the procedure definitions can be viewed. Explaining them can be done as a group activity, as before, or as a written task. The investigation tasksheet is included.

For more able children, another partial solution is included in the resources. This has a correctly defined forever loop, but several of the procedures have been left undefined. Defining these can be quite challenging, provided they don't have their original file to refer to.

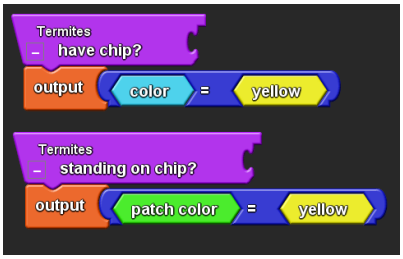
Before we finish with Termites it is worth looking at a couple of the procedures covered in the tasksheet to highlight a couple more points.



Here is the Take Chip procedure. Children need to be aware that picking up and setting down chips is an illusion. It is achieved by increasing or decreasing the height of a terrain patch. In this case, when a chip is 'taken', the height of the stack is decreased through the built in dig command.

Two questions to ask to ensure understanding.

- Why is there a test for patch height? (A termite can take a chip from any stack. If there are other chips underneath, the patch needs to remain yellow. If there are no more chips, the patch returns to green)
- Why is the termite's colour changed? This is used to indicate whether a termite is carrying a chip. We see how it is used on the next slide.



The blocks in the Conditions drawer, 'standing on chip?' and 'have chip?' are a different shape to the other procedure blocks. You see this in the code snippet (right). Look at their definitions (left). How do they differ from other procedures? They have an output command in the definition, followed by a test for equality. Each will output True or False.

They are, in effect, Boolean functions, which can then be used in conditional statement tests, like the one displayed. Although StarLogo doesn't use the term functions, a procedure that returns a value will have a shape that can be used in logical and arithmetic operations. By contrast, a procedure that returns nothing has a shape that allows it to be added to other sequences of commands.

Developing Pedagogy

When introducing StarLogo we mentioned its similarity to Scratch. That isn't surprising as they were both developed by Mitch Resnick. StarLogoTNG origins lie in an earlier project Mitch developed at MIT long before the development of block based languages. In 1994 he published 'Turtles, Termites and Traffic Jams', subtitled 'Explorations in Massively Parallel Microworlds' which was followed in 2001 by 'Adventures in Modeling'. Many of the ideas used in this Unit are based on this work. A short academic paper he co-wrote in 1997 with Uri Wilensky, 'Diving Into Complexity' looks at some of the role play activities we've tried. It is included in the resources and well worth reading before trying them in the classroom. Resnick's work has firm roots in educational pedagogy. His PhD was supervised by Seymour Papert, the creator of Logo. Papert coined the term 'constructionism' to suggest an approach to learning built on the constructivist ideas of his friend Jean Piaget. In essence, children construct knowledge through their own activities, developing an understanding through 'doing'. Papert focused on the learning potential involved in children creating and building things. The research group at MIT's Media Lab in which Papert and Resnick collaborated were responsible for developing the first programmable bricks that went on to become the hugely successful Lego Mindstorms.

The emphasis on pedagogy was aptly summarised by Alan Kay, another close collaborator with an impressive track record in promoting educational computing. Kay designed the first laptop and developed Squeak, another block based language. Children learn by doing, and most importantly by doing it wrong. The director of the MIT Media Lab, Nicholas Negroponte points out the broader value of this approach. Constructing programs allows children to think about how a computer 'thinks'. And when those programs fail, children have to develop a mental toolkit to reason systematically about what they observe. Engaging in model building is a powerful tool for developing more generic problem solving. At its heart is fostering Computational Thinking.