Available online at www.sciencedirect.com
ScienceDirect
Computers & Education 50 (2008) 179–192
COMPUTERS & EDUCATION
www.elsevier.com/locate/compedu

# An approach to effortless construction of program animations

J. Ángel Velázquez-Iturbide [a,*], Cristóbal Pareja-Flores [b], Jaime Urquiza-Fuentes [a]

[a] *Departamento de Lenguajes y Sistemas Informáticos, Universidad Rey Juan Carlos, C/Tulipán s/n, 28933 Móstoles, Madrid, Spain*
[b] *Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid, Avda. Puerta de Hierro s/n, 28040 Madrid, Spain*

**Abstract**

Program animation systems have not been as widely adopted by computer science educators as we might expect from the firm belief that they can help in enhancing computer science education. One of the most notable obstacles to their adoption is the considerable effort that the production of program animations represents for the instructor. We present here an approach to reduce such a workload based on the automatic generation of visualizations and animations. The user may customize them in a user-friendly way to construct more expressive program animations. These operations are carried out by means of a user-friendly manipulation based on the metaphor of office documents. We have applied this approach to the functional paradigm by extending the WinHIPE programming environment. Finally, we report on the successful results of an evaluation performed to measure its ease of use.


*Keywords:* Authoring tools and methods; Evaluation of CAL systems; Improving classroom teaching; Interactive learning environments; Programming and programming languages

## 1. Introduction

Learning environments for computer programming often use visualizations and animations as alternative representations to written programs (Stasko, Domingue, Brown, & Price, 1998). In general, multiple external representations (MERs) are a common feature in learning environments. They increase student motivation and can be designed to serve several learning purposes (Ainsworth, 1999).

There is a firm belief among computer science educators that visualization can make a difference in helping students learn programming concepts more effectively. For instance, in a survey made among participants at the ITiCSE 2002 conference of the ACM, 93% of the 66 respondents agreed or strongly agreed that this was the case, 7% were neutral or had no opinion, and none disagreed (Naps et al., 2003a). However, research on algorithm animation systems has mainly focused on their technical implementation. As a consequence, two notable obstacles (Naps et al., 2003a) to their widespread adoption in education have not been sufficiently

* Corresponding author. Fax: +34 91 488 7049.
*E-mail address:* angel.velazquez@urjc.es (J.Á. Velázquez-Iturbide).

0360-1315/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compedu.2006.04.005

addressed and they now are the foci of attention. Firstly, there is no evidence of their educational benefits. Empirical evaluations of the educational effectiveness of animations (Hundhausen, Douglas, & Stasko, 2002) have led to consensus about the importance of student engagement in obtaining educational success (Naps et al., 2003a). Secondly, the construction of animations requires considerable effort on the part of instructors, to the point of being a bottleneck for the widespread adoption of animation systems. This article focuses on this second factor.

Several formal or informal studies support the importance of instructors' workload. For instance, a well documented experience is reported by Pollack and Ben-Ari (2004), who describe several animation systems and the criteria used to select one of them. Some factors relate to the adequacy of the system to the course or to the control of animations, whereas others relate to the effort required of the teacher: ease of installation, support to easily create demonstrations, support to create data sets to answer questions, and a save/load facility.

In a more general setting, several surveys have been conducted on the educational use of visualizations. The report of a working group organized at the ITiCSE 2002 conference contains information about three surveys concerning the educational use of visualizations (Naps et al., 2003a). The ''preconference survey'' contains more elaborate 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 the time it takes to prepare the infrastructure (e.g. to install the software), the time it takes to develop animations, and the quality and adequacy of the tools (e.g. to adapt animations to the teaching approach of a course). The option most cited as a major impediment was the time it takes to develop animations (by two thirds of the respondents). Karavirta, Korhonen, and Tenhunen (2004) have also recently conducted a more specific survey on development effort, involving 22 respondents. According to their results, the two most common advantages of animation systems were the features allowing users to create animations easily, and sufficient navigation possibilities in the final animation.

A key dimension in the effort problem is the level of abstraction of animations. Program visualization is typically performed automatically (e.g. by pretty-printing the code or by displaying the state of data structures), but the level of abstraction obtained is low. The use of algorithm animations provides a more abstract view that relates to the underlying algorithmic ideas, but their construction is typically very demanding. As effortlessness can only be achieved automatically at the level of program animation, we wondered whether this could be a satisfactory starting point to more expressive animations. This hypothesis is supported by the existence of program animation systems which provide complex animations. For instance, the Jeliot system (Haajanen et al., 1997; Sutinen, Tarhio, & Tërasvirta, 2003) allows program animations to be generated semi-automatically. The system automatically identifies the ''interesting events'' of the animation of a Java program and generates an applet to animate it. The user defines and customizes the views by means of simple dialogs. However, Jeliot exhibits some important drawbacks for educational use. In particular, the lack of a load/store facility directly affects the instructor's workload.

In this article, we describe a new, user-friendly approach to constructing animations requiring minimal effort. Our approach simultaneously allows users to generate visualizations and animations automatically, while also giving them powerful customization facilities. Visualizations and animations are obtained as a side-effect of program execution, so the animation system is embedded in a general purpose programming environment. The user can make program animations more attractive in two ways. Firstly, he/she may customize visualizations to meet his/her visual and typographical preferences. Secondly, and more importantly, the user may choose the relevant parts of a program execution (according to his/her own criteria) to form an animation which is more meaningful than one that would be obtained by simply employing a complete step-by-step or one-step execution. In order to keep user workload low, these facilities are carried out via a simple, user-friendly interactive manipulation, typically by means of menus or dialogs, thus avoiding the need to learn a customization or script language.

We have applied our approach to the functional paradigm. Compared to other paradigms, visualization of functional programs has seldom been addressed; the interested reader may consult a comprehensive survey elsewhere (Urquiza-Fuentes & Velázquez-Iturbide, 2004). From the visualization point of view and given its simplicity in comparison to other paradigms, functional programming offers a unique opportunity to experiment. In particular, our work is based on the programming environment WinHIPE (Velázquez-Iturbide, 1994; WinHIPE, 2006), which offers a view of the evaluation of expressions as term rewriting.