MAKING EXPLORATION VISIBLE: ON SOFTWARE DESIGN AND SCHOOL MATHEMATICS REFORM

Michal Yerushalmy, University of Haifa, Michalyr@construct.haifa.ac.il

The writing of this paper was supported by a grant from MISES:
The Institute for the Study of Educational Systems, Milken Institute, Jerusalem.

There is plenty of software in use today in the secondary school math curriculum. It runs on computers or on advanced calculators. Introducing software for exploration in the secondary math class does require a radical change in the curricular agenda because it allows and promotes new organization of both content and learning styles. In fact, software that meant to introduce new ways of building and reflecting upon knowledge was indeed the essence of the word "microworld" when first introduced (Hoyles 1993). The bulk of the currently used software is different from the educational software tools implemented a few years ago. While the previous generation of educational software that was mostly inspired from Papert's visions of use of technology (the Geometric Supposer, Function Prob, Cabri and other educational applications of the 80s) were developed as tools for exploration within school curricula, many of the current tools at the secondary school were originally designed for uses outside the classroom—tools for "solving",[1] such as symbolic manipulators, graph plotters, and electronic spreadsheets. This pragmatic trend can be explained by both economic (the very limited sources for developing educational software is a major problem) and educational factors. The question to be explored here is to what extent can technology design act as a driving force in supporting the reform in the math curriculum. Can software that emphasizes ideas as mathematical objects to be manipulated, that unveils common procedures by providing tools for explanations, and that makes explicit the mapping between objects and representations support exploration differently than "solution software" does?

There is encouraging evidence about the impact various specific software capabilities have on exploration (Yerushalmy 1997). There is also discouraging evidence about work with educational software that does not always act as the idea generator it was designed to be (Balacheff, 1997). There is no universal measure to evaluate design considerations but there seems to be a growing agreement on the need for an open conversation about tool design and designers' intentions (Kynigos et al. 1997, Collins 1996). Such discussions can help designers to realize and articulate what are perhaps unconscious decisions and make them conscious design considerations. These discussions may also help teachers focus on the finer properties and messages of the tools they use in the classroom. If a software tool is going to play a major role in a teacher's agenda then he/she should understand the tool, not just in terms of its functionality but also in terms of the invisible reasoning that makes the tool what it is.

Visible "Exploration" Software Carries a Clear Statement about Methods of Exploration

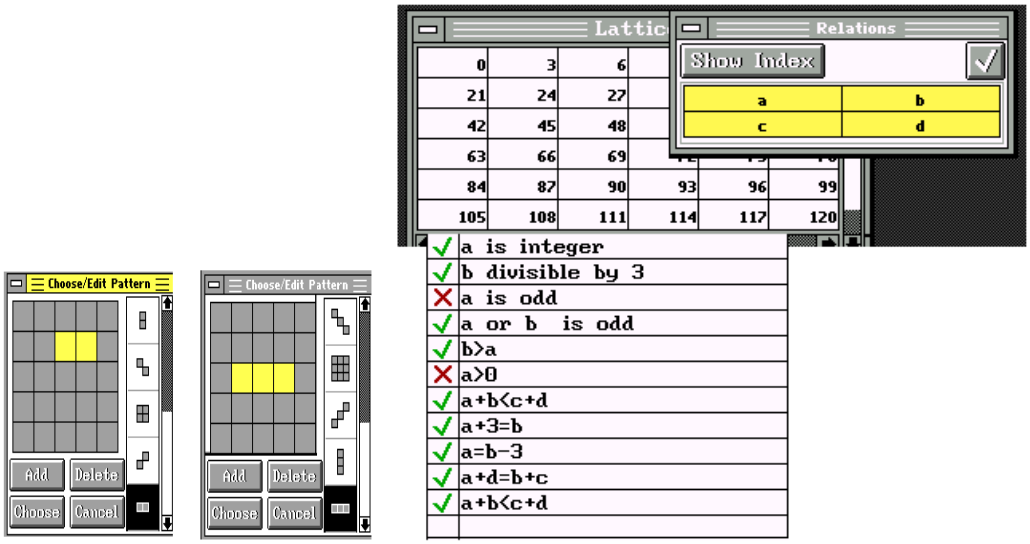
The current reform in mathematics education centers on school as a place that provides opportunities for students to construct knowledge and think mathematically through exploration. In the general tradition and habits of schools, students are adapted not to seek understanding (Scardamalia and Bereiter 1996) and reform therefore requires intentional effort to ensure that "understanding" is what counts in school. How could the design of the tool increase the chances for exploration in a "standard" math class buried under the agenda of having an immediate product, to present a result, to solve, to memorize a procedure?

"Solution" software obviously supports large collections of solved examples. But, for this collection of examples to grow into an idea, complex cognitive processes that do not emerge spontaneously from creating and observing examples are required. In order to support the transition from examples into concepts exploration software should act as a thought organizer. Organizer in the sense of using a limited collection of important terms: mathematical objects and actions. Organizer in the sense of providing tools and options to vary objects systematically; in the sense of providing meaningful-links among environments and representations and in the sense of supporting consistent use of language in the formation of conjectures. These organizational principles, when built to be visible to the user, have the potential to become mathematical thinking tools for planning and for problem posing. Schwartz (1995) suggests that tools for learning should be designed to allow the learner to jump into what he calls the "interesting middle." To do that, the building blocks (the major options offered by the tool) should be mathematical objects and processes that are primitive enough to allow construction of new objects by the given processes, but interesting enough to promote uses of higher-order mathematical language, argumentation, and proof. Here are examples from exploration software that provides tools to sample examples, to script conjectures and generalizations using procedures written in different linguistic dialects and to generate new examples using these conjectures. Both examples deal with local and global descriptions of patterns in early algebra and in advanced calculus.

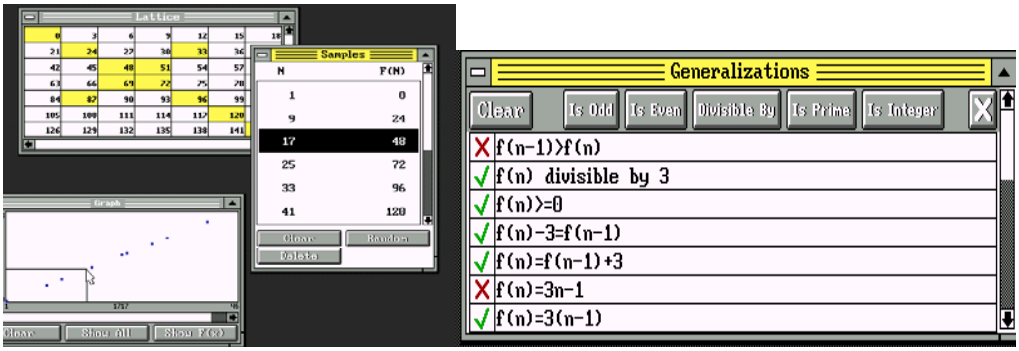
[1] "Solving" means here tools created to **use** mathematics -- to provide a **result** as an output of a commonly used procedure: the solution of an equation, the graph of a set of numerical data, etc.

The study of patterns of numbers is now widely considered to be a pre-algebra activity that provides an arena for exploration, conjectures and symbolization. We were bothered by the obvious gap between intuitive views of numerical properties and the ability to use language to formulate these visible phenomena. With the intention of bridging this gap, we developed a software environment, ”Algebraic Patterns” (Yerushalmy & Shternberg 1995), that supports exploration and generalization of patterns. We were interested in using technology as a means of creating a microworld that will support generation of ideas about patterns in number lattices and will provide ways of communication about these generalization using various modes of symbolization. Forming a global rule for a given lattice requires a single specific answer (often embedded in an activity of the type “guess my rule”) and is therefore a more demanding algebraic activity. Forming local statements requires flexibility and creativity in observing relations within the lattice and also requires ways of communication about numbers. Consequently, a microworld that supports both local and global observations on patterns has a chance to create not just various modes of symbolization but to be an arena for considerations of equivalency, proofs and argumentation.

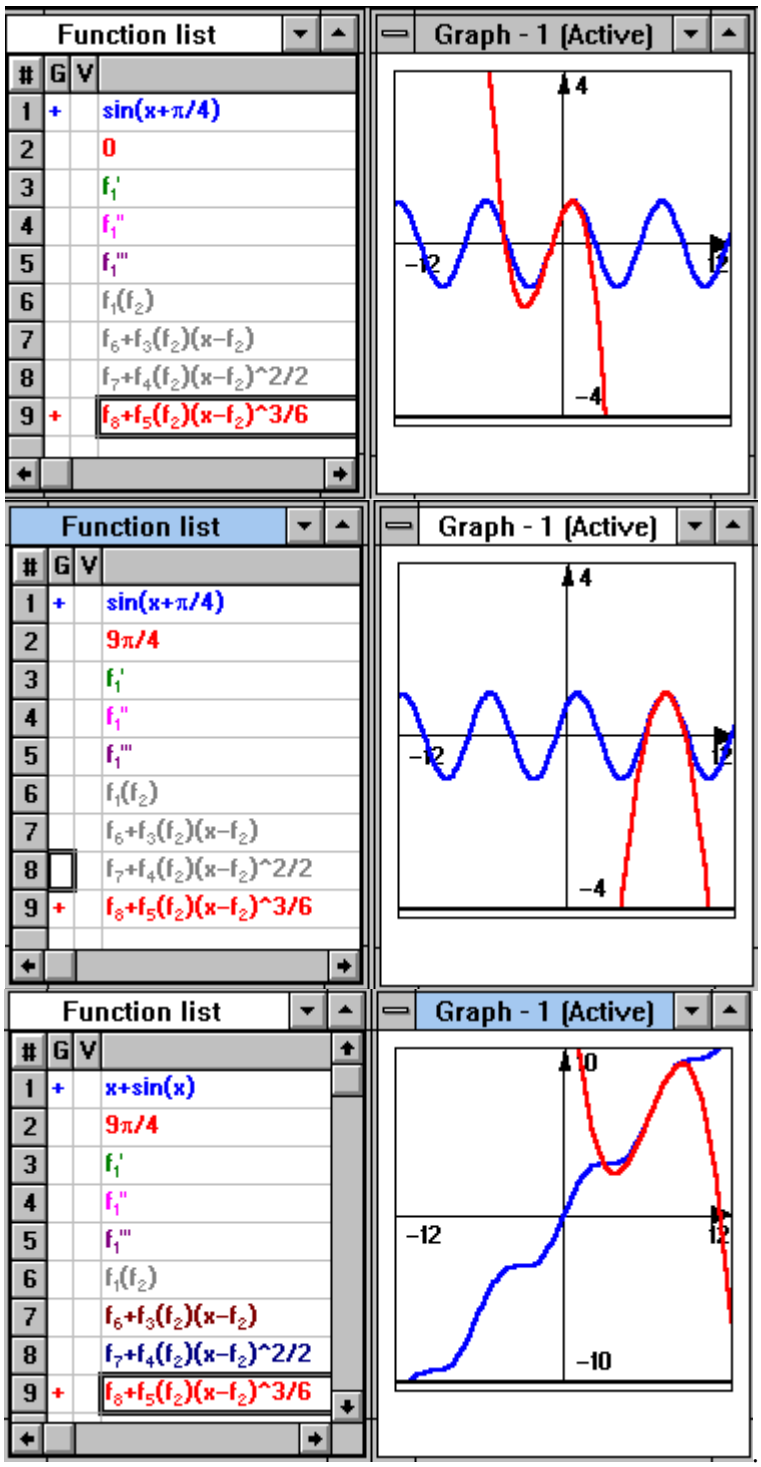
The pattern stamps editor (a drawing tool on a grid that serves to define local behavior that can be tested along the pattern) is considered to be a key element in turning the rote work of rule guessing into a serious and vivid inquiry of local relations.



In order to describe the same pattern globally one requires other tools (such as table of values or a graph) and different linguistic dialect (in the example below we used the function on integer syntax):



Any of the statements can later on be tested with various lattices – thus it turn to be a generalization that on one hand summarizes and concludes from samples of examples but is also a procedure which acts on lattices as its independent variable. This same idea of shuttling between the local view and the global view of a phenomenon is one of the powerful ideas of the connections between calculus and algebra. Using calculus to build mathematical models requires the writing of procedure on/with functions that relate quantities and their rates of change in a small region of their variation (these procedures are often stated as differential expressions or equations). In contrast, an algebraic model requires to understand directly at the outset how a quantity varies over its full range of variation. It is often far simpler conceptually to build the kind of local model that the calculus permits. Using the language of the calculus, especially when cast in the form of procedures on functions allows us to express the essence of the phenomenon being modeled without requiring us, at first, to be overly specific about details.



Above is an example from Calculus UnLimited (Schwartz, Yerushalmy 1995) which presents a procedure operates on variables of the Taylor expansion Watch the differences between the first 2 figures (same function different point) and between the second and the third figures (different function same point). As in the early algebra patterns where the statements are tested on various lattices and patterns, here as well the procedure takes both the function and the point as its independent variables.

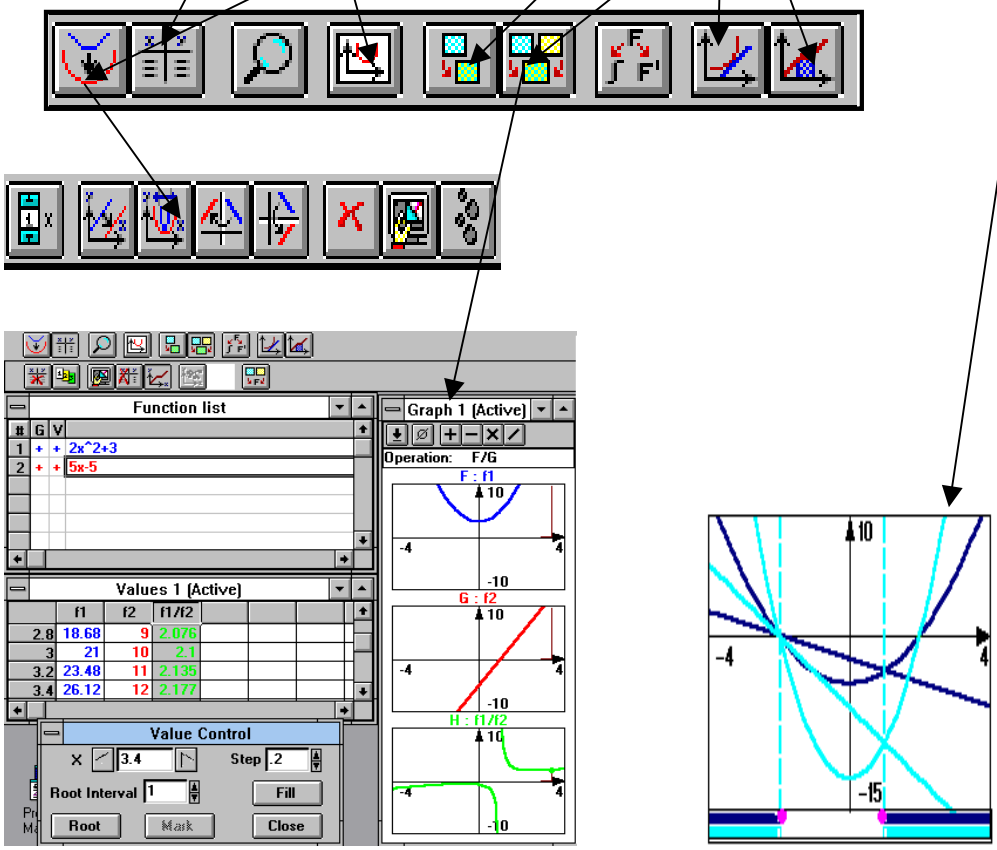
Thus, using the mathematical big idea as the a design principle that applies along what seemed to be very different parts of the curriculum provides the learner control over mathematical explorations.

Visible "Exploration" Software Carries a Message about the Organization of the Curriculum

A common concern among constructivist attempts to reform curriculum is the problem of organization. If curriculum is to rely on inquiry, problem posing, and the construction of knowledge rather than on acquiring a sequence of procedures, then curriculum developers, teachers, and students need to form a clear message about its organization. If

a teacher is trying to honor and respect students as thinkers, then creating bridges between the curricular content to the child is a heavy responsibility. New navigation ideas are needed in order to shift from mapping the content along practiced sequences of computations into relations among central objects and actions of the mathematics. Making such new maps and navigation mechanisms visible to the students is one of the hardest challenges teachers are facing. Indeed, this was an essential part of the difficulty and risk described by Chazan (1999) about his own teaching experiment of algebra to all students. The Visual Math 7-12 algebra and Calculus curriculum (CET 1995) is organized around objects (types of functions) and actions (on and with functions). The terms used to organize the curriculum units also formed the building blocks and the actions embedded within the Calculus UnLimited environment (ibid.). In the following 2D map the rows are mathematical objects (functions), the columns are types of actions on these objects and each cell in this matrix is a unit in the algebra curriculum. The actions are also the main components of the software (middle) and can be tangibly worked out on the objects (below).

	Represe ntation	**re-Repres entation**	**Families** **Transfor- mations**	**Unary operat -ions**	**Binary operat -ions**	**Com paris ons**	**Equivalent comparisons**
Linear							
Polynomial							
Periodic							
Exponential							



The new curriculum involves algebra, pre-calculus and calculus activities, teachers' materials and alternative assessment with and without technology. It attempts to build relational understanding by turning the "units" in the curricular sequence from being chapters in the textbook into thinking tools that can be employed using software.

Visible "Exploration" Software should present a clear Message about the Role of Technology

Here is a recent description of the current situation by Demanna (1998): "In our opinion it is not the goal of the reform effort to abandon algebraic or analytic techniques. Yet teachers sometimes give this impression or mistakenly believe that this is true" (p. 5). Why does this description so often match the impression educators and parents have about current algebra and calculus reform? Why do so many people misunderstand curriculum reform with respect to the use of computer algebra systems? Why is there still so much confusion about the role of the four-operation

calculator in teaching arithmetic? “There has been and continues to be concern, confusion and resistance over the use of hand held calculators, even some 15 years after their introduction to primary schools” (Pimm 1995, p. 82).

At least one major reason is that some very strange messages are being delivered to schools about the role of the technology in the curriculum. The major message of exploration software is that syntactic computation discourse is not the agenda. The design of CAS (Computer Algebra System) for example often delivers the opposite message (Yerushalmy 1999). Software for explorations does not try to make complex algorithms easy nor does it manipulate using sophisticated procedures. It does not make solution the central feature and it often provides tools that are explicitly trivial for professionals. Making this agenda visible should be in my view the major design goal of educational software. Solutions tools are (or soon will be) common among the adult community that uses mathematics to solve problems outside mathematics. Software that reflects the intentions of mathematics educators to educate for mathematical habits of work and mind cannot in most cases be “off-the-shelf.” Rather, it is an artifact, prepared specifically to make exploration and the construction of mathematical knowledge at any level, a habit of life.

References

Balacheff, N. (1997). Some questions on mathematical learning environments. In E. Pehkonen (Ed.), The Proceedings of the 21st Conference of the Psychology of Mathematics Education, Volume 1. Lahti, Finland, 99-104.

Chazan, D. (1999) On teachers mathematical knowledge and student exploration: A personal story about teaching a technologically supported approach to school algebra. International Journal of Computers in Mathematical Education.

Collins, A. (1996). Design issues for learning environments. In S. Vosniadou, E. de Corte, R. Glaser, H. Mandel (Eds.) International Perspectives on the Design of Technology-Supported Learning Environments. (pp.347-62). Hillsdale, NJ: Erlbaum.

Demana, F. (1998). Improving student understanding about computational processes, problem solving techniques and proof using hand held technology. In W. Yang, K. Shirayanagi, S. Chu, and G. FitzGerald (Eds.) Proceedings of the 3rd Asian Technology Conference in Mathematics. (pp.3-12). Stukuba, Japan: Springer.

Hoyles, C. (1993). Microworlds/Schoolworlds: The transformation of an innovation. In C. Keitel, & K. Ruthven (Eds.), Learning from Computers: Mathematics Education and Technology, (pp.1-17), Springer-Verlag, Berlin Heidelberg.

Kynigos, C., Koutlis, S., Hadzilacos, T. (1997) Mathematics with component oriented exploratory software. International Journal of Computers for Mathematical Learning. 2(3), 229-250.

Pimm, D. (1995). Symbols and Meanings in School Mathematics. London: Routledge.

Scardamalia, M. and C. Bereiter. (1996). Adaptation and understanding: A case for new cultures in schooling. In S. Vosniadou, E. de Corte, R. Glaser, and H. Mandel (Eds.) International Perspectives on the Design of Technology-Supported Learning Environments. (pp.149-64). Hillsdale, NJ: Erlbaum.

Schwartz, J. L. (1995). The right size byte: Reflections on educational software designer. In: D. Perkins, J. Schwartz, M. West, and S. Wiske, (Eds.) Software Goes to School. (pp.172-82). New York: Oxford University Press.

Schwartz, J.L., Yerushalmy, M. (1999) The Geometric Supposer. Computer software. Center for Educational Technology, Tel-Aviv. http://www.cet.ac.il/visual-math

Schwartz, J.L., Yerushalmy, M. (1996) Calculus UnLimited. Computer software. Center for Educational Technology, Tel-Aviv, http://www.visual-math.com/

Yerushalmy, M., Shternberg, B. (1995) Algebraic Patterns. Computer software. Center for Educational Technology, Tel-Aviv. http://www.cet.ac.il/visual-math

Yerushalmy, M. (1999) Making Exploration Visible: On Software Design and School Algebra Curriculum. International Journal of Computers in Mathematical Education.