GUIDELINES FOR USING COMPUTERS CREATIVELY IN MATHEMATICS EDUCATION

ULRICH KORTENKAMP

ABSTRACT. Computers are the ultimate tool for teaching: They provide microworlds that can be explored; they can be forgiving and strict; they take account of everything that's happening; they can create maps of what the learner knows or still has to learn; they are fun to use; they provide interactive illustrations that enlighten those who use them — or at least, that's what they could be or do. There are significant drawbacks: many people do not know what they could do with them; if they know it, they probably do not know how to make them do it; and, finally, it is not validated whether it is of any didactic value to use them in the way they are used.

In this text, we present some basic guidelines that can help in designing and evaluating electronic material. Starting from an example, we identify some roles the computer can take and discuss their implications for the development of mathematics tools.

1. INTRODUCTION

Claim 1. *Using a computer is by no means a guarantee for better teaching.*

Claim 2. *Using a computer is a chance to significantly enhance teaching.*

Today, everyone — students, parents, government, university administration, just to name a few — require the use of computers in teaching, as they do not want to miss the chance for the enhancement of Claim 2. Unfortunately, taking Claim 1 into account, we are only left with a faint hope. As many teachers and professors neither want to use the computer nor know what to do with it, if we add the technical insufficiencies of many computer installations for teaching, we even might end up with

Claim 3. *Using a computer can be hazardous to teaching.*

This article intends to identify the advantages of new media and make some of the underlying mechanisms more visible. If you intend to (or have to) use computers in teaching, you should checkmark the framed boxes below and reflect on each of them in your context.

2. EXEMPLARY USE OF A COMPUTER IN TEACHING DISCRETE MATHEMATICS

The material that is introduced in this section has been used for students of grades 4 to 8; nevertheless, it could be adapted to university teaching as well, as it is concerned with an important basic topic from Discrete Mathematics. This

Key words and phrases. Dynamic Geometry software, discrete mathematics, teaching, Visage, new media.

Supported by the DFG Research Center MATHEON.

FIGURE 1. A screenshot of an applet used to teach Eulerian graphs and tours to 13-year old students. The general topic of the sequence is "What is the tour of a garbage collection vehicle?" Students can draw a graph onto a map of their hometown Schriesheim and ask for a Eulerian tour through it.

activity is intended to be used for the introduction of the concept of graphs, and to explore the notion of Eulerian tours.

The general topic is the optimization of tours for garbage collection. Based on the observation that the garbage collection vehicle has to go through every street at least once — which is also sufficient, as two garbage collectors serve both sides of the street — students explore tours that do not use any street twice. First, they work on an electronic exercise sheet that shows a city map of their hometown (Fig.1). They can add vertices and edges of a graph using the mouse, and they can run an algorithm that tries to find a round-trip tour that visits all edges exactly once.

The electronic exercise sheet is an HTML page with an embedded Java applet. The Java applet was created using the Visage-Extension [2] of Cinderella [4]. This exercise can run standalone from a local disc or CD-ROM or it can be put on the web. Besides a Java runtime, which is included by default on major operating systems, no software installation is necessary.

As a next step, we let the students work with pencil and paper. They are supposed to check by hand whether a given graph admits a Eulerian tour, i.e. a closed tour that visits every edge exactly once.

The solutions will not be checked by the teacher, but by the students themselves. They use the next electronic work sheet to draw the graph and ask for a Eulerian tour. If there is one, the computer will show one, otherwise the algorithm will fail and highlight a problematic part of the graph. In Fig. 2 we can see a graph drawn by a student.

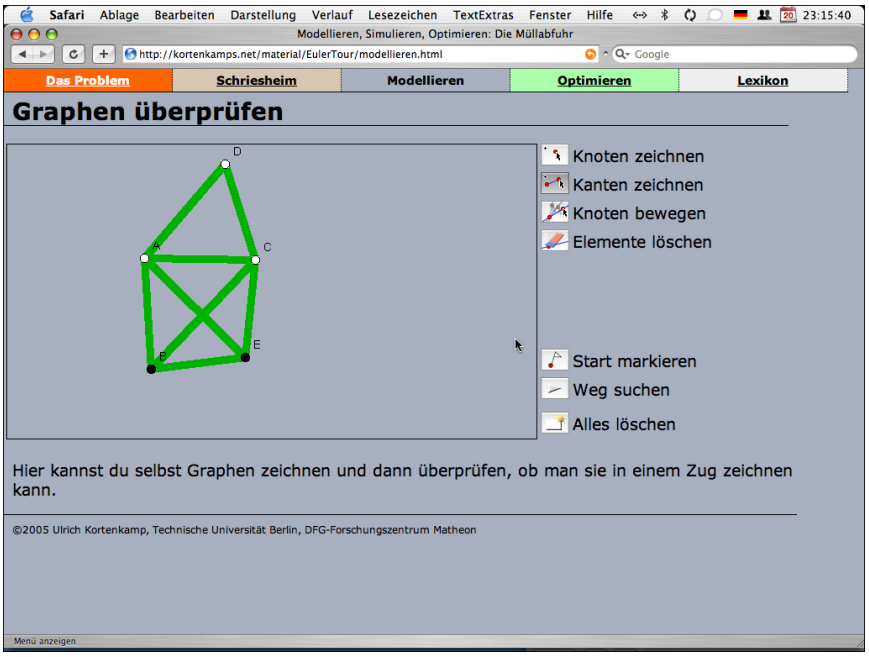

FIGURE 2. The open learning environment for drawing and testing graphs. On the left hand side we see a 5-vertex graph drawn by the student. The coloring of vertices is done automatically (see text). On the lower right there are buttons for selecting a start vertex and starting the algorithm that finds Eulerian tours.

During this phase students naturally will ask themselves[1] whether there is an easy way to distinguish between Eulerian graphs and non-Eulerian ones. The answer to this question is positive — any graph that contains only vertices of even degree, that is, has an even number of edges incident to it, is Eulerian, and only these. The proof of that fact is easy, and can be done by an induction argument, but discovering this conjecture is not that easy.

In this activity we have added a subliminal clue for arriving at that concept. The vertices in Fig. 2 are colored automatically in either white or black, depending on their degree. White vertices have an even degree, black vertices have an odd degree. Our experience is that students will not notice the coloring at all at first. However, when they investigate possible reasons for some graphs to be “good” and others being “bad”, they can use this coloring as a first hint: Graphs that were proved being Eulerian (using the built-in algorithm) have only white vertices.

The last part of the sequence is another electronic work sheet (Fig. 3) that opens the problem further and can be used a starting point for other, related activities. Students are given a graph that contains both black and white vertices, and their task is to change the color of all vertices to white. This is always possible — a theorem about graphs states that every graph contains an even number of odd-degree vertices, so we can connect pairs of them with paths. As the students do not know the theorem yet, they are invited to play a game where they should try to give a graph that their classmates cannot complete to a Eulerian graph.

[1]If they do not ask themselves, the teacher should activate this discussion

Building on this, we could either pursue different directions, or bring this sequence to an end. Possible continuations are, for example, proofs by induction for various graph properties, shortest-path algorithms, or (weighted) matchings in graphs.

3. Roles of the Computer

We will now discuss the various aspects of computer use in this example. We will use the term *computer*, even though we mean all computer-like media and, in particular, appropriate software. The first two roles are evident, but we want to mention them nevertheless.

1. Computers can be used as a motivating element.

It is widely known that using a computer at all can have a positive effect on the motivation of students. However, that might easily change in the future when using a computer becomes a day-to-day event, so we should not rely on it in our teaching.

2. Computers can provide or enhance the visualization of concepts.

Visualization and interactive presentation of content has become much easier with a computer. In our example, it is really easy to demonstrate a tour on a city map in a professional way, something which needed much more preparation (like preparing many hand-drawn slides in advance) before. As most teachers are aware of this publishing aspect of computers, we will not focus on it here, but instead we will concentrate on the following, not so apparent issues.

3. Computers can restrict the actions of learners and thus help them to develop appropriate mental models of representation.

At first thought, this might appear a drawback of the computer, and not an advantage. However, even in an open learning environment, it is not desirable that students be able to do anything. There might be situations in which it is possible to work completely unrestricted, but usually this is not the case. If we create really open situations, we have to accept that the students might not reach the original goal (which is, in the above case, to learn about the concept of graphs for modeling). If we accept that, the teaching might still be instructive and good, but we risk that we — the students and teachers – end up with unsatisfactory semi-results. I strongly disagree with the opinion that it is sufficient to think about a problem and find some result. This assumes that anybody has both the creativity and intellectual capacity to solve all problems. Even if students work in groups, this is not true, and one of the great advantages of humanity is the ability to store and recall others' inventions using speak and writing.

How is this implemented in the Euler Tour example? Students are allowed to draw on the city map using the computer. Of course, they could also use a pencil on a photocopy of the city map, but then they could draw anything, not only graphs composed of vertices and edges. The difference between abstract model (the graph) and drawing (the city map) could not be established that way — the graph is just another drawing.

4. Computers can give an immediate hands-on experience on abstract models.

The graph that is drawn on the city map is "used" directly. This creates the important immediate feedback for learning. With traditional methods, we had to rely on the students knowing how to find the right tour in advance for checking; if that were the case, we would not have to teach them. Using the computer, they can try out many more examples and get immediate, correct feedback.

5. Computers can act as referees.

It is difficult to supervise a group of more than 20 students who work with a computer. In particular non-homogeneous groups pose a problem: it is not possible to check with each student his or her individual progress and give advice or hints. If we fall back to addressing the whole class, we lose the fact that each student can settle for his own learning method and speed. Or, if we restrict checking of exercises to giving the answers (either by the teacher or by a student), some students can be left with unanswered questions: Why is it wrong — or even, why is it correct what I did? The necessary temporal synchronization between the students takes away the time to answers these questions.

With the help of the computer, it is possible to overcome this. The computer can act as an authority for right or wrong, and it can even show why something is wrong. In the garbage collection example, we exhibit this by using the computer for checking the students' answers. When a graph is Eulerian, the computer can prove this by visualizing the corresponding tour. When it is not, the software will highlight problematic areas of the graph to encourage further inspection.

Also, this allows for other teaching scenarios. For example, students can play a two-person game: One person is thinking of a graph, the other person has to decide whether it has the Euler property. Without the help of the computer, we cannot be sure that the answers are correct, putting the whole educational effect of the game at risk.

6. Computers can give hints and guidance.

In the example, we exploit the coloring of vertices for giving an almost subliminal hint. Our experience shows that even people who know that the evenness of the valence of a vertex is important don't immediately recognize the reason for the coloring. It's just something to think about, a crystallization point for thoughts, a thought-provoking impulse. In addition, it gives a temporary nomenclature for a property of vertices — students can talk and think about black and white vertices instead of vertices of even- and odd degrees. This constitutes a significant aid for developing the right concept.

7. Computers make it easy to customize material for a certain audience.

There is no doubt that computers with their word processing and desktop publishing tools made it possible for everybody to produce material that looks professional. If one has access to an electronic version of a work sheet, then it is easy to change some values, or to exchange a figure or drawing by another one. For teachers, this implies they can use their colleagues' material even if it is not exactly fit for their course. An enormous amount of preparation work can be re-used for other purposes.

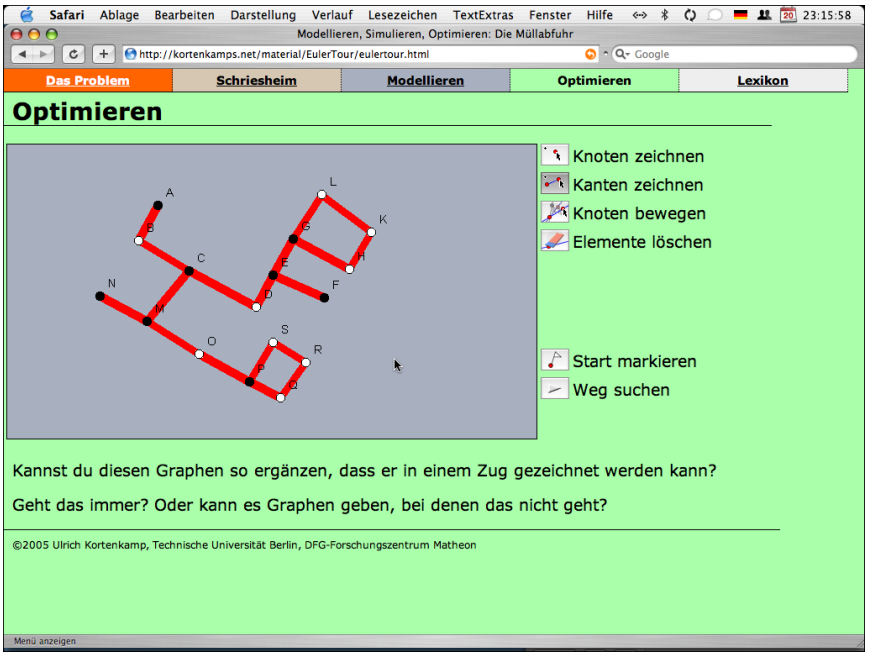

FIGURE 3. The electronic work sheet for "optimizing". The German instructions are: *Is it possible to extend this graph to make it possible to draw it without lifting the pen? Is this always possible (for any graph)? Or are there graphs that do not permit it?*

The same is true for electronic material. In our example, it is easy to replace the city map with the map of another town, or a subway plan, or a milling plan for a printed circuit board.

Also, the hints described in the previous section can be omitted or changed, in order to adapt to the abilities of the students. This can be done individually for each of them.

8. Computers can show the need for analyzing the structure of a problem.

The last general role of computers we want to bring up here is another general property of introducing algorithmic thinking into schools. If we are really looking for a way to use the computer to solve a problem, we have to break it down into small parts, that can be solved by basic building blocks.

In the example sequence, we used this only implicitly by using graphs as a model for street maps. If time permits, this can be explored further by examining the Eulerian Tour algorithm. What data does the algorithm need? Which decisions have to be made? It all comes down to marking edges and finding unmarked edges incident to a vertex. If we arrived at this conclusion, we can see that the graph model is perfect for answering questions like this.

4. IMPLEMENTATION ASPECTS

We cannot rely on the fact that every teacher is both willing to and able to create electronic activities him- or herself. Quite the contrary, most teachers depend on using material that has been made available to them by third parties (i.e., software manufacturers, government, or colleagues, to name the most important ones).

This section addresses both software manufacturers and software users. We highlight three important aspects of software that have to be taken into account when creating or evaluating software products.

4.1. **User Interface.** The user interface of a piece of software is a key component. If the software is not accessible for one reason or another, then the "inner values" cannot be revealed and are irrelevant. While this seems to be common sense, many educational software products still come with poor human-computer-interfaces.[2] This renders them useless, at worst.

It is relatively easy to check the user interface of a software product — much easier than creating it in the first place.[3] Here are some rules of thumb for judgment.

- Is the functionality of the software carefully selected, instead of just complete? Is the toolbar (if any) cluttered?
- Are there default choices in the configuration that apply to your situation?
- Does the interface comply to the user interface guidelines of the operating system you are using, i.e., is its handling and look-and-feel similar to other software on your computer?
- Has somebody taken care of making the software look visually appealing?
- Is it possible to adapt the software to your needs (see Box 7. above)?

You will probably be able to add to this list.

Unfortunately, most open-source-software[4] fails the tests above — and this is related to the answer to the meta-question: *Is anybody aware of the issue and responsible for the human-computer-interface?*

4.2. **Modularity and Programming Interfaces.** Even when using new technology for teaching, the teacher is still in charge of providing the content and material. A software company cannot anticipate the special needs for all teaching/learning situations; if so, then it would be easy to provide a teacher-less package for online teaching.

This implies that a teacher has to be able to choose from the available material and rearrange it according to the pedagogical situation. Monolithic blocks of content are not suitable here. Every bit should be re-usable.

Also, there should be a way to add to the material as described in the previous section. The black-white-hints were not built into the software package used, but they are a custom add-on done by the teacher himself. As it is completely unclear what other ideas a teacher could have, there has to be a general way to extend the software, i.e. a programming interface (or API).

Again, we want to list a few rules of thumb for a first evaluation:

- Is it possible to use parts of the software independently or is it a all-or-nothing decision?

---

[2]The same applies to scientific talks. Although most people know that the scientific value of a talk cannot be recovered if its presentation is poor, they seem to rely on the inner qualities and neglect the performance.

[3]An example is the toolbar for dynamic geometry software (DGS) for which there is no good solution so far. See [3] for a discussion about how to move to a zero-interface for geometric constructions.

[4]This is by no means a vote against open-source-software. We just want to emphasize that this is one of the areas which definitely needs some progress in the next years.

- Does the software use standard document formats like HTML or PDF, or is a separate viewer necessary that might not be available for all (current and future) platforms?
- Is there an API for custom extensions? Is it easy to learn?
- Does the software license allow for free redistribution of content created with it?

4.3. **Mathematical Foundation.** As we are focussing on mathematical software for mathematics education, we should not forget that we want to teach *mathematics*. This creates a dividing line between general-purpose software like media players or animation tools and special-purpose software for doing mathematics like spreadsheets, DGS or CAS.

Media players are not aware of the content they are playing, and there interaction capabilities are restricted to linking pre-produced content to certain choices. We can have a wealth of material that is interconnect by clickable links, but still this is finite and can hardly respond to every aspect that might come up during teaching. Only very closed teaching situations can afford this type of material, unless it is meant as accompanying media, e.g. material that is used as a reference.

In order to support explorative and experimenting learning, the software has to implement the mathematical concepts as exactly as possible. To give a clear-cut example, we refer to K-2 teaching: A calculator for one-digit additions can be implemented using a table that stores all problems and their solutions. Using it is similar to using a media player — we press the channel number (for $3+5$ we choose "channel 35´´) and get a pre-produced unit about adding three and five. A student who wants to explore addition more in-depth cannot choose a channel for $10+1$ as the calculator was not designed for this. An open software environment would have included the mathematical theory for addition and would be able to do all possible additions.[5]

This example may be exaggerated, but many educational software products today fail in this category! Everything which might be outside the current scope for the lesson has been omitted, as it leads to higher production costs. This is caused by the common approach to do an implementation that works for most cases, but not in general, and to "fix" all the cases that are relevant for teaching — at least those that the software developer did think of.

Let us conclude again with the rules of thumb for software evaluation in this category:

- Is the software just playing media or does it use a structural representation of the mathematical content?
- Are special cases handled by a general theoretic approach or using a series of "if-then-statements" within the software?
- Is it possible to try experiments that were not foreseen by the software developer?
- Is unexpected user input handled as an error, not at all, or in a reasonable way?

[5]See [1] for a discussion of this in the context of Dynamic Geometry software.

5. CONCLUSION

As we have seen, there are many ways of using a computer to enhance teaching, some of which are well-known, while others are much subtler. Our contribution here was to identify a few concepts, generalize them, and make them accessible for other situations.

In Sec. 3 we discussed three main components of software development that are areas worth of inspecting when evaluating (or implementing) software. In order to support the roles of the computer identified before, these areas constitute key components.

6. ACKNOWLEDGMENTS

I would like to thank for the opportunity to present this work at the 1st KAIST International Symposium on Enhancing University Mathematics Teaching in May, 2005. Parts of this work have been supported by the DFG Research Center MATHEON in Berlin. Many thanks to Dirk Materlik for his work within the Visage project, and to Brigitte Lutz-Westphal for fruitful discussions.

REFERENCES

[1] Kortenkamp, U. *Foundations of Dynamics Geometry* Dissertation, ETH Zurich, 1999. http://kortenkamps.net/papers/1999/diss.pdf

[2] Kortenkamp, U. and Materlik, D. Visage. A software package for visualizing graph algorithms using interactive geometry software. See http://cinderella.de/visage.

[3] Kortenkamp, U. and Materlik, D. *Pen-based input of Geometric Constructions.* Proceedings of MathUI 2004, http://kortenkamps.net/papers/2004/Scribbling-article.pdf.

[4] Richter-Gebert, J. and Kortenkamp, U. *The Interactive Geometry Software Cinderella.* Springer-Verlag, Heidelberg, 1999, http://cinderella.de.

TECHNISCHE UNIVERSITÄT BERLIN, DIDAKTIK DER MATHEMATIK, STRASSE DES 17. JUNI 136, 10623 BERLIN

E-mail address: kortenkamp@math.tu-berlin.de