Experimental Mathematics and Proofs in the Classroom

Ulrich Kortenkamp, Berlin

Abstract: Experimental mathematics is a serious branch of mathematics that starts gaining attention both in mathematics education and research. We give examples of using experimental techniques (not only) in the classroom. At first sight it seems that introducing experiments will weaken the formal rules and the abstractness of mathematics that are considered a valuable contribution to education as a whole. By putting proof and experiment side by side we show how this can be avoided. We also highlight consequences of experimentation for educational computer software.

Kurzreferat: Experimentelle Mathematik hat sich zu einem ernst zu nehmenden Teil der mathematischen Forschung und Ausbildung entwickelt, der immer mehr Beachtung findet. Wir geben Beispiele, wie solche experimentellen Techniken nicht nur, aber auch im Unterricht eingesetzt werden können. Zunächst scheint es so, als würden Experimente die formalen Regeln und Abstraktionen der Mathematik schwächen, die ein eigentliches Ziel des Mathematikunterrichts darstellen. Im Vergleich von Beweis und Experiment stellen wir dar, wie diese Schwächung vermieden werden kann. Weiterhin betonen wir die Auswirkungen dieser Methoden auf Lern- und Lehrsoftware.

ZDM-Classifikation: G90, N70, N90, U70

1 Experimental Mathematics

For centuries Mathematics has been known as a kind of science that is independent of empirical observations. Mathematicians can work at any place, with or without tools, creating abstract worlds of their own in their mind only. However, when a mathematician fixes an axiomatic frame for her[1] studies this leads inevitably to a set of rules that may be considered "laws of nature" for these worlds that cannot be broken. Thus it is feasible to ask for possible ways of performing experiments in these worlds that exhibit coherences for further investigation and deeper understanding.

Exploring these worlds experimentally has become part of mathematical science, as we can see by the creation of institutions like the Centre of Experimental and Computational Mathematics[2] at Simon Fraser University, Burnaby, Canada, or the *Institut für Experimentelle Mathematik* in Essen[3]. Their common theme is that they ask for *systematic ways* of creating experiments that will help in investigating a certain problem. This enabled them to solve long-standing questions, and you can find many examples like the discovery of the BBP formula in the literature, for example in the surveys of Jonathan Borwein (2003).

It is not surprising that it is a rather new trend to use simulations, massive brute-force calculations, or visualization for mathematics, as the necessary technical improvements in computer science are recent. Today it is easily possible to quickly check a formula for several million random values, or to compare an integer sequence against a database of several ten thousands of examples (Sloane). For further details of successful applications of computers and the next major steps to expect we again refer to the overviews of Borwein.

In mathematics education it is now common to use experimental setups for teaching (instead of giving single references, we just mention the series of the proceedings of the workshops of the *Arbeitskreis Mathematikunterricht und Informatik der GDM*), as is the use of computers. Still we claim that the conception of most of the experiments done in the classroom and the ones done in research is fundamentally different. At first glance, discovery activities seem to be experimental activities from the viewpoint of a student, but to us it is more a game of hiding the solution, a game that could create more frustration than comprehension. We will discuss this situation and ask for another kind of activities in the next section.

We cannot replace the abstract and logical concepts that are taught in mathematics by experiments alone. There are educational goals beyond learning arithmetic and geometry. Proving, logical deduction, the ability to clearly state facts and conclusions and to communicate these to others are at least as important as learning to work with numbers. We will address this in section 3.

2 Using Computer Software for Experiments

In disciplines that inherit experimental approaches for centuries there exists a rich set of setups that can be used to demonstrate, explore and understand the laws of nature. Mathematicians and mathematics teachers cannot refer to such setups. In fact, most of them will not even know how to design a mathematical experiment. They are used to work without the uncertainties that are inherent to experimental approaches, and probably they do not like the idea of not being sure about what will happen next.

There is a (more or less historical) reason for this attitude. It is hard to *just try what will happen next*, because one has to carry out lots of calculations, do many symbolic manipulations, draw thousands of diagrams – most of it without getting any usable result. "Poking around" – or trial-and-error – in the set of possible solutions will not yield an answer of any kind in reasonable time, if every try takes a long time. Neither positive nor negative results are likely, and it seems to be a better advise to think harder before trying harder.

As an example, let us consider the task to solve a polynomial equation (and assume we do not know how to do this, and the criteria whether it is possible at all). If we just assume that there is a solution that is a nested square and cubic root expression, we can start enumerating these, either systematically or by random. For each we can search for suitable coefficients that will solve the

[1] Whenever we use the words „his“ or „her“ you can try to replace them by „her“ or „his.“ If the sentence still makes sense, we meant to include both.
[2] http://www.cecm.sfu.ca
[3] http://www.exp-math.uni-essen.de

original equation. For polynomials of degree less than 5 we might eventually succeed, for most polynomials of degree at least 5 we will not. Either way, it will take a very, very long time, and it does not seem like a good strategy of dealing with this task. Actually, just *doing something* is like guessing, and the successful applications of guessing in mathematics are rather limited.

This situation has changed dramatically. The incredible speed of computers (and the speed's insane growth rate) makes *guessing* an option for many problems today. There will be more and more cases where it is just faster to find a solution by (complete or incomplete) enumeration than by thinking of a solution. Brute-force approaches start becoming feasible (without becoming mathematically appealing, though). In the appendix we discuss how this compares to the fact that there are problems known to be computationally intractable.

Still, one should be aware that even if guessing becomes feasible, this does not mean that we do not need thinking anymore. Au contraire, the a priori work (designing the experiment) and the a posteriori work (deducing the results) of the experiment require deep (mathematical) knowledge. We should see this as an opportunity to change the focus of teaching from the arithmetic to mathematics in a more global fashion. Similarly to the paradigm shift that was induced by the introduction of pocket calculators, which made mental arithmetic less important than it used to be, we should use the computer as *a tool that clears the way for the essential.*

If we accept the fact that computer software is a suitable tool to improve teaching, we immediately come to the conclusion that there are some basic requirements for the software. Let us list them as postulates for experimentation software:

First, the software has to be mathematically correct. All calculations and derived displays should be either correct, or they have to be clearly marked as approximations. Today, many users are familiar with the fact that computer software may be "buggy," but still there is a kind of slavish obedience. People tend to believe that everything that is not apparently wrong on a computer screen can be taken as a true fact. We can observe the same behavior, which is very similar to the way television is perceived, for example when the world-wide web is used: If the first web page that Google finds looks reasonably serious, its contents will be taken as hard facts. Another example is the automatic spell checker in today's word processors: If a person is unsure about the spelling of a word, then he tends to believe in the suggestion of the software.

Taking this obedience into account, we see that companies like Microsoft can use their dominance in the software market to change the spelling of words, just by making many people believe that it is correct like the software proposes it. Even worse, companies like Google can replace the information society by a mediocre society that is based on rumors.

The above may sound exaggerated, but it should point out that it is important to take care that software used for teaching and research is either reliable or clearly marked wherever it is not reliable. Otherwise, we are in danger to loose control about what we teach!

Second, the software has to be as easy to use and as accessible as possible. Unfortunately, most people think that scientific profoundness and ease of use are contrary concepts. This led to a lot of wonderful, but basically unusable, though powerful, software products written by mathematicians; in fact, even professional packages like Maple or Mathematica cannot be used intuitively. Most resources of the companies are used to improve the kernel of the software, less are used to enable others to use the kernel.

This might be acceptable from a scientific point of view, but it inhibits the use of the software in educational scenarios. There is not enough time in the classroom to introduce a complex software package in order to perform a single experiment.

We also have to mention here that usability starts with the installation of the software. If we cannot even install the software on a computer, or if it takes a significant amount of time, we take away this time from the total time we have to teach or research.

A more detailed discussion of requirements for mathematical software can be found in Kortenkamp (2001).

3 Experiment versus Proof

The headline suggests that experimentation differs from the rigorous concept of proof as it is usually used in mathematics. Despite that suggestion we want to make clear that experimental techniques do not have to be orthogonal to proving, but they can supplement, enrich and enhance traditional proving.

For a concise overview over the role of proof in the classroom we refer to Reiss and Renkl (2002).

3.1 Using experiments in order to motivate proofs

A well-known problem in mathematics education is the fact that students are not able to identify situations in which they have to prove something. The difference between facts and theorems, trivialities and surprising results, hypothesis and conclusion, or – in algorithmic settings – *input* and *output*, seems to be hidden for people who are not skilled in mathematics. Before we can teach *how* to prove a theorem, we have to teach *when* to prove a theorem.

Using experiments can help, if we take care of the setup. It does not suffice to perform an experiment that will show the desired result, but we will also need an experiment that will contrast the perceived "facts" of the first experiment.

Without going into the details, we just mention the myriads of triangle points that can be found and visualized with geometry software. Without at least one example of three lines that will *not* meet in a common point, there will be no motivation at all to give a proof that another triple of lines does meet. We have to contrast a special situation with the general one to create this desire.

3.2 Using experiments to eliminate proofs

Many times an experiment can eliminate a proof. This is done not by using the experiment as a proof, but by performing an experiment that shows that there is no theorem to proof, because the conjecture is false!

Closely related to proving and disproving by example, we can use experimental techniques for quickly finding the necessary conditions for a theorem to be true at all and to give counterexamples for the other cases. This is a valuable time saver in research, and it can also help to focus the attention of the students to the right cases in the classroom by "pruning the wrong branches".

For illustration, we want to refer to a real-world example that occurred in one of the first year teacher students' lectures at the Technical University of Berlin. The course on elementary geometry is based on lecture notes that are used for at least some twenty years by now. Geometry is introduced in a way that is similar to Hilbert's approach. Starting with the axioms of incidence, more and more axioms are introduced, aiming at a model that is as close as possible to the "usual" geometry that the teachers shall teach to their students in a few years.

Almost all courses like the one described here will introduce the Parallel Postulate at some point– for each line g and each point P there is exactly one line parallel to g through P –, which a crucial step on the path to Euclidean geometry. At the Technical University, we do *not* rely on this axiom, but instead we require that every central dilation is a similarity (which is the case in Euclidean geometry).

The Parallel Postulate can be proven using this axiom. This implies that the axiom cannot be fulfilled in non-Euclidean models. Here we can apply an experiment successfully: In one activity, the students construct the dilated image of a segment in the hyperbolic halfplane model using geometry software.[4] They can observe immediately that dilations map lines to something else, which are not lines. This observation both makes them understand one fundamental difference between Euclidean and non-Euclidean geometries and helps them to remember it.

3.3 Using experiments to make proofs understandable

In some cases it is feasible to replace a proper argument by an experimental fact, that is, when we want to highlight the global structure of a proof. In a complex proof that should be understandable it is not possible to rigorously show every detail. The usual and important technique, in fact, one of the most important contributions of mathematics, is to subdivide the proof into several simpler proofs, creating new lemmas, propositions and definitions. Using the technical armor it should then be easier to follow the proof as a whole. Compare this to the recent work of Hartmann (2003).

Unfortunately, it is not; at least not without more time to digest the additional vocabulary and tools. This leads to teachers (and lecturers) leaving out the details of the proof of a lemma ("easy!" – "trivial!" – "homework!"). Here it is much more honest and helpful to use a quick experimental check for the "easy!"-parts. If it is really not important to know the detailed proof, an experiment will be good enough, if it is important, then we already have a motivating example.

3.4 Using experiments to find proofs

Now we address the most appealing, in our view, way to use an experiment: use it for finding a proof.

It is hard to give general rules for designing experiments that can help in finding a proof. As this is a research grade activity, we cannot expect to be able to find such a general rule at all. But there is at least one technique that we can try to apply in many cases.

The "(*n-1)*-rule" which is described by Weth (2002), but is a pattern that is traditionally known by mathematicians, proposes to prove theorems by relaxing one of the *n* constraints of the hypothesis and looking for the family of solutions for the relaxed system. Weth applies this to geometric theorems using dynamic geometry software. Then the family of solutions is usually given by the locus of an easily constructible point, and we can start by examining this locus.

Creating a locus is a prime example of an experimental activity using dynamic geometry software. We want to illustrate the process of using loci with an example that is based on a discussion thread in the geometry forum.[5]

The original question was:

From: ken@forum.swarthmore.edu
Subject: Regular 4-gon in hyperbolic geometry?
Date: 03/05/2000 23:42:46 MEZ
To: geometry-software-dynamic@forum.swarthmore.edu

```
  Hi,

  I've been trying for a while to construct a
regular 4-gon in hyperbolic geometry (4 congru-
ent sides, 4 congruent angles) and haven't been
able to do it yet.  I'm using Sketchpad with the
Hyperbolic tools from
http://forum.swarthmore.edu/sketchpad/gsp.galler
y/poincare/poincare.html .

  Does anyone know how this can be done (I as-
sume it can)?  Note: the figure will *not* have
any right angles.

  How about regular 5-gons and 6-gons?

  Constructions based on measurements or trans-
formations are fine.  My goal is to come up with
a set of tools (scripts) that carry out these
constructions.

  Thanks.
```

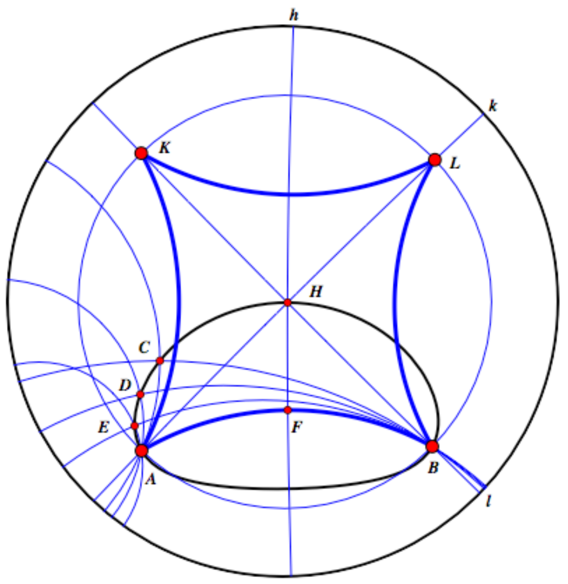
Followed by this a day later:

```
  A clarification: I can already construct an
*inscribed* regular 4-gon, i.e. "regular 4-gon
by center & radius."  I want to find a technique
for "regular 4-gon by edge."
```

Here is a solution, depicted in the Poincaré disk model of hyperbolic geometry:

[4] You can view the construction online at http://www.math.tu-berlin.de/Vorlesungen/WS03/EleGeo/Poincare/Streckung.html.

[5] Now hosted at Drexel University, online at http://www.mathforum.org



Given the segment *AB*, we start by constructing the points *C*, *D*, and *E*. These are the intersections of arbitrary lines through A and their respective hyperbolic perpendiculars through *B*. The black curve is the conic through these five points. The line *h* is the perpendicular bisector of *A* and *B*, and the intersection of the conic and this line is the center of the regular 4-gon *ABKL* given by the segment *AB*.

The correctness of the construction follows from the fact that there is a theorem of Thales type for hyperbolic geometry: all points that form a 90° angle with the points *A* and *B* lie on a common conic.[6] This conic is constructed by finding three additional points on it (*A* and *B* are on it already). The perpendicular bisector is at the same time an angle bisector of the isosceles triangle *ABH*. If we observe that we now know a quarter of the 4-gon, then the rest of the proof is easy and left to the reader.

This construction is not easy to find, although it is rather straightforward once we know that there is a "Thales-conic". The construction was found by Jürgen Richter-Gebert and published in the Mathforum a few days later. He started with the segment *AB* and created the locus of a point that was constructed like *C*, *D*, or *E* above. The locus looked strange in the Poincaré disc, and actually one would not guess that it is indeed a conic section.

However, the dynamic geometry software Cinderella he used for his experimentation was able to display this locus simultaneously in the Beltrami-Klein model, where it looks like a conic, and it also gave the quadratic equation in the construction text. It was only this support by the software that made him realize that he could devise the construction shown above, and it was easy to prove its correctness later.

It is a matter of fact that the mathematics used to address this special problem now have matured and were used to implement general (Euclidean and non-Euclidean) *n*-gons in Cinderella version 2.

[6] A generalization would be that the locus of all points that form any given angle with two other fixed points is a conic; however, this is not true in general: the locus will be a curve of degree 4. Only the special uniqueness property of the right angle causes the curve to degenerate to degree 2.

4 Student Scenarios

We have seen some examples of successful use of experimentation software. Still, it is not easy to give general guidelines for the successful use of software experiments in the classroom.

Here we want to report on a recent experimental type activity that was part of the Advent calendar of the DFG Research Center Modeling, Simulation and Optimization of Real World Processes, Berlin. This Advent calendar is an online calendar[7] that contains a new mathematical problem each day. Students of grades 9 to 12 can solve these questions online. Registered participants will take part in a prize draw among those who had the most and earliest right answers. The number of participants was about 1,000 in December 2003.

On day 14, students were asked to give the maximal number of *halving lines* for a set of ten points. Two points of the set are said to define a halving line, if the line through them cuts the set in two parts of equal size.

The interactive experiment was created using the new API (application programming interface) of Cinderella (Richter-Gebert and Kortenkamp, 2003). With only a few lines of Java-code it is possible to extend the basic functionality of Cinderella to include custom problem formulations like this one. These environments can then be exported to the WWW.

This problem was completely new to the students, and they had no experience at all with problems from computational geometry, so we could not expect that all of them were able to handle the problem. But the outcome was much worse than we expected: Only 50 students had the right solution after two days.

Further investigation showed that the main reason for this disastrous result was a technical one. We did not test the activity on all the platforms that were used by the students, and for many of them the page just did not load correctly – only 150 students were able to submit a solution at all, and not all of them had access to the online experiment. This again shows that technical problems are much more of a barrier than we want them to be. It also highlights the need for professional development and testing.

We also ran in a problem of mathematical correctness. The algorithm that was used for the experiment was not stable for degenerate situations, which made it possible to have more halving lines than the maximal value possible in theory. This could be fixed immediately after it was reported by one of the students.

When we want to avoid such problems, we have to use some resources to create a canon of successfully used experiments on the one hand, and on the other hand we have to spend some resources on reliable and supported framework software that can be used to create such experiments. For the single teacher it is too time consuming and difficult to come up with these himself, and even if he does, then the chance that this material is re-usable by other teachers (or even for other classes of the same

[7] Available at http://www.fzt86.de/Adventskalender

teacher) is almost zero. It is a research problem to create software that is suitable for experimentation.

5 Documentation

The use of computer tools for experimentation immediately leads us to a question that tends to be ignored when new technology is introduced in the classroom: How can experiments be documented in a suitable form, for example for keeping a diary of learning progress?

We just want to give a very brief overview over this highly important topic and refer to the article describing the CINErella extension of Cinderella (Kortenkamp, 2004).

The key point is that the result of an experiment is not only its final outcome, but also the way leading to it. A student should be able to describe the experimental setup and the implementations. It would be ideal if this could be done using the same technical tools that are used for the experiment. This is possible, for example, with Mathematica and its *notebooks*, where intermediate results and calculations can be augmented with text and media. These notebooks are self-contained units that can be used to reproduce the thoughts of the one who performed the experiment.

The main observation is that we have to use the interactive and computing power of computers. We want to use them for more than just audio or video playback, and we have to make sure that we do not lose the content by presenting it nicely.[8]

6 Conclusions and Acknowledgements

Let us collect the necessary conditions for successfully employing experimentation in teaching:

Designing experiments is difficult. Finding the right experiment can be most important thing to learn, but will often be too difficult for the average student.

Mathematical rigor and proofs are not contrary to experience-based mathematics; instead both supplement each other. Mathematical intuition can be build by *doing* math, and proofs will be easier found and better understood when backed by an experiment.

Good software is a crucial ingredient for computer-based experimentation in mathematics. It must be mathematically correct, as bug-free as possible, easy to use and accessible for a large audience. Currently, all software packages have to improve a lot to meet these criteria.

For custom-made experiments, a suitable development toolkit should be available that makes it possible to concentrate on the mathematics and not on the technical implementation. This is also very important for proper documentation of the experiment that can also be used by others.

The paradigm shift and the generally wider acceptance of mathematical experiments will help to create databases of "good" experiments. This is necessary, as no tradition of using experiments exists like there is in other disciplines.

I would like to thank Jürgen Richter-Gebert (in particular for the 4-gon construction and the discussion in Obidos that lead to the Appendix) and Jon Borwein.

Appendix: Solving NP-hard problems in linear time

It is rather uncommon to have proofs of P=NP, one of the Millenium-problems of the Clay institute[9] in an appendix of an otherwise unrelated article, and thus we will not prove it here. On the contrary, we will not give any proof about the relationship of the two complexity classes, but we will just point out that it is reasonable to expect that we can run algorithms with worst-case runtime that is bounded from above and below with an exponential function, which is contrary to common belief. You might consider the proposed approach below cheating, and we will discuss this later to justify it.

As an example we will use the traveling salesman problem (TSP), which is known to be NP-complete in the general case. In the traveling salesman problem we are given a graph G consisting of n vertices V and m edges E. Each edge e of E is weighted with a real number w_e. We are looking for a *round-trip tour* of *minimal weight* that passes through all vertices exactly once, moving from vertex to vertex by following the edges. The weight of such a tour is the sum of all the n edges that are used by the tour.

The brute-force approach to finding the shortest round-trip would be to try all possible tours. This algorithm had to evaluate $n!$ different tours, so its running time would be at least $n!$ multiplied with some constant. It is immediate that its running time is thus bounded by an exponential function in n both from above and below. Unfortunately, in this case, there is no algorithm known yet that is significantly faster; in particular, there is no known polynomial-time algorithm. It is also rather unlikely that there is such a thing, as it would disprove the widely accepted conjecture that P (the deterministic polynomial time algorithms) is a proper subset of NP (the polynomial time algorithms on non-deterministic Turing machines).

Now we claim that we *can* solve the TSP in linear time in real life. We have to assume *Moore's law*: The average processing speed of computers will double every 18 months. This law has been proven to be true for many years by now – please take a moment to recall your first computer and its CPU clock rate. Divide the number of years between the time you got your *first* computer to the time you got your *last* computer by 1.5 and take 2 to the power of this number. Now check whether the ratio of your current computers clock rate is approximately the same number. It should, in most cases.

Once we are allowed to extrapolate from Moore's law into the future, we are done. For simplicity, assume that the running time of the algorithm – measured in instructions that have to be carried out – is bounded from above by $c2^n$, with $c>0$ being a (fixed) constant. We modify the algorithm and get a new one in the following way:

(1) Wait for $3n/2$ years ($18n$ months)

[8] http://www.wired.com/wired/archive/11.09/ppt2.html

[9] http://www.claymath.org/Millennium_Prize_Problems/

(2) Buy a new computer.
(3) Run the algorithm on the new computer.

How much time will this new algorithm take? Step (1) takes $3n/2$ years, Step (2) can be done within at most a week, and Step (3) will depend on the number of instructions per second the new computer will be able to execute. If the current computer you have can execute a million instructions per second (which is highly underestimated), the new one will be able to do $1{,}000{,}000 \cdot 2^n$ instructions per second. This means, that Step (3) finishes within $c2^n / (1{,}000{,}000 \cdot 2^n) = c / 1{,}000{,}000$ seconds. All in all this is linear time.

Where is the cheating? We are changing the complexity analysis from counting instructions to measuring seconds, which is apparently a completely different thing. However, it is a fact of matter that most people do not care about the number of instructions the algorithm will go through, but about the point in time when they will get the result.

Of course, this appendix should not be taken too seriously, but you should keep in mind that "being NP-hard" is not a K.O.-criterion for an algorithm. If we talk about problems that can be tackled experimentally, we can expect that their number is growing. As a last example let us mention the enumeration of reorientation classes of oriented matroids: While it was completely out of reach to do brute-force enumerations of all 11-point configurations five years ago, it is now more-or-less easy to do, and we can start to think about 12 points. In another five years, we should be able to think about 13 points, which means that we can do n points in about $5(n-12)$ years. Not fast, but linear time – you get the point.

References

Borwein, J. (2003). Mathematics by Experiment: Plausible Reasoning in the 21st Century, October 2003, available online at http://www.cecm.sfu.ca/personal/jborwein/rsc-talk.pdf

Elschenbroich, H.-J. (2003), Funktionen dynamisch entdecken, *Bericht über die 20. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der GDM* (pp. 43-54). Hildesheim: Franzbecker.

Gawlick, Th. (2003), DGS als Trägermedium für interaktive Arbeitsblätter in der Differentialrechnung, *Bericht über die 20. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der GDM* (pp. 54-66). Hildesheim: Franzbecker.

Hartmann, M. (2003), Formen multimedialen Lehrens – ein Vergleich, *Bericht über die 20. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der GDM* (pp. 54-66). Hildesheim: Franzbecker.

Hartmann, M. (2003), Steigerung der Effektivität multimedialer Lernumgebungen durch Pop-up-Ikonogramme, *Beiträge zum Mathematikunterricht 2003* (pp. 277-280). Hildesheim: Franzbecker

Heintz, G. (2003), Einsatz von DGS am Beispiel von *Cinderella*, *Bericht über die 20. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der GDM* (pp. 54-66). Hildesheim: Franzbecker.

Kortenkamp, U., Richter-Gebert, J. (1998), Geometry and Education in the Internet Age, *Proceedings of the ED-MEDIA & ED-TELECOM 1998 World Conference on Educational Multimedia, Hypermedia and Telecommunications,* Freiburg: AACE. Online: http://www.cinderella.de/papers/geo-i.pdf.gz

Kortenkamp, U. (2001). The Future of Mathematical Software, *Proceedings of MTCM 2000*. Heidelberg: Springer-Verlag.

Kortenkamp, U. (2004). Algorithmen, Dokumentation, und der Einsatz neuer Medien im Unterricht, in preparation for: *Bericht über die 21. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der GDM,* Hildesheim: Franzbecker.

Oldenburg, R. (2003), Feli-X: Ein Prototyp zur Integration von CAS und DGS, *Bericht über die 20. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der GDM* (pp. 54-66). Hildesheim: Franzbecker.

Reiss, K., Renkl, A (2002), Learning to prove: The idea of heuristic examples, *Zentralblatt für Didaktik der Mathematik* (Vol. 34 (1), pp. 29-34), Karlsruhe: FIZ.

Richter-Gebert, J., Kortenkamp, U (1999). The Interactive Geometry Software Cinderella, Heidelberg: Springer-Verlag.

Richter-Gebert, J., Kortenkamp, U. (2003). Beta-Release of Version 2 of Cinderella, available on request by email to the author.

Schumann, H. (2003), A dynamic approach to 'simple' algebraic curves, *Zentralblatt für Didaktik der Mathematik* (Vol. 35 (6), pp. 301-316), Karlsruhe: FIZ.

Weth, Th. (2002). Die *n-1*-Stragie, *Tagungsband zum Nürnberger Kolloquium zur Didaktik der Mathematik 2002 (proceedings)*, available online at http://www.didmath.ewf.uni-erlangen.de/Vortrag/vortrag02/weth_n-1Strategie.pdf

Author

Kortenkamp, Ulrich, Prof. Dr., Mathematisches Institut, Technische Universität Berlin, Sekr. MA 6-2, Straße des 17. Juni 136, 10623 Berlin
Email: kortenkamp@math.tu-berlin.de