Designing a learnable mathematics: a fundamental role for the computer?

Richard Noss
London Knowledge Lab
Institute of Education, University of London, UK [1]

I take as a starting point two assumptions: that there is a rising demand on individuals and communities to understand something of how social and technical systems operate; and that increasingly, it is becoming more and more difficult to catch sight of how things work, as the mechanisms that drive them become ever more hidden in invisible computer code. Put simply, competence in constructing, interpreting and critiquing mathematical models has become a core part of social and professional life in the twenty-first century, but it is becoming more and more difficult to achieve that competence.

The implications for the design of learning systems are manifold. I will focus here on just three. First, we should try to design learning environments where people can make models of things, either physically or – my focus in this paper – virtually. Second, learners need to experience how to share and critique their models, to talk about interesting mathematical phenomena that underpin them. Third, and as a corollary of the first two, researchers and teachers will need to find new ways to represent mathematical knowledge that are designed simultaneously to be learnable (in the way that, for instance, algebra was not) and rigorous (so that mathematical integrity is not sacrificed on the altar of simplicity).

In this paper, I will illustrate the approach based on several strands of work that have emerged from recent studies with colleagues in London including the "Playground" and "Weblabs" projects.

Introduction

My concern in this paper is the possibility of a *fundamental* role for the computer. The history of the computer's presence in the field of mathematics education is very short. When compared with the time that it took for our current notational systems to evolve, a mere thirty years is practically insignificant and it is hardly surprising that the development of alternative representational infrastructures for mathematical expression has not reached an equal maturity (see Kaput, Hoyles and Noss, 2002).

I should define what I mean by *fundamental [2]*. I mean change that is both basic and essential from several perspectives. From a cognitive point of view, in which I think it is legitimate to imagine radically different ways that people come to understand mathematical objects and relationships. From an epistemological point of view, in which the very nature of the subject changes in subtle and perhaps largely unforeseen ways – of course in the broader mathematical landscape, this change is already happening apace. And from a pedagogical point of view, in which the established norms and routines of curricula approaches, the modes of instruction and the design of materials are subject to radical reappraisal, in the light of what technologies can invite their users to do, to think, and to share.

Of course definitions of this kind are not like mathematical definitions. What actually counts as a qualitative fundamental change will in fact be comprised of numerous and contested quantitative changes, no single one of which can be labelled unambiguously

as "fundamental". Nonetheless, I think that with the exception of some small corners of the mathematical spectrum, I am safe in assuming that fundamental change of this kind has hardly occurred, and in so far as it applies to the wider educational system, has yet to begin.

My theme is not going to be only focused on whether there *exists* a fundamental role, but what it means to build new kinds of representational systems that could help to construct such a role. My rationale is not simply that the technology makes such constructions possible – but that the need for mathematical knowledge, and for new kinds of mathematical knowledge has never been greater. From a cultural point of view, we live in a world where workplaces are dominated by computer screens, each one of which is merely a peephole into a set of mathematical models; a world in which newspapers and TV screens are awash with statistical diagrams; and a culture in which more and more artefacts have a computational element (we are only at the beginning of ubiquitous computing) that hide computationally-instantiated pieces of mathematical knowledge. I will not make the case here that more and more people need to know something – something significant – about the mathematical knowledge that drives these devices (see Noss, 2002 for such a case).

So at the risk of trivialising the issue, let me share a resonant example. I was informed yesterday in a restaurant that it was 'impossible' to order a starter as a main course because the computer would not allow it. I would have liked, at least, that the bearer of this news should have appreciated what was funny about this situation, and what role the computer is being attributed in this interchange! More seriously, when I click on a search engine, I do not for a moment imagine that each of us should become expert in the algorithm that drives it. But I do envision that such information is important at some level to everyone who uses a search engine. And it turns out, when all the important details of the mechanism underlying it are left out, that it is very explainable indeed. Here is a description of google's search algorithm – PageRank - provided by the founders (Page and Brin):

> PageRank relies on the uniquely democratic nature of the web by using its vast link structure as an indicator of an individual page's value. In essence, Google interprets a link from page A to page B as a vote, by page A, for page B. But, Google looks at more than the sheer volume of votes, or links a page receives; it also analyzes the page that casts the vote. Votes cast by pages that are themselves "important" weigh more heavily and help to make other pages "important."
>
> www.google.com/technology/

Many questions might arise in the mind of the user who reads such information. Surely what is important to one person may not be important to another? How does the system combine 'importance' with 'relevance'? In fact, these questions are answered quite adequately on the Google website – in principle (if not in practice) there is a simple explanation. The key point is that appreciating that there *are* algorithms to drive the query is at least as important – for nearly everyone, *more* important – than what the algorithms actually are. Inviting good questions, and interesting discussions of this kind is an important imperative of modern life.

I think there is a corresponding a didactical imperative. of view. In order to illustrate the point, I propose to tell a true story.

> Two mathematicians and a physicist are on a fast train from Paris to London. The physicist has recently been to an art exhibition, at which he has been

> impressed by a video installation consisting of the same event taken from two nearby cameras. As the train passes through Northern France, he wonders aloud whether the same effect could be achieved by a single camera mounted on a suitably fast-moving object, like the train. I will not bore the reader with the details of the discussion that ensued. Suffice to say that there was enough mathematical meat to take us under the English channel and well into the English countryside before we had finished.
>
> Later that day, all three had dinner with a friend who is a very highly-acclaimed and well-educated lawyer. The physicist engaged him in conversation, and shared the problem with him. "Imagine", he suggested, "a video of a cow standing in a field outside the train's window.
>
> L: "Of course you could tell that there was a time-lapse between two videos – the cow might twitch its tail!".
>
> P: "OK, then a tree"
>
> L: "Well, then you'd see the leaves blowing in the wind".
>
> P: "Well, OK, a point then".
>
> L: "Look, if it was a point, then you wouldn't be able to see it from the train!".

There is no doubt in my mind that the last line of this story is funny *only* to a mathematician. In fact, my own informal research telling it to various people backs up this view: the idea that one could consider an abstract notion like a point as a legitimate object, when the problem is couched in terms of cows and trees, is uniquely mathematical. Most people simply do not think that way. And yet, as I argued earlier, this kind of thinking – what Al Cuoco would call adopting a "Mathematical Habit of Mind" or what Papert has called "a Mathematical Way of Thinking", is a uniquely important accomplishment, and one that is increasingly important in modern life when the mechanisms that drive the objects that surround us are so opaque [3].

These two dimensions of problem, the one cognitive/epistemological and the other didactical, present some serious design challenges. I am going to focus on one design area, not because I dismiss other important advances over recent years, but because I want to focus on mechanisms, the ways that things work. This is one of the major contributions that the computer may make and I will claim that it is, potentially at least, indeed fundamental. Can it be achieved in ways that preserve the rigour of mathematics but simultaneously enhances its learnability? This is a difficult challenge. There are too many examples to list of attempts to enhance learnability that have sacrificed rigour; and the reverse difficulty is epitomised by the 'new math' movement in the middle of the last century. I will not go so far as to say that the resolution of the problem presupposes the existence of ubiquitous and powerful computing (it doesn't), but I *will* argue that such computing power makes it more possible and more achievable. The central tenet is that if one wants both rigour *and* learnability, we are forced to attend to the notational systems we use. And if notational systems are our focus, then it really does make sense to cast the computer in a fundamental role.

I will focus on just a single class of computational environments, which I call *autoexpressive* systems (see Hoyles & Noss, 2003), a system in which the things one needs to do to effect change, to make objects move or react, is contained within the system itself as part of a symbolic system with precise rules of transformation. Until recently, this role could only be conceived as a computer programming language. But

as several presentations at ICME 10 have shown, the line that separates a programming language and a more general class of specialised computational system is blurring every day. More and more of the systems we are using are employing functionalities that 15 years ago, would only have accrued from programming languages. Making visible the relationships involved in, say, a geometrical construction, has become a commonplace in the powerful software that we are beginning to take for granted.

We have learned a lot in this time. We now have some three decades of research into the design and utility of (especially) autoexpressive systems, and research is becoming cumulative. We know, for example, that such systems have transformative potential, not only for the learner, but for the knowledge to be learned (for a review of this literature, see Hoyles and Noss, 2003). We have learned that such systems invite users to construct a unique connection between experience (the objects they see and feel) and analysis (how they express the relationships between them). And we have seen how such systems provide researchers and teachers with a unique window on the creation of mathematical ideas (for an elaboration of this last point, see Noss and Hoyles, 1996).

Let me now give an overview of the remainder of my paper. My underlying theme is to elaborate what are the possibilities of the new representational systems that can be designed. I am going to give just three examples. Let me say at the outset that I do not believe that any of them are going specifically to indicate *the* way forward. Neither do I think it likely that any of them will scale in any significant sense. Researchers cannot answer all questions – or even ask them – and I have no evidence that the examples I will use indicate more than the mathematical possibilities inherent in the approach. Turning possibilities into generalised reality will have to wait for other researchers, curriculum developers and software designers.

The three examples, or as I will call them "case studies", will each illustrate one possibility of the new representations. All three will involve attempts to link children's culture with mathematics. The first represents an initial attempt at a novel representational system and the ways in which it shaped learners' mathematical expression. The second will provide a closer focus – via a generic example – on the kinds of knowledge developed, and how they align with traditional mathematics. And the third will focus on an under-researched area in the field, namely the creation of mathematical cultures in a connected web-based environment. This last will represent a modest attempt to exploit the web as something more than simply a means to find 'information'.

Case study 1: Developing a new formalism for expressing causality and inference

The first example, taken from the Playground project [4], involved a group of researchers based in four European countries that focused on the development of a system for children, aged 8 or less, to play, construct and share computer games.

A major premise of the Playground project was that it made sense to try to tap into children's own culture. Of course, this would be relatively straightforward if we sacrificed the goals of the exercise – to learn mathematics. Indeed, videogames do not at first sight present an appealing domain: they are often violent, usually sexist, and many do not seem to have much connection with learning, other than learning how to take quick decisions and coordinate hand and eye.

But a closer look suggests that this may be a too-hasty conclusion. Computer games are a microcosm of a formal structure, a place where making something happen means expressing a set of rules within a rigorous system: when a set of conditions C are true, then all objects with properties p_C (properties that are triggered when C is satisfied) will display a set of specified behaviours. In one sense, becoming expert at playing a videogame consists of building a mental database of these rules, appreciating how they interact, and welding them into an overall narrative that carries the game forward. Of course, all this is mostly tacit, and little if any can straightforwardly be mapped onto anything we – or the players – might recognise *prima facie* as mathematical learning.

There is, however, at least one interesting manifestation of an alternative culture, in which video game players seem dissatisfied to be cast merely in the role of players. A casual google search for "video game cheats" currently reveals the existence of 174000 pages. A cheat is a means to outwit the game designer, usually a set of instructions for pressing a combination of keys that will subvert the behaviour of the game. Without wishing to over-romanticise this phenomenon, it appears like something of a cry of dissent from the game-players, an attempt to wrestle a little of the programmer's craft, a piece of the designer's expertise, back from the game producers.

It is this expertise that we wanted, in the Playground project, to confer on students: in a real sense, we wanted our students to 'play with rules'. The broad hypothesis of this work – partially but yet to be fully confirmed – is that deep connection with a formal system of this kind can serve as a powerful knowledge substrate upon which future mathematical learning and teaching can be built. Our goal was for the students to become game designers and builders rather than mere players of games designed by someone else so that they too could participate in what it means to build models of how things work.

Our first attempt at designing a system with these properties resulted in a system we called "Pathways", in which students could build iconic representations of (loosely) object-oriented programs [5]. I will give just one illustration of what I mean. Figure 1 shows a typical video game screen. A spacecraft, some space monsters, a couple of score displays (in the top corners). And, less typically, a set of controls along the bottom of the screen.

Figure 1: a game display.

These controls allow the player to reshape (or completely recast) the game, but introducing new objects, new backgrounds, and new rules that govern the behaviour of each object.

Figure 2: two boxes that were opened from the icons in the control bar at the foot of the screen. The Toolbox at the top of the figure contains the following icons: a wand for copying; a bomb for deleting; a star for changing various settings of an object including its shape; a microphone for giving comments to objects and the mouth for speaking comments (in a variety of languages). The opened Stonesbox (below the Toolbox) is currently revealing some of the when stones. The when stones represent conditions such as "when I am touching". The Stonesbox also contains do stones that represent actions such as "I jump" and behaviours that allow an object to be controlled by the mouse, joystick or keyboard. There are further icons in the control box to open a stored game, change the background, to create a new object in a game and to switch a game on or off.

Figure 3, shows the set of rules that govern the behaviour of the spacecraft:

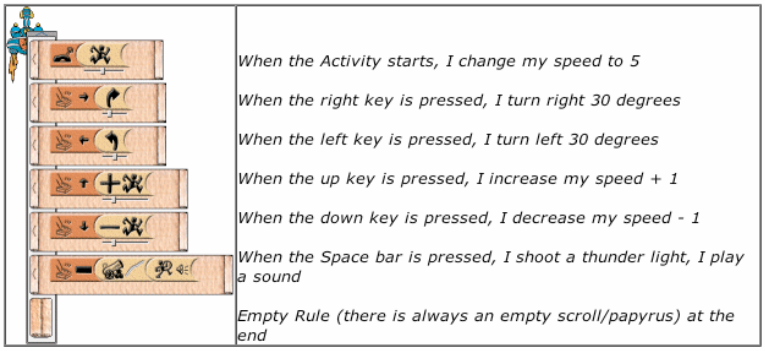


Figure 3: the rules governing the spacecraft's behaviour.

As Figure 3 makes clear, the expression of each rule (they run in parallel on the 'back' of each object) is a matter of dragging an appropriate icon and snapping it into position (the precise semiotics of the colouring of the different icons, and a variety of other significant design features are not important here – and in general were not problematic for any of the students we worked with).

In fact, what Figure 3 does not show is the ways in which each command can be spoken by the system, and how the sliders speak their values. These are quite powerful functionalities given our intention – they invite students to build a rich set of games with appropriate properties. It is not a Turing complete system, in which – in principle – *anything* might be expressed. In fact, the decision to stop short of a

complete computational medium was problematic for us as designers, and raises an important – and unresolved – issue. What, precisely, is the right grain size of functionality to provide students simultaneously with sufficiently interesting objects to study, and means to understand and recombine them? Posed in this way, the question is over-simplified, in that it leaves out the crucial issues of who is to use the system, for what purposes, and in what setting. Nevertheless, it points to a continuing problematic thread in our research, and one which recurs in all the examples I will give here.

The question of using ideas – as executable representations of mathematical ideas – without necessarily constructing or 'understanding' how these ideas work, is a new version of an old question. Just to what extent is it possible or desirable in mathematics to ask students to use an idea before they understand it? Traditionally – at least in terms of standard curricula – the answer has been 'not at all'. But with the advent of technology, it becomes possible to envisage students using a system without understanding, and to develop understanding in the course of use. It is my belief – and I will state it without proof – is that this possibility of developing understand *in use* points to one of the fundamental roles for the computer.

Before I leave Pathways, let me outline – in just enough detail to highlight the important issues – a real example (a full description of this episode is given in Goldstein, R., Kalas, I., Noss, R. and Pratt D., 2001).

Figure 4: The Millennium Dome game.

In Figure 4, two children have generated an idea for a game, based on an earlier version they were invited to play. The idea is for the player, who controls the dog, to collect the spikes of the dome before they are automatically collected by the system. In deciding how to implement their design, they have to confront a number of problems. Here are four of them:

1. what does it mean to 'collect' a spike? Should the spike explode? Disappear? How should collection be effected (by the dog touching it? How should "touch" be implemented?)

2. where does the rule go? There is a choice – the rule could go on the dog (when I touch a spike…) or on the spike (when I touch the dog). This situation is symmetrical from a mathematical point of view, but not from the computing point of view, as one child realises:

> Oh no. Which would I have to change the rule on? The dog or … [the spike]? I need to do it to each of the spikes because if the dog will only go into one spike it will happen. If it goes into a different one, nothing will happen.

She realises, it seems, that if the rule is on the dog, then the dog will need six rules (one for each spike – there is no way currently in Pathways to tell the system that the spikes are all objects in the same class). On the other hand, if it goes on the spike, while each spike will need a rule, it will be *the same* rule – triggered by touching the dog.

3. does the system 'know' when all the spikes are gone?

In initial discussion about the game, one of the students expressed an assumption that once the spikes had disappeared, the game would stop. In fact, of course, there is no way for the system to know the last spike *is* the last remaining spike, unless it has some way of keeping count. In that case, something needs to happen – and one student suggests that it happens to the dog (the first version was more violent than the second!):

> When the dog touches the spikes, it could explode. We could tell the dog to hide when it gets touched and then to show again.

4. how can we keep score?

In Pathways, there is a mechanism for any object to 'broadcast' a message (in the form of a coloured envelope) and for any number of objects to receive it. In fact, the idea that the score itself should be an object (and that – because it is a special kind of object consisting of a number) it is possible to do arithmetic with that object (e.g. add 1 to it) is not straightforward, and is a small but significant generalisation of the kind of object-orientation on which the system is based. This last issue, incidentally, points to an significant way in which it is possible to build a synergy between the powerful idea (in computer science terms) of object-orientation, and mathematical expression. Rules that live with their objects bring the rules to life, in ways that say – algebraic rules – generally do not.

These four problems suggest ways in which the constraints and possibilities of the representational system shape the ways in which expression can occur. It is clear, even from the short extracts presented, that these constraints were operating at a conscious level. I contend that this is an interesting and potentially important avenue on which to follow up: so much of the representational infrastructure of mathematical expression constrains and affords invisibly, and to an extent at least, this invisibility is seriously problematic for students. Becoming aware of the constraints of the system is a very important part of what it means to think mathematically. More generally, even this toy system represented by the Pathways software, provided a setting in which rigorous formalization was a means to an end, rather than an end in itself.

Case study 2: building video games with ToonTalk

A second strand of the Playground project sought to build a video game construction toolset in a completely new programming language called *ToonTalk* (see Kahn, 1996; www.toontalk.com). Toontalk is highly unusual: it is a programming system in which

every computational element (programs, objects, conditionals, communication…) is instantiated in terms of animated cartoon-like characters. By 'instantiated' I do not imply that these animations are merely 'representations' of some other (perhaps, textual) program – the animation is the source code of the program.

I cannot describe ToonTalk here. It is, in fact, a completely novel type of programming system, and we are still trying to assess its possibilities and limitations for students. What one gains by being able to construct programs in terms of narrative (to write a program, one simply gives a sequence of instructions to a 'robot' to 'train' him), one may lose by being unable to 'read' – at least read in the conventional sense – the text of a program subsequently. We do not yet know. The point I want to make here does not concern the difficulty or ease of the system, but rather the kinds of learning that novel representational structures make possible.

An illustrative episode [6].

Mitchell is an eight year-old boy in an inner-city school. He has been helping the researchers to design and debug the Playground system for about a year. He has participated in several experimental sessions, and is a member of his school's computer club, in which all Playground sessions take place. He is playing a game where he controls a character called ‘Dusty’, the creature with two legs on the right side of Figure 5. Dusty shoots out flowers every time the force joystick trigger is pressed (it jumps in the hand as the button is depressed), and the player collects points by hitting an animated target moving vertically up the left-hand edge of the screen. Whenever it is hit by a flower, the target changes shape into another animated character. Mitchell finds it very easy to play and achieve high scores since he can move his character as close to the target as he pleases.

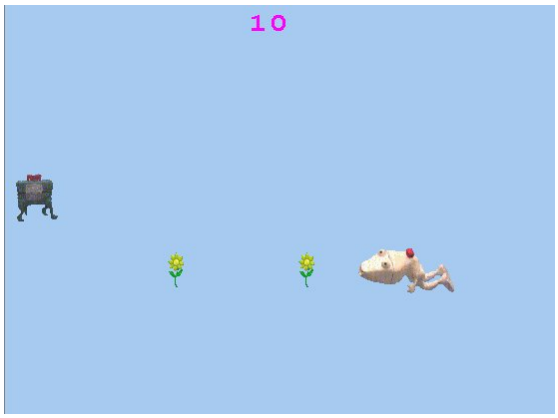


Figure 5: A screenshot of the game. Dusty is shooting flowers from right to left. A 'target' is moving up the left-hand edge of the screen. The target changes shape as it is hit by a flower.

Mitchell decides that the game would be more fun if it was competitive. So, at his request, the researcher adds another player character, this one controlled by the mouse. Although she talks through what she is doing, the programming is done by her, with Mitchell watching as she copies the original Dusty and replaces its joystick behaviours with mouse behaviours. Making these modifications is facilitated by the modularity of the behaviours or components – we call them "animagadgets"', as they are represented by animated pictures – and is essentially achieved by replacing one component, "I move with joystick" with another "I move with mouse". Any object has

its program on its back – there is a direct connection between a thing and what the thing does, accessible merely by "flipping" it over. Although Mitchell can read these descriptions, he can, if he wishes, listen to the behaviours say what they do (a choice of voices is available) and there is, in addition, an animated representation of what the behaviour does (visible in static form in the top left-hand corner of Figure 6).

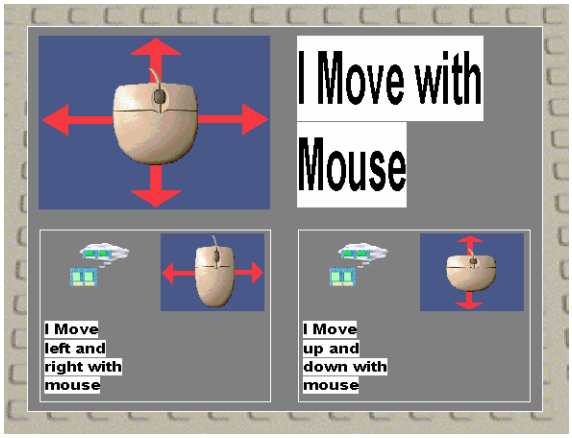


Figure 6: The 'I move with mouse' gadget, and the two sub-gadgets moving the mouse horizontally and vertically.

One important aspect of the "I move with mouse" behaviour, is that it actually consists of two sub-behaviours, "I move left and right with mouse", and "I move up and down with mouse" (the same applied, of course, to the joystick gadgets). Both of these program-fragments are visible at the bottom of Figure 6. As it turned out, the way that we had designed this behaviour had unexpected consequences.

Mitchell plays the two-player game with the researcher. Suddenly, he tells everyone in the room to close their eyes – "no peeping!". He deftly removes the "I move left and right with mouse" component of the "I move with mouse" behaviour from the back of his opponent's character – effectively disabling her mouse by restricting it only to vertical movements, while he has two-dimensional control and can get as close as he likes to the target.

When we are all allowed to open our eyes, Mitchell triumphantly demands a rematch which, unsurprisingly – now that his opponent's motion is restricted to vertical movements – he wins conclusively, much to his satisfaction.

There is much that can be gleaned from this episode (an obvious affective dimension, a question of expertise with the system, the possibilities of such learning in a real school-based learning environment). Here I want to focus on just one facet, namely the ways in which what Mathew actually knew was instantiated within the representational system he had to hand (and with which he had attained some facility).

There is, of course, much that Mathew did not 'know'. He certainly could not – aged 8 – be said to have realised that two-dimensional motion could be decomposed into the vector sum of two linear motions. But there is one thing that it is reasonable to assert that Mathew did know, and we can express it as something like: "I can build a program to move my object around the whole screen by 'adding' horizontal and vertical robots". This is already a generalization from what he actually *did* ("I can

constrain my opponent's motion by removing the horizontal component of her motion") but it forms a reasonable conjecture as to his thinking.

The point of the episode is that the difficulty of the idea of adding (or subtracting) vectors – however expressed – is not so much in the idea itself, but in the *representation* of that idea. In traditional mathematical terms, the idea of vector only takes shape when one begins to operate on vectors [7]: knowing what a vector 'is' presupposes that one knows how to add and subtract them, and much more besides. It is this set of operations on the object that confers conceptual power to the idea, but it is also what makes the notion difficult: a classic difficulty for students is realising that the 'addition' of vectors is only a metaphor, and that adding vectors and adding numbers confuses two different meanings of the word 'adding'.

In the Playground setting, the addition of the two vectors was taken care of by the simultaneously-executed program-fragments (move up, move right), which in turn 'resulted' from the object-oriented and parallel behaviour of the system itself. To what extent Mitchell was explicitly aware of these structures, we do not know. But it is likely that he 'knew' – perhaps 'took for granted' would be a better term – the implications of these ways of representing processes, and certainly took advantage of them when it came to his trick.

It is difficult to characterise Mitchell's insight. What kind of knowledge does it represent? We have found it useful to characterise this kind of knowledge as *situated abstraction*, a form of abstraction that is – like formal abstraction – an expression of invariant relationships, but one that is locked into and expressed within the tools and language – the representational structures – of the setting. In Mitchell's case, his actions and his manipulation of the program expressed his knowledge eloquently enough, but not as a mathematical abstraction in the conventional sense. The idea of situated abstraction is elaborated in Noss and Hoyles, 1996. Here, the important point is that expressing abstract ideas within novel representations gives rise to new ways to think about abstraction for the student, and perhaps equally challenging, for the researcher.

The Playground project taught us that if the representational infrastructure of a computational/mathematical environment is so critical, then getting the design of that infrastructure right was, and is, a delicate and formidable challenge. Further, it reminded us just how difficult it often is to align our mathematical goals as teachers with those that children spontaneously adopted: we certainly could not advocate learning mathematics entirely through chance encounters in game-construction. It is this issue we confronted in the *WebLabs* studies described next.

Case study 3: The WebLabs project

WebLabs is a three-year project (weblabs.eu.com) in which we have adopted a more explicit focus on mathematical and scientific ideas. By designing alternative ways to express and share mathematical and scientific ideas, we hope to make them accessible to students who might otherwise not encounter them until much later - if at all.

There are two main focal points of our design effort. First, to construct a set of tools and activities, based on *ToonTalk*, that allow students to address various knowledge domains (like convergence or limit) much like one might do in an experimental lab – do experiments, test conjectures, look for counterexamples, take part in interesting and challenging discussions. We have built a series of *transparent modules*, toolkits of working models for students to think about and manipulate, as well as sequences of

activities for them to try – transparent in the sense that it is easy to look not only at what the models do, but how they work.

The second focus is the construction of *WebReports*, a web-based system for collaborating, co-constructing and thinking about models, as well as a simple mechanism for sharing the models themselves. *WebReports* allows us a window on the ways students can share their models of evolving knowledge at a distance, discuss, change and manipulate them, and provides a way to assess how the new representational structures influence student thinking. Figure 7 illustrates some of the topics that are available as one enters WebReports system, while Figure 8 shows some of the tools available for exploring one of the topic domains, namely *sequences.*

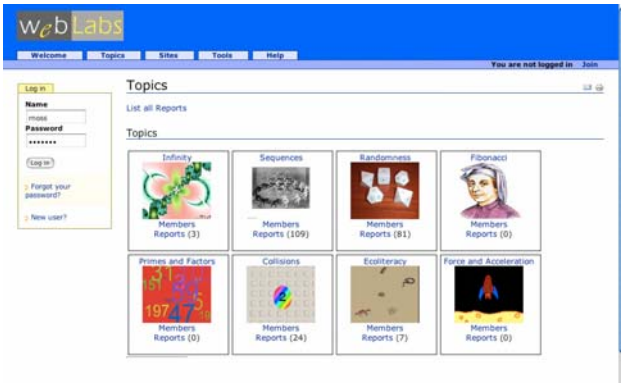


Figure 7: the topics available in WebReports: infinity, sequences, randomness, Fibonnacci numbers etc.

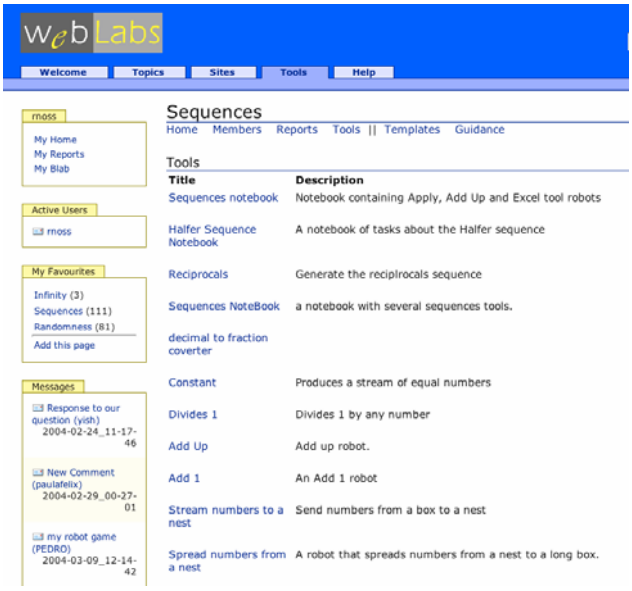


Figure 8: some of the tools available for exploring sequences.

In order to establish a flavour of the kinds of interactions that can take place in Weblabs, I will describe a sequence of episodes that were initiated by a particular activity, called "Guess my robot". The key players in the story are Rita, a 12-year old girl in Lisbon, and Nasko, a 12-year-old boy in Sofia [8]. The Sofia group consists of 6 boys and girls, aged 11-12, working with *WebLabs* researchers. They have been working with *ToonTalk* for several months, approximately once a week for a couple of hours. The second group is from a village south of Lisbon. Paula, a teacher and researcher in the *WebLabs* team, worked with a school group (aged 12-13) there during the first project year. Researchers in both groups act as teachers, guiding the students through the mathematical ideas as well as through the programming skills. At the same time, the researchers facilitate interactions, by pointing children to interesting peer reports and helping them to add a few words in English to their own reports.

The activity we designed was based on the well-known “Guess my rule” game (see Mor & Sendova, 2003). It has been used in many classrooms in the UK over many years to provoke children to discuss and compare the formulation of rules, and in particular the equivalence (or not) of their algebraic symbolism. In its classical form, it has been used as an introduction to functions and to formal algebraic notation. As Carraher and Earnest (2003) have recently reported, even children in younger grades enjoy participating in this game, and can be drawn into discussions of an algebraic nature through using it.

Our version of the game is somewhat different. The idea is that a student sets out a challenge, in the form of a sequence that has been produced by a robot (computer program) he or she has built, and posts it on the web. The challenge is for the second player to produce a robot that results in the same sequence. So, in our game, *proposers* (students) invent a rule and program it so that it generates a numerical sequence, and publish the first few terms it generates in a *WebReport. Responders* then have to build a robot that will produce this sequence, and thus work out an underlying rule for its generation. The new element in our variant of the game is that “rules” have to be encoded as robots: one responds to a challenge sequence by posting a robot that produces “the same” sequence. So the encoding of the rule takes the form of a process-description in the form of a 'program'. Managing to reproduce someone else's sequence by training a robot, is a way to show that you have grasped how the sequence may have been originally generated. As one girl said:

“*So, like, the robot is my proof that I got it?*”

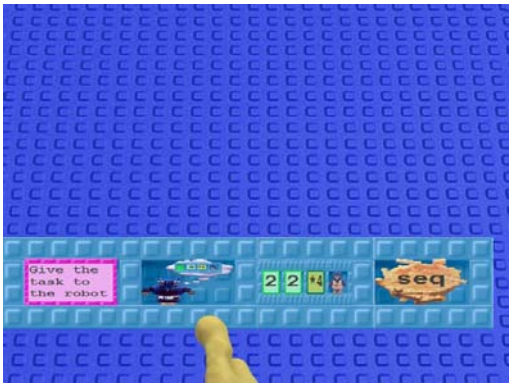
Rita found the 'guess my robot' activity, and decided to pose her own challenge. The sequence she posted was

2, 16, 72, 296, 1192 … (see Figure 9).

Figure 9: Rita's "Guess My Robot" challenge.

A few days after she posted it, the Sofia *WebLabs* group held a session, and some of the students tried solving Rita's challenge (the robot Rita used, and an algebraic description of its actions are given in Figure 10). Nasko posted his response. He had built a robot that produced Rita's five terms, but it turned out to be different to Nasko's (this point, incidentally, illustrates how model-construction and sharing generates interesting questions like uniqueness and existence theorems, that are usually passed over). Nasko also realised that the same robot could be used to generate other sequences by changing its initial inputs. So, he posed a two-part challenge back at Rita:

- Could she use *his* robot to generate a new sequence of five terms?
- Could she use *her* robot to generate the same sequence?



$$A_1 = 2$$

$$A_n = (A_{n-1} + 2) * 4$$

Figure 10: Rita's robot, used to produce the initial challenge, and an algebraic representation of the robot's actions.

Nasko's response to Rita is given in Figure 11 (in fact, Nasko's robot computes $A_n = A_{n-1} + 14 * 4^{n-2}; A_1 = 2$.

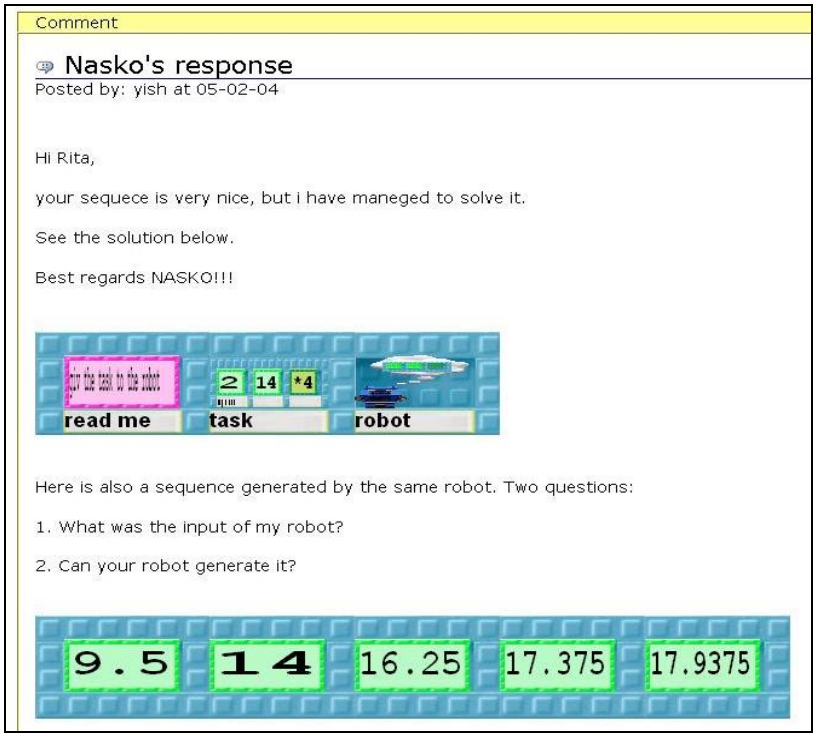


Figure 11: Nasko's response to Rita and his twofold new challenge.

The story continued. Ivan (a friend of Nasko's) joined in the discussion, and other robots were built and exchanged. A few days later Rita came to her next session. She was very excited to find comments on her page – and from children on the other side of Europe! She immediately clicked on the *ToonTalk* robots in the responses, and watched them step through the process of rule-generation. She was totally surprised: Nasko and Ivan had solved her challenge, but their robots seemed completely different from hers (and one from the other)!

The details of the story are given in Mor et al. (in press). Here I will focus on just one 'ending', which involved Rita's response to Nasko. She worked out what inputs Nasko must have given his robot, and showed that *her* robot could in fact generate the same output as his. Then we – the researchers in UK, lurking on the WebReport system – asked Rita how she had worked out what the sequence was, and how she had ended up with the robot she built?

Rita responded as follows:

> In Nasko's box for my sequence he used 2 in the first hole, and added 14 from the second hole to get the second term (16), and multiplies that by 4. Then I think to get the first term of Nasko's sequence I need 9.5 in the first hole. The number in the second hole has to be a number that you add to 9.5 to get 14 (the second term), this number is 4.5. In my sequence he uses x4 to get the third term (16 + 14 x 4). Then for his sequence I think like this: 14 + 4.5 * "something" should give me 16.25, or 4.5 "something" should give me 2.25. But 2.25 is half of 4.5, then in third hole of the box I need to put /2.

Rita's response is clear enough: she has used the structure of the boxes and what she knows about the robot construction she has seen (remember these have been shared alongside the discussions). But still we are not satisfied. We ask Rita again: "How do you know that the two robots generate the same sequence"? The next day, Rita surprises us. She has generated two robots, one is hers and the other Nasko's. Then she has made a new robot that subtracts one stream of outputs from the other. She has watched the robots create a stream of zeros, and she has generated thousands of zeros in this way. Sure enough, this is a 'proof' of her conjecture that the sequences are the same!

The 'joke' in the last paragraph is worth considering. Of course this is not a proof in any conventional sense. Or even in the context of the novel representations with which the students were working (we do not know what the 6001st. term of the subtracted stream would be!). But it is a *kind* of proof, surely: something like "Given any integer N, my program will generate a stream of N zeros…". Still not proved, of course, but at least a basis for establishing the structural rationale that could – in fact – provide the basis for a conventional (and correct) proof.

Conclusion: A fundamental role?

Mathematical models and representations have to be considered as a pair. They will be key in the coming years. Seeking, and increasingly finding, alternative ways to think about and even define mathematical objects using computational media, will be a major challenge of the next few decades. The complexity of an idea is often in the notational system, as the examples above illustrate: Mitchell's difficulty would have been to describe his discovery in algebra of vectors; Rita's in conventional algebraic manipulation. Neither were dealing with ideas that were too complex, only with *ideas* that hitherto could not be adequately considered without mastery of the rules of transformation governing the symbol-system that has evolved for express conventional mathematics – a system that is predicated on the static, non-narrative, timeless and depopulated world of mathematics.

New notational systems make us aware of how *all* notational shape what it is possible to express, and how it is expressed. Knowledge is transformed, abstractions become situated abstractions, proofs enlarge their scope, definitions change their character. New systems for representing mathematical objects and relationships help to make us aware of how sensitive representations are to initial conditions: small changes in the ways we express things make huge changes in the ease or difficulty of expression (think of Leibniz's calculus notation: for a fascinating discussion, see diSessa, 2000). It is for this reason, if no other, that design of activities – not just of 'systems' – is a critical research challenge.

Notational systems do not merely shape what can be expressed: they are shaped by *persons* in use. As Rita showed, new possibilities open up, and new tools, recombination of elements, novel versions of existing pieces, get built. The dialectic between learner and artefact in which each is mutually constituted in action has recently been a focus for French didactical researchers: see, for example, Vérillon (2000), for a useful summary of these ideas; also Vérillon & Rabardel (1995). For a seminal mathematical perspective, see Artigue (2002).

These issues raise the question of *legitimation* that I raised earlier. Just how do we judge what counts as legitimate mathematical knowledge? When knowledge, tool and activity are coevolving, little – if anything - can be kept invariant. It will be some time

before research and practice have answers to this question: in terms of systemic change in the educational domain, it is perhaps the most difficult of all.

My final point concerns the nature of e-cultures. In the wider world of mathematical education, some of the most interesting research has explored the emergence of classroom cultures, how understandings are mediated by collaboration, and how discourse shapes mathematical cultures. But we have yet to take even the first step towards appreciating how the computer's presence mediates these cultures. Rita and Nasko's story is hardly typical, even in the time and resource-consuming context of a large research project. We have tiny indicators (for an intriguing hypothesis in this field, see Noss, R., Hoyles, C., Gurtner, J-L., Adamson, R. and Lowe, S., 2002). What is certain, as computing power becomes ubiquitous and invisible, is that the ways in which technology mediates the creation and maintenance of mathematical cultures in learning situations will be substantially different from the ways it develops (or fails to develop) at present.

Endnotes

[1]: A version of this paper was presented as a Regular Lecture at the Tenth International Congress of Mathematics Education, Copenhagen, 2005.

[2]: I am grateful to Seymour Papert for a fascinating conversation on this theme.

[3]: I remember as a child taking my father's watch to pieces and trying to reassemble the cogs and gears. Were my children to have tried the same thing, they would have found a chip.

[4]: I would like to express my thanks to all my colleagues in London and elsewhere who have collaborated on the projects I report here. Special thanks are due to Celia Hoyles, who has co-directed both the Playground and Weblabs projects with me, and who made helpful comments on an earlier draft of this paper.

[5] Pathways has subsequently been marketed as "Magic Forest" (see http://www.logo.com/cat/view/magicforest.html). It was written in Imagine, a new and extremely powerful version of Logo: see Kalas and Blaho (2000).

[6]: The following five paragraphs are based on Noss (2001).

[7]: This is, surely, a generalisable property of mathematical objects.

[8]: This episode is based on a description that is due to appear in Mor et al (in press).

References

Artigue, M. (2002). 'Learning Mathematics in a CAS environment: The Genesis of a Reflection about Instrumentation and the Dialectics between Technical and Conceptual work'. *International Journal of Computers for Mathematical Learning*, 7, 3, 245-274.

Carraher, D. & Earnest, D. (2003) Guess My Rule Revisited in *Proceedings of 27th International Conference for the Psychology of Mathematics Education*, Honolulu.

DiSessa, A. (2000). *Changing minds, computers, learning and literacy*. Cambridge, MA: MIT Press.

Goldstein, R., Kalas, I., Noss, R. and Pratt D. (2001). Building Rules. In M. Beynon, C. L. Nehaniv & K. Dautenhahn (Eds), *Proceedings of the 4th International Conference of Cognitive Technology CT2001*, 267-281. University of Warwick, Coventry: UK.

Hoyles, C. and Noss, R. (2003). What can digital technologies take from and bring to research in mathematics education? In: Bishop, A. Second International

Handbook of Research in Mathematics Education. Dordrecht: Kluwer. page 323- 349.

Kahn, K. (1996) ToonTalk - An Animated Programming Environment for Children, *Journal of Visual Languages and Computing*, 7(2), pp. 197-217.

Kalas, I. and Blaho, A. (2000). Imagine… New Generation of Logo: Programmable Pictures. In: *Proceedings of WCC2000*. Beijing.

Kaput J., Hoyles, C., Noss R. (2002). Developing New Notations for a Learnable Mathematics in the Computational Era. In English, L. (Ed) *Handbook of International Research in Mathematics Education*. London: Lawrence Erlbaum. pp. 51-75

Mor Y., Hoyles, C., Kahn K., Noss R & Simpson G. (in press). Thinking in Process. *Micromath*.

Mor, Y. & Sendova, E. (2003) ToonTalking about Mathematics, in: I. Derzhanski, N. Dimitrova, S. Grozdev & E. Sendova (Eds) *History and Education in Mathematics and Informatics, Attracting Talent to Science; Proceedings of the International Congress MASSEE 2003*, september 15-21, Borovets, Bulgaria (Latvia, University of Latvia).

Noss R. & Hoyles, C. (1996) *Windows on Mathematical Meanings: Learning, Cultures and Computers*. Dordrecht: Kluwer.

Noss, R. (2001) For a Learnable Mathematics in the Digital Culture. *Educational Studies in Mathematics*, 48, 21-46.

Noss, R. (2002) Mathematical Epistemologies at Work. *For the Learning of Mathematics*. 22(2), pp. 2-13.

Noss, R., Hoyles, C., Gurtner, J-L., Adamson, R. and Lowe, S. (2002). Face-to-face and online collaboration: appreciating rules and adding complexity. *International Journal of Continuing Engineering Education and Lifelong Learning* 12, 5/6, 521- 539.

Vérillon, P. & Rabardel, P. (1995). 'Cognition and Artefacts: a contribution to the study of thought in relation to instrumented activity', *European Journal of Psychology of Education*, 10 (1), pp. 77-101.

Vérillon, P. (2000). "Revisiting Piaget and Vygotsky: In Search of a Learning Model for Technology Education". *The Journal of Technology Studies XXVI*, 1.
[download: http://scholar.lib.vt.edu/ejournals/JTS/Winter-Spring-2000/]