# The Poor Man's Eyetracker Tool of ActiveMath

Carsten Ullrich and Erica Melis
DFKI Saarbrücken, Germany
cullrich—melis@dfki.de

**Abstract**

We address the problem of tracking some of the student's activities in an e-learning system to obtain data that can be used to diagnose and to react appropriately to the student's actions. In order to find out how long a student deals with a single paragraph/text-unit on a web-page we developed a poor man's eye tracker (PME) software, DFKeye. This tool can be configured for several appearances. The usage of a PME avoids the drawbacks of hardware eyetrackers that are not usable in an everyday learning environment.

## 1 Motivation

In empirical psychology, eyetrackers are a common tool for tracking the user's attention and for investigating design and usability features of computational systems and their user interfaces, see, e.g., [3]. These eyetracker facilities can, however, only be used in a laboratory, are difficult to adjust, and impose restrictions on the user. Therefore, their usage outside the lab, e.g., in everyday e-learning or other non-experimental activities, is impractical. Furthermore, the very detailed information of the user's visual focus which a hardware eyetracker provides is even too detailed for e-learning diagnosis purposes and has to be abstracted.

Nevertheless, for a diagnosis in an e-learning system, information about the user is required to appropriately determine the system's user-adaptive behavior and its suggestions. This information can partly be deduced or estimated from the student's learning activities. Especially the learner's performance in exercises and her navigation can be evaluated to estimate her capabilities without too much intrusion. Therefore, many intelligent systems use those activities to update the user model.

For reading and watching activities detailed information is rarely handled; some approaches measure the reading time for complete pages. These times can be compared to dynamically calculated optimal reading times, see for instance [5]. However, as the taken time refers to a complete page, it does not allow any conclusion about *parts* of a page, e.g., whether/which parts of the text have been skipped. Another disadvantage is the static nature of the observation. That is, when the student has worked on a page and goes to the cafeteria to relax before she resumes learning, then there is no way to distinguish the actual reading time from her time in the cafeteria.

We know of only one attempt that uses a finer grained measurement. That is Conati's masking interface that makes time taking for single problem solving steps possible by covering or fading the text of the steps [2]. Conati's interface is, however, implemented inside the Andes system rather than stand-alone and is not browser-based, and hence not reusable. Moreover, it has the disadvantage of hiding all the text that is outside the direct focus of attention. Since all the other steps in the worked-out example are invisible, the student no longer has an overview on the structure of the solution. The cafeteria problem arises, too.

We designed and implemented a PME that serves the learning environment ACTIVEMATH [4]. ACTIVEMATH is a generic web-based learning environment that dynamically generates interactive (mathematical) courses adapted to the student's goals, preferences, capabilities, and knowledge. The content is represented in an semantic XML-based format. It consists of units such as definition, example, exercise, or remark. For each user, the appropriate content is retrieved from a knowledge base and the course is generated individually according to pedagogical rules. The course is rendered to HTML and presented to the user via a standard web-browser. The student's interactions are used to update the user model and analyzed by a next-best suggestion mechanism that provides the user with hints about which actions to take next. The quality of the user modeling and the suggestions strongly depend on their input, that is the user's behavior. The more accurate it is tracked, the higher the quality.

Therefore, we needed and designed a browser-based time tracking facility that

- can provide the time spent on a *single* unit of a HTML-page rather than on the page,
- acts as a *stand-alone* module and communicates with other modules, e.g., with the user model;
- does not hide or change but *keeps and highlights the structure* of the example or document at hand,
- is *parameterizable* because different users might react differently to different PMEs, and finally
- is *dynamic* and notices when the user does not pay attention to the learning document.

To the best of our knowledge such a tool does not exist yet. In the remainder of this paper we describe this tool.

## 2 Poor Man's Eyetracker DFKeye

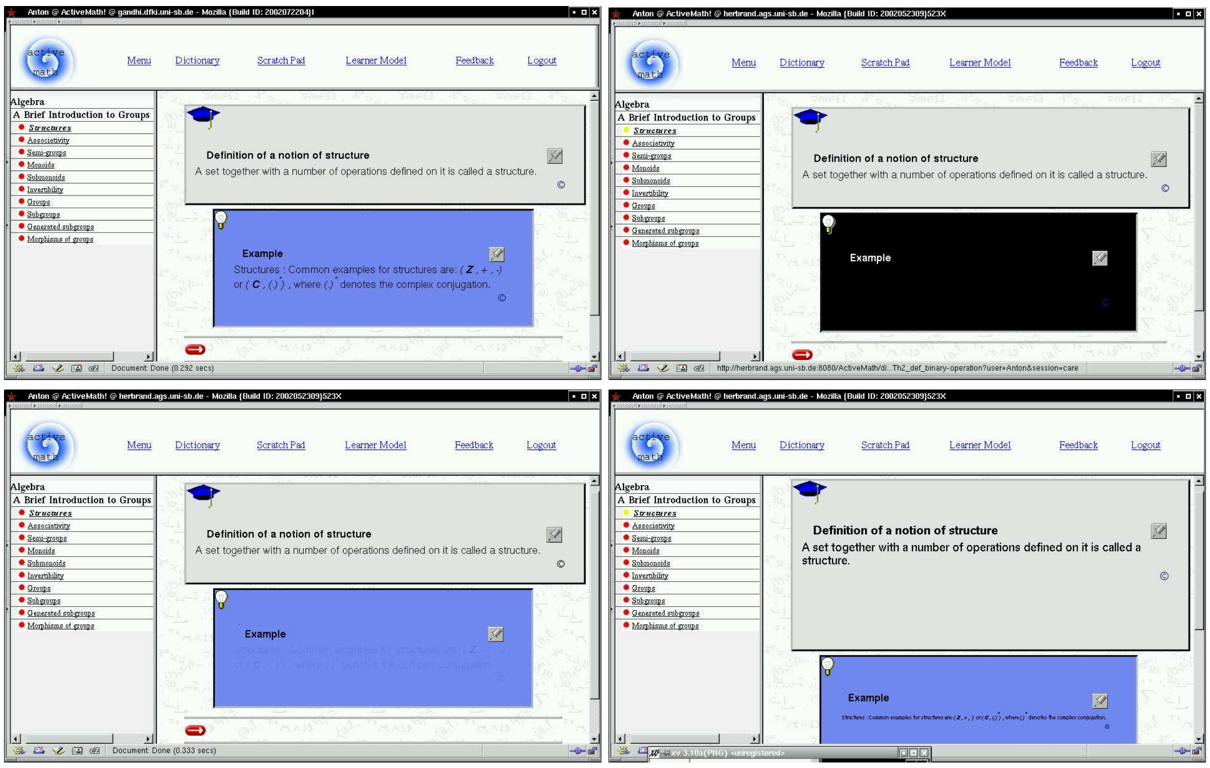
Human-computer interaction research reports experiences [1] on that and how the mouse movement follows the eye movements. Therefore, it is natural (rather than an additional effort) to expect from the learner to move the mouse as she goes along with reading a document such that the mouse location indicates the current focus of attention. However, some student do not move the mouse in an appropriate fashion and thus have to be softly forced to do so.

The information that can be expected from such a mouse movement is less detailed than the information of an eyetracker or, put differently, at a more abstract level, namely the order in which the elements have been focused and the duration of the focus. This does not harm its envisioned applications in e-learning, however.

The ability to measure the time a student takes to deal with a text-unit requires a structure of the web-presentation. This structure naturally arises from ActiveMath's underlying knowledge representation that represents the content not as complete pages but as single paragraphs/units.

When ActiveMath displays a page, the PME makes the text contained within a unit (slightly) unreadable. Three different variants have been implemented: *cover* (puts a black cover over the text), *fade* (changes the font color to nearly match the background color), and *zoom* (changes the font size to miniscule). If the learner focuses a unit by moving the mouse pointer on it, the PME unhides its text and starts a timer. If a unit looses the focus as the mouse pointer is moved away from it, its timer is stopped and its value is send to the DFKeye web-server.

The delay between touching the unit and focusing it is parameterizable (per default 0.7 seconds). We have chosen to not change the focus immediately as we wanted to minimize the amount of irrelevant data: often the learner moves from one unit to another one that is not directly adjacent. In this case she doe not focus on in-between units for which therefore no time information is taken. To tackle the cafeteria problem, a unit automatically looses the focus after a parameterizable delay (per default 30 seconds). If the student moves the mouse again, the unit regains the focus. Both delay times are not yet empirically confirmed.



Definition of a notion of structure

A set together with a number of operations defined on it is called a structure.

Example

Structures : Common examples for structures are: ( **Z** , + , ·) or ( **C** , ( )̄ ) , where ( )̄ denotes the complex conjugation.

Figure 1: ACTIVEMATH eyetracker variants: none (top left), cover (top right), fade (bottom left) and zoom (bottom right)

Let's have a look at the example in Figure 1. It shows the same page in ACTIVEMATH in the different PME variants. The visual separation of the paragraphs reflects the structured representation of the content: Each paragraph corresponds to a proper unit. The learner focuses the upper unit, the definition of a mathematical concept. if no eyetracker is used, the complete page is readable (top left). Otherwise the text in the example (the paragraph below) is covered, faded (only faintly visible in the figure), or shrunken, resp. The structure of the page remains visible, though. The icon and the color scheme clearly indicate the type of a unit and its title stays readable.

Thanks to the PME, ACTIVEMATH can analyze and react to the learner's reading behavior. For instance, if the learner focuses an example over and over again and then solves an exercise correctly, ACTIVEMATH can assume that she has studied the example throughoutly and the user model strongly increases the user's application level of the concept the exercise is for. Otherwise, if the learner never touches the example and subsequently performs poorly on the exercise, ACTIVEMATH's suggestion mechanism can invite her to look at the skipped example.

**Technical Realization** ACTIVEMATH integrates the PME as a Javascript application. It mainly uses `onMouseOver`/`onMouseOut` event handlers on `HTML div` elements that catch the mouse movements. The time information is send to a servlet (java server application) immediately after a unit looses the focus. The servlet sends the data to the interested modules such as the user model and a suggestion mechanism.

The PME was implemented in strict conformance to the W3C standards, and is usable in all W3C-compliant browsers (Mozilla, Netscape6) and in IE5. We implemented a PME-kit that can be easily integrated into existing web-based systems.

## 3 Conclusion

Although the information that a PME provides is less detailed than the information of an eyetracker, this does not harm its application in e-learning because the less detailed information corresponds to an abstraction of the common eyetracker information that is useful for e-learning diagnosis.

The tool can be used for other purposes for which the level of information provided by the PME is appropriate. For instance, we are currently conducting an experiment in which we investigate to what extend the PME can be used to evaluate design and usability issues.

The PME's implementation of ACTIVEMATH runs reliably since March 2002. A demo is available under `http://www.activemath.org/demo`. Experiments are being conducted with the different PME versions in order to find out which one is the most appropriate for a particular purpose and type of user and which is least distracting.

**Acknowledgment** We thank Michael Dietrich for his work for the implementation of various PM eyetrackers in ACTIVEMATH.

## References

[1] M.C. Chen, J.R. Anderson, and M.H. Sohn. What can a mouse cursor tell us more? correlation of eye/mouse movements on web browsing. In *Computer Human Interaction (CHI) 2001*, pages 280–281, 2001.

[2] C. Conati and K. VanLehn. Teaching meta-cognitive skills: Implementation and evaluation of a tutoring system to guide self-explanation while learning from examples. In S.P. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education*, pages 297–304. IOS Press, 1999.

[3] S. Lajoie and M. Vivet, editors. *Artificial Intelligence and Education*. IOS Press, 1999.

[4] E. Melis, J. Büdenbender, E. Andres, Adrian Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich. Activemath: A generic and adaptive web-based learning environment. *Artificial Intelligence and Education*, 12(4), 2001.

[5] M. H. Ng, W. Hall, P. Maier, and R. Armstrong. Using effective reading speed to integrate adaptivity into web-based learning. In *AH 2002*, LNCS, pages 428–431. Springer, 2002.