Reducing graphical user interfaces in DGS

Florian Schimpf

University of Education Ludwigsburg
Germany

+497141140729

schimpf@ph-ludwigsburg.de

Christian Spannagel

University of Education Heidelberg
Germany

+496221477281

spannagel@ph-heidelberg.de,

Abstract

Graphical user interfaces (GUIs) of dynamic or interactive geometry software (DGS) allow users to interact with the DGS by using a computer mouse. Clicking on a GUI icon performs an action like choosing a construction tool or manipulating an object. For novices it can be difficult to recognize and recall the icons needed for a task. Learning mathematics and learning the use of a dynamic geometry system could lead to cognitive overload. Several dynamic geometry software systems try to solve this problem by offering different GUIs: Expert users can choose between a wide range of icons while for novice users only the most basic icons are presented.
In an experiment carried out with full and reduced interfaces of the dynamic geometry software Cinderella the eye movements and gaze points of the users were recorded by using eye tracking. It has been measured how long users need to find given icons in different types of interfaces. The findings of this experiment are described and ideas for further studies are discussed.

Keywords

DGS; GUI reduction; eye tracking

1 Introduction

In schools the use of dynamic geometry software (DGS) has become more and more common over the last years (see Richter-Gebert and Kortenkamp 2010 for an overview). The benefits of using such programs in geometry like moving constructions, discovering connections, and exploring proofs offer new ways of learning. However, learning mathematics and learning how to use a specific DGS at the same time need a lot of cognitive resources. The wide variety of functions of DGS may be confusing not only for novice learners. Due to the fact that cognitive capacity is limited, the cognitive resources that are used to learn the usage of the DGS are not available for learning mathematics.

DGS are controlled via a graphical user interface (GUI). By using the computer mouse the user clicks on an icon to perform a specific action. When solving mathematical problems like constructing a triangle or measuring the angles of a given polygon, a complex sequence of GUI actions has to be performed by the user. With complex or not properly designed GUIs this can be a frustrating experience and constrain the learning process (Kortenkamp and Dohrmann 2008). The user is confronted with a large number of GUI elements and must therefore know which command leads to which action in the DGS. To perform a specific tasks only a subset of the functionality of the DGS is needed. To reduce the complexity and increase the usability of the DGS the user interface can be adapted to specific demands. For example, teachers can modify the GUI of the DGS by customizing the tool bar in order to offer a reduced interface to students. However, the effect of reducing the functionality of GUIs on learning has not yet been sufficiently evaluated (Findlater and McGrenere 2007). It is not clear whether reducing the GUI has a positive effect on learning with a DGS. In this article, an experiment is described where the effects of reducing the GUI of the DGS Cinderella are investigated.

The theoretical background of interfaces with a reduced functionality is described in section 2. In section 3, different means of reducing the complexity of DGS are explained. In the fourth section an experiment with various reduced GUIs of the DGS Cinderella is described, and its results are reported and discussed. The article ends with concluding remarks and ideas for further experiments.

2 Theoretical Background

The functionality of a GUI can be reduced in order to free cognitive capacities for learning. Therefore, cognitive load theory as a central theoretical basis is described first. In the second part of this section, different approaches of GUI modifications are explained. A special focus is laid on training wheels interfaces which offer only a basic subset of functions in order to support novices in learning to use the software.

2.1 Cognitive Load Theory

The cognitive load theory (CLT) provides a theoretical framework for designing instructional material and graphical interfaces. As a general assumption the CLT proposes that in the human cognitive architecture the working memory is limited (Sweller et al. 1998). The CLT describes how information processing leads to a load in working memory. Because processes relevant for learning such as schema acquisition are performed in working memory, this part can be seen as critical point for learning. The CLT distinguishes three different types of cognitive load which are additive: intrinsic, extraneous, and germane cognitive load (Sweller et al. 1998).

Intrinsic cognitive load accrues from the complexity and the level of difficulty of the task that must be performed by the learner. For instructors and designers of GUIs the *extraneous cognitive load* is important, because this kind of load is the result of improperly designed instructional material. If users are overwhelmed by software features offered in the GUI the extraneous cognitive load will rise and therefore hamper learning. Finally, the *germane load* is the positive load that occurs when the capacity of free working memory is used for deeper construction and automation of schemata (Bannert 2002).

Based on CLT, reducing extraneous load by properly designing user interfaces should free cognitive capacity which is then available for germane load needed for learning mathematics.

2.2 GUI Adaptation

GUI adaptation can be realized within three different approaches: adaptive interfaces, adaptable interfaces, and a mixed approach. *Adaptive interfaces* automatically change by maintaining a model of the user's skills or by detecting the mostly used features (Findlater and McGrenere 2004). *Adaptable interfaces* can be customized by the user himself or by a second person who wants to offer a specialized interface to the user. Adaptable interfaces are implemented in several DGS. For example, the teacher may adapt the interface to the students' needs by choosing a set of icons for teaching in schools in the DGS Cinderella. In the *mixed approach*, adaptive and adaptable elements are combined.

A special type of adaptation is the reduction of the complexity and functionality of GUIs. Early forms of GUI reductions were *training wheels interfaces* (Caroll and Carrithers 1984). Software users often do not read manuals or ask for help. Novice users rather tend to explore software applications and often use trial or error strategies. This can lead to troublesome error states of the program from which a return without help is difficult (Catrambone and Carroll 1987). To beware users of frustrating experiences, training wheels interfaces offer only a basic set of functions, while all unnecessary functions are blocked. When trying to use a disabled function the system generates a message that informs the user about the unavailability of the specific choice (Carroll 1984b). In a study with a word processing software Carroll and Carrithers (1984) showed that beginners using training wheels interfaces were faster and performed better than users with the full package.

In a field study realized within a text processing course subjects using training wheels performed faster than participants using full interfaces (Bannert 2000). Besides measuring the learning outcome, subjects were asked for their subjective satisfaction of using training wheels in this study. The mode of blocking irrelevant functions was evaluated as too restrictive. It can be assumed that especially experts feel constricted by training wheels interfaces because they don't have access to features they perhaps want to use. In another experiment by Spannagel et al. (2008), no positive effects of training wheels interfaces have been found.

In the work of Findlater and McGrenere (2007) the terms *findability* and *awareness* are introduced as indicators for measuring the effects of GUIs with a reduced functionality: "Findability is the speed with which the user can locate a function she knows exists. The set of findable functions includes those the user has already used […], or those the user has used in a previous version or reduced-functionality version. Awareness is the degree to which the user notices (through using an application) and recall functions which do not fall in the findable function set above." (Findlater and McGrenere 2007, 593-394). Results of their study indicate that there is a trade-off between these two measures when learning with reduced interfaces or with full interfaces. The findability tends to be better when subjects have learned with reduced interfaces, but subjects who have worked with full interfaces are more aware of features they have not used for solving a specific task (Findlater 2007).

3 Reducing the Complexity of a DGS

Given the whole functionality of a DGS, the user may be overwhelmed by its complexity when searching for functions he wants to use. Therefore, users have to be supported in orienting themselves in the DGS. First, developers can reduce the complexity of the DGS by using common means of user interface design. Second, methods can be offered to the user that enable him to reduce the complexity by himself.

In the following, methods of reducing the complexity of a DGS are described in two subsections. Complexity can be reduced by *structuring the functionality* and by *removing functionality*.

Reducing complexity by structuring

- *grouping the functionality in dropdown menus*
  This is well known from many applications. For example, in Cabri the menu "file" offers the choices of "new", "open", "close", "save", "print", etc. in a dropdown menu. In Geogebra the icons of the tools are arranged as dropdown menus and only the last actively used appears in the toolbar.

- *subdividing dropdown menus into cascading menus*
  In Geogebra, this is used for the menu "options". When selecting the submenu "point style" another submenu will open that allows the user to choose between seven different point styles. To bundle actions like choosing styles or sizes is widespread in DGS.

- *showing only a part of the dropdown menu*
  In MS Word the menu "file" only shows a few options like "save", "save as", "print", etc. by default. An arrow at the bottom of the dropdown menu indicates that the dropdown menu offers further choices for experts.

- *offering right-click context menus*
  Context menus are available by clicking with the right mouse button on an object. If using the context menu in Cinderella, information about the selected element and further actions on this object are accessible.

- *grouping functions in tabs*
  Attributes of objects such as points or lines are often grouped in tabs, e.g. in Geobegra.

- *using optional palette windows*
  In many graphical applications like Gimp or Photoshop a tool palette is available, offering an easy access to functions in separate window.

Reducing complexity by removing functionality

- *disabling inapplicable items of a menu or a toolbar*
  If in Cabri no construction in the drawing area is marked or active, the options "cut", "copy", and "paste" are written in grey color in the dropdown menu and they are blocked ("disabled"). This indicates that the user cannot use these functions and therefore can concentrate on the available functionality.

- *customizing toolbars*
  In many DGS users can select icons to be displayed in the toolbar. This can be done by selecting single icons to be shown or not (e.g. in GeoNEXT), or by choosing whole toolbars like in the DGS Cinderella.

- *offering multi-layer interfaces*
  Different interface layers can be selected by the user, with layer 1 showing only the most basic features for novices (Shneiderman, 2003). While the expertise of the user grows, more complex layers can be chosen. No DGS known to us offers multi-layer interfaces at the moment.

Most of these strategies to reduce the complexity of a DGS can only be used by the developers of the DGS. From the teacher's point of view, a DGS can be chosen to be used in the classroom based on considerations whether the interface of the DGS is properly designed or not. For example, a teacher may decide to use Geogebra because icons in the toolbar are arranged in dropdown menus.

The only way to influence the complexity of the interface by a teacher himself is to customize the toolbar of the DGS. Icons of tools which have to be used in order to solve a specific task can be chosen to be shown in the interface. For educational purposes it can be helpful to offer only a subset of the whole functionality to learners. When students work with a reduced-functionality interface they may have more free cognitive capacity for understanding the mathematical concepts. It can be assumed that reducing the functionality in the toolbar reduced extraneous cognitive load and increases the findability of construction tools. Disadvantageous may be the fact that the awareness of functions decreases by hiding them in the toolbar. In the next section, an experiment is described which has been conducted in order to investigate the effects of reducing the functionality in the toolbar of the DGS Cinderella on findability and awareness.

4 Experiment

According to the cognitive load theory, instructors and designers can reduce the extraneous cognitive load by removing functionality from the user interface of the DGS. This can free cognitive capacity which can then be used for the learning process itself. Therefore, reducing the functionality by removing icons from the toolbar of a DGS may have the potential to foster learning mathematics. It can by hypothesized that reducing the functionality of GUIs can lead to a lower extraneous cognitive load, because a lower amount of cognitive capacity has to be used for searching screen elements at the GUI (higher findability). However, transfer to the full interface may be hampered (lower awareness).

In the experiment described below, the complex GUI of the DGS Cinderella was reduced by removing icons from the toolbar. According to the findings from Findlater, findability and awareness have been measured. For the first time in this context, an eye tracking device has been used in order to determine the times users need to find icons in a DGS.

4.1 Method

Seventy students of the University of Education Ludwigsburg, Germany, participated in the study. Only students who do not study mathematics as subject were asked to participate in the experiment. This ensured that the subjects had no previous knowledge about the DGS Cinderella. The 70 participants were randomized to three experimental conditions with three different user interfaces in the learning phase. In the full interface condition no changes were made compared to the standard Cinderella GUI with 48 icons in the toolbar (Fig. 1). In the reduced interface condition only 23 icons were visible. 25 icons were hidden, and the visible icons where shown at their original position (Fig. 2). In the compressed interface condition the 23 visible icons were positioned at the beginning of the toolbars (Fig. 3).

The experiment consisted of two phases (Fig. 4). In the learning phase, the subjects had to find five icons in the user interface of their experimental group (full interface, reduced interface, or compressed interface). First, the picture of a single icon was presented for three seconds. Afterwards, the Cinderella GUI was shown for 10 seconds. The task was to find the icon in the interface in less than 10 seconds. After this time, the next icon was shown, and so on.

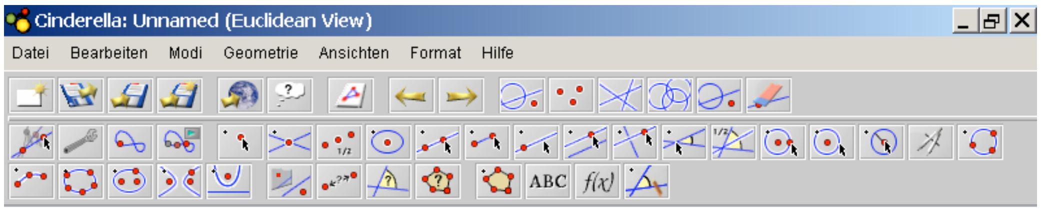

Figure 1. Full Cinderella interface

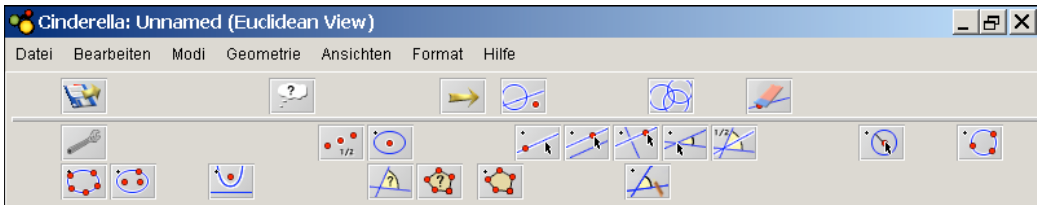

Figure 2. Reduced Cinderella interface

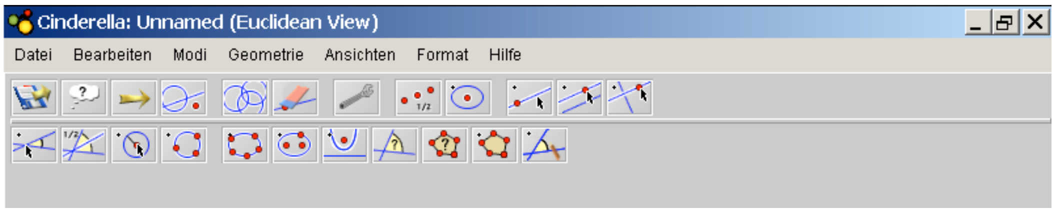
Cinderella: Unnamed (Euclidean View)
Datei Bearbeiten Modi Geometrie Ansichten Format Hilfe

Figure 3. Compressed Cinderella interface

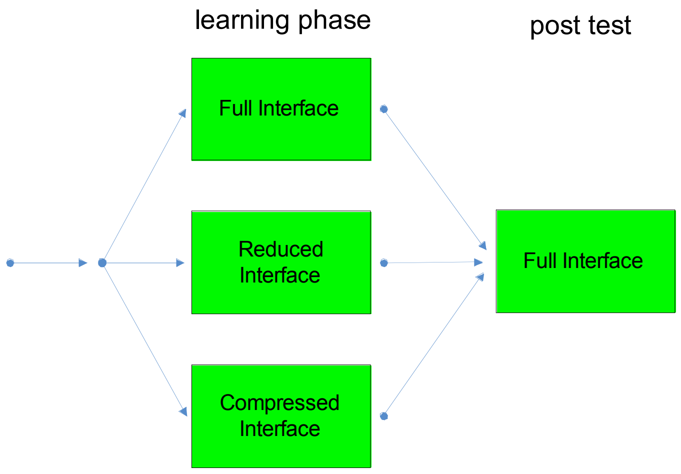

Figure 4. Two phases of the experiment

In the second phase (post test), all subjects had to find icons in the full interface of Cinderella. Besides the five icons known from the learning phase subjects had to find five new icons in addition. The new icons were contained in the full interface during the learning phase, but not in the reduced or compressed interfaces.

The subjects' gaze points have been recorded with a contact free eye tracking device and the Software Nyan™. An area of interest was laid around the target icon in order to measure the time when a subject looks at this icon. The time from the presentation of the GUI to the subject's first fixation of the target icon was taken as time the subject needed to find the target icon.

4.2 Hypotheses

The hypotheses of the experiment are:

- In the learning phase, users in the full interface condition need more time to find the icons than users in the reduced and compressed interface condition (findability).
- In the post test phase, users in the full interface condition need less time to find new icons than users in the reduced and compresses interface condition, because they had the chance to notice these icons in the learning phase (awareness).

- In the post test phase, users that had the compressed interface during the learning phase need more time to find icons known from the learning phase than users in the full and reduced interface condition, because icons changed their places in the toolbar.

4.3 Results

Fifty-one of 70 datasets were valid. For data analysis, icons were grouped into icons of the learning phase (*icons learning phase*), icons of the post test phase which were already known from the learning phase (*old icons test phase*), and new icons in the test phase (*new icons test phase*). Figure 5 shows the means of the times needed to find the icons of the different icon groups, separated for the three experimental conditions.

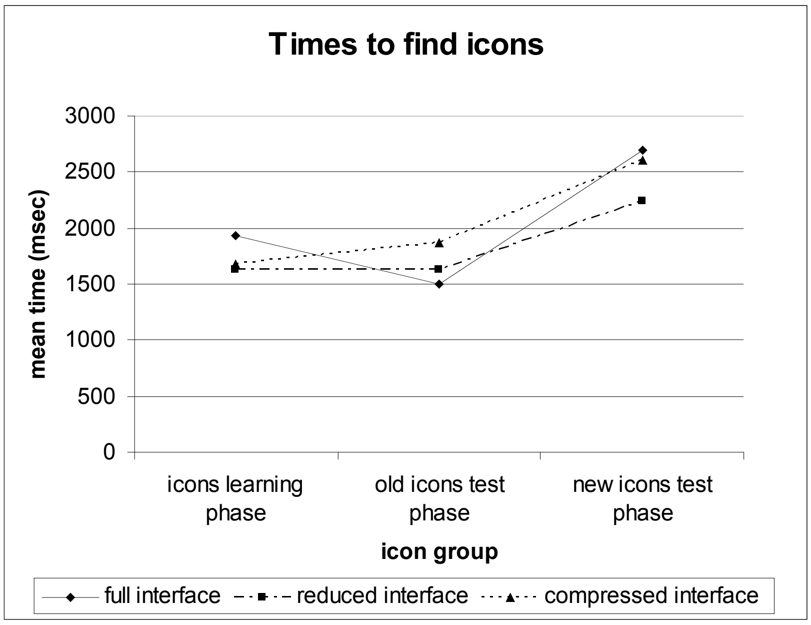

Figure 5. Mean times to find the icons in the three experimental groups

A two-factorial analysis of variance with repeated measures has been conducted with “experimental group” as factor A and “icon group” as factor B. Only the effect of the factor “icon group” has been significant (F=22.85, critical value $F_{(2,47)}<3.2$ for $\alpha=0.05$). There was no significant interaction effect between icon group and experimental condition (F=2.15, critical value $F_{(4,94)}<2.47$ for $\alpha=0.05$). In a post hoc test, differences between the three icon groups have been analyzed. With a Bonferroni-corrected alpha level, none of the differences were significant. Nevertheless, the search times for new icons in the test phase seem to be higher for all experimental groups (M=2518 msec) than the search times for the icons in the learning phase (M=1744 msec) and for the old icons in the test phase (M=1665 msec).

4.4 Discussion

The results of the experiment indicate that there is no relevant difference in search times between the experimental groups with regard to different icon groups (icons in the learning phase, old icons in the test phase, new icons in the test phase). This means that it makes no difference if a full interface, a reduced interface, or a compressed interface is presented. Thus, the hypotheses have not been confirmed. The findability is not better in the reduced interface groups, and the awareness of full interface group with regard to new icons presented in the test phase is not higher than in the two other experimental groups.

Differences of the mean times are not higher than ca. one second. This indicates that even the higher search times for the new icons in the test phase seem not really to be relevant. It can be concluded that reducing the functionality of a DGS by removing icons from the toolbar seems not to have any relevant effect on search times.

5 Conclusion and future work

In this article an experiment has been described where users had to find icons in different user interfaces of the DGS Cinderella. Eye tracking has been used to measure the times users needed to find the icons. The results showed that with reduced interfaces the findability of already known icons is not better than with full interfaces. In addition, the awareness of users who had learned with a full interface was not higher than the awareness of users in the other experimental conditions with respect to icons which had not to be found in the learning phase. This gives teachers the free choice of using a full or reduced interface with regard to specific educational considerations. It could make sense to reduce the amount of icons of a GUI for specific tasks, but it seems that using the full interface does not irritate users as much as expected.

In this experiment, no learning task had to be performed. Subjects only had to find icons in the graphical user interface. In further experiments, students will have to solve a geometric problem using different kinds of user interfaces. In this way it can be investigated if reducing the interface helps solving the task. In addition, a time series experiment will be conducted where subjects have to find icons in user interfaces with an alternating complexity in order to replicate the results found in the study presented in this article.

Acknowledgements

We would like to thank Mrs. Irene Reeb Ramos for her support during the experiment. Christian Spannagel is indebted to the BADEN-WÜRTTEMBERG STIFTUNG for the financial support of his research project by the Eliteprogramme for Postdocs.

Literature

Bannert, M. (2000). The Effects of training wheels and self-learning materials in software training. *Journal of Computer-Assisted Learning*, 16(4), 336-346.

Bannert, M. (2002). Managing cognitive load. Recent trends in cognitive load theory. *Learning & Instruction*, 12, 139-146.

Carroll, J., & Carrithers, C. (1984).Training wheels in a user interface. *Communication of the ACM*, 27(8), 800-806.

Carroll, J. (1984b). Blocking learner error states in a training-wheel system. *Human Factors*, 26 (4), 377-389.

Catrambone, R., & Carroll, J. (1987). Learning a word processing system with training wheels and guided exploration. *ACM SIGCHI Bulletin*, 17, 169-174.

Findlater, L., & McGrenere, J. (2004). A comparison of static, adaptive, and adaptable menus. *Proc. ACM Conference on Human Factors in Computing Systems* (CHI 2004), (pp. 89-96). New York: ACM.

Findlater, L. (2007). Design and evaluation of reduced-functionality interfaces. *ACM Conference on Human Factors in Computing Systems (CHI 2007)* (pp. 1637-1640). New York: ACM.

Findlater, L., McGrenre, J. (2008). *Comprehensive user evaluation of adaptive graphical user interfaces.* Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2008. New York: ACM.

Findlater, L.(2004). *Supporting Features Awareness and Improving Performance with Personalized Graphical User Interfaces.* http://faculty.washington.edu/leahkf/pubs/findlater-phd-thesis.pdf. Accessed 13 June 2010.

Findlater, L. and McGrenere, J. (2007). Evaluating reduced-functionality interfaces according to feature findability and awareness. *Proceedings of IFIP Interact 2007*, 592-605.

Kortenkamp, U., Dohrmann, C. (2010). User interface design for dynamic geometry software. *Acta Didactica Napocensia*, 3(2), 59-66.

Richter-Gebert, J., Kortenkamp, U. (2010). The power of scripting: DGS meets programming. *Acta Didactica Napocensia*, 3(2), 67-78

Shneiderman, B. (2003). Promoting universal usability with multi-layer interface design. *Proceedings of the ACM conference on Universal Usability 2003* (pp. 1-8). New York: ACM.

Spannagel, C., Girwidz, R., Löthe, H., Zendler, A., & Schroeder, U. (2008). Animated demonstrations and training wheels interfaces in a complex learning environment. *Interacting with Computers*, 20(1), 97-111.

Sweller, J., Van Merriënboer, J., & Paas, F. (1998). Cognitive architecture and instructional design. *Educational Psychology Review*, 10, 251-296.