The ANIMAL Algorithm Animation Tool

Guido Rößling, Markus Schüler and Bernd Freisleben
Department of Electrical Engineering and Computer Science
University of Siegen
Hölderlinstr. 3, D-57068 Siegen, Germany
{**roessling, schueler, freisleb**}**@informatik.uni-siegen.de**

Abstract

In this paper, we present ANIMAL, a new tool for developing animations to be used in lectures. ANIMAL offers a small but powerful set of graphical operators. Animations are generated using a visual editor, by scripting or via API calls. All animations can be edited visually. ANIMAL supports source and pseudo code inclusion and highlighting as well as precise user-defined delays between actions. The paper evaluates the functionality of ANIMAL in comparison to other animation tools.

1 Introduction

In order to improve the quality of teaching, many educators have become interested in using animations in their courses. This is especially true for dynamic behavior that is often hard to explain using slides or blackboards. In computer science education, the main focus of animation tools available for this purpose is the animation of *algorithms* and *data structures*.

Some of the animation tools available [2] display predefined animation sequences without active user participation. Others depend on the current underlying data structure [1], but allow user interaction.

Some animation tools [5, 10] are very good at displaying animations, but restrict the user's editing abilities considerably. If the only way to generate animations is by API calls, computer laypersons are prevented from effectively using these tools. Several current animation tools [6, 12] try to avoid this problem by using a scripting language. Explicitly generating the scripts may be difficult for most non-programmers.

ITiCSE 2000 7/00 Helsinki, Finland

In this paper, we present ANIMAL, a new animation tool we developed to address the shortcomings of other animation tools. ANIMAL offers visual editing of the animation and is thus very easy to use. A simple scripting language and an animation generation API are also provided. ANIMAL offers a selection of powerful features that can easily be combined to create and display animations of algorithms, data structures and many other topics.

There is a growing awareness that animations and visualisations should *supplement*, but not *supplant* textual commentary. ANIMAL acknowledges this by supporting the integration of both source/pseudo code and textual descriptions into animations.

ANIMAL is used in our introductory computer science courses at the University of Siegen, Germany containing more than 250 students of different majors.

The paper is organized as follows. In section 2, we discuss desirable features of animation tools. Section 3 evaluates the currently available tools. Section 4 presents the functionality of ANIMAL, and section 5 evaluates ANIMAL. Section 6 concludes the paper and outlines areas of future research.

2 Desirable Features of Animation Tools

In the following, we present a list of features that we think an animation tool of wide usability and targeted for diverse audiences should provide.

1. Platform independence

   The animation tool should run at least under Windows and Linux/Unix-based operating systems.

2. Free availability for teachers *and* students

3. Easy usability

   Ideally, the tool should require no programming for displaying or generating animations.

4. Network independence

This is necessary because student apartments and lecture theatres may not have network access.

5. Support for inclusion of source or pseudo code and textual descriptions

6. Support for code and element highlighting and marking

   Related features like array index pointers need not be included automatically, but should be supported in some way.

7. Operations should take user-adjustable time

8. Wide applicability

   The tool should not be limited to algorithm or data structure animation.

We do not claim that this list is complete. However, for our teaching environment, we needed all these features at least part of the time. See also [8] for a survey of animation generation strategies.

3 Evaluation of Available Animation Tools

We evaluated the following tools according to our requirements: *JAWAA* [6], *Jeliot* [3], *JSamba* [12] and the recently announced tool *AlgAE* [13].

All evaluated tools are implemented in Java and thus satisfy our expectation of platform independence. *Jeliot* relies on a central server, thus failing the free availability and network independence requirements (features 2, 4).

Animations are generated either using a special scripting language (*JAWAA*, *JSamba*) or directly from Java source code (*AlgAE*, *Jeliot*). Having to provide Java input restricts the tools to teachers with knowledge of Java, and using Java in their course. Even a simple scripting language as employed by *JAWAA* and *JSamba* may prove to be too difficult for novice users, especially those coming from other fields than computer science.

AlgAE and *Jeliot* offer very good support for inclusion of source code and code highlighting; however, both are restricted to Java or - for *AlgAE* - C++ source code, preventing their use for topics like Turing Machines or verification. Neither tools supports pseudo code.

JAWAA and *JSamba* do not support embedding source code directly. By entering the code as text components and changing the text color, code highlighting can be simulated. Code indentation may be bothersome.

None of the tested tools seems to allow assigning execution time to individual operations, offering only a "delay time" between animation steps. Thus, none of the tools was able to fulfill all our needs. We started developing our own tool to address our requirements, which is described in the next section.

4 The Animation Tool ANIMAL

ANIMAL is an acronym for "*A N*ew *I*nteractive *M*odeler for *A*nimations in *L*ectures". ANIMAL is written completely in Java using Java's *Swing* library. To make ANIMAL as easy to use as possible, animations can be generated and edited on a drawing pane. Thus, animation authors do not need programming knowledge.

ANIMAL offers the five basic graphic primitives *point*, *polygon/polyline*, *text*, *list element*, and *arc*. All components can be comfortably configured. For example, two mouse clicks are sufficient for opening a given arc editor window and changing it to a circle segment.

More specific objects can be generated from a given basic object in the same way, for example toggling between arc and circle elements with a single mouse click. "Open" objects may possess a forward and backward arrow. The number and position of attached pointers for list elements can be changed similarly. The program tracks the current settings for primitive types and uses them as default for the next generated object.

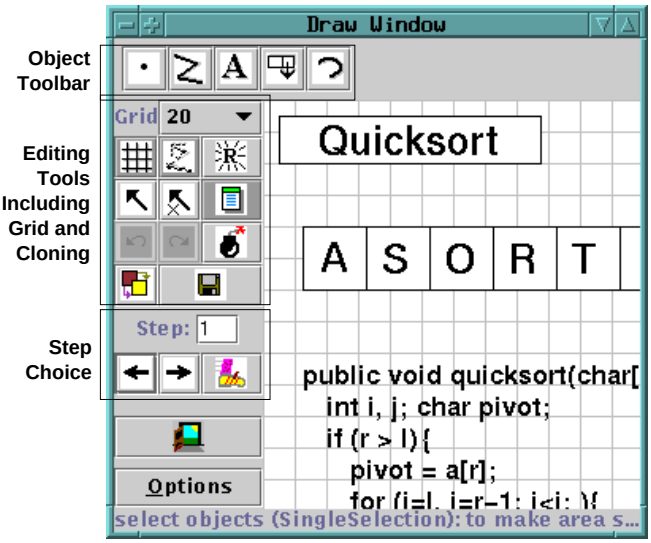

Figure 1: Screen shot of ANIMAL's drawing window

Objects are generated and placed via drag and drop. After the first setting of a point, a rubber band drawing algorithm is used to show the current object. Figure 1 shows a screen shot of ANIMAL's drawing window.

The editing tools provide buttons for toggling grid, snap, single versus multiple selection mode and a clone operation. All objects are edited by clicking on the object and dragging either the whole object or selected edges. In our experience, it took at most a few minutes for animation authors to become sufficiently familiar with the drawing interface to start generating objects.

Animations consist of several animation steps, each contain-

ing an arbitrary number of animation effects. For consistent behavior and undoable operations, each object can only be used in one animation effect per step. The author can decide whether the next step is shown only after pressing the "play" button, or automatically after a flexible delay time.

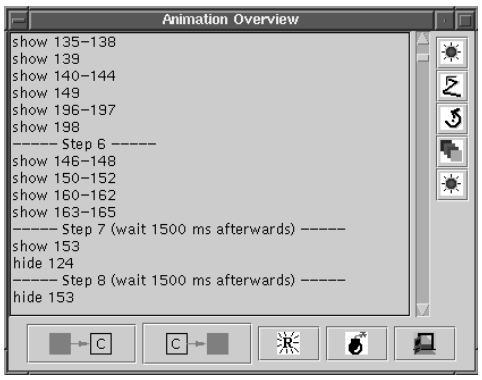

Figure 2: Overview of animation steps for *Quicksort*

Animation steps are edited in the window shown in figure 2. ANIMAL provides *move*, *rotate*, *change color*, and *timed show/timed hide* effects. All effects can be configured by given an *offset* and *duration*, measured either in milliseconds or internal time units called "ticks", and may operate on an arbitrary number of visible objects.

Users can decide on generating animations using ANIMAL's built-in scripting language. The language notation is similar to the ones used by other systems [6, 12]. Elements are placed at absolute coordinates or relative to a given object box. As the base point, all edges and centers of the object's bounding box can be used.

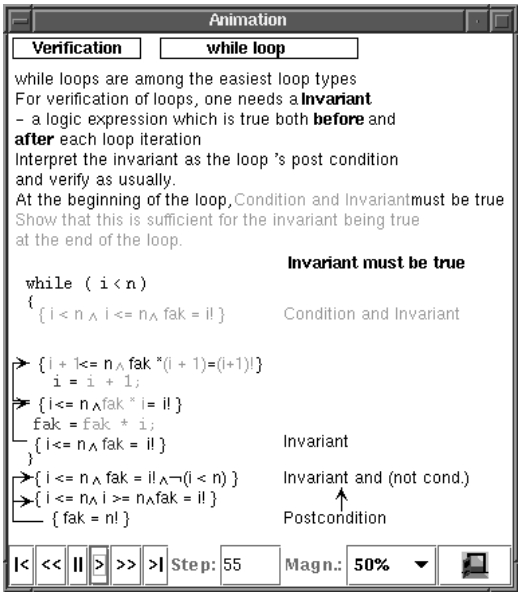

Figure 3: Example Animation: *Verification*

Users wanting to generate animations automatically can use either the scripting interface or the ANIMAL API. The API generates animation files in ANIMAL's built-in scripting language and thus offers the same functionality as the scripting language. Experienced programmers may prefer embedding API calls in their programs over scripting. The API is currently only available in Java, but porting it to other languages should be very easy.

Animations are displayed with video player-like functionality including play, pause and a direct jump to a given step. The display can be scaled to 50%, 71%, 100%, 141% or 200%. Figure 3 shows a snapshot an animation scaled to 50%.

ANIMAL's limited number of operations is beneficial for generating interesting animations in short time. ANIMAL can also be used by virtually anybody after a brief introduction into the object drawing and generating animation steps.

Another advantage of our animation tool is the simple, human-readable storage format. The largest current animation uses 780 graphical objects in nearly 200 steps, but takes up only 10kB. Parsing and generating the complete animation takes about 4 seconds on a Pentium II-350 running Linux.

ANIMAL is freely available at `http://www.informatik.uni-siegen.de/~inf/Software/Animal/index.html`. Our animation web page [7] currently contains 47 animations complete with screen shots and German and English descriptions.

5 Evaluation of ANIMAL

ANIMAL contains all features we initially listed as essential in section 2. It offers flexible timing conditions that can be used to arrange separate operations to overlap at a specified point in time.

ANIMAL does not make any assumptions about specific knowledge of users. Thus, it can be easily used for other topics outside the range of many other tools. Using the visual editor, computer laypersons should also be able to generate animations. More experienced users will appreciate the embedded scripting language and animation generation API.

ANIMAL offers more flexible object types than many other animation tools. For example, *JSamba* restricts polygons to at most seven nodes. ANIMAL supports arc types and allows moving objects along arbitrary polyline or arc objects.

We employed ANIMAL for one topic often not used in animations: *verification* using Hoare's calculus – see figure 3 for a snapshot. We managed to reach more students with this admittedly difficult topic than our predecessors. The exact role of ANIMAL remains hard to evaluate, though we believe it helped many students gain a basic understanding of the topic.

As we integrated the animations directly in our lectures,

we cannot really evaluate the animations' effect. Comparing two successive courses [9], the course using animation seemed to grasp the idea behind even complex algorithms much better. However, it is difficult to assess how much of this effect was caused by individual differences between the students of the two courses.

A recent study [4] suggests that animations facilitate learning by making algorithms more accessible and less intimidating. This effect was clearly visible, with an increased number of students taking an active role. However, as the study also mentions, a true evaluation of animation usage benefits is difficult.

In our evaluation at the end of the course, the students honored the use of animations in lectures: 85% said that we should continue using animations in future lectures.

The motivational aspect of using animations should not be underestimated. For example, both the announcement of the first animation and its presentation were met with spontaneous applause. We also got many students to actively think about the topic when we showed the presentation. Students managed to find most of the inconsistencies that slipped into our animations, and were anticipating and discussing what the result of the next step should be.

6 Conclusions

In this paper, we have presented our animation tool ANIMAL, which has proven to be very helpful in introductory computer science lectures held at the University of Siegen, Germany. ANIMAL avoids many of the shortcomings of other animation tools. It offers a set of powerful features to create, modify and display animations in a simple manner without being limited to specific topics. Due to its flexibility, usage is not limited to these areas. ANIMAL's visual editor can easily be used by laypersons to generate and edit animations.

There are several issues for future work. First, we plan to support parsing the format of *Tango/Samba* [11] and *JAWAA* [6]. We will then try to integrate animations generated on demand during the lecture to address specific questions by the students using our animation API.

Finally, we plan to extend the animations generated with ANIMAL which are currently collected on our animation page [7] for free access world-wide.

References

[1] Baker, R. S., Boilen, M., Goodrich, M. T., Tamassia, R., and Stibel, B. A. Testers and Visualizers for Teaching Data Structures. *SIGCSE '99 Proceedings* (March 1999), 261–265.

[2] Barbu, A., Dromowicz, M., Gao, X., Koester, M., and Wolf, C. Bubblesort-Animation, 1998. Available at `http://www-cg-hci.informatik.uni-oldenburg.de/~da/fpsort/Animation.html`.

[3] Haajanen, J., Pesonius, M., Sutinen, E., Tarhio, J., Teräsvirta, T., and Vanninen, P. Animation of User Algorithms on the Web. *IEEE Symposium on Visual Languages* (1997), 360–367.

[4] Kehoe, C., Stasko, J., and Taylor, A. Rethinking the Evaluation of Algorithm Animations as Learning Aids: An Observational Study. Tech. rep., Georgia Institute of Technology, March 1999.

[5] Rasala, R. Automatic Array Algorithm Animation in C++. *SIGCSE '99 Proceedings* (March 1999), 257–260.

[6] Rodger, S. JAWAA and Other Resources, 1997. Available at `http://www.cs.duke.edu/~rodger/tools/tools.html`.

[7] Rößling, G. Collection of ANIMAL Animations, 1999. Available at `http://www.informatik.uni-siegen.de/cgi-bin/roesslin/animations.pm`.

[8] Rößling, G., and Freisleben, B. Approaches for Generating Animations for Lectures. *Proceedings of the 11th International Conference of the Society for Information Technology and Teacher Education (SITE 2000)* (2000), 809–814.

[9] Rößling, G., and Freisleben, B. Experiences In Using Animations in Introductory Computer Science Lectures. *SIGCSE 2000 Proceedings* (2000). To Appear.

[10] Silicon Graphics. JAL Algorithm Animation, 1998. Available at `http://reality.sgi.com/austern/java/demo/demo.html`.

[11] Stasko, J. Using Student-built Algorithm Animations as Learning Aids. *SIGCSE Session Proceedings* (February 1997), 25–29.

[12] Stasko, J. Samba Algorithm Animation System, 1998. Available at `http://www.cc.gatech.edu/gvu/softviz/algoanim/samba.html`.

[13] Zeil, S. J. AlgAE: Algorithm Animation Engine v2.0, 1999. Available at `http://www.cs.odu.edu/~zeil/algae.html`.