Animated Examples as Practice Content in a Java Programming Course

Roya Hosseini
University of Pittsburgh
roh38@pitt.edu

Teemu Sirkiä
Aalto University
teemu.sirkia@aalto.fi

Julio Guerra
University of Pittsburgh
jdg60@pitt.edu

Peter Brusilovsky
University of Pittsburgh
peterb@pitt.edu

Lauri Malmi
Aalto University
lauri.malmi@aalto.fi

ABSTRACT

Code examples are commonly used learning resources that help students grasp various programming structures and concepts. However, example code usually requires explanations about what each line or part of the code does. Otherwise, students may find it difficult to follow an example. In this paper, we compare two types of code examples that use different techniques to describe important concepts in the code: annotated and animated examples. The former displays an explanation for a subset of lines in plain text, whereas the latter visualizes code execution. We studied the use and impact of these enhanced examples, provided as non-mandatory practice content, in three introductory Java courses. Our results suggest that animated examples are more engaging and have a positive impact on students' learning. As compared to annotated examples, students spent more time with animated examples and more likely completed them. Also, a positive relationship was found between the number of explored animated examples and the overall course grade.

Keywords

code examples; learning; Java programming; annotated examples; animated examples; program visualization

1. INTRODUCTION

Understanding program dynamics (i.e., how program execution is carried out in computer memory) is one of the central challenges of learning computer programming. Du Boulay [7] wrote "...there are difficulties associated with understanding the general properties of the machine that one is learning to control, the notional machine, and realizing how the behavior of the physical machine relates to this notional machine." These difficulties are generally related to the abstract nature of program execution.

CS teachers and researchers have developed numerous methods and tools to support students in learning program dynamics. One basic method is annotating program code to explain what happens when each line is executed. A more advanced method is *program visualization*, a whole research direction [15] that focuses on building specialized software tools and interactive techniques to support following and exploring program execution in a graphical form.

Even though sophisticated tools have been developed and evaluated, they frequently fail to meet their goals, because teachers and students simply do not use them [12]. Packaging interactive learning content into online practice systems such as Codingbat[1] has recently emerged as a way to decrease the teacher engagement threshold: teachers can now point students to these systems, rather than master these tools to integrate individual content items into their teaching. However, practice systems still face the problem of student engagement. Since practice content is usually offered for students' own benefit while awarding no additional credit, students tend to skip practice materials to focus on tasks that give them credits. Turning examples from practice to mandatory content is not an optimal solution either, because it can lead to mindless clicking through the content to gain points without any understanding. A relevant overall research question in this area remains: *what kind of non-compulsory learning resources are both beneficial and engaging for students in programming education?*

In this paper, we examine the value of animated program examples as practice content for both educational impact and prospects for engagement. To examine the added value of *animated examples*, we compare them to *annotated examples*, which are a more traditional way to support students in learning to understand programs. Both types of resources in our study were provided as voluntary practice materials.

2. RELATED WORK

A printed textbook, which could be considered as a traditional medium to present program examples to learners, provides little help in delivering the essence of an example or helping students to locate more relevant examples. At most, a textbook author could provide line-by-line example code explanations. This approach has been used in a number of textbooks, and in some cases, has defined the nature of a textbook [9]. To overcome these limitations and to provide better support for learning from examples, instructors and researchers have suggested a range of interactive computer tools. Roughly, the work on these tools could be classified

[1] http://codingbat.com/

SIGCSE '16, March 02-05, 2016, Memphis, TN, USA

DOI: http://dx.doi.org/10.1145/2839509.2844639

in three directions. One direction has focused on providing better access to the right example at the right time - although the examples could still be traditional, plain-code examples [6, 16]. The other two directions have focused on augmenting the examples themselves. One has focused on adding richer expert explanations to the example code by using interactive linking [4]. The other has attempted to promote deeper knowledge of language semantics by offering visual animations of example code [15]. Below, we briefly review some work along the latter two directions that are most relevant to the paper.

The concept behind interactive explained examples is to add expert comments on the role of different constructs in achieving the goal of the provided code, as well as other related comments. These kinds of worked-out examples are known in many other science and engineering domains; however, programming was one of the first areas where interactive systems were developed to support these kinds of examples. Starting from the pioneering work of Linn and Boyle [2, 10], a number of researchers explored systems that offer access to explained examples [4, 6]. In providing access to explanations without breaking the example code, these systems typically used interactive hypermedia features.

Animated examples, which fall in the area of program visualization, are a more advanced and complicated technology. The idea of animated examples is to make the dynamic execution or program examples visible. Without such visualization, novice students are not able to see how the given code is really executed, because the execution process is normally invisible. This may lead to different kinds of misconceptions that prevent learning new concepts, as the code does not behave as expected. Animated examples make the execution process visible so that novice students can easily see and follow all the important steps.

A recent review of the existing program visualization tools [15] indicates that many different tools have been developed for novice programmers to create animated examples. The earliest tools appeared in the 1980s; however, it is only over the past 10 years that this technology has become broadly available for instructors. One well-known program visualization system is Jeliot 3 [1] and its ancestors. They have been used for almost two decades in introductory programming. Jeliot 3 is a Java-based application for Java programs that can even visualize students' own code. Many research papers have presented positive learning results when Jeliot 3 is used in CS1 courses (eg. [5]). One newer tool for creating animated examples is the Online Python Tutor [8], which can create embeddable animated examples that run in a browser.

3. TOOLS USED IN THE STUDY

3.1 Mastery Grids Portal for Practice Content

All practice content in our study was accessed through the Mastery Grids portal [11], which integrates three kinds of content: code execution problems, annotated examples, and animated examples [4, 13]. In integrating external content, Mastery Grids portal follows the suggestions of the recent ITiCSE working group report [3]. Instead of directly embedding each kind of examples into the portal, all interactive examples are technically external resources that are hosted by separate content servers. This design allows for reuse of content in multiple portals or course management systems.

To engage students to work with the content, the Mastery Grids portal provides visual personal progress tracking, as well as social comparison visualizations, which allow a student to compare her progress to the progress of the rest of the class or with the most advanced students. Mastery Grids organizes the content in topics that are represented as a series of colored cells, which get darker as the student completes the content within a topic. A part of the topic cells of the programming course can be seen in Figure 1, where an animated example from the topic "Strings" is shown overlaying Mastery Grids. For a detailed explanation of the Mastery Grids interface, see previous work [11]. In line with its practice nature, the use of the system and any of its content was not mandatory.

3.2 Exploring Examples with Jsvee

Jsvee [13] is a JavaScript library to create animated examples. The library is language-independent, but can be extended to have language-specific features: for example, there is limited support for built-in Java libraries that are needed in the animations. The main idea of the Jsvee library is that the original code is transformed into an intermediate language that the library can visually execute. Jsvee is able to produce smooth expression-level visualizations, which means that all the intermediate steps to evaluate the current line are shown. For example, the order of evaluating arithmetic operations or nested function calls are visible. With textual descriptions or line-based visualization tools, such as Online Python Tutor [8], it is difficult to explain or demonstrate these important evaluation steps.

These interactive visualizations (see Figure 1) can be embedded into online course materials or used as individual small exercises. They are pure HTML, CSS, and JavaScript, which allows them to be also used with mobile devices, because plugins (such as Flash) are not required. Jsvee also automatically generates short explanation texts for the animation of each step.

To get information how the students use animated examples, Jsvee collects log traces. Every time when a student uses the controls, such as "step forward" or "undo," a log entry with the timestamp and current step number is created. Entries are sent to the server every 20 seconds, as well as after the example is completed. With this data it is possible to measure, for example, how much time students have spent on each step.

3.3 Exploring Examples with WebEx

WebEx is a web-based system for the interactive exploration of programming examples. The key idea behind the Webex system is to generate self-sufficient interactive examples by adding explanations to example code. With WebEx, the code of the example is shown as an easy-to-grasp single chunk, while line-by-line explanation are accessible by clicking on lines of interest. As a result, from being a passive reading activity, the work on examples changes to an interactive exploration.

Starting in 2001 with several dozen annotated programming examples in C, this technology has been used to produce a large volume of annotated examples for several programming languages (C, Java, SQL) with explanations that address both the semantics and pragmatics of the examples [4]. Figure 2 shows a typical WebEx example in Java for teaching students the concept of *pre/post-increment*. A green bullet to the left of a code line indicates the availability of

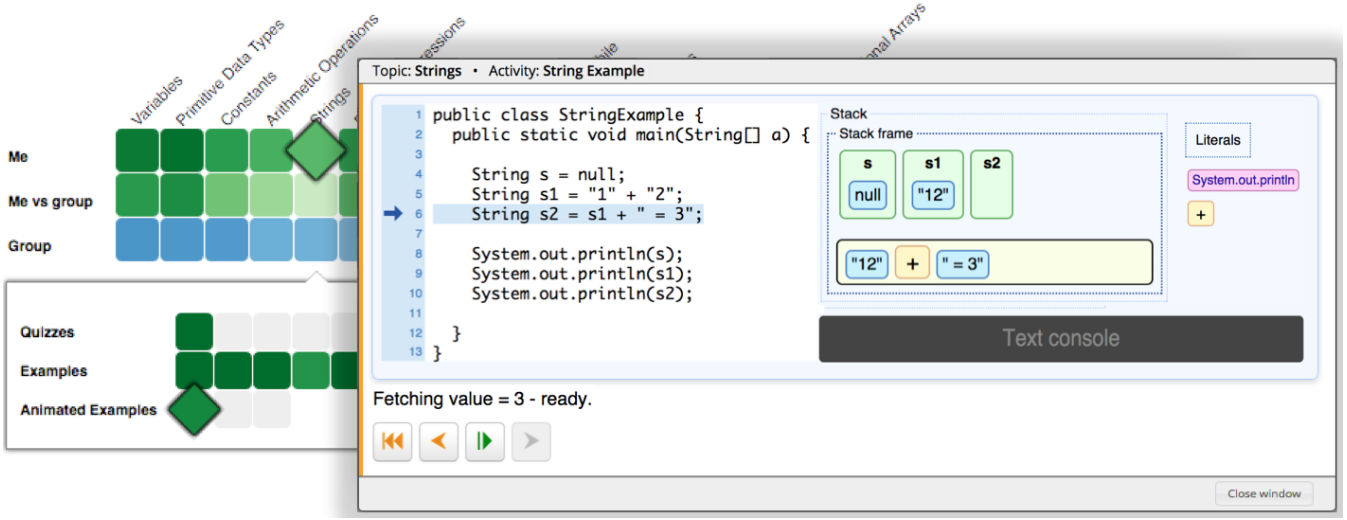

Figure 1: An illustration of an animated example in Mastery Grids. When a student clicks on an animated example, the activity is shown in an iframe that overlays Mastery Grids, and the student can click on the forward button to see the animations. For example, in the current line (line 6), the value of the variable `s1` will be concatenated with the literal value `"= 3"`. Jsvee demonstrates how the local variables are updated by showing all the intermediate steps of fetching values and evaluating the operator - not simply the final result.

explanations for this line. Explanations are shown as the student clicks on each bullet. The WebEx system logs all user interactions with the tool, which enables seeing how students work with examples in the context of real courses.

4. THE STUDY

To investigate the impact of different example types, we analyzed the data of classroom studies that were carried out at the University of Pittsburgh (PITT) and Winston-Salem State University (WSSU) during the fall 2014 and spring 2015 semesters. The subjects were undergraduate students taking an introductory course in Java programming. All students were informed by course instructors to access supplementary materials for the course through the Mastery Grids portal. A pretest and post-test was administered at the beginning and end of the course to measure students' learning at the end of the course. Both tests had the same questions.

The practice content offered by the course through the Mastery Grids portal included three types of interactive content, organized under 19 topics: annotated examples, animated examples, and parameterized code execution problem. A code execution problem provides a fragment of code to a student and asks about the value of a specific variable or the content printed on a console after the code is executed. The system evaluates the student's answer and reports whether the answer was correct or incorrect. Students can try the same problem many times. Each time, the code is generated with randomly selected values for the problem parameters, and as a result, the correct answer will be different each time. Table 1 provides more details for the number of students and practice content items in the studied groups.

5. THE RESULTS

We evaluated the impact of annotated and animated examples on students' engagement and learning from the following four aspects: engagement level, problem solving performance, amount of learning, and course grade. In the following subsections, we describe the analyses we performed to discover the importance of examples to these aspects of learning.

Table 1: Classroom studies: participants and course materials

	PITT 2014	PITT 2015	WSSU 2014
Students registered in the course	65	61	28
Students logged into the portal	56	33	20
Problems	101	111	111
Annotated examples	79	104	104
Animated examples	31	52	52

5.1 Engagement Level

Here, we evaluated the level of user engagement by defining a set of usage measures that summarize students' attempts on example activities. The measures that we used are as follows: (1) the number of distinct examples viewed; (2) the percentage of viewed examples (that is the number of distinct examples viewed divided by the number of available examples); (3) the total time spent on examples (seconds); (4) the average percentage of example completion; and (5) the average time spent per example. The average completion was calculated by dividing the average clicks actually made by the total number of clicks needed to view the whole example. Only started examples were taken into account. For WebEx examples, a click means an action to view an explanation, and for animated examples, a click means a step to move forward. Since the work with the system was non-mandatory, the amount of work with practice content done by the students offers a good approximation of engagement.

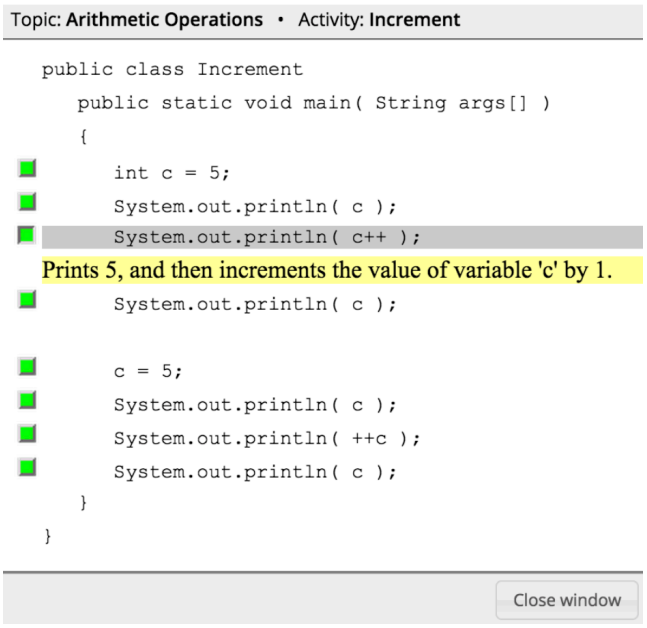

Figure 2: A WebEx example for demonstrating a *pre/post-increment* concept in the *Arithmetic Operations* topic. When a student clicks on the green bullets, an explanation appears below the line.

Table 2 provides summary statistics for the above measures. As the table shows, students viewed significantly more annotated examples than animated examples (Kruskal-Wallis test: $\chi^2(1) = 10.177, p = .001$). Also, the average total time spent on annotated examples was significantly higher than animated examples ($\chi^2(1) = 15.048, p < .001$). However, these numbers should be considered with care, since the number of more easy-to-author annotated examples in the system was considerably larger.

Looking at the percentage of example completion, we found that students completed an average of 95.0% of each animated example that they viewed. This was significantly larger (23.6% more) than the percentage of completed annotated examples ($\chi^2(1) = 29.424, p < .001$). Moreover, students spent twice as much time interacting with an animated example than with an annotated example ($\chi^2(1) = 78.667, p = .001$). It indicates that both kinds of examples were engaging; however, individual animated examples were more engaging, and motivated students to stay longer and follow more lines. Whether this longer work was beneficial for learning will be addressed in the rest of this section.

Note that annotated and animated examples differ considerably in their interactivity and freedom of navigation. Their usage should be compared with understanding this nature. In particular, it is easy to navigate through annotations in any order skipping less important lines, while the sequential nature of animations allows for comparatively little freedom, with no skipping or random access.

5.2 Problem Solving Performance

As mentioned earlier, the practice content offered by the course in the Mastery Grids portal included parameterized code execution problems for self-assessment of students' knowledge. We assumed that problem solving performance

Table 2: Comparison of engagement level between annotated & animated examples

Usage summary	Annotated ex. Mean (SE)	Animated ex. Mean (SE)
Distinct examples	29.7 (2.4)	8.7 (1.0)
Distinct examples (%)	33.0% (3.0%)	24.0% (3.0%)
Total time (sec)	2196.5 (285.6)	1585.7 (426.5)
Avg. completion (%)	71.4% (3.5%)	95.0% (1.5%)
Avg. time per example	61.1 (6.3)	128.2 (22.1)

(correctness of problem answers) could be used to evaluate student knowledge and attempted to explore the impact of annotated and animated examples on student knowledge by looking into the relationship between *success rate* in answering problems and the usage of different example types.

Success rate is defined as the number of correct attempts on problems divided by the total problem attempts. Using data from all students ($n = 109$), a significant negative correlation was found between the number of distinct annotated examples that students viewed and their success rate. A higher number of annotated example views was associated with a lower success rate ($\rho = -0.22$, $p = .029$). This result looks counter-intuitive, but it is important to remember that the content to practice is freely selected by students. As we observed, those failing to answer problems turned to examples in a hope to bridge the knowledge gap. Moreover, annotated examples were much more frequently accessed after a failure. Following 19.71% of failures to solve a problem, students seek help in easy-to-parse annotated examples, while in 3.91% of failures, they turned to animated examples. Considering all attempts to access each type of example, 20.01% of all annotated examples and 13.61% of animated examples were accessed immediately after failure, which largely makes failures a remediation tool, rather than just a learning tool. As a result, two opposite processes connecting the work with examples and performance took place. It is likely that examples do help students to increase knowledge (positive connection between student work with examples and performance), but on the other hand, due to free content choice, lower knowledge and failures led to the increased use of examples (negative connection between examples and performance). As we see, in case of annotated examples, the latter process overcame the effect of learning from examples.

We further investigated the parameters that influence problem solving performance by fitting stepwise regression models to predict the number of correct problem attempts in terms of activities on topics, examples, and problems; time spent on examples; pretest scores; and the group that a student belongs to. Table 3 shows the result of the model with the highest goodness-of-fit ($Adjusted\ R^2 = 0.95$, $F(7, 89) = 253.6$, $p < .001$) that uses the data of all students who used the system ($n = 109$). We found that the number of viewed topics, attempted problems, viewed animated examples, total time spent on annotated examples, and students' groups were reliable predictors for the number of correct attempts on problems, while annotated examples and time spent on animated examples were not.

More specifically, as expected, students with a higher pretest score had more correct attempts on problems; one

Table 3: Summary of the regression model for number of correct problem attempts ($n = 109$)

Predictor	Estimate	Std. error	p-value
Topics covered	3.47	0.24	<.001
Problem att.	0.08	0.02	<.001
Distinct animated ex.	0.48	0.09	<.001
Time on annotated ex.	-7.6e-4	2.7e-4	.006
Pretest	7.28	3.23	.027

unit increase in the normalized pretest score increased the number of correct attempts on problems by 7.28 ($SE = 3.23$). Also, every additional topic that was covered increased the number of correct problem attempts by 3.47 ($SE = 0.24$). Attempting a problem and an animated example increased the number of correct problem attempts by 0.08 ($SE = 0.02$), and 0.48 ($SE = 0.09$), respectively. On the other hand, each second of time that is spent on annotated examples decreased the correct problem attempts by -7.6e-4 (SE =2.7e-4). This data hints that students are likely learning more from animated examples. As a result, the positive impact of examples overcame the negative process of associating examples with poor knowledge, and their joint impact is still positive. In contrast, annotated examples appeared to be less useful for learning, allowing the reverse process to overcome the positive impact on knowledge. These findings reveal a stronger impact of animated examples on problem solving performance; however, more data and more careful analysis should be made to determine a reliable conclusion.

5.3 Amount of Learning

We measured learning by predicting students' post-test scores at the end of the semester. We considered various predictors, including system usage factors (covered topics, total attempts on code execution problems, correct attempts on problems, distinct annotated/animated examples viewed, and time spent on annotated/animated examples), prior knowledge of students as measured by pretests, and the group that indicates the course and semester that study was performed. Pretest and post-test scores were normalized in the 0–1 range. Also, out of 83 students who had taken both the pretest and the post-test, there were 8 students whose post-test score were less than their pretest. We discarded data from those students, as we assumed that no unlearning occurs. Table 4 summarizes the final results of the stepwise regressions for the model, which explains 61% of the variance for the remaining 75 students ($Adjusted\ R^2 = 0.61,\ F(7, 67) = 17.43,\ p < .001$).

As expected, student prior knowledge measured by pretest was among the reliable post-test predictors. One unit increase in the pretest score increased the normalized post-test by 0.615 ($SE = 0.099$) units. Among measures related to student work with practice content, the following four achieved at least a marginal impact on the scores of the post-test: correct attempts on problems, distinct views on annotated examples, and animated examples were all marginally significant (p <.10), and the total time spent on animated examples was significant at the 0.035 level. In particular, similar to the previous analysis, we observed that a switch from annotated to animated examples turns the connection between amount of work and performance from negative to positive: each explored annotated example *decreased* the normalized post-test

Table 4: Summary of the regression model for post-test scores ($n = 75$)

Predictor	Estimate	Std. error	p-value
Correct problem att.	1.4e-3	7.2e-4	.050
Distinct annotated ex.	-0.003	0.002	.077
Distinct animated ex.	0.007	0.004	.088
Time on animated ex.	-8.9e-6	4.1e-6	.035
Pretest	0.615	0.099	<.001

Table 5: Summary of the regression model for course grade ($n = 75$)

Predictor	Estimate	Std. error	p-value
Correct problem att.	8.2e-4	3.1e-4	.010
Distinct annotated ex.	-0.003	0.001	.003
Distinct animated ex.	0.006	0.002	.015
Pretest	0.173	0.050	.001

by 0.003 ($SE = 0.002$) while each explored animated example *increased* the post-test by 0.007 ($SE = 0.004$). While spending too much time with animated examples doesn't add enough extra knowledge to overcome the negative process, the overall connection is still negative: every additional second that students spent on animated examples decreased the normalized post-test by -8.9e-6 (SE=4.1e-6).

5.4 Course Grade

To explore the relationships between the example usage and *course grade*, we used the data of students for whom we had access to their course grades ($n = 82$). Spearman correlation showed marginally significant positive correlation between the number of distinct animated examples that students viewed and their course grade. More views of animated examples was associated with a higher course grade ($\rho = 0.19,\ p = .085$).

Regression analysis was performed to predict the final course grade of students in terms of the same predictors, as mentioned in 5.2. Among 82 students for whom we had access to their grades, 7 had missing pretest and were excluded from the analysis. So, data of 75 students were used in the regression model and Table 5 shows the effect of each predictor on students' grade in the best fitted model ($Adjusted\ R^2 = 0.27,\ F(5, 69) = 6.353,\ p < .001$). As the table shows, the number of correct problem attempts, animated example views, and pretest score positively influenced the course grade. Every additional problem attempt or view of distinct animated examples increased the course grade by 8.2e-4 (SE =3.1e-4) and 0.006 ($SE = 0.002$), respectively. For every unit increase in the normalized pretest score, there was a corresponding predicted increase of 0.173 ($SE = 0.050$) in the course grade. But, the number of views on annotated examples had a negative effect on the course grade: every single view of a distinct annotated example decreased the course grade by 0.003 ($SE = 0.001$). This is another case where a switch to animated examples turns the balance positive.

5.5 Student Feedback

Students from the PITT group were surveyed at the end of the term about the usefulness of the system. One of the questions was about animated examples ("Animated examples helped me to learn Java") and was answered in a

5-point Likert scale, from *Strongly disagree* = 1 to *Strongly agree* = 5. A total of 48 students who used the system and saw animated examples at least once completed the questionnaire. The results show a mildly positive opinion (Mean = 3.44, Median = 4 (Agree), and Mode = 4). Certain differences in the response patterns among low and high pretest groups exist, as well. Students in the low pretest group have a strong tendency to Agree: 58% (14) of them give value 4. By contrast, students in the high pretest group are more heterogeneous: 29% (6) give no opinion (value 3), 29% agreed and 24% (5) strongly agreed (value 5). No significant correlation was found between the answer to this question and the usage of the system (number of animated examples viewed), but a marginal significant negative correlation was found between question responses and the pretest score for high pretest group B= -.345, p=.05, N=21. Very high pretest students found animated examples to be less useful.

6. CONCLUSION AND FUTURE WORK

This paper compared the impact of traditional annotated examples with more advanced animated examples as practice content for learning Java programming. Our results indicated that animated examples better engaged students increasing their interest in completing examples. Animated examples also provided better impact on several performance measures such as problem solving success, post-test scores, and course grade turning the relationship between the amount of work with examples and performance from negative to positive.

The present work confirms previous findings about the usage of animated examples. A recent research by Sirkiä and Sorva [14] studied usage of animated examples as a part of an electronic textbook. Similar to what we observed in this work, students viewed almost all animated examples that they started, and viewing animated examples in the textbook was associated with higher course grades. An interesting difference between these two studies is in the treatment of examples. While in our study examples were offered as additional non-mandatory practice content, in the Sirkiä and Sorva [14] study, animated examples were the core part of the course, and were embedded into the course materials together with text. As expected, core content got more attention than practice content: in that study, students viewed on average 65.3% of embedded animated examples, 63.2% higher than in the case of practice content offered by the Mastery Grids portal.

In the future, we plan to combine the ideas of annotated and animated examples by adding annotation support for animated examples. This means that a teacher can add custom texts to animations in order to emphasize and explain the most important steps in more detail. The current automatically-generated explanations frequently fail to explain what is actually happening. Teachers can create better explanations, just as they can now in WebEx, but instead of explaining the whole line, the explanation can refer to a single animation step, which is only a part of the line. In this way, students are able to follow the animated examples with deeper understanding.

7. ACKNOWLEDGEMENTS

This work is supported by the Advanced Distributed Learning (ADL) Initiative. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of Defense or the Joint Staff. The U.S. Government is authorized to reproduce and distribute reprints for government purposes. Also, the authors want to thank Rebecca Caldwell, Darina Dicheva, and Dmitry Babichenko for their help with data collection.

8. REFERENCES

[1] M. Ben-Ari, R. Bednarik, R. B.-B. Levy, G. Ebel, A. Moreno, N. Myller, and E. Sutinen. A decade of research and development on program animation: The Jeliot experience. *Journal of Visual Languages & Computing*, 22(5):375 – 384, 2011.

[2] T. Boyle, J. Gray, B. Wendl, and M. Davies. Taking the plunge with CLEM: the design and evaluation of a large scale CAL system. *Computers & Education*, 22(1):19–26, 1994.

[3] P. Brusilovsky, S. Edwards, A. Kumar, L. Malmi, L. Benotti, D. Buck, P. Ihantola, R. Prince, T. Sirkiä, S. Sosnovsky, et al. Increasing adoption of smart learning content for computer science education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, pages 31–57, 2014.

[4] P. Brusilovsky and M. Yudelson. From WebEx to NavEx: Interactive Access to Annotated Program Examples. *Proc. of the IEEE*, 96(6):990–999, 2008.

[5] S. M. Čisar, R. Pinter, D. Radosav, and P. Čisar. Effectiveness of program visualization in learning Java: a case study with Jeliot 3. *International Journal of Computers, Communications & Control*, 6(4), 2011.

[6] A. Davidovic, J. Warren, and E. Trichina. Learning benefits of structural example-based adaptive tutoring systems. *IEEE Transactions on Education*, 46(2):241–251, 2003.

[7] B. du Boulay. Some difficulties of learning to program. *Studying the Novice Programmer*, page 283, 2013.

[8] P. J. Guo. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 579–584. ACM, 2013.

[9] A. Kelley and I. Pohl. *C by dissection : The essentials of C programming*. Addison-Wesley, New York, 1995.

[10] M. Linn. Can experts' explanations help students develop program design skills. *International Journal on the Man-Machine Studies*, 36:511–551, 1992.

[11] T. Loboda, J. Guerra, R. Hosseini, and P. Brusilovsky. Mastery Grids: An open source social educational progress visualization. In *9th European Conference on Technology Enhanced Learning (EC-TEL 2014)*, pages 235–248.

[12] T. Naps, G. Rossling, J. Anderson, S. Cooper, W. Dann, R. Fleischer, B. Koldehofe, A. Korhonen, M. Kuittinen, C. Leska, M. McNally, L. Malmi, J. Rantakokko, and R. J. Ross. Evaluating the educational impact of visualization. *ACM SIGCSE bulletin*, 35(4):124–136, 2003.

[13] T. Sirkiä. A JavaScript library for visualizing program execution. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, Koli Calling '13, pages 189–190. ACM, 2013.

[14] T. Sirkiä and J. Sorva. How do students use program visualizations within an interactive ebook? In *Proceedings of the eleventh annual International Conference on International Computing Education Research*, pages 179–188. ACM, 2015.

[15] J. Sorva, V. Karavirta, and L. Malmi. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education*, 13(4):1–64, 2013.

[16] G. Weber. Individual selection of examples in an intelligent learning environment. *Journal of Artificial Intelligence in Education*, 7(1):3–31, 1996.