Effortless Construction and Management of Program Animations on the Web

Jaime Urquiza-Fuentes and J. Ángel Velázquez-Iturbide

Departamento de Informática, Estadística y Telemática, Universidad Rey Juan Carlos, C/ Tulipán s/n, 28933 Móstoles, Madrid, Spain
{jaime.urquiza, angel.velazquez}@urjc.es

Abstract. We describe an extension of a programming environment to generate web-based program animations. Emphasis is put on requiring little effort from the instructor that handles the system. User interaction is reduced to a minimum, mostly for customizing the animations. Both construction and maintenance are considered in order to guarantee low effort in an actual educational scenario. We describe several aspects of a program animation: the different kinds of information that compose it, its construction process, alternative graphical designs for web publishing, and its implementation as a package. In general, the instructor will wish to use the system to construct and handle a collection of animations for one or several courses. Therefore, we also consider the creation and management of collections of animations in a effortless way. Finally, we describe our experience as well as related work.

1 Introduction

Software animations are used in computer science education since the early eighties [1]. They can be used in different scenarios [2]: passive or interactive lectures, laboratories, self-study, etc. In spite of their educational potential, they have not been incorporated into the mainstream of computer science education. The report of a working group organized at the ITiCSE 2002 conference contains information about three surveys made on educational use of visualizations [3]. The "preconference survey" contains detailed information about the factors that make the respondent or respondent's colleagues reluctant or unable to use animations. The options can be grouped into factors related to time necessary to prepare infrastructure (e.g. to install the software), time necessary to develop animations, and quality and adequacy of the tools to the course (e.g. to adapt animations to the teaching approach of a course).

Another working group [4] focused on how to reduce the effort to use animations in education. It identifies sources of workload for the instructor as well as advices to facilitate the adoption of animation systems. In particular, the authors argue for platform independence and give much importance to their dissemination among the education community. Both factors are typically enhanced through the Web.

We have developed an effortless approach to building and maintaining program animations [5], [6]. In summary, the integrated development environment (IDE) WinHIPE was extended to automatically generate static visualizations that the user can later customize to automatically constitute an animation. Animations are discrete, i.e. they are composed of a sequence of static visualizations or snapshots.

In this paper, we extend our framework to deal with Web-based animations. We are mainly concerned with providing the instructor an effortless tool. Within this framework, two different issues are addressed. Firstly, we deal with the problem of generating, using and maintaining individual animations. Secondly, we deal with the handing of collections of animation, i.e. their organization and web publishing. Afterwards, we describe our experience and discuss related work.

2 Web Animations

In this section we deal with the contents, construction process and graphical design of Web animations. Publishing algorithm animations on the Web is a topic that other authors addressed in the past [2]. Some web-based animation systems are applets designed ad-hoc to visualize particular algorithms [7], [8], [9], [10]. Other systems are more flexible, allowing the user to generate his/her own animations [11], [12], [13], [14], [15], [16], [17], [18]. In this paper we refer to the latter category.

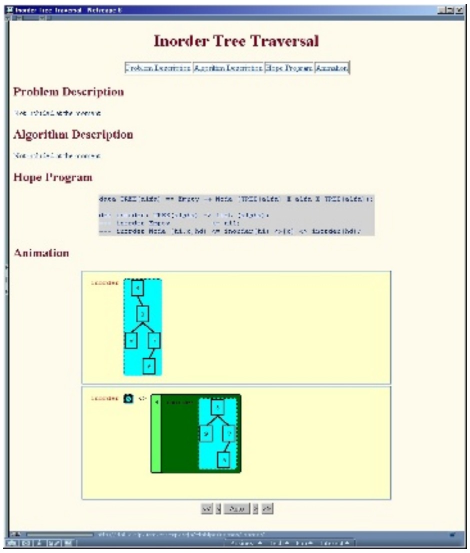


Fig. 1. A web animation as generated in a previous version of our system

In a preliminary work, we addressed the automatic generation of Web-based animations within the WinHIPE IDE [15]. The outcome was a dynamic web page with the structure of a lesson on a particular algorithm. It consisted in four sections (see Fig. 1). The two first sections described the problem statement and an algorithm solving the problem. These sections were generated with a dummy comment and the user had the responsibility of editing them afterwards to fully explain them. The third and fourth sections contained the source code of the program to animate and the animation itself, respectively.

Text contents and the graphical design of the web page could be changed with a web editor. However, any change in the animation had to be made from scratch. It does not require much effort but the user is not assisted in mundane tasks: looking for the program and expression files used as well as keeping the same customization information (expressions selected and graphical design of visualizations).

Our main concern is providing the instructor with a tool that allows him/her constructing web animations in an effortless way. Consequently, we have redesigned and reimplemented our tool. The contents of web pages and the internal construction process of web animations were modified. Finally, we also enriched the graphical design of web animations to improve their usability and flexibility.

2.1 Contents of Web Animations

Contents of web-based animations have been modified to provide a more friendly modification process. In effect, a web animation now contains all the information necessary to change it, either with respect to its look, or the contents of the textual sections or even the animation itself. Look information describes the style and layout of the web page that contains the animation. Textual contents include information about the problem statement and the algorithm used to solve it. Animation contents contain information to play and rebuild the animation: source code, evaluated expression, list of snapshots that form the animation, and configuration information describing the typographic features of visualizations.

In general, contents information can be classified into two classes. On the one hand, visible information can be processed and displayed by any web browser: HTML

Table 1. Different classes of contents and of information

Info. \ Cont.	Textual contents	Animation contents	Look contents
Maintenance information	XML elements describing the title and the problem and solution sections	XML elements describing source and expression code, the list of snapshots, and typography of visualizations	XML elements describing web page style
Visible information	HTML elements describing the problem and solution sections	HTML elements for source and expression code, the list of snapshots, and typography of visualizations	CSS style sheet

code, CSS information and images. On the other hand, maintenance information is necessary to modify or handle animations, and it is structured by means of XML. Given the lack of space, we do not describe here its DTD.

We want to remark that web animations are two-fold reusable. On the one hand, they are reusable by the user because a web animation now contains all the information necessary to modify it. On the other hand, they are reusable in different applications or platforms because their implementation is based on XML. Consequently, not only can they be used by the WinHIPE IDE but also by any application supporting XML.

To sum up, the contents of new web animations reduce the user effort to modify and maintain them because he/she no longer has to remember any kind of configuration or contents information. They also provide user and application reuse.

2.2 Construction and Maintenance Process

The construction and maintenance of web animations are divided into two phases: generation of the program animation and generation of the web animation (see Fig. 2).

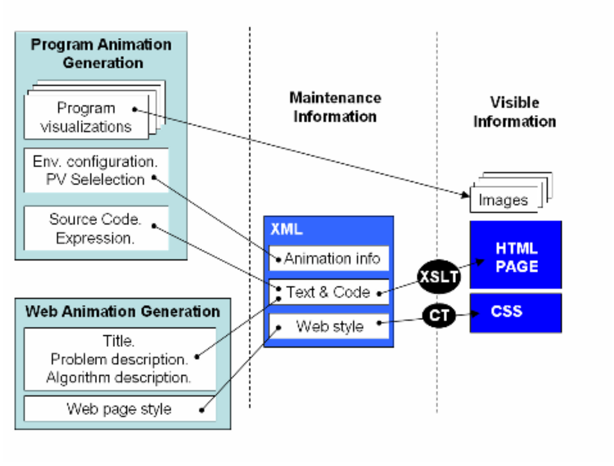


Fig. 2. Construction/maintenance process of web animations

In broad terms, the phase of program animation generation consists in the following steps: edition and compilation of the program, edition and evaluation of the expression and selection of the snapshots that will form the animation. The interested reader in a more detailed explanation of these steps can refer elsewhere [5], [6]. Maintenance information corresponding to animation contents is created here.

The web animation is also generated through the user interface provided by the WinHIPE IDE. Now, the user must provide maintenance information corresponding to textual and look contents.

Notice that user interaction is limited to building the program animation, explaining the problem and its solution, and specifying the web look of the animation. Any additional processing is automatic.

A final, automatic step generates all the visible information other than snapshots. By means of an XSL transformation (XSLT), HTML code is built (distributed in one or several HTML documents, as explained in the next subsection) corresponding to visible information of textual contents and the animation. The tags of this code are represented using a style sheet CSS that is built by means of a custom transformation (CT) of the maintenance information corresponding to the look contents.

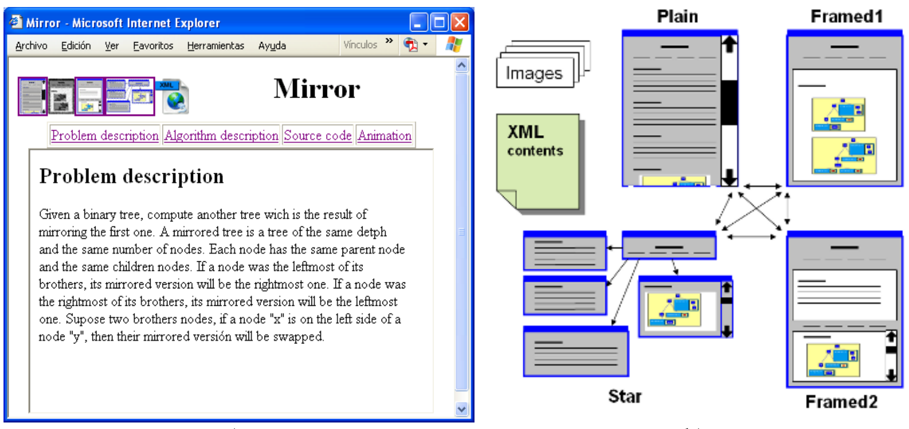
2.3 Graphic Design of Web Pages

In our previous work [15], web-based animations consisted in an HTML page navigable by means of a local index and the scroll bar, where the textual contents and the animation were shown (see Fig. 1).

We have developed three new graphical designs. Two of them are inspired by general principles of web pages usability [19] (we call them Framed1 and Framed2), and the third one is intended to enhance flexibility of user interaction (we call it Star). The user may choose one of the four graphical designs by means of a navigation bar placed in the upper left corner (see Fig. 3a). The navigation bar also provides access to XML contents of the web animation.

The main usability requirements considered were: minimizing the amount of hidden information on the screen, adjusting navigation possibilities to a maximum of 20% of screen space, working the predictable response times (given that they depend on the number and size of snapshots), use of linked CSS and inline frames.

We show here a sample of the Framed1 graphical design (see Fig. 3a) as well as a schematic version for the other designs (see Fig. 3b). It uses an inline frame, and



Problem description

Given a binary tree, compute another tree wich is the result of mirroring the first one. A mirrored tree is a tree of the same detph and the same number of nodes. Each node has the same parent node and the same children nodes. If a node was the leftmost of its brothers, its mirrored version will be the rightmost one. If a node was the rightmost of its brothers, its mirrored version will be the leftmost one. Supose two brothers nodes, if a node "x" is on the left side of a node "y", then their mirrored versión will be swapped.

a) b)

Fig. 3. a) A web animation of type Framed1. b) Elements of a Web animation package allows the user accessing each section of the web animation with one single mouse click, leaving the use of the scroll bar (hidden information) just for those cases with contents of the section larger than space available on the screen. The title and links to the sections always are visible in the screen.

2.4 A Web Animation Package

A package of a web animation consists of all the maintenance information coded as an XML document, the snapshots that form the animation, and the web pages corresponding to the four possible presentation styles of the web animation. A package is stored in a directory. Fig. 4b illustrates this implementation decision.

3 Collections of Web Animations

Typically, an instructor will construct more than one or two animations. Given the effort required to find an animation system, installing it, learning its usage, building animations and performing other minor, mundane tasks, the instructor will wish to use the animation system for more than one topic [4]. Consequently, he/she will build and maintain a collection of animations, including their organization. It is also common to make changes in one or several animations, either with respect to their contents, their presentation style or their classification. In this section we describe support to handle collections of animations.

A collection is formed by a set of web animations about a common theme: a course, a class of problems, etc. The user may generate as many independent collections as he/she wishes. A given collection is internally organized hierarchically. The hierarchy may have an arbitrary depth. Each level of the hierarchy may have an arbitrary number of categories.

The hierarchical structure of collections as well as the nature of their contents drove us to use again XML as the supporting technology.

3.1 Construction and Management of a Web Collection

Constructing a collection typically starts by creating an empty collection. At least, this collection must have one category, which must be given a name. Another typical situation occurs when a new collection is created from one or several already existing categories which have grown as much as to deserve to be a collection by themselves.

Management of a collection consists in managing its hierarchical structure, managing web animations, and managing their look.

Management of the hierarchical structure comprises inserting, removing or modifying categories. When a category is inserted, it is given a name, a short description and its placement within the hierarchy. When a no-empty category is removed, the user is asked what to do with its contents: either deleting or moving them to a different category. Finally, a category may be modified by changing its name, its description or its placement. In all of these operations, name conflicts with the siblings of the category are checked.

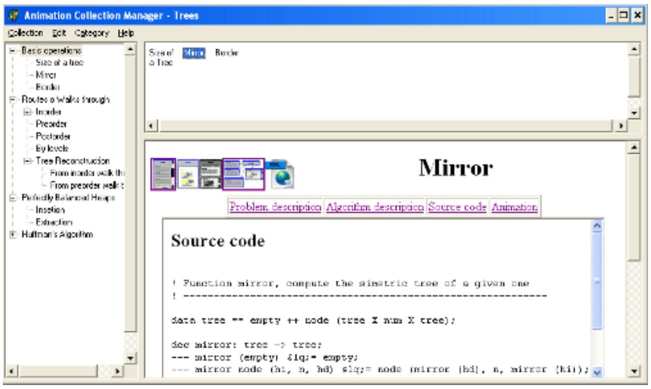
The management of a collection also allows inserting a new animation, removing it (in this case, it can be physically deleted or it can be moved to another category), or even modifying a single animation (as explained in section 2.2).

Each category may have associated a given look, giving greater homogeneity to the collection. This association can be applied to all the animations of a category and also to its descendants. We also allow giving a particular look to an animation, independently from the look of its corresponding category. Finally, changes in the structure of a collection prompt the user to choose between the source look and the look of the target category, if it exits.

These operations are performed with a simple and intuitive user interface. It is similar to the directory interface of Windows Explorer (see Fig. 4). The left panel shows the hierarchical structure of the collection. There are also two panels at the right, where the upper panel shows the contents of the category selected at the left (sublevels and animations) and the lower panel allows showing and modifying the animation selected in the upper right panel.

3.2 Web Publishing of a Web Collection

Web publishing of a collection consists in transforming the internal, hierarchical XML structure into a web site by means of XSL transformations. A root page is created with a description of the collection and links to all the categories at the first level. One page is created for each category, containing the category title, its description and links to its associated animations and daughter categories. A map of the collection allows accessing directly to any category in the collection. Navigation within the collection is further facilitated by making available to all the pages in a category links to its father category, root page and to the collection map.



Mirror

Source code

```
! Function mirror, compute the simetric tree of a given one
! -------------------------------------------------------------

data tree == empty ++ node (tree X num X tree);

dec mirror: tree -> tree;
--- mirror (empty) &lq;= empty;
--- mirror node (hi, n, hd) &lq;= node (mirror (hd), n, mirror (hi));
```

Fig. 4. User interface of the collection manager

4 Experience

We have just concluded the development within the WinHIPE IDE of the new tool for web-based animations. Therefore, we have limited experience with its usage in the classroom. Currently, we have only been able to use some animations during the first semester of the current academic year in a course on algorithms. In particular, we have displayed four animations on integers and sorting. For the next academic year, we plan to make a more extensive use of web animations. More importantly, we will be able to use the collection facilities to create a collection for such a course, develop web-based animations, and structure them into a hierarchy of categories.

We also plan to use and evaluate web animations and collections in the second semester of the present year in a programming languages course with a strong bias to functional programming [20]. We evaluated WinHIPE animations [6], but not their web version, with respect to "effortlessness" during a laboratory session. The results obtained were that students consider construction, use and maintenance of animations easy to learn and effortless tasks. Thus, they used them extensively for debugging.

We plan to evaluate web-based animations in different ways, depending on the education actor. With respect to teachers, we will track their success building and managing both web animations and the collection that contains them. With respect to students, we plan to evaluate their activity as follows. A number of animations, covering most of the topics in the course, will be available during the course and their usage will be monitored. During the last laboratory session, students will have to accomplish two tasks. First, they will answer a questionnaire about the animations available during the course. Afterwards, they will be asked to build several web animations and they will answer a questionnaire about the building process.

5 Related Work

As far as we know, the term "effortless" was used for the first time to describe the anonymous, non-web system by LaFollette et al. [21]. Effortless creation of animations (or simply "effortlessness") is a fuzzy term. Therefore it is too subjective to become an accurate measure. Korhonen et al. [22] tried to deepen into the understanding of effortlessness by estimating it for several animation systems and relating their results to several categories. They identified generality and presentation style as the categories (extracted from the taxonomy by Price et al. [23]) most relevant to effortlessness. However, more research is required to understand "effortlessness". A promising direction consists in identifying the approaches most widely used in effortless systems and studying how they manage to reduce workload to the user.

There is not much work on collections of web-based algorithm animations. There are repositories of teaching materials for computer science courses where we often find animations. These repositories are not structured or they are simply structured in several topics. An example of these repositories is SIGCSE Education Links [24]. It contains over eighty resources, that can be searched by keywords or browsed by several independent keys (all, by course, resource type, author, recent submissions, and most popular). Visualizations are one resource type containing 13 contributions.

The Computer Science Teaching Center [25] is another repository of CS laboratories. They should be high-quality laboratory materials to be shared by the community of CS educators. It contains 105 resources that can be browsed by category and review status. The category of visualization contains 35 resources.

Other repositories are specialized to algorithm visualizations and animations. The Complete Collection of Algorithm Animations [26] is probably the most popular, although other authors have also developed their own. They typically have a very simple structuring based on several topics (e.g. tree or sorting algorithms).

A final category of repositories related to web-based animations are repositories of exercises. They may be highly structured and therefore based on data bases or mark-up languages. However, they only contain static visualizations of exercises, so their utility is very limited in our context. Examples of these systems are eXercita [27] and SAIL [28], both of them based on LaTeX.

The emphasis in any of these categories is very different from ours. General repositories seek making available educational resources to instructors. They are typically supported by organizations or institutions. Algorithm repositories have the same aim but focused on animations. They are web sites developed by individuals. In both cases, the emphasis is on giving a simple structuring to facilitate searching and browsing, but they cannot be used freely by instructors. Repositories of exercises are typically developed by teams, and structuring is given much more importance. Exercises can be handled in more flexible ways, such as generating different instances of a given exercise or customizing an exercise for publication. They are research tools where individuals may make a big effort to yield polished exercises.

6 Conclusions

We have described an extension of a programming environment to generate web-based program animations. The emphasis is put on requiring little effort to the instructor that handles the system. Thus, user interaction is reduced to a minimum, and both construction and maintenance are considered. Not only single animations were considered but also the creation and management of collections of animations.

This work focuses on the effort required to the instructor. However, it is just one of the different issues that refrain educators and learners from using animations more widely. We expect to see improvements in different directions that will foster more wide usage of animations for computer science education.

Acknowledgments

This work is supported by the research project TIN2004-07568 of the Spanish Ministry of Education and Science.

References

1. Baecker, R., Price, B.: The early history of software visualization. In: Stasko, J.T., Domingue, J., Brown, M.H., Price, B.A. (eds.): Software Visualization. MIT Press, Cambridge MA (1998) 29-34
2. Pareja-Flores, C., Velázquez-Iturbide, J. Á.: Program execution and visualization on the Web. In: A. Aggarwal (ed.), Web-Based Learning and Teaching Technologies: Opportunities and Challenges. Idea-Group Publishing, Hershey, PA (2002) 236-259
3. Naps, T., et al: Exploring the role of visualization and engagement in computer science education. SIGCSE Bulletin 35(2) (2003) 131-152
4. Naps, T., et al: Evaluating the educational impact of visualization. SIGCSE Bulletin 35(4) (2003) 124-136
5. Naharro-Berrocal, F., Pareja-Flores, C., Velázquez-Iturbide, J.Á.: Automatic generation of algorithm animations in a programming environment. Proc. 30th ASEE/IEEE Frontiers in Education Conf. Stiples Publishing (2000) S2C 6-12
6. Velázquez-Iturbide, J.Á., Pareja-Flores, C., Urquiza-Fuentes, J.: An approach to effortless construction of program animations. Under review
7. Dershem, H.L., McFall, R.L., Uti, N.: Animation of Java linked lists. Proc. 33rd SIGCSE Technical Symp. Computer Science Education. ACM Press, New York (2002) 53–57
8. Hadlock, F., et al.: An Internet based algorithm visualization system. Journal of Computing Sciences in Colleges 20(2) (2004) 304 - 310
9. Najork, M.: Web-based algorithm animation. Proc. 38th Conf. Design Automation (2001) 506-511
10. Stern, L., Søndergaard, H., Naish, L.: A strategy for managing content complexity in algorithm animation. Proc. 4th Annual SIGCSE/SIGCUE Conf. Innovation and Technology in Computer Science Education. ACM Press (1999) 127-130
11. Akingbade, A., Finley, T., Jackson, D., Patel, P., Rodger, S.H.: JAWAA: Easy web-based animation from CS0 to advanced CS courses. Proc. 34th SIGCSE Technical Symp. Computer Science Education. ACM Press (2003) 162-166
12. Ross, R.J., Grinder, M.T.: Hypertextbooks: Animated, active learning, comprehensive teaching and learning resources for the Web. In: Diehl, S. (ed.): Software Visualization. LNCS, Vol. 2269. Springer-Verlag, Berlin Heidelberg (2001) 269-283
13. Esponda Argüero, M., Rojas, R.: Learning algorithms with an electronic chalkboard over the Web. Advances in Web-Based Learning – ICWL 2004. LNCS, Vol. 3143. Springer-Verlag, Berlin Heidelberg (2004) 1-10
14. Korhonen, A., Malmi, L.: Algorithm simulation with automatic assessment. Proc. 5th Annual SIGCSE/SIGCUE Conf. Innovation and Technology in Computer Science Education. ACM Press (2000) 160-163
15. Naharro-Berrocal, F., et al.: Automatic Web publishing of algorithm animations. Upgrade II(2) (2001) 41-45
16. Naps, T., Eagan, J., Norton, L.: JHAVÉ: An environment to actively engage students in Web-based algorithm visualizations. Proc. 31st SIGCSE Technical Symp. Computer Science Education. ACM Press (2000) 109–113
17. Rossling, G., Freisleben, B.: Program visualization using AnimalScript. Proc. First International Program Visualization Workshop. University of Joensuu Press (2001) 41-52
18. Sutinen, E., Tarhio, J., Teräsvirta, T.: Easy algorithm animation on the Web. Multimedia Tools and Applications, 19(2) (2003) 179-194
19. Nielsen, J.: Designing Web Usability. New Riders Publishing, Indianapolis IN USA, 1999

20. Velázquez-Iturbide, J.Á.: A programming languages course for freshmen. Proc. the 10th Annual SIGCSE/SIGCUE Conf. Innovation and Technology in Computer Science Education. ACM Press (2005) in press
21. LaFollette, P., Korsh, J., Sangwan, R.: A visual interface for effortless animation of C/C++ programs. Journal of Visual Languages and Systems 11 (2000) 27-48
22. Karavirta, V., et al.: Effortless creation of algorithm visualization. Proceedings of the Second Annual Finish/Baltic Conf. Computer Science Education (2002) 52-56
23. Price, B., Baecker, R., Small, I.: A principled taxonomy of software visualization. Journal of Visual Languages and Systems 4, 3 (1993) 211-271
24. ACM SIGCSE:SIGCSE Education Links. www.sigcse.org/topics/ (updated December 2004)
25. The Computer Science Teaching Center. www.cstc.org/ (updated 2003)
26. Brummund, P., Uti, N.V.: The Complete Collection of Algorithm Animations. cs.hope.edu/~alganim/ccaa (updated June 2001)
27. Gregorio-Rodríguez, et al: eXercita: A system for archiving and publishing programming exercises. In: Ortega, M., Bravo, J. (eds.). Computers and Education: Towards an Interconnected Society. Kluwer Academic Press (2001) 187-197
28. Kovourov, S. et al. SAIL: A system for generating, archiving and retrieving specialized assignments using LaTeX. Proc. 31st SIGCSE Technical Symp. Computer Science Education. ACM Press (2000) 300-304