White Paper for Project CHAD

September 23, 2003 – Fall 2003

CS W4115: Programming Languages and Translators
Professor Stephen A. Edwards

CHAD Group Members:
*****Haronil Esteves – he99@columbia.edu
Diana Jackson – dj2115@columbia.edu
Catherine MacInnes – cam240@columbia.edu
Adam Rosenzweig – amr152@columbia.edu

*****denotes group leader

Language Overview

The instruction of computer science is often hampered by the abstract nature of programming. Data and data structures, while intuitively, graphical structures, are represented in programs with words making it difficult for students to grasp the concepts involved. This problem is made worse when the data structures are manipulated to perform operations such as sorts and searches. Instructors are often reduced to drawing crude illustrations on black boards, or even using students to portray array elements. It is our belief that many instructors would appreciate a way to easily illustrate these data structures and algorithms in the context of a programming language. We hope to implement some of the more common structures in the context of a language, which could in principle be extended to include other structures or algorithms.

Goal

The goal of CHAD is to allow instructors to easily create animations that display basic data structures and the common algorithms associated with them. The structures we plan to include in our language are queues, array, linked lists, stacks and trees. Through these animations, the process of explaining and demonstrating the use of these data structures will be facilitated. Instructors will be supplied with primitives for these data structures and will be able to specify their integer or string data. Instructors will also be provided with a basic set of instructions for each data structure, which will allow them to write search and sort algorithms that will then be illustrated by CHAD. This language will also include a trace feature where a small subset of the variables can be highlighted throughout the algorithm to show clearly how a single piece of data is manipulated. We believe that through the use of this language, instructors will be provide a more effective way of helping students grasp basic computer concepts; therefore providing the students with a better understanding of what their own programs are actually doing.

Easy to Use

As previously stated, one of the main goals of CHAD is to make the illustration of data structures and algorithms more easily depicted. To this end we will implement a syntactically simple language. By limiting the algorithms that can be implemented, we will be able to write an extremely simple language that will be interpreted visually. We desire these animations not only to provide a better understanding of data structures, but also for the animations to be easy to create. Below are a few examples of the tentative syntax that we will use:

```
[array | queue | stack] [name_of_array][#_of_elements][elements (separated by spaces)][outline_color][fill_color][text_color]

[name_of_array] [sortAZ | sort ZA]

[name_of_queue] [enqueue element_to_add | dequeue]

[name_of_stack] [push element_to_add | pop]
```

Specific Example of a CHAD script:

```
array myArray 4 Catherine Haronil Adam Diana black yellow blue
myArray sortAZ
```

With the execution of each of the two previous commands, the animation will display the following two frames: (frames of each animated step of the sorting algorithm not shown)

Initial Frame

Catherine
Haronil
Adam
Diana

Final Frame

Adam
Catherine
Diana
Haronil

The example showing some two commands demonstrates how easy it will be to create this script. In a limited time, the syntax structure to create data structures will be intuitively easy to remember. We are still in the process of determining all of the commands we would like to implement, as well as coming up with a suitable syntax to be used for the more difficult data structures such as trees and linked lists. However, they all will follow a similar structure.

Implementation

CHAD will be implemented as an interpreted scripting language. The text files generated will be fed directly into a Java program, which will interpret them. Using the Java.Swing package will animate the data structures and algorithms generated from the CHAD program will be animated. Writing in CHAD will not require any knowledge of the implementation.

Portability

The scripts written in CHAD will be highly portable. The only thing needed to write them is a text editor; the only thin needed to run them is a single Java program and Java Virtual Machine.

Summary

To summarize, we feel that CHAD will be an amazing enhancement to computer science education. Instructors will be able to easily create animations to help students visualize a concept, and students will grasp the idea behind data structures and common algorithms more readily.