Development of XML-based Tools to Support User Interaction with Algorithm Visualizations

Thomas L. Naps
Department of Computer Science
University of Wisconsin Oshkosh
Oshkosh, WI 54901
`naps@uwosh.edu`

Guido Rößling
Department of Computer Science
Darmstadt University of Technology
D-64289 Darmstadt, Germany
`roessling@acm.org`

Abstract

An increasing body of evidence suggests algorithm visualization (AV) is effective only in conjunction with other techniques that force a degree of user interaction beyond the mere "watching" of an algorithm.

One of the difficulties faced by instructors who design visualizations to use in their teaching lies in the time required to develop materials that employ these supplementary techniques. The focus of this working group will be to begin a focused and coordinated effort toward the development of tools designed to remedy this problem. In particular, such tools will be based on a set of XML specifications whose creation will be one of the primary goals of the working group.

1 Background

Algorithm visualization (AV) technology can be used to graphically illustrate various concepts in computer science. However, the instructional effectiveness of using such visualization remains in doubt. A meta-analysis of 21 experiential evaluations of AV systems suggests that a higher level of learner involvement can improve learning outcomes [3].

In an attempt to formalize some of these results, the 2002 ITiCSE working group on "Improving the Educational Impact of Algorithm Visualization" [6] developed a framework for conducting experimental studies of visualization effectiveness. Central to this framework was a taxonomy of learner engagement with the visualization technology. This taxonomy defines six different forms of learner engagement with visualization technology:

1. No viewing (no visualization technology is used at all)
2. Viewing
3. Responding
4. Changing
5. Constructing
6. Presenting

"Viewing" can be considered the core form of engagement, since all other forms of engagement with visualization technology fundamentally entail some kind of viewing. Viewing can also include pace or direction control, and may use multiple windows or embedded explanations. Viewing by itself is the most passive of the forms of engagement. The remaining four categories all include viewing, but then go beyond it in what is expected of the learner.

The third category in the engagement taxonomy is "Responding". The key activity in this category is answering questions concerning the visualization presented by the system. For example, learners might be asked:

- "What will the next frame in this visualization look like?"
- "What source code does this visualization represent?"
- "Is the algorithm shown in this visualization free of bugs?"

"Changing" entails modifying the visualization. The most common example is allowing the learner to change the input of the algorithm under study in order to explore the algorithm's behavior in different cases.

"Constructing" requires learners to construct their own visualizations of the algorithms under study. The most common construction technique is to have learners develop a program that implements an algorithm and then annotate the algorithm with commands to produce an animation – similar to the fashion in which one would identify key events at which to set a breakpoint in a debugging environment.

Finally, "Presenting" requires learners to explain the visualized algorithm to an audience, needing a deeper understanding of the algorithm and an effective way of explaining it.

2 Goals, Methodology, and Activities

Adding these user engagement features to algorithm visualizations is presently hard, time-consuming work. That detracts, in two ways, from progress in using AV. First, many interesting AV systems continue to see little use as actual learning environments because their developers do not have the time to add the user engagement features that would make them effective. Second, carrying out an effectiveness study with respect to the categories in the engagement taxonomy requires that the AV tool has that mode of engagement built into it. Otherwise, considerable time must be spent developing the software add-ons to support that mode of engagement as a prelude to actually doing the study.

Research in the effectiveness of AV could be tremendously accelerated if certain aspects of constructing visualizations and the pedagogic engagement techniques to support them became portable across multiple visualization systems. Toward that end, this working group will begin the process of creating a set of design specifications for such portability tools. The tools will facilitate sharing the following types of resources across different AV systems:

Graphical primitives that are used for constructing the graphics of the visualizations themselves in the form of:

- Data structure specifications that the AV system would then be responsible for rendering – e.g. trees, graphs, and so forth. This is the style of primitives used in AV systems such as *GAIGS* [5] and *JAWAA* [1].
- Graphic primitives particularly amenable to rendering algorithms. This is the style used in script-based AV systems such as ANIMAL [7] and *JSamba* [9].

Hypertext documents that supplement the visualization with various forms of text to be read by the learner in conjunction with watching the visualization. These could include dynamically updated Web pages that present explanations of the algorithm in a fashion that is aware of the values currently being manipulated by the visualization.

Interactive questions ask the viewer to predict the next action taken by the algorithm. The questions typically take the form of true-false, fill-in-the-blank, multiple-choice, and multiple-selection (similar to multiple-choice, but allowing the selection of more than one answer). Questions can be prepared in advance or dynamically generated.

To aid instructors in leading their students toward discovery-based learning, questions may optionally have "hints" that are offered when a student's initial response to a question is wrong. This facilitates the question system's becoming an "intelligent tutor" that could gradually guide the student toward an understanding of the algorithm.

For instructors who want to monitor the use being made of AV systems by their students, it would also be convenient to have specifications for recording students' responses to interactive questions in a database.

A prototypical implementation of a tool-independent interaction support component is already available [8]. However, it still needs to be evaluated and probably adapted to other systems.

Interactive input generators support user construction of input provided to the algorithm being visualized. For sorting algorithms these input generators may allow options such as randomly generated data, data in order to start with, data in reverse order to start with, data "nearly sorted" to start with, and data completely specified by the user. For graph algorithms, such an input generator may take the form of a graph editor by which the user may construct the graph to be acted upon by the algorithm being viewed.

Content generation libraries aim to aid in constructing new visualizations. For example, a class library may visualize data structures using the appropriate graphical primitives described above. This can help programmers to see what their program does, and make generating visualizations (much) easier.

Alternatively, members of the group can examine the area of simulation exercises [4]. Storyboard systems such as *ALVIS* [2] and drag-and-drop based systems such as ANIMAL [7] also promise easy and quick content generation. This promise can be evaluated within the working group.

To ensure portability across different AV systems, each of the support tools will present its associated data to the AV systems in XML form. The working group therefore also has to design an XML language for describing the components. After ITiCSE 2005, a team of participants shall begin implementing a parser for this language that converts the XML into an object tree. This object tree can then be encoded in an appropriate way for the set of supported tools. Instead of having *n* implementations of the same XML parser, we work towards one parser and a set of AV system "plug-ins" using this shared parser, making the adoption of the shared XML format easier and less time-consuming for each AV system developer.

In electronic communication before the working group convenes in Portugal, members will attempt to develop consensus on what types of interaction tools are needed. The list provided above is only meant as a starting point for discussion. We fully expect that other researchers may see the need for different tools. Therefore, pre-conference interaction will be intended to "flesh out" the list provided above.

During ITiCSE 2005, the group will develop detailed design specifications for these tools. These design specifications will be developed first in the form of XML-based data type specification and then in the form of Java class interfaces that could support the delivery of such XML documents to AV systems developed in Java. After members of the working group return to their home institutions, it is hoped that many of them will begin the development (either by their own efforts or those of their research students) of the Java classes that meet the interface specifications that appear in the working group report.

We also hope that many members of the working group will be interested in re-convening in 2006 to discuss the progress of their work in what we hope becomes a loosely connected software development project. More specifically, this re-convening would allow us to bring better closure to the software development efforts that will have taken place in the intervening year. Hopefully, these efforts will have progressed enough that the 2006 iteration of the group could make considerable progress in efforts to deploy these class libraries across multiple AV systems and to disseminate the results of such work.

3 Qualifications of the co-chairs

Tom Naps received the PhD in Mathematical Logic from the University of Notre Dame in 1975. Since then he has taught a broad range of mathematics and computer science courses, first in the University of Wisconsin Center System (1975-81), then at Lawrence University (1981-2001), and now at the University of Wisconsin - Oshkosh.

Since 1987 he has pursued AV both from the perspective of an instructor who wants to design visualizations of particular algorithms to help his students and as the developer of the GAIGS and JHAVÉ AV systems. Naps has written twelve papers in the area of AV, conducted workshops on AV under the NSF's Undergraduate Faculty Enhancement Program (1991), conducted a workshop on AV at the 1992 ACM SIGCSE Technical Symposium, conducted a tutorial on Java-based AV at the 2000 ITiCSE conference, and co-chaired previous international working groups on visualization at recent ITiCSE conferences. Over sixty faculty members at other institutions have used his AV systems. In developing GAIGS and JHAVÉ, he has worked with over twenty undergraduate research assistants. He has collaborated with John Stasko of Georgia Tech and Guido Rößling of the Darmstadt University of Technology to incorporate their scripting languages (Samba and Animal respectively) into the JHAVÉ environment. He is currently working with Scott Grissom (Grand Valley State University) and Myles McNally (Alma College) under a three-year National Science Foundation grant to develop instructional materials to support AV.

Guido Rößling received the Diploma in Computer Science from the Darmstadt University of Technology, Germany, in 1996. From 1996 to 2001, he worked as a research assistant at the University of Siegen, Germany. He finished his Ph.D. thesis on AV system design in 2002. In November 2001, he joined the Darmstadt University of Technology as a research assistant for e-learning applications.

Since 1998, he has developed the extensible AV system ANIMAL that is now also used in Naps' JHAVÉ system. He has published his research on e-learning applications since 2000. This includes several conference papers and journal articles on AV. He was a member of the program chair for the 2002 and 2004 Program Visualization Workshop, held in conjunction with ITiCSE 2002 and 2004. He is also part of the editorial board for building an extensive Web-based AV repository.

4 Potential Participants

All of the following researchers have been contacted to provide feedback on our proposal. All have indicated an interest in participating in the gorup if their time and financial constraints allow them to do so:

- Jay Anderson, Franklin and Marshall College, Lancaster, PA
- Scott Grissom, Grand Valley State University, Allendale, MI
- Chris Hundhausen, Washington State University, Pullman, WA
- Duane Jarc, University of Maryland University College, MD
- Lauri Malmi, Helsinki University of Technology, Helsinki, Finnland
- Andres Moreno Garcia, University of Joensuu, Joensuu, Finnland
- Nico Myller, University of Joensuu, Joensuu, Finnland
- Susan Rodger, Duke University, Durham, NC
- David Stratton,Ballarat University, Ballarat, Australia

References

[1] Akingbade, A., Finley, T., Jackson, D., Patel, P., and Rodger, S. H. JAWAA: Easy Web-Based Animation from CS 0 to Advanced CS Courses. In *Proceedings of the 34th ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2003), Reno, Nevada* (2003), ACM Press, New York, pp. 162–166.

[2] Hundhausen, C. D., and Douglas, S. SALSA and ALVIS: A Language and System for Constructing and Presenting Low Fidelity Algorithm Visualizations. *IEEE Symposium on Visual Languages, Los Alamitos, California* (2000), 67–68.

[3] Hundhausen, C. D., Douglas, S. A., and Stasko, J. T. A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages and Computing 13*, 3 (2002), 259–290.

[4] Korhonen, A., and Malmi, L. Taxonomy of Visual Algorithm Simulation Exercises. In *Proceedings of the Third Program Visualization Workshop, University of Warwick, UK* (July 2004), pp. 105–111.

[5] Naps, T. L., and Bressler, E. A multi-windowed environment for simultaneous visualization of related algorithms on the World Wide Web. 29th *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '98), Atlanta, Georgia* (mar 1998), 277–281.

[6] Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., and Velázquez-Iturbide, J. Á. Exploring the Role of Visualization and Engagement in Computer Science Education. *ACM SIGCSE Bulletin 35*, 2 (June 2003), 131–152.

[7] Rößling, G., and Freisleben, B. ANIMAL: A System for Supporting Multiple Roles in Algorithm Animation. *Journal of Visual Languages and Computing 13*, 2 (2002), 341–354.

[8] Rößling, G., and Häussge, G. Towards Tool-Independent Interaction Support. In *Proceedings of the Third Program Visualization Workshop, University of Warwick, UK* (July 2004), pp. 99–103.

[9] Stasko, J. Smooth Continuous Animation for Portraying Algorithms and Processes. In *Software Visualization*, J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, Eds. MIT Press, 1998, ch. 8, pp. 103–118.