DISPLAYING DATASTRUCTURES AND ALGORITHMS IN A GRAPHICAL USER INTERFACE

By

Islam A. Ebeid

Submitted to the Faculty of the Department of Computer and Information Science

Arkansas Tech University

Thesis Proposal

In partial fulfillment of the requirements

For the degree of

MASTERS OF SCIENCE IN INFORMATION TECHNOLOGY

December 2012

THESIS COMMITTEE

Name	Dr. Larry Morell
Department	Computer & Information Science Department
Position	Professor
Office	Corley
E-mail	lmorell@atu.edu
Committee Role	Committee Chairperson

Name	Dr. Jerry Wood
Department	Computer & Information Science Department
Position	Professor
Office	Corley
E-mail	jwood@atu.edu
Committee Role	Committee Member

Name	Dr. David Middleton
Department	Computer & Information Science Department
Position	Professor
Office	Corley
E-mail	dmiddleton@atu.edu
Committee Role	Committee Member

TABLE OF CONTENTS

INTRODUCTION

An algorithm is a set of recalculated steps to solve a certain problem. In programming it is a procedural way to write a program to do a certain task like sorting some values or reversing a string of characters or finding the prime numbers in a string of numerical values.

“An algorithm is a human-readable description of steps needed to solve a particular problem” (Morell)

"The term algorithm is used in computer science to describe a problem solving method suitable for implementation as a computer program" (Sedgewick)

As you go on and read about algorithms you realize that implementing an algorithm is the basic skill that a programmer should acquire in his early days in programming. Any problem is resolved through certain steps, the human mind tends to visualize objects and symbols and tie them to the problem to be able to come up with a suitable set of steps to be followed.

Novice programming students usually have hard times visualizing changes in data structures as an algorithm proceeds. The main goal of this system is to provide a graphical representation of these changes in two ways: displaying the behavior of an existing algorithm and moving graphical objects

around the screen to simulate an algorithm and the system automatically generates the code.

Applications for the proposed system:

- Teaching algorithms for entry level programming students using Genesis programming language.
- Visually debugging algorithms written in Genesis.

DESCRIPTION

Displaying data structures and algorithms in a rich graphical user interface with animated graphical objects that directly and dynamically reflects the changes in a runtime environment is a topic that hasn't been researched much. Though some work has been done in this area since 1984 where the BALSA[1] algorithm animation system has been emerged all the way until the 2000's where jGrasp[2] visual debugging framework and Alice[3] appeared.

I am proposing a modern, solid and robust system that captures the changes in a runtime environment and displays it in a rich interface helping

[1] A software environment is described which provides facilities at a variety of levels for animating algorithms: exposing properties of programs by displaying multiple dynamic views of the program and associated data structures. (Brown and Sedgewick, A system for algorithm animation)
[2] "jGRASP" IDE provides *dynamic viewers* specifically intended to support fine-grained understanding of data structures and other objects. (Cross II)
[3] An educational software that teaches students computer programming in a 3D environment

learners digest the basic ideas of programming and algorithms.

It's not easy to implement these kinds of systems as it requires access to the runtime environment of an executing program. Also translating the manipulation of graphical objects to code is challenging. A set of manipulation primitives must be defined that will correspond to particular code blocks in the underlying language. The application of these primitives must be recognizable and the translation must be made. Some option for undoing previous operations will be implemented that will reset the screen as well as the underlying run-time representation. Using Genesis[4] as an open source and a simple programming language should make it easier to implement.

Genesis implements a single data structure: the list. Genesis represents each value with an object called a "Sticky Note". Accessing and displaying the list structure will be challenging because Genesis allows the same "Sticky Note" to appear in more than one list, lists can be nested, and they can be heterogeneous.

Using Java technology will give me the flexibility and robustness of a software design taking into consideration that I am using a relatively new framework introduced by Java 1.6 called JavaFX[5] which is able to produce a rich

[4] Genesis is an interpreted language implemented in Java; Genesis is therefore runs on any Java-enabled platform (Morell)

[5] The JavaFX platform is the evolution of the Java client platform designed to enable application developers to easily create and deploy rich internet applications (RIAs) that behave consistently

and modern interface, consistently tied to the underlying layer of controller classes.

Taking MVC[6] design pattern into consideration, a prototype design will be as follows:

Layer 1: Genesis Runtime Environment; A full implementation of the current version of Genesis that will be called to parse and interpret the input code or vice versa

Layer 2: A Model Layer; Some classes that represents real entities in Genesis runtime environment and a set of classes that contains the manipulation primitives that must be defined which will correspond to particular code blocks in the underlying language

Layer 3: A Controller layer which moderates the interaction between the view and Genesis runtime environment

Layer 4: A graphical user interface built using "**Java FX"** that interacts with the

---

across multiple platforms. Built on Java technology, the JavaFX platform provides a rich set of graphics and media API with high-performance hardware-accelerated graphics and media engines that simplify development of data-driven enterprise client applications. (Castillo)

[6] Model–View–Controller (MVC) is an architecture that separates the representation of information from the user's interaction with it. The model consists of application data and business rules, and the controller mediates input, converting it to commands for the model or view. A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as a pie chart for management and a tabular view for accountants. The central idea behind MVC is code reusability and separation of concerns. (Wikipedia)

controller classes through a XML based tag library which allows defining and developing interfaces in a fashionable way.

LITERATURE REVIEW

In 1984 Marc H Brown and R Sedgwick introduced the “BALSA Algorithm Animation System” (Brown and Sedgewick), one of the first attempts in this field. BALSA provided three capabilities; interpreting while the algorithm is executing, displaying and manipulating the interface and a shell for controlling BALSA primitives. BALSA at that time was used even outside of the computer field like in scientific laboratories.

In 1987 G. Michael Barnes and Gary A. Kind published a paper titled “Visual simulations of data structures during lectures” (Barnes and Kind). It described a microcomputer based software package for the graphical simulation of fundamental binary search tree algorithm. It was designed for an IBM PC. The paper then mentioned Computer Aided Instruction and how it is important and how it is an emerging area at that time. Also the paper mentioned reviewing the area of graphical simulations of fundamental data structures and their algorithms. They developed CABTO (Computer Animation of Binary Tree Operations). They also mentioned that there were two reasons for them to develop and use simulation oriented courseware as instructional aids in computer science: first,

many topics in computer science can be represented graphically. Second, graphical simulation can be effectively used to illustrate the dynamic nature of software structures over time.

In 1990 John T. Stasko wrote an article about his PHD thesis titled “TANGO: A Framework and System for Algorithm Animation” (Stasko), he defined algorithm animation as “The process of abstracting the data, operations and semantics of computer programs, and then creating animated graphical views of those abstractions” (Stasko). Tango supported color animations. Following BALSA it used a central message-server to interpret the incoming input code to trigger the corresponding animation routine.

In 1992 a paper was published by William Hibbard, Charles R. Dyer and Brian Paul at the “University of Wisconsin in Madison” titled “Display of Scientific Data Structures for Algorithm Visualization” (Hibbard, Dyer and Paul). They developed a system called VIS-AD or “Visualization for Algorithm Development”. Mainly they mentioned defining graphical depictions for data types defined in an algorithm through implementing the ability to display combinations of data objects in a “common frame coupled with interactive control of algorithm execution” (Hibbard, Dyer and Paul).

In 2004 Jan Lonnberg, Ari Korhonen and Lauri Malmi published an article titled “MVT – A system for visual testing of software” (Lonnberg, Korhonen and Malmi) at Helsinki University of Technology, Finland. The article presented a prototype for a debugging tool MVT “Matrix Visual Tester”, the programmer can use this tool to examine and manipulate a running program and its data structures. This tool combines many aspects; like visual algorithm simulation, high level data visualization and visual debugging, it also makes it easier to test, debug and understand the algorithm.

One of the latest developments in this field is jGrasp. It is very useful visual debugger that integrates to Java through three approaches: debugger, Workbench and text-based interactions.

Also “Alice” is one of the advanced software visualization tools. Alice is a new way of teaching object orientation to students. Basically you move real 3D objects around to generate code.

“In Alice’s interactive interface, students drag and drop graphic tiles to create a program, where the instructions correspond to standard statements in a production oriented programming language such as Java, C++ and C#” (Alice)

For non-programs Alice may seem quite complex, having a lot of configuration and tweaking to do the student may feel over whelmed.

METHODS OF DEVELOPMENT

In 2007 Java FX was introduced , later in 2012 Java FX 2.2 was introduced with the new FXML framework which allows the developer to describe your interface in an xml based markup language without the need to learn a new scripting language also they introduced a new IDE to develop FXML files graphically. Java FX supports all kinds of events that can be used in this kind of project. FXML will allow me to easily use a MVC “Model View Controller” design pattern approach.

Specifications shall be defined including functionalities, design and use cases. As a design prototype; the view will be written in FXML, the Controller in Plain Old Java and the model will be the classes that I will define to access and run Genesis. The model will hold also the classes that are responsible to interpret from Graphical objects to Genesis language code generation and vice versa.

CONCLUSION

In this project I want to develop a system simple enough for newbie programming students to understand basic algorithm operations and to visualize them. Combining all of these technologies shall produce a robust system capable

of fulfilling its main goal of helping programming students understand and visualize algorithms using Genesis programming language. In this project I will provide the ability to generate code by manipulating graphical objects on the screen. This capability was not implemented in any of the work that I reviewed earlier in this proposal. Having the experience of manipulating objects around the screen and generating code accordingly will definitely give the student a different angle on how to develop algorithms.

BIBLIOGRAPHY

Barnes, G. Michael and Gary A. Kind. "Visual Simulations of Data Structures During Lecture." SIGCSE '87 Proceedings of the eighteenth SIGCSE technical symposium on Computer science education. 1987. 267-276.

Brown, Marc H. "An introduction to Zeus audio visualization of some elementary sequential and parallel sorting algorithms." CHI '92 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (1992): 663-664.

Brown, Marc H and Robert Sedgewick. "A system for algorithm animation." SIGGRAPH '84 Proceedings of the 11th annual conference on Computer graphics and interactive techniques. 1984. 177-186.

Castillo, Cindy. What Is JavaFX? August 2012. <http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm#>.

Cross II, James H., et al. "Robust generation of dynamic data structure visualizations with multiple interaction approaches." ACM Transactions on Computing Education (TOCE) 9.2 (2009): 13.

Hibbard, William, Charles R. Dyer and Bryan Paul. "Display of scientific data structures for algorithm visualization." VIS '92 Proceedings of the 3rd conference on Visualization '92. 1992. 139-146.

Lonnberg, Jan, Ari Korhonen and Lauri Malmi. "MVT: a system for visual testing of software." AVI '04 Proceedings of the working conference on Advanced visual interfaces. 2004. 385-388.

Morell, Larry. Genesis From the Beginning. n.d. <http://www.cs.atu.edu/~morell/genesis/book/ch1.html>.

Sedgewick, Robert. Algorithms in Java. Vol. 1. Addison-Wesley Professional, 2003.

Stasko, John T. "TANGO: A FRAMEWORK AND SYSTEM FOR ALGORITHM ANIMATION." ACM SIGCHI Bulletin Homepage archive (1990): 59 - 60.

Wikipedia. n.d. <http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>.