Easy, Fast, and Flexible Algorithm Animation Generation

Guido Rößling
roessling@acm.org

Simon Kulessa
Simon.Kulessa@web.de

Silke Schneider
schneider_silke@gmx.de

Department of Computer Science
Darmstadt University of Technology
64289 Darmstadt, Germany

ABSTRACT

A common reason for not using algorithm animation in class or exercises is the amount of time needed to locate, adapt, or generate content. We present several components that try to improve this situation by providing easy, fast and flexible content generation. The different scope of the approaches makes it likely that they offer "something for almost everyone".

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education—*Computer-assisted instruction*

General Terms

Management

Keywords

Algorithm, Data Structure, Animation, Generation, Visualization

1. WIZARD-BASED GENERATION

The lack of use of algorithm visualizations (AV) is often linked to the time required to search for good materials, learn the associated tools, and adapt materials to one's own courses - or even generate new materials [1]. Therefore, one central challenge for AV system developers who want to see their systems in use is to reduce the amount of time needed by other educators to actually incorporate AV into their teaching.

We have extended the ANIMAL algorithm animation system [2] with a generator framework which supports content generation in just four steps, taking less than one minute in total to complete:

1. The user selects the topic area from a list, e.g., searching, sorting, or cryptography.
2. The user selects the concrete content generator from a list. Each generator also provides a textual description and pseudocode to enable the user to find the appropriate generator.
3. The user adjusts generator properties, especially the concrete algorithm parameters. Depending on the generator, some visual properties such as color settings may also be adjustable.
4. The user saves the animation and loads it into ANIMAL.

ITiCSE'07, June 23–27, 2007, Dundee, Scotland, United Kingdom.
ACM 978-1-59593-610-3/07/0006.

2. GRAPH ALGORITHM ANIMATIONS

A second tool supports the generation of graph algorithms. The user can create a graph by dragging the nodes and editing the edges. This can be done individually for each node or edge, by editing the adjacency matrix, or by using a built-in scripting notation. The user can then choose an algorithm to run on the graph. A similar set of options are available, adjacency matrix.

Currently, the tool supports *Dijkstra's Shortest Path Algorithm* (in the standard or a bidirectional form), *graph coloring*, *breadth-* and *depth-first searching, Kruskal, Relabel-to-Front* and the detection of *negative cycles*. It also distinguishes between the different types of graphs needed for the algorithms, such as *bipartite, Manhattan,* or *residual* graphs. This especially concerns the scripting notation, which is adapted to the selected type of graph.

3. TREE ANIMATIONS

A third stand-alone application supports the generation and visualization of different binary and m-way tree types, including B- and AVL-trees. The applications contains documentation on the tree types and the different insertion or deletion operations. On generating a tree animation, the user can decide whether he or she wants to be asked interactive questions during the animation, such as which type of (AVL) rotation will take place to rebalance the tree. The actual operation is shown on the current tree. A schematic rotation view makes the application of the algorithm and its documentation to the concrete tree easier to follow for users.

Short documentation comments can be embedded into the animation on three different (user-choosable) detail levels. In contrast to the fixed HTML documentation of the tree operations, these comments use the concrete values or node names used in the current step.

4. ACKNOWLEDGEMENTS

We want to thank a set of students who participated in developing animation components: Tobias Ackermann, Eike Kohnert, Michael Maur, Stephan Mehlhase, Jens Pfau, and Yasen Toshev.

5. REFERENCES

[1] Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., and Velázquez-Iturbide, J. Á. Exploring the Role of Visualization and Engagement in Computer Science Education. *ACM SIGCSE Bulletin 35*, 2 (June 2003), 131–152.

[2] Rößling, G., and Freisleben, B. ANIMAL: A System for Supporting Multiple Roles in Algorithm Animation. *Journal of Visual Languages and Computing 13*, 2 (2002), 341–354.