USING PROGRAMMING WITH RURAL CHILDREN FOR LEARNING TO THINK MATHEMATICALLY

Sanjeev Ranganathan[1], Bala Anand,
Sundranandhan Kothandaraman & Vaidegi Gunasekar

Aura Auro Design, SAIIER, Auroville (India)

[1]sanjeev@auraauro.com

Is it possible to use computer labs in a rural setting that encourages reasoning, visualization, abstraction in children (as envisioned in NCF 2005) while at the same time addressing curricular needs? This paper addresses the question through the use of programming in two rural schools including integration of curricular areas for fractions, cube roots, algebra, compound interest, data handling, geometry, etc. We explore three styles of instruction - projects for children to demonstrate their understanding, challenges to visualize abstract concepts, and games created by children themselves for mastery.

CONTEXT

We are presenting the work in two outreach schools of Auroville – Isai Ambalam School and Udavi School- that cater to the villages of Annainagar, Edyanchavadi, Irumbai, Pooturai, Pattannur, etc. We used **existing** computer facilities of the school, working with children on Mathematics through programming.

These are rural schools and have demographics similar to government and NGO run schools throughout the country. The schools believe in the holistic development of the child and school managements are progressive and encourage experimentation and research. From middle school onward much of the time is spent on academic subjects, in line with the expectation of parents. Both the schools have computer facilities. In this sense too they are typical, as over 75% of the secondary and higher secondary schools have computer facilities (DISE, 2014).

Aura Auro Design is a team of four engineers who volunteer 3 hours a day at the schools. We work with around 50 children from 6-8 grades along with their teachers. Classes range from 8 to 16 children. We are presenting the use of programming in this paper, however, we complement programming with puzzles, and strategy games, building physical models and TLM to create an environment of joyful mental exploration in the schools (Ranganathan, 2014).

Philosophies in Learning

Constructivist Education Theory (Bruner, 1960) indicates that knowledge is not delivered into the learner (whether child or adult) but recreated by the learner on his or her own. Children actively construct their knowledge by connecting new knowledge to what they already know. Constructivist education encourages discovery learning and learning by doing. It encourages activities that challenge a child's worldview since conceptual change and deeper conceptual learning come from experiential and interactive activities. Bruner further suggests “spiral learning”, claiming that any subject could be introduced to any child at any age if it is in some form that is honest.

In India, Sri Aurobindo (Aurobindo, 1910) indicates that nothing can be taught, but the teacher can support and encourage a child in the process of learning, thus guiding them towards perfection. More recently, Mukunda (2009) describes the three aspects of learning that are relevant to schools - conceptual knowledge, procedural knowledge and higher order reasoning. Conceptual knowledge (and change), she states, greatly benefit from constructivist approaches. In this paper we look at all three aspects of learning.

The Constructionism theory (Papert & Hare, 1991), adds to the constructivist theory the belief that children construct their own knowledge best by creating something outside their minds that is often sharable. In this research project, we explore creation in the virtual media through computer programming.

The National Curricular Framework 2005 (Pal et al., 2005) states that the 'useful' capabilities relating to numeracy, number operations, measurements, decimals and percentages are only a narrow goal of Mathematics education. Most middle schools across the country focus on these narrow goals for better marks in examinations. Examinations presently test children primarily on procedural learning: drill and rote learning become the primary tools for education. The NCF 2005 points out that across the country children do not enjoy Mathematics and are poor at applying these concepts or handling complexity.

The higher purpose of Mathematics, it says, is Mathematization: the understanding and application of mathematics in different situations with a focus on abstraction, patient problem solving and logical thinking. Meeting this goal requires a fundamental change in the approach used in schools. It requires classrooms to move away from simplistic 'sums' to more complex problem solving and contexts. It requires a shift in conversations in the classroom from the 'right answer' to considering and discovering approaches to problem solving. Our team thinks of Mathematics **in its wholeness,** achieving higher goals while also meeting the narrow goals. We also provide alternative ways for children to demonstrate their mastery of a subject beyond examinations.

We believe that the teacher is not an instructor or taskmaster, but a helper and a guide (Aurobindo, 1910). We believe that as teachers we should be aware of the situations when we need to step in (technical difficulties – mouth down frustration) and when we need to step back (struggle necessary for learning – mouth up frustration) (Martinez & Stager, 2013).

Programming and Children Learning

Use of programming to teach children Mathematics (Papert, 1986) happened before personal computing had reached its peak. A programming language LOGO was created to help children communicate with the computer and instruct a robotic “turtle” that could move and draw on paper. The positive effects of programming on children's cognitive learning were also examined (Pea & Kurland, 1984). A variety of hardware was made accessible to children to program including cars and robots (Lego Mindstorms) and resulted in the Maker movement.

The Maker movement focuses on learning through inventing: making, tinkering and engineering (Martinez & Stager, 2013). **Making** taps into the innate nature of human beings to create and its active role in learning requires visualization of a 'product'. **Tinkering** is a mindset – a playful approach to solving problems through experience, experimentation and discovery. **Engineering** is the process of extracting the principles from experience and organizing it to bridge intuition to formal learning, enabling better understanding and prediction of results. We find all three aspects necessary in our research for a more meaningful way of utilizing computers in school.

Programming Language and Setup

Scratch 2 (Resnick et al., 2009) is an advanced visual programming language built beyond the capabilities of LOGO. It has a low floor (easy to learn: you can stitch code together), high ceiling (includes variables, functions and event driven simulation) and broad walls (allows for users with different interests from drawing, music, animation, and computation). The availability of such a program at no cost enabled us to use it with rural children who have limited English skills to take up programming.

We used the Scratch 2 off-line editor to enable work without the internet. We installed public domain OS Ubuntu 14.04. The OS and software(s) are available free of cost and offline and the setup can be replicated in any computer center rural or urban across the country. We also set up a local LAN to allow centralized storage of files. This allowed children to save and continue their work from any machine. The complexity of the children's programs significantly increased when they were assured that their work was saved and available.

Educational Computation and Children Programming in India

In urban India there is a significant movement for children to learn programming beyond school. Among younger children Scratch is a popular program. Computing availability in rural India is limited and when computer facilities exist, they are rarely used beyond an hour or two in a day in a school.

Progressive schools do introduce children to Scratch, sometimes as a creative medium for animation and to develop higher level thinking. In Udavi School Scratch was already used by children. With minimal guidance they had played and tinkered with example games.

USING PROGRAMMING WITH MATHEMATICAL CONCEPTS

We present these case studies of different aspects of children's learning through programming.

Cubes and Cube Roots

Pooja (8th grade) encounters cubes and cube roots; she often mistakes x^3 (x multiplied by itself three times) with $3x$ (x multiplied by three). For perfect cubes (e.g. **830**58$\underline{4}$) one can guess their two-digit cube roots (e.g. **9**$\underline{4}$) by estimating how big a number is (how many 1000s) and looking at its unit digit. I hoped this would help her get a sense of numbers. She was unable to follow the procedure and had difficulty with the sense of numbers. (She is not alone).

She starts to program to find the cube roots of numbers. Her initial goal is to print the first 10 cubes. Power is not an available expression and she needs to construct the expression for a *number* (variable). In time she creates *result*=*number*×*number*×*number*. She increments the *number* each time and puts it in a loop. To view the results, I ask her to add a delay of 1 second after each operation. I ask her to notice the numbers, but she doesn't find a pattern. She then changes the loop condition to keep running till the *result* becomes a large cube given in her book. Now, she is interested in where the program stops and intently looks at the results. She soon figures out when she is too far and needs to wait for the result and asks to reduce the time between calculations. I tell her she could change the program for fewer calculations, but retain the time after a calculation to see the result.

She decides to skip numbers in her program. I connect it to the original process and ask her to change in steps of 10 (10, 20, 30, etc) till the *result* is too large, go to the previous step and then go in steps of 1. She implements this as two loops: first loop in steps of 10 and second loop in steps of 1. It takes her time and she makes errors, but she understands what she is trying to do and debugs it with known cube numbers and their cube roots.

To use the program she generates a random two-digit number (feature available in Scratch) and uses its cube as the *target* (variable). This time when she looks at the steps of 10 she notices the last three digits are 000s. She then notices that the non-zero digits have the same pattern as cubes of single-digit numbers. When the second loop starts the numbers are much more complicated. I ask her to focus on the units place. Now, she notices a pattern e.g. 1 in the units gives 1 in the units of the cube ($\mathbf{2}\underline{1}^3 = \mathbf{9}26\underline{1}$). Similarly, 4 gives 4, 5 gives 5, 6 gives 6, 9 gives 9. The others were flips 3 with 7 ($\mathbf{4}\underline{3}^3 = \mathbf{79}50\underline{7}$ and $\mathbf{5}\underline{7}^3=\mathbf{185}19\underline{3}$) and 2 with 8.

She starts working this out systematically as the computer does and now she gets it. In time she skips the 'unnecessary' steps and gets straight to the cube of the 10s before and (to check the 10s after) and then writes the number including the units. She makes the program a game that accepts inputs to check her answer. The program still works through all the steps for her to cross check her thinking and then announces if the result is accurate. In the next class she works out 50 cube roots in an hour in her notebook and checks with the computer. She gets one wrong and understands why that one confused her.

She now revisits squares wanting to do something similar there. Though the numbers are smaller the process is more involved as it has the mapping in the units place and is not one-to-one (e.g. units place 4 and 6 result in a square with units place 6) and you need to estimate which square it is closer to. However, she masters it by following a similar process.

In this example we see that it is possible to learn a higher order skill – sense of numbers, logically thinking with learning a procedural skill and developing an understanding of the concept. We also notice that in the process of creation the learning becomes her own.

Multiplication and Corresponding Division Stories

Much of science that children encounter in school is one quantity (distance) as a product of the two others (speed and time). Other examples are density, mass, volume; mass, force, acceleration; changing units. The simplest form of these boil down to:

Multiplication story: 1 box has 6 apples, how many apples are there in 4 boxes.

Division story 1: There are 24 apples in 4 boxes, how many apples are there in one box.
Division story 2: There are 6 apples in 1 box, how many such boxes are needed for 24 apples.

Rahul (6th grade) seemed to get the concept, but was unable to retain it. He started to create a program to animate his story. Scratch provides a stage and you need to bring in characters that do their part. He needed to think of a concrete example and stay with the example while he programmed the appropriate apples and boxes to appear and disappear based on his story. He needed to synchronize the timing of his voice and the corresponding display. The rigor of staying with a problem helped him to retain this concrete story. Knowing one concrete example well helped him abstract other stories. The process of personalization of learning through the process is so strong that at the end of the year when we displayed the work of children, not just Rahul, but every child could recognize their work by just the initial stage even before we started playing their demonstrations.

Circles

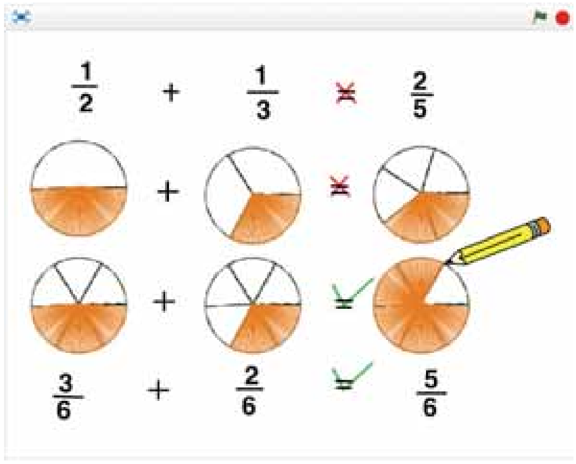
We planned to use the process of personalizing the projects for adding fractions as well. In visualizing fractions the children decided to use a circle to represent the whole (as it's obvious when something is missing). We added a constraint: children would need to instruct (program) the computer to draw rather than draw themselves by hand.

Scratch has bare bones pen commands allowing you to draw from point-to-point (lines with coordinates) or lines of arbitrary length at arbitrary angles from a point. How then do you get a circle (more accurately a good approximation of a circle) with lines?

With *making* of the fractions project in mind, the children started to *tinker* with various shapes with lines that would resemble a circle. This led to interesting conversations about what fundamentally a circle is, something that is constantly changing its angle. They then broke the full angle (360) into angle chunks, moving a constant distance and rotating by that angle chunk (*engineering*). Depending on the size of the chunk they got various regular polygons. Starting from equilateral triangles to squares to pentagons and so on finally settling at a shape with large enough sides to look like a circle. Eventually, a simple program *repeat 360 {move 1 step; rotate 1 degree}* gave them a very good approximation of a circle.

Fractions

Given that we had spent quite some time on this wonderful deviation we decided to experiment with a higher level thought process for playing with functions and created a base function for drawing a fraction with various inputs. The idea of controlling where the fraction should be drawn, and how big it should be was a significant exercise in their understanding of coordinate geometry. The last screen of such an animation by Ahalya (7th Grade is shown below).



$\frac{1}{2} + \frac{1}{3} \neq \frac{2}{5}$
$\frac{3}{6} + \frac{2}{6} = \frac{5}{6}$

Figure 1: Final image of one animation project on fraction additions

Children who participated in that session were required to think in a different way about circles and were asked to examine fractions in detail. A few months after making their animations the children were tested on adding fractions. All of them knew that you couldn't just add the numerators and denominators, most were able to create equivalent fractions to add fractions, and, on being prompted half were also able to resort to LCM to add fractions which is the expected procedure.

Percentages, Algebra, Compound Interest

The conceptual understanding of abstract concepts were better understood by children when they were asked to demonstrate their understanding visually. For percentages they created programs to make pie charts of the time spent on activities in a day. For this they took the hours and framed fractions of the day, scaled angles with different colors of the pie and also represented percentages. This helped them link multiple independently dealt with concepts at the same time. The process of debugging (fixing errors) helped them face the assumptions and misconceptions they had e.g. Jaya had assumed that now that she was working with percentages fractions needed to be scaled by 100 for everything including angles. Only part of her circle appears densely colored. She debugs her program and realizes that even though they are dealing with percentages it does not scale angles automatically from 360 degrees. This helped Jaya realize that what she knew before had not suddenly become irrelevant now.

Data Handling

The fundamental art in having to explain something to a computer is that children need to be clear about the procedure and break it down into simple steps they can code. Equally importantly, they need to create test cases that they are absolutely confident of. The meta-cognition of average children in middle school is low and being able to say they are absolutely sure of something, especially a new concept, gets them to stretch beyond their comfort zone. They start trying to understand the concept to come up with a simple test case e.g. one group of children were trying to get an average in their book exactly right, in another group a child started a discussion on what would happen if all the data was the same since this was easy to generate on the computer, indeed this resulted in a much more trivial test case, but more involved understanding of the concept. As before, plotting the results helped them estimate averages even before calculating them. Further, Scratch's ability to create large random data helped them notice the law of large numbers by themselves i.e. average of randomly distributed data between two numbers tends to be the average of the two numbers.

Observations

As children explain a procedure step-by-step through programming it helps clarify these procedures in their minds. Further, it fundamentally changes their relationship with computers. In the year-end surveys one of the children remarked that only now she understood how much effort and intelligence it goes into making the computer *look* smart. Other than the appreciation of what it takes to do something in real life, the import was that she no longer considered the computer smart *in itself.* Computer usage in schools is generally an extension of an authority figure which is always correct e.g. softwares that explain something to children or test them (openly or through 'games'). In these cases, the computer is all knowing and always right and children have nothing to offer to it but 'right' answers. It provides no scope for invention, questioning or possibility of higher order learning.

MAKING, TINKERING, ENGINEERING

Conversations which lead to learning in deeper mathematics can come up in many making situations. As part of the study on the moon the children in 7th grade decided to make a time-scaled animation of the earth spinning on its axis, going around the sun with the moon rotating around the earth. They also decided that they wanted earth to travel in an ellipse rather than a circle. This started an exploration of how to get an ellipse rather than a circle.

One solution took them through attempting to extract this information by solving the equation that Geogebra gave. The idea that a two dimensional expression e.g. $1.37x^2+2.25y^2-61798.1$ that they did not know how to process could be solved by the computer by selecting a value of y and sweeping x till the expression became zero was fascinating for them. Since they were drawing pixels they only needed the integer part of the solutions, but a curve does not only have integer solutions. They needed to use the fact that expressions change sign when it steps over the solution (expression changes sign as it crosses zero). They developed a healthy respect for integer multiplication when they realized that this could use the product of two successive results to locate a solution (*engineering*).

The children *tinkered* around with the limits of using these sweeps and realized that in closed curves there are no solutions beyond a *y* value point for any value of *x*. The earth was now rotating around the Sun. For a sum that is generally simple getting an answer is enough, however, in making you can gauge the quality of work and there is always room for progress.

The children noticed that the earth seemed to be speeding up the upper extremes (y was being stepped linearly and the slope was close to zero). They further tinkered and made multiple ranges putting points closer together as they came to the extreme to compensate for change of slope.

Another group of children simply tinkered around longer to come up with a different solution by visualizing the circle as getting stretched in the center to create an ellipse. They divided the angles between a smaller and bigger circle to achieve the same result. The code went something like this *repeat 45 {move 1, rotate 1}, repeat 90 {move 2, rotate 1}, repeat 45 {move 1, rotate 1},* and so on for the other half of the ellipse.

DIFFICULTIES AND ASSESSMENTS

Most children we worked with had already been using the computer to play (educational) games, watch videos or work with Paint. Their initial excitement for using the computer had died out and there was resistance to intellectual work using the computer. What helped us was that children still enjoyed making and tinkering (as suggested by their surveys). The drift to engineering depends on the ability of the instructors to create a situation where children want to create a project and struggle to find a solution with tinkering and want more predictability in their work e.g. making a circle can be accomplished through tinkering, but making a set of specific circles needs understanding about perimeter and radius of circles.

Initially, the time projects took concerned us. We gave it time since the children were engaged and challenged. Teachers spend a lot of time repeating concepts that the children apparently learnt and have 'forgotten', when in fact, the children had not really learnt it (Brooks & Brooks, 1993). We found that children retained concepts they learnt over time.

The biggest difficulty is actually that we, as teachers, want to teach (and instruct) but that often steals the most important learning from the child. In time we learnt to notice when we were too keen to teach and learnt to step back to allow the children to struggle and learn on their own.

One group of girls had created a wind chime, calculated 22.5% lengths of the pipes, hacksawed the steel pipes, used a power drill to drill holes and complete their product. When we did surveys with the children at the end of the year we had expected this to be the top of their list of accomplishments. It was hence a surprise when Ahalya indicated that the fractions program (Fig. 1) was her finest work. I reminded her of the wind chime, she smiled and said that it was a great experience, but the fractions were her best work. It helped me realize that in this increasingly technological world children see the virtual world as something tangible and real.

CONCLUSIONS

Programming allows an important interaction between a child and a computer altering fundamentally the equation for being a user to being a programmer, from being a receiver to being a creator. Programming is significantly different from using passive media that converts the computer to a personalized television, or enables children to play games on the computer as users only. This fosters the assumption that the computer is always right and we/they are always playing catch-up. We must let children program the computer instead of attempting to program the children through computers (Papert, 1993).

Making projects (through programming) can be a way for children to demonstrate their learning and offer alternatives to examinations as the only form of assessment. This also offers an opportunity for self-evaluation and constant progress.

Programming a computer helps children learn conceptual ideas because they need to break it down into small bites for a computer to follow. It also helps them visualize abstract concepts. They can also create their own games to develop rigor.

Rural children are growing up with an increasing access to technology and programming can be meaningfully used with them to support their higher order learning and mathematical thinking while addressing curricular aspects in a more meaningful way. This paper is a case study intended to show that the 330,000 schools across the country that have computer resources could use this resource in a creative way to develop higher order reasoning skills through discovery and invention while addressing useful academic skills.

Acknowledgements

We would like to thank Aura Semiconductor Pvt. Ltd., Udavi School and Isai Ambalam School for their support and the children for their desire to learn.

References

Aurobindo, S. (1910). *The human mind.* India: Karmayogin.

Brooks, J. G., & Brooks, M. G. (1993) *The case for constructivist classrooms*. Alexandria, Virginia: Association for supervision of curriculum development.

Bruner, J. S. (1960). *The process of education*. Cambridge: Harvard University Press.

District Information System for Education (DISE). (2014). Elementary education in India: Where are we?. Retrieved from http://dise.in/Downloads/Elementary-STRC-2013-14/All-India.pdf

Martinez, S. L., & Stager, G. (2013). *Invent to learn: Making, tinkering, and engineering in the classroom.* CA: Constructing Modern Knowledge Press.

Mukunda, K. V. (2009) *What did you ask at school today?* New Delhi: Harper Collins.

Pal, Y., et al. (2005). *National Curricular Framework.* New Delhi: National Council of Educational Research and Training. Retrieved from http://www.ncert.nic.in/rightside/links/nc_framework.html

Papert, S. (1986). *Constructionism: A new opportunity for elementary science education.* Cambridge, MA: M.I.T Media Laboratory, Epistemology and Learning Group (NSF Grant Proposal).

Papert, S. (1993). *Mindstorms: Children, computers and powerful ideas.* NY: Basic Books.

Papert, S., & Harel, I. (1991). *Constructionism.* NJ: Ablex Publishing Corporation.

Pea, R., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2*, 137-168.

Resnick, M. et al. (2009). Scratch: Programming for all. *Communications of the ACM, 51*(11), 60-67.

Ranganathan, S. (2014) *Program to encourage critical thinking in children – 2013-2014*. Grant report by Udavi School to SAIIER.