ACTA DIDACTICA NAPOCENSIA

Volume 3, Number 2, 2010

CREATING INTERACTIVE USER FEEDBACK IN DGS USING SCRIPTING INTERFACES

Andreas Fest

Abstract: Feedback is an important component of interactive learning software. A conclusion from cognitive learning theory is that good software must give the learner more information about what he did. Following the ideas of constructivist learning theory the user should be in control of both the time and the level of feedback he receives. At the same time the feedback system must identify and review different possible solution strategies in an open learning environment.

The interactive geometry software Cinderella offers an easy-to-use programming interface. It can be used to implement application specific feedback by the author of learning units.

In this paper we present two exemplary learning units implementing two kinds of interactive feedback: feedback on demand and immediate feedback. The presented units come from discrete mathematics and from the theory of line reflections and congruencies in geometry. The units are implemented in a process-oriented design. Various directly given or hidden hints help the students to understand the mathematical principles behind the given problems. Our tools analyses the student's solution processes automatically and generates additional feedback on demand.

The second learning environment can also be used in conjunction with recording of user actions. This allows additional feedback given later by the teacher whenever the automatic feedback system fails in analyzing the users' learning processes. First experiences using the units in teaching are presented.

Key words: Dynamic Geometry, Scripting, Individual Feedback, Graph Algorithms, Congruencies

1. Introduction

The idea of guided discovery learning seems to be a good concept for teaching mathematics, see e.g. Mayer [7], Orton [12]. Students work on certain mathematical problems and develop their own solutions. They are free in choosing their own individual learning paths, their solution strategies and the tools they use. This should result in a deeper understanding of the mathematical structure of the given problem. The teacher should accompany the students and supports their learning process. He attends the students, gives hints, stimuli and feedback.

Nowadays those ideas are well known and more or less established in school teaching. But at the university level the situation is still different. Due to the large number of students, the teachers often are not able to attend the whole learning process of each student. They don't have enough time to accompany each student. As a consequence, classical lectures are often chosen as the main type of lessons at universities.

Our approach is to use interactive learning software with an intelligent feedback system for exercises in addition to the classical course material. The students work on given mathematical problems or questions using the computer. According to the ideas of the constructivist learning theory suitable learning software should open the door for individual learning paths to the students (Schulmeister

---

This paper is an expanded version of the presentation given to the second Computer Algebra and Dynamic Geometry Systems in Mathematics Education (CADGME) conference at the University of Linz, Austria, in July 2009.

Published: 1 June 2010.

[14]). The software analyses the solution process of the students and gives individual feedback that offers more information than just a “right” or “wrong”. For an optimal support of the student’s learning, the software should analyze the whole learning process and not just the results.

Especially in mathematics there often is an innumerable amount of possible solution strategies. It is not possible for the author of an electronic learning tool to predetermine each of those solution strategies. Also it often is impossible to validate all of them automatically. But in fact this is no problem in our approach. Most students follow one of a few common solution strategies. Even the mistakes students make are often the same and are based on the same misconceptions. By detecting those common solutions automatically and removing them from the set of all students’ solutions, usually only a few special cases remain that can be handled manually by the teacher. This idea is called a semi-automatical assessment (Bescherer et al. [1]).

This concept can be applied to many learning environments. Bescherer and Spannagel [2] formulated it as a general design pattern in mathematical learning environments. In particular, the *Feedback on demand*-pattern can be applied to educational mathematical software. In this pattern it is claimed, that the students should get feedback on their work whenever they need it. Especially for interactive software it is proposed to record and analyze the whole learning process. This can be done either by a generic tool like Jacareto [13],[15] or by the learning software itself. The software should detect standard solutions (including standard mistakes) and present an individual process oriented feedback for those solutions to the students. Exceptional solutions are also detected automatically. The teacher gets a notification on those non-standard solutions for analyzing them manually. As a consequence, the teacher is relieved from investigation of common standard solutions. He or she gains more time to analyze interesting exceptions and to discuss the most important problems with the students. Therefore the teacher should have access to all recorded processes.

In the project *SAiL-M: Semi-automatic Analysis of individual Learning Processes in Mathematics*, we develop among other thing prototypical software implementing the Feedback on demand-pattern. In section 5 an example developed in this project is presented.

2. Classification of Feedback

Feedback is an important benefit of interactive learning software. While a human teacher is not able to give extensive feedback to each student whenever he or she needs it, the computer can generate continuously feedback on the student’s learning process. Hence the theory of feedback is an important area in the theory of hypermedial teaching and it is as old as the development of educational software. Schulmeister [14] gives a short survey on the different types of feedback used in hypermedial learning systems.

To distinguish between the different kinds of interactive feedback, we use the following characterizations based on several aspects how feedback can be given.

Timing of feedback. The time feedback is given to the student can be very important for the success of a student’s learning process. Feedback can be given immediately after each user interaction, delayed, on the learner’s demand, after a completed session or at the end of the whole learning process. (cf. Cohen [3]).

Classical teaching scenarios tend to generate feedback a long time after the learning of a student. A human teacher is correcting the student’s homework afterwards. Exams are a tool to get feedback at the end of the learning process. Often, this is too late to influence the students learning process. On the other hand, an immediate feedback given continuously given by the software can lead the student to a try and error strategy for the problem solving process. Hence, we prefer the feedback on demand approach in which the student can ask for feedback whenever he or she needs it.

Information content of feedback. Pridemore and Klein distinguish between a “verification feedback” and an “elaboration feedback” [10]. While “verification feedback” just informs about the correctness of a solution, an “elaboration feedback” presents the correct solution and an explanation. In our view sometimes it also may be beneficial to show only partial solutions, so there are more than two levels fort hat dimension of feedback.

In contrast to the idea of Pridemore & Klein, elaboration feedback should not show and explain the right solution, but it should analyze the learner's solution attempt. Nevertheless, showing a right solution might be a helpful feature in some cases. As an enhancement of classical feedback engines that give feedback only based on the results of the learner, we prefer to analyze the whole learning process of the student.

Finally, we like to distinguish between directly given feedback and hidden hints, which should help the student to improve his or her solution on their own.

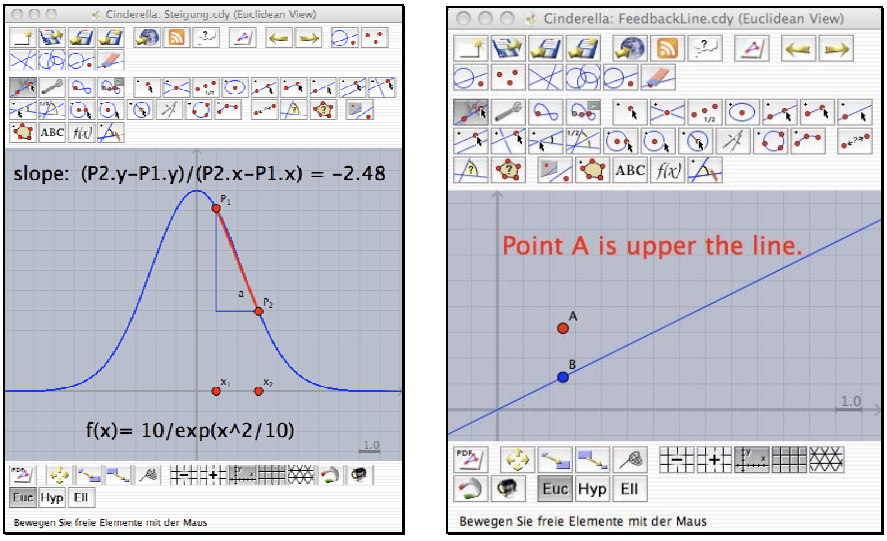
Presentation of feedback. Lastly, there are different ways to present feedback to the learner. Feedback can be visually or acoustically. Feedback might be given in a graphical (iconic) or a textual form. Visual feedback can be presented animated or statically. According the theory of multimedia learning a combination of different presentation types leads to better learning success (cf. Mayer [8]). Park and Gittleman claim that animated visual feedback can superior any static type of feedback [9].

3. Scripting in Dynamic Geometry

A scripting interface is a very important feature for mathematical software. Especially in mathematics there is a wide range of possible applications for corresponding software, but the developer of the software can't think about all of them. Hence the possibility for enhancements of the software by the user is essential. A scripting interface is the easiest way to implement such extensions. Kortenkamp & Fest discussed the necessity of and the opportunities opened by connecting modern geometry software with a programming interface in [5].

Nevertheless only a few Dynamic Geometry systems offer such an interface. The Interactive Geometry Software Cinderella by Richter-Gebert & Kortenkamp [11] has a built-in scripting language called CindyScript since version 2.0.

CindyScript is an easy-to-learn programming language that already was used successfully for students' programming at secondary school (see [4]). The syntax of this functional language is near to the syntax of the Mathematica programming language and hence very close to a formal mathematical language. This allows an easy transformation of mathematical expressions into programming code for executing numerical calculations inside the geometric environment of Cinderella. CindyScript supports a direct interaction with each geometric object in the construction plane, is event driven, offers different drawing functions, and allows the user to include algorithms that will be applied to the geometric objects.



Picture 1. *Using scripting for visualization:* a) slope of secant, b) reducing information

For example, a visualization of the slope of the secant of a function graph (see Picture 1a) can be implemented with the following steps: define a function by

```
f(x) := 10/exp(x^2/10);
```

Draw its graph:

```
plot(f(x));
```

Add two points `P1` and `P1` and fix them on the graph:

```
P1.y=f(P1.x);     P2.y=f(P2.x);
```

Connect the two points by a segment `a` and calculate the slope `s` of segment `a`:

```
s=(P2.y-P1.y)/(P2.x-P2.x);
```

Print the result:

```
drawtext([3,10],"slope: "+s);
```

Using the features of the CindyScript interface it is also easily possible to reduce information in special situations to draw the focus of the students' attention to a specific important aspect. An example is shown in Picture 1b. With just one line of script code the position of a point relative to a line is described:

```
drawtext([0,6],"Point A is "+
                        if(A.y>B.y, "upper",
                        if(A.y<B.y, "below", "on" ))+" the line.");
```

The color of the point A can also be applied quickly:

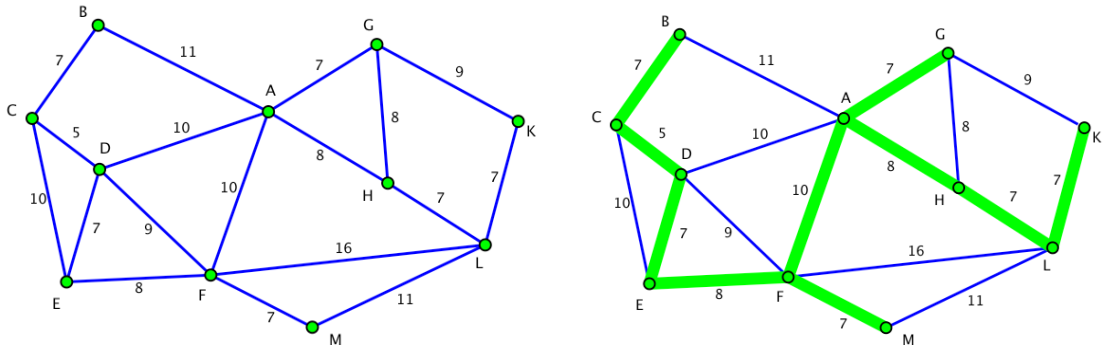
```
A.color=if(A.y>B.y, red(1), if(A.y<B.y, green(1), blue(1) ));
```

This could be the core of an electronic exercise with an interactive immediate visual feedback. This shows that only a little programming knowledge is necessary for authoring interactive mathematical worksheets with computer-generated feedback by using the CindyScript programming interface. In the following we will show two examples of electronic worksheets providing such interactive feedback implemented in CindyScript.

4. An example from Discrete Mathematics

Our first example arises from the area of discrete mathematics. According to the Berlin curriculum for secondary schools at level 7/8 [6], we implemented an exercise for the module "Diskrete Strukturen in der Umwelt" ("Discrete structures in the environment") on the topic of Minimum Spanning Trees (MST).

The problem setting is the following: Given is a graph with nodes and connecting edges. To each edge a positive number – called edge weight – is assigned. The aim is to find a cycle-free selection of edges connecting all nodes pair wise such that the total sum of all edge weights is as small as possible. Picture 2 shows an exemplary graph and its minimum spanning tree.



Picture 2. *A weighted graph and its minimum spanning tree.*

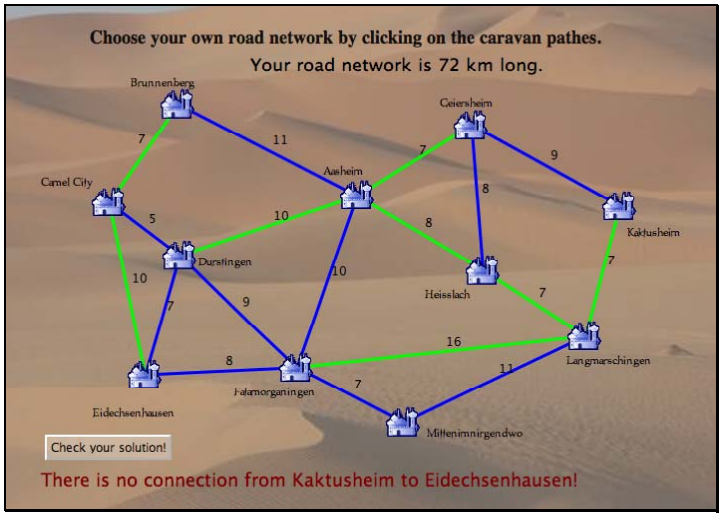
We embedded the problem into the following background story:

> ***The caliphate Sandyrealm*** *– The caliph of the dessert country Sandyrealm wants to connect the eleven cities of his country by new roads. Every city should be reachable from each other. The new roads must be built along the already existing old caravan trails, because only there are enough oasis and wells.*
> *Since Sandyrealm is a poor country the road construction should be as cheap as possible.*
> *Which caravan trails should be developed? Which criteria do you choose? How many roads must be built?*

The electronic worksheet[1] provides the map of the caravan trails shown in Picture 3. The map was created as a Cinderella construction that is embedded in an HTML document as a Java applet. Different stages of textual feedback on the student's interaction are implemented. In the electronic map the student can select or deselect any caravan path (edge) by a simple mouse click on the path. As a first immediate feedback the total weight of the selected edges is shown. This helps the students to avoid mistakes arising from wrong summation of the edge weights. At any time the student can ask for a more detailed feedback (feedback on demand) of his or her actual solution. The software detects automatically the rightness and if so the optimality of the students solution by applying standard graph algorithms on the network.

First, it is checked if all cities are connected by the solution. If this is not the case, the student gets the hint „*There is no connection from A to B*", where A and B are two non-connected cities. This is a direct hint on the source of his or her mistake. Otherwise the solution is checked for the presence of cycles. An existing cycle is reported by the little more indirect hint „*Can you build a shorter road net by omitting some edges?*" The student gets the information that he or she has chosen to many edges but not where to find the cycle. Afterwards, comparing the student's solution with an optimal solution checks the optimality. If the student's solution is not optimal, the following hint is shown: „*Your road network is very well! Nevertheless, can you build a shorter network?*" On the one hand the student gets the affirmation that he or she constructed a spanning tree. On the other hand he or she gets the indirect hint that his or her solution is not the best possible. Finally, if the student found an optimal spanning tree, he or she gets the verification feedback „Fine! You have found an optimal road net!"

[1] available at http://cinderella.de/visage



Picture 3. *An electronic worksheet for MST.*

We tested this worksheet in classroom teaching at different levels of a Berlin secondary school. Initially the students got a paper version of the exercise. They also got multiple paper copies of the map for the development of their own solution. A particular aim of the exercise was to reflect on the structure of the solutions developed by the students. During their first attempts the students only used one of the sheets, so after a while they had set as many marks and labels on the paper map that they could no more see any structure in their solution trials. After an adequate handling time they got the electronic version of the worksheet as an additional helping tool. It turned out that the attention of the students got more focused on the important aspects of the task. After working with the electronic worksheet and regarding the given feedback, they were able to create good solutions faster and to discover the structure of their approach. Most of them finally had the capacity to verbalize their own way of finding a solution during the following group discussion. More detailed results of the classroom testing can be found in [4].

5. Feedback in Interactive Geometry: Congruencies and line reflections

Our second example arises from the theory of congruencies in the Euclidean plane. We created a collection of – until now – six so called learning laboratories, i.e. electronic experimentation worksheets on this topic[2]. The laboratories can be used as separate exercises or in sequence as a learning unit.

The basic idea of our collection is the theorem that each congruency transformation can be generated by at most three line reflections. As a consequence, our software opens two different views on congruencies: Congruencies as geometric transformations and congruencies as compositions of line reflections. In the transformation view, main characteristics of a congruency can be explored, e.g. the rotation angle and the center of a rotation. In the composition view, corresponding axes defining a congruency are given (e.g. two lines intersecting at the center of a rotation) and all steps of reflecting a given object along those axes can be displayed.

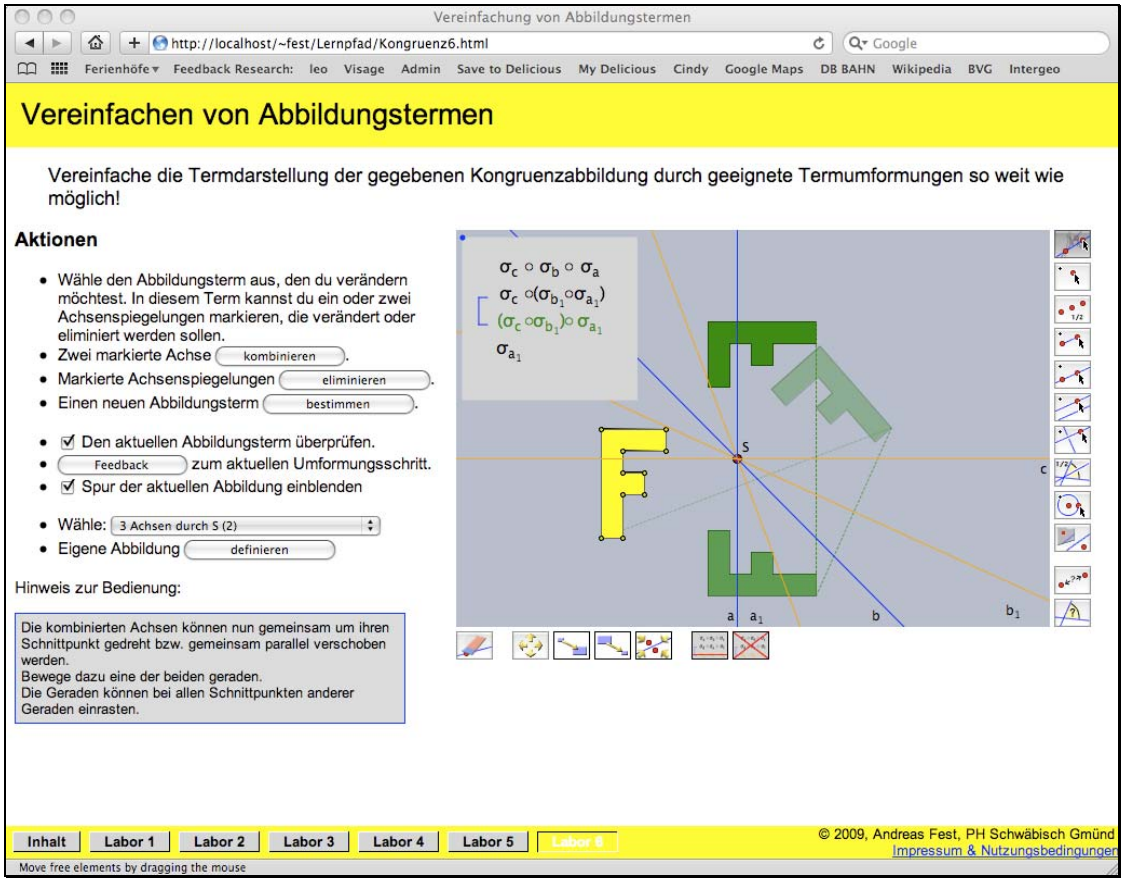
Each laboratory offers a sample of predefined examples. Additionally, a student can construct and define own examples for further explorations. The software is able to detect the type and the parameters of each user-defined congruency.

[2] available at http://geometrie.sail-m.de/Lernpfad

There are different learning targets that can be followed by the unit. First, the students should get first ideas of and a feeling for congruencies. They should learn how to create different types of congruencies as a composition of line reflections and how to find corresponding axes. Finally they should develop ideas how to reduce a given composition of (more than three) line reflections.

All laboratories are enriched with a wide range of different types of feedback, depending on the special task. The design of some laboratories directs the student to a common standard solution, but all laboratories are also open for differing creative solution strategies. Since automated feedback is not possible for all exceptional solution strategies, we implemented the possibility to record and store the whole user interaction process on a server via a TCP/IP connection. The teacher has full access to those recordings and can use them to analyze the learning process of all students and give individual feedback whenever this is necessary.

Exemplarily, we like to present a laboratory on the reduction theorem for line reflections. A screenshot of this laboratory is shown in Picture 4. In this laboratory, a given composition of line reflections should be reduced to a composition of at most three line reflections such that the type of the resulting congruency can be determined. The initial composition is given as an algebraic term that might be transformed by using some of a few predefined transformation steps. After each algebraic transformation step an additional term is displayed. Beside the symbolic representation there is a graphical view depicting the actual corresponding geometric transformation, regarding the spatial and the temporal contiguity principles of Meyer [8].



Picture 4. *A laboratory for reducing compositions of line reflections.*

To perform a single transformation step of the algebraic term, the term to change can be selected using the mouse. In the selected term, one or two line reflections can be selected for the next transformation. If two line reflections were selected, the corresponding axes can be connected. This means, a new copy replaces the selected lines. These new lines can now be rotated simultaneously around their intersection point in the graphical view, respectively translated simultaneously when they are parallel lines.

Such a simultaneous movement of the defining axes doesn't change the defined congruency provided that the two axes are neighbored in the algebraic term. Also, two neighbored line reflections with congruent corresponding axes can be eliminated from the term without changing the congruency. Hence an available transformation step is to eliminate one or two selected line reflections from the term. Additionally, students can define an own term by selecting a sequence of axes constructed in the graphical view. This allows user defined transformations of the algebraic term besides the standard solution strategy. All transformation steps are implemented to allow mistakes in the solution process.

As a first immediate visual iconic verification feedback, the graphical view can be used. Changing the selection of an algebraic term corresponds to a change of the displayed congruency. If the position of the blue transformation image of the yellow figure does not change, the transformation step between the two terms is probably correct. If the position changes, then a mistake was done. Using the "trace", i.e. displaying all steps of the geometric transformation in the graphical view, the effect of the change in the algebraic term to the corresponding congruencies gets even more clarified, resulting in a visual iconic elaborative feedback on demand.

Additionally, the rightness of the actual selected algebraic term can be checked directly. In case of a right term it is displayed in green color, respectively red otherwise (visual verification feedback on demand). Finally the student can ask for a detailed textual feedback on the selected term on demand. For its generation different algebraic and numerical control algorithms were implemented using the scripting interface of Cinderella. Those algorithms analyze and evaluate the actual transformation step and make reasonable hints available on its correctness or faultiness. The algebraic checking algorithms allow detection of both correctness and type of term transformations and are a basis for analyzing the student's solution process automatically as long as it is possible. Otherwise the software remarks that automatic detection is not possible. Then the student can ask his or her teacher. In that case, the numerical check is the starting point to initiate a manual analysis of the solution process by the teacher.

We used the software at two geometry courses for teacher students in the first year at the Universities of Education in Schwäbisch Gmünd and Ludwigsburg, Germany. The students were encouraged to use the laboratories as an exercising tool for deepening their understanding of congruencies. Afterwards we ask the students a questionnaire. The students' feedback was mostly positive. The question *"What do you like particularly when you learn using the computer as a tool?"* they said, they like *"fast feedback whether 'my' result is right"*. They also claimed *"Ideas can quickly be implemented graphical. Hence rightness can be examined."* Also the way automated feedback did help them for the exercise was described by the student: *"Because there are different 'levels' of rightness, solving the exercises gets easier"*; *"When you don't know how to go on it is useful that the computer shows the next step"*; *"When wrong, I could explain it myself after seeing the solution"*.

The only reasons why students claimed that our software did not support their learning process were based on technical problems. First, there were some incompatibilities with a few web browsers or computer configurations, which we couldn't solve during the testing. Second, some students without any previous experiences in the software Cinderella had problems to handle the user interface. Here we were able to detect different common difficulties. We look forward to solve most of the problems described by the students in the next version, e.g. by creating small pre-exercises and video tutorials on using the software, according the principles for multimedia learning by Meyer [8].

6. Conclusion

Scripting is an important tool also for dynamic geometry software. In the near future more and more DGS systems will have to support scripting interfaces. In particular, to create adjusted and activating learning environments with an individual feedback engine, a flexible and easy-to-use programming interface is essential. The success of learning in an environment supported by electronic tools strongly depends on the quality of the available feedback. Automated feedback can help the student to understand mathematics. And it can help the teacher to find interesting special solutions and common mistakes and misunderstandings.

The next step is to develop methods for a semi-automatic analysis of the students' whole learning processes. The benefits of our approach for the learning of students must be explored empirical.

Literature

[1] Bescherer, C., Kortenkamp, U., Müller, W., & Spannagel, C. (in press). Research in the field of intelligent computer-aided assessment. To appear in McDougall, A. (ed.), *Researching IT in Education: Theory, Practice and Future Directions*

[2] Bescherer, C. & Spannagel, C. (2009). Design Patterns for the Use of Technology in Introductory Mathematics Tutorials. To appear in: *Proceedings of the 9th IFIP World Conference on Computers in Education (WCCE 2009)*. Brazil.

[3] Cohen, V.B. (1985). A Reexamination of Feedback in Computer-Based Instruction: Implications for Instructional Design. *Educational Technology*, 25(1), 33-37.

[4] Fest, A. (2009). Using the Cinderella/Visage Framework for Studying Graph Algorithms, *Proceedings of CERME 6, Group 7: Technologies and resources in mathematical education*, Lyon.

[5] Kortenkamp, U. & Fest, A. (2009), From CAS/DGS Integration to Algorithms in Educational Math Software, *The Electronic Journal of Mathematics and Technology*, to appear.

[6] LISUM (2006), *Rahmenplan für die Sekundarstufe I, Mathematik*. Senatsverwaltung für Bildung, Jugend und Sport, Berlin.

[7] Mayer, R. E. (2004), Should there be a three-strike rule against pure discovery learning? *American Psychologist*, 59(1), 14-19.

[8] Mayer, R. E. (2009), *Multimedia Learning (2nd Edition)*, Cambridge University Press.

[9] Park, O.-C. & Gittleman, S. S. (1992). Selective Use of Animation and Feedback in Computer-base Instruction. *Educational Technology, Research and Development*, 40(4), 27-28.

[10] Pridemore, D. R. & Klein, J. D. (1991). Control of Feedback in Computer-Assisted Instruction. *Educational Technology, Research and Development*, 39(4), 27-32

[11] Richter-Gebert, J. & Kortenkamp, U (2006), The Interactive Geometry Software Cinderella, Version 2.0. Available online at http://cinderella.de.

[12] Orton, A. (2004), *Learning Mathematics: Issues, Theory, and Classroom Practice (3rd Edition)*, Continuum International Publishing Group.

[13] Schroeder, U. & Spannagel, C. (2006). Supporting the Active Learning Process. *International Journal on E-Learning*, 5(2), 245-264.

[14] Schulmeister, R. (2007). *Grundlagen hypermedialer Lernsysteme* (4th Edition), München: Oldenbourg.

[15] Spannagel, C. & Kortenkamp, U. (2009). Demonstrating, Guiding, and Analyzing Processes in Dynamic Geometry Systems. In: Bardini, C., Fortin, P., Oldknow, A. & Vagost, D. (Eds.). *Proceedings of the 9th International Conference on Technology in Mathematics Teaching*. Metz, France: ICTMT 9.

Authors

Andreas Fest, University of Education, Schwäbisch Gmünd, Germany, e-mail: andreas.fest@ph-gmuend.de

Acknowledgement

Parts of this work were supported by the DFG Research Center MATHEON.

Parts of this work were supported by the German Federal Ministry of Education and Research.