Effective Mathematics Education Software in the Primary School: A Teachers' Perspective

Janelle Pollard Roger Duke
janellep@csee.uq.edu.au rduke@csee.uq.edu.au
University of Queensland (Australia)
CSEE - UQ

Abstract

What types of mathematics education software do primary school teachers want? This paper reports the finding of a year long study focused on extracting ideas and beliefs from teachers about what types of mathematics education software they feel they could and would use within their classrooms. This paper addresses the following two questions; "What design aspects do teachers desire in mathematics education programs?" and "What type of mathematics education programs do teachers believe will most benefit their students?". These questions were addressed through a series of interviews conducted with teachers from five local primary school. A test suite of programs was developed to emphases different types of existing mathematical programs. The teachers responses to each of these programs were carefully recorded and analysed. At a software development level the teachers feedback suggested fifteen possible starting hypnotises for design principles on how to develop effective mathematics education programs.

1 Introduction

What types of mathematics education software do primary school teachers want? Currently there is little data exploring the needs of teachers and what exactly they are looking for from mathematics educational software.

Historically a lot of research has gone into developing and analysing programs like Cabri Geometry [10], Graphical Calculators [1], Java applets [12], JavaSketchPad [11], Geometer's Sketchpad [5], Scientific Notebook, Maple and Mathematica [13], CAL [7], and so forth. However this research has generally been student focused. This paper focuses on the needs of the teacher, rather than directly on the needs of the student and how they learn. Clearly, teachers have the needs of students firmly in the forefront of their minds. But the focus in this paper is on extracting ideas and beliefs from the teachers themselves about mathematics education software which teachers feel they could and would use within their classrooms.

Primary school teachers are typically not making effective use of existing mathematics education software in their everyday mathematics lessons [15]. Why? Is it because they feel the software does not teach students effectively? In order to create mathematics education software which teachers want to use, we must first find out what teachers believe works at the classroom level. This paper addresses the following two questions; "What design aspects do teachers desire in mathematics education programs?" and "What type of mathematics-education programs do teachers believe will most benefit their students?".

These issues were addressed in a year long study, by firstly analysing existing mathematics education software and dividing them into seven categorises, then creating test programs specifically for each category in the form of java applets. The rational behind creating the test suit of programs was that by giving the teachers small programs focused on emphasising different point, it would make it easier for them to identify what they felt would work, and what would not. Five schools from the Brisbane region, Australia, were involved in the interviewing process. The interviewing process consisted of sitting down with teachers one on one with screen shots of the test programs. The functionality of each program was verbalised and the teacher's responses recorded. A questionnaire about their experience as a teacher, current use of technology in the classroom and their future desires/ideas about the use of computers in their classroom, was also completed within this interviewing time.

This initial study is part of a larger project aimed at developing best-practice guidelines for the creation of the most suitable interfaces for mathematics education software. The aim of the initial study reported in this paper is to collect information which can be used to help formulate realistic hypotheses for more specific and in-depth studies.

Several surprising possible design principles were extracted from the teachers' feedback. We believe this should assist in the creation of more effective mathematics education programs.

2 Categorising mathematics education software

Mathematics education software was grouped into seven distinct categories. These categories are independent of subject matter and focused on the educational role of the software. The seven categories are as follows:

1. Problem orientated
2. Tool base
3. Simulation
4. Technique skill base
5. Conceptual skill base
6. Games
7. Resource based

Other categorizations of mathematics education software have been developed and tend to capture the same fundamental ideas as the categories which were created above. For example, Mawata [5] divided java applets developed for on-line lessons into seven types:

1. Applets to generate examples
2. Applets that give students simple exercises.
3. Applets that generate data.
4. Applets that guide a student through a sequence of steps.
5. Applets that present "picture proofs".
6. An applet can also be in the form of a mathematical puzzle.
7. An applet can set a theme for a whole course.

Other categorizations of mathematics education software have been developed and superficially seem to capture the same fundamental ideas as the categories which were created above. For example, Mawata [5] divided java applets developed for on-line lessons into seven types:

- Applets that give students *simple exercises* to make sure that they have *understood* a *definition* or *concept* is what I defined as a *conceptual skill based* program.
- Applets that *generate data* are *simulation* programs.
- Applets that guide a student through a *sequence of steps* are *technique skill based* programs.
- An applet in the form of a *mathematical puzzle* is a *game*.
- An applet which sets a *theme* for a whole course is a *problem orientated* program.

But there are some differences. Both categories *Applets to generate examples* and *Applets that present "picture proofs"* do not directly match with any of the categories I created because both of these categories are very specific. All other categories can be implemented in a variety of manners except these two, which focus specifically on the use of applet animation. A generalization of *Applets to generate examples* is a *resource based* program. This category covers programs that mimic classroom resources, which includes programs that animate examples, since examples are a classroom resource.

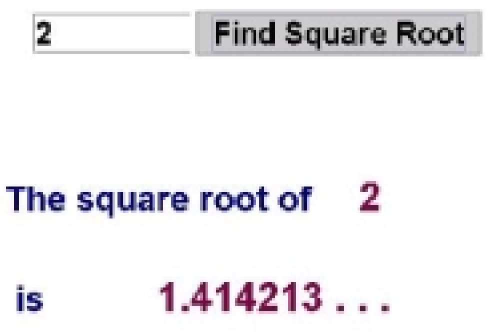
One category not mentioned was *tool based*. This is because *tool based* programs tend to be large, since they typically have to deal with complex computations. Java Applets are not usually used for this type of programming. However this category is clearly recognized within the current mathematics education literature [4] [13] [15].

3 The categories and teachers' reactions

This section looks in detail at each category, defining what type of programs belong in each category through specific examples and discussing teachers' reactions to these programs. All of the Java Applet examples are aimed at the primary school level.

3.1 Problem Orientated

A *problem orientated* program is written to solve one type of problem. Such a program is purpose built, with one functionality. An example is a program which finds the square root of the input number.



Advantages: The clear advantages of such programs is that they stand alone and are good at targeting students specific needs. However they lack extensionality because they are designed for a specific purpose, and this limits their classroom re-useability.

Teacher's comments: None of the teachers liked this type of program. They desired programs which target a wider audience and which could be used for more than one or two set activities.

3.2 Tool Based

Tool based programming involves the creation of complex calculators.

A *tool based* program takes the view that computers are machines designed to remove complex and time consuming tasks from humans. This is one of the strongest methodologies in conventional programming and influences how computers are integrated into classrooms [4] [15]. Currently there are a lot of commercial software packages with this design philosophy. A few examples are Matlab, Maple, Calculus Live [3], Mathematica and graphics calculators. However all of these examples are aimed at a high school or above level, not primary school. There have been several studies on how to successfully use *tool based* programs, especially graphics calculators, in teaching mathematics [1] [13] [15].

Advantages: According to Majewski [13], used well, *tool based* programs can allow students to focus on problem solving instead of algebra manipulation. By hiding the complexity of tasks students are able to explore problems they previously could not attempt because of the time consuming calculations. As a side effect, teachers can set more realistic mathematical problems and hence start to relate mathematical problems to students' life experiences [15].

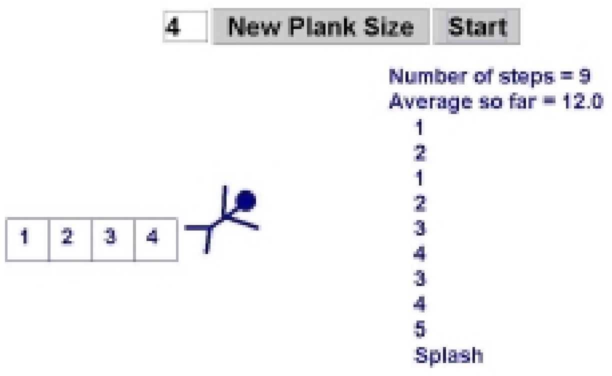
Teacher's comments: Teachers felt it was hard to see tool based programs being implemented in the primary school classroom because, at a primary school level the aim is to teach the students how to perform the manipulations and calculations. By hiding how the calculations are performed, students are not able to see/practice the calculation themselves. One of the teachers felt that this type of program would be useful in introducing a topic in mathematics by first letting students discover and explore a mathematical problem for themselves, and only then teaching them a formal method for solving the problem. However, she pointed out that there is barely enough class time to teach the basic core mathematical skills required and could give no specific examples of areas of the curriculum where this would clearly work.

3.3 Simulation

The aim of simulation is to visually represent problems so as to foster understanding of complex maths through estimation.

Simulations differ from a Tool Based view in that the tool gives the answer to a question, whereas a simulation gives the *data* and the students are expected to draw conclusions/hypotheses from the information. The point of this style of programming is not to solve a problem or prove it, but rather automate the task of creating data to analyse.

Walking the Plank is a simulation of a pirate being forced to walk the plank. The user can choose the size of the plank. The pirate can only go forward from the first position, but from then on chooses to go forward or backwards one step at a time, at random. How many steps will it take the pirate to fall off the plank? When you press the start button the program displays the position of the pirate. When the pirate fall off the end, the totalnumber of steps taken this round is displayed, along with the average number of steps so far for that plank size.



```
Number of steps = 9
Average so far = 12.0
    1
    2
    1
    2
    3
    4
    3
    4
    5
    Splash
```

By running the program several times with different plank sizes students can suggest likely formulae for approximating the average number of steps it would take for the pirate to fall off a plank of size n. An extension of this problem could be to change the rules controlling the pirate. For example he could go forwards or backwards one square; or jump forward three places. Now how many steps on average would it take the pirate to fall off a plank of size n?

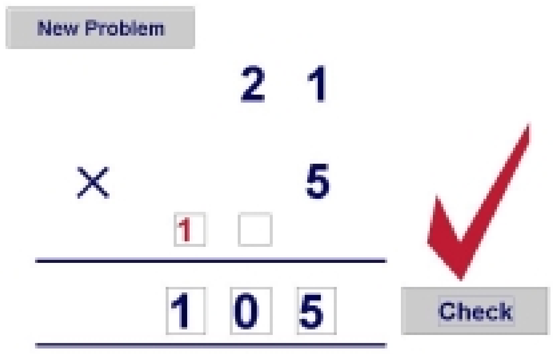
Advantages: The point of this programming style is not to solve the problems, but merely present ideas for open-ended investigations. Simulations of real life occurrences could be programmed, hence connecting mathematics to the students' life experiences. Yanagimoto [15] suggest that this would encourage students to actively participate in mathematics because they could see how mathematics fitted into their daily lives. Simulations also focus on higher level skills like problem solving and analysis, rather than merely mathematical technique.

Teacher's comments: About half of the teachers liked the concept and aim of the program, however they felt that this example was too advanced for primary school students. They were also a bit reticent about the practicalities of implementing such programs since this type of programs can only be used for one activity. Teachers would have to plan a lesson, or several lessons, focused on the theme of the problem so that the students had the knowledge to solve the problem. Also these programs were definitely not "set and forget" activities. Most students would need attention to encourage them in positive investigative directions. This would be difficult to set up if a teacher only had say, three students on computers and was trying to teach another lesson to the rest of the class.

3.4 Technique Skill Base

A skill based view of programming looks at teaching a specific skill (or subset of skills) through a single program or a small group of programs.

An example would be the program *Multiplication*, which does not teach what multiplication is, but rather teaches the process of multiplication. In this example students are given multiplication problems. If the calculation is incorrect then the computer hints at the right answer by correcting the carry values.



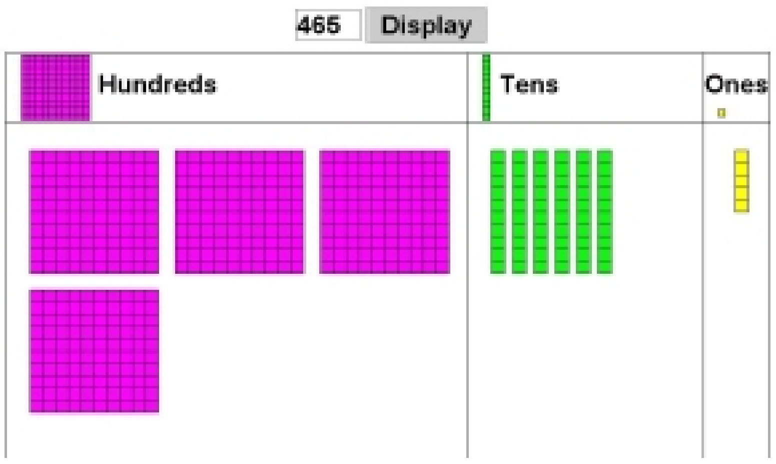
The defining aspect of the methodology is that the program aims to teach students how to perform a mathematical calculation, not the underlying reasoning and concepts of the technique.

Advantages: The largest advantage for the students is that their answers are checked immediately. With normal worksheets the teacher has to photocopy a class set or write the problems on the board and have the students copy them into their exercise books. Then typically the student only receives feedback once completing all of the problems on the sheet. Hence errors are often compounded because they are not discovered till the end.

Teacher's comments: All of the teachers liked this program because they could easily see how to implement the program in the classroom. Students struggling with multiplication could simply practice on the computers independently. There was a large amount of discussion about the placement of the carry boxes. The teachers had each taught their own placements of carry figures, and in all cases this differed from the style recommended in the Department of Education Queensland Years 1 to 10 Mathematics Sourcebook [16]. Clearly all styles would need to be programmed so that teachers can choose the one they prefer; teachers will not use a program if it does not conform to their teaching style as that would confuse the students. (One of the teachers commented that this was a major reason why he didn't use a lot of the currently available software.)

3.5 Conceptual Skill Based

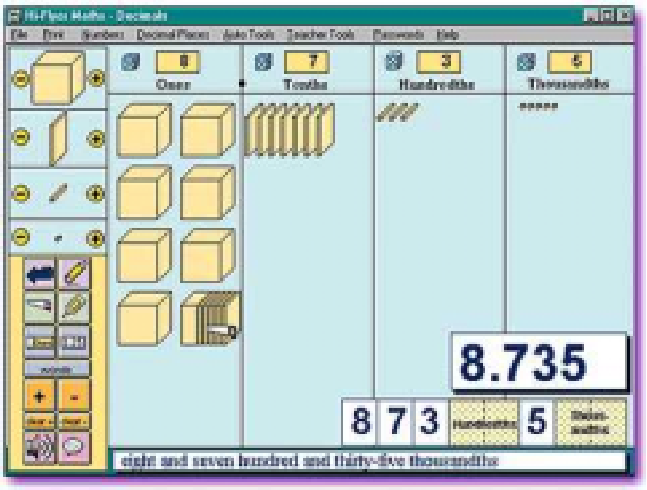
A conceptual skill based program constructs models for representing a mathematical concept. The aim of the *Ones, Tens and Hundreds* program was to mimic the resource MAB blocks (Multi-base Arithmetic Blocks). The conceptual aim was to show how a number, 465 say, is comprised of 4 hundreds, 6 tens and 5 ones; and then show the relationship between hundreds, tens and one through the metaphor of blocks. The program was designed to allow students to explore the concept that the place of a number is important, (ie 200 is not the same as 2) and the concept of adding and subtracting at a visual level.



Units can be added by clicking on the icon in each column heading. Units can be subtracted by clicking on each unit block within the column display. The number is automatically updated. The other option is to simply type a new number in the text field and click on display.

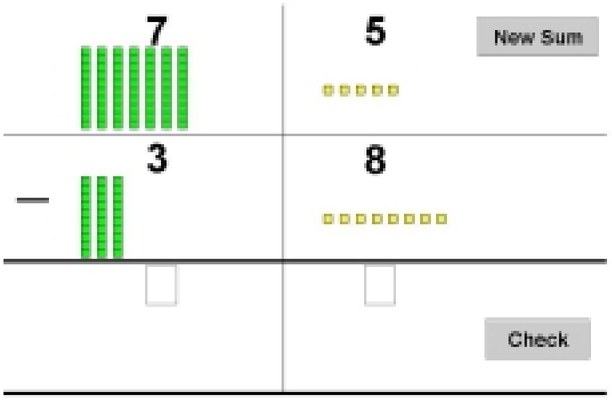
Advantages: This style of programming specifically addresses the area of teaching mathematical concepts. Such programs can provided students with different presentations of the mathematical concepts discussed in class, hence increasing the likelihood of all students understanding the concept.

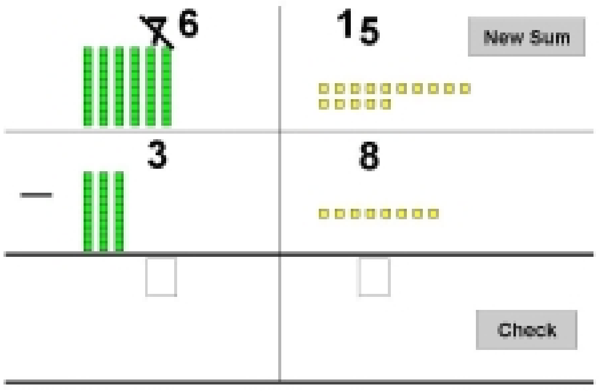
Teacher's comments: All teachers liked this program. No suggestions of improvement or extension were given. The only comment was that one of the teachers had seen a promotion of a similar program. This program, Hi-Flyer Maths Decimals [9], uses the metaphor MAB like the *Ones, Tens and Hundreds* program. Students can saw blocks or glue them together. Also numbers can be read out for the students. The teachers at this particular school didn't use this program regularly in the classroom because the noise generated is distracting for the other students. The students kept on sawing up blocks, rather than doing the set activities.


Hi-Flyer Maths - Decimals
Ones
Tenths
Hundredths
Thousandths
8.735
8 7 3 5
eight and seven hundred and thirty-five thousandths

3.6 Combination of Conceptual and Technique Skill Base

The *technique* program and the *conceptual* program can often be combined to teach the single mathematical skill via both methods simultaneously. Consider for example the program *Subtraction*. The aim of the *Subtraction* program is to combine how subtraction is taught in classrooms with the conceptual aid of virtual MAB blocks. The method of decomposition was modelled in one of the programs.





Clicking on one of the ten-blocks and dragging it into the ones column achieves decomposition.
If the move is not necessary then the ten-block bounces back and the problem is left unchanged.
If the move is needed, then the ten-block is exploded into ten ones and the number in the tens column is crossed out and rewritten as in the screen shot to the left.

Teacher's comments: The decomposition program was the most popular program. All teachers interviewed really like the program. It removed the mess caused by trying to use the MAB, and translated the idea of regrouping in a natural manner by virtually picking the ten block up and placing it in the one column.

3.7 Games

In the context of teaching mathematics, this programming paradigm takes in games of strategies, logic and puzzles.

Games can develop problem-solving strategies. There is a lot of commercial educational gamming software on the market. Two sites that contain some very good mathematical orientated games for all ages are:

- TwoCows: http://games.tucows.pacific.net.au/games.html
- Funbrain.com. http://www.funbrain.com/

There are many more sites and game stores packed with "educational games". Also toy companies like Lego, and TV stations like ABC, are setting up web sites with interactive games, chat rooms for children, etc.

Advantages: Such programs tend to be interesting and fun, hence require minimal supervision because the students are motivated to progress through the program. Also most games can be started and ended very quickly, which is great for filling four to ten minutes of "free time" before a recess break.

Teacher's comments: Typically education games are combined with *skill based* ideologies. Or more accurately put, *skill based* programs are wrapped up in games. An example is Desert Quest [8]. However, more than half of the teachers felt that the students did not learn very many mathematical skills from such programs because the students spent most of the time playing with the graphics, rather than doing the actual problems.

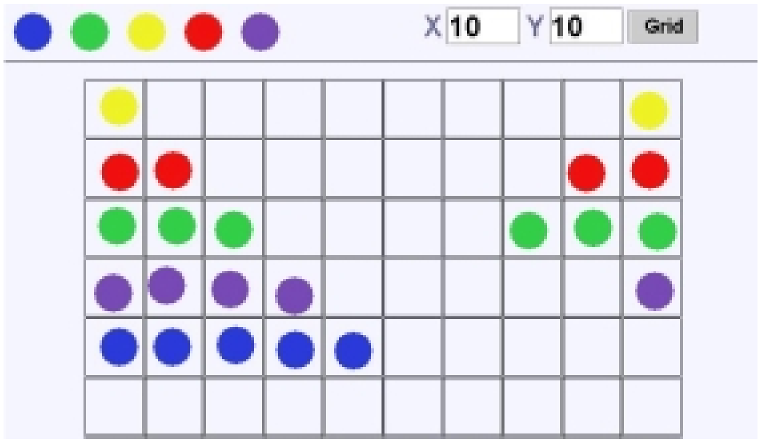
3.8 Resource Based

The aim of *resource based* programming is to supply a cheap alternative to a relatively expensive resources which can be hard to find and obtain. The program aims to replace the physical resource.

The program *Counters* is an example of this design perspective. In *resource based* programming the computer does nothing more than what one could do with the given physical resource. For the *Counters* example, the program performs the same functionality as a piece of paper, pen and a handful of counters. Typically, programs are not purely resource based but are combined with another category. For example *Subtraction* and Hiflyer Maths Decimals [10].

Advantages: Some of the advantages of the program are:

- The physical counters don't need to be handed out and collected, saving precious classroom teaching time.
- The counters won't run out or become lost or end up in peculiar places.
- Students can have access to them at any time.
- Grids of any size can be added or removed instantly saving paper and photocopying.

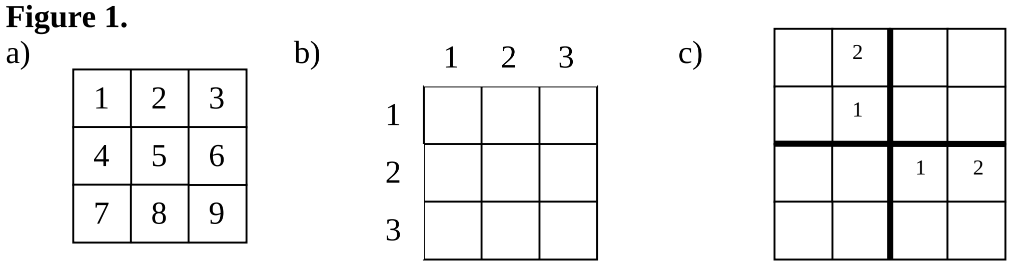


The key advantage to resource based programming is the fact that the programs are *problem independent*. This allows teachers the flexibility of tailoring problems to the specific domain knowledge and needs of individual students. With this type of program students are in control of their own learning; the computer is merely facilitating their learning.

Teacher's comments: This program received mixed reactions from teachers. Two teachers discounted the program as uninteresting. Another two suggested that it would be more useful if it contained set activities. For example a booklet of worksheets which involves the program. It was also suggested that more options like:

a) Numbers in the grid. (e.g. for bingo.)
b) Grid coordinates. (e.g. for mapping problems, blind noughts and crosses.)
c) Axes. (e.g. for coordinate reading.)

Figure 1.



This would allow more activities ranging from completing the pattern in low grades to coordinate mapping in the higher grades. One teacher loved it just the way it is. When shown the program she exclaimed, "great, no mess; counters always get lost all over the classroom when I try activities with them."

4 What we can learn from this

From the teachers' comments it seems that teachers would like to see *skill based* programs which contain *technique* and *conceptual* components, with strong underlying *resource based* metaphors. This is not a surprising result. Hong Kong has already recognised this as an importance area by setting up pilot schools with the emphasis on using computer software in concept learning and consolidation [4], the two main aims of *skill based* programs.

5 Design principles emerging from this study

At a software development level we can convert the teachers feedback into possible starting hypotheses for design principles on how to develop effective mathematics education programs. For example some possible design principles drawn from the analysis of the interviewed teachers' reactions to my eleven programs would be:

- Teachers' prefer programs which are tailored appropriately to users' mathematical skill level, otherwise students become flustered or bored. (Supported by Cocking, et. Al. [6])
- Teachers desire programs which are useful for more than one lesson because they perceive installing and setting up programs as a time consuming tasks and want as much benefit out of their class preparation time as possible.
- Teachers look closely at whether educational software is curriculum compliant because they do not want programs which conflict with what they teach students in the classroom, since inconsistencies can confuse students.
- Similarly, programs should mimic mathematical techniques used by teachers, not just the curriculum, since often teachers do not teach problems exactly how they are set out in the curriculum guidelines.

- The functionality of the interface should need no explanations because teachers do not have time to teach every child how to use the program and instruction booklets tend to become quickly lost.
- The program must go beyond the functionality of existing classroom resources because teachers see computers as extension tools, not as resources to replace what they can already effectively demonstrate.
- Metaphors and analogies should come directly from experienced teachers because they are the experts. They know what does and does not works when teaching students.
- Computers should only be used if they enhance learning. (Bodner supports this with his study where the control group performed better in the final exam than the group with access to computer based learning resources [2].)
- A metaphor should not take over from the primary learning goal.
- Simplicity is always better than complexity. Teachers really liked the fact that the test programs did not have miscellaneous functionality to confuse and distract the students from the learning task.
- Teachers feel that sound and animation should be used in programs with caution because they can be more distracting than useful. (However there is a new type of animation, interactive animation. Interactive animation overcomes some of the problems with typical animation since the user is actively involved in the activity, instead of simply watching a preset sequence. (For some examples see [14] and http://www.utc.edu/~cpmawata/transformations/translations) However, a lot more research needs to be undertaken in the area before we can judge its impact on student learning.)
- Teachers like programs which address one specific concept. It gives them the ability to identify students experiencing a difficulty with a particular mathematical idea and hence present that student with a program which specifically targets the concepts they are struggling with.
- Teachers desire programs which focuses on areas of mathematics which they know students struggle with. They are not interested in programs that simply mimic what they currently teach effectively.
- The way users input their answers into programs should be natural and as effortless as possible. Teachers want the focus to be on solving the problem, not learning the program. (Majewski emphasis this point [13].)
- The time it takes users to input answers should be insignificant compared to the time it takes to solve the problem.

All of these points are teachers opinions that come from a small sample population and hence can not be seen as conclusive. However, they suggest many excellent hypotheses for further research.

Conclusion

This project revealed that teachers feel that computers can be used to teach mathematics in classroom. Unfortunately there is not enough data to conclude *how effective* computers are at teaching mathematics in the classroom setting. (A goal for future research.) The most powerful conclusion which can be draw from the data collected was that:

*Teachers desire **Skill Based** educational programs which contain **Technique** and **Conceptua**l components, with strong underlying **Resource Based** analogies.*

This conclusion, together with the design principles suggested in section 5 will form the basis for our continuing research aimed at developing best-practice guidelines for the creation of the most suitable interfaces for mathematics education software.

References

[1] Anderson, M. Bloom, L. Mueller, U. Pedler, P. (1997) **Graphic Calculators: Some Implications for Course Content and Examination.** Edith Cowan University. ATCM.
[2] Bodner, George. (1997) **Confessions of a Modern Luddite: A Critique of Computer-Based Instruction.** CAL-laborate (Vol 1).
[3] ***Calculus Live.*** John Wiley & Sons.
Available: http://store.wolfram.com/view/ISBN0471317888/
[4] Chow Wai Man, Raymond. Peter Jones. (1999) **Technology in the Proposed Primary and Secondary Mathematics Curriculum in Hong Kong.** Swinburne University of Technology. ATCM.
[5] Chuan, J. (1997) **Studying Synthetic Geometry with Technology.** National Tsing Hua University. ATCM.
[6] Cocking, R. Mestre, J & Brown, A. (2000) **New Developments in the Science of Learning: *Using Research to Help Students Learn Science and Mathematics.*** Journal of Applied Developmental Psychology 21(1):1-11.
[7] Cook, J. (1997) **Basic Mathematical Skills: Are they important?** Glasgow Caledonian University. ATCM.
[8] **Desert Quest.** Greygum Software: Queensland.
Available: http://www.4mation.co.uk/products/dq.html
[9] ***HI-Flyer MATHS Decimals.*** HiFlyer Software: Australia. Available:
http://www.nh.com.au/NH/Html/Edsection/Eddocs/E_Ma/hifly.html
[10] Iguchi, I. & Suzuki, K. (1998) **Improving Junior-High Geometry by using Construction Software.** Tohoku Gakuin University. ATCM.
[11] Jiang, Z. (1999) **Explore Geometry over the Internet Using JavaSketchPad.** Florida International University. ATCM.
[12] Mawata, C. (1998) **Uses of Java Applets in Mathematics Education.** University of Tennessee at Chattanooga. ATCM.
[13] Miroslaw, M. (1998) **Pitfalls and Benefits of the use of Technology in Teaching Mathematics.** Inter-University Institute of Macau. ATCM.
[14] Tsuyuki, S. (1997) **Visualizing Concepts with Interactive Java Animation on Internet.** ATCM
[15] Yanagimoto, T. (1999) **How should the coming mathematics education get connected with the computer? - starting by solving problems within our daily life.** Osaka Kyoiku University. ATCM.
[16] (1987) ***Years 1 to 10 Mathematics Sourcebook: Activities for teaching Mathematics in Year 4.*** Department ofEducation, Queensland.