During automatic program animation, explanations after animations have greater impact than before animations

Peng Wang
University of Eastern Finland
School of Computing
Joensuu
pwang@student.uef.fi

Roman Bednarik
University of Eastern Finland
School of Computing
Joensuu
roman.bednarik@uef.fi

Andrés Moreno
University of Eastern Finland
School of Computing
Joensuu
andres.moreno@uef.fi

ABSTRACT

Little is known about the effectiveness of automatic explanations in educational program visualization. We designed a study in which the order of animations and related explanations was manipulated. Two groups of a total of 18 participants interacted with either animation-first or explanation-first version of a tool. The results indicate that animation-first approach is significantly more effective. On the grounds of these findings and students' input about the explanation generation and layout, we discuss the design implications of the findings.

Categories and Subject Descriptors

K.3.2 [**Computers and education**]: Computer and Information Science Education—*computer science education, information systems education*

General Terms

Human factors, Experimentation, Design

Keywords

program animation, learning programming, educational technologies, Jeliot 3

1. INTRODUCTION

Program animation, when interaction with it is properly designed, has been shown to be beneficial for learning programming [10]. Others have stressed specifically adequate teacher support [3] as one of the key ingredients for successful learning.

It has been previously reported that inexpert users of visualizations take longer to understand and make efficient use the visualizations than experts, partly due to the visualizations being designed by the experts themselves and partly because experts already possess mental models that help in understanding [22]. In the domain of computer programming education, it is well known students "cannot make sense of visualisations" [4]. Then, a question arises, what methods, pedagogically and empirically sound, should be used when the teachers are not present and cannot cue students in to engage in meaningful interactions with a program visualization tool? Naps et al. [20] suggested to "complement visualizations with explanations", based on research showing that animations are better understood if they are accompanied with concurrently narrated explanations [15]. Naps et al. suggested that also in programming education explanations could be added to visualizations in two ways: 1) using accompanying text or 2) providing coordinated audio explanations.

Auditory explanations fit the dual-coding theory better. They complement simultaneously the visual stimulation to create new knowledge, and its effectiveness has been proved. In their experiment, Mayer and Anderson [15] demonstrated how concurrent verbal explanations improved students' problem solving transfer skills. Students who received the explanations before the animation did significantly worse than those with concurrent audio explanations. These results have been extended in [17], providing arguments for employing multimedia-learning theory principles for learning with dynamic visualizations.

Several automatic animation systems, that is those systems in which the animation is dynamically created from users' own data set or program source code, offer benefit to students from explanations as they help to build the relationships between the animations and the concept explained. However, program animation systems' designers have so far preferred to employ textual explanations rather than verbal explanations, and these explanations are often displayed simultaneously.

While it is assumed that students make use of the explanations when interacting with the program animation systems, there are neither studies nor guidelines regarding the temporal arrangement of textual explanations and animations. In this paper thus, we explore the effects of the arrangements on students' learning of principal Java programming concepts.

Koli Calling '12, November 15–18, Tahko, Finland

2. RELATED SYSTEMS AND RESEARCH

2.1 Program visualization and explanations

Many program visualization tools such as MatrixPro [11], ALVIS LIVE! [9], Jeliot 3 [2] and WinHIPE [21] do not have explanations of the animations. MatrixPro and WinHIPE provide exercises with textual descriptions and explanations about the program or algorithm.

On the other hand, ViLLE [23], WADEIn II [6], and VARScope [12] provide explanations of animations or programs, and these explanations are shown during animations. We next present a short summary of these systems.

ViLLE [23] and UUhistle [25] are program visualization tools that animate, and let the student simulate in the case of UUhistle, the execution of a program. They highlight code lines, displays the states of variables, and creates frames representing newly executed methods. At the same time, in both tools, explanations are automatically generated in a separate frame at the bottom. A study of Rajala et al. [24] on effectiveness of ViLLE was carried out, demonstrating that ViLLE is especially useful for inexperienced programmers.

WADEIn II [6] is a web-based program visualization application. It visualizes the process of expression evaluation in C language and it supports twenty-four C operators. WADEIn II displays animations and related explanations close to each other in the "blackboard" region, and they are presented simultaneously. As students' knowledge increases, the system evaluates it, parts of explanations are hidden until no more explanations are presented, and animations become faster.

VARScope [12] is a program visualization system focusing on the concept and usage of variable scope in C programming language. Visualization in VARScope includes highlighted code line, value of the variable, animating the active and hidden variables, and the detailed explanations of each code line. Explanations and visualizations are displayed simultaneously in separate windows.

In summary, to our knowledge ViLLE and UUhistle are the only general purpose visualization tools that contains automatic explanations during visualization. WADEIn II and VARScope are more focused on certain programming concepts they explain, but have interesting features like adaptation. Automatic explanations in these tools are mostly presented simultaneously and in different windows than the main representations.

2.2 Temporal arrangement of explanations

Few previous studies investigated the arrangement of animations and explanations in time. In order to evaluate the effects of verbal and visual representation in time, Mayer [14] applied a number of retention and transfer tests. The result was that students who received simultaneous animation and narration outperformed those who received successive animation and narration on problem-solving test. In retention test there was no statistical difference between simultaneous presentation and successive presentation. In Mayer's studies, however, there was little information on the presentation of *textual* narrations and animations. What has been found is the advantage of multimodal representation use, that is of the combination of verbal and visual materials.

Lawrence [13] carried out an experiment regarding the order of presentation of text and animation in algorithm visualization. The conclusion of Lawrence's research was that students in text-first condition did not achieve better result than those in animation-first condition. Although no significant difference was observed, text-first approach was selected finally for the reason that the text-first group achieved a slightly higher score than the other group. Lawrence thought that condition of text first rather than animation first was preferred by participants. In Lawrence's study, XTango [26] was used to animate relevant algorithms and twelve students were separated equally into two groups. An analysis of each group's post-test score determined if the order of presentation had effects on result.

Lawrence's research is quite similar to ours in a few aspects. We too put an emphasis on the impact of the arrangement of explanations and animations in time and it is also our goal to improve understanding of certain behaviors of the visualization and thus of certain concepts being visualized. However, Lawrence's experiment only compared each group's post-test score, while we here present a pre- post-test design.

3. JELIOT 3

We selected Jeliot 3 as a system to test the effectiveness of explanations and their temporal arrangement for few reasons. First, Jeliot 3 is distributed as an open source, it is well documented[1] and its architecture allows for such modifications [2]. Second, as we show below, it has been repeatedly shown to be effective in learning programming. Here, it has been modified to automatically display explanations for certain concepts during the animation of students' programs.

3.1 Previous research on Jeliot effectiveness

Jeliot 3 employs automatically generated animations that display the execution of a Java program. Teachers and students can use these animations in a movie like fashion or in a step by step way. Several studies have demonstrated that Jeliot 3 has positive impacts on learning programming [3, 7, 8]. A study of [3] was carried out to evaluate a predecessor of Jeliot 3 in a one-year programming course. In that experiment, students were divided into a control group and animation group. Between the two groups only the animation group was treated with Jeliot. Ben-Bassat et al. found that there was no statistically significant difference between pre- and post-test results in the control group, whereas there was statistically significant improvement in the grades of the animation group. Furthermore, in the animation group it was demonstrated that mediocre students benefited more from long-term use of the tool than either strong or weak students.

A study of Cisar et al. [7] verifies that Jeliot 3 affects learning of Java. In that study, results of 20 multiple choice questions by 400 students were analyzed. It was shown that students who learned with the help of Jeliot 3 outperformed those who did not use Jeliot 3. Hongwarittorrn and Krairit [8]

[1] `http://cs.uef.fi/jeliot/`

confirms that Jeliot 3 leads to better learning of Java, especially in object-oriented programming (OOP). In that study conducted with 54 participants, those who learned Java with Jeliot achieved better results than those who learned without the tool.

However, other research [16] indicates that some students misunderstand the animations in Jeliot 3. In that study, after 10 weeks voluntarily using Jeliot 3 as a programming tool for weekly assignment completion, six maths undergraduate students were interviewed to explore their attitudes towards the tool and to assess their comprehension of animation. Although almost all subjects understood animation referring to basic statements such as variable declaration, some of them failed to describe the animation of an object allocation correctly. The "this" reference which is used to point to the current object, and argument passing to parameter of the constructor, were found to be the most puzzling.

3.2 Objectives and hypotheses

In this paper, the aim is to inspect the effect of the sequence of animation and related explanation on learning outcome during programming. In particular, the investigation we present compares the impact in understanding critical Java programming concepts when explanations are either displayed *after* animations or *before* animations.

The null hypothesis we investigate is that there is no difference in the other of animation and related explanation in terms of learning outcome.

4. METHOD

We designed a pre—post-test study in which participants were assigned to one of two conditions: either they are interacting with a modified Jeliot 3 system that presented explanations of key concept before the animation of a concept, or they were using a version of the tool that presented explanations after the concept animation.

Lawrence's research method is similar to the one is used in this study. She also focused on the impact of the temporal arrangement of explanations and animations in time. However, Lawrence's experiment only compared each group's post-test score, while this study uses a pre- post-test design.

4.1 Design and materials

The experiment was designed as a between subject study, where the order of the animation and explanation was the primary factor with two levels: one level was explanation first while the other level was animation first.

Both groups had the same short Java program for experiment (see Appendix for listing) and the same test before and after the experiment. The only difference between two groups was the order of explanations and animations. In the animation-first group, the corresponding explanation was presented after each animation and it described what the previous animation represented. In contrast, in the explanation-first group, related explanation was displayed before each animation and it described what the next animation would represent. The content of the explanations was same for both groups.

Table 1: Background of the participants. There was no statistically significant difference between two groups in OOP grades ($p = 0.127$) and in self-ratings ($p = 0.227$) according to a t-test. Note: M = male, F = female.

Group	N	OOP grade	Self-rating	M	F
Anim.-first	10	3.44 (0.88)	3.30 (0.95)	9	1
Explan.-first	8	2.75 (0.89)	2.75 (0.89)	6	2

There were altogether three target animations related to three fundamental Java object-oriented concepts:

1. Object initialization and "this" keyword
2. Reference return and assignment
3. Garbage collection

These concepts were chosen as the animations of object-oriented concepts in Jeliot 3 were identified to be the most difficult for students to explain after watching them [16]. As well, the first two concepts have been considered either critical or difficult to learn on a study surveying faculty members [5].

As an example of an animation in Jeliot 3 Figure 1 and Figure 2 show the sequence of animation steps for object initialization and "this" keyword concepts in animation first and explanation first conditions, respectively.

4.2 Participants

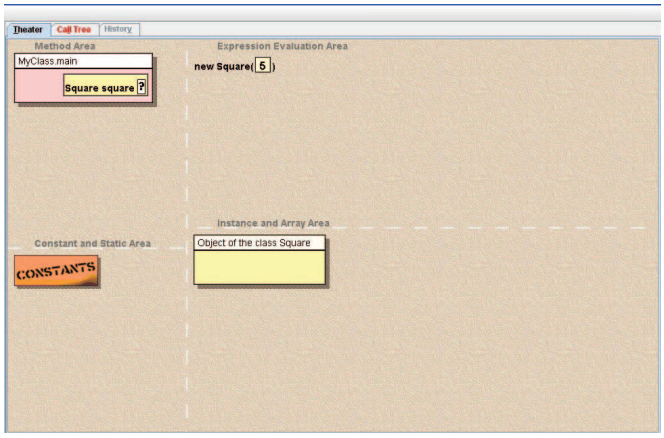
There were a total of 18 volunteering participants in this experiment, 15 male and 3 female. The participants were computing postgraduate and Master's students at one Finnish university. In overall, they had very little or no experience with Jeliot 3. All participants had some knowledge of object oriented programming (OOP) in Java as they recently took a Java class in an undergraduate course. A grade from the OOP was collected as a background measure of OOP understanding along with self-rating of OOP skills, both on the scale from 1 (worst) - 5 (best).

They were divided into two groups: the animation-first group (10 participants) and the explanation-first group (8 participants). Table 1 shows no significant differences between the groups in terms of previous grade in OOP class and self-rating.

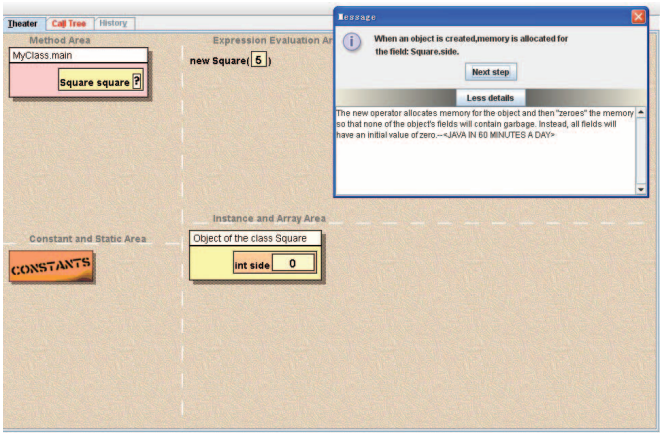
4.3 Procedure

Participants were given a short introduction to Jeliot 3 by an assistant. The introduction included what each area of the animation frame displays and how to control the process of animation through buttons. After the introduction, participants were required to get familiar with Jeliot 3 by running an object-oriented program. Participants were allowed to ask questions on Jeliot 3. The time reserved for this introduction and practice was 10 minutes.

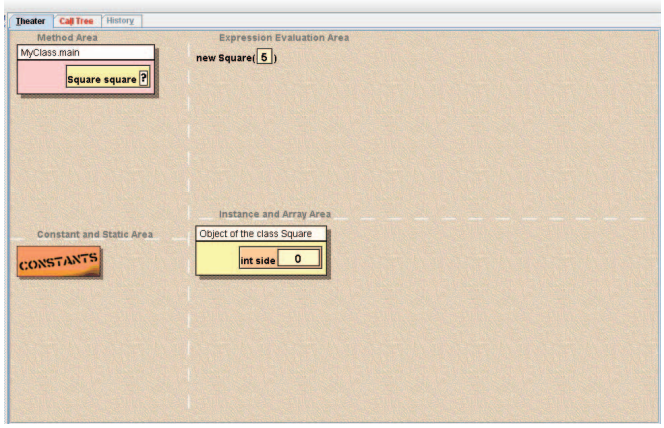
Afterwards, participants completed a test which comprised three questions in 20 minutes. Each question could award



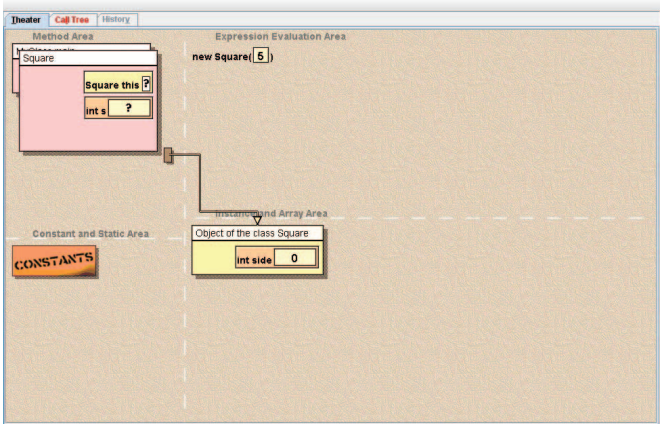
(a) Animation of object initialization starts.



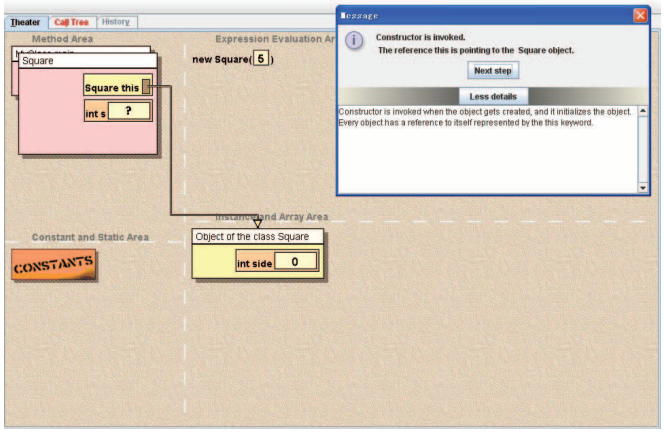
(b) Animation of initialization ends and related explanation appears.



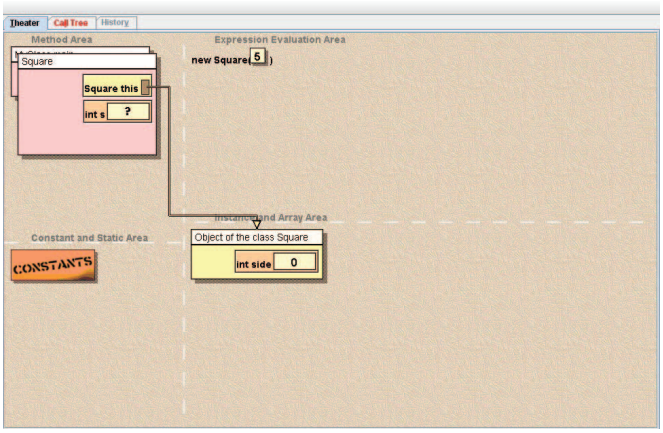
(c) Related explanation disappears.



(d) Animation of "this" keyword starts.

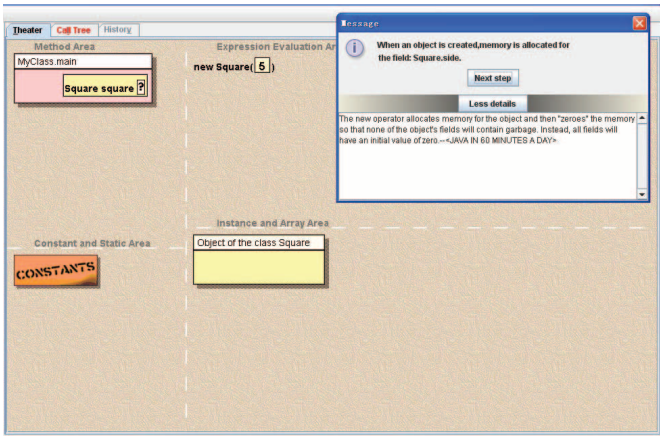


(e) Animation of "this" ends and related explanation appears.

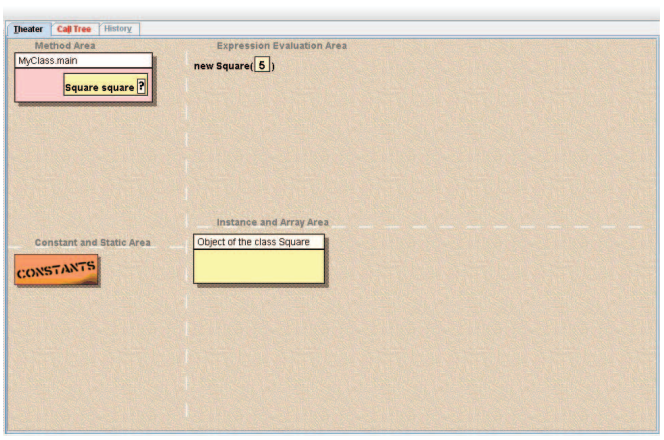


(f) Related explanation disappears.

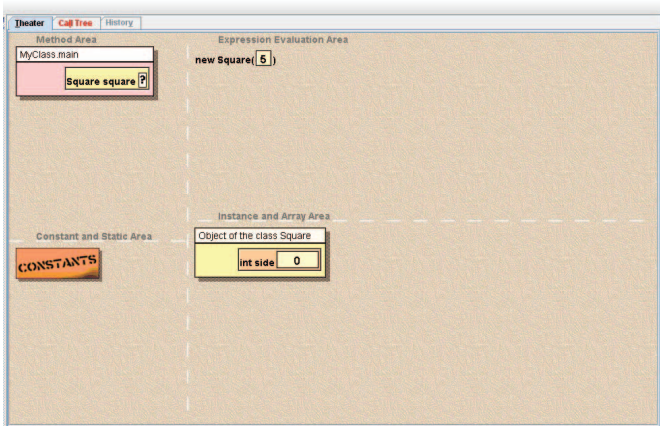
Figure 1: In animation first condition, animations of object initialization and "this" keyword are shown before the respective explanation appears.



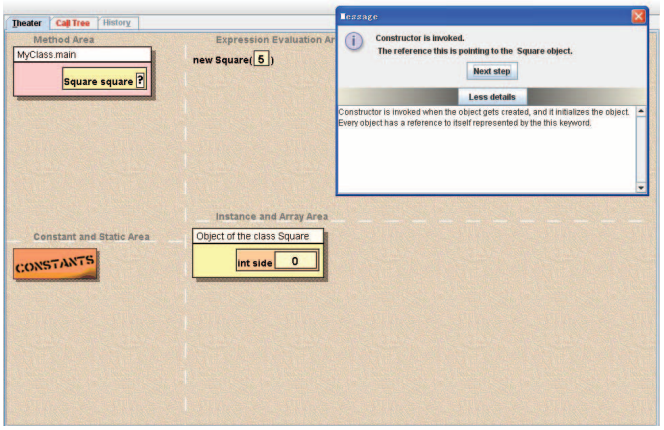
(a) Explanation of object initialization appears.



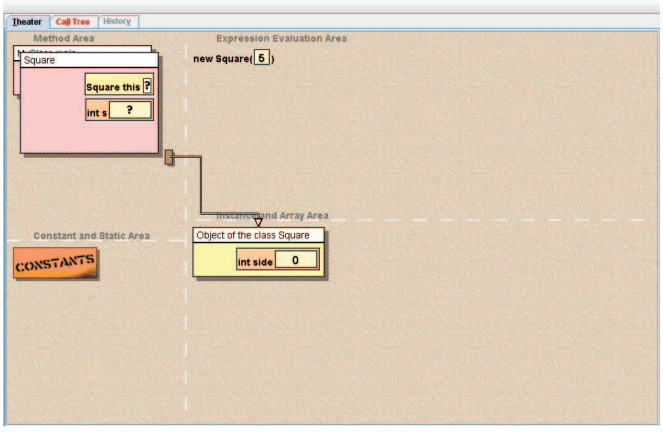
(b) Explanation disappears and related animation starts.



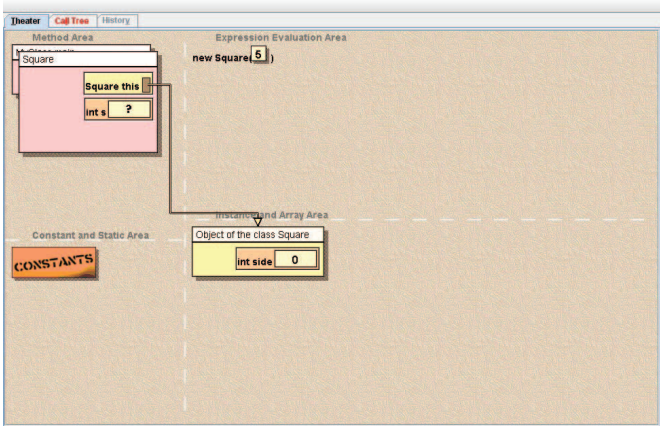
(c) Related animation ends.



(d) Explanation of ”this” keyword appears.



(e) Explanation disappears and related animation starts.



(f) Related animation ends.

Figure 2: In explanation first condition, animations of object initialization and ”this” keyword are shown only after the respective explanation.

the student a maximum score of 5, for a maximum total of 15 points. During the test, participants could use Jeliot 3, with the options for explanations deactivated, to visualize the animation associated with each question.

After the test, explanations were added to Jeliot 3. Participants were required to run the same program again and read explanations in 15 minutes.

In the end, participants completed a test in 15 minutes. During this test, they were not allowed to use Jeliot 3. Those three questions in this test were same as the previous ones.

4.4 Analysis

In this paper we present an analysis of scores achieved on pre-test ($score_1$) and post-test ($score_2$) evaluations. We compute the raw score difference as well as learning gain, which depends on the maximum number of points that can be awarded (max). We employ the following formula to compute the learning gain:

$$Learning\ gain = \frac{score_2 - score_1}{max - score_1}$$

Learning gain allows to evaluate the relative increase in score given the pre-test score (how much the student improved out of total possible improvement), while raw score difference does not consider the starting level of the assessment.

5. RESULTS

In this paper we analyze performance in terms of pre-post test differences. Table 2 shows the distribution of pre-test scores. A statistical analysis[2] shows that there were no significant differences between the two groups on the pre-test performance although the animation first group performed somewhat better on the reference return and assignment concept. The same question though was also easier for participants as it received the highest score from the three concepts.

The scores on the post-test are shown in Table 3. It shows that on first and last concept the animation-first group overperformed the explanation-first group. We treat the differences between post and pre-test in the following section.

We also computed correlations of the pre-test and post-test scores with the knowledge of the participants measured by the grade obtained from a previous OOP course, see Table 4. Such analysis allows to answer a question, whether explanations have homogenous effect on participants regarding their background knowledge.

It turned out that only second question related to reference return and assignment and pre-test scores were significantly correlated. This fact indicates that those with better OOP knowledge did better in answering that question related to garbage collection before the use of explanations. The score on the post-test was not correlated with previous knowledge in any of the questions. This is due to the fact, the intervention improved score on the second question of only few participants with lower grades and was not effective on the majority of the users.

[2]A 1-Sample Kolmogorov-Smirnov test verified that the distributions of participants' grades both in pre-test and post-test were normal distributions as well as the distributions of the changes in the score. Hence we applied a series of independent-samples t-tests in the following analyses.

Table 4: Correlations of pre- and post-test scores with OOP understanding for all study participants, a 2-tailed p-value in parenthesis; * denotes p<.02.

	Q 1	Q 2	Q 3
pre-test (N=18)	0.25 (0.33)	0.58 (0.02*)	0.20 (0.46)
post-test (N=18)	0.30 (0.24)	0.28 (0.28)	0.23 (0.37)

5.1 Learning score changes and learning gain

We first computed for each participant the difference between pre- and post-test scores, shown as group-aggregated mean values in Table 5. In total, there were significant differences in the raw score changes between the two groups.

In particular, the animation-first group improved by 1.7 points on average in total, while the explanation-first group improved only by 0.25 points on average in total. The original standard deviation of the animation-first group was 2.0 thus the learning improvement measured in standard deviation shift by 1.7 corresponds to about 0.85 σ. When analyzing the score change within the groups statistically, using a paired-sample t-test we discovered a significant difference between pre- and post-test scores in the animation-first group ($t(9) = 3.6$, $p = .006$), while there was no difference in the performance of the explanation-first group ($t(7) = 1.528$, $p = .170$).

Analysis of learning gain scores discovered that the animation-first group improved by 15% on average while the explanation-first group improved by 2% on average, see Table 6. The overall difference between the groups was significant on the 3% level. There was no difference in the learning gains on the second concept related to reference return and assignment.

5.2 Analysis of written answers

5.2.1 Object initialization and this-keyword

In the animation-first group, three of the ten (30%) participants corrected their answers on object initialization, while another three of the ten (30%) participants corrected their answers on "this" keyword. However, in the explanation-first group, no participants improved their scores after using the explanation-version of the tool. Table 7 captures the differences between pre- and post-test answers.

5.2.2 Reference return and assignment

As shown above, there was no difference in the understanding of reference return and this-keyword concepts between groups and that there was very little or no improvement after using explanations. Only two and two participants in each group scored better by one point; Table 8 shows the rare changes in answers.

Table 2: *Before:* Means, standard deviations (in parenthesis), t value, and 2-tailed p-value of each question, before using explanations (pre-test)

	Q 1	Q 2	Q 3	Total questions
Animation-first (N=10)	0.90 (0.74)	1.90 (0.74)	0.90 (0.74)	3.70 (2.00)
Explanation-first (N=8)	0.88 (0.35)	1.38 (0.74)	0.88 (0.83)	3.13 (1.13)
t value	0.088	1.495	0.067	0.723
p (2-tailed)	0.931	0.154	0.947	0.480

Table 3: *After:* Means, standard deviations (in parenthesis), t value, and 2-tailed p value of each question, after using explanations (post-test)

	Q 1	Q 2	Q 3	Total questions
Animation-first (N=10)	1.60 (1.17)	2.10 (0.74)	1.70 (0.95)	5.40 (2.59)
Explanation-first (N=8)	0.88 (0.35)	1.63 (0.74)	0.88 (0.83)	3.38 (1.92)
t value	1.851	1.352	1.931	2.009
p value (2-tailed)	0.091	0.195	0.071	0.062

Table 5: Pre-post-test mean differences in raw score, standard deviations (in parenthesis), t value, and 2-tailed p value of each question.

	Q 1	Q 2	Q 3	Total questions
Animation-first (N=10)	0.70 (0.82)	0.20 (0.42)	0.80 (0.79)	1.70 (1.49)
Explanation-first (N=8)	0.00 (0.00)	0.25 (0.46)	0.00 (0.00)	0.25 (0.46)
t value	2.689	-0.239	3.207	2.899
p value (2-tailed)	0.025	0.814	0.011	0.014

Table 6: Mean learning gains, standard deviations (in parenthesis), t value, and 2-tailed p value

	Q 1	Q 2	Q 3	Mean gain
Animation-first (N=10)	0.18 (0.23)	0.06 (0.13)	0.19 (0.20)	0.15
Explanation-first (N=8)	0.00 (0.00)	0.07 (0.12)	0.00 (0.00)	0.02
t value	2.250	-0.139	2.732	2.413
p value (2-tailed)	0.039	0.891	0.015	0.028

Table 7: Example answers from pre- and post-test on object initialization and this-keyword, all participants from animation-first group.

Participant	In pre-test	In post-test	Changes
A	"...create a new object square."	"...this arrow to square."	This participant misunderstood the meaning of arrow in pre-test, but after reading explanations he realized arrow as a reference to an object.
B	"The arrow represents the relationship between the current object and its variables."	"Every object has a reference to itself and this keyword indicates that. The arrow means the reference to the object."	This participant thought of arrow as relationship. However, later he was able to understand why used "this" so that explanations made sense for him.
C	"The arrow refers to the constructor of the class."	"The arrow means the memory is located for new object."	This participant changed his answers from reference to constructor to reference to object.

Table 8: Example answers from pre- and post-test on reference return and assignment.

Participant	In pre-test	In post-test	Changes
A participant in animation-first group	"the movement of square from 'Expression evaluation area' to 'Method area' means to select the value of the variable "side" and put it in the object of class square."	"The movement of the small rectangle from 'evaluation area' to 'method area' means to assign to the variable a reference from the new created object."	This participant was not able to express the meaning of movement precisely in pre-test, but after reading explanations he could consider the movement as assignment correctly.
A participant in explanation-first group	"The newly created instance square has been initialized by the instance of class square. It will return to the main function with the reference to the new instance square."	"...assign it to the newly created instance."	This participant changed his answer from initialization and return to assignment.

5.2.3 Garbage collection

After interacting with explanations, six of the ten participants in the animation-first group improved their scores. However, no participants in the explanation-first group improved their scores. Table 9 shows examples of three participants' answers in pre-test and post-test.

5.3 Summary of the findings

The main findings can be summarised as following:

- *Sequencing of animations and explanations matters* Explanations after animations have positive effect on learning gain while explanations before animations have little to no effect.
- *Students quickly assimilate the vocabulary of the explanations* Students improved their descriptions, and scores, borrowing the text provided in the explanations.

6. GENERAL DISCUSSION AND CONCLUSIONS

In this paper we presented an empirical evaluation of the effect of temporal arrangement of explanations in learning three Java concepts using automatic program animation. We extended a well studied tool by automatic explanation generation and conducted a study in which participants used either a version that presented explanations before the animated concept or after it.

The results show that there are differences in learning contingent with the temporal arrangement of animations and explanations. Interestingly, even short-term interaction with explanations after animations is sufficient to improve understanding of some core Java programming concepts.

Previous research indicates that interactive elements such as prediction questions [18] shown during program animation cause student to pause and are beneficial for learning as they increase the levels of engagement [19]. The present findings can be seen as in line with the previous research. The design of animation-first condition creates pauses during the display of the explanation in which students seem to reflect on what happened during the animation. On the contrary, when the explanation goes first, the dynamic nature of the animation does not let the student to mentally retrieve the text from the explanation she just read. We plan to investigate the actual step-to-step use of explanations using gaze-tracking methodology that will allow us to estimate when and how explanations and visualizations are attended.

Should textual explanations be displayed at the same time with animation? Intuitively and theoretically, such design should be mostly effective [15, 17, 14]. There are however at least two counter-arguments against concurrent explanations in programming education. First, the new explanations presented here are not verbal, so the temporal-contiguity effect would not entirely apply. In previous research on multimodal learning, the additional modality was often verbal. We see this approach as impractical for classroom use and for eventual automatic implementation, though we do not dismiss this possibility.

Second, and more importantly we believe that a juxtaposed explanation of a concurrently animated programming concept may bring more harm than gain. It has been previously shown that novice programmers are not able to coordinate multiple representations concurrently [1]. Same applies here, where introducing a new attention demanding element into an already complex and dynamic visual stimuli would result in further increase of load. Monitoring the ongoing animation, source code, output, and an additional explanation would simply be beyond the possibilities of a student.

The research presented here opens new paths into the topic of interaction with explanations. While the explanations were generated automatically using generic templates for various concepts, further work could ask the student to write the explanation of each concept herself, to be later compared with the experts' explanation.

Table 9: Example answers from pre- and post-test on garbage collection, all participants from the animation-first group.

Participant	In pre-test	In post-test	Changes
A	"System run out of main(); So the Square() is removed."	"...the garbage collect the object, therefore, Square square is removed."	After reading explanations, this participant was able to recalled garbage collection.
B	"It means that the program has finished and a new object has been created with the value(s) given. The new object is shown in the instance and array with the variables visible."	"...memory will be freed by the garbage collector..."	Although this participant used garbage collector not garbage collection in post-test, it was found that his answer changed a lot from pre-test to post-test.
C	"That means it went out of the scope and does not exist anymore."	"...will be removed once garbage collector will start its work."	This participant also recalled garbage collection after reading explanations so that her answer changed from pre-test to post-test.

Further research needs to consider the effects of explanations from a long-term perspective. In our study students were exposed to the intervention for a short period of time, and, even though the observed effects were clearly visible, a course-long exposure is needed to establish an evidence of more permanent effects.

Acknowledgments

The work of Roman Bednarik was supported by a grant of Academy of Finland #137773.

7. REFERENCES

[1] R. Bednarik. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *Int. J. Hum.-Comput. Stud.*, 70(2):143–155, Feb. 2012.

[2] M. Ben-Ari, R. Bednarik, R. Ben-Bassat 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.

[3] R. Ben-Bassat Levy, M. Ben-Ari, and P. Uronen. The jeliot 2000 program animation system. *Computers & Education*, 40(1):1–15, 2003.

[4] P. Brusilovsky. Explanatory visualization in an educational programming environment: Connecting examples with general knowledge. In B. Blumenthal, J. Gornostaev, and C. Unger, editors, *Human-Computer Interaction*, volume 876 of *Lecture Notes in Computer Science*, pages 202–212. Springer Berlin / Heidelberg, 1994.

[5] P. Brusilovsky, J. Grady, M. Spring, and C.-H. Lee. What should be visualized?: faculty perception of priority topics for program visualization. *SIGCSE Bull.*, 38(2):44–48, June 2006.

[6] P. Brusilovsky and T. Loboda. Wadein ii: a case for adaptive explanatory visualization. In *ACM SIGCSE Bulletin*, volume 38, pages 48–52. ACM, 2006.

[7] S. Cisar, D. Radosav, R. Pinter, P. Cisar, D. Radosav, and P. Cisar. Effectiveness of program visualization in learning java: a case study with jeliot 3. *International Journal of Computers Communications & Control*, 6(4):669–682, 2011.

[8] N. Hongwarittorrn and D. Krairit. Effects of program visualization (jeliot3) on students' performance and attitudes towards java programming. In *The spring 8th International conference on Computing, Communication and Control Technologies*, 2010.

[9] C. Hundhausen and J. Brown. What you see is what you code: A radically dynamic algorithm visualization development model for novice learners. In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 163–170. IEEE, 2005.

[10] C. Hundhausen, S. Douglas, and J. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, 2002.

[11] V. Karavirta, A. Korhonen, L. Malmi, and K. Stålnacke. Matrixpro-a tool for on-the-fly demonstration of data structures and algorithms. In *Proceedings of the Third Program Visualization Workshop*, pages 26–33, 2004.

[12] G. Krishnamoorthy and P. Brusilovsky. Personalized guidance for example selection in an explanatory visualization system. In *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, volume 2006, pages 2122–2127, 2006.

[13] A. Lawrence. Empirical studies of the value of algorithm animation in algorithm understanding. Technical report, DTIC Document, 1993.

[14] R. Mayer. *Multimedia learning*. Elsevier, 2002.

[15] R. Mayer and R. Anderson. Animations need narrations: An experimental test of a dual-coding hypothesis. *Journal of educational psychology*, 83(4):484, 1991.

[16] A. Moreno and M. Joy. Jeliot 3 in a demanding educational setting. *Electronic Notes in Theoretical Computer Science*, 178:51–59, 2007.

[17] R. Moreno and R. Mayer. Cognitive principles of

multimedia learning: The role of modality and contiguity. *Journal of educational psychology*, 91(2):358, 1999.

[18] N. Myller. Automatic generation of prediction questions during program visualization. *Electronic Notes in Theoretical Computer Science*, 178:43–49, 2007.

[19] N. Myller, R. Bednarik, E. Sutinen, and M. Ben-Ari. Extending the engagement taxonomy: Software visualization and collaborative learning. *Trans. Comput. Educ.*, 9(1):7:1–7:27, Mar. 2009.

[20] T. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, et al. Exploring the role of visualization and engagement in computer science education. In *ACM SIGCSE Bulletin*, volume 35, pages 131–152. ACM, 2002.

[21] C. Pareja-Flores, J. Urquiza-Fuentes, and J. Velázquez-Iturbide. Winhipe: an ide for functional programming based on rewriting and visualization. *ACM SIGPLAN Notices*, 42(3):14–23, 2007.

[22] M. Petre. Why looking isn't always seeing: readership skills and graphical programming. *Commun. ACM*, 38(6):33–44, June 1995.

[23] T. Rajala, M. Laakso, E. Kaila, and T. Salakoski. Ville–a language-independent program visualization tool. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland, November 15-18, 2007. Conferences in Research and Practice in Information Technology*, volume 88, 2007.

[24] T. Rajala, M. Laakso, E. Kaila, and T. Salakoski. Effectiveness of program visualization: A case study with the ville tool. *Journal of Information Technology Education*, 7:15–32, 2008.

[25] J. Sorva and T. Sirkiä. UUhistle – a Software Tool for Visual Program Simulation. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, pages 49–54. ACM, 2010.

[26] J. Stasko. Animating algorithms with xtango. *ACM SIGACT News*, 23(2):67–71, 1992.

Appendix

Listing 1: The Java program employed in the experiment

```
public class Square{
        int side;
        Square(){ side = 0;}

        Square(int s){ side = s;}
}
public class MyClass {

        public static void main() {
                Square square = new Square(5);
        }
}
```