W

Web-Based Algorithm and Program Visualization for Education

Cristóbal Pareja-Flores
Universidad Complutense de Madrid, Spain

Jaime Urquiza-Fuentes
Universidad Rey Juan Carlos, Spain

J. Ángel Velázquez Iturbide
Universidad Rey Juan Carlos, Spain

INTRODUCTION

Programming is a central activity in the computing profession. It is facilitated by different tools (editors, compilers, debuggers, etc.), which are often integrated into programming environments. Programming also plays a central role in computer science education. For this purpose, a number of complementary tools were developed during the last decade: algorithm animators, program visualizers, problem generators, assignment graders, and so forth.

After the Web explosion, teachers of programming rapidly turned their attention to the Web. Although the Web has speed and power limitations, it also has several advantages that make it invaluable for educational purposes. Mainly, it provides universal accessibility and platform independence, and solves the distribution problem by always making available the last version of any tool.

Probably, the most common use of the Web for programming courses is as a communication medium, facilitating submission and administration of assignments and grades (Burd, 2000). Another common use of the Web for programming education is as a public repository of high quality problems, such as the *Lab Repository* (Knox, 2006) and the *ACM International Collegiate Programming Contest* (Skiena & Revilla, 2003). Web sites may also host other resources, such as slides and audio lectures (Skiena & Revilla, 2003), algorithm animations (Brummond, 2001), or programming tools (English, 2001). These collections have no structure or, at best, are lineally or hierarchically structured, but more advanced repositories are possible. In this case, a management system must be delivered that, using (semi)structured mark-up languages, allows retrieving, maintaining, and publishing. A good representative is the eXercita system (Gregorio-Rodríguez et al., 2000, 2002). Finally, programming tools have been ported to be executed on the Web (Pareja-Flores & Velázquez-Iturbide, 2002).

This article describes a different class of Web-based tools for programming education, namely tools for algorithm and program visualization. After the Background section, we describe the evolution of these systems, educational uses, and lessons learned. Finally, we outline future trends in the use of the Web for programming education and our personal conclusions.

BACKGROUND

Algorithm animation is a research field that is now 20 years old and still evolving. There is a consensus with respect to the videotape *Sorting out Sorting* presented in 1981 by Baecker (1998), which is considered a landmark on animation. It included animations of nine sorting algorithms. Afterward, some works established the main techniques for specifying and implementing algorithm animation: BALSA (Brown, 1988), Tango (Stasko, 1990), and Pavane (Roman, Cox, Wilcox, & Plun, 1992). Systematic categorizations of software visualizations have been proposed since then (Price, Baecker, & Small, 1998).

In a general setting, software visualization studies the visual representation of software entities (Stasko, Domingue, Brown, & Price, 1998). Visualization requires an effort to abstract the target entity to visualize and to make a good graphical design that may yield many different representations: text versus graphics, level of abstraction, displaying control versus data structures, static versus dynamic visualizations, one or multiple views, behavior versus performance, and so forth. This general aim can be achieved in different ways. Program visualization is aimed at the visualization of a piece of software so that the representation has a close relationship to the source code. Algorithm animation is aimed at the dynamic visualization of a piece of software illustrating the main ideas or steps (i.e., its algorithmic behavior), but without a close relationship to the source code. The graphical nature of software visualization in general and algorithm animation in particular makes them very conductive to the hypermedia features of the Web.

WEB-BASED ALGORITHM AND PROGRAM VISUALIZATION SYSTEMS

In the mid 1990s, many of the algorithm animation systems were ported to the Web, and many additional systems were specifically designed for the Web. A representative work from these years is that of Naps (1996), in which he carried out a study of the technical alternatives that could be used to make animations produced by previous systems available on the Web. Other pioneer systems are Mocha (Baker, Cruz, Liotta, & Tamassia, 1996), JCAT (Brown & Raisamo, 1997), and Jeliot (Haajanen et al., 1997).

Some animation systems are general purpose, so they can be used to generate visualizations of any entity, including non-programming entities. They typically provide a scripting language that makes animation creation a relatively easy task. Good representatives are the ANIMAL (Roessling & Freisleben, 2002), JAWAA (Pierson & Rodger, 1998), and Samba (Stasko, 1997) systems. The JHAVÉ system (Naps, Eagan, & Norton, 2000) is not an algorithm visualization system itself, but rather a support environment to render visualizations from a variety of visualization systems. Currently, it supports algorithm visualizations in three scripting languages: ANIMAL, GAIGS (Naps, 1990), and Samba. With JHAVÉ, students can explore algorithms, controlling movement and responding to pop-up questions. Leonardo Web (Bonifaci, Demetrescu, Finocchi, & Laura, 2003) is a collection of tools for creating animated representations via the use of a visual editor and a Java library. The presentations can be viewed by a simple Java player, so being possible to integrate the animations in Web pages, or to browse them off-line.

Another category of systems automatically produce program animations, enhanced with graphical representations. Some of these systems are computer-supported learning systems that only provide visualization support for a given domain and with a specific didactic purpose. For instance, the KIEL system (Berghammer & Milanese, 2001) gives support to learning the ML functional language. The system allows students to input a program and an expression to evaluate and then graphically displays the evaluation of the expression. The student may navigate both forward and backward through the evaluation process. A successful experience comes from the "problets" by Kumar (2003). They are tutors aimed at specific programming topics, which randomly generate problems and ask questions to the student. Visualizations are used as a visual aid for problem solving. Problets have been developed for a number of topics: expression evaluation, identifier scope, and so forth. A similar approach has been addressed elsewhere for data structures (Baker, Boilen, Goodrich, Tamassia, & Stibel, 1999).

A different category of systems providing automatic visualizations are based on the availability of complete language processors. The WinHIPE system (Velázquez-Iturbide, Pareja-Flores, & Urquiza-Fuentes, 2006) is an integrated development environment (IDE) for functional programming that allows (semi)automatically generating animations of the evaluation of a functional expression. Then, the animation can be played within the IDE or exported to the Web (Urquiza-Fuentes & Velázquez-Iturbide, 2005), also giving support to manage collections of animations.

Other automatic visualization systems use the Web as a user interface for programming, and allow loading and executing programs. For instance, the ISVL is designed to learn Prolog programming online (Domingue & Mulholland, 1998). ISVL allows students to generate animations based on AND-OR trees that can be communicated to the course tutor and even stored as films. The tutoring may insert annotations on the student animations to respond to their questions or to explain where a programming error was made.

The Jeliot family (Ben-Ari, Myller, Sutinen, & Tarhio, 2002) is a collection of program and algorithm visualization tools designed for novices learning programming, algorithms, and data structures with Java. The last member of the family is Jeliot 3 (Moreno, Myller, Sutinen, & Ben-Ari, 2004), specially focused to animate object oriented features, such as inheritance.

The last class of animation systems is based on simulation. MatrixPro (Karavirta, Korhonen, Malmi, & Stalnacke, 2004) enables instructors to create algorithm animations by direct manipulation of a data structures library. Teachers can demonstrate the visual simulation of an algorithm by providing on the fly different input sets. "What-if" questions can also be invoked in lectures. A related tool, TRAKLA2 (Korhonen, Malmi, & Silvasti, 2003), is designed for building and solving interactive algorithm simulation exercises, made available as applets, for learning data structures and algorithms. In these exercises, students directly manipulate conceptual visualizations of data structures in order to simulate the working of algorithms.

EDUCATIONAL USES OF WEB-BASED ALGORITHM AND PROGRAM VISUALIZATION

From the outset, the main use of algorithm animation was educational, rather that industrial (for instance, as a debugging tool). Algorithm animation systems have been used in several ways: as a complement to lectures on algorithms, for self-study, or within laboratories. A more demanding use of animation systems consists in requiring students to build their own animations.

The best documented experience ran for about 20 years at the Computer Science Department of Brown University (Bazik, Tamassia, Reiss, & van Dam, 1998). All the experiences reported in the available literature agree that students