Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 178 (2007) 69–78
www.elsevier.com/locate/entcs

An Integrated and "Engaging" Package for Tree Animations

Guido Rößling[1]

Computer Science Department
Technische Universität Darmstadt
Darmstadt, Germany

Silke Schneider[2]

Computer Science Department
Technische Universität Darmstadt
Darmstadt, Germany

Abstract

This paper describes a prototypical system for tree and tree algorithm animations. The system allows the user to create his or her own tree from a selection of supported tree types by inserting and removing nodes. At any point in time, the user can ask for a visualization of the tree's generation to see the effects of the chosen actions. The user can also request embedded documentation on different levels of detail. The system may prompt the user to answer questions about the displayed contents. The online help further assists the user in learning about the tree types and operations. The package presented in this paper corresponds to the engagement levels *viewing, responding, changing* and also supports the *presenting* engagement level as defined in the engagement taxonomy [3].

Keywords: Algorithm visualization, tree animations, interaction, engagement

1 Introduction

The ITiCSE 2002 Working Group on *Improving the Educational Impact of Algorithm Visualization* proposed the following engagement taxonomy for the educational use of algorithm visualizations and animations (AV) [3]:

1: No viewing – AV content is not used at all;

2: Viewing – AV is used, but the user's involvement is centered on consuming the content or controlling the progress (forward / pause / faster / ...);

[1] Email: roessling@acm.org
[2] Email: sschneid@rbg.informatik.tu-darmstadt.de

doi:10.1016/j.entcs.2007.01.031

3: Responding – the user is asked certain questions while the content is presented, typically requesting him or her to predict the next state of the animation ("interactive prediction");

4: Changing – the user can modify the visualization, usually by specifying the input data set or selecting actions that cause an update of the visualization;

5: Constructing – the user generates a new animation of a given algorithm;

6: Presenting – the user presents a visualization to an audience for feedback and discussion. The content of the visualization does not have to be created by the user.

The Working Group also stated two hypotheses: first, that there is no significant difference regarding learning outcomes between the two lower levels; and second, that there will be a significant difference in learning outcomes when comparing any other "higher" level with a "lower" level. Thus, each step in the hierarchy (apart from the first two) is supposed to increase the possible improvement in learning outcomes.

In this paper, we present a new Java application for animating trees and tree algorithms. The motivation for developing this application was not only the development of a new educational tool for trees and tree algorithms. We also wanted to explore the integration of a set of recommended features in a single AV system, as listed in [10]:

- reliably reaching a large target audience by using Java,
- using a general-purpose system as the underlying visualization engine—in this case, the established ANIMAL system [6],
- allowing users to provide input to the algorithm,
- supporting a flexible *rewind* operation,
- incorporating interactive prediction,
- providing hypertext explanations of the visual display,
- and using smooth motion instead of static images.

Apart from the engagement levels 1 and 2 supported by all AV tools, the application also supports levels 3 (responding) and 4 (changing), and can be used for level 6 (presenting).

2 Creating and Visualizing Tree Animations

Figure 1 shows that the application is divided into four areas. At the top is the control area, containing the application's menu bar and (draggable) tool bar for creating or modifying the current trees. The main part of the window is taken up by the *documentation* and *animation area*, which shows the current content.

The right side contains an index of the web pages as a reference. This page is essentially identical to the welcome page initially shown in the documentation tab of Figure 1. The *Data objects* view provides access to the current instances of the

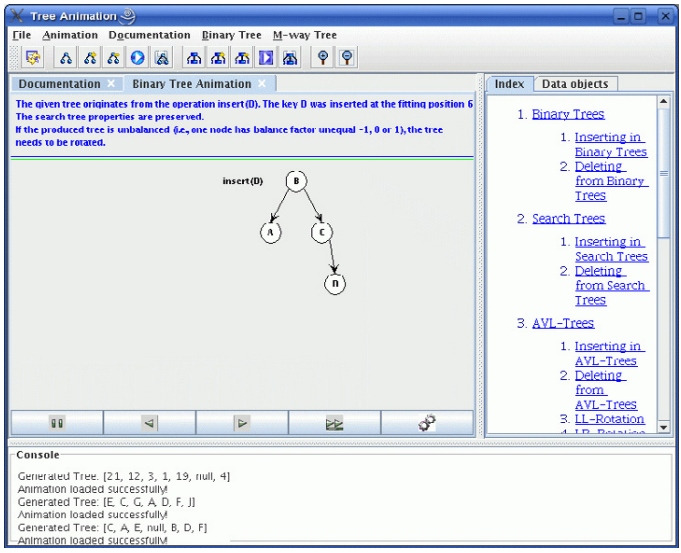

Fig. 1. The tree visualization application with index and AVL tree documentation

two supported tree types, binary and m-way trees, as shown in Figure 2. Finally, a console at the bottom of the window displays the output of the operations.

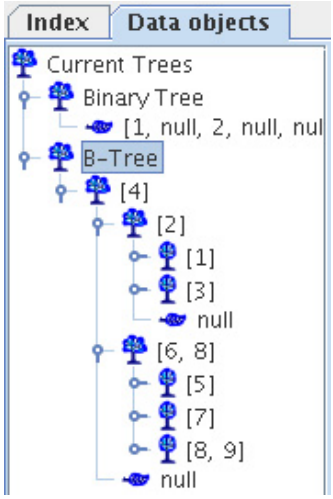

Fig. 2. The Data Objects tab shows the generated binary and B-Tree in array notation

The buttons in the toolbar can be used for performing tree operations. The first and the last two buttons shown in Figure 3 can be used to load a new animation, and zoom in and out of the animation, respectively. The two groups of five buttons each are used for creating a new tree, inserting a new key, removing a key, showing the generated animation, and saving the animation to disk. The first set of buttons is for binary trees, while the second group is for m-way trees, as indicated by the shape of the node used for the icons.

The application currently supports the following types of trees:

Fig. 3. The application's toolbar

- binary trees,
- binary search trees,
- AVL trees,
- heaps, implemented as *min* or *max heaps*,
- and m-way search trees with the implementation of a B tree.

The trees were modeled as described in the literature, e.g. in [1]. Adding new tree types is simple, and described together with other extensions in more detail in Section 5.

The user can create a new tree instance by selecting either binary or m-way trees from the menu. He can then decide whether documentation shall be included in the display, and whether interactive questions should be asked during execution, as described by the 2002 ITiCSE Working Group [3]. The user then works with the tree using the buttons shown in the toolbar of Figure 3 to insert or remove keys.

Each operation causes the generation of animation content parallel to the performance of the user-requested operation. The generation process itself is invisible to the user, until the animation is displayed by the user selecting the appropriate "show animation" button.

The animation code is generated in AnimalScript [7] and displayed by a customized front-end of Animal [6]. Figure 4 shows an excerpt of the animation tab for an AVL tree animation. In Figure 4, the insertion of the key *19* will cause a *RL(i) rotation* to rebalance the AVL tree. Double rotations in AVL trees are often poorly understood by students. Therefore, the animation incorporates a schematic display of the state of the (sub-)tree before and after the operation, shown on the left side of Figure 4. In the schematic display, the balancing factor of the nodes is placed above the nodes.

After the animation is started, the user can continue modifying the tree using the buttons described above. The application creates add-on visualization code for the current animation to reflect the results of the actions taken by the user. This can then be loaded in using the "incremental loading" button shown at the bottom right of Figure 4, indicated by the interlocking cogwheels. Instead of parsing the complete animation, only the code generated for the operations performed since the

animation was last loaded or updated using "incremental loading" will be parsed and added to the end of the current visualization, boosting the speed of the display.

The application described so far thus combines the generation of a tree according to the user's operations and the stepwise animation of the output. Therefore, the functionality of the application as described in this section places it on the levels 2 (viewing) and 4 (changing) of the Working Group engagement taxonomy [3].

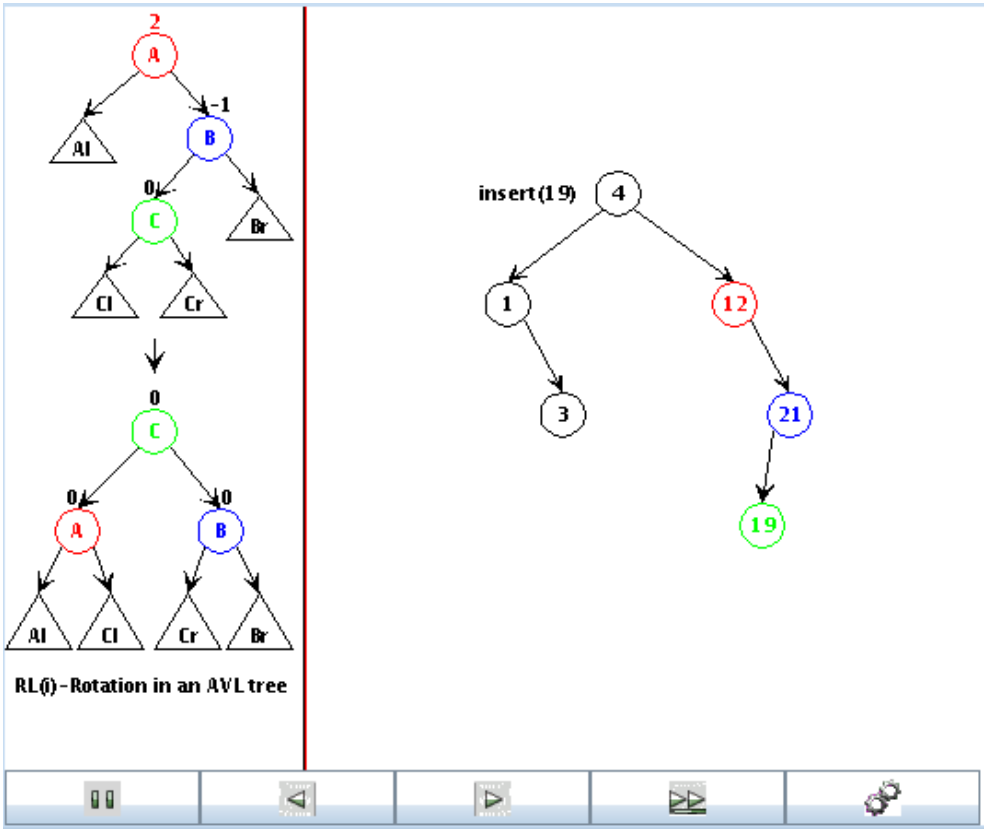

Fig. 4. Excerpt of the AVL animation illustrating a *RL(i) double rotation*

3 Embedded Documentation

Rößling and Naps [10] stated the importance of incorporating hypertext explanations of the visual display. This documentation is meant to describe the operation of the underlying visualization engine, the mapping of the algorithm to the visualization, and should be "ideally adaptive to the current state of the algorithm" [10, p. 97].

The approach up to this stage is similar to some existing AV systems, especially *MatrixPro* [2]. However, some of the planned features, especially considering embedded schematic displays as shown in Figure 4, are not easily implemented in these systems. Basing the work on AnimalScript makes generating content like

this relatively easy. The notation of AnimalScript is easy to learn, and already incorporates support for interactive elements used in conjunction with Naps' JHAVÉ system [10].

Users of our application can use the built-in documentation provided in HTML format. Figure 5 shows part of the HTML page explaining the *RL(i)* rotation animated in Figure 4. This HTML documentation does not refer to actual values, but provides one page for each possible operation (insertion and removal of tree nodes and all balancing or reordering operations).

Additionally, the user can decide to turn on additional documentation to explain the individual operations. The documentation is adaptive to the current state of the algorithm and also mentions the concrete affected elements for any operation.

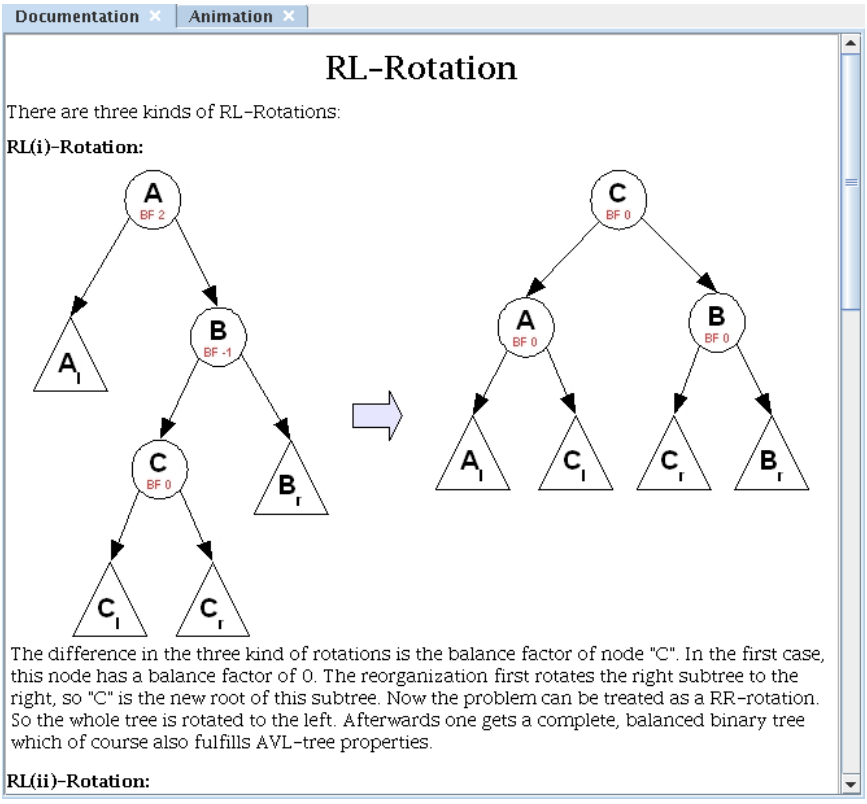

Fig. 5. Example documentation: performing a *RL(i)* rotation on an AVL tree

The adaptive documentation is included in the animation, to prevent the user from having to switch views. When it appears, it is placed at the top of the animation window. The documentation is currently available for all supported tree types except for B-Trees. It documents both insertion and removal operations in one of four levels of detail:

BEGINNER offers up to four lines of documentation for each insertion or removal;

INTERMEDIATE offers two lines of documentation for each operation;

EXPERT offers a single line of documentation;

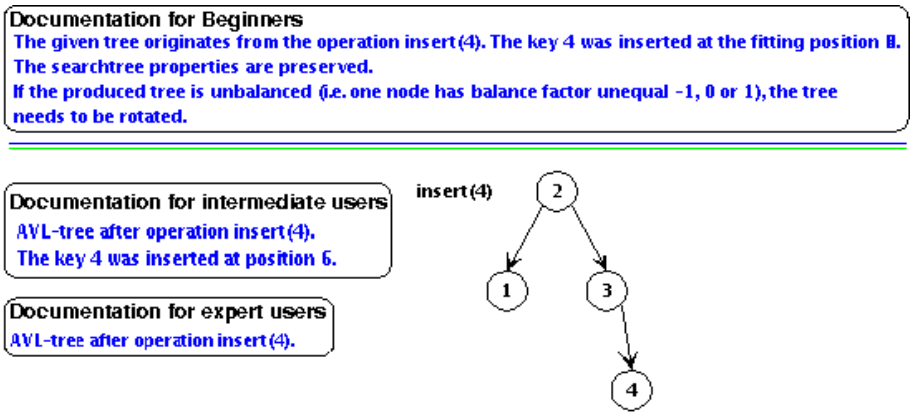

Fig. 6. Documentation levels in the animation: beginner, intermediate, and expert

NO_DOCUMENTATION turns off all documentation features within the animation.

Figure 6 shows the three documentation levels (*NO_DOCUMENTATION* is not shown, as it is by definition empty). The basic animation content is the documentation for beginners, with the tree to the right. The documentation for intermediates and experts was grafted to Figure 6 to allow for easy comparison. The documentation, and indeed the whole application, is prepared for internationalization, and already addresses English and German. Adding more languages is fairly straightforward, as it mainly requires translating the resource text files.

The combination of live tree exploration with embedded *and* external documentation brings the application close to the realm of hypertextbooks described in [4], also addressed by an ITiCSE 2006 Working Group and reported in a forthcoming publication [9]. It also makes it easy to use the application to present tree algorithms to an audience, making the system applicable to level 6 (presenting) of the engagement taxonomy for its (admittedly) narrow focus.

4 Interactive Prediction Support

The *responding* category in the engagement taxonomy expects the animation to be interrupted by questions. These questions will typically prompt the user to predict the next step performed by the algorithm, or ask him or her to describe the current visualization state.

For this end, we have incorporated the *avInteraction* package [8] to the application. The functionality of this package is similar to the "interactive prediction" supported by *JHAVÉ* [10]. The avInteraction package can easily be extended or adapted to other systems. Additionally, it offers other interesting features, such as skipping questions once a specified number of "related" questions were answered correctly.

Figure 7 shows an example pop-up window that prompts the user to predict which rotation will occur in the next step of the AVL tree reorganization. Currently,

only AVL trees and B-trees incorporate interactive prediction, although this can be changed easily.

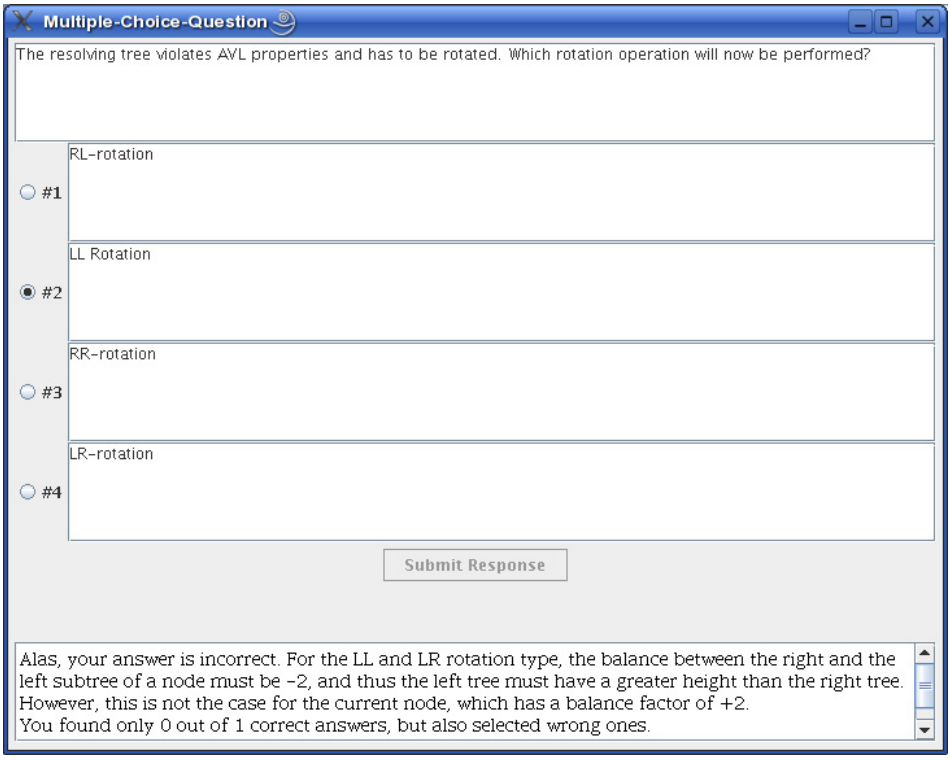

Fig. 7. Interactive Prediction: determining the correct type of rotation

5 Adapting or Extending the Package

There are at least four different ways how the application described in this paper can be adapted or extended:

- supporting additional languages,
- modifying the interactive prediction elements for existing or additional tree types,
- adding interactive prediction support for additional tree types.
- and finally adding support for additional tree types or tree operations,

To support an additional language, the user only has to translate one or two language resource files. The first resource file contains the translations for all GUI elements and other messages inside the application. The second file contains the messages for the menus for the two basic tree types - binary and m-way trees. The latter file has to be modified if additional elements, such as the ability to further configure a new tree type, need to be supported. This is only needed for special tree parameters, such as the order for a B-Tree (which is already included in the file).

The main language file contains about 150 lines of text, with a file size of 4 kB, and the additional file has 31 lines of text with about 800 characters. The translation process can be accomplished by anyone proficient in the target language and using a standard text editor - no programming skill is needed. Additionally, the application has to be made aware that there is a new language available by entering this information into a configuration file.

Modifying the interactive prediction elements for existing or additional tree types also requires the editing of a resource file coded as a standard text file. However, the user performing the translation also needs to be aware of the notation used for the interactive prediction elements described in the specification of the *avInteraction* package [8] and included in this package's documentation. The concrete values for a given question are included as parameters specified by placeholders in curly braces and substituted at invocation using the *translator* package [5].

Adding interactive prediction support for a tree type that did not previously support prediction requires the generation of another text file that specifies the interaction elements. This is the same file mentioned in the previous paragraph and usually has a size of about 2-5 kB.

Implementing support for a new tree type first requires the programmer to determine the proper superclass for the new type, e. g., the binary search tree class. The tree can then be implemented according to its definition. The package provides a service API for easily generating AnimalScript commands for the tree operations, so that the implementation of a new tree type is rather straightforward. Additionally, the programmer will have to slightly modify parts of the code to ensure that the new tree types appears in the selection dialog for the concrete subtype of binary or m-way trees - a task typically achieved in a few minutes.

6 Summary and Further Work

In this paper, we have presented an extensive application for creating tree visualizations. Based on the underlying Animal system, generating the animation code is fairly simple. The application merges several requirements from past Working Group reports and research papers [10,3]: both "static" and "live" documentation are included, and interactive questions can be activated to make the learner's session more engaging. Finally, the extensive collection of HTML pages describing the underlying data structures and algorithms makes this a promising learning tool for trees.

We have incorporated some new ideas into this prototype. This is one of the first systems that we are aware of that uses the actual values in its built-in documentation, and supports different levels of documentation detail. The ability to add code to the visualization without having to re-parse from the beginning is also a new feature, and definitely a premiere for the Animal system.

In the future, we want to further explore the connection between the ideas used in this system and the hypertextbook concept [4]. We plan to add some of the tried and tested features, especially the "incremental loading", into the Animal

AV system.

We are also looking for educators interested in trying out the application. We would especially appreciate colleagues who are willing to take part in an evaluation of the system regarding learning outcomes, as our teaching obligations currently do not include access to the data structures course, and therefore keep us from the key clientele.

References

[1] Goodrich, M. T. and R. Tamassia, "Data Structures & Algorithms in Java," Wiley & Sons, 2005.

[2] Karavirta, V., A. Korhonen, L. Malmi and K. Stålnacke, *MatrixPro - A Tool for Ex Tempore Demonstration of Data Structures and Algorithms*, in: *Proceedings of the Third Program Visualization Workshop, University of Warwick, UK*, 2004, pp. 27–33.

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

[4] Ross, R. J. and M. T. Grinder, *Hypertextbooks: Animated, Active Learning, Comprehensive Teaching and Learning Resources for the Web*, in: S. Diehl, editor, *Software Visualization*, number 2269 in Lecture Notes in Computer Science (2002), pp. 269–284.

[5] Rößling, G., *Translator: A Package for Internationalization for Java-based Applications and GUIs*, in: *Proceedings of the* 11^{th} *Annual ACM SIGCSE / SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2006), Bologna, Italy*, 2006, p. 312.

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

[7] Rößling, G., F. Gliesche, T. Jajeh and T. Widjaja, *Enhanced Expressiveness in Scripting Using* ANIMALSCRIPT V2, in: *Proceedings of the Third Program Visualization Workshop, University of Warwick, UK*, 2004, pp. 15–19.

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

[9] Rößling, G., T. Naps, M. S. Hall, V. Karavirta, A. Kerren, C. Leska, A. Moreno, R. Oechsle, S. H. Rodger, J. Urquiza-Fuentes and J. Á. Velázquez-Iturbide, *Merging Interactive Visualizations with Hypertextbooks and Course Management*, ACM SIGCSE Bulletin **38** (2006), p. forthcoming.

[10] Rößling, G. and T. L. Naps, *A Testbed for Pedagogical Requirements in Algorithm Visualizations*, Proceedings of the 7^{th} Annual ACM SIGCSE / SIGCUE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2002), Århus, Denmark (2002), pp. 96–100.