EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2011)
T. Hammond and A. Nealen (Editors)

CSTutor: A Pen-Based Tool for Visualizing Data Structures

Sarah Buchanan, Brandon Ochs and Joseph J. LaViola Jr.

University of Central Florida, Department of EECS, Orlando, FL, USA

Abstract

We present CSTutor, a pen-based application for data structure visualization that lets users manipulate data structures through the recognition of handwritten symbols and gestures as well as edit the corresponding code. The UI consists of a sketching area where the user can draw a data structure in a way that is as natural as pen and paper. Running in parallel with the visualization is a code view where users can make changes to the source code and add functions which manipulate the data structure on the canvas in real time.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles K.3.2 [Computers And Education]: Computer and Information Science Education—Computer Science Education

1. Introduction

Data structures are generally taught by referencing hand written examples on the white board or by learning diagrams in a textbook while separately studying the code that drives these examples. This type of learning creates a disconnect between the diagram and the code that can often be confusing to students. Additionally, it is hard to determine which line of code corresponds to which change in the diagram, and this can prevent students from learning some of the key concepts of different data structures.

Alternatively, algorithm visualization tools are used by students and instructors as a supplemental learning aid. However, studies have shown that algorithm visualization alone does not improve a student's level of learning; it is the level of student engagement with the algorithm visualization that is effective [GMN03]. In addition, many existing systems are not practical for use in the classroom or by students because creating the data structures and animations is too time consuming. These systems are also not flexible enough to support creating and editing examples on the fly.

To facilitate a more unified approach to learning introductory computer science concepts, we present CSTutor, a novel, pen-based application for data structure visualization that combines data structure animation and programming. CSTutor lets the user create diagrams easily with a sketch-based interface directly linked to an animation engine. This combined system creates an up-to-date diagram with each code change. CSTutor has the flexibility of diagramming a data structure with the familiarity of pen and paper, while also being able to dynamically animate and update the data structure based upon any operations or code changes. In other words, if the user sketches changes to the diagram the code will dynamically change, and if the user edits the code the diagram will animate and update itself.

2. CSTutor User Interface

2.1. Sketching Area

CSTutor was designed to have an interaction model that makes the users feel like they are writing on pen and paper. We accomplish this by providing a sketching area for the user to create a data structure diagram (see Figure 1). Using a stylus, the user creates data structure components through writing and performing a variety of gestures. The data structures represented in CSTutor (Linked List, Doubly Linked List, Binary Search Tree, and Heap) were chosen specifically because they are the most basic data structures taught in introductory computer science classes. We create an interaction mode for each of these data structures, which the user can select from the menu buttons. In order to keep the UI as natural as possible, a combination of several basic gestures are used for each interaction mode. Linked Lists and Doubly Linked Lists use a combination of rectangle, line, and scribble erase gestures. For the trees (Binary Search Tree

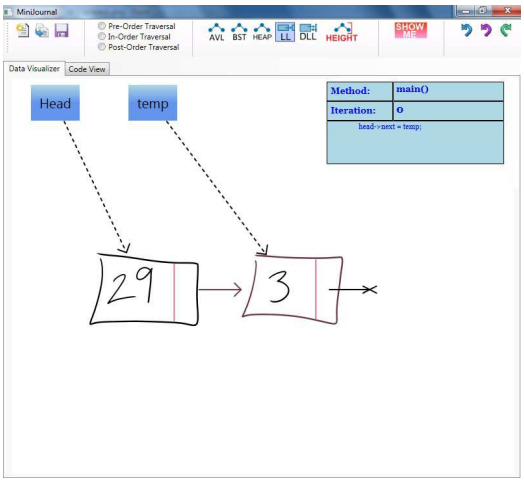

Figure 1: *The sketching area in Linked List mode after nodes 29 and 3 are added.*

and Heap) a combination of circle, arrow, tap, and scribble erase gestures are used.

2.2. Interaction Modes

There are different ways of sketching a conceptual operation on each type of data structure, which is translated directly to automatically generated code. All of the gestural operations that we have chosen are based on how a professor would teach these concepts using a diagram, and the content comes from a variety of course lectures and textbooks, including [Sta95].

The sketching area is designed to allow the user to do any operations on the data structure through sketching as they could with the code. They can create nodes, connect and disconnect nodes, assign/un-assign pointers to nodes, delete nodes and pointers, and update the data and next values. For example, a linked list node is created by writing the value of the node and then boxing that region in with a rectangle. When the rectangle gesture is completed, the surrounded strokes are sent to an online recognizer [AW10] that analyzes the user's handwriting to determine the numerical value. After the node is drawn it is translated and scaled to make room for subsequent sketches. Then, a small box is drawn within the node to represent the allocated memory for a pointer to the next node. A null pointer stroke is also added to this memory field for nodes where the next link is null (see Figure 1). Sketching a node produces the following steps in code: (1) allocating a node in memory, (2) assigning a value to that node, and (3) setting the node's next value to null. The applicable code is then displayed in a text area above the completed node, and simultaneously the code is added to the code area. To connect two nodes in a linked list, a line is drawn from one node to the destination node, resulting in a reference arrow (see Figure 1).

Tree nodes are created by writing the value of the node and then circling that number, which sends the surrounded strokes to an online recognizer, just as with the Linked List mode. After the node is drawn it is dynamically arranged on the canvas by translating and scaling the nodes to make room for subsequent tree nodes. Lines are drawn to each child node if the children are not null. In addition to inserting new nodes in the tree, the user can delete nodes by scratching out the node they want to delete. Since insertion and deletion are the main operations made to a binary search tree, gestural operations are provided for them and any other operations on the tree must be implemented in the user's code. AVL tree and Heap interaction modes are similar to the Binary Search Tree, however AVL trees allow rotations with a left or right arrow gesture and Heaps allow swapping nodes with up and down arrow gestures.

2.3. Code Area

Any sketch created in the canvas area has an association with a section of code. When a sketch is completed, the corresponding code appears in an information box on the canvas, and is simultaneously added to the main method in the code area. Users can edit the auto-generated code, add their own functions and operations, compile and run the code, and step through the resulting changes to the diagram.

3. Conclusion

We have presented CSTutor, a pen-based tool that supports the dynamic visualization of data structures including linked lists, doubly linked lists, binary search trees, and heaps. CSTutor creates a bidirectional interaction flow by synergistically combining hand drawn data structures with the corresponding code so the interaction between them can be visually explored. This synergy lets users change either the code or the hand drawn data structure to observe the effects.

4. Acknowledgments

This work is supported in part by NSF CAREER award IIS-0845921 and NSF Award IIS-0856045.

References

[AW10] ANTHONY L., WOBBROCK J. O.: A lightweight multistroke recognizer for user interface prototypes. In *Proceedings of Graphics Interface 2010* (Toronto, Ont., Canada, Canada, 2010), GI '10, Canadian Information Processing Society, pp. 245–252. 2

[GMN03] GRISSOM S., MCNALLY M. F., NAPS T.: Algorithm visualization in cs education: comparing levels of student engagement. In *Proceedings of the 2003 ACM symposium on Software visualization* (New York, NY, USA, 2003), SoftVis '03, ACM, pp. 87–94. 1

[Sta95] STANDISH T. A.: *Data Structures, Algorithms, and Software Principles in C.* Addison-Wesley, Reading, Massachusetts, 1995. 2