# A Little of that Human Touch in the Algorithm Visualization Construction Approach

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

**Abstract**: Current approaches to algorithm visualization (AV) construction run the gamut from low-level programming to sketching on the Natural User Interface (NUI). While we witness a constant improvement in terms of ease of use and learning curve, existing methods still strike us as somewhat wanting in naturalness, if not downright restrictive. Based on observations of an on-going study, we argue that in addition to drawing any graphics, students should also be able to directly manipulate the sketched graphics and control their movement. Moreover, recording of students' spoken explanation during the construction process should be supported by the AV system. By integrating hand movement and voice narration, two basic human affordances, into a new focus of the algorithm visualization construction approach, our work-in-progress aims to replicate the kind of low-tech, pen-and-paper AV construction experience and translates it into a computer-based AV system.

## Introduction

The subject of algorithm is a challenging one for many Computer Science students, especially those who have just begun developing their coding skills. Not only do they need to understand how computers think, they also need to think like a computer. They learn to write special recipes for the computer to cook up the right kind of software. They also follow someone else's recipe to find out why a baked program does not come out of the oven the kind of cake it is supposed to be.

These special recipes are step by step instructions to the computer written in programming languages, and learning new languages, human or otherwise, is a demanding task. To help students better code and decode these special recipes, graphical illustrations are often drawn on whiteboards or in textbooks to demonstrate the abstract algorithmic processes. The proliferation of personal computers in the education sector encouraged the attempts to bring these illustrative diagrams and graphics from whiteboards and textbooks to the computer screen. It was not long before educators began to take advantage of computers' multimedia power to animate these initially static graphics. When Baecker presented his *Sorting Out Sorting* video [Baecker 1981], it quickly captured the imagination of CS educators and inspired their enthusiasm that led to a series studies and development of algorithm visualization (AV) software tools.

In the early days, AV was often nothing more than computer animations shown to students in the classroom. Students watched these animations as if they were watching TV, though probably with less interest. However, as studies on the pedagogical value of AV remained inconclusive, the skepticism among educators also grew. Studies, such as (Hundhausen, Douglas & Stasko 2002), indicated that students benefit very little from passively viewing algorithm visualizations prepared by their instructors. To foster an environment conducive to active learning, students should actively construct their own visualizations (Stasko, Badre & Clayton 1993). As the didactic focus shifts from teacher provided content to student constructed knowledge, whether an AV system supports students creating their own visualizations becomes a pivotal factor in discerning its pedagogical value. As a result, the ease, time-efficiency of content creation begins to receive more attention from AV designers.

## The Many Ways of AV Construction

Early AV systems were made for and used by faculty members to facilitate instructions in the classroom. User-friendliness often only came as an afterthought. To create visualizations, users had to use programming languages dictated by the AV system to laboriously code graphics and their behaviors.

Programming everything from the scratch is a time-consuming endeavor, and it poses a daunting task for an unversed programmer. Moreover, studies affirming algorithm visualization's pedagogical effectiveness were still yet to be seen. For many CS education practitioners, there was very little to justify the amount of time and effort needed just to algorithm visualizations for the students.

As student-generated content began to receive increasing didactic focus, and the significance of student-constructed visualization was recognized, ease of AV construction also became increasingly important. Not only should AV construction become easier and faster, it should be attainable for someone with limited programming experience too. Developers achieved this goal by putting an abstraction layer on top of their software systems in form of a set of limited commands with simplified syntax. Using these commands, users are able to put simple, pre-programmed elements such as dots, lines, shapes and colors onto the computer screen, and then change their properties, such as their size or spatial location.

With the omnipresence of the Windows, Icon, Menu and Pointing device (WIMP) interface, all instructions and commands became iconized and buttonized. Users use an input device to dive into layers of menus, drive through multiples of icons, and click on an array of buttons to put pre-made graphics in the pre-programmed areas on the fly.

The latest advances in browser technology and the Natural User Interface (NUI) enable users to sketch anything freely on the computer screen. Such implementation for AV construction is still only sporadic with very limited functionality.

The AV construction methods outlined above constitute varied degrees of user-friendliness and learning curve. Even if different AV systems differ with one another in their pedagogical focuses and core functionalities, their AV construction methods tend to belong to one of these four categories. A few systems offer either two or more methods completely independent of one another, or a combination of more than one methods for constructing visualization.

From having to churn out line after line of code to utilizing the more intuitive WIMP and NUI is certainly a big step forward in terms of intuitivity and time-efficiency. How do we further contribute to not only the newbie-friendliness methodology but also the pedagogical effectiveness of an AV system?

## The Case for More "Human Touch"

Inspired by an early study (Hundhausen, Douglas & McKeown 1995), we began conducting a similar experiment in which students are observed using art supplies such as pen, paper, scissors and glue to construct handmade algorithm visualizations. Our study differs from the previous one in that our students are free to pick any algorithms to demonstrate, instead of everyone working on the same algorithm (in this case, bubble sort). The objective of our study is to identify and categorize the manual activities and operations students have undertaken in a real world setting, and then determine the functionalities we need to implement in our AV system so as to eliminate all coding and issuing of commands from the AV construction process.

Among the various property changes of objects during the construction process to signify state changes, we are especially interested in the spatial one. Object movement is an important metaphor that is hard to visualize mentally in an algorithmic procedure and constitutes a vital part of the visualization construction process. Although generating or drawing graphics is already made possible through WIMP or NUI, the spatial manipulation of the said graphics still relies on some coding or commands.

We therefore advocate the direct manipulation of spatial movement in order to simulate the kind of intuitive and natural AV construction process students experience with simple art materials, and import that experience into the computer-based environment. Students should be able to create any graphics by direct sketching, and be able to move these sketched objects on the screen freely by dragging and dropping. Our arguments are presented in the following.

### A more genuine construction experience

The Constructionist learning theory expands on Constructivism and advocates learning through the creation of public entities, or, put trivially, "learning by making" (Papert 1991, p.1). We venture to say that writing or sketching something is intuitively a more immediate and substantive experience of constructing something than typing or clicking on something. Our hypothesis is that the former constitutes a more innate and instinctive human affordance than the latter. Furthermore, when we write out the word *KEY*, there is also more physical movement and cognitive processing involved than just tapping your right middle finger, left middle

finger and then your right index finger in succession to put those three letters onto the computer screen. Analogically, the experience of clicking on a button to automatically generate a graphic of a square is arguably less expressive and memorable than manually traversing a path that starts with a point and goes 2 inches up North before taking a sharp left turn, proceeds another 2 inches before making another 90 degrees left turn to travel down South, and when it reaches the same y-coordinate as the starting point, make one final sharp left turn again in order to move towards the starting point until they finally meet. In short, we think it important to give our students that *hands-on* experience. Literally.

**Less restrictiveness on creative possibilities**

Giving a student colored pens, paper and a pair of scissors, there is practically no limits to what they can come up with for low fidelity algorithm visualizations. Theoretically, they may achieve almost the same result given time and some strenuous programming. To make the AV construction more accessible, users are given a set of limited commands or a few buttons to click on for content generation and manipulation. This approach may very well speed up the construction process, but introduces nevertheless a new limitation. The graphics that can be generated through mouse clicks or commands are limited to the design of the system developer. Even more restrictive are the paths in which generated graphics could move. Lastly, the constellation of different graphics placed in relation to one another is typically confined to a rigid alignment or grid as well.

We see that although we have moved one step closer towards accessibility by utilizing the WIMP interfaces and natural commands, we are also restricting students in what they can do and forcing them to adapt themselves to the AV system. To address this issue, we believe AV systems should take advantage of the NUI and the natural human affordances that will allow students to draw anything anywhere they please and directly manipulate their drawings any way they want.

**Real-time construction and presentation**

A survey (Naps et al. 2002) on why education practitioners are reluctant or unwilling to use algorithm visualization in the classroom shows that among the 29 conference attendees surveyed, 9 out of 10 hold that the amount of time they have to spend on learning new AV tools or developing new visualization content is the biggest barrier to the integration of AV in the curriculum.

One indication of how much preparation and effort is needed for construction the visualization is the degrees in which instructors could use the system on the fly (Ihantola et al. 2005). Clearly, an AV system which instructors could simply fire up and immediately start drawing objects and animating them ad lib would belong to the top echelon of the effortlessness taxonomy proposed by Ihantola et al. (ibid.) The decrease in time to construct AV materials for use in the classroom and the direct manipulation of graphics mean that instructors will be able to carry out an ad lib presentation of real-time visualization in the classroom.

For the students, this approach also translates to lower learning curve and higher motivation. For not only are they able to give free reign to their imagination and creativity to create the kind of algorithm visualization they want, their construction process is a demonstration of the algorithm in real-time.

**In keeping with the evolution of computing devices**

We can safely assume that most existing AV systems are designed for desktops or laptops, where a physical keyboard and mouse is the norm. However, we are witnessing the emergence of a plethora of computing devices that have mobility as their design priority. Compared with the traditional computing devices, the physical and technical properties of Smartphones and Tablets constitute a somewhat different computing experience that is also becoming more prevalent and ubiquitous. One of the most prominent hallmarks of these mobile devices is their lack of a physical keyboard. AV systems that rely heavily on a keyboard or mouse to construct animations may eventually become outdated and atypical. The possibility of using a stylus or finger as an input device also accommodates our proposed setup very well, in which all typing and clicking give way to sketching and dragging and dropping.

**AV also stands for Audio-Visual**

During our still on-going study as we observe students construct and demonstrate their chosen algorithms, it becomes apparent that our low fidelity AV construction process is very much like a show-and-tell session. Students do not just use their hands to make and animate things, they also explain what they are doing

and why they are doing what they are doing in relevance to the algorithm. Despite algorithm *visualization* terminology, we have yet to come across a speechless student whose AV construction and demonstration solely relies on the spectator's visual capacity. In this respect, the lack of voice integration as part of the construction process in current AV systems grows all the more conspicuous. Not only is speaking one of our most powerful human affordances, the *Modality Principle* of the eLearning design guidelines (Clark & Mayer 2008) also suggests that in a multimedia setting, students learn better with animation accompanied by narration than with animation and on-screen texts. In light of these findings, we believe that the spoken explanation by students should be an integral part of the AV construction process, and the support for the recording thereof should be supported by the system to enhance the pedagogical potential.

## Conclusion

In the age of eEverything, where our activities and information become incessantly digitalized, it is easy to lose sight of our inherent human behavioral patterns that should always serve as the best design guidelines for our increasingly computerized environment. As new technological advances keep bringing us new possibilities of compensating for our physical impossibilities, we sometimes forget it is the machine that should adapt to the people and not the other way around.

The use of algorithm visualization in CS education has come a long way pedagogically from the early days of the mere display of instructor prepared computer animated sequences, to today's interactive system, with which students are able to actively construct graphical interpretations of their mental models. Our on-going study allows us to observe how students use simple art supplies to construct visualizations and gives us an insight of what physical activities may need to be translated into the computer-based environment in order to simulate that low tech, low fidelity "hands-on" experience.

By putting a new focus on the human affordances, our work-in-progress aims to improve the AV construction methods, enabling students to sketch and directly manipulate anything they fancy on the computer screen without pre-defined restrictions. Furthermore, their voice narration should also be recorded as an integral part of their visualization construction. By employing the hand movement to create and animate visual artifacts, and by adding our voice narration to reinforce the construction of knowledge, we hope to add a little of that "human touch" to the algorithm visualization construction approach.

## References


Baecker, R. (1981). Sorting Out Sorting. 30 minutes color film created with assistance of Dave Sherman. Dynamic Graphics Porject. University of Toronto.

Clark, R. C., Mayer, R. E. (2008). *e-Learning and the science of instruction: proven guidelines for consumers and designers of multimedia learning.* 2nd ed. San Francisco, CA: Pfeiffer.

Douglas, S., Hundhausen, C. & McKeown, D. (1995). Toward empirically-based software visualization languages. In VL '95: Proceedings of the 11th International IEEE Symposium on Visual Languages. (pp.342-49). Washington, DC: IEEE Computer Society.

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. In *Journal of Visual Languages and Computing*. Volume 13, Issue 3, June. (pp.259-90). Amsterdam: Elsevier

Ihantola, P. et al. (2005). Taxonomy of effortless creation of algorithm visualizations. In *ICER '05: Proceedings of the first international workshop on computing education research.* (pp.123-33). New York, NY: ACM.

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*. (pp. 131-52). New York, NY: ACM.

Papert, S. (1991). Situating Constructionism. In Harel, I., Papert, S. (eds.) *Constructionism: Research reports and essays, 1985-190 by the epistemology & learning research group.* (pp. 1-11). Norwood, NJ: Ablex Pub. Corp.

Stasko, J., Badre, A. & Clayton, L. (1993) Do algorithm animations assist learning?: an empirical study and analysis. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on human factors in computing systems*. (pp.61-6). New York, NY: ACM

## Acknowledgements

We borrowed the line "a little of that human touch" from *Human Touch* by Bruce Springsteen, which appears in his 1992 album of the same title.
We are grateful for the funding provided by the German Research Foundation (DFG) in support of our research project.