A 3D GRAPHICAL APPROACH FOR VISUALIZING DATA STRUCTURES AND ALGORITHMS

Aaron Rababaah, University of Maryland Eastern Shore, USA (arrababaah@umes.edu)

Brian Miller, University of Maryland Eastern Shore, USA (bcmiller@umes.edu)

Rakesh Joshi, University of Maryland Eastern Shore, USA (rjoshi@umes.edu)

ABSTRACT

In Computer Science education, data structures and algorithms are important topics for undergraduate students. Observation of existing data structure and algorithm visualization applications show a number of faults. Many visualization applications place too much emphasis on presenting every type of data structure or algorithm. These applications fail to demonstrate said constructs in a clear and concise manner. Therefore, the development of a dynamic, interactive environment to assist computer science students in visualizing data structures and algorithms would fill a much needed void in today's learning environment. In this paper we propose and develop an application to help students visualize data structures and algorithms while making use of a range of visual stimulus and providing a connection to real world applications. Concepts are demonstrated using animated, three-dimensional objects to enable students to visualize the data structures and algorithms. These concepts are represented as dynamically colored, labeled, and animated three-dimensional objects to create a fully immersive and visually stimulating experience. Our application provides students an alternative method to visually conceptualize the workings of specific data structures and algorithms. The students are able to see how these various structures and algorithms behave to provide them with a greater understanding of this abstract topic.

***Keywords**: Data structure, Visualization, Computer Science, Algorithm*

INTRODUCTION AND RELATED WORK

In this section, the related work to the proposed system of visualizing data structures and algorithms in the literature is discussed. There are several software tools that have been created to assist students visualize and conceptualize data structures and algorithms. Roberrevisionst Meolic [2] presented an approach to teaching sorting algorithms on a mobile platform. The algorithms were represented with pseudocode and a graphical display. Sort algorithms such as the insertion sort and the quicksort are implemented on a mobile application designed for smartphones. Rafael del Vado Virseda [3] presented a visualization tool to help computer science students learn about data structures and algorithms. The intention was to help students separate the implementation and specification of different data structures. This tool was available to computer science students enrolled in a data structures course at the Complutense University of Madrid. The tool used flash animations. Lauri Malmi, Ville Karavirta, Ari Korhonen, Jussi Nikander, Otto Seppala, Panu Silvasti [4] presented a Java applet that can display different algorithms and data structures. The name of this system is TRAKLA2. TRAKLA2 provides students with a set of visual exercises and then grades these exercises. The user can add also create new exercises. The graphics of this system are 2D. Chen Tao, Sobh Tarek [5] implemented a data structure visualization tool that features 2D data structure visualizations. This software focuses on the basic functions of the commonly used data structure. This tool does not include visualizations of sort or search algorithms. Jun Yang, Clifford Shaffer, Lenwood Heath [6] present a student controllable data structure visualization system called SWAN. SWAN was designed to help students visualize lists, trees, and arrays. SWAN uses a simple graphical user interface but does not provide sort or search algorithm animations. Urvashi et. al [1] did a literature review on various visualization tools used from 2001 to 2013. These tools were reviewed to investigate the advantages and disadvantages of various visualization tools.

THEORETICAL BACKGROUND AND TECHNICAL APPROACH

The HawkViz application is a visually immersive environment for visualizing data structures and algorithms. Its distinguishing features include the following elements:

1. A simple, intuitive, and consistent user interface schema
2. 3D graphics to keep students engaged
3. In-house developed OpenGL rendering engine that provides a high degree of portability
4. Application developed in C# with an extensive set of thread safe, high level classes that make adding new functionality easy

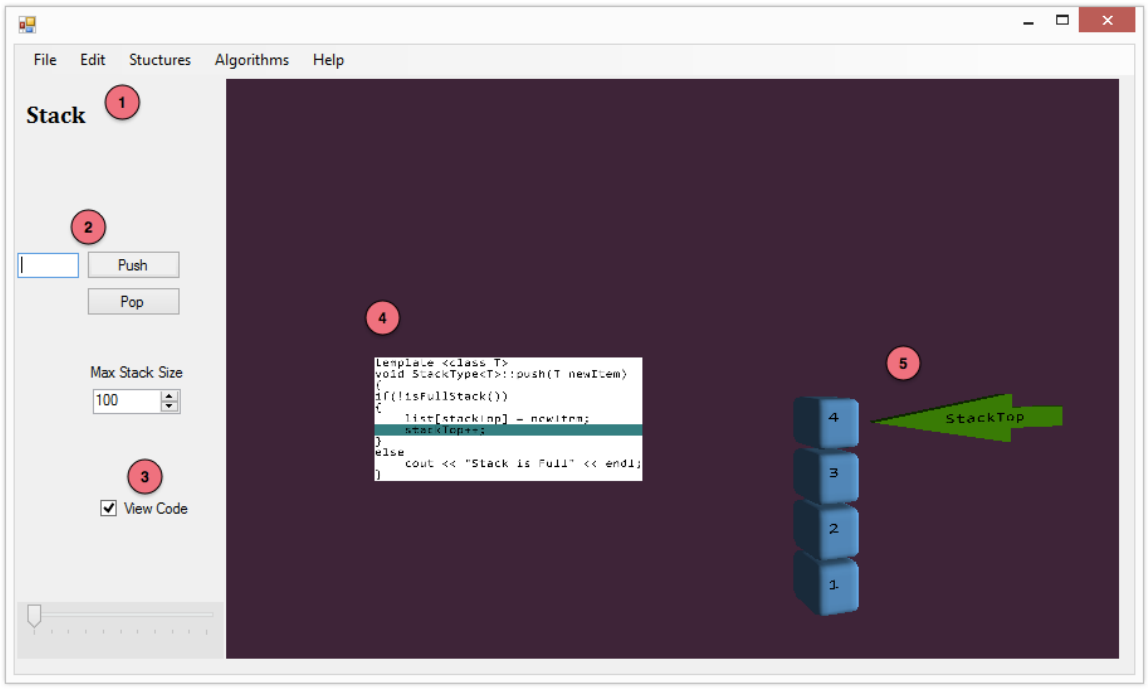

Figure 1: Annotated User interface overview

The HawkViz user interface consists of five main components. The first major component is the menu bar at the top of the application. The student selects which kind of visualization he or she wants by selecting an item in either the structures main menu or the algorithm main menu. The rightmost main menu item brings up the integrated help system which provides a manual whereby the user can learn about the various visualization settings as well as a guide for how to use the application in general. Just below the menu system, on the left side of the application, is a textual label which informs the user of which visualization is currently active. In Annotation #1, on Figure 1, we see that the currently selected visualization is the Stack example.

The second major component of the HawkViz user interface consists of various user interface widgets with which the user interacts with in order to manipulate the visualization. Every different kind of visualization has its own set of unique user interface widgets. These user interface elements are subobjects of a Windows Forms panel. Development of new visualizations is simple. All that is required for the user interface is to create a new instance of a panel with its own associated sub elements.

Every visualization module has C++ code associated with it. Annotation #3, in Figure 1 highlights an option to toggle the display of the code associated with a particular visualization. The display of the code itself is

highlighted in Annotation #4 of Figure 1. The code is rendered in an orthographic manner, with individual characters placed in front of a solid rectangle. Another thin rectangle is placed between the characters and the backdrop rectangle and functions as a visual indicator as to which line of code is currently being executed. It is intended for this to be similar to what one would experience if they were running a debugger and stepping through code in any one of a number of popular IDEs that are used for professional development, such as Visual Studio or Eclipse CDT. A side effect of doing the code visualization in this manner is that the screen window can be scaled to various resolutions and still be legible. The line selection indicator is animated in order to make it easier, and more obvious to the student, as to how the flow of execution is proceeding. In a traditional debugging application, it is common for the flow of execution to pop around suddenly, in our setup, the selector gracefully slides from one line to the next, with the speed of this transition increasing proportionally with the distance between jumps. The speed of the flow of execution is set by a slider bar in the lower left corner of the user manipulation area. This feature is especially useful in the algorithm visualizations as the user can speed up, or slow down the simulation. The underlying functionality of many algorithms is more pronounced when run at fast speeds whereas basic operations on data structures is often better visualized at a relatively slow pace.

The final and most important subsection of the HawkViz user interface is the actual visualization of the data structure or algorithm itself. All of the elements involved with the various visualizations are rendered in 3D. All of the elements in the visualization consist of 3D primitives, each of which can have an associated textual label placed in the front to convey information. For example, in the Array visualization, each element on the array is represented as a cube object, with the numeric value of each element affixed to the front of each cube as shown in Figure 2.

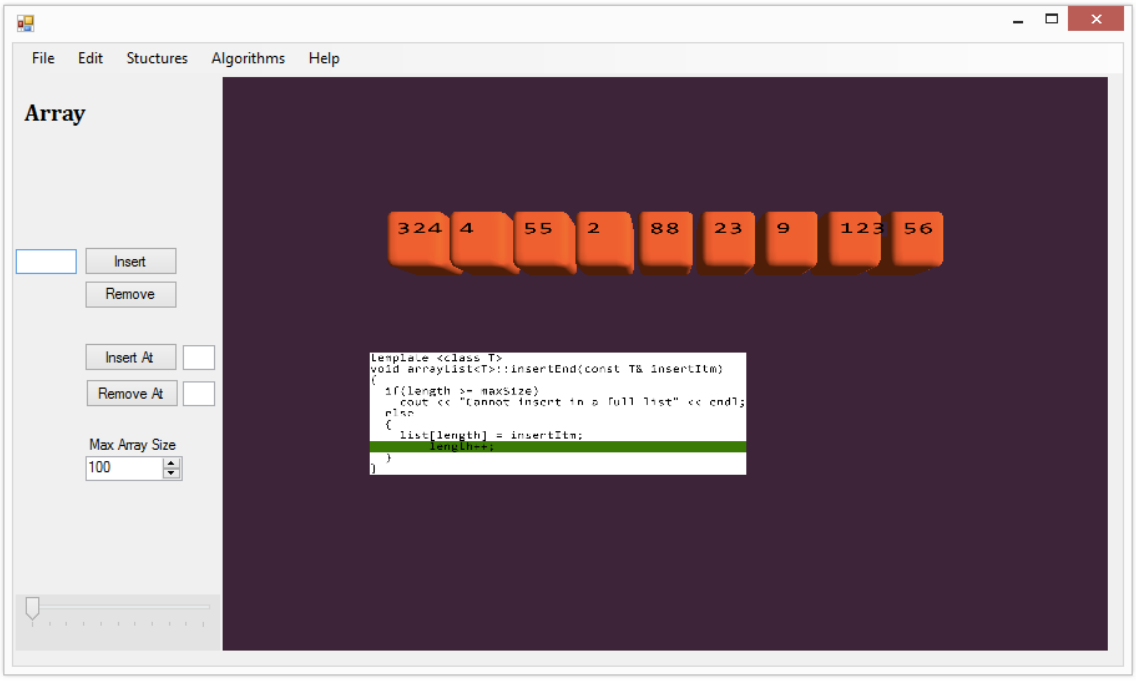

Figure 2: Example of Array Visualization

The objects themselves tie into a multithreaded animation layer which is capable of animating a large number of objects independently, or in groups simultaneously. The most basic form of animation is triggered by assigning an object a new coordinate in three space. In that instance, the object moves from its original position to the new position in a linear manner over the course of a specified number of frame intervals, which by default is set to 15. This has the effect of taking one half a second for an object to reach its destination. The frame rate of the application is throttled to 30 frames per second. Figure 3 shows a nice

example of the animation system with the Array visualization. In this example a user has elected to add a new element into the second position of the array. The new element is instanced above the position it is to occupy, and the right three most elements are shifted to the right as a group. When the void becomes available, the new element is dropped into place. This all takes place in just a few lines of code, utilizing high level routines to accomplish the visualization task.

Figure 3: Animation example involving the Array visualization

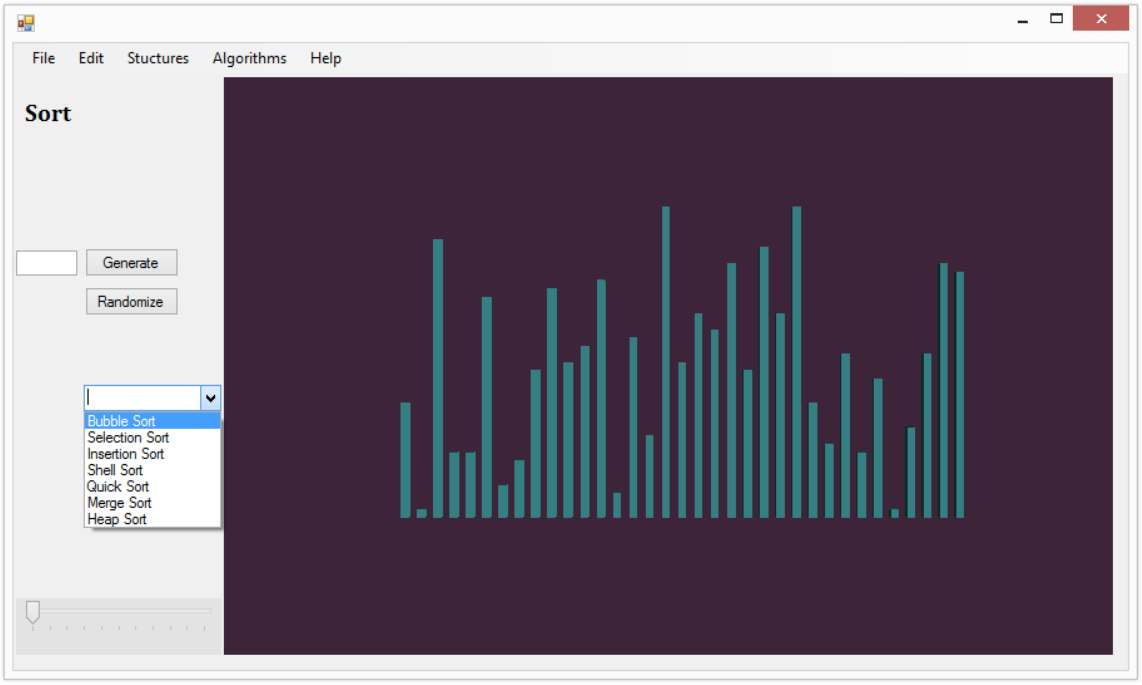

Figure 4: Animation example involving Sort Algorithms

Algorithm visualizations make extensive use of animation to provide the student with an engaging experience. Upon invoking a sorting algorithm from the list, the user is presented with an option to choose how many elements will be in the random set. After this value is set, the random elements are instanced into the viewing area in a manner that makes them appear to "rain" from the top of the screen into their initial resting place as the viewing camera slowly pans backwards to present them all in full view. A wide selection of sorts are available from the the selection menu in the user interface panel located at the leftmost side of the application. Sorting operations are animated by moving the bars around in a fluid manner that is consistent with the indicated algorithm

Search algorithm visualizations work in a similar manner as the sorting algorithm visualizations. The user selects the number of elements in the set and the elements appear into the viewport of the application in the same manner as above. The search algorithm animations do not involve any moving elements. Instead, the item selectors and other associated constructs toggle the color of the object in in a manner that is consistent with the indicated algorithm. Refer to Figure 5 for an example of this.

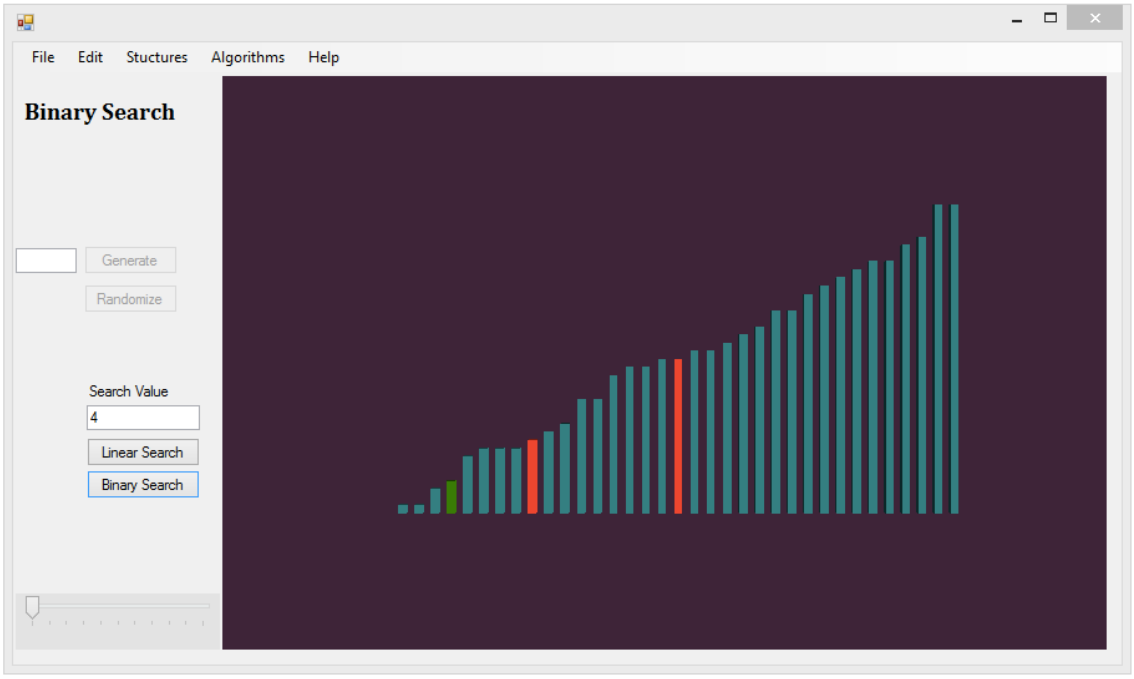

Figure 5: Animation example involving the Binary Search Algorithm

CONCLUSIONS AND FUTURE WORK

We believe that HawkVis can be a valuable asset to the educational environment for demonstrating to students how various data structures and algorithms work. The use of 3D animation coupled with direct user manipulation provides instant feedback in an appealing manner that students find familiar. Future developments will involve the following topics:

1. A more sophisticated animation subsystem that supports breathing effects on objects and other types of visually appealing animations
2. Render object text labels with shaders to avoid z-fighting issues that crop up with large object sets
3. Increase the quantity and types of various data structure and algorithm visualizations
4. Add functionality to allow the state of a visualization to be saved to and recalled from disc

We plan to release the source code for this application under the Gnu Public License (GPL) in the hope that it may be useful to the community and foster collaboration.

ACKNOWLEDGMENTS

We would like to acknowledge the kind assistance of the Math and Computer Science Department for their support with facilities and resources to make this project possible.

REFERENCES

[1] Urvashi et al., International Journal of Advanced Research in Computer Science and Software Engineering 4(3), March - 2014, pp. 338-341

[2] Meolic Robert (2013), "Demonstration of Sorting Algorithms on Mobile Platforms", Proceedings of International Conference on Computer Supported Education, pp. 136-141.

[3] V. Rafael d. V. (2010), "A Visualization Tool for Tutoring the Interactive Learning of Data Structures and Algorithmic Schemes", Proceedings of the 41st ACM technical symposium on Computer science education, pp: 187-191.

[4] Karavirta, V., Korhonen, A., Malmi, L., Nikander, J., Seppala, O., & Silvasti, P. (2003). Visual Algorithm Simulation Exercise System with Automatic Assessment: TRAKLA2. Informatics in Education, 3 (2), 267–288.

[5] Chen Tao, Sobh Tarek (2001), "A Tool for data structure visualization and user-defined algorithm animation", Frontiers in Education Conference, 2001. 31st annual, vol: 1, pp: 2-7.

[6] Yang J., C. A. Shaffer, and L. S. Heath (1995), "SWAN: A Student-Controllable Data Structure Visualization System" Proceedings of Graph Drawing, Springer Lecture Notes in Computer Science 1027.