Constructivist and Constructionist Approaches to Algorithm Visualization Construction: A Proposal

Ming-Han Lee Guido Rößling
Department of Computer Science
TU Darmstadt
Darmstadt, Germany
{minghan, guido}@tk.informatik.tu-darmstadt.de

***Abstract*—The didactic focus on the use of algorithm visualization has been shifting to its construction and presentation by students. Reviewing the principles of Constructivism and Constructionism, this paper proposes some new approaches to constructing algorithm visualizations.**

Keywords-Algorithm Visualization, Computer Science Education, Constructivism, Constructionism

I. INTRODUCTION

CS educators have used algorithm visualization (AV) animating abstract programming languages to facilitate learning. Researchers conclude that passive viewing of computerized animations contributes little to the learning experience [7]. Stastko [18] suggests that students construct their own visualizations instead of relying on instructors as content-providers. Hundhause et al. [5] also recommend that students not only construct but also present their AV constructions. Both approaches are in accordance with the principles of Constructivism and Constructionism.

Drawing on the Constructivist and Constructionist learning theories, we have reflected on new approaches to the AV system design and AV deployment that may shed some light and expand the current practice of algorithm visualization construction.

II. THE *V* WORD AND THE *N* WORD

Papert refers to ConstructiVism and ConstructioNism as the V word and the N word [15] respectively due to their close kinship. This paper addresses Constructivism with reference to Piaget's learning theories, upon which Papert's Constructionism is based.

A. Constructivism

Let us imagine for a moment that all IKEA assembly instructions came in pages of writing without any illustrations. No matter how good the writing is, putting together a piece of furniture just seems a lot more formidable. In the similar way IKEA's graphical illustrations considerably lessen the cognitive load demanded of customers; visualizing programming code converts abstract process and state changes into tangible representations. Doing so alleviates the students' cognitive load by giving them direct mental mappings and auxiliary memory capacities. It is about using dynamic visual artifacts in our everyday experiences such as numbers, colors and shapes to represent imperceptible mathematical calculation or evanescent functional sequences.

The idea of deploying AV technology in the classroom corresponds with the Constructivist learning theory. It postulates that knowledge is never simply passed from the giver to the receiver, but a product of active construction based on an individual's experience and disposition.

B. Constructionism

The Constructivist theorists have provided IKEA customers with graphical illustrations as step by step instructions. However, they are inherently more attentive to the utilization of familiar symbol systems to enhance customers' comprehension; the focus is on knowledge construction by providing a mental image. Constructionist theorists, on the other hand, are concerned with the actual building of a piece of furniture. In other words, Constructionists encourage direct hands-on experience by putting together a piece of furniture before or without consulting an instruction manual.

Whereas Cognitivism came into being as a contending response to Behaviorism, Constructionism evolved out of Constructivism. Constructivism spotlights knowledge as a cognitive construction internally. Constructionism, on the other hand, underlines the external construction of entities as a learning process.

III. OUR PROPOSAL

We believe both learning theories contribute directly to the pedagogical effectiveness of AV deployment in the curriculum. Following is a set of outlines we derived from the constructivist and constructionist principles. We propose that an AV system can improve its pedagogical effectiveness by the following five guidelines.

A. Institute A Platform for Experimentation

By virtue of the terminology, *visualization* comes after *algorithm*. Current AV systems are historically conceptualized and designed as a tool that generates animations to make code understandable. Students would first encounter a certain algorithm they need to learn for the

course, and are then given the visualization or asked to construct the visualization for it. No code, no visualization.

The Constructivist and Constructionist theories, however, inspire a different didactic methodology. What if we reverse the usual practice, and have visualization come first and code second? What if we have the students tackle a given problem first before teaching them the algorithm? Classically, we employ the Instructionist approach where an instructor teaches bubble sort by first presenting the code and then uses illustrations and examples to explain its inner workings. Alternatively, why not give our students an actual sorting problem which they need to solve with constructing visualizations before they actually know what bubble sort does? In this scenario, students do not simply construct visualizations to reflect the workings of a certain algorithm; they explore their own algorithmic solutions of the problem before learning the "official" solution. We will not be too surprised if students stumble upon bubble sort or related sorting algorithms on their own, or even come up with their own algorithms.

B. Use Real-World Model

In many cases, text books, instructors and a number of AV systems have used weighted numbers or sticks arranged in random order lined up in an array to illustrate various sorting procedures. Numbers and sticks are indeed very *effective* in getting the message across due to their instant accessibility. However, such genericness may leave less of a long-term impression compared to visualizations constructed based on real-world examples, such as the Storyboard technique pioneered by Hundhausen et al. [5], that makes learning anchored in students' experience and therefore *meaningful*. When trying to solve a problem or quickly comprehend something, it is helpful and effective to reduce our problem to a set of numbers or simple graphics. Such abstraction helps us to arrive at a solution more quickly. However, by reversing that abstraction process and encourage students to relate to their personal experience and seek out real-world examples when constructing their own AV, students are more likely to think outside of the box and transfer their algorithmic knowledge from a binary environment to concrete and real applications. Learning should not be merely effective; more importantly, it needs to be meaningful too.

C. Enable Direct Manipulation

Looking at the evolution of AV construction methods, we observe a trend in which the creation of graphical objects through strenuous coding is gradually replaced by more natural and intuitive human behaviors that does not require much learning. While manually sketching graphics -- as opposed to clicking on a button to generate pre-made graphics -- is already supported by a few AV systems, we hope for a system that support the direct manipulation of sketched graphical objects without having to describe the action [14]. This has more to do with the essence of constructing something than just saving the time overhead. Writing or something is debatably a more immediate and authentic experience than typing, and the experience of clicking on a button to generate a pre-made graphic is understandably a less expressive and less memorable one compared to what users experience when they are given free reign to their imagination and can draw anything they wish. To be able to move objects freely without a formal description also emphatically adds spatial movement, an important signifier for state changes to the repertoire of visual representations that are otherwise difficult to illustrate with other properties.

D. Incorporate Audio into AV Construction

Many students will attest that only when they are capable of explaining a subject matter, either to themselves or someone else, can they be sure of that they have really understood something. By speaking out loud, students can "hear themselves learn" by making the internal and implicit external and explicit. The contemporary support for AV construction has mainly focused on the visual, but neglected the audio. In our experience, however, almost all students that are asked to visualize algorithms with simple art materials would invariably explain orally what they are doing. We see no convincing reason why the support of audio input, with which students capture not only the visual, but also the audio part of their AV construction, should be left out of the system implementation. Not only will the oral narration augment the visual representation and therefore avoid ambiguity and increases understanding, it also allows the students to hear themselves think. When both the visual and audio are available for playback, it also makes it easier to identify logical or semantic errors, if there are any. Should students for some reason be unable or unwilling to provide a voice narration during AV construction, they should have the option of writing down and documenting their thoughts. The idea here is to capture and document as much the reasoning and thinking process as possible as a public entity Papert speaks of for future reference.

E. Open Access to Peer-Generated Content

By constructing their own algorithm visualization, students construct their own version of that algorithmic knowledge. By making their constructed version of knowledge accessible to other students, they invite feedback and comments. By comparing their own interpretation of that piece of knowledge with others', it induces the assimilation and accommodation processes, two fundamental phenomena crucial to learning theorized by Piaget. Old, false mental models are disregarded and updated by new ones; incomplete information is complemented or supplemented. The concept of collaborative learning also envisions an AV system where students can even work in groups to construct AV together.

Doing so brings about the social interaction that is favorable to successful knowledge construction [2]. When we make the sharing, debating, collaborating and evaluating part of their active learning process, students are truly conducting a dialogue of collaborative construction of knowledge.

IV. Conclusion and Future Work

Having understood the importance of active learning, and with the intention of increasing student engagement in using the AV technology, educators and researchers have implemented new features into AV systems, most notably the support for students to construct their own algorithm visualization. Studies have indicated that having students construct their own algorithm visualization has more impact on learning than having them passively view AV pre-made by instructors. With the methods for constructing visualization improving with each iteration in terms of user-friendliness, intuitiveness and time efficiency; with the didactic discourse centers around the construction and presentation of algorithm visualization, there is all the more reason to address pedagogical effectiveness of the AV construction methodology based on an appropriate theoretical framework.

This motivates us to review the Constructivist and Constructionist learning theories and reflect on how both interrelated theories can contribute to the AV's pedagogical effectiveness. Particularly compelled by the Constructionist standpoint, this paper focuses on the AV construction methodology as the principal factor that facilitates students' knowledge construction. We then propose a set of guidelines derived from both learning theories for the design and deployment of AV systems in the curriculum.

Based on these guidelines, we have begun working on a browser-based AV system that seeks to give students a natural visualization construction experience through manual sketching and direct manipulation without any coding. The browser is to be the piece of paper where students conceptualize, experiment and devise their solutions for given problems. Students will not only manually construct "low fidelity" [6] visualizations, their oral narration during the visualization will also be captured. Students using the AV system will have access to each other's work and be able to playback, evaluate and comment on each piece of "constructed knowledge".

A study to determine the functionalities needed for a computer-based AV system to replicate the low tech AV construction experience as close as possible is currently under way.

Acknowledgment

We are grateful for the funding provided by German Research Foundation (DFG) in support of our research project.

References

[1] Ackermann, E. 2001. Piaget's Constructivism, Papert's Constructionism: what's the difference? In Constructivism: uses and perspectives in education, Vol. 1&2. Conference proceedings, Geneva: research center in education. p.85-94.

[2] Ben-Ari, M. 1998. Constructivism in computer science education. In SIGCSE Bulletin Vol. 31 Nr.1. p.257-261. ACM, New York.

[3] Cross II, J. H., Hendrix, T. D. 2006. jGRASP: a lightweight IDE with dynamic object viewers for CS1 and CS2. In Proceedings of the 11th annual SIGCSE conference on innovation and technology in computer science education, ITiCSE 2006. ACM, New York.

[4] Douglas, S., Hundhausen, C. and McKeown, D. 1996. Exploring human visualization of computing algorithms. In GI '96: proceedings of the conference on Graphics interface '96. p.9-16. Canadian information processing society, Toronto, Canada.

[5] Hundhausen, C.D., Douglas, S. A. 2000. SALVA and ALVIS: a language and system for constructing and presenting low fidelity algorithm visualizations. In VL, p.67-68.

[6] Hundhausen, C. D., Douglas, S. A. 2000. Shifting from "high fidelity" to "low fidelity" algorithm visualization technology. In SIGCHI 2000 extended abstracts. Conference on human factors in computing systems. p. 179-180. ACM, NewYork.

[7] Hundhausen, C.D., Douglas, S. A. and Stasko, J. T. 2002. A meta-study of algorithm visualization effectiveness. In Journal of Visual Languages & Computing. Vol. 13, Nr. 3, p.259-290.

[8] Karavirta, V. et al. 2004. MatrixPro - a tool for demonstrating data structures and algorithms ex tempore. In Proceedings of the IEEE international conference on advanced learning technologies, ICALT 2004. IEEE Computer Society.

[9] Jonassen, D. 1994. Thinking technology: Toward a constructivist design model. In educational technology, 34(4), p.34-37.

[10] Malmi, L. et al. 2004. Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. In Informatics in education, Vol. 3, Nr. 2, p.267-288.

[11] Moreno, A. et al. 2004. Program animation in jeliot 3. In Proceedings of the 9th Annual SIGCSE conference on innovation and technology in computer science education, ITiCSE 2004. p.265. ACM, New York.

[12] Naps, T. L., Eagan, J. and Norton, L. L. 2000. JHAVÉ – an environment to actively engage students in web-based algorithm visualizations. In Proceedings of the 31st SIGCE technical symposium on computer science education. p. 109-113. ACM, New York.

[13] Naps, T. L. et al. 2002. Exploring the role of visualization and engagement in computer science education. In ITiCSE-WGR '02: working group reports from ITiCSE on innovation and technology in computer science education. p.131-152. ACM, New York.

[14] Narayanan, N. H and Hübscher, R. 1998. Visual language theory: towards a human-computer interaction perspective. In Visual language theory. p.87-128. Springer-Verlag. New York.

[15] Papert, S. 1991. Situating Constructionism. In Constructionism. MIT Press. Cambridge, MA.

[16] Papert, S. 1993. The children's machine, rethinking school in the age of the computer. BasicBooks. New York.

[17] Rößling, G., Schüler, M. And Bernd, S. 2000. The ANIMAL algorithm animation tool. In ITiCSE '00: Proceedings of the 5th annual conference on innovation and technology in computer science education. ACM, New York.

[18] Stasko, J. T. Using student-built algorithm animations as learning aids. In SIGCSE bulletin. Vol. 29, Nr. 1. p.25-29. ACM, New York