CALIFORNIA STATE UNIVERSITY, SAN BERNARDINO

School of Computer Science
And Engineering

CSE 455

{AlgorithmA }; 2011

Software Requirement Specification
Second Iteration

CSE 455, Inc.
CEO: Dr. Concepcion
Project Manager: Anthony Thomas
Assistant Manager: Erwin Soekianto
Team Leads: Matthew Compean and Patrick Ridge

Table Of Contents

1 Introduction

The CEO of CSE 455, Inc. has requested further development of {AlgorithmA}; from the process begun by the CS455, Inc. of 2010. The following document lists and explains the further development options that are planned for possible implementation during 2011. Please review this document and verify that the proposed solutions are adequate and to expectations. Upon the approval from the CEO of the content contained within this document, development will commence immediately.

1.1 Purpose

The main goal of {AlgorithmA}; 2011 is to provide an interface to the academic community, in particular new students interested in the field of Computer Science and Engineering, to explore and learn about various algorithms. End users will have the ability to view a step-by-step animation for each of the algorithms that illustrate how they operate.

Several generations of {AlgorithmA}; have introduced mathematical concepts for end-users with mathematical knowledge and/or interest in order to provide a basic understanding of them. {AlgorithmA}; 2011 will continue to build upon these introductions, providing graphics where appropriate and refining the already existing explanations to better suit the end user, {AlgorithmA}; 2011 will address the following requests:

- Continue algorithm implementations
- Implement authoring system
- Update and add new documentation

1.2 Scope

{AlgorithmA}; 2011 is an end user application designed to provide a depth insight of Computer Science and Engineering areas to students and faculty to explore various mathematical algorithms. The main goal of {AlgorithmA}; is to provide step-by-step visualization of the various types of algorithms already implemented in the preceding {AlgorithmA}; versions.

{AlgorithmA}; 2011 will continue on forward with the progress of {AlgorithmA}; 2010 by continuing the re-implementation of algorithms from previous versions of {AlgorithmA}; that were not re-implemented by the CS455, Inc. 2010. Furthermore, {AlgorithmA}; 2011 will contain an authoring system that will allow the user to write algorithm's and see the result of their pseudo code animated in real time using the animation system implemented by {AlgorithmA}; 2010.

The following algorithms will be implemented this iteration:

Sorting Algorithms

- Heap Sort
- Shaker Sort
- Selection Sort

Gaussian Elimination

Authoring System

- Array Intensive
- Basic Sort
- Basic Programming

Pseudo Code Standardization

C++ Code Viewer

User Interface Improvement

1.3 Definitions and Acronyms

{AlgorithmA};

PattE's sister project, which stands for "Algorithm Animation."

Animation

A visual display that depicts selected algorithms being studied. The display is based on the Cartesian x-y coordinate system.

Computer Science

Study of information and computation.

CS

Acronym for Computer Science.

Deployment Diagram

Deployment diagrams serve to model the hardware used in system implementations and the associations between those components.

Fault

An abnormal condition or defect at the component, equipment, or subsystem level, which may lead to failure. It is informally linked with "bug."

Graphical User Interface

A GUI is a method of interacting with a computer through a metaphor of direct manipulation of graphical images and widgets in addition to text.

HTML

Acronym for Hypertext Markup Language, the authoring language used to create documents on the World Wide Web.

HTTP

Acronym for Hypertext Transfer Protocol. It is the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web Servers and browsers should take in response to various commands.

IDE

Acronym for Integrated Development Environment. IDEs assist computer programmers in

developing software.

Interface

The communication boundary between two entities such as software and its users.

Iteration

The repetition of a process.

Java

A high-level programming language developed by Sun Microsystems.

Model-View-Controller

A software architecture that separates an application's data model, user interface, and control logic into three distinct components so that modification to one component can be made with minimal impact to the others.

Mozilla Firefox

A free, cross-platform, graphical web browser that complies with many of today's standards on the World Wide Web. The most important standards for {AlgorithmA}; 2010 will be the W3C web standards.

Open Source Software

Software whose source code is published and made available to the public, enabling anyone to copy, modify and redistribute the source code without paying royalties or fees. Open source code evolves through community cooperation.

PHP

Self-referential acronym for PHP: Hypertext Preprocessor, an open source, server-side, HTML embedded scripting language used to create dynamic Web pages. In an HTML document, PHP script (similar syntax to that of Perl or C) is enclosed within special PHP tags.

Software Requirements Specification

An SRS is used to describe all the tasks that go into the instigation, scoping, and definition of a new or altered computer system.

SRS

An acronym for Software Requirements Specification.

W3C

An acronym for the World Wide Web Consortium.

World Wide Web Consortium

An international organization that works to define standards for the World Wide Web.

1.4 References

Wikpedia, http://en.wikipedia.org/

Bruegge, Bernd, and Allen H. Dutoit. Object-Oriented Software Engineering. 3rd ed. New Jersey: Pearson Prentice Hall, 2010.

W3schools, http://www.w3schools.com/jsref/default.asp.

"SRS_1st_Iteration_Final" CS455 Inc. Team. 2010.

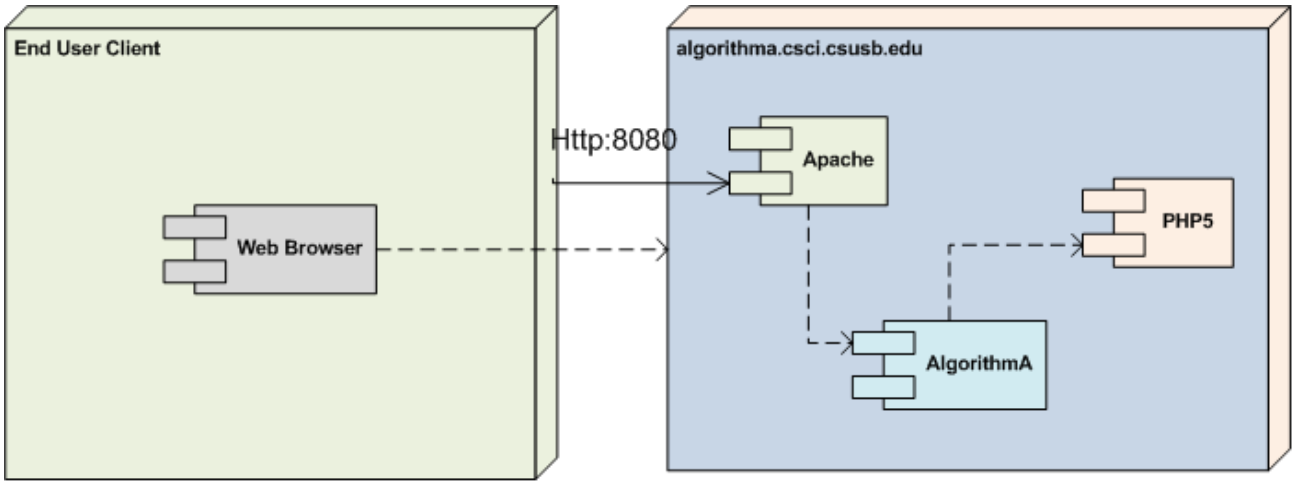
Fowler, Martin. UML Distilled Third Edition. A Brief Guide to the Standard Object Modeling Language. Boston: Pearson Education, Inc. 2009.

IEEE SRS Std. 830-1998, http://ieeexplore.ieee.org/Xplore/login.jsp?url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel4%2F5841%2F15571%2F00720574.pdf&authDecision=-203.

2 Overall Description

2.1 Product Description

2.1.1 System Interfaces



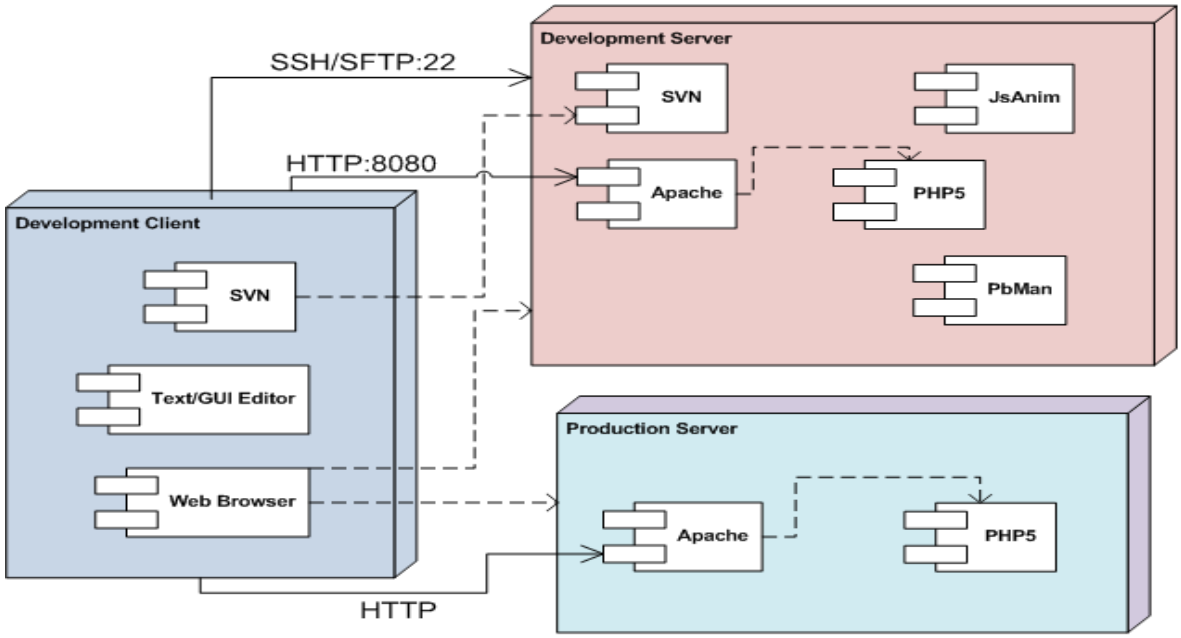
Deployment Server Diagram

The deployment diagram pictured above interprets the system interactions for the iteration 2 of {AlgorithmA}; 2011. The entire application is divided into a “front-end” and “back-end”.

The initial installation of the system development infrastructure will consist on the installation of two separate application servers running CentOS. The first server, a development server, will house all development applications including Subversion, Apache HTTP server, and PHP. The second server, production server, will be used as the final repository of the final product, which can be used for end users to connect to {AlgorithmA}; 2011. Pictured above is the deployment diagram planned by the server team. The purpose of a development deployment is to outline what services will be required in each Autonomous System.

When the iteration reaches completion, the Subversion repository for the iteration 2 will be exported to a production server. The project will be given a web address that users can access through HTTP. Browsers will continue to be supported. The production server will now only contain the latest milestone release. Milestones will be considered a completed iteration. We expect to give our final presentation from this server.

2.1.1.1 Development Server



Deployment Servers Diagram

Pictured above is the deployment diagram planned by the server team. The purpose of a development deployment is to outline what services will be required in each Autonomous System.

2.1.2 User Interfaces

{AlgorithmA}; 2011 consists of four basic user interfaces. The first interface will consist of the website itself and will serve to provide the navigation between all of the other interfaces in {AlgorithmA}; 2011. The second interface will be an animation interface, which will provide a graphical animation of an algorithm. The third interface will be a walk-through interface, which will consist of two panes, one for the animation of an algorithm, and the other for a walk-through of code that corresponds to the animation. The fourth interface will provide specific context-sensitive help for hovering and operating each animation. In addition, a structured non-technical file will be provided within {AlgorithmA}; 2011 explaining mathematical concepts for each algorithm.

2.1.2.1 Website Interface

The primary interface will consist of the website itself and will serve to provide the navigation between all of the other interfaces in {AlgorithmA}; 2011.

In iteration 2, the website interface will change, reflected in the following ways:

- A new {AlgorithmA}; 2011 logo.
- Compact design to maintain a visual preference for those with screen resolutions 1024x768 and up.
- Simple overall design so as to not overwhelm the user with information and action.

2.1.2.2 Walk-through Interface

The walk-through interface would consist of C++ code and Pseudo-code, which would be highlighted as the animations go along. The highlighter for the C++ code and Pseudo-code are in yellow color. The walk-through interface is located in the left bottom side.

2.1.2.3 Animations Interface

The animation interface is located next to the walk-through interface, which is in the right bottom of the web page. As the user click the dome button, then the animation would start, and show how the algorithms work. The animations contains of the circle or rectangle or square contains number, and it would move and animate to show the algorithms.

2.1.2.4 Authoring Interface

Authoring Interface is the place where the system allow the user to enter or edit the pseudo-code or C++ code, and see how the code works in animations. System also provides preset code for specific algorithms or concepts, and user would be able to edit them and see how the changes effect the animations and algorithms.

2.1.3 Software Interfaces

{AlgorithmA}; 2011 requires a web browser to be viewed. Any web browser may be used that follows the W3C web standards.

2.1.4 Communication Interfaces

{{AlgorithmA}; 2011 will be implemented using JavaScript. All client-side communication with the application shall use HTTP from the client's internet browser. Server-side communication will be controlled by the web-server.

2.1.5 Memory Constraints

{AlgorithmA}; 2011 will require 128 MB of RAM to be viewed.

2.1.6 Operations

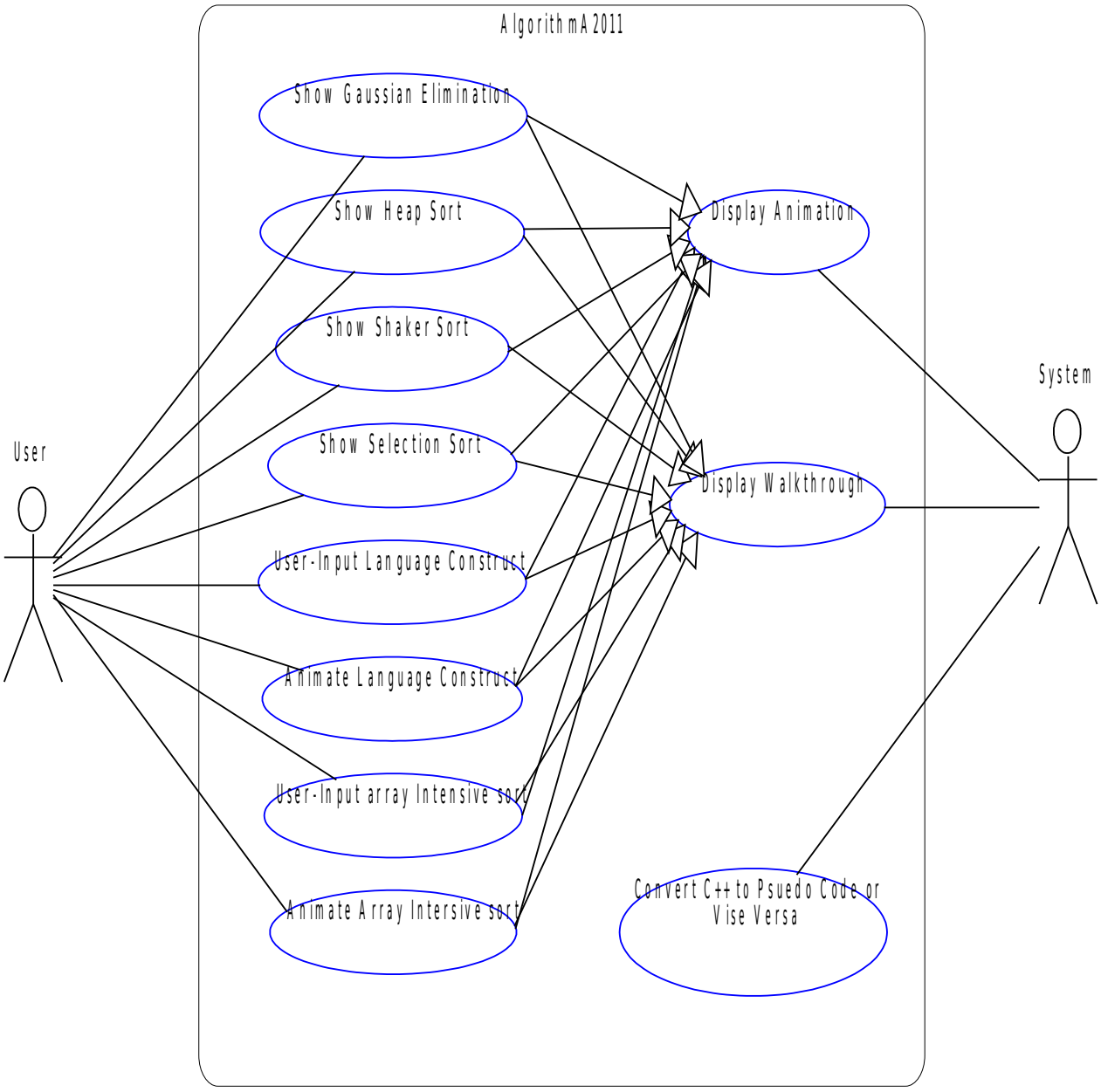
{AlgorithmA}; 2011 will be maintained during the winter quarter (January to March of 2011) and operated 24 hours a day, 7 days a week throughout the year. The maintenance will be conducted by CSE 455, Inc., and the hosting will be provided by the CSE Department of California State University, San Bernardino.

2.1.7 Site Adaptation Requirements

The on-campus workstations meet the current requirements of the JavaScript. Home workstations will need to ensure that the requirements in section 2.1.5 are met.

2.2 Product Functions

The use case diagram presented below illustrates which functions an end-user may choose to perform in {AlgorithmA}; 2011.



Small enhancements will be made to the currently existing animation system to further ensure that the user understands the meaning behind the animation. Such enhancements include:

1. Show the pseudo-code and C++ by default
2. Change the colors of the text to make it easier to read

2.3 User Characteristics

The target users of {AlgorithmA}; 2011 are students that attend CSE 201, 202, and 330 courses (Introduction to C++, Intermediate C++, and Data Structures respectively). {AlgorithmA}; 2011 is meant to visually display algorithms and common data structures in order to better facilitate a means of learning for these students.

2.4 Constraints

{AlgorithmA}; 2011 and algorithms module shall be functioning and deployed for presentation for the client by Finals week of the winter quarter, 2011. All program's faults must identified and fixed prior to the final demonstration, a presentation of its progress will be delivered during an analysis that is to occur 7 weeks into the project.

2.5 Assumptions and Dependencies

{AlgorithmA}; 2011 assumes that all previous modules and documentation are available and working at an acceptable level. It assumes that the MVC architecture implemented by the previous CS455, Inc., is functional and ready for inclusion of code necessary to facilitate {AlgorithmA}; 2011's core functionality.

3 Specific Requirements

3.1 External Interfaces Requirements

The logo designed for {AlgorithmA}; 2011 was created by David Cushman:

{AlgorithmA}; 2011 Logo

3.1.1 User Interfaces

When the user first visits the {AlgorithmA}; website, there will be a navigation bar on top, header, footer, and the main frame at the center of page. The navigation bar will have options of the different algorithms associated with computer science. Then each of those topics will feature a drop menu showing the algorithms associated with that topic.

3.1.1.1 General Interface

{AlgorithmA}; 2011 Main Webpage

When the end-user first visits the {AlgorithmA}; 2011 website. In addition, this page will contain the logo and a synopsis of {AlgorithmA}; 2011 capabilities and functions. Below is a concept of the introduction page of {AlgorithmA}; 2011.

3.1.2 Hardware Interfaces

a. Server side

The web application will be hosted on one of the departments' Linux servers and connecting to one of the school Internet gateways. The web server is listening on port 8080.

b. Client side

The system is a web based application; clients are requiring using a high speed Internet connection and using a up-to-date web browser such as Microsoft Internet Explorer and Mozilla Firefox.

3.1.3 Software Interfaces

3.1.3.1 JavaScript

JavaScript will be implemented throughout the website in order to display the correct feature the user requested. Once the user makes a request to see a specific walk-through or animation, JavaScript sends a request to provide that particular feature.

3.1.4 Communication Interfaces

{AlgorithmA}; 2011is designed to be viewed on any internet browser, provided that:

1. JavaScript is enabled
2. Images are enabled

Performance may vary slightly between browsers. However, the functionality of the site should not be impaired.

3.2 Functional Requirements

The functions specified in this section directly correspond to work that will be conducted on the {{AlgorithmA}; 2011 project. Given the large scope of the project, it may not be possible to complete all specified components in the time frame that is given. If this is the case, the {{AlgorithmA}; 2011 team will perform whatever measures are necessary to complete the specified components and render them in a functional state. This means constructing modules that follow the appropriate architecture while avoiding technical, artistic, and time consuming elements that are implied given the nature of the project. More on this is specified in the SPMP and SQAP documents.

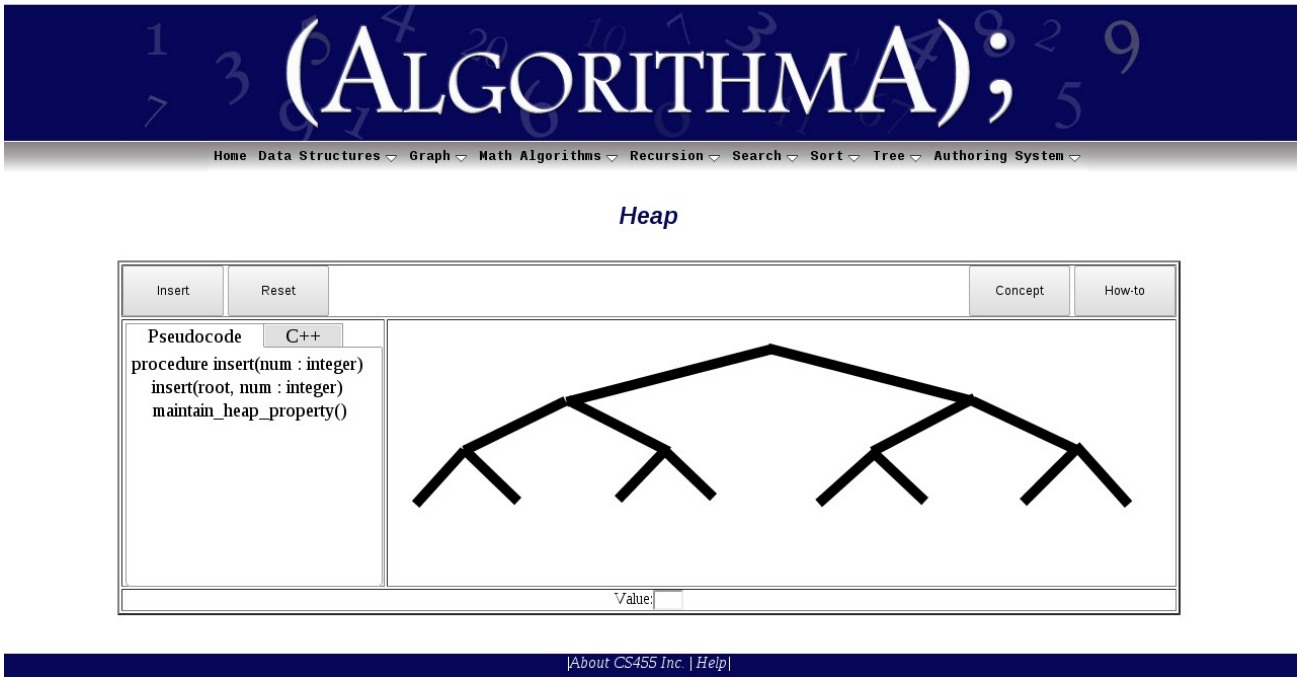
3.2.1 Sorting – Heap Sort

3.2.1.1 Overview

Heap sort is a comparison-based sorting algorithm, and is part of the selection sort family. Although somewhat slower in practice on most machines than a well implemented quicksort, it has the advantage of a more favorable worst-case O(n log n) run time.

3.2.1.2 Layout

Here's the layout of Heap sort user interface:



3.2.1.3 Functionality

Heap sort should should contain at the bare minimum

- Demo – Start the algorithm and let it run until finished
- Forward – Step through the algorithm line by line
- Back – Step back to the previous node.
- Reset – Reset to the initial layout when first loaded

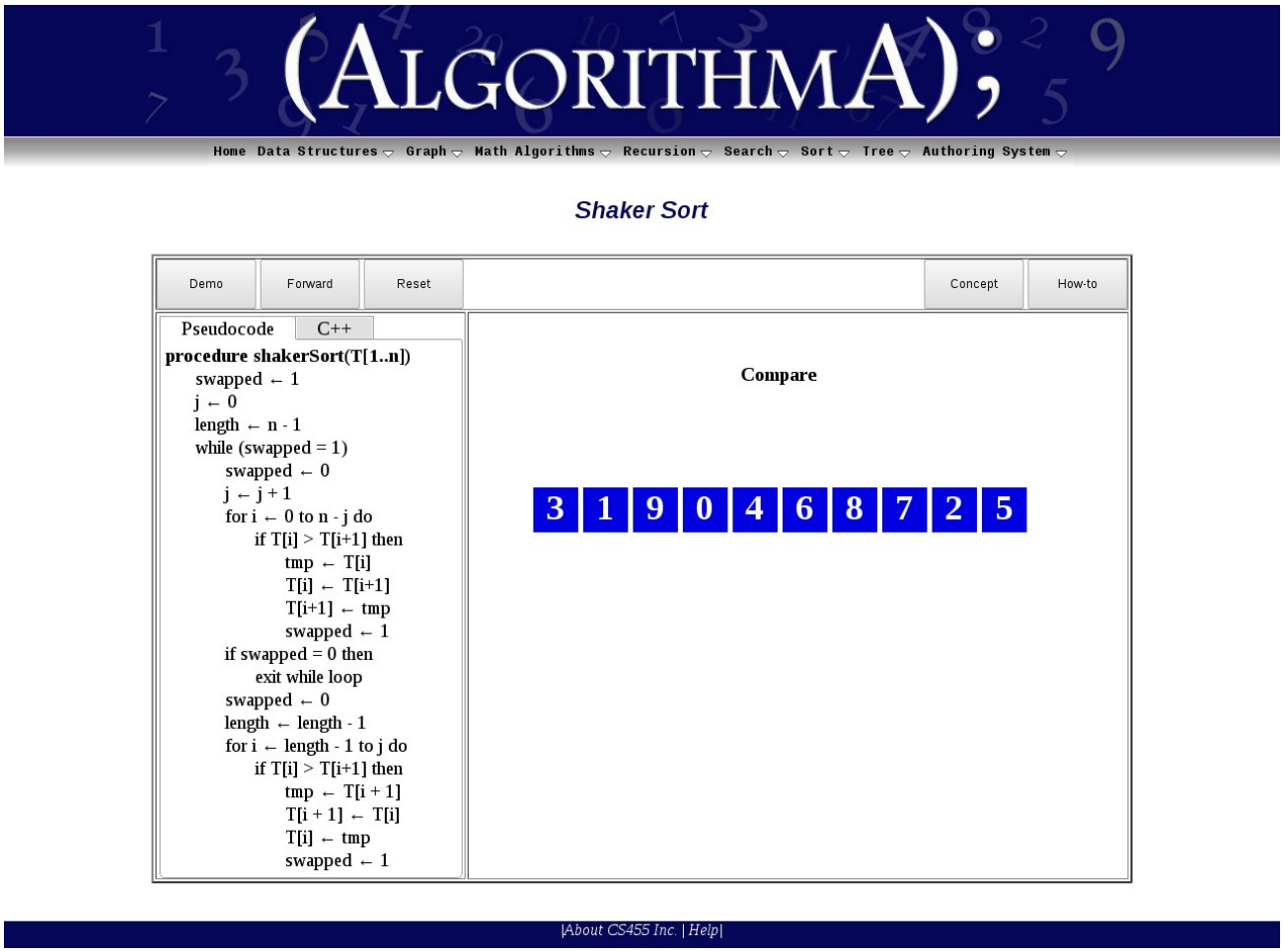
3.2.2 Sorting – Shaker Sort

3.2.2.1 Overview

Shaker sort is much like the bubble sort except that it sorts in both directions on each pass through an array. Shaker sort is marginally more difficult to implement than the bubble sort, however it solves the problem of 'turtles,' the event that a low number appears at the top of the array and thus takes longer to move to the bottom of the array, in bubble sort. Under best-case conditions (the list is already sorted), the shaker sort can approach a constant O (n) level of complexity. General-case is an abysmal O.

3.2.2.2 Layout

Here's the layout of Shaker sort user interface below:



3.2.2.3 Functionality

The shaker sort should contain at the bare minimum

- Demo – Start the algorithm and let it run until finished
- Forward – Step through the algorithm line by line
- Back – Step back to the previous node.
- Reset – Reset to the initial layout when first loaded

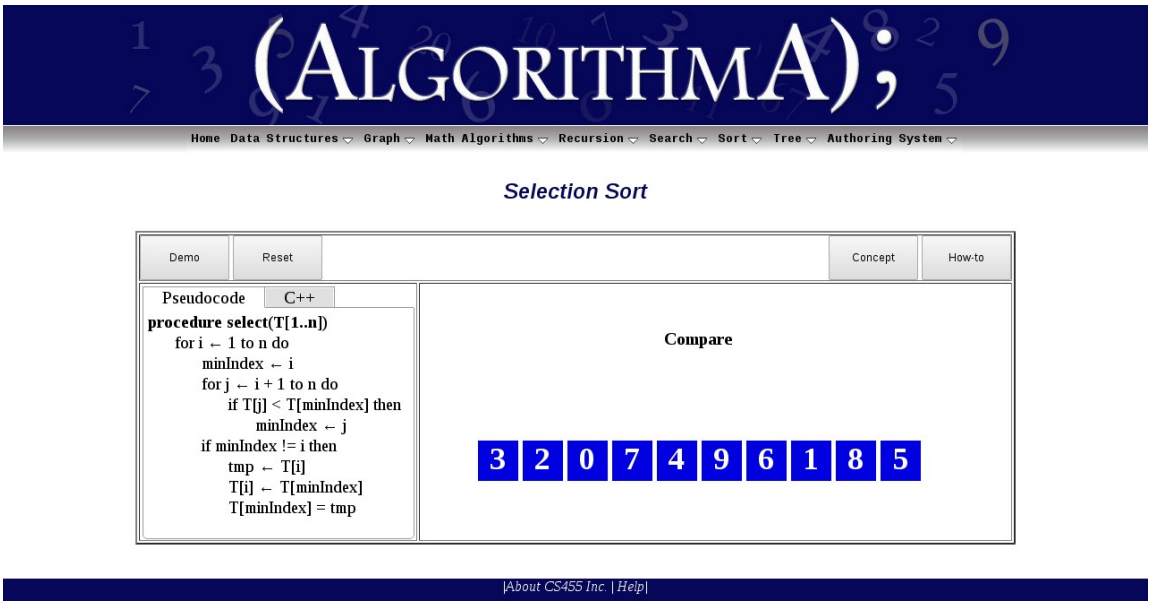
3.2.3 Sorting – Selection Sort

3.2.3.1 Overview

Selection sort is a sorting algorithm. With a O(n^2) complexity, selection sort is unfit to handle large lists, but is notable for its programming simplicity. It almost always outperforms bubble sort and gnome sort, and performs worse than insertion sort. Selection sort finds the minimum value in the list and then swaps it with the value in the first position. Continue to repeat those two steps for the remainder of the list while starting at the second position and advancing each time.

3.2.3.2 Layout

Here's the layout of Selection sort user interface:



3.2.3.3 Functionality

The selection sort should contain at the bare minimum

- Demo – Start the algorithm and let it run until finished
- Forward – Step through the algorithm line by line
- Back – Step back to the previous node.
- Reset – Reset to the initial layout when first loaded

3.2.4 Gaussian Elimination

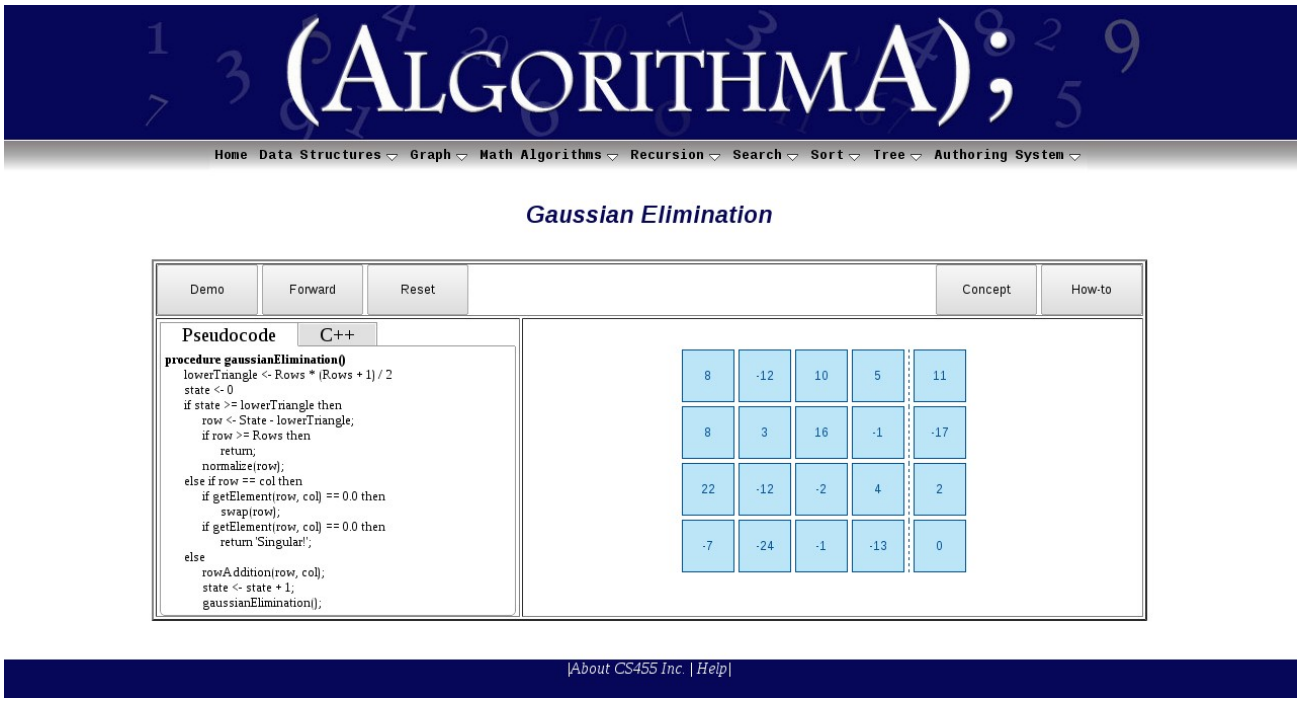
3.2.4.1 Overview

Gaussian elimination is an algorithm for solving systems of linear equations, finding the rank of a matrix, and calculating the inverse of an invertible square matrix. The method is named after Carl Fried rich Gauss but it was not invented by him.

Elementary row operations are used to reduce a matrix to row echelon form. Gauss–Jordan elimination, an extension of this algorithm, reduces the matrix further to reduced row echelon form. Gaussian elimination alone is sufficient for many applications, and requires fewer calculations than the -Jordan version.

3.2.4.2 Layout

Here's the layout of Gaussian Elimination user interface:



3.2.4.3 Functionality

The Gaussian Elimination should contain at the bare minimum

- Demo – Start the algorithm and let it run until finished
- Forward – Step through the algorithm line by line
- Back – Step back to the previous node.
- Reset – Reset to the initial layout when first loaded

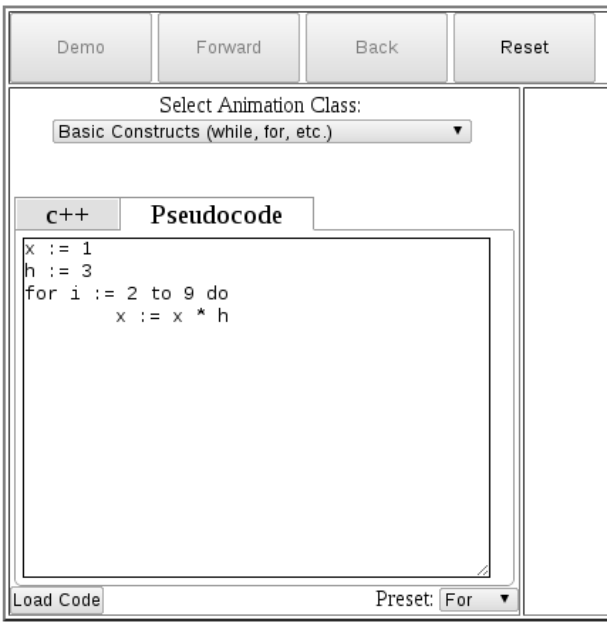
3.2.5 Pseudo Code and C++ Converter

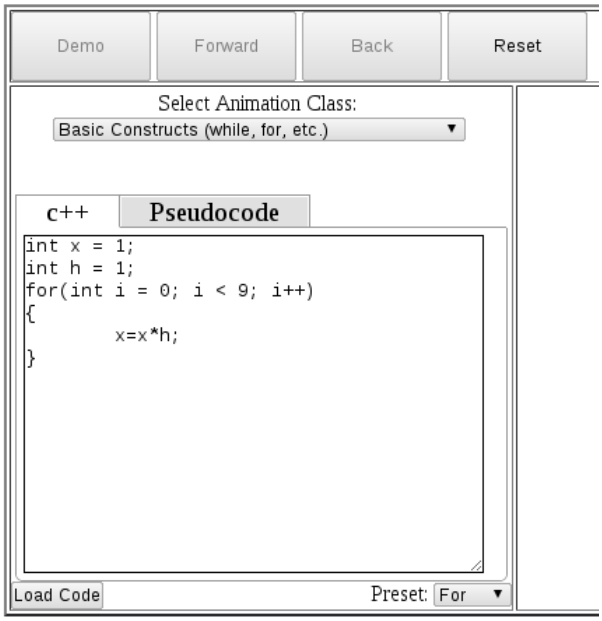
3.2.5.1 Overview

This feature allow users to view, enter or edit either Pseudo-code or C++ Code, to see the animations of the code they loaded. System would accept the C++ code or pseudo-code and convert to pseudo-code or c++ code, and compile the code to display the animations. This feature is useful because our target users, CSE 201 and CSE 202 students, are familiar with C++, so it would help them to understand better with the language they are familiar with.

3.2.5.2 Layout

Here's the layout of Pseudo-code and C++ Converter user interface:





3.2.5.3 Functionality

The Converter should only contain the bare minimum. This includes:

- Pseudo Code Tab – To view the pseudo-code
- C++ Tab – to view C++ code
- Load Code – To load the code for animations

3.2.6 Authoring System

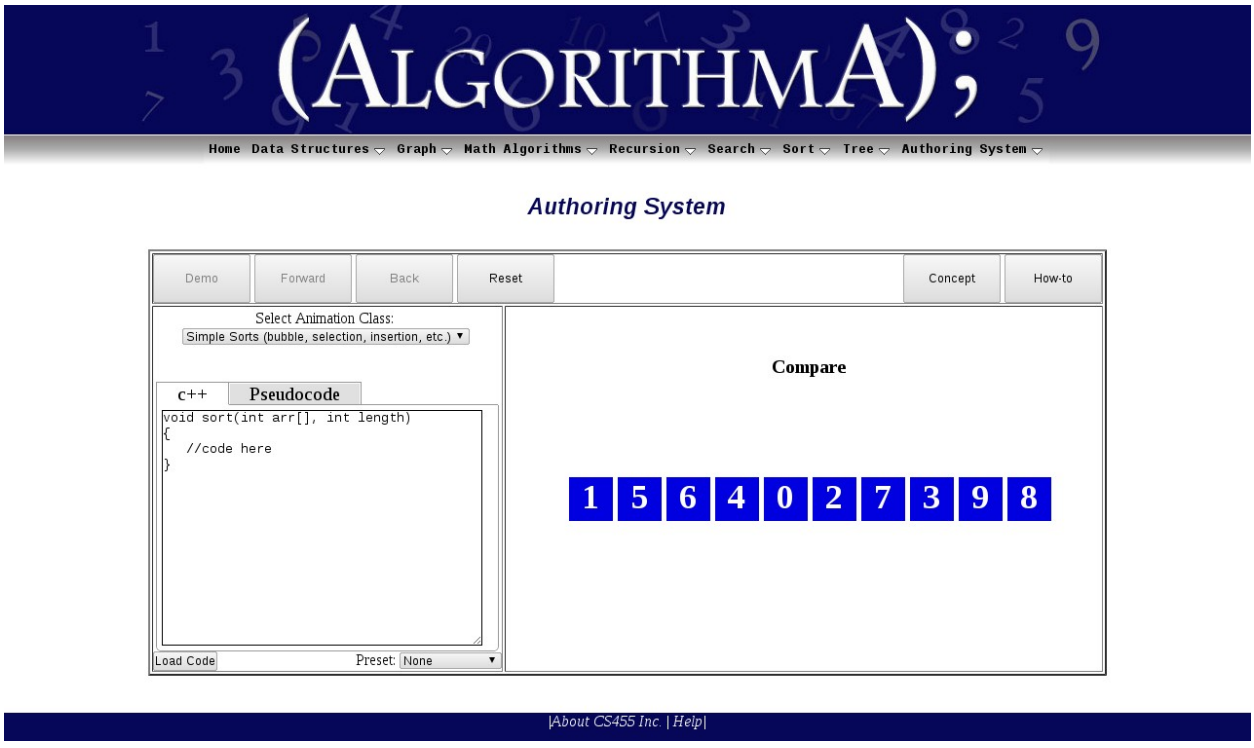
3.2.6.1 Overview

Authoring system is the innovation of {AlgorithmA} 2011, it would allow user to enter or edit the pseudo-code or c++ code, and see the changes in animations accordingly. However there would be some limitation on entering and editing the code to prevent the errors such as infinite loop, syntax error, etc. But this feature would allow user to understand the algorithms better.

This authoring system in this iteration would cover array intensive sort (Merge Sort, Quick Sort, etc), basic sort algorithms(Insertion sort, Bubble sort, Selection sort, etc) and basic programmings concept(for loop, while loop, etc).

3.2.6.2 Layout

Here's the layout of Authoring system user interface:



3.2.6.3 Functionality

The Authoring system should only contain the bare minimum. This includes:

- Pseudo Code Tab – To enter, edit or view the pseudo-code
- C++ Tab – to enter, edit or view C++ code
- Load Code – To load the code for animations
- Select Animations Class – To select type of animations
- Preset code – To load preset code to be displayed
- Demo – Start the algorithm and let it run until finished
- Forward – Step through the algorithm line by line
- Back – Step back to the previous node.
- Reset – Reset to the initial layout when first loaded

3.3 Performance Requirements

Our performance goals are to keep the high performance of the {AlgorithmA}; 2010 version. As the project is continuing in Java Script using previously created architecture, there should be no reason why the performance of the software should degrade.

3.4 Design Constraints

We are picking up from where the {AlgorithmA}; 2010 team has left off. Therefore, we are constrained by the software architecture that they built when re-engineering the project. We must also constrain ourselves to the Coding Standards document.

3.5 Software System Attributes

Much like the 2010 {AlgorithmA}; team we aim to keep the same standards of coding and documentation, especially documentation of the Authoring System to be implemented in the second iteration of this project.

3.6 Other Requirements

{AlgorithmA}; 2011 must remain supportable: It has to be maintained well enough to be smoothly taken over by the next CSE 455, Inc. {AlgorithmA}; 2011 must be easy to use for instructors and students interested in learning algorithms. {AlgorithmA}; 2011 must maintain its present reliability.

Moreover, {AlgorithmA}; 2011 must continue to run on the computer science school web site well after development for future project groups to access. It should be available 24 hours a day, 7 days a week, and 365 days a year. The only time when the site will become unavailable is during short maintenance periods.

3.7 Documentation

The same level of documentation is planned to be maintained from the previous version of {AlgorithmA};