

Roles of Animation Tools in Understanding Programming Concepts

ANDRES MORENO

University of Eastern Finland
amoreno@student.uef.fi

MIKE JOY

University of Warwick, UK
mike.joy@warwick.ac.uk

ERKKI SUTINEN

University of Eastern Finland
erkki.sutinen@uef.fi

Computer generated animations are resources used to explain how programs are executed in order to clarify the relevant programming concepts. However, whilst trying to understand new programming concepts it is not clear how and when students benefit from an animation if they are using the tool on their own. To clarify the role of an animation tool in the student learning process, six students from an introductory programming course at a Tanzanian university attended three individual sessions. In each session students used a program animation tool to understand a new programming concept, and they were asked to orally describe the animations as they watched them. The authors conducted a qualitative analysis on the video and audio recordings taken when the students described the animations. Through an inductive category generation process, five roles played by the animation tool were identified: no-role or empty, exploratory, confusing, teaching, and evaluating. This classification of roles helps us to understand the potential of animation tools, and suggests that versatile use of an animation tool can enrich the teaching and learning process. Animation tools could diagnose a student's current learning stage and adapt a different role to engaging the student in the learning activity.

Keywords: Educational technologies, qualitative, animation, multimedia, cognitive tools, programming, computer science, Jeliot 3

Programming is often perceived as difficult to understand. Several multi-institutional studies at international universities have revealed students' low level of programming skills after their first year at a computer science program (McCracken *et al.*, 2001, Whalley *et al.*, 2006). Programming requires skills that students might not have yet developed, and Mayer (1981) claims that concrete models are beneficial for novices to acquire new programming knowledge.

Programming animation tools are cognitive tools (Kim and Reeves, 2007) designed to scaffold the learning of programming. They have been developed in an attempt to provide concrete models to students, who might otherwise have problems with the abstract nature of programming. In such tools, each programming concept or statement has a corresponding graphical representation, which is consistent with, and descriptive of, the steps taken by a program as it is executed.

Program animation is a resource that teachers can use to explain new programming concepts to students. After the teacher has presented the basics of a programming concept, they will demonstrate and explain the steps that the programming animation tool animates when executing the introduced concept. Students can later use the tool to review the concept or solve programming assignments related to it. Two factors drive the use of animations according to Byrne *et al.* (1999). First, it is expected that students will benefit from the animation tool, as to a textbook, because the dynamic nature of programming is not well represented either by the source code or by written explanations, and secondly because of the concrete and visual model the animation provides.

Although several studies claim that programming animation tools have a beneficial impact in learning (Ben-Bassat Levy *et al.*, 2003), Hundhausen *et al.* (2002) note that the effectiveness of the tools is not yet clearly established, with studies identifying only minor effects of the tools. For this and other practical reasons, programming animation tools have as yet had little acceptance from teachers (Ben-Bassat Levy and Ben-Ari, 2007). Solutions have been proposed to overcome the practical problems found by teachers (Naps *et al.*, 2003), but it is not clear why animations are not always beneficial to the students. One possible reason is that the concrete model provided by the animation is not enough to advance the *zone of proximal development* (Vigostky, 1978) of the student, that is, the student fails to understand a new concept using the tool.

Studies considered by Hundhausen *et al.* (2002) in their meta-analysis reflect the trend in evaluating animation tools in Computer Science education. They mostly measured students' performance before and after the in-

tervention with the animation tool, and the effectiveness of animations was assessed and conclusions were drawn based on quantitative data. However performance-based studies do not reveal students' relationships with the cognitive tool (Kim and Reeves, 2007), nor patterns using it.

This relationship can be uncovered by studying the patterns of tool usage, which Iiyoshi *et al.* (2005) define as “the manner in which tool use varies according to learner differences, task complexity and tool flexibility”. As a student's knowledge varies, so does the use of the animation tool by the student, even over short periods of time. For example, after a student has completely understood a concept with the animation, the tool will not have the same role. After understanding and identifying these patterns, or *roles*, we can start rethinking the animation tools so they fulfil their potential.

The four main roles identified in this study — exploratory, confusing, teaching and evaluating — suggest a linear progress of students' understanding, as depicted in Fig. 1. In other words, using the animation tool, the student should go through the stages of exploration, confusion, teaching/learning, and evaluation of their knowledge.

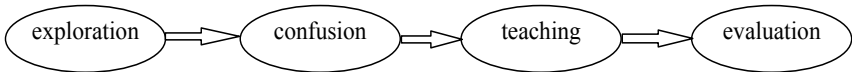


Figure 1. Expected linear progress of students' understanding of a programming concept using an animation tool

In this paper, we report a qualitative analysis of students' verbal descriptions of animations. This analysis identifies the four main roles of an animation tool, and explores the changing role of the tool as it is used by the student.

JELIOT 3

Jeliot 3 is a tool that animates object oriented programs step by step (Moreno *et al.*, 2004). The execution of a program shows how each line is interpreted by the computer. We could say that the supporting theory behind Jeliot 3 is epistemological validity: the animation is used to convey to the student the expert's representation of the program execution.

Jeliot 3's main ability is to automatically create a representation of a Java program in an animation window, see Fig. 2. Different actors in the program are represented in the animation as boxes containing their name and value. Following program execution, new actors are displayed and values are moved from box to box according to assignments and constructs dynamically highlighted as the program is executed. Jeliot 3 can be used as a devel-

opment environment. After successful compilation of a student’s program, Jeliot 3 animates the program at the user-adjusted speed. System output is presented in a console window following animation steps. Thus, in order to get the output of a program, Jeliot 3 animates the whole of it.

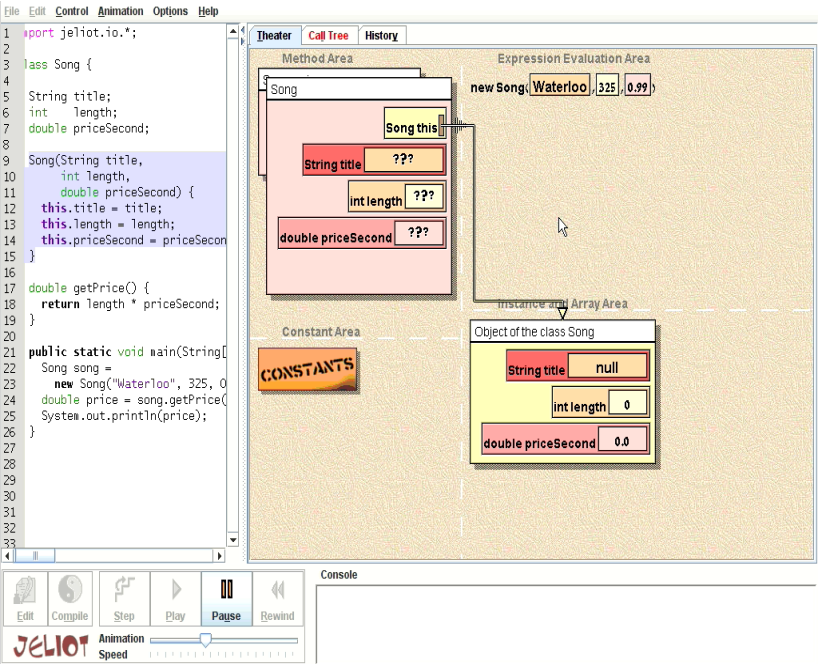


Figure 2. This screenshot of Jeliot 3 shows a new object being created. The source code of the program is in the left-hand side side, and the animation is on the right.

THE STUDY

Sample

Participants in the study were second year undergraduate students at Iringa University College, Tumaini University (Tanzania) studying mathematics education, a Bachelor programme with a strong ICT component. The students were taking part in a 12-week introductory course to programming. Course goals were to introduce basic imperative and object-oriented programming using Java as the programming language.

Students were stratified according to a simple test that graded their computer skills in low, medium and high competence. For each level of com-

petence, two participants were randomly chosen, adding up to a total of six participants taking part in the study. The names of the participants have been changed into pseudonyms in the article to avoid their identification.

Before joining the sessions, participants read and signed an authorization allowing the collected data to be used for research purposes. Formal permission from the university was granted to collect the data within the university campus.

Data Collection

In this study we wanted to see how the students developed their understanding of programming concepts and of their respective Jeliot 3 animations. Thus, to capture the relationship between both aspects and how they evolved, we asked them to describe animations of programs as they were animated.

Furthermore, to check how repeatedly visualizing animations affected subsequent visualizations, students would afterwards visualize and describe a similar program and then write down a description of the same steps they had just watched. This enabled us to track the possible progress in students' understanding and the effect of the animation.

Interviews have provided the data for most qualitative studies in computer science education research. Fleury (2000) showed a set of programs to students and asked them whether those programs will work and why they thought so. Holmboe (2000), on the other hand, asked directly "what is ..?" questions to find out students' understanding of concepts, for example, "what is a foreign key?" In both cases, interviews captured a static image of the students' knowledge, though Holmboe admits that students' verbal answers carry more information (such as hesitation) than written tests.

Verbal descriptions of animations have not previously been a source material of qualitative analysis. In this study, we consider animation descriptions to be analogue to interviews or think-aloud protocols (Ericsson and Simon, 1993)., as they also answer the question "what is happening [on the screen] now?"

Each participant attended three sessions, which were appointed individually and conducted by the first author. Each session focused on one concept, namely, array creation, method calling, and object creation. These concepts were chosen due to the detailed animation that Jeliot 3 produces when animating them. Ecological validity was assured by focusing each session on a programming concept that had been recently taught in the lectures. Sessions had no time limit, but each one lasted an average of 45 minutes. They consisted of five consecutive tasks:

At the beginning of every experimental session students were asked to complete a *pre-test* in order to assess if they had the basic knowledge of the

concepts before proceeding with the session. If students had problems with the tested concept, the first author assisted them before the visualization tasks. In order to ensure no student was disadvantaged by not taking part in the experimental sessions, problems and misunderstandings encountered during those sessions were later reviewed at the course lectures attended by all of the students.

Following this, students moved to a computer where they were shown the source code of a program. They were asked to suggest the purpose of the program before starting its animation, and then were asked to describe every action of the animation during this *first visualization task*. Voice recordings of their descriptions and video recording of the simultaneous events happening on the screen were made while Jeliot 3 was animating a Java program.

Students were then asked to solve a simple *programming task* that required them to slightly modify (2-4 lines of code) the program previously animated in the first task. This was followed by a *second visualization task* in which their solution was again animated and students described it step by step. Finally, students completed a written *post-test* that asked for written descriptions of the steps taken in the animation.

Data Analysis

Recordings of the visualization tasks were transcribed. In order to assess the learning and animation impact, deeper analysis of the transcriptions was restricted to narration of session concepts — array creation, method calling, object creation — as assessed in the pre- and post-tests. Preliminary analysis of the data focused on how the students' knowledge changed as they described Jeliot 3 animations. This analysis revealed that students were not always describing what was happening on the screen as requested, which made it difficult to assess their particular knowledge and its evolution from the available data. Thus, the focus of the qualitative analysis shifted towards the relationship between the student and the tool.

The data analysis reported here follows an inductive category generation process, a technique also used by Eckerdal (2009, p 31) to extract categories from students' interviews. Using inductive category generation, the categories are extracted and refined through an iterative and "reductive" analysis of the data. In her case, the aim was to reveal strategies computer science students use to get *unstuck*. In our case we aimed to identify the roles of the animation tool.

From their descriptions, four categories were initially identified to describe the role of the animations: *teaching*, *exploring*, *asserting* and *confusing*. The categories were redefined and agreed by all the authors after collaboratively coding a set of samples from a transcription. In this phase, the main author's experience with the tool was fundamental when discuss-

ing the meaning of the narration fragments, and the possible roles that they could reveal. This is similar to the importance given by Strauss and Corbin (1998) to *theoretical sensitivity* in Grounded Theory, where the ability of the researcher to gain “insight” due to previous professional or personal experience, allows the researcher to give meaning to the data and identify what is “pertinent” or not to the research.

In this process, the asserting role was relabelled as evaluating role because it captured better the variety of predictive statements given by the participants. With the redefined and agreed categories the rest of the transcriptions were coded using the Transana software (www.transana.org) to find trends on the changing roles of animations.

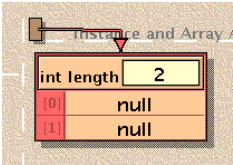
Table 1

Snippet of Student Transcription for the Animation of Array Creation in Jeliot 3 and Categorization of Roles, with Examples of *Exploring* and *Teaching* Roles.

Transcription	Role	Explanation
...	Exploring	The tool makes the student to watch the animation in silence.
Ok, in this case the declaration is String names	Teaching	After watching the animation, the concept is correctly identified.
String names this is in case of arrays	Exploring	The student tries to make sense of the animation.
String brackets names equals new string people	No role	The student reads the source code
We use people, the one we have declared above	No role	The student explains the source code
There string names..	Exploring	The tool prompts an explanation of the animation but the student cannot yet describe it properly and keeps watching
What are you seeing now?	NA	Prompt by the interviewer to describe the animation
int length is number of any kind of element. Two in this case	Teaching	Student correctly describes the array length component

Tables 1 and 2 contain examples of the coding process we used, and illustrate how we mapped the transcribed data to the role categories. In this example, each sentence is categorized in one of the four roles (which are described in the next section) and an explanation is provided to justify the categorization.

Table 2
Snippet of Student Transcription for the Animation of Array Creation in
Jeliot 3 and Categorization of Roles with Examples of the *Exploring*,
Evaluating and *Confusing* Roles.

Transcription	Role	Explanation
This place... let me go to the next step then I will explain...	Exploring	The student is actively following the animation
There are two people		
The name of the people will be placed here.	Evaluating	The student feels he understands the following steps of the animation and tries to guess what will happen next
The first person: names[0]	Exploring	The student is actively following the animation
 <p>Number of people is connected to names.</p>	Confusing	The student mistakenly interprets the visualization of the array and the arrow that points to it. The visualization of an array in Jeliot 3 includes a field for the length of the array (number of people), and the student thinks that the arrow is only pointing at it, rather than the whole array. The picture on the left shows Jeliot 3's visualization with an array and the arrow that points to it.

FINDINGS

Roles of animations

The *teaching role* of the animations is the most obvious one and it shows that animations can be considered beneficial in learning. As well, we can distinguish three more roles, *exploratory*, *evaluating*, and *confusing*. Furthermore, at times the tool had no observable role in the student learning, and thus, we also consider an *empty role*.

Empty role

We say that the tool has an empty role when the student does not actively visualize the animation, or when the visualization changes neither the student's knowledge nor their attitude in an appreciable way.

In our study, several students were quiet when animations of the new concepts were shown repeatedly. In the post-test, most of the times, they did not reveal a better understanding of the animation. For example, John could not correctly describe most of the steps in the object creation, but he was able to complete the programming task correctly. This reveals a gap between the conceptual or theoretical knowledge and the practical knowledge.

Holmboe (1996) developed a “cognitive framework for knowledge in informatics” while analyzing people’s understanding of object orientation. Knowledge is structured in four levels, from *hunches*, or first attempts to understanding, to *holistic knowledge*, where the knowledge is the result of interconnecting the two other kinds of knowledge, *practical knowledge* and *theoretical knowledge*. Students advance in their knowledge starting from hunches and by practice and study they achieve a holistic knowledge. It is at this stage when students have accommodated the knowledge and can operate with it; in constructivism terms, the learnt concept is now part of their first order language.

In Holmboe’s terms we can claim that John has a degree of practical knowledge but he only has hunches of the theoretical knowledge about the concept shown. After repeated views of the animation, he has not made the connection between his practical knowledge and the theoretical knowledge. We can conclude that the animation itself has not been able to improve his knowledge of that concept. In his case we think he has not made an active effort to understand the animation as he could have thought that the practical knowledge he already had was enough.

A similar effect was noted with animation of object references. Even if they were present during the whole object creation process, students did not attempt to describe them. While the concept of references is important, the student still had a practical knowledge of objects that let them modify the source code and understand what the program did.

Exploring role

Animation tools have an exploratory role when they prompt the student to explore or to discover the meaning of the animation and the animated concept through an active visualization.

Jeliot 3 animations regularly resulted in students making guesses of their meaning independently of the students’ previous knowledge. Through their narrations, we could detect students making a conscious effort to actively visualize the animation.

The exploring attitude could be seen in a range of situations. David and Martin, for example, did not narrate any of the object creation steps, but asked what the animation meant and then watched it again. Their missing knowledge became apparent to them, and they were ready to use the animation to gain new insights. At times, students halted their narrations while they tried to understand the animation — they started describing the animation, but became silent when they found out that they could not describe all the steps involved. We could detect the exploring role of the animation because they tried to describe the animation. However, in some cases, after watching the animation in silence a student would not describe what had happened, resulting in an apparent empty role for the animation.

The exploring role is indicative of the potential of animation tools, as it requires a mentally active visualization by the student that usually precedes the learning. However, not all the students engaged in exploration, and not all students who explored appeared to have improved their knowledge.

Confusing role

The confusing role of an animation tool is often an unintended one. Animations are made to clarify or to serve as a learning tool. It occurs when the student cannot answer the questions that the active visualization provokes.

After exploring the animation, students' first narrations often contained statements that revealed their *fragile knowledge* (Perkins and Martin, 1986). Four kinds of fragile knowledge were described by Perkins and Martin — missing, inert, misplaced, and conglomerated — the last two being exposed by the visualization tool through the students' descriptions. *Misplaced knowledge* represents knowledge that the student applies but is not relevant in the current context, whereas *conglomerated knowledge* represents those cases when students' code contains "disparate elements" that are not supposed to be together.

This fragile knowledge can relate to students' understanding of the animation and the concepts the animation explains, and students' conceptual misunderstandings were revealed by incorrect usage of terms. In the description of an array creation, we can find good examples of this fragile knowledge. For instance, David mixed the terms *variable*, *literal* and *array*, showing his conglomerated knowledge. Michael described a newly created array without using the word "array", but used what he could recognize on the screen, namely the variable called "length" of such an array. This is an example of misplaced knowledge, as he equates array creation to variable declaration.

Fragile knowledge is partly due to the confusing role of the animations, which are not always self-explanatory — we have observed that they may cause students in their descriptions to put disparate concepts together or to focus on secondary/supporting actors.

Students' descriptions are not only the result of the animation, but also of their previous knowledge. Thus, their attempts to describe a phenomenon for which they do not have words may result in an incoherent narration, similar to the hunches described by Holmboe (1999).

After repeated visualizations in the sessions, most of the students' descriptions remained confusing, or students stopped describing the whole concept, not being able to put the correct words to what they were seeing. In a few cases the confusing description of the animation was the prelude for learning, but we cannot identify the reasons that caused confusion to be a trigger for learning.

To summarise, we consider that Jeliot 3 had a confusing role when the animations failed to improve students' fragile knowledge; this role was identified when students' descriptions consisted of apparently incoherent sentences.

Teaching role

An animation tool will have a teaching role when it has successfully been used for learning by the student. This role is expected for any animation tool, as learning through animations is the final goal.

In the case of Jeliot 3 in this experiment, the teaching followed a behaviouristic approach: students were supposed to learn when they watched repeatedly the animation. In our study, students improved their descriptions of the animations after two visualizations, albeit in different grades, and not for every concept. In other words, they were able to describe correctly more steps in the post-test, and to some extent in the second visualization than in the first visualization. Thus, animations have had a general teaching effect.

The teaching role was not only seen after repeated visualizations, but also during the visualizations, for example, when students gave a proper description of the animation after a moment of exploration. The exploring role in this case was a catalyst for learning.

Repeated visualizations of the animation also prompted students to correct their confusing statements, improving their understanding. For example, Nicholas's first description of the object creation mentioned a new method call when the object reference was returned by the constructor. His explanation was an incorrect interpretation of the animation. When he described it in the second visualization, the return related steps were correctly described, and there was no mention of any new method call.

Evaluating role

The animation tools have an evaluating role when students use them to evaluate or assert their own knowledge.

The evaluating role of the animation was evident when students were describing the animation steps before they actually happened. The required descriptions turned into spontaneous predictions which showed how students had already matched the code to its corresponding animation. However, the fact that they wanted to predict rather than describe was two-fold. On one hand, the predictions were actually assertions of the students' knowledge — students considered that the animation itself would not produce any new information, and that there was no need to actually watch it to describe it. On the other hand, the predictions had a self-evaluating component — we observed that students actually watched the animation and incorrectly predicted it. After visualizing the animation, students either corrected their predictions or did not do so.

The most common assertion referred to assigning values to parameters and attributes. This could be explained by the fact that simple assignments had been visualized several times before, and the students thought that they had mastered that concept. However, for more complex statements, their predictions were incomplete or confusing, and the evaluating role of the animation tool showed students' misunderstandings. For example, Michael's prediction of a method call shows a problem expressing clearly what he thinks is going to happen. He refers to the method call animation as a substitution of values, in other words, the method call representation is substituted by the value it returns.

When predicting, and as a part of the self-evaluation role of the tool, some students attempted to add an explanatory prediction. For example, Timothy tried to go beyond merely describing the change of flow that a return statement would cause, but also tried to explain why it has happened, suggesting a higher level of understanding.

As mentioned before, incorrect assertions were often maintained after watching the animation, revealing students' fragile knowledge. In this case, while the tool prompted students to evaluate their knowledge, it did not have a role to correct it. For example, David's confusion with "array" and "length" was a prediction that was repeated after two visualizations. So in his case, the tool had three simultaneous roles: 1) confusing, as he cannot distinguish the length from the whole array; 2) asserting, as he is repeating the confusion in a prediction; and 3) empty, as the tool did not help him gain the correct understanding.

Variation of animation roles across time

In the introduction, we hypothesized that these four roles would linearly guide students' progress towards understanding: from exploring new concepts to evaluating the knowledge they have gained, passing through confusion and learning. However, the analysis of the data revealed that was often not the case.

In Fig. 3 we have depicted the variety of usage trends of the animation tools by individual students. Links between the roles indicate the role change of the tool, or lack of it in the case of a loop, when a student tries to understand a single concept.

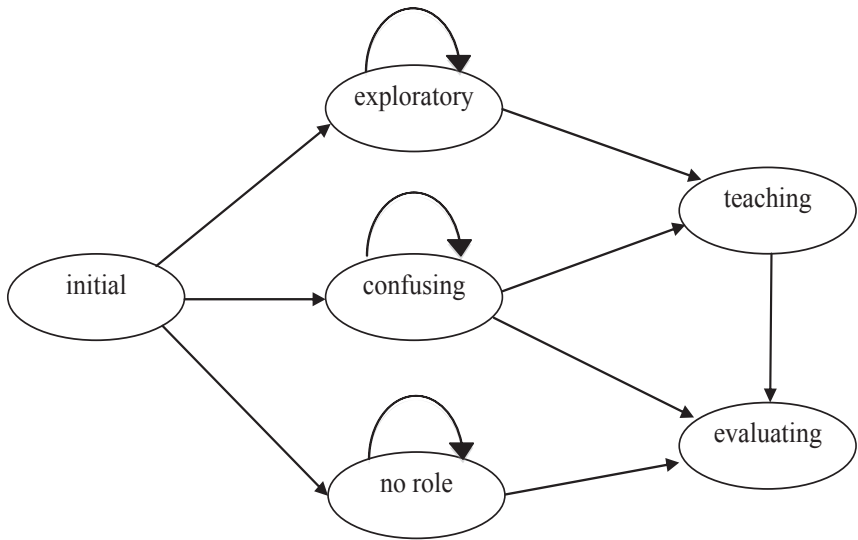


Figure 3. The tool takes several roles as students use it to learn a new concept. In the Figure the arrows represent the changes of roles as the animation of the new concept is watched once or more times.

Before reaching the evaluating role, students used the tool in many different combinations, and a clear pattern did not emerge from the data. Moreover, not all of the students used the tool to evaluate their knowledge of a given concept. In some cases, these students found themselves only using the tool to explore; in other cases, the animation remained confusing to them, even after having used Jeliot 3 repeatedly to understand the same concept. Also, some students started by evaluating their knowledge, with animations seemingly not taking any active role during the session. This is represented by the path in the bottom of Fig. 3: *initial* \rightarrow *no role* \rightarrow *evaluating*.

A good example of the linearly changing roles of the tool is Martin's narrations of array creation. He was brief in his narrations; he abstained from predicting the array creation on the first event, and requested to watch the animation first. After watching the array creation for the first time, he confused the box representing the length of the array with the whole array. When the second array creation was going to be visualized he anticipated the reference assignment step, albeit at the wrong time. He made the same error the next time the second event was animated. Finally, in the post-test he showed knowledge about space allocation and the reference assignment. During his visualization of the array creation, the animation tool prompted

him to *explore* the animation; he became *confused* when describing one of the steps, and finally showed that the tool had *taught* him the steps in the animation as reflected in the post-test. In this occasion the tool had not had a role in evaluating Martin's knowledge of array creation.

DISCUSSION

Think-aloud protocols have been used before to investigate learning in context (Chi, 1997), when students are solving a task, and to infer the mental models of the students regarding the topic. In our case, students were not solving a problem by describing the animations, but describing animations resulted in a problem, where students have to think and produce explanations they have not thought of before the task itself.

Asking students to describe animations on the screen proved not to be as descriptive as we had expected. Students' descriptions usually related to the source code rather than its representation on the screen. On the other hand, students' timing of the narrations has given us insight into the role of the tool, especially for the evaluating role.

The fact that the animation tool also has an empty role was unexpected, and may be explained by several factors that we had no control over. For example, a student may not have been motivated to use the tool, or may have missed the lecture about the animated concept, resulting in a non-ideal use of the tool by the student.

The exploring and evaluating roles reported here may be dependent on the task that students were involved in. The act of describing an animation can force the student to explore it and to ask himself whether he will know what will happen next. If the student had been using the animation tool for debugging purposes, exploring or evaluating the animation would have taken a secondary role. These two roles are important for learning, and animation tools can deliver them.

The confusing role of the animation reveals one problem from Jeliot 3, and potentially other animation tools. Simple concepts and their animations were described better than the complex ones. Students' descriptions of complex concepts, or lack thereof, revealed the problems they had in understanding the animations and the concepts explained with the animations. Students were seeing what was happening in the program, but they were not able to abstract the animation metaphors to programming concepts. When they tried, they showed a mix of conglomerated and misplaced knowledge.

This partly matches Ben-Bassat Levy *et al.*'s (2003) findings: Jeliot 3 is more beneficial with simpler concepts than with complex ones. Ben-Bassat Levy *et al.* also mention that Jeliot 3 helps students by establishing a common vocabulary to communicate with the teacher. In our case, this did not happen when complex concepts were being described. Following Holm-

boe's (2000) interpretation of Vygotsky's ideas on language and learning, one possible reason is that for simpler concepts, a student's previous vocabulary is enough to describe what happens on the screen and in the program. For more complex concepts, students are still assimilating the concept, and the terms required to describe the concept do not form part of the students' first order language (composed of the words that are self-explanatory).

We can assume that when the tool is taking a confusing role, the students find themselves in the zone of proximal development (Vigostky, 1978), in which they need the assistance of a tutor to advance to the learning role. If the tool or the teacher were able to determine when the tool is taking the confusing role, they could react and provide the necessary *scaffolding*, or interaction with the student, to make the student achieve a holistic knowledge of the explained concept.

Repeated use of Jeliot 3 in order to complete the exercises or to describe the animations has not helped all the students to understand the animations or the concepts, showing a lack of theoretical knowledge.

The lack of theoretical knowledge is not helped by the fact that Jeliot 3 provides visual clues to the concept, and little verbal information about what is happening. Mayer and Anderson (1992) have noted the importance of simultaneously providing verbal explanations for visual animations to benefit from simultaneous aural and visual processing. Animation tools should be used in contexts where the link between the visual material and the theory is emphasized, especially the relevant vocabulary.

The roles described here could be the basis of real time adaptation of animation tools. First, real time data could be collected by means of interactive features, such as simple questions (Myller, 2007). Then, for example, when the tool identifies its own no-role, the tool would adapt to make itself more relevant. Loboda and Brusilovsky (2010) have created a tool, WADEIn II, which adapts the visualization and textual commentary to the variable knowledge of the user. Moreover, it features two modes of interacting with the tool — exploration and evaluation — akin to the roles explained here. In WADEIn II, the student chooses which role the tool takes; adaptation could be taken to another level if the tool changed seamlessly between the exploration and evaluation roles, and if the tool identified the two new active roles: teaching and confusing.

Further research should explain the reasons for successful transitions from the confusing role to the teaching role in animation tools. One trigger for the transition is a cognitive conflict in the student's mind (Ma *et al.*, 2008), which happens when a student is informed that their current perception or understanding is incorrect. In the research carried out by Ma *et al.* (2008) students were asked specific programming questions, and watching animations resulted in better learning for those who were confronted first

with their incorrect assumptions. Thus, the first stage for a successful transition is for students to be aware of the confusing role of the animation and of their own lack of understanding.

The results of this study suggest that the novel activity of describing the animation, rather than just watching it, have had a positive impact on students' learning. A future empirical study could compare the learning impact of describing animations with other ways of interacting with visualization, as those from engagement taxonomy described by Naps *et al.* (2002).

CONCLUSION

This study has identified and described four roles that an animation tool takes when it is used to describe new programming concepts. The four roles — exploring, confusing, teaching and evaluating — reflect the relationship of the students with the tool. When learning a new programming concept, the roles taken by the tool may change in unpredictable ways.

The task of describing the animations stimulated students to explore the animations and to evaluate their learning through the animation, thus promoting active visualization. This active visualization revealed the potential of the tool's automatic animations. It also revealed that, sometimes, students could not understand or explain its animations. Based on our findings, we will modify the tool to include verbal descriptions to explain complex concepts. For educators, we suggest that they encourage their students to use the tool in versatile ways, especially for active visualization of the animation that promotes the exploring and evaluating roles of the tool. Finally, further research should look into the transition of the roles of the tool and how to make identify the current role to guide the student to learn the new concept using the tool.

References

- Ben-Bassat Levy, R., and Ben-Ari, M. (2007). We work so hard and they don't use it: acceptance of software tools by teachers. *SIGCSE Bulletin*, 39(3), 246–250.
- Ben-Bassat Levy, R, Ben-Ari, M., and Uronen, P. A. (2003). The Jeliot 2000 program animation system. *Computers and Education*, 40(1), 1-15.
- Byrne, M. D., Catrambone, R., and Stasko, J. T. (1999). Evaluating animations as student aids in learning computer algorithms. *Computers and Education*, 33(4), 253-278.
- Chi, M. T. H. (1997). Quantifying qualitative analyses of verbal data: A practical guide. *Journal of Learning Sciences*, 6, 271-315.
- Eckerdal, A. (2009). *Novice programming students' learning of concepts and practise*. Doctoral dissertation, Uppsala University. URL: <http://uu.diva-portal.org/smash/record.jsf?pid=diva2:173221>
- Ericsson, K., and Simon, H. (1993). *Protocol Analysis: Verbal Reports as Data* (2nd ed.). Boston: MIT Press

- Fleury, A. E. (2000). Programming in java: student-constructed rules. *SIGCSE Bulletin*, 32(1), 197-201.
- Holmboe, C. (1999). A cognitive framework for knowledge in informatics: the case of object-orientation. In *ITiCSE '99: Proceedings of the 4th annual SIGCSE conference on innovation and technology in computer science education* (pp. 17–20). New York, NY, USA: ACM.
- Holmboe, C. (2000). A framework for knowledge: Analysing high school students' understanding of data modelling. In A. B. E. Bilotta (Ed.), *12th workshop of the psychology of programming interest group* (pp. 267–279).
- Hundhausen, C. D., Douglas, S. A., and Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13(3), 259–290.
- Iiyoshi, T., Hannafin, M., and Wang, F. (2005). Cognitive tools and student centred learning: rethinking tools, functions and applications. *Educational Media International*, 42(4), 281–296.
- Loboda, T. D., and Brusilovsky, P. (2010). User-adaptive explanatory program visualization: evaluation and insights from eye movements. *User Modeling and User-Adapted Interaction*, 20(3), 191-226
- Kim, B., and Reeves, T. C. (2007). Reframing research on learning with technology: in search of the meaning of cognitive tools. *Instructional Science*, 35(3), 207–256.
- Ma, L., Ferguson, J. D., Roper, M., Ross, I., and Wood, M. (2008). Using cognitive conflict and visualisation to improve mental models held by novice programmers. In *Proceedings of the 39th SIGCSE technical symposium on computer science education* (pp. 342–346). New York, NY, USA: ACM.
- Mayer, R., and Anderson, R. (1992). The instructive animation: helping students build connections between words and pictures in multimedia learning. *Journal of Educational Psychology*, 84, 444-452.
- Mayer, R. E. (1981). The psychology of how novices learn computer programming. *ACM Computer Survey*, 13(1), 121–141.
- McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *Working Group Reports From ITiCSE on innovation and Technology in Computer Science Education* (pp. 125-180). ITiCSE-WGR '01. ACM, New York, NY.
- McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., et al. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *SIGCSE Bulletin*, 33(4), 125–180.
- Moreno, A., Myller, N., Ben-Ari, M., and Sutinen, E. (2004). Program animation in Jeliot 3. *SIGCSE Bulletin*, 36(3), 265–265.
- Myller N. (2007). Automatic Generation of Prediction Questions during Program Visualization. *Electron. Notes Theor. Comput. Sci.* 178, 43-49.
- Naps, T.L., Rößling, G.R., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. and Velázquez-Iturbide, J.A. (2002). Exploring the Role of Visualization and Engagement in Computer Science. *SIGCSE Bulletin*, 35(2), 131-152.
- Naps, T., Cooper, S., Koldehofe, B., Leska, C., Rößling, G., Dann, W., Korhonen, A., Malmi, L., Rantakokko, J., Ross, R. J., Anderson, J., Fleischer, R., Kuittinen, M., and McNally, M. (2003). Evaluating the educational impact of visualization. *SIGCSE Bulletin*, 35(4), 124-136.

- Perkins, D. N. and Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. In *Papers Presented At the First Workshop on Empirical Studies of Programmers on Empirical Studies of Programmers* (pp.213–229). E. Soloway and S. Iyengar, Eds. Ablex Publishing Corp., Norwood, NJ, 213-229.
- Strauss, A., and Corbin, J. M. (1998). Basics of qualitative research : Techniques and procedures for developing grounded theory. SAGE Publications. Paperback.
- Vygotsky, L. S. (1978). Interaction between learning and development. In M. Cole, V. John-Steiner, S. Scribner, & E. Souberman (Eds.), *Mind in society: The development of higher psychological processes* (pp. 79-91). Cambridge, MA: Harvard University Press.
- Whalley J. L., Lister R., Thompson E., Clear T., Robbins P., Kumar A., and Prasad C. (2006). An Australasian study of reading and comprehension skills in novice programmers, using the bloom and SOLO taxonomies. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52 (ACE '06)*, Denise Tolhurst and Samuel Mann (Eds.), Vol. 52. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 243-252.

Author Notes

Andrés Moreno is a PhD student in the Department of Computer Science and Statistics at the University of Eastern Finland. He is currently working in educational technology and his main interests are visualization tools, programming education, and ICT for development. He has been an active developer of the programming visualization tool Jeliot 3, which he has used to teach programming in Finland, England and Tanzania, and he has published more than 20 papers in journals and conferences.

Mike Joy received the masters' degrees in mathematics from Cambridge University and in postcompulsory education from the University of Warwick, the PhD degree in computer science from the University of East Anglia, and has also received both the CEng and CSci degrees. He is an associate professor in the Department of Computer Science at the University of Warwick and a member of the Intelligent and Adaptive Systems research group. His research interests include educational technology, computer science education, object-oriented programming, and Internet software, and he is the author or coauthor of more than 100 papers. He is a chartered fellow of the British Computer Society and a fellow of the Higher Education Academy.

Erkki Sutinen received the PhD degree in computer science from the University of Helsinki in 1998. He is the leader of the edTech group (www.cs.joensuu.fi/edtech). He has been the head of the Department of Computer Science and Statistics at the University of Joensuu, currently University of Eastern Finland. His research interests include ICT for development (ICT4D) and designing and analyzing technologies for complex subject domains, like programming, in developing countries, and within special education. The applied techniques cover visualization, information retrieval, data mining, robotics, and design models. He has coauthored and published more than 100 research papers, and 12 of his supervised or cosupervised PhD students have completed their studies. He has also worked at Purdue University (1998-1999), the University of Linköping (2000-2001), and Massey University (2006), and is an adjunct professor at Tumaini University, Tanzania.