

LCSI



MicroWorlds, Computational Thinking, and 21st Century Learning

LCSI White Paper

Author: Susan Einhorn
© LCSI, 2012

MicroWorlds, Computational Thinking, and 21st Century Learning

“Understanding procedures and processes is important in math. There’s a fantastic way to do that – it’s called programming.”(Conrad Wolfram: *Teaching kids real math with computers*,

http://www.ted.com/talks/conrad_wolfram_teaching_kids_real_math_with_computers.html)

Human/computer Mutualism

As we have changed technology, technology has also changed us – especially in how we think about thinking and seek new ways to solve the many questions and problems we face. Technology has radically enhanced communication and global collaboration and made it easier to carry out vast numbers of complex, yet routine calculations. It has, also, at a different level, and maybe even more importantly, provided us with a medium in which to develop new patterns of thinking. As scientists, whether in the arena of physical, health, or social sciences, are influenced by computer science, they have gained new perspectives on how to approach old and new problems and innovation in research design and interpretation.

Computational Thinking – An Essential Skill for the 21st Century

Computers have freed us from the onerous and sometimes impossible task of running long, complex calculations, the type often required in research, so that researchers now more easily can focus on the big ideas and patterns that emerge. In thinking as a computer scientist, researchers become aware of behaviors and reactions that can be captured in algorithms or can be analyzed within an algorithmic framework.

Computational thinking now gives them a different framework for visualizing and analyzing – a whole new perspective. To rephrase a common idiom, “Until you have a screwdriver, everything looks like a nail.”

Computational thinking develops a variety of skills (logic, creativity, algorithmic thinking, modeling/simulations), involves the use of scientific methodologies, and helps develop both inventiveness and innovative thinking. It has roots in mathematics, engineering, technology, and science, and in the synthesis of ideas from all these fields, has created a way of thinking that is only just beginning to generate enormous changes and benefits.

Thinking about Thinking through Programming

Just using computers does not necessarily lead to the development of computational thinking. Facebook, Twitter, Flickr, Google, while all great applications, do not require or involve the same skills. Computational thinking is a learned approach and there’s no better way to learn it than through programming. Programming employs all the

components of computational thinking and the knowledge gained through the experience of tackling programming challenges – both explicit and tacit - can provide a framework not only for computer science, but for any field from natural and health sciences, to the social sciences and humanities.

So, here we have an important, essential and very truly 21st century “skill”- computational thinking - that is best learned through experience, interactions, actively doing. It allows students who learn to express themselves through programming (and who have the time to gain this knowledge) to not only answer questions but also generate new ones as they begin to view these challenges through the lens of the tacit knowledge intrinsic to computational thinking.

A student, when using programming to tackle a question, has to develop a hypothesis as to how best to solve or answer it, then build, through analysis of the problem, a set of rules (an algorithm) that can be used to test the hypothesis, after which she can review the results (data), and revise the solution. The art of programming requires creativity and inventiveness, logic, algorithmic thinking, and an appreciation of the recursive nature of this process, as the student learns from her failures, refines her work, and gets a deeper understanding of the problem. As with any creation, even once a solution is found – a pattern, an algorithm – the solution can be refined, simplified and beautified, made more elegant. In a way, programming provides the same satisfaction as a video game – the opportunity to find a path – one of many - through a problem. (It’s logical – video games are created by programmers!) The difference here is that students can answer their own questions and create their own challenges.

MicroWorlds JR

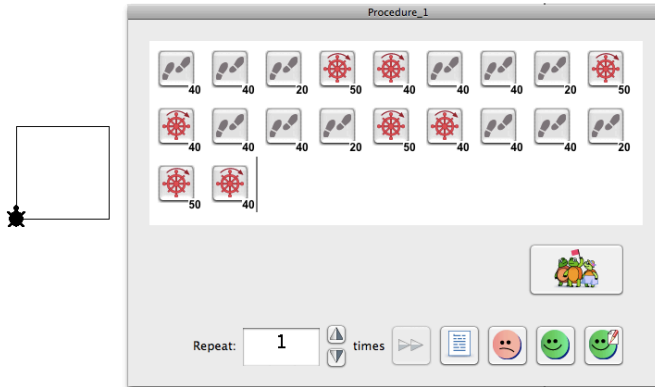
With products such as MicroWorlds JR and MicroWorlds EX, each with the Logo language at its core, children have the opportunity to develop their computational thinking abilities with age-appropriate tools. The approach to thinking that exploring with these tools helps develop can become a lens for how they understand and frame ideas and tackle challenges in all areas of the curriculum.

With MicroWorlds JR and its iconic commands, students experiment with mathematical ideas, for example, large and small numbers, angles, and geometric shapes, using the turtle as an object with which to explore. As they gain experience, they can begin to sequence instructions and see the outcomes, hypothesizing as to which sequence creates the result they want and then testing their ideas. It is through this sequence of actions – seeing a pattern, creating a rule (an algorithm) that describes that pattern and then testing to see if the logic is correct – repeated over time and in a playful, exploratory approach – that young learners begin to develop a new perspective on how to approach questions/challenges in other areas. This is particularly powerful if a teacher is there to coach them as they think about what they did to tackle a previous challenge and how that can be used to tackle the next in order to help them learn about their learning.

Children just beginning to use MicroWorlds JR test each command one at a time, outside a procedure, in order to see the effect each command has on the turtle. As children

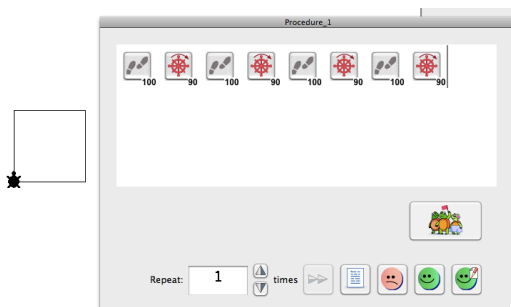
explore, they begin to understand how a series of actions can create a specific result. Click on an instruction, see a reaction. It's the core of data collection. Next, combine some instructions and see the result. This extends the exploration.

Once a child begins to realize she can not only put a group of instructions together, but she can save them together by creating a procedure, the results could look like this as she tries to capture how she drew a square:

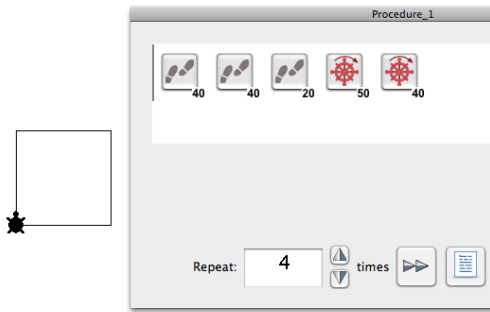


Notice that to get the distance she desired, she had the turtle go forward (footsteps icon) three times. To make a right angle (wheel icon), it took her two attempts. So, her list of instructions is long, but in the end she got the results she wanted.

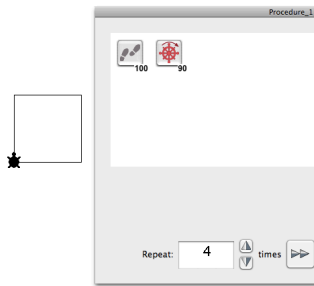
When encouraged to look at how to simplify this series of instructions, the child may realize that going forward first 40 steps, then 40 more, and finally 20 more is the same as going forward 100 steps all at once. Likewise, turning 50 degrees and then 40 degrees more is the same as turning 90 degrees all at once. She could edit her instructions to look like this:



Or, the student may notice that the forward instructions combined with the right turn instructions are repeated four times, a different type of pattern. She may, in turn, add a “repeat 4 times” option to the list of instructions.

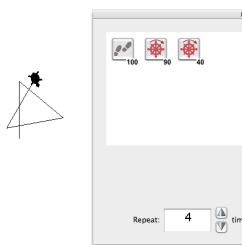


In coming to the most precise (elegant) version, she may recognize both patterns

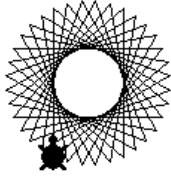


Each set of instructions is correct, and, more importantly, each set provides new opportunities to explore.

For example, what if the student added another instruction at the end of the list of instructions or left an extra instruction in by mistake?



Instead of this being an error, this is an opportunity to analyze what happened. The shape looks somewhat like the beginning of a star. Could it be that stars have similar patterns? What happens if these instructions are repeated?



The student may ask: What is this shape? (A torus) Why did this happen? (The turtle turned $90 + 40$ degrees, not just 90 degrees). How many times do I need to repeat these instructions to get back to the place where I started? How can I make this result – this drawing - have more/fewer points? A larger/smaller space in the center? In other words, she learns how to ask questions, how to look for patterns, how to test ideas, and how to create instructions (rules) that let her either repeat the same pattern or adapt it.

For beginning programmers, having a good coach who helps them look for patterns and try to figure out from the data (the drawing) why the resulting drawing occurred, is essential until the students begin to gain more experience in this way of thinking.

With MicroWorlds Jr, students can begin to transition to a textual programming language as they create procedures with command blocks and then click to see what the procedures are as text. This helps develop the vocabulary with which the student can talk about her ideas and instructions in order to verbalize rules and patterns and discuss and compare results.

MicroWorlds EX

Once students have transitioned to a textual version of Logo, they can begin to use MicroWorlds EX. As anyone who has used a computer during the last few decades realizes, there's no limit to what computer programmers can create – and MicroWorlds EX opens this door to all learners, no matter what their age.

For example, one aspect of computational thinking is an understanding and recognition of recursive patterns. Patterns are often repeated in mathematical solutions, nature, in other areas of science, and often these patterns are repeated with modifications – modifications that can be codified since they are based on some rule. Recursion is the process of describing an action in terms of itself. (This will become clearer below.) The more one plays around with the idea of recursion, the more one begins to recognize its presence in other disciplines.

The following procedure shows a simple example of recursion through programming. (Procedures extend the MicroWorlds EX vocabulary with words that you define yourself. A procedure is a group of instructions with a name that you assign to it. When you define a procedure, it becomes part of the MicroWorlds EX vocabulary for that project.)

to move	The word to and the procedure name begins every procedure definition.
forward 1	This is the first instruction. It tells the turtle to move forward one pixel.
move	The second instruction says run this procedure again.
end	All procedures end with the word end – letting MicroWorlds know this is the end of the procedure.

Each time the **move** procedure runs, the turtle moves forward one pixel (one “turtle step”) and then runs **move** (again), which tells MicroWorlds to move the turtle forward

one pixel and then run `move` (again), and so on. Notice that the built-in command “forward” requires an input to tell it how much the turtle should move forward.

This is recursion at its most basic level. By changing the procedure slightly, various effects can be observed.

```
to move :step      :step is a variable, standing in for a value.
forward :step     Now the turtle moves forward whatever value is provided for :step
move :step + 1    The next time move is run, :step will increase by 1
end
```

The `move` procedure now requires an input, just as `forward` requires an input. Now, to run the `move` procedure, an initial value for `:step` (which stands for ‘the variable named step’) needs to be added, so the instruction would be:

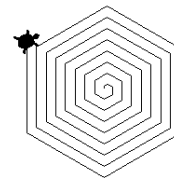
```
move 1
```

Now each time the `move` procedure runs, the turtle moves forward whatever the value of `:step` is. The first time, the turtle moves forward one pixel and then runs `move` (again), but now the value of `:step` increases by one. This time when the `move` procedure runs, MicroWorlds EX moves the turtle forward *two* pixels and then runs `move` (again) increasing `:step` by 1 again, and so on. In this version of `move`, the turtle doesn’t just continue to move forward, but it accelerates (eventually moving so fast that it becomes a mere blur on the screen).

Being able to express the rules of a recursive pattern through programming helps students better understand and recognize these patterns. Here is another example:

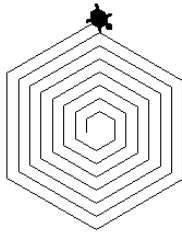
```
to spiral :step :angle
if :step = 100 [stop] This conditional statement says if :step equals 100, the procedure
                    stops. This prevents the turtle from spiraling forever.
forward :step
right :angle         Right tells the turtle to pivot a specific number of degrees.
spiral :step + 2 :angle
end
```

`spiral 2 60` would look like this when run:

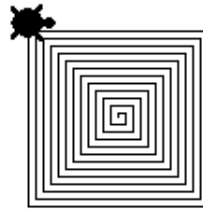


Programming provides an opportunity to play with pattern rules to see the effect of simple changes. For example:

Spiral 4 60 – the first side is 4 pixels long, but the spiral is very similar to the previous one.



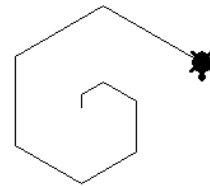
Spiral 2 90 - the angle is different...



Changing the procedure also creates different results:

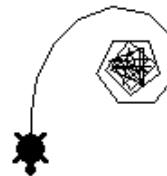
```
to spiral :step :angle
if :step = 100 [stop]
forward :step
right :angle
spiral :step + 10 :angle
end
```

Notice the change here.



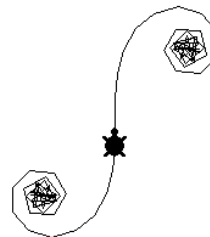
```
to spiral :step :angle
if :angle = 360 [stop]
forward :step
right :angle
spiral :step :angle + 5
end
```

...and here.

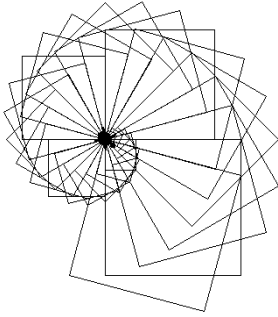


```
to spiral :step :angle
if :angle = 720 [stop]
forward :step
right :angle
spiral :step :angle + 5
end
```

...and here.



And with some more exploration...



Compare these two images:



Through an understanding and recognition of these patterns students will gain both experience and tacit knowledge of this form of patterning that may provide new ways to understand our world from the smallest forms of matter to the largest.

Transfer of Learning

The learning explorations made possible through the type of programming described above and the resulting development of computational thinking is of value to the learner in the immediate context, but if these skills and ways of interacting with the world transfer to other domains, the impact would be greatly amplified, leading to new patterns of thinking in different knowledge domains and an innovative, inventive perspective on finding new solutions to old problems.

The question one needs to ask is do these computational thinking skills transfer and positively impact learning in other domains. Anecdotal evidence of this transfer seems to indicate it does.

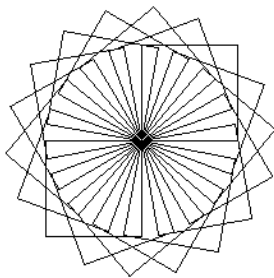


Figure 1. A subprocedure for creating a square is shown within a superprocedure for creating a flower. Procedures in MicroWorlds Ex can become part of larger procedures. The included ones are called subprocedures and the enclosing programs are called superprocedures.

```
to square  
repeat 4 [forward 50 right 90]  
end
```

```
to flower  
repeat 18 [square right 20]  
end
```

(Peter Skillen, *The Construction Zone*)

Peter Skillen, Manager, Social Media Professional Development with the YMCA of Greater Toronto, provides the following story to provide evidence of transfer in a blog post entitled *Deep Understanding & the Issue of Transfer* (<http://theconstructionzone.wordpress.com/2010/03/07/deep-understanding-the-issue-of-transfer/>):

Jeffrey, a Grade 2 student, made a most interesting leap from Logo to a completely different domain one day.

We were having a discussion inspired by the flight of the space shuttle piggybacked on a jumbo jet. Our Grades 2/3 class had the opportunity to watch the flight. When we returned to the classroom, a discussion of space naturally arose. One child asked if Earth was in space, and in asking the question, she determined it must be, because it wasn't sitting on anything. The discussion continued until Jeffrey piped up.

"You know . . . it's sort of like Logo."

We stopped and looked at him curiously. "What do you mean?" I asked him studiously.

He replied, "Well, Earth is like a procedure. It's like a subprocedure inside the solar system. The solar system is the superprocedure. And the solar system is like a subprocedure inside the universe. The universe is like the superprocedure."

"Fascinating," I said, then asked, "What's the biggest superprocedure?"

After a moment he replied, "I don't know. I guess the universe."

Peter continues, "I was truly amazed at the generalization across domains that Jeffrey had made. He clearly demonstrated significant transfer of a concept from his experiences with Logo to an authentic event. Although Jeffrey's illumination happened spontaneously, I learned that I could play an important role in helping students to acquire [Gavriel] Salomon's 'effects of' [technology] by providing opportunities for them to look for these comparisons across subject areas."

Opportunities to Explore

There are few opportunities in most school curricula to explore recursive patterns (although recursive patterns appear throughout nature and mathematics), develop and test algorithms, invent solutions to student-generated questions, or see their world through a different lens. The teacher/coach plays a key role in helping learners reflect on their thinking in order to bring about these lasting changes in metacognition. And, programming with products such as MicroWorlds EX and MicroWorlds JR create opportunities for even young children to explore big ideas as they develop true 21st century thinking skills.

"Computational thinking will be a fundamental skill used by everyone in the world. To reading, writing, and arithmetic, lets add computational thinking to every child's analytical ability." Jeannette Wing, Professor of Computer Science and Department Head, Computer Science Department, Carnegie Mellon University
<http://www.youtube.com/watch?v=C2Pq4N-iE4I>

"The role of the teacher is to create the conditions for invention rather than provide ready-made knowledge." Seymour Papert, professor emeritus, MIT/MIT Media Lab.