ACTA DIDACTICA NAPOCENSIA

Volume 3, Number 2, 2010

USER INTERFACE DESIGN FOR DYNAMIC GEOMETRY SOFTWARE

Ulrich Kortenkamp, Christian Dohrmann

Abstract: In this article we describe long-standing user interface issues with Dynamic Geometry Software and common approaches to address them. We describe first prototypes of multi-touch-capable DGS. We also give some hints on the educational benefits of proper user interface design.

Key words: Human Computer Interaction, Multi-Touch, Gesture Recognition

1. Introduction

Dynamic or Interactive Geometry Software (DGS) has become a standard tool in ICT-based teaching of mathematics. A wide variety of products exists, and all programs share the intuitive "drag-mode" approach to geometry (and more generally, mathematics). Students are able not only to construct and create, but also to modify and observe.

The necessity of a mathematical foundation for DGS has been discussed already (e.g., see Kortenkamp 1999). Still, it is debatable whether the internal architecture of such a program does influence its use in the classroom. As the current activities (see http://i2geo.net for a great variety) usually focus on manipulating ready-made example constructions, the subtleties of mathematics can be avoided on the "customer" side – they still persist on the authoring side, however.

In our experience the user interface of a DGS seems to be a major obstacle with respect to learning. This is not surprising, as all tools, even traditional ones like ruler and compass, or even squared paper (Brock & Price 1980), are not intuitive, in the sense of "can be used without instruction". It is even more true for computer software (Raskin 2000), so we have to take care that we do not invest too much time into "learning software" as opposed to "learning mathematics". In this paper we want to start a discussion that needs to be complemented with further research and usability tests in the classroom.

2. User Interface issues for DGS

We start by describing some special problems of user interface design for DGS. We do not discuss solutions, nor give recommendations which approach might be better suited for teaching or research in each case, as no aspect can be approached individually alone, but the whole range has to be taken into consideration when searching for better user interfaces for a certain audience.

2.1. Wide Functional Range vs. Usability

The complexity of software increases with every additional feature. Thus, software for educational use should be restricted to only the bare necessities. On the other hand, users demand advanced features. This is a standard problem also in general – users want a lot of choice without having to choose. In an educational setting it is even worse: If students learn to combine several steps into a more complex operation (say, instead of using the compass twice and connecting the intersections to find the midpoint between two points one can use the midpoint operation provided by the software) this reduces their workload when doing constructions, but it increases the complexity of their software use. Thus, by understanding more mathematics they are required to manage a more complex software system.

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.

2.2. DGS vs. Drawing Software

Drawing software (e.g. Adobe Illustrator or The Gimp) shares some of the visuals with DGS and also some metaphors are similar. A student who can draw a point in a drawing software may not realize that the similar operation in a DGS carries another concept. As drawing software usually does not store relations between objects (like "is intersection of") but still respects relations like that during the creation of elements (a point can snap to an intersection in "magnetic mode"), students may not be aware that they are working with a completely different tool.

2.3. Traditional Tools vs. DGS

Some operations in a DGS are very close to their traditional counterparts (there is even a DGS called "C.a.R.", *circle and ruler*). Here, the object that is being created (a line or circle) is usually confused with the tool that is being used (a ruler or compass). For students, the only way to create a straight line on paper is the one using a ruler. On the computer, although all straight lines look equal, they may be created using a variety of tools – the "line through two points" tool, the "line through one point" tool, the "line perpendicular to" tool, the "parallel line" tool, etc. For circles, it is even more confusing, and the tool corresponding to a real compass is used rarely and many teachers and students don't understand its semantics (Kittel 2007).

2.4. Objects and Relations

The difficulty described in the previous two sections can be seen in broader sense. The tools in DGS combine the creation or manipulation of objects *and* relations between the objects at the same time. Adding, say, a parallel line both adds a line *and* the parallelism constraint to the construction sequence. Adding an orthogonal line both adds a line *and* the orthogonality constraint to the construction. For the user, both lines look the same, unless the construction text (if available) is inspected. Then, the line usually is "a parallel" or "a perpendicular", that is, the relation of this object is again identified with the object itself.

Even some persons who develop DGS sometimes ignore this problem. The i2g file format for DGS construction exchange (Hendriks et al. 2009) tries to emphasize this distinction on the software level by separating *elements* and *constraints*.

2.5. Object/Action vs. Action/Object

When defining new elements there are two fundamentally different approaches:

(a) First choose the objects, then an action using these objects, or

(b) First choose an action, then choose the objects used for this action.

The discussion about the "better way" is as old as Dynamic Geometry Software. The two pioneering DGS, Geometers' Sketchpad and Cabri Geometry, follow different strategies, and there are arguments for (and against) both. Actually, most software currently uses a mixed approach, where it is possible to either pre-select objects and apply an operation, or select an operation and post-select the objects. While this is the common approach in many computer programs (including word processing software and other standard tools), it is not clear that this is the best solution for the O/A-A/O-problem.

2.6. Order of objects

For multi-input constraints the order of the defining objects can influence the defined elements. As the simplest example we mention polygons, where the order of the vertices is crucial, even in the triangle case if the area calculation is orientation sensitive. This particularly makes a difference in the O/A approach (see above), as the order of the selected elements is usually invisible for the user and only implicit.

2.7. Ease of Use vs. Intellectual Barriers

The meta-problem of DGS user interfaces and user interfaces of educational software in general is the decision between designing an easy-to-use interface vs. designing an interface that deliberately enforces intellectual activity of the users by being "difficult". This is a decision that cannot be independent of the target users, as any intellectual barrier has to be designed appropriately to match the users' experience, prior knowledge, motivation, context, goals, and requirements.

As such, we are in a dilemma, as we cannot find a one-size-fits-all user interface. On the other hand, we could identify several criteria that can help in finding the appropriate design based on pedagogical principles.

3. Common Approaches

We want to demonstrate briefly some common approaches to the problems of the last section.

3.1. Menu Configuration

The large number of tools provided by some software is often overwhelming for students and teachers (Kittel 2007). They prefer to use a clean and tidy interface without distractions from tools (represented by icons) they don't use. Also, it is difficult to identify the correct icon from a choice of three or more that look almost identical.

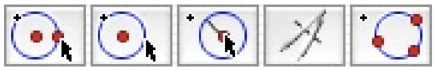

Figure 1. *Cinderella, several tools for creating circles.*

This multitude of icons in Fig. 1 is caused by the problem in Sec. 2.4 – each relation needs a different action, even if the output is (or looks) the same.

A traditional solution is to cluster the icons and to create drop-down menus, similar to the tool palette in Adobe Illustrator or Photoshop. This concept is used by Cabri, see Fig. 2.

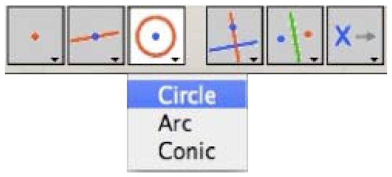

Figure 1. *Cabri tool clusters*

In the Adobe products, the tool palette gives access to many tools very quickly, and the drop-down menu is used only rarely. Thus, most actions are accessible using only a single click of the mouse. Due to the clustering of tools in Cabri which is based on the output of the elements, the usual interaction with the tool palette is similar to the press-drag-release operation for a menu item. Thus, apart from having a varying icon as the menu title and being located in a non-standard screen location, there is no difference to a standard menu, and in fact it is more difficult to use – but, based on many discussions with users of DGS, this kind of menu seems to be preferred nevertheless.[1]

3.2. Adaptive UI

Another way to solve the "icon explosion" problem could be to add new icons or remove unused ones automatically based on the users' progress and experience. Unfortunately, we can neither guarantee that the same user is using the software (imagine a classroom scenario where students take turns), nor can we infer anything about the users' needs based on what they did before. How many circles does a user have to construct before he or she starts with ellipses? If a user starts measuring areas, does this mean he or she will no longer use the protractor?

So the automatic way does not work. Can we let the user decide? That's a catch-22 situation. We have to provide a way to let users ask for extended functionality. If so – why no offer the extended functionality in the first place? Even if we enable groups of icons at once, as done in Cinderella by

[1] Even if we can find a user interface that is superior to any other, there is no guarantee that users like it better, in particular if they don't get any training on it. Again, based on our experience with users of DGS without any further statistical analysis, usually the first interface approach users are exposed to is the one they prefer.

choosing toolbars, this only introduces just another step before we have to solve the original problem of having too many indistinguishable icons. However, this approach works for posing well-defined problems in form of electronic exercise sheets: Exporting a small set of tools as an Applet worked for the collection of activities for Grades 5-9 that accompanied the school edition of Cinderella.

3.3. Wizards

“Wizards”, series of dialogues in question-answer-form, are a very common approach for configuring software and hardware or creating products from software (say, burning a DVD containing material found in several places on the hard disk). Unfortunately, this solution has several drawbacks. The step-by-step procedure requires users to read and understand all dialogs, which can be tedious and boring and is thus prone to errors.

From a pedagogical point of view, “wizards” replicate a teaching model that is based on teachers asking questions and students replying to them (and only to them), which is in strong contrast to the constructivist approaches that can be supported by DGS. So, while wizards might be a solution for software developers, they don’t seem to be appropriate in educational software.

4. Hardware based approaches

Some notable progress has been made with respect to alternate input/output devices. The mouse is the standard input device today, and users have been trained to use it. Most people have no difficulties in moving the mouse pointer correctly, although the movement of the mouse on the table with relative positioning is very different from the absolute movement of the mouse pointer on screen. Like piano players, most people have no problem in looking at a different place (the written music, the screen) while acting in another (the piano keyboard, the mouse pad).

At second sight, this is debatable. While it might be easy to learn how to move a mouse pointer, which is comparable to learning to drive or mastering a bicycle, this indirection introduces the need to build up complex abstract cognitive models in the unconscious. This decoupling of action and (visual) experience could be the cause for a lack of understanding, and it suggests having a closer look at alternate approaches.

It is very difficult to follow a computer demonstration where it is necessary to follow the mouse pointer. This is an issue when educational computer software is used for classroom demonstrations. While it might be easy for the teacher or student to present facts and relations, it is much harder for the audience to comprehend them.

4.1. Interactive Whiteboards & Pen-based devices

Interactive Whiteboards can reduce the cognitive load while following a presentation and also offer an interface that is close to the pen-and-paper interface everybody grew up with. The absolute positioning of the pen directly corresponds to the absolute position of the mouse pointer. The same is true for pen-based handheld devices.

The nearness to pen-and-paper also challenges the user interface of the DGS use. It is more natural to draw line through two points than to define a line by clicking on the two points, it is more natural to draw a circle around a point, or through three points, or through one point around a midpoint, than it is to select a mode and click on the defining elements using the pen. Here, DGS that uses a press-drag-release pattern for user input is closer to the “blackboard-paradigm” than DGS that uses “select element” strategies.

As pointed out by Materlik (2003) and Kortenkamp & Materlik (2004) it is also problematic to use an icon toolbar at all. The icons either force users to move their hand very far (whiteboard) or fill up precious screen estate (handheld). The solutions suggested and demonstrated there (the “No-UI-approach”) use a form of “sketch recognition” that enables users to encode the relations between the objects they draw easily. The necessary gestures are easy to learn and memorize.

Still, despite the wide availability of interactive whiteboards most users stick to the traditional, icon-based interface. One possible reason is that users have to switch back to the traditional interface anyway as soon as they are back to their desktop computer. Again we see the effect that most people refuse to learn more than one interface.

4.2. Multi-Touch

Multi-Touch interfaces were made popular through the iPhone and its easy-to-use interaction. People are used to being able to manipulate things using their fingers, and there is only a historical explanation for the fact that computers usually support only one mouse pointer. In fact, Moscovich (2007) shows that "in a wide variety of tasks, continuous graphical interaction using several fingers allows users to communicate information to a computer faster and more fluently than single-point graphical interaction techniques".

By looking at the specific UI problems of DGS we identify another potential of Multi-Touch. Using several fingers allows the users to give information about both the objects to be created and the relations that govern them at the same time. As an example, consider drawing a line with one finger while "holding" another – existing – line. If the new line is approximately parallel to the one held, then it should be constructed as the parallel line. This could be a gesture in a MT-enabled DGS.

Also, being able to touch and move several objects at once enables users for the first time to work with freely movable lines. Until now, when moving such a line it either had to rotate around a point or it could only be translated. With two "handles" it is easy to translate and rotate it at the same time.

In order to explore the possibilities of Multi-Touch input we created a prototype of Cinderella (available at http://beta.cinderella.de/public) that is capable of interpreting Multi-Touch events delivered by the TUIO protocol (see http://www.tuio.org). The TUIO protocol was developed by Kaltenbrunner, Bovermann, Bencina and Costanza (2005) and delivers information over a TCP channel, based on the Open Sound Control (OSC) music protocol. It can be used both locally on computer or across a network. Events are created by TUIO enabled servers, for example MSA Remote on the iPhone, or TongSeng on Mac OS X, which translates the touchpad events into TUIO events, or WM_Touch which translates Touch Events in Windows 7 into TUIO events. We chose TUIO for its cross-platform availability and the straightforward API.

Each gesture of a finger creates a Cursor Create/Move/Destroy sequence and can easily be identified using a cursor ID. If several fingers touch simultaneously, then several create events will be sent before the corresponding cursors will be destroyed again when the fingers are moved away from the surface. TUIO also supports events for arbitrary objects that may be tracked on a surface.

The experimental MT interface was based on a design done by Ulrich Lang, Thomas Techert, Kirstin Walker und Fabian Gronbach of the HfG Schwäbisch Gmünd. Besides a "multi-move" mode, where it is possible to drag several points simultaneously, we also added functionality to create points and lines and segments. Moving elements is done in the most natural way possible; you can just grab and move them. While moving one point, it is still possible to move others; the only limit is the number of fingers available. Touching in free space will create a point, touching with two fingers in free space will create two points and a connecting line, unless the two points are very close to each other – in that case, a line segment will be created. The newly created elements are still temporary and movable while being touched, and it is possible to move them to either an intersection, a line, or an existing point, which will create an intersection point or a point on a line, or re-use the existing point. This is indeed similar to the non-MT interface of Cinderella, where a new point is still temporary while the mouse button is pressed, and it is possible to attach this point to an existing one or create an intersection point by moving it there (Richter-Gebert & Kortenkamp, 1999).

The implementation of the MT interface made it necessary to change some assertions in the implementation. Usually, software developers can assert that they only have to handle a single press/drag/release cycle of the mouse. Thus, in the "mouse down" event all setup for the following tasks can be done, and the "mouse drag" and "mouse up" events can use that information. A mode in the user interface corresponds to an object instance that handles mouse input. For MT, this is not sufficient. A "cursor create" event indeed has to create an object that is responsible for handling all subsequent events of that cursor (identified by its ID). At the same time, these event handler objects need ways to communicate with each other, as shown by the “create a line using two fingers” example. [2]

The prototype has been presented and tested on various MT hardware. The images below show some interaction using a home-built back projection MT screen based on the Laser Light Plane (LLP) technology. According to this technical approach, infrared lasers are used to create a thin (1mm) laser light plane right above the surface. By touching the screen with a finger a diffuse reflection is generated that is registered by a camera as a touch point and translated into TUIO events by the tracking application (CCV). A demonstration video of both the hardware and software prototype is available online at http://www.youtube.com/watch?v=qraL4nIfkbI.

First experiments have shown that the modeless approach introduces the problem of accidental object creation. This suggests that it is still necessary to distinguish between construction modes and manipulation modes.

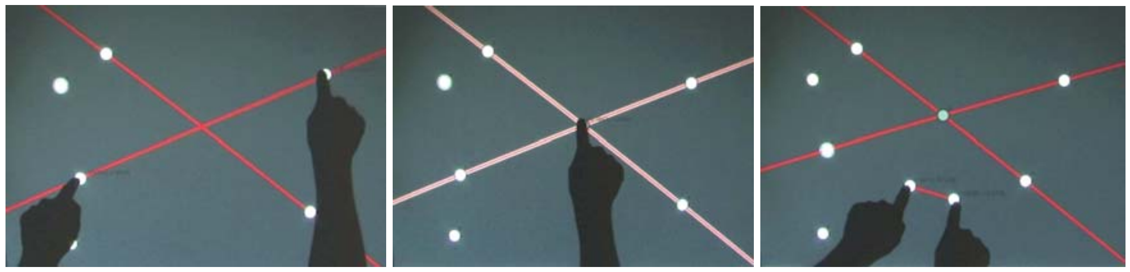

Figure 3–5. *Creating a line using two fingers; creating the intersection point; creating a line segment*

5. Educational issues

In general there is a trend in mathematics education to pay more attention on the understanding of concepts instead of building up a large repertoire of knowledge and skills. (NCTM 2000, KMK 2004). In particular students should be activated to establish deeper insights into mathematical concepts. Following this trend, the stated requirements have to be considered for ICT-based teaching and as well for the design concepts of "pedagogical-enriched" user interfaces.

5.1. Process Orientation

If we want children and learners in general to be able to learn by doing mathematics, we should remove actions that do not help them in achieving what they have in mind. Working with software does not constitute a value by itself. If the software is too hard to use, this constructive process can be slowed down unnecessarily or even stop. In that case, learning is impossible.

5.2. Emphasize Choices

For many people who are not fluent with ICT, computer software use can result in guesswork. Even for people working with computers in their jobs it is not uncommon to stick to a set of routines. If anything is not working as expected, these routines fail and the users are unable to proceed. This “blind” use of a tool has to be avoided when working with software in education. For a true understanding, students have to be able to make explicit choices of their next actions, they have to be able to understand the consequences of their actions *in advance* in order to be able to plan their work.

This being said, a well-designed user interface must help users to know what they are doing and emphasize their choices. Too much magic can be harmful here.

[2] The development of new user interfaces usually needs new software paradigms and design patterns. A premier example is Ian Sutherlands Sketchpad, which was developed in the 60s at MIT, which anticipates object-orientation and other techniques.

5.3. Work together

For students it is important to work together, not only to develop social skills, but also to be able to communicate about mathematics. Discussion and arguments are indispensable elements of teaching and learning. In ICT supported environments these interactions(!) are often neglected. There is only one mouse for every computer, even if students work in pairs then one is in charge. Bully-effects have been reported, also in conjunction with DGS (Hölzl 1994).

With Multi-Touch technology several users can work simultaneously on the same screen. It is even possible to develop cooperative exercises that can only be solved when two or more students help each other. Thus, MT has the potential to enhance social skills and communication skills.

As a final example, in Fig. 6 you see the prototypical software DOPPELMOPPEL based on Cinderella that enables elementary school children to work with virtual enactive and symbolic representations of numbers, their doubles and their halvings (Ladel & Kortenkamp 2008).

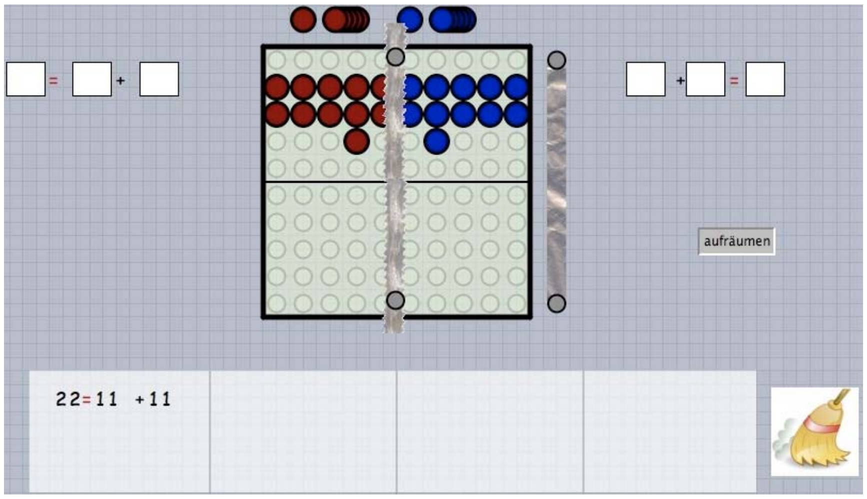

Figure 6. *DOPPELMOPPEL, prototype available at* ***http://kortenkamps.net/material/doppelmoppel***

6. Conclusion and Further Work

In this article we briefly discussed the central issues in user interface design for DGS. Some of these issues are still not solved satisfactorily, but there exist some strategies that now need usability testing. Pen-based devices and interactive whiteboards have been in use for some years now and already ask for completely different approaches. The recent introduction of Multi-Touch technology and its foreseeable broad availability shows even more opportunities for a pedagogically reasonable evolution of user interfaces.

Based on these first steps, it its now necessary to develop and evaluate MT user interfaces. In particular, we ask for a set of gestures that is designed to not only facilitate the use of geometry software, but that supports a deeper understanding of mathematical concepts.

Literature

[1] Brock, W. H & Price, M.H. (1980): Squared paper in the Nineteenth Century: Instrument of Science and Engineering, and Symbol of Reform in Mathematical Education, *Educational Studies of Mathematics*, 11, (4), 365-381, Springer Netherlands.

[2] Hendriks, M., Kortenkamp, U., Kreis, Y., Marquès, D. (2009): D3.6: I2G Common File Format v2. Technical Report, Intergeo Project, http://i2geo.net/files/D3.6-Common-File-Format-v2.pdf

[3] Hölzl, R. (1994): Im Zugmodus der Cabri-Geometrie. Interaktionsstudien und Analysen zum Mathematiklernen mit dem Computer, Deutscher Studien Verlag, Weinheim.

[4] Kittel, A. (2007): Dynamische Geometrie-Systeme in der Hauptschule. Eine interpretative Untersuchung an Fallbeispielen und ausgewählten Aufgaben der Sekundarstufe. Verlag Franzbecker, Hildesheim/Berlin.

[5] KMK (eds) (2004): Bildungsstandards im Fach Mathematik für den Mittleren Schulabschluss – Beschluss der Kultusministerkonferenz vom 4. 12. 2003, Wolters Kluwer, München.

[6] Kortenkamp, U. & Materlik, D. (2004): Pen-based Input of Geometric Constructions. Proceedings of the Math UI workshop 2004, Bialowicze, Poland. Available online at http://www.activemath.org/~paul/MathUI04/proceedings/Scribbling_Cinderella.html

[7] Kortenkamp, U. & Richter-Gebert, J. (1998): Geometry and Education in the Internet Age. In Ottmann, T. & Tomek, I. (eds.): Proceedings of ED-MEDIA 98, Freiburg: AACE.

[8] Kortenkamp, U. (1999): Foundations of Dynamic Geometry, Ph.D. thesis, ETH Zürich.

[9] Kortenkamp, U., Materlik, D. (2004): Pen-based Input of Geometric Constructions, in Proceedings of MathUI 2004.

[10] Materlik, D. (2003): Using Sketch Recognition to Enhance the Human-Computer Interface of Geometry Software, Diploma thesis, FU Berlin.

[11] Moscovich, T. (2007): Principles and Applications of Multi-touch Interaction, Ph.D. thesis, Brown University.

[12] NCTM (2000): Principles and Standards for School Mathematics, Reston.

[13] Raskin, J. (2000): The Humane Interface: New Directions for Designing Interactive Systems, Addison Wesley.

[14] Richter-Gebert, J. & Kortenkamp, U. (1999): User Manual for The Interactive Geometry Software Cinderella, Springer-Verlag, Heidelberg.

[15] Richter-Gebert, J. & Kortenkamp, U. (2006): The Interactive Geometry Software Cinderella, v 2.0, http://cinderella.de

Authors

Ulrich Kortenkamp, Centre for Educational Research in Mathematics and Technology (CERMAT), University of Education Karlsruhe, Germany, e-mail: kortenkamp@cermat.org

Christian Dohrmann, CERMAT, Karlsruhe, Germany, e-mail: dohrmann@cermat.org

Biographical Notes

Ulrich Kortenkamp is full professor for Mathematics and Education at the University of Education Karlsruhe, Germany, and Head of the Centre for Educational Research in Mathematics and Technology (CERMAT). He is working at the interface of mathematics, computer science, and education. His primary research interests are in Interactive Geometry and its applications to teaching and learning.

Christian Dohrmann is a graduate student for Mathematics and Education at the University of Education Schwäbisch Gmünd, Germany and at CERMAT, Karlsruhe. His research focus is the development of ICT based learning environments for mathematics and computer science, with a particular emphasis on hardware and software design for Interactive Geometry.