Introducing Eye Movement Modeling Examples for Programming Education and the Role of Teacher's Didactic Guidance

Selina N. Emhardt
Department of Educational Sciences, Open Universiteit, Heerlen, the Netherlands
Selina.emhardt@ou.nl

Halszka Jarodzka
Department of Educational Sciences, Open Universiteit, Heerlen, the Netherlands
halszka.jarodzka@ou.nl

Christian Drumm
Faculty of Business Studies, FH Aachen University of Applied Sciences, Aachen, Germany
drumm@fh-aachen.de

Tamara van Gog
Department of Education, Utrecht University, Utrecht, the Netherlands
t.vangog@uu.nl

Saskia Brand-Gruwel
Board of Governors, Zuyd Hogeschool, Heerlen
saskia.brand@zuyd.nl

ABSTRACT

In this article, we introduce how eye-tracking technology might become a promising tool to teach programming skills, such as debugging with 'Eye Movement Modeling Examples' (EMME). EMME are tutorial videos that visualize an expert's (e.g., a programming teacher's) eye movements during task performance to guide students' attention, e.g., as a moving dot or circle. We first introduce the general idea behind the EMME method and present studies that showed first promising results regarding the benefits of EMME to support programming education. However, we argue that the instructional design of EMME varies notably across them, as evidence-based guidelines on how to create effective EMME are often lacking. As an example, we present our ongoing research on the effects of different ways to instruct the EMME model prior to video creation. Finally, we highlight open questions for future investigations that could help improving the design of EMME for (programming) education.

CCS CONCEPTS

• **Applied computing** → Education; Computer-assisted instruction; • **Social and professional topics** → Professional topics; Computing education; • **Human-centered computing** → Visualization; Visualization techniques.

KEYWORDS

Eye Movement Modeling Examples, eye tracking, instructional design, expertise

ETRA '20 Short Papers, June 2–5, 2020, Stuttgart, Germany

ACM ISBN 978-1-4503-7134-6/20/06…$15.00
https://doi.org/10.1145/3379156.3391978

ACM Reference Format:
Selina N. Emhardt, Halszka Jarodzka, Christian Drumm, Tamara van Gog, and Saskia Brand-Gruwel. 2020. Introducing Eye Movement Modeling Examples for Programming Education and the Role of Teacher's Didactic Guidance. In *Symposium on Eye Tracking Research and Applications (ETRA '20 Short Papers), June 2–5, 2020, Stuttgart, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3379156.3391978

1 INTRODUCING THE CONCEPT OF EMME

Nowadays, online tutorials in which programmers show how to perform a task (e.g., how to use a debugger) are rapidly spreading on platforms such as YouTube, Coursera or Khahn Academy. These videos traditionally consist of screencasts with voice-overs of the model's (e.g., an expert teacher's) narration that explain how they perform the task [Kefalas and Stamatopoulou 2018]. From educational research, we know that learning by observing a knowledgeable model is a powerful way of learning [Renkl 2014]; [van Gog et al. 2019].

New technological advances have the potential to enrich screencast tutorial videos and foster learning. Eye trackers can detect and display where people look over the course of time and have recently become more affordable and easy to use. Eye Movement Modeling Examples (EMME) make use of this technology to create learning videos that display the teacher's eye movements during exemplary task performance to the learners [van Gog et al. 2009]. To create these videos, a programmer records his or her activities on the screen, usually along with verbal explanations. Additionally, an eye-tracking device continuously captures where the programmer looks on the screen at each moment in time. These two are then combined to create the EMME: the screen-recording video with a superimposed visualization of the recorded eye movements (e.g., e.g., as moving dot, circle, or spotlight [Jarodzka et al. 2012]; [Jarodzka et al. 2013]. Figure 1 shows an exemplary screenshot of an EMME that visualizes a programmer's focus of visual attention (fixations and saccades) during code debugging as a grey, moving dot. In the subsequent section, we present the basic idea behind using these eye-movement visualizations to foster learning and outline empirical studies about the effectiveness of EMME in the specific domain of programming research.

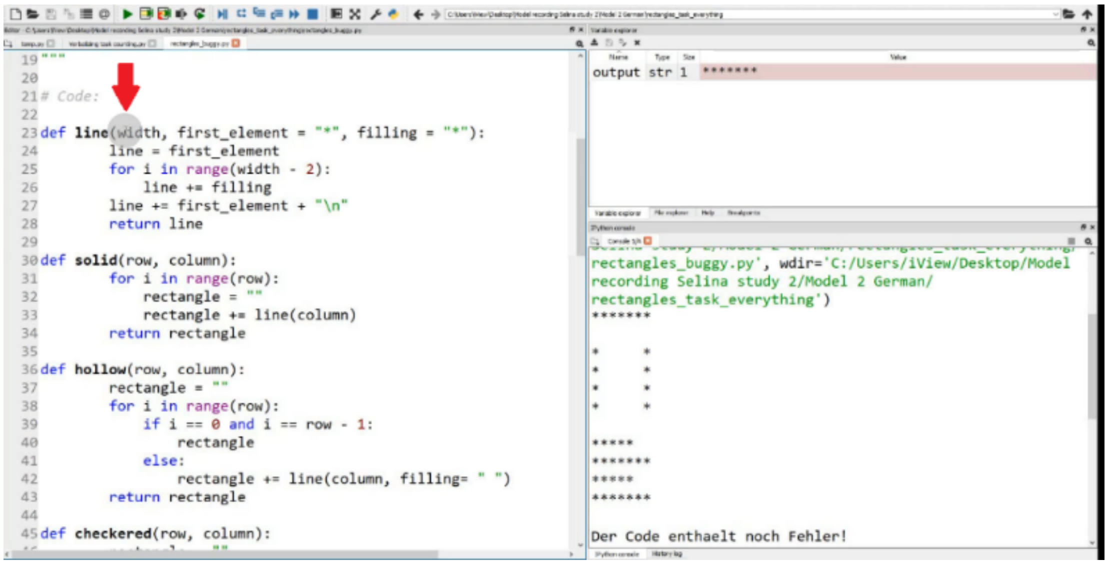

Figure 1: Screenshot of an EMME video in which the model's focus of visual attention during code debugging is displayed as gray, moving dot (see red arrow).

2 WHY EMME MIGHT FOSTER LEARNING

While observing modeling examples is a powerful way of learning, it is important that both, the teacher and the learner attend to the same objects at the same time and thereby establish a state of so-called 'joint attention' (see e.g., [Butterworth and Jarrett 1991]], [Tomasello and Farrar 1986]). For instance, [Sharma et al. 2019] recently showed that when watching learning video about programming in Massive Open Online Courses , higher levels of student-teacher co-attention were associated with higher student performance and motivation.

However, establishing joint attention between an expert acting as the teacher and the inexperienced students can be challenging. Various studies from expertise research reveal that experts process information faster and more globally and can, hence, naturally attend faster to more relevant task elements than novices (for exemplary reviews see [Reingold and Sheridan 2011] or [Sheridan and Reingold 2017]). In line with this, eye-tracking studies from programming research found that programmers with different levels of expertise look at other code elements [Aschwanden and Crosby 2006]. For instance, more experienced programmers focus more on complex statements [Crosby et al. 2002] but less on code comments than less experienced programmers [Crosby and Stelovsky 1990]. In a similar manner, expert teachers and novice learners might often attend to different task elements in learning videos - especially when the task contains a lot of complex, visual information (e.g., long, pre-written computer code that needs to be debugged).

In order to foster joint attention, expert models might try to bridge the discrepancy in (visual) attention allocation through verbal explanations. However, experts' problem-solving processes might often be automatized and therefore difficult to verbalize and their actual explanations might not always be sufficient to guide their students' attention to the relevant task elements. In some cases, for instance, the learners might not understand all terms that an expert naturally uses (e.g., referring to an empty list when referring to the '[ ]'-symbol in Python). Additionally, experts tend to automatize some of their task-solving processes, often without awareness [Samuels and Flor 1997]. This makes it difficult to communicate these processes to beginners. In such cases, novice learners might still experience severe difficulties trying to 'catch up' with the expert model in the video.

Displaying the expert model's eye movements in EMME might help to guide the learners' attention to the relevant task elements and can clarify what the model is referring to in tutorial videos. In the case in which an expert programmer talks about an empty list, learners would additionally see that they look at the [ ]-symbol in an EMME.

First findings from programming research reveal promising effects of displaying a task performer's eye-movements to foster learning and understanding. In the study of [Bednarik et al. 2018], students saw a video in which an advanced programmer performed a source-code comprehension task in a successful manner. The videos displayed the programmer's eye movements during task-performance and included verbal explanations of another person. These verbalizations didactically highlighted beneficial strategic aspects of the programmer's code reading behavior (based on the block model of code comprehension of [Schulte 2008]). Learners who watched these videos later showed better code comprehension performance than learners who did not watch a video with eye-movement displays and explanations. In another study by [Stein and Brennan 2004], observing displays of a professional programmer's eye movement - this time without any (didactic) verbalizations - helped other professional programmers to more efficiently solve the same debugging tasks later. While these findings suggest that displays of another task performer's (here programmer's) eye movements can foster understanding, there are many unanswered questions about how to design the most effective EMME tutorials.

3 DESIGN GUIDELINES TO CREATE EFFECTIVE EMME: THE EXAMPLE OF MODEL INSTRUCTIONS

Until now, the creation process of EMME videos varies notably across studies - often with unknown effects on learning. We therefore argue that there is a need for empirical studies that investigate ways to design effective EMME for (programming) education. As an example, we now present our research on the effects of instructing EMME models in different manners prior to their video creation: When comparing different EMME studies, it is striking that the instructions that the models receive prior to creating the videos differ substantially. In some studies, models received specific instructions to explain their performance in a 'didactic' manner that is understandable for an audience with little prior knowledge [Jarodzka et al. 2012]; [Jarodzka et al. 2013]. In contrast, models in other studies received no instructions and therefore simply solved tasks in their regular manner, i.e., naturally (e.g., [Litchfield et al. 2010; Stein and Brennan 2004]). However, it was yet unknown whether and how these different instructions affect EMME displays and students' learning outcomes.

We recently conducted two empirical studies to investigate the effects of using different model instructions (i.e., natural vs. didactic) on EMME displays and learning. In our first study [Emhardt et al. submitted], we explored whether and how programming experts would alter their natural debugging behavior (i.e., eye-movements and mouse clicks that are displayed in EMME) when being instructed to behave didactically. Additionally, we compared experts' and novices' regular debugging behavior to be able to draw conclusions about the direction of experts' changes when behaving didactically (becoming more or less similar to novices).

First, programming novices and experts debugged unknown, short Python code snippets without any instruction (natural) while verbalizing their thoughts. After solving each task, the group of experts was additionally instructed to explain their solution in a more didactic manner to a group of fictitious programming beginners (instructions based on [Jucks et al. 2007]). We first compared the eye and mouse movements of naturally behaving experts with novices' behavior and then investigated experts' behavioral changes when being instructed to behave didactically. Our results (details described in Emhardt et al. [submitted]) showed that the behavior of naturally behaving experts during debugging differed substantially from that of novices. For instance, naturally behaving experts had shorter fixations in the code area, made fewer transitions between the code and the output area when updating the output information, ran the code less frequently and showed significantly more linear (approximately line-wise) code processing in comparison to novices.

In a next step, we aimed to answer the question of whether experts' non-verbal behavior becomes more similar to novices' behavior when behaving didactically. Indeed, experts' eye movements and mouse clicks substantially changed when they performed the task didactically compared to naturally. Experts became more similar to novices on measures that we associated with experts' automatized processes (i.e., having longer fixations in the code area, more transitions between code and output per click on the run button when behaving didactically). One explanation might be that adaptation makes it easier for novices to follow or imitate the expert behavior. In contrast, experts became less similar to novices for measures associated with more strategic behavior when behaving didactically (i.e., even more linear code reading, even fewer clicks on run button). We concluded that model behavior (natural vs. didactic) strongly influences the characteristics of the EMME displays with unknown effects on learning.

In a second study, we are currently investigating the effects of displaying natural and didactic model behavior in EMME about code debugging on student learning [Emhardt et al. in preparation]. Unexpectedly, preliminary analysis do not find effects of displaying these different kinds of model behavior on students' video understanding and later debugging performance. One possible explanation for these findings is that the videos in both conditions affect learners' understanding to a similar extent, but presumably through different mechanisms. While natural model behavior might stimulate deeper reflection and insights into authentic expert behavior, didactic EMME could ease understanding and could leave more cognitive capacities to process relevant information. It is important to note, though, that even thought we did not compare the general effects EMME on learning (i.e., comparing learning with EMME vs. regular tutorial videos) within our own studies as these effects were already shown elsewhere (e.g., [Bednarik et al. 2018]). Our preliminary finding that using different model instructions (natural vs. didactic) did not to affect students' learning with EMME videos would provide researchers and practitioners with freedom to design their tutorial videos with different kinds of model instructions.

4 CONCLUSION AND FUTURE RESEARCH QUESTIONS FOR EMME RESEARCH

In this article, we introduced the concept of EMME and provided a brief overview on why EMME are assumed to foster learning. First studies have found promising results regarding the benefits of learning with EMME, for instance in the domain of programming education (e.g., [Bednarik et al. 2018]). However, we argue that guidelines on how to create an effective EMME are often missing and, therefore, EMME characteristics differ substantially across studies. The effects of different EMME designs on learning are until now mostly unknown. We argued that systematic investigations on how to create effective design guidelines to create EMME are therefore of high importance. To provide examples of such investigations, we outlined two of our recent empirical studies. These studies focused on the effects of instructing EMME models to behave in a didactic, learner-focused manner on EMME displays and learning [Emhardt et al. submitted]; [Emhardt et al. in preparation]. We found that the characteristics of EMME displays (i.e., the displayed mouse and eye movements) change substantially with model instruction. However, preliminary results of our subsequent study indicate that the displayed model behavior (natural vs. didactic) does not affect programming learning. Future studies should generalize these findings to a broader variety of task materials and situations. In this context, they could investigate the effects of learners' prior knowledge levels on learning with EMME that display natural or didactic model behavior. In addition, future studies could explore the effects of other design choices when creating EMME, such as the type of eye movement visualizations (i.e., the effects of using a moving dot, circle, heat map or spotlight that blurs out the rest of the screen, see e.g., [Jarodzka et al. 2012]). Finally, future studies could aim to adopt the EMME method in more authentic educational practices. Could EMME, for instance, be useful for self-study purposes at home, as addition to students' regular programming classes or could they even enrich programming lectures? We hope that this overview article inspires programmers and programming teachers to consider eye-tracking technology, and EMME in particular, as promising educational tool and topic of future investigations.

ACKNOWLEDGMENTS

This research was funded by a grant from The Netherlands Initiative for Education Research (NRO-PROO # 405-17-301).

REFERENCES

Christoph Aschwanden and Martha Crosby. 2006. Code scanning patterns in program comprehension. In *Proceedings of the 39th Hawaii International Conference on System Sciences (HICSS '06)*. Kauai, Hawaii, 2006.

Roman Bednarik, Carsten Schulte, Lea Budde, Birte Heinemann, and Hana Vrzakova. 2018. Eye-movement modeling examples in source code comprehension: a classroom study. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling '18)*. Association for Computing Machinery, New York, NY, 1-8. DOI: https://doi.org/10.1145/3279720.3279722.

Georg Butterworth and Nicholas Jarrett. 1991. What minds have in common is space: Spatial mechanisms serving joint visual attention in infancy. *British Journal of Developmental Psychology* 9, 1 (March 1991), 55-72. DOI: https://doi.org/10.1111/j.2044-835X.1991.tb00862.x.

Selina N. Emhardt, Ellen Kok, Halszka Jarodzka, Saskia Brand-Gruwel, Christian Drumm and Tamara van Gog. Manuscript submitted to Cognitive Science. How experts adapt their gaze behavior when modeling a task to novices.

Selina N. Emhardt, Halszka Jarodzka, Saskia Brand-Gruwel, Christian Drumm and Tamara van Gog. Manuscript in preparation.

Martha E. Crosby, Jean Scholtz, and Susan Wiedenbeck. 2002. The roles beacons play in comprehension for novice and expert programmers. In*14th Workshop of the Psychology of Programming Interest Group (PPIG14)*, 58–73. DOI: https://doi.org/10.1.1.106.4959.

Martha E. Crosby and Jan Stelovsky. 1990. How do we read algorithms? A case study. *Computer* 23, 1 (January 1990), 25-35. DOI: https://doi.org/10.1109/2.48797.

Halszka Jarodzka, Thomas Balslev, Kenneth Holmqvist, Marcus Nyström, Katharina Scheiter, Peter Gerjets, and Berit Eika. 2012. Conveying clinical reasoning based on visual observation via eye-movement modelling examples. *Instructional Science* 40, 5 (March 2012), 813-827. DOI: https://doi.org/10.1007/s11251-012-9218-5.

Halszka Jarodzka, Tamara van Gog, Michael Dorr, Katharina Scheiter, and Peter Gerjets. 2013. Learning to see: Guiding students' attention via a model's eye movements fosters learning. *Learning and Instruction* 25 (June 2013), 62-70. DOI: https://doi.org/10.1016/j.learninstruc.2012.11.004.

Regina Jucks, Petra Schulte-Löbbert, and Rainer Bromme. 2007. Supporting experts' written knowledge communication through reflective prompts on the use of specialist concepts. *Zeitschrift für Psychologie/Journal of Psychology*, 215 (first published online February 2015), 237-247. DOI: https://doi.org/10.1027/0044-3409.215.4.237.

Petros Kefalas and Ionana Stamatopoulou. 2018. Using screencasts to enhance coding skills: the case of logic programming? *Computer Science & Information Systems* 15, 3 (October 2018), 775-798. DOI: https://doi.org/10.2298/CSIS180312028K.

Damien Litchfield, Linden J. Ball, Tim Donovan, Donovan J. Manning, and Trevor Crawford. 2010. Viewing another person's eye movements improves identification of pulmonary nodules in chest x-ray inspection. *Journal of Experimental Psychology: Applied* 16, 3 (September 2010), 251-262. DOI: https://doi.org/10.1037/a0020082.

Eyal M. Reingold and Heather Sheridan. 2011. Eye movements and visual expertise in chess and medicine. In *Oxford Handbook on Eye Movements*, Simon P. Liversedge (Ed.). Oxford: Oxford University Press, Oxford, 528-550. DOI: https://doi.org/10.1093/oxfordhb/9780199539789.013.0029.

Alexander Renkl. 2014. The worked examples principle in multimedia learning. In *The Cambridge Handbook of Multimedia Learning*, Richard E. Mayer (Ed.). Cambridge University Press, Cambridge, 391-412. DOI: https://doi.org/10.1017/CBO9781139547369.020.

Jay S. Samuels and Richard F. Flor. 1997. The importance of automaticity for developing expertise in reading. *Reading & Writing Quarterly: Overcoming Learning Difficulties* 13, 2 (published online July 2006), 107-121. DOI: https://doi.org/10.1080/1057356970130202.

Carsten Schulte. 2008. Block Model: an educational model of program comprehension as a tool for a scholarly approach to teaching. In *Proceedings of the Fourth international Workshop on Computing Education Research (ICER'08)*. 149-160, DOI: https://doi.org/10.1145/1404520.1404535.

Kshitij Sharma, Pierre Dillenbourg, and Michail Giannakos. 2019. Stimuli-based gaze analytics to enhance motivation and learning in MOOCs. In*2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT*), IEEE, Maceió, Brazil, 199-203. DOI: https://doi.org/10.1109/ICALT.2019.00052.

Heather Sheridan and Eyal M. Reingold. 2017. The holistic processing account of visual expertise in medical image perception: a review. *Frontiers in Psychology* 8, 1620 (September 2017). DOI: https://doi.org/10.3389/fpsyg.2017.01620.

Randy Stein and Susan E. Brennan. 2004. Another person's eye gaze as a cue in solving programming problems. In *Proceedings of the 6th international conference on Multimodal interfaces (ICMI'04)*, Association for Computing MachineryNew York, NY, 9-15, DOI: https://doi.org/10.1145/1027933.1027936.

Michael Tomasello and Michael J. Farrar. 1986. Joint attention and early language. *Child Development* 57, 6 (December 1986), 1454-1463. DOI: https://doi.org/10.2307/1130423.

Tamara van Gog, Halszka Jarodzka, Katharina Scheiter, Peter Gerjets, and Fred Paas. 2009. Attention guidance during example study via the model's eye movements. *Computers in Human Behavior* 25, 3 (May 2009), 785-791. DOI: https://doi.org/10.1016/j.chb.2009.02.007.

Tamara van Gog, Nikol Rummel, and Alexander Renkl. 2019. Learning how to solve problems by studying examples. *The Cambridge Handbook of Cognition and Education*, John Dunlosky (Ed.). Cambridge University Press, Cambridge, 183-208. DOi: https://doi.org/10.1017/9781108235631.009.