DsCats: Animating Data Structures for CS 2 and CS 3 Courses

Justin Cappos
Computer Science
University of Arizona South
Sierra Vista, AZ 85635
justincappos@hotmail.com

Patrick Homer
Computer Science
University of Arizona
Tucson, AZ 85721-0077
patrick@cs.arizona.edu

Abstract

A new data structure animation tool called DsCats is available for classroom use. This tool supports educator presentations, student experimentation, and programming assignments. It implements a user-centered approach supporting a wide range of detail levels, the ability to jump to any point in the animation, and on the fly variations in the data structure during animations. The tool is written in Java with modularity and expandability in mind.

1 Introduction

The Data Structure Computer Animation Tools (DsCats) Project is developing animation techniques for improving the presentation and understanding of data structures in Computer Science classes. The overall goal is to help students understand the details of the algorithms used to implement different storage techniques. Animation can benefit both the presentation of data structure algorithms in lecture and lab settings, and the individual exploration of the structures by the students. The initial data structures chosen for implementation are tree data structures (binary search trees, AVL trees, and B-Trees) that are commonly taught in CS 2 and junior-level advanced data structure courses.

The current DsCats tool combines a number of features not commonly present in other tools. These include the ability to

- vary the level of detail during the animation,
- move backward and forward at will through an animation, and
- display large data sets.

A set of commands can be used in input data sets to support the creation of animations that contain descriptive annotations and break points. The user (instructor or student) can pause animations and then insert or delete data at that point to view its effect on the data structure. A new input file can then be saved to allow later re-creation of the animation. The command set is independent of the data structure being animated, allowing students to do side-by-side comparisons without the need to create multiple input files.

DsCats is implemented in Java for portability reasons. New data structures are added by extending the DataStructure class, providing methods that describe the basics of drawing the data structure so DsCats can create animations.

2 Background

Data structure animation is a specialized field within the broader field of algorithm animation. Algorithm animation began with the video *Sorting out Sorting* presented by Baecker [1] in 1981. The video depicts different sorting algorithms running in an animated graphical fashion intended for CS 2 students. Obviously, since this is a videotape the student interaction is limited to watching the tape multiple times. Brown and Sedgewick took this idea from video to software in 1984 with the development of BALSA [2]. Brown evolved the basic idea into a number of new systems, including BALSA II, ZEUS, ZEUS 3-D, CAT and JCAT [3]. JCAT (Java-Based Collaborative Active Textbooks) supports 3-D graphics and simultaneous displays on multiple machines. Other innovative features include allowing everyone in a classroom to watch the same algorithm but allow each student to view the algorithm from a different perspective. A large number of other general-purpose algorithm animation systems have been developed with similar results [10,11]. Differences between these projects and DsCats include that DsCats has the ability to move backward and forward through the algorithm while varying the detail level at will, and the use of commands in the input file to control the animation.

Data structure specific animation tools include BST [12] and Interactive Data Structure Visualizations [4]. BST is a data structure animation tool developed at Clark University. BST uses animation to show newly inserted nodes dropping into the tree from the top and routed to their correct position in the tree. The physics of the animation seem to be the main focus of this project. DsCats focuses on learning the algorithms.

Interactive Data Structure Visualizations [4] has an interesting and useful feature called “I’ll Try Mode” where a student may try to demonstrate their knowledge of how a data structure algorithm works by showing how the algorithm would operate on a given data set. However this program has not been shown to produce positive results

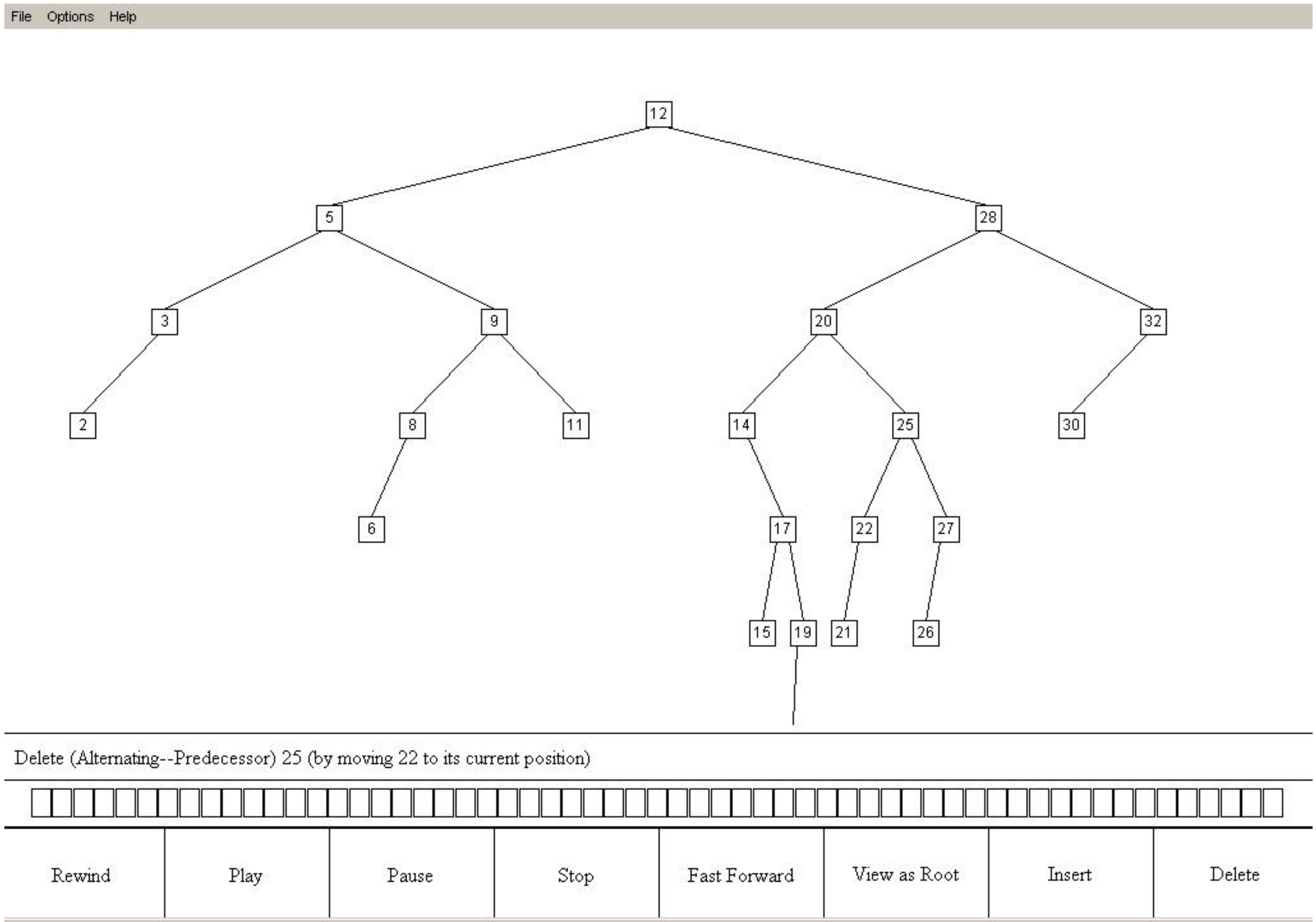

Figure 1 – A binary search tree about to perform a delete. On screen, color is used to highlight the node to be deleted (25) and the node that will replace it (22).

with the students that have used it. Appropriate use of animation, not just for entertainment, and students needing to gain understanding of the algorithms, not just guessing when they do not understand, seem to be missing factors.

According to research [5] and theory [6], a data structure animation tool should have a number of features in order to be useful for educational purposes. Students should be involved with the tool as much as possible. An example of this is creating the data sets that the program runs on. Students should also be able to change the data set of the algorithm during visualization to allow them to hypothesize about the algorithm. Studies have shown it to be important to allow the user to repeat steps and move backward through the algorithm [6]. It has also been shown that the steps of the algorithm should be redundantly labeled to maximize user learning. The user interface should have simple, well labeled controls so that beginning users are not intimidated by them, but should also permit advanced users to control a large portion of the algorithm such as changing the detail level displayed, or altering the view of the algorithm. The system should be forgiving to invalid input and perform simple error checking in order to present the user with understandable error messages. Color should be used effectively to highlight important portions of a data structure or illustrate other data such as the portrayal of time [7]. The tool should be distributed on the web in order to maximize the availability and should be available on a large number of platforms. If the tool will be used for demonstration purposes as well as individually by students, it should be downloaded and run as Java code, not as a Java applet running from a web page because of potential web delay problems during classroom presentations.

3 DsCats Overview

3.1 Features

DsCats is a new data structure animation tool designed for educational use. This tool currently implements binary search tree, AVL tree, and B-Tree data structures, and can be extended to animate additional structures. DsCats implements a number of new features to facilitate its use as a learning tool [8].

Data structure animations can be replayed either forward or backward. The user can move backward to replay a portion of the animation through the rewind button or by clicking on the time-line bar. Animations can be “run”, using a user-specified delay between frames, or the user can single step through the animation. At any point in an animation, the user can diverge from the data set and enter different values to be inserted or deleted from the structure. This effectively creates a new animation from that point. These new animations can then be saved for later re-use through the File menu. For example, a student can pause a B-Tree animation and enter several new values to see where internal nodes are created. Such explorations can be saved to create new input data files for later re-use.

Users can specify the level of detail used in animating the data structure through commands in the input file or via a pop-up window. For example, when performing a series of insertions, the detail level can be set at a high level where each frame of the animation contains one more insertion. At a low detail level, the same series of insertions will show a series of frames that include the comparisons being made and the specific branches of the tree being followed. The level of detail can be specific to a particular data structure. For example, the AVL tree shows the steps involved in deciding when to rotate, and the details of each rotation.

Large data sets are supported through the ability to change the viewpoint. The user can click on one of the visible nodes and DsCats will reset the viewpoint so that node appears as the root and displays the data structure from that starting point. This process can be repeated to explore downward through a data set. Such explorations can be done at any point during an animation. The user can always return to the top-level view by clicking the appropriate button at the top of the screen.

Each data structure can implement features specific to that structure. For example, the current binary search tree implementation supports deletion using both the in-order predecessor and in-order successor. A student can run experiments to see the result of deletions done with only one of the two techniques and compare with deletions that alternate between the two or randomly choose between the two. Such data structure specific settings, when implemented, appear as menu choices.

3.2 Input Files

Instructors and students may easily create their own data files for this program that represent the algorithmic actions happening with the data structure. For example:

```
# This is a comment
# Use the AVL tree data structure
OPTION DS AVLTREE
## Initialize the tree (in one step)
INSERT 20 15 30 2 18 24 70 3 45
## Insert these items individually
INSERT 10
INSERT 53
## Delete some elements
DELETE 24
DELETE 20
```

Lines that begin with a pound sign indicate comments. A line that begins with two pound signs signifies that the text on that line will be displayed at that point of the animation.

The creator of the data file may also specify options such as the data structure, play speed, and detail level. The play speed and detail level settings are changed to the values specified in the input file but the user is free to change the values while viewing the animation. When the data file creator specifies which data structure to use, the user must use that data structure for the data set.

To use one data file with multiple data structures, do not use the OPTION command to specify the data structure. The user specifies in the GUI any data structure that supports the commands used in the file. For example the above input file can be used for the binary search tree, B-Tree, and AVL tree if the OPTION DS AVLTREE line is removed.

It is also possible to click the appropriate buttons and menu options in the GUI to create a data file from the current animation with the desired options, insertions, and deletions. In this case, if comments are desired they must be added later by hand.

3.3 Implementation Details

The abilities to jump to any point and to vary the detail level are accomplished by creating a "movie" for the animation. Each frame of the movie represents one step in the animation at the lowest, most detailed, level. When the movie is played, only those frames that match or exceed the current detail level are displayed. Each frame includes a line of text that explains what is happening.

To implement a new data structure, a new class that extends DataStructure is written that draws frames based on specific commands. For example, the binary search tree has to handle an insert command, creating some number of frames. The exact number of frames depends on the lowest-level detail that the new class supports. Each frame of the movie is then handed to the DataStructureMovie class provided by DsCats. This class handles playing, rewinding, and stepping through the movie, determining which frames to display based on the current detail level.

To display a frame of the movie, the drawing method of the data structure is called to create a vector of objects, which DsCats then uses to draw the current frame. The line of text that explains the current frame is added to the frame.

The implementor of a data structure can provide additional features. For example, the B-Tree code in the current tool provides statistics for each frame that include the number of internal nodes in the tree at this stage, the number of levels traversed, and the number of comparisons performed.

This tool is written in Java and has been tested on a number of operating systems including Mac OS X, Windows 98, Windows NT, and Solaris 8. Binary search tree, AVL tree, and B-Tree data structures are implemented. Since the purpose of the tool is to provide an animation that the user can control, performance is based on human perceptions of "quick" response. As a result, we have not tried to precisely quantify performance. We have tried running tests that insert 100 or 250 elements. These work successfully with no noticeable delays on a 270 MHz Sun Ultra 5 running Solaris 8, a Gateway 400 MHz Celeron running NT 4.0, and an Apple 500 MHz Titanium running Mac OS X.

4 Use in Computer Science Courses

The motivation for this project is based on problems with explaining data structure algorithms in CS 2 and CS 3 courses. The process is too often an inherently static presentation. The real difficulty is that students may think they are understanding the presentation, but when they go to review their notes, they have a confusing set of static images that they recorded and insufficient information about how each image was derived from the previous one.

DsCats addresses this problem by allowing demonstrations based on prepared input files. The progress of the construction can be paused at any point to highlight what is happening at that stage. The demonstration input files can be made available to the students for study outside class. The instructor may choose to add notes as viewed comments within the data file so that students will see the notes as they walk through the data set.

More to the point, the detail level can be changed in response to questions from the students about what is happening, the rewind feature allows repetition of important points and supports answering questions from students about earlier steps. The ability to change an animation on the fly by specifying new inserts or deletes allows the instructor to respond to questions. These changes can be saved into new input files, which students use to later replay the animation when reviewing their notes, or when asking the instructor about the lecture outside of class. They can replay the demonstration, changing the speed or detail level. Students can enter values of their own to insert or delete at a certain point to explore how the data structure changes.

Exercises can be devised that play an animation to a certain point, then ask students to try different combinations of inserts and deletes. Statistics about the data structures can be easily collected by the students, such as comparing the balance of trees based on changing the ordering of the insert and delete commands, for example.

The instructor may choose to add notes as viewed comments within the data file so that students may view the notes as they walk through the data set again. Students also have the freedom to individually experiment with instructor given data sets to see what happens when the given insertions happen in a different order.

There are a number of theoretical observations that students can discover. They can be provided with an example AVL tree and asked to delete all of the nodes in a way that causes no rotations or alternately, a maximum number of rotations. This effectively tests and enhances the student's knowledge of the balance property of AVL trees.

Some additional assignment examples include

- Determining an ordering of inserts to create a balanced tree.
- Comparing the effect on the data structure of different deletion algorithms.
- Collecting statistics (number of null child pointers, number of internal nodes vs. leafs in a B-Tree)

Programming assignments can be made that involve changes in, or additions to, existing DsCats implementations. The assignments do not have to start from scratch. Students can take advantage of the code already present in DsCats, and concentrate on the specific algorithm assigned. Some examples include providing a different deletion algorithm, for instance using lazy deletion. The "deleted" nodes would be highlighted in a different color, and would be re-used when new insertions can be placed in those nodes. Statistics on re-use could be generated by the new code and displayed during animations. The animation is helpful in programming projects since it serves as a very good visual debugging tool. For cases of more serious bugs, a printing traversal is built into DsCats to help students debug problems when the display does not come up.

A more involved project is to implement a complete data structure in DsCats. Examples can include splay trees, red-black trees, and binary heaps.

5 Future Directions

The current DsCats tool implements three data structures and is being tested in sophomore and junior CS courses. Future work will include evaluations of the effectiveness of the tool in the courses. Additional tree-based data structures can be added fairly easily. We are exploring how to extend this work to support non-tree structures, including hash tables, linked lists, and heaps. DsCats is available for download at http://www.cs.arizona.edu/dscats/.

References

[1] Baecker, R., "Sorting Out Sorting", *SIGGRAPH '81*, Los Altos, CA, 1981

[2] Brown, M.H., Sedgewick R., "A System for Algorithm Animation", *Computer Graphics*, July 1984, 177-186.

[3] Najork, M.A., Brown, M.H., "Three-Dimensional Web-Based Algorithm Animations", *SRC Research Report 170*, July 31, 2001

[4] Jarc, D.J., Feldman, M.B., Heller, R.S., "Assessing the Benefits of Interactive Prediction Using Web-based Algorithm Animation Courseware", *SIGCSE 2000*, 377-381

[5] Hundhausen, C.H., "Toward Effective Algorithm Visualization Artifacts: Designing for Participation and Communication in an Undergraduate Algorithms Course", Unpublished Doctoral Dissertation, *Technical Report CIS-TR-99-07*, June, 1999, 18-32

[6] Khuri, S., "Designing Effective Algorithm Visualizations", *Program Visualization Workshop*, July 7-8, 2000

[7] Brown, M. H., "Color and Sound in Algorithm Animation", *SRC Research Report 76a*, August 30, 1991

[8] Cappos, J. A., "DsCats User Manual", Unpublished Technical Manual, August 12, 2001

[9] Cappos, J. A., "DsCats Programmer Manual", Unpublished Technical Manual, August 31, 2001

[10] Pierson, W. C., Rodger, S. H., "Web-based Animation of Data Structures Using JAWAA", *Twenty-ninth SIGCSE Technical Symposium on Computer Science Education*, p. 267-271, 1998.

[11] Stasko, J., "POLKA Animation System", *http://www.cc.gatech.edu/gvu/softviz/parviz/polka.htm*, August 30, 2001

[12] Ierardi, D., Li, T. W., "Binary Search Tree Applets", *http://aleph0.clarku.edu/%7Eachou/cs102/examples/bst_animation/*, September 1, 2001