ANIMATIONS AND INTERACTIVE MATERIAL FOR IMPROVING THE EFFECTIVENESS OF LEARNING THE FUNDAMENTALS OF COMPUTER SCIENCE

by

William S. Gilley

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Computer Science

Osman Balci, Chairman
N. Dwight Barnette
Richard E. Nance

May 8, 2001
Blacksburg, Virginia

Keywords: animations, computer-based instruction, courseware, CS1, educational technology, Flash, interactive learning, Introduction to Computer Science, Java applets, multimedia.

ANIMATIONS AND INTERACTIVE MATERIAL FOR IMPROVING THE EFFECTIVENESS OF LEARNING THE FUNDAMENTALS OF COMPUTER SCIENCE

by William S. Gilley

Committee Chair: Osman Balci
Computer Science

ABSTRACT

Due to the rapid proliferation of the World Wide Web (WWW) in recent years, many educators are now seeking to improve the effectiveness of their instruction by providing interactive, web-based course material to their students. The purpose of this thesis is to document a set of eight online learning modules created to improve the effectiveness of learning the fundamentals of Computer Science. The modules are as follows:

1. Algorithms - Definition and specification of algorithms, with a comparison and analysis of several sorting algorithms as examples.
2. Artificial Intelligence - Overview of current applications in this discipline.
3. Data Structures - Explanation of basic data structures, including an introduction to computer memory and pointers, and a comparison of logical and physical representations of commonly used data structures.
4. Machine Architecture - Explanation of data storage, gates and circuits, and the central processing unit.
5. Number Systems - Discussion of number representation and arithmetic in number systems other than the decimal number system, with a focus on binary numbers and binary arithmetic.
6. Operating Systems - Explanation of the purpose of operating systems and the major components that make up an operating system.
7. Programming Languages - Explanation of the fundamental concepts in procedural programming languages.
8. Software Engineering - Introduction to software life cycle models and an overview of the procedural and object-oriented paradigms.

Each module consists of a set of lessons and review questions written in HyperText Markup Language (HTML). Embedded in these pages are various interactive components implemented as Flash animations or Java applets. The modules currently reside on the Computer Science courseware server of Virginia Polytechnic Institute and State University (Virginia Tech) and can be viewed at the following WWW site: http://courses.cs.vt.edu/csonline/.

ACKNOWLEDGEMENTS

First of all, I want to thank my Lord and Savior Jesus Christ who is my constant refuge and strength (Psalm 62:7-8). Second, I want to thank my mother and father for their wisdom in sending me to Virginia Tech to study and their help throughout my years of graduate school. Third, I want to thank Dr. Balci for working as my advisor and giving me much wise counsel during the past two years of graduate school. I also want to thank Dwight Barnette and Dr. Lee for their ideas and guidance as I designed various online modules. Fourth, I want to thank Rob Adams and Emre Tunar who actually programmed most of the animations and applets I designed. Fifth, I want to thank Dr. Nance for sitting on my committee at the last moment and for his insightful criticisms. And last, I want to thank the Bakers who graciously had me in their home nearly every week to eat so that I would not die from my diet of Ramen noodles.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 STATEMENT OF THE PROBLEM

Due to the rapid proliferation of the World Wide Web (WWW) in recent years, many educators are now seeking to improve the effectiveness of their instruction by providing interactive, web-based course material to their students. Why is this? First, the WWW allows educators to provide instruction beyond the bounds of the classroom. By providing supplemental learning material through the WWW, students can review information and learn at their own pace outside of class. Such material also gives students a non-intimidating environment in which to explore and answer questions that may not be practically answered in a large class with rigid time constraints.

Second, the WWW supports the development of interactive teaching material in which students "learn-by-doing" [Schank 1994] rather than just reading and memorizing. Through the use of multimedia development tools such as Flash and Java, educators can now create material that engages students to learn through active participation rather than passive absorption.

Third, the WWW provides a convenient means for reuse and sharing of educational materials. This change opens the possibility for educators to reuse existing materials rather than redeveloping something that already exists. It also allows educators to develop new materials and make these immediately available to others [Tsichritzis 1999].

Of course, it would be naive to assume that recent changes in technology have resulted only in new opportunities and introduced no new challenges. The reality is that many difficult challenges exist to developing beneficial, interactive learning material. One such challenge is the enormous development overhead associated with creating quality educational materials that promote interaction. Pilgrim *et al*. [1997] estimate that "material corresponding to one hour of student interaction may take up to 200 person-hours to develop." Another challenge is the development of material that truly exploits the potential of interactive multimedia. While posting existing slides and lecture notes on the WWW is easy, just making this information available by a new medium is not likely to increase the effectiveness of learning.

This thesis documents the work of a year-long project funded by the Center for Innovation in Learning at Virginia Polytechnic Institute and State University (Virginia Tech). The purpose of this project was to develop interactive course material for improving the effectiveness of learning the fundamentals of Computer Science by utilizing the unique advantages of multimedia and the WWW and attempting to mitigate their present challenges. During the course of the project, the project team developed eight web-based learning modules that span the fundamental topics of Computer Science. The modules are as follows:

1. Algorithms - Definition and specification of algorithms, with a comparison and analysis of several sorting algorithms as examples.
2. Artificial Intelligence - Overview of current applications in this discipline.
3. Data Structures - Explanation of basic data structures, including an introduction to computer memory and pointers, and a comparison of logical and physical representations of commonly used data structures.
4. Machine Architecture - Explanation of data storage, gates and circuits, and the central processing unit.
5. Number Systems – Discussion of number representation and arithmetic in number systems other than the decimal number system, with a focus on binary numbers and binary arithmetic.
6. Operating Systems - Explanation of the purpose of operating systems and the major components that make up an operating system.
7. Programming Languages - Explanation of the fundamental concepts in procedural programming languages.
8. Software Engineering - Introduction to software life cycle models and an overview of the procedural and object-oriented paradigms.

Each module consists of a set of lessons and review questions written in HyperText Markup Language (HTML). Embedded in these pages are various interactive components implemented as Flash animations and Java applets. The eight modules now reside on the Computer Science courseware server of Virginia Tech and can be viewed at the following WWW site: http://courses.cs.vt.edu/csonline/.

1.2 STATEMENT OF THE OBJECTIVES

Four main objectives guided the development of the learning modules: (1) provide access to the modules via the WWW, (2) teach the topics in an interactive, animated manner, (3) reuse existing material, and (4) implement independent, extendable modules.

1.2.1 Providing Access to the Modules via the WWW

As stated earlier, the WWW affords unique opportunities to increase the effectiveness of learning. Naps *et al.* [1997] discuss several particular advantages to using the WWW as a medium for delivering interactive material. These include remote accessibility to materials, support for distance learning, the development of system independent applications, and the use of hypertext, an inherently interactive medium. With these advantages in mind, we chose from the start of the project to design learning material that would be web-based, that is, embedded in HTML pages.

1.2.2 Teaching the Topics in an Interactive, Animated Manner

Presenting material in an interactive manner is an effective technique for making learning memorable. A good example of this principle is the ELIZA program, an early program designed to mimic human intelligence through conversation. In his introductory text to Computer Science, Brookshear [1997] cites this example and then attempts to describe the functionality of the program. His use of this illustration is motivated by the philosophical question of whether computers can possess real intelligence. While describing the ELIZA system does give students a general idea of its purpose, it is not nearly as memorable as using ELIZA (see Figure 1.1). For this project, we espoused the latter approach to addressing the philosophical question of computer intelligence so that students can decide for themselves whether computer intelligence can match human intelligence.

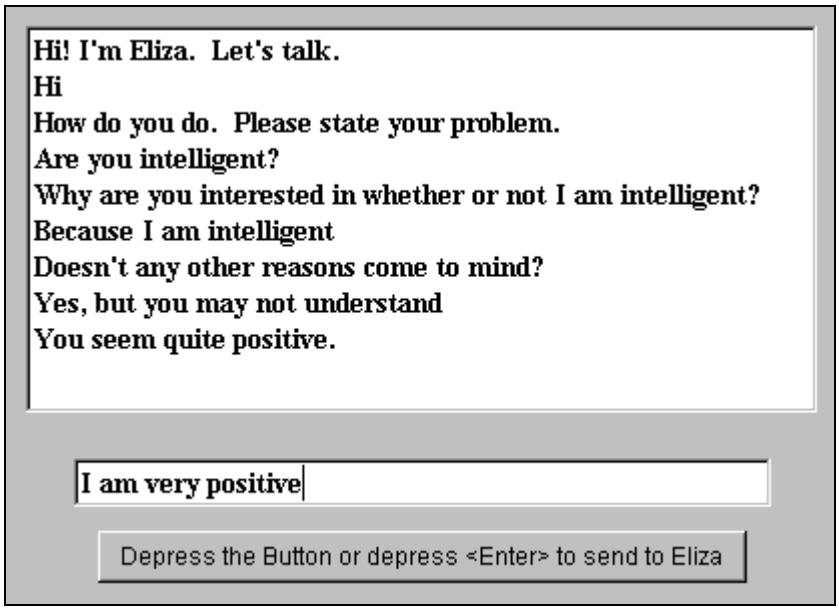


Figure 1.1 Interactive use of the ELIZA applet

Presenting material in an animated manner is also an effective technique for improving learning. Wetzel *et al.* [1994] describe a variety of characteristics that make animations uniquely effective for instruction. First, animations allow an instructor to emphasize the important parts of a subject by removing unnecessary and distracting details. Second, animations have the ability to visually present subjects and ideas that would be difficult to illustrate in the real world. Third, animations can aid in discrimination by presenting cues that direct the viewer's attention to salient points of the presentation. Throughout the various animations in this project, we have attempted to make use of these characteristics in order to make learning more effective.

1.2.3 Reusing Existing Material

Because of the large amount of time needed to develop quality materials, reusing existing materials was a crucial objective. The philosophy that guided this principle was "Don't reinvent the wheel!" As a result, many of the Java applets we embedded in the modules were included with the permission of the author and reused to illustrate various topics. This approach helped significantly to mitigate the time challenge of developing quality educational material. While most of the borrowed applets required some modifications, these changes were significantly less time consuming than writing a comparable applet from scratch. A good example of reuse in this project is the Eight-

puzzle applet (see Figure 1.2) in the Artificial Intelligence Module. Locating and modifying this applet required approximately 20 person hours while developing a similar applet from scratch would require anywhere from 50 to 100 person hours.

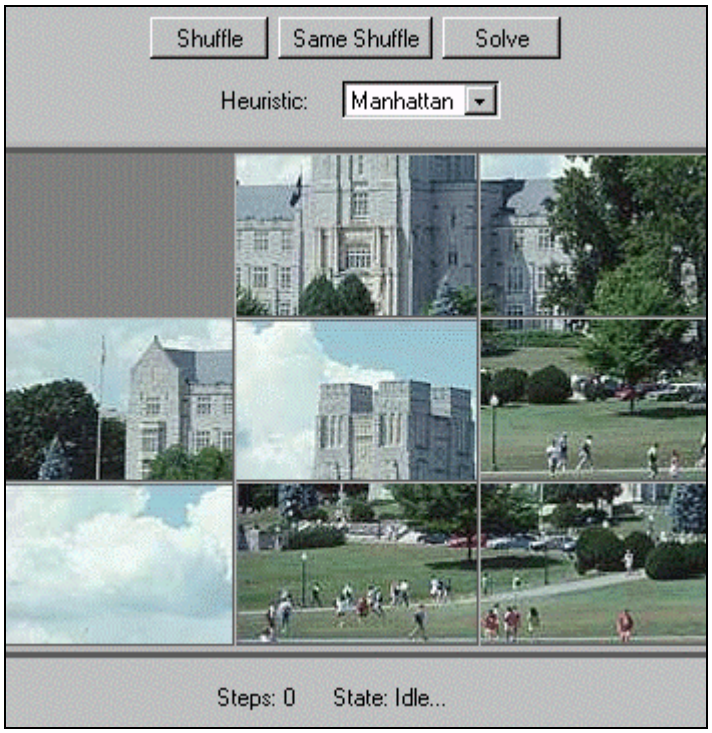


Figure 1.2 Modified eight-puzzle applet

1.2.4 Implementing Independent, Extendable Modules

The objective of implementing independent, extendable modules was another strategy for mitigating the challenges of developing interactive material. While the previous objective focused on reducing our development costs, this objective focuses on reducing the costs of future developers by striving for a high level of cohesion in each module. This allows other educators to download and install individual modules on their own web servers. By reusing these modules, educators can immediately benefit from the development time invested during this project with minimal added investment of their own. In addition, the high level of cohesion within modules allows for easier development and integration of future modules.

1.3 OVERVIEW OF THE THESIS

This thesis consists of five chapters. Chapter 1 discusses the advantages and challenges of developing interactive learning modules for the WWW, discusses the four objectives we used in developing our modules, and provides an overview of the thesis. Chapter 2 surveys several related works that are currently accessible on the WWW. Chapter 3 describes the hardware and software tools used to create the online modules. Chapter 4 documents the eight modules we created with each subsection corresponding to one module. The subsections state the learning objectives for the module and describe the lessons and interactive components embedded in the module. Chapter 5 summarizes the project, discusses some preliminary student responses to the modules, and makes recommendations for future work.

2 RELATED WORKS

In this chapter, we present three areas of related work. First, we examine several college-level resources outside the field of Computer Science that provide multimedia learning material. These serve as general examples of what is currently being done to use technology and the WWW to improve the effectiveness of learning. Second, we examine several multimedia college-level resources within the field of Computer Science. Third, we describe four archives of multimedia resources for Computer Science educators. These archives provide a valuable resource for educators who desire to reuse existing interactive material to improve the effectiveness of their own instruction.

2.1 MULTIMEDIA LEARNING MATERIAL FOR NON-CS COURSES

The following four web sites are examples of animations and applets that were designed to improve the effectiveness of learning various college-level topics. The first three sites are online resources created to accompany various college textbooks. The last site is a collection of applets for teaching topics in Statistics to students from various social science disciplines.

2.1.1 Essential Chemistry Flash Animations

This web site [Chang 2000] provides a collection of 18 interactive Flash animations for teaching various concepts in chemistry. The animations include narration that introduces the topic and instructs the student on how to interact with the animation. The animations range in size from 175 KB to 1.1 MB, however, since the Flash format is streamed, even the largest animation begins playing with very little wait. Students interact by clicking on buttons at the bottom of the animation to jump to various scenes. For example, one animation discusses radioactive decay by demonstrating how scientists detect radioactive particles that are emitted by various elements. Students control the animation by clicking buttons that correspond to the elements discussed in the animation. Each of the animations at this site correspond to a particular chapter or topic in Robert Chang's chemistry textbook, *Essential Chemistry*. Chang's support material also includes some labs and quizzes, however, this material is not directly accessible from the animations.

2.1.2 Explorations QuickTime Animations

This web site [Arny 2000] provides a collection of 43 QuickTime animations for illustrating various concepts in astronomy. The animations include narration which is also displayed as HTML text below the animation. These animations are quite large in size ranging from 500 KB to 15 MB. They are also non-interactive and tend to be short in length. For example, the largest animation has a running time of 30 seconds and covers three sentences of narration. Each animation corresponds to a particular chapter or topic in Thomas Arny's astronomy textbook *Explorations: An Introduction to Astronomy*. Arny's web site also includes some labs and quizzes, however, this material is not directly accessible from the animations.

2.1.3 Environmental Science RealMedia Animations

This web site [Enger and Smith 2001] provides a collection of 23 RealMedia animations for illustrating various concepts in environmental science. The animations include narration and can be accessed from online review quizzes. The size of the animations ranges from 500 KB to several megabytes, however, the authors include a disclaimer that these animations should be downloaded from a suitable high-speed network connection. The animations are not interactive, however, the review quizzes ask questions that cover the material presented in the linked animation. Each animation corresponds to a chapter or topic in Enger and Smith's textbook *Environmental Science: A Study of Interrelationships*.

2.1.4 SABLE Applets

The Statistics Activity-Based Learning Environment (SABLE) [Shaffer 2000] provides an collection of 11 tutorials with interactive Java applets that teach students basic concepts from Statistics. The authors describe SABLE as a learning environment in the following manner:

> The student completes tasks that lead to an understanding of statistics principles and their application to social sciences data. Each tutorial employs visualizations of data and relationships and also allows the student to download and interact with data sets collected in the General Social Survey and others. The Learning Environment experience culminates in the student's use of the visualizations to analyze and draw conclusions from real data.

Some of the topics covered by the tutorials include measures of central tendency and dispersion, hypothesis testing, T-distributions, analysis of variance, and linear regression. Each of the statistics tutorials are linked to a glossary of terms that is displayed in a frame at the bottom of the screen. Students interact with the applets in a wide variety of ways. For example, one applet which illustrates the concepts of median and mode displays two histograms of data. Students are then given the following instructions:

> You should see two copies of the histogram. The upper histogram allows you to drag the red vertical line to help locate the median. Numbers on either side of the red line show you how many values exist above and below the line. The lower histogram allows you to move a triangle within the range of the distribution which acts like a fulcrum for a see-saw. The mean is located at the point where the histogram is balanced. Use these tools–the red vertical line and the fulcrum–to find the median and mean of the data.

As the students drag the line and the fulcrum with the mouse, they can see the meaning of median and mode visually.

2.2 MULTIMEDIA LEARNING MATERIAL FOR CS COURSES

The following five web sites are examples of collections of animations and applets that were designed to improve the effectiveness of learning various Computer Science topics.

2.2.1 PACER

Personally Active Computing Exploration Resource (PACER) [Palakal 1997] is a WWW instructional supplement to the introductory computing courses at the Indiana University-Purdue University at Indianapolis Department of Computer Science. PACER consists of a set of Java applets to illustrate concepts in three areas of computer science: computer logic, algorithms, and data structures. The applets are presented with minimal instructions and no explanation of the topic being illustrated. The site also appears to be incomplete since some of the topic links lead to pages stating that no applets exists for the selected topic.

2.2.2 The HyperLearning Center

The HyperLearning Center [Denning and Menascé 1998] at George Mason University provides a set of workbenches to "demonstrate advanced concepts from

computer systems." The workbenches consist of 16 Java applets illustrating the following topics:

- Performance of operating systems,
- Virtual memory,
- Synchronization in centralized operating systems,
- Synchronization in distributed systems,
- Election in distributed systems, and
- Process scheduling.

Each applet is accompanied by a brief introduction to the topic and instructions on using the applet. The HyperLearning center also provides some tutorial modules on various computer science topics. These modules are mainly text and graphics, however, some do link to relevant workbenches in order to illustrate the topic interactively.

2.2.3 Operating Systems 85349

Operating Systems 85349 is a course offered by the Faculty of Informatics and Communication at Central Queensland University (CQU). The course is taught both online and at the university. The web site for the course [CQU 2000] contains some audio lectures, complete lecture slides, and seven Flash animations. The animations illustrate topics such as the instruction cycle, the process life cycle, and semaphores. They are indexed according to the course textbook and are also directly accessible from the lecture slides. Each animation includes a set of controls similar to a CD player that allow viewers to stop, pause, skip ahead one frame, skip back one frame, or run the entire animation.

2.2.4 The Analytical Engine Online

The Analytical Engine Online [Decker and Hirshfield 1998] is a collection of 10 online labs that supplement the textbook *The Analytical Engine*. The labs are interactive exercises corresponding to the topics covered in the textbook. Most of the interactive content centers around six Java applets: a CPU simulator, a Turing machine, a simple inference engine, a logic simulator, a programming code parser, and a haiku poetry generator. The parser and CPU simulator are nicely integrated such that code generated

from the parser applet can be executed on the simulator. The site also includes review quizzes and an interactive game that are implemented with JavaScript.

2.3 ARCHIVES OF MULTIMEDIA RESOURCES

In addition to the web sites reviewed above, several archives of interactive resources are available on the WWW. The first is the Computing Science Teaching Center (CSTC) [CSTC 1999]. The mission statement of the CSTC states that "CSTC is an internet-based repository of peer-reviewed teaching resources for computer science educators" and the web site exists as "a framework for developing and distributing three kinds of teaching materials: visualization tools, computing laboratories, and multimedia resources."

The second archive is the ACM Journal on Educational Resources in Computing (JERIC) [JERIC 2000]. According to the journal charter, JERIC is

> an electronic publication providing access to high quality, archival resources suitable for use in support of computing education. Resources include scholarly articles with wide applicability and potential impact as well as multimedia and visualization resources, laboratory materials and other materials of practical use in support of learning in the computing sciences.

The third archive is the Java Applet Rating Service (JARS) [JARS 2001]. This rating service is composed of over 100 volunteers around the world who review Java applets and applications. JARS maintains a database of program submissions that point to resources on the WWW. This resource is particularly useful since it gives a reasonable judgment of the value of existing interactive resources.

The final two archives are lists of links to other resources on the WWW. Peter Brummund [1997] maintains the Complete Collection of Algorithm Animations. The links to resources are organized both by site and by algorithm. Joseph Bergin [1997] maintains a set of links to web based visualizations. This site was the result of a workshop held at the 1997 Annual Joint Conference on Integrating Technology into Computer Science Education. The resources are classified according to a hierarchy developed by the workgroup (c.f., Naps *et al.* [1997]).

3 PROJECT TOOLS AND TECHNOLOGY

In order to develop the eight interactive learning modules, we purchased two computers and various software tools with funding provided by the Center for Innovation in Learning at Virginia Tech. These tools are described below.

3.1 HARDWARE TOOLS

The development platform for the learning modules was two Pentium III 500 MHz computers running Windows NT 4.0 with 256MB of RAM. These machines were also equipped with 19 inch high-resolution monitors and video cards with 16MB of video memory.

3.2 SOFTWARE TOOLS

3.2.1 Dreamweaver 2.0

Dreamweaver is a professional visual editor for creating and managing web pages. This product was particularly helpful in two regards. First, the visual interface simplified the creation of lesson pages and helped in designing the layout of the pages. Second, Dreamweaver supports proprietary templates that make site-wide changes possible by editing a single template. This feature was indispensable since the learning modules contain over 1,300 web pages and required regular updates during development.

3.2.2 Flash 4.0

Flash is a tool for creating interactive, web-based animations. Interaction is achieved through the use of Flash's scripting language. This language makes it possible for animations to detect user input such as mouse or keyboard events and respond with scripted actions. This tool was the primary technology we used to create interactive components.

3.2.3 Fireworks 3.0 and Paint Shop Pro 6.0

Both of these programs are graphical editors. Fireworks is optimized for the design of web graphics (i.e., GIF and JPEG formats) while Paint Shop Pro is more general and supports a wider range of formats. The primary use of these tools was the

design of graphics embedded in module lessons and the design of images used in various interactive components.

3.2.4 Java

For the creation of applets, we used two different tools. The first tool was the Sun Microsystems' Java 2 Software Development Kit (SDK). This SDK is a free download from Sun and provides all the necessary Java classes for applet development. The second tool was Microsoft's Visual J++ 6.0 Standard Edition. This tool provides an integrated developing environment in addition to Microsoft's own set of Java classes and virtual machine.

4 COMPUTER SCIENCE ONLINE MODULES

This section describes the content of the eight online modules created during the project. Each subsection corresponds to a single online module and includes a summary of the lessons and interactive content contained in the module. For the reader's convenience, WWW links pointing to the appropriate lessons are included in the discussion.

4.1 ALGORITHMS

The Algorithms Module consists of 20 lessons that introduce the topic of algorithms by discussing the characteristics of a good algorithm and comparing several sorting algorithms to illustrate algorithm efficiency and algorithm analysis. The learning objectives for the module are as follows:

- Specify simple algorithms using Structured English,
- Sort numbers using Simple, Insertion, and Selection sorts, and
- Compare simple algorithms for space and time efficiency.

Table 4.1 lists each of the lessons in this module and briefly describes the content of each lesson. The right column of the table indicates any interactive components that are embedded in the lesson.

Table 4.1 Lessons in the Algorithms Module

Lessons	*Interactive Components* †
Introduction to Algorithms – discussion of the relationship between algorithms and Computer Science	• Manual Sort (F)
The Definition of an Algorithm – discussion of five characteristics of an algorithm	
Specifying Algorithms – examples of ways to specify algorithms	
Sorting Algorithms – introduction to the problem of sorting	
Basic Operations – explanation of the comparison and swap operations	• Swap Operation (F)
The Simple Card Sort – introduction to this algorithm using playing cards	• Simple Card Sort (F) • Sort Quiz (J)
The Simple Sort – discussion of how this algorithm is performed on a list of numbers in a computer	
The Insertion Card Sort – introduction to this algorithm using playing cards	• Insertion Card Sort (F) • Sort Quiz (J)
The Insertion Sort – discussion of how this algorithm is performed on a list of numbers in a computer	
The Selection Card Sort – introduction to this algorithm using playing cards	• Selection Card Sort (F) • Sort Quiz (J)
The Selection Sort – discussion of how this algorithm is performed on a list of numbers in a computer	
Algorithm Analysis – introduction to space and time efficiency	
Space Efficiency – discussion of the space efficiency of the three sorting algorithms	• Space Efficiency Quiz (F)
Time Efficiency – discussion of the time efficiency of the three sorting algorithms	• Time Efficiency Quiz (F)
Comparison of Sorts – comparative analysis of the three sorting algorithms	
Analysis of Selection Sort – application of worst-case analysis to this sort	• Selection Sort Algorithm Analysis (F)
Analysis of Other Sorts – application of worst-case analysis to Simple and Insertion sorts	
Worst Case Comparison – comparative analysis of the three sorting algorithms in the worst case	• Sort Analysis Quizzes (F)
Order Notation – introduction to order notation	
Summary – review of the main ideas in Algorithms	

† (F) – Flash animation (J) – Java applet

4.1.1 Manual Sort Animation

This animation illustrates the concept of an algorithm by asking students to sort a sequence of letters and a sequence of playing cards. They are also asked to record the steps they used so that another person could perform the same sort. In this way the students have written their own sorting algorithm before being introduced to the "official" sorting algorithms later in the module. The animation encourages interactivity by allowing students to manipulate the letters or playing cards with the mouse (see Figure 4.1). Once the objects are in sorted order, the animation automatically advances to the next scene.

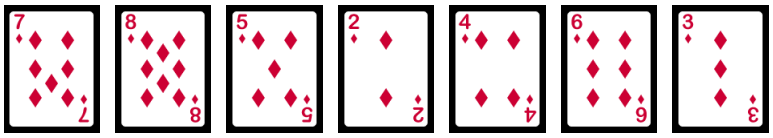
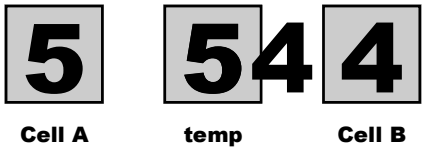


Figure 4.1 Manual Sort Animation

Link: http://courses.cs.vt.edu/csonline/Algorithms/Lessons/Introduction/index.html

4.1.2 Swap Operation Animation

This animation demonstrates a simple algorithm for swapping two numbers in memory and the need to create an additional temporary memory cell to complete the operation. Students are first asked to attempt the swap operation by dragging numbers between two memory cells. The approach, however, always causes one of the original numbers to be lost. To correct this problem, students are then asked to perform the swap operation again with the help of a third memory cell (see Figure 4.2). In addition to teaching the swap operation, this animation shows how new operations (e.g., the swap operation) can be composed of basic operations (e.g., the copy operation).



Then we copy the contents of cell B to cell A. Although the 4 replaces the 5, we still have a copy of the 5 in temp.

Continue

Figure 4.2 Swap Operation Animation

Link: http://courses.cs.vt.edu/csonline/Algorithms/Lessons/BasicOperations/index.html

4.1.3 Simple Sort Algorithm Animation

This animation teaches students how to sort numbers using a basic sorting algorithm called the Simple Sort. Other names for this algorithm include the Min-Max Sort or the Two-Array Selection Sort. The animation demonstrates this sort procedure using the familiar problem of ordering playing cards by their face values. Each step of the algorithm is illustrated in the top half of the animation while the associated step of the algorithm is highlighted in red at the bottom (see Figure 4.3). The animation is accompanied by narration so students can focus on the algorithm rather than trying to read the explanation. Students also control the pace of the animation by a "Continue" button to advance to the next scene or a "Replay" button to repeat the current scene.

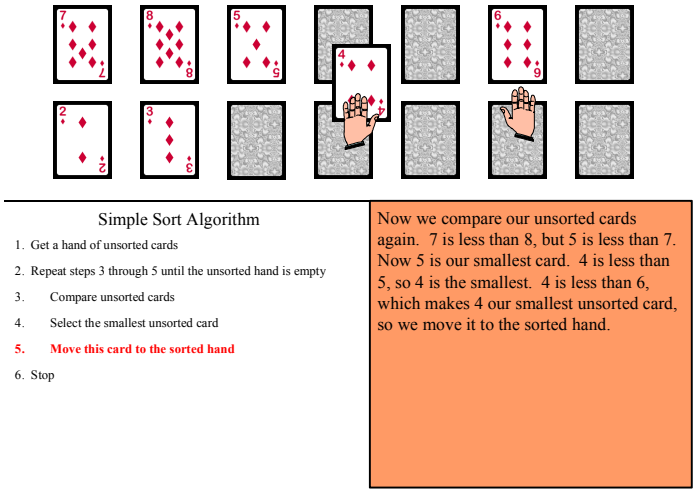


Figure 4.3 Simple Sort Algorithm Animation

Link: http://courses.cs.vt.edu/csonline/Algorithms/Lessons/SimpleCardSort/index.html

4.1.4 Insertion Sort Algorithm Animation

This animation teaches students how to sort items using the Insertion Sort algorithm. The layout and functionality of the animation are similar to the Simple Sort Algorithm described above.

Link: http://courses.cs.vt.edu/csonline/Algorithms/Lessons/InsertionCardSort/index.html

4.1.5 Selection Sort Algorithm Animation

This animation teaches students how to sort items using the Selection Sort algorithm. The layout and functionality of the animation are similar to the Simple Sort Algorithm described above.

Link: http://courses.cs.vt.edu/csonline/Algorithms/Lessons/SelectionCardSort/index.html

4.1.6 Sort Quiz Applet

The Sort Quiz applet tests student comprehension of the three sorting algorithms presented in this module. Students select one of the algorithms via the set of radio buttons at the bottom of the applet (see Figure 4.4). The applet then creates a random permutation of the playing cards and prompts the student to enter the order of the cards after one step of the sort. The applet displays the correct answer after two incorrect attempts to input the next order of the cards. Students can also choose the "Show

Answer" button to watch the applet sort the cards in step-by-step fashion. Since seven unique numbers are used for the quiz, each sort has over 5,000 possible permutations.

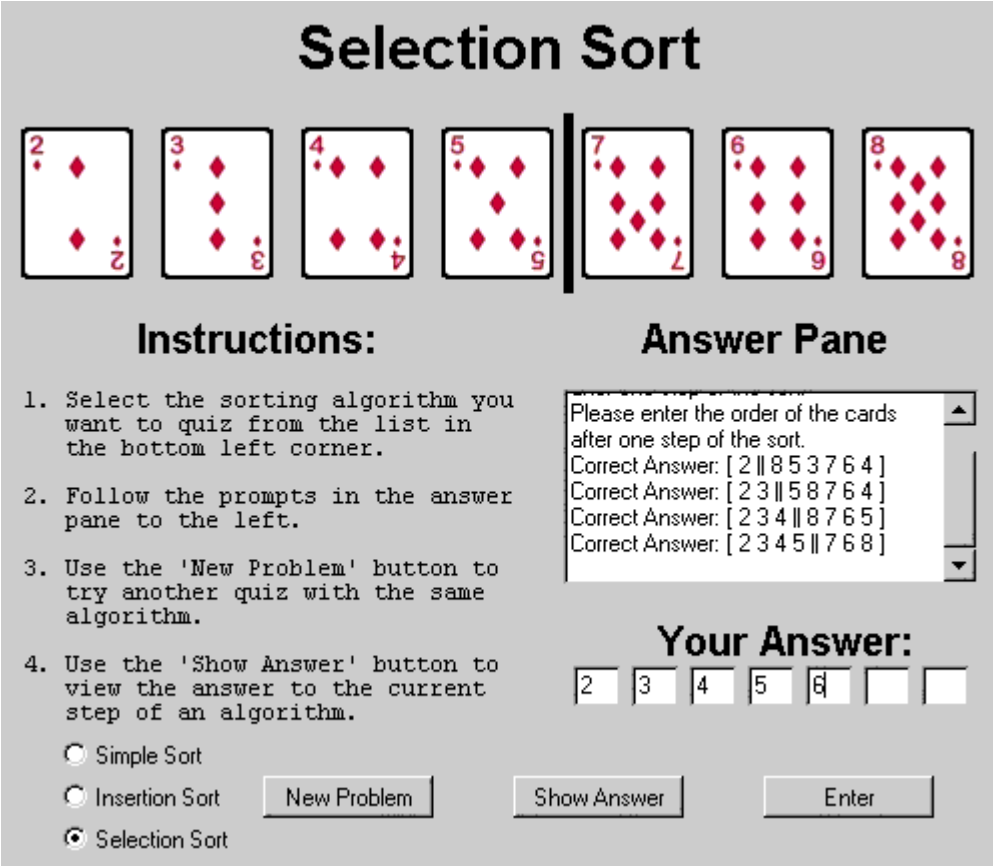


Figure 4.4 Sort Quiz Applet

Link: http://courses.cs.vt.edu/csonline/Algorithms/Questions/SortQuiz/SortQuizSelection.html

4.1.7 Space Efficiency Quiz

This quiz teaches students how to measure space efficiency by comparing the number of memory cells required by the three sorts in this module. The quiz is actually a simple Flash animation that responds with a green check for each correct answer and a red X for each incorrect answer. Students can go directly to the lessons explaining the various sorting algorithms by clicking on the name of the algorithm in the left column (see Figure 4.5).

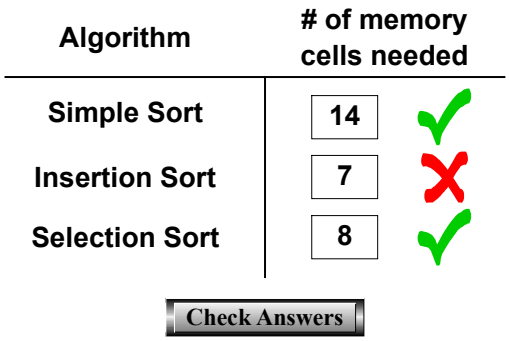


Figure 4.5 Space Efficiency Quiz

Link: http://courses.cs.vt.edu/csonline/Algorithms/Lessons/SpaceEfficiency/index.html

4.1.8 Time Efficiency Quizzes

These quizzes teach students how to measure time efficiency by calculating the number of comparisons and swaps required by the Insertion Sort and the Selection Sort. The design of these quizzes is similar to the Space Efficiency Quiz above.

Link: http://courses.cs.vt.edu/csonline/Algorithms/Lessons/TimeEfficiency/index.html

4.1.9 Selection Sort Algorithm Analysis Animation

This animation introduces the concept of mathematical algorithm analysis by presenting a worst-case analysis of the Selection Sort. First, the animation demonstrates a technique for counting and recording the number of comparisons and swaps performed by the sort at each step of the algorithm (see Figure 4.6). Next, the animation shows how to derive a mathematical formula from this data and explains that these formulas are useful for calculating the algorithm efficiency when sorting lists of arbitrary sizes.

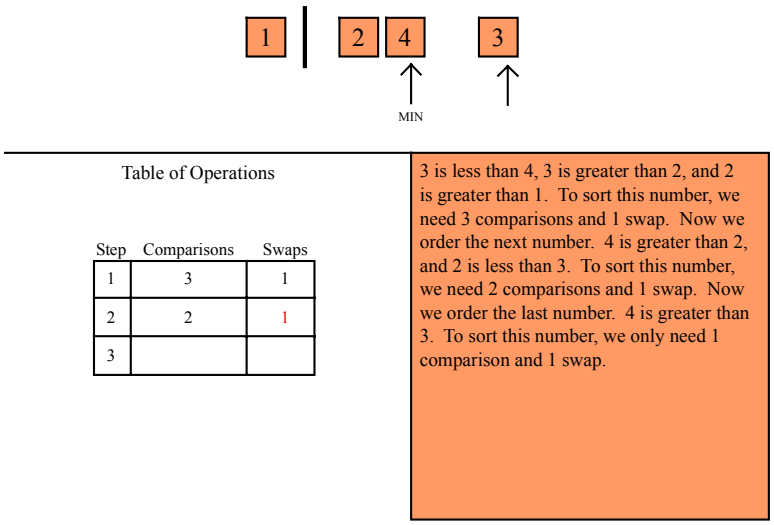


Figure 4.6 Selection Sort Algorithm Analysis Animation

Link: http://courses.cs.vt.edu/csonline/Algorithms/Lessons/WorstCaseSelectionSort/index.html

4.1.10 Simple/Insertion Sort Analysis Quizzes

These two quizzes build on the material presented in the Selection Sort Algorithm Analysis animation above. The quizzes require students to build a data table similar to the one illustrated in the previous animation. Clicking the title of the quiz (see Figure 4.7) will open a new browser window displaying the lesson of the associated sort so that students can review the algorithm. Similar to the Space Efficiency Quiz, correct answers are indicated by a green check and incorrect answers by a red X.

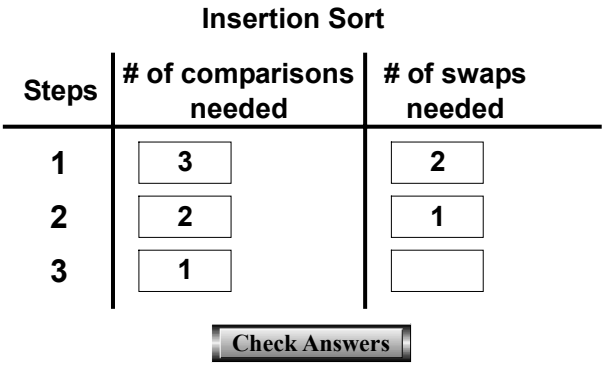


Figure 4.7 Insertion Sort Analysis Quiz

Link: http://courses.cs.vt.edu/csonline/Algorithms/Lessons/WorstCaseOtherSorts/index.html

4.2 ARTIFICIAL INTELLIGENCE

The Artificial Intelligence Module consists of 8 lessons that introduce the topic of artificial intelligence (AI) in computers by surveying several of the major application domains of AI. These application domains include language processing, visual processing, game playing, expert systems, and neural networks. The learning objectives for the module are as follows:

- Understand current applications of AI,
- Recognize the limitations of AI, and
- Compare the human mind to computer intelligence.

Table 4.2 lists each of the lessons in this module and briefly describes the content of each lesson. The right column of the table indicates any interactive components that are embedded in the lesson.

Table 4.2 Lessons in the Artificial Intelligence Module

Lessons	*Interactive Components* †
Introduction to AI – discussion of the goal of AI and the nature of complex tasks	
Humans Versus Computers – comparison of the intelligence of machines and humans	• Humans Versus Computers (F) • Checkers (J) • ELIZA (J)
Natural Language Processing – discussion of several applications of natural language processing	
Game Playing – explanation of search trees and their use in computer game playing	• Eight-Puzzle (J)
Visual Processing – discussion of optical character recognition and handwriting recognition	• JRec (J)
Neural Networks – explanation of the basic structure of a neural network	• JRec (J)
Expert Systems – discussion of the general architecture of expert systems using the example of MYCIN	
Summary – review of the main ideas in Artificial Intelligence	

† (F) – Flash animation (J) – Java applet

4.2.1 Humans Versus Computers Animation

This animation compares the human mind against computer intelligence to illustrate the strengths and weaknesses of each. The storage capacity, connection complexity, and data transfer speed of brains and computers are contrasted, and the animation ends with the observation that "brains still have an edge over computers." While the animation is entertaining, it also teaches students to recognize the types of tasks that are well suited for the human mind and those that are well suited for a computer. Students control the pace of the animation using the "Replay" and "Continue" buttons at the bottom of the animation (see Figure 4.8).

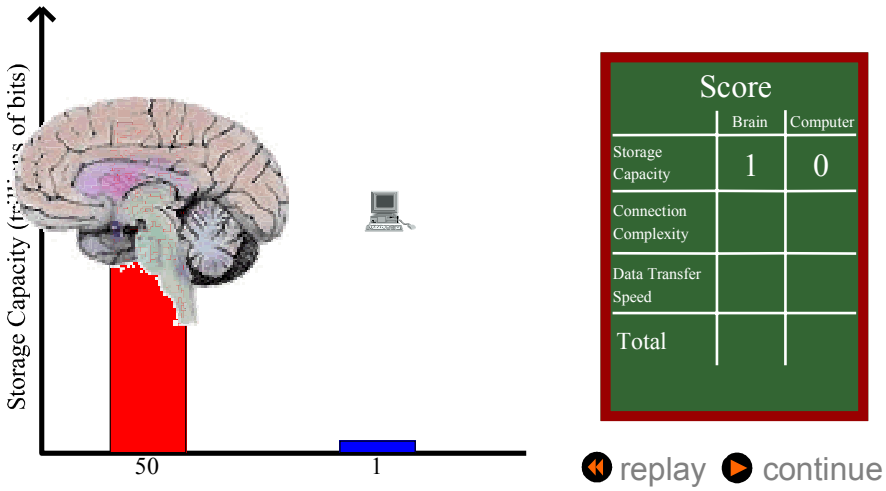


Figure 4.8 Humans Versus Computers Animation

Link: http://courses.cs.vt.edu/csonline/AI/Lessons/HversusC/index.html

4.2.2 Checkers Applet

This applet allows students to play a game of checkers against their own computer (see Figure 4.9). The game illustrates a task that is well suited to computers since it can be modeled by simple numerical procedures. In fact, computers play checkers so well that a computer named "Chinook" is the current man versus machine champion of the world. The applet was reused by permission [Fabio 1997] and the interface was translated from Italian to English. As a result of reusing this applet and several others in this module, the total development time required for the AI module was significantly less than the other modules.

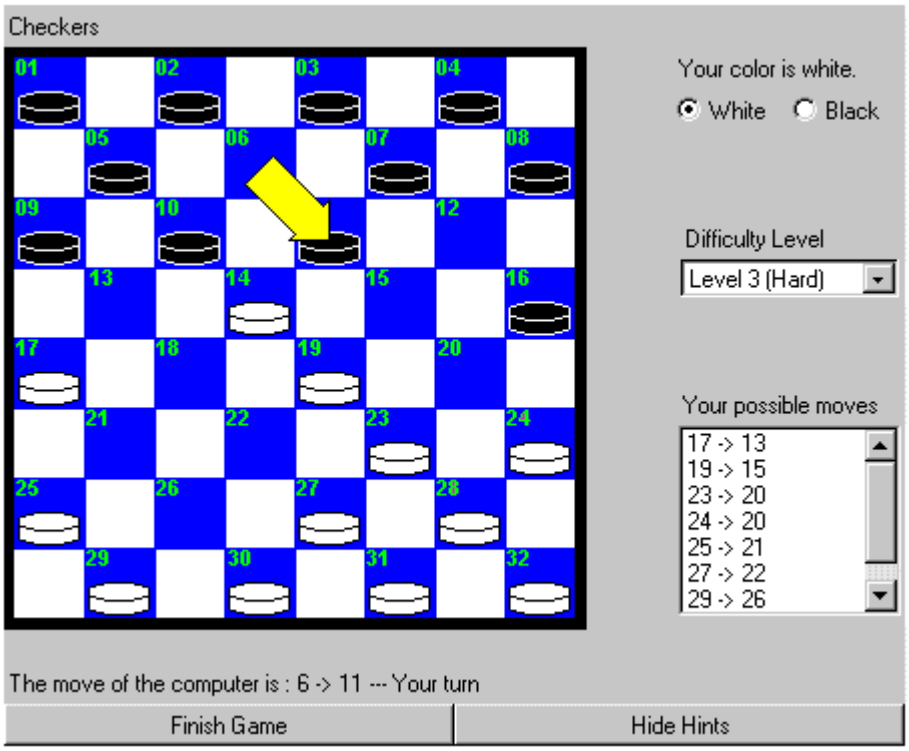


Figure 4.9 Checkers Applet

Link: http://courses.cs.vt.edu/csonline/AI/Lessons/HversusC/index.html

4.2.3 ELIZA Applet

This applet allows students to attempt to carry on a conversation with their computer. Students interact with ELIZA by typing in the text box at the bottom of the applet (see Figure 4.10). The interaction quickly illustrates that conversation is a task that is well suited to humans but not computers. This applet makes a good contrast to the previous Checkers applet which demonstrated one area of AI in which computers are strong. This applet was reused by permission [Goerlich 1996] and the interface was modified slightly.

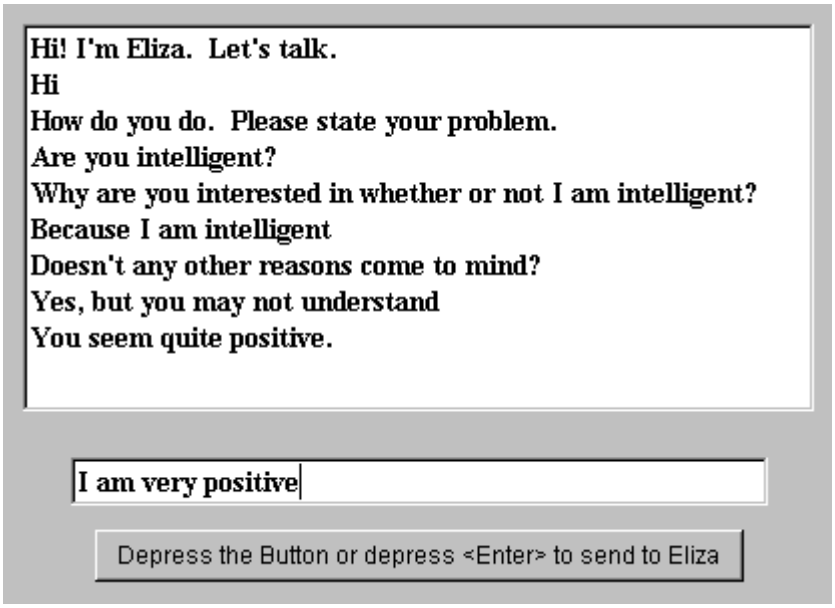


Figure 4.10 Eliza Applet

Link: http://courses.cs.vt.edu/csonline/AI/Lessons/HversusC/index.html

4.2.4 Eight-Puzzle Applet

This applet demonstrates the AI game playing technique of building a search tree to evaluate possible moves. The applet implements a heuristic that students can turn off or on when they ask the computer to solve the puzzle. Since the computer must consider a very large number of possibilities when searching for a solution to the puzzle, the use of the heuristic makes a noticeable difference. The applet also implements a step counting feature and a "Same Shuffle" button (see Figure 4.11) so that students can attempt to solve the puzzle on their own and then instruct the computer to solve the puzzle from the same initial configuration. In many cases, the computer is likely to discover the optimal solution and solve the puzzle in fewer steps than the student.

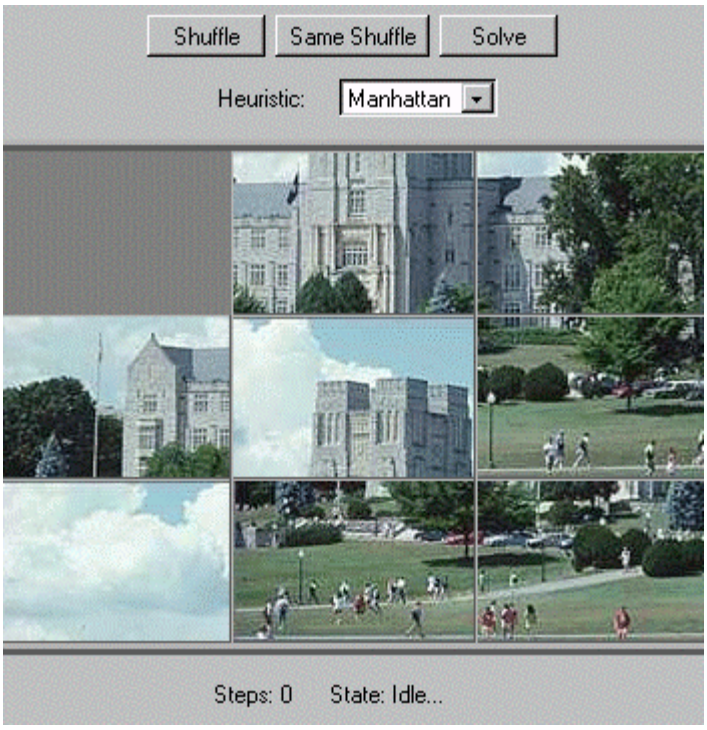


Figure 4.11 Eight-Puzzle Applet

Link: http://courses.cs.vt.edu/csonline/AI/Lessons/GamePlaying/index.html

4.2.5 JRec Applet

This applet illustrates two different areas of AI research: hand writing recognition and neural networks. The applet implements a simple neural network that can be trained to recognize various shapes. Once the applet has been trained with two example cases, it will attempt to identify unknown symbols drawn by the student. The screen shot below (see Figure 4.12) shows the applet correctly identifying the number four. This applet was reused by permission [Mitchell 1998] and the interface was modified slightly to accommodate the layout of the online modules.

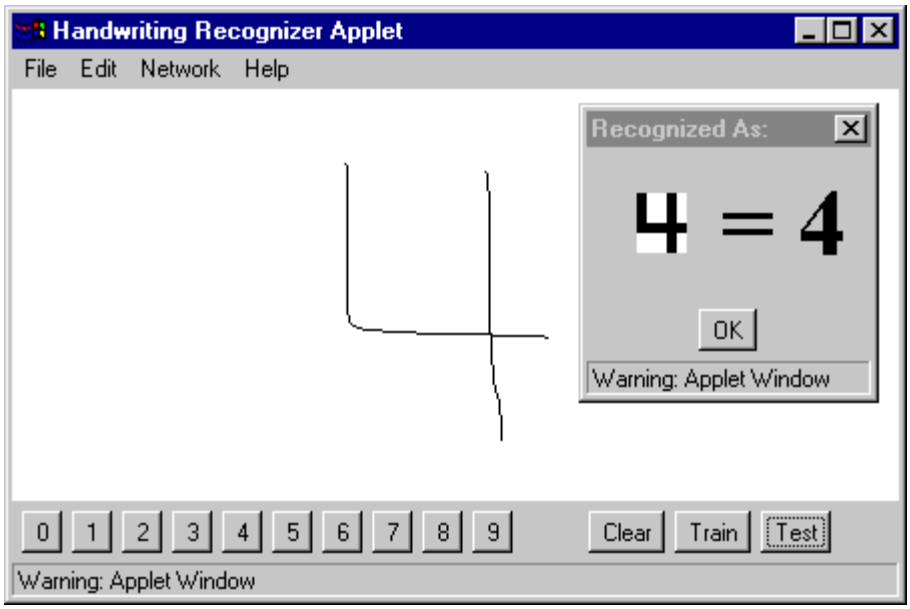


Figure 4.12 JRec Applet

Link: http://courses.cs.vt.edu/csonline/AI/Lessons/VisualProcessing/index.html

4.3 DATA STRUCTURES

The Data Structures Module consists of 16 lessons that introduce the topic of data structures by comparing how data is actually stored in a computer with the abstract structures that programmers use. To illustrate this comparison, several basic data structures such as lists, stacks, and queues are described as well as some non-linear data structures such as multidimensional arrays, graphs, and bags. The learning objectives for the module are as follows:

- Show how data structures map onto physical memory,
- Identify linear versus nonlinear data structures,
- Manipulate data structures with basic operations, and
- Compare different implementations of the same data structure.

Table 4.3 lists each of the lessons in this module and briefly describes the content of each lesson. The right column of the table indicates any interactive components that are embedded in the lesson.

Table 4.3 Lessons in the Data Structures Module

Lessons	*Interactive Components* †
Introduction to Data Structures – comparison of a list and tree data structure	
Computer Memory – discussion of computer memory as linear data cells	
Pointers and Indirection – explanation of pointers as one solution to representing non-linear data structures	
Linear Data Structures – discussion of lists and various ways to view them	
Ordered List: The Abstract View – introduction to basic list operations	
Ordered List: The Implementation View – comparison of array lists and linked lists	• Ordered Arrays (F) • Ordered Linked List (F)
Stacks: The Abstract View – introduction to basic stack operations	• Abstract Stack (F)
Stacks: The Implementation View – comparison of array and pointer implementations of stacks	
Queues: The Abstract View – introduction to basic queue operations	• Abstract Queue (F)
Queues: The Implementation View – discussion of the logical and physical representations of queues	• Queue (J)
Nonlinear Data Structures – discussion of a tree as a type of nonlinear data structure	
Multidimensional arrays – introduction to higher dimensional array structures	• Two-Dimensional Arrays (F)
Trees – introduction to tree terminology and binary trees	
Graphs – introduction to graph terminology and adjacency matrices	• Graphs (F)
Abstract Data Types – comparison of various data structures and their abstract representations	• Bag Abstract Data Type (J)
Summary – review of the main ideas in Data Structures	

† (F) – Flash animation (J) – Java applet

4.3.1 Ordered Arrays Animation

This animation illustrates how the operations of the Ordered List data structure are implemented using a one-dimensional array. The animation instructs students to manually perform the insertions and deletions of list elements using the mouse (see Figure 4.13). While performing the operations, students discover some difficulties with one-dimensional arrays such as limited size and the need to shift many elements to perform various insertions and deletions.

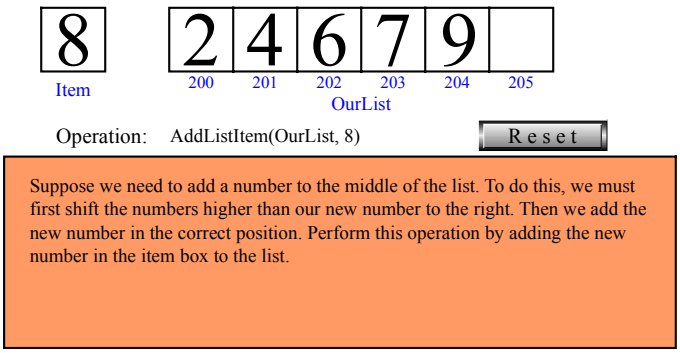


Figure 4.13 Ordered Arrays Animation

Link: http://courses.cs.vt.edu/csonline/DataStructures/Lessons/OrderedListImplementationView/index.html

4.3.2 Ordered Linked List Animation

This animation introduces the concept of the linked list and discusses how it can be used to implement an ordered list data structure. The animation shows how the add and remove operations are implemented in a linked list by changing pointers and contrasts this method with the one-dimensional array implementation presented in the previous animation (see Figure 4.14). This contrast highlights some of the advantages of using a linked list to implement an ordered list.

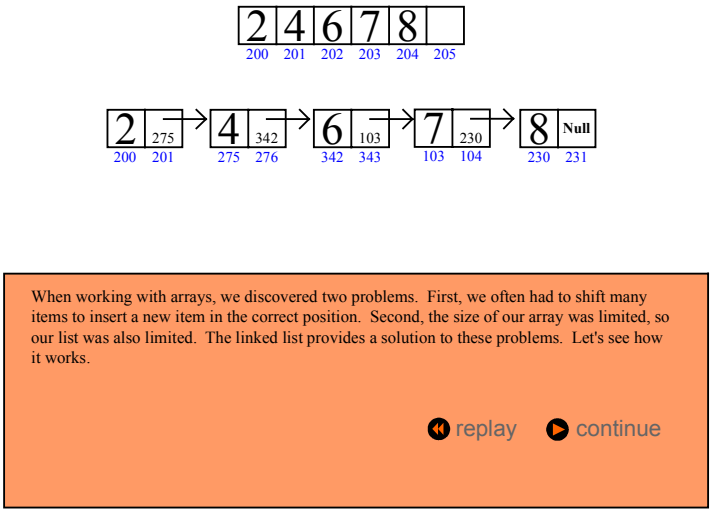


Figure 4.14 Ordered Linked List Animation

Link: http://courses.cs.vt.edu/csonline/DataStructures/Lessons/OrderedListImplementationView/index.html

4.3.3 Abstract Stack Animation

This animation simulates the behavior of an abstract stack data structure. Students can "push" various shapes onto the stack by dragging them with the mouse to the top of the stack, and they can "pop" the top shape off the stack by dragging it outside the stack (see Figure 4.15). The animation instructions also ask students to try performing an illegal operation such as removing an item that is not on the top of the stack. Since this action is not allowed, the animation helps students gain a feel for the behavior of stacks.

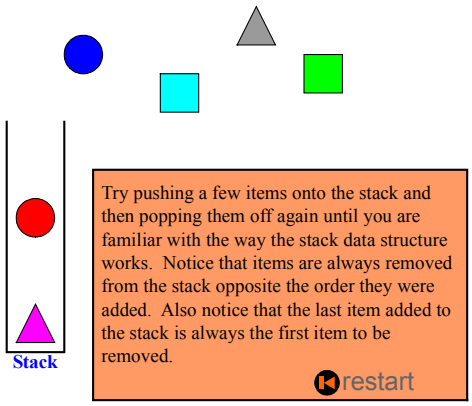


Figure 4.15 Abstract Stack Animation

Link: http://courses.cs.vt.edu/csonline/DataStructures/Lessons/StacksAbstractView/index.html

4.3.4 Abstract Queue Animation

This animation simulates the behavior of an abstract queue data structure. Students can "enqueue" various shapes into the queue by dragging them with the mouse to the tail of the queue (see Figure 4.16). Similarly, students can "dequeue" the shapes by dragging them from the head of the queue. The animation instructions also ask students to try performing an illegal operation such as dequeuing a shape from the middle of the queue or enqueuing a shape at the head of the queue. Since these actions are not allowed, the animation helps students gain a feel for the behavior of queues..

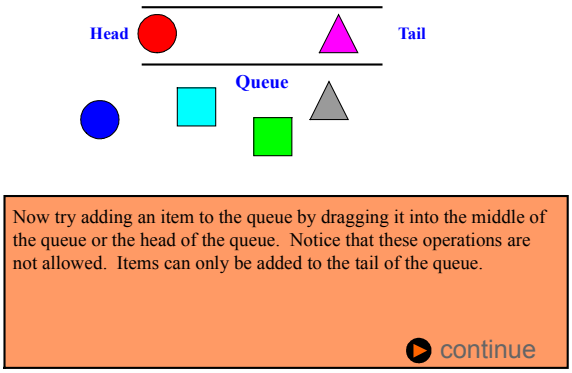


Figure 4.16 Abstract Queue Animation

Link: http://courses.cs.vt.edu/csonline/DataStructures/Lessons/QueuesAbstractView/index.html

4.3.5 Queue Applet

This applet demonstrates three different views of the queue data structure: an abstract view of a general queue, a logical representation of a circular queue, and a physical representation of a circular queue. As students enqueue and dequeue letters, the applet updates each view to its new state by displaying or removing a letter and shifting the head and tail references (see Figure 4.17). The applet also allows students to hide the two implementation views and see only the general queue. In this way, students use the queue without knowing the underlying implementation. This interaction emphasizes the separation between a data structure's implementation and its interface.

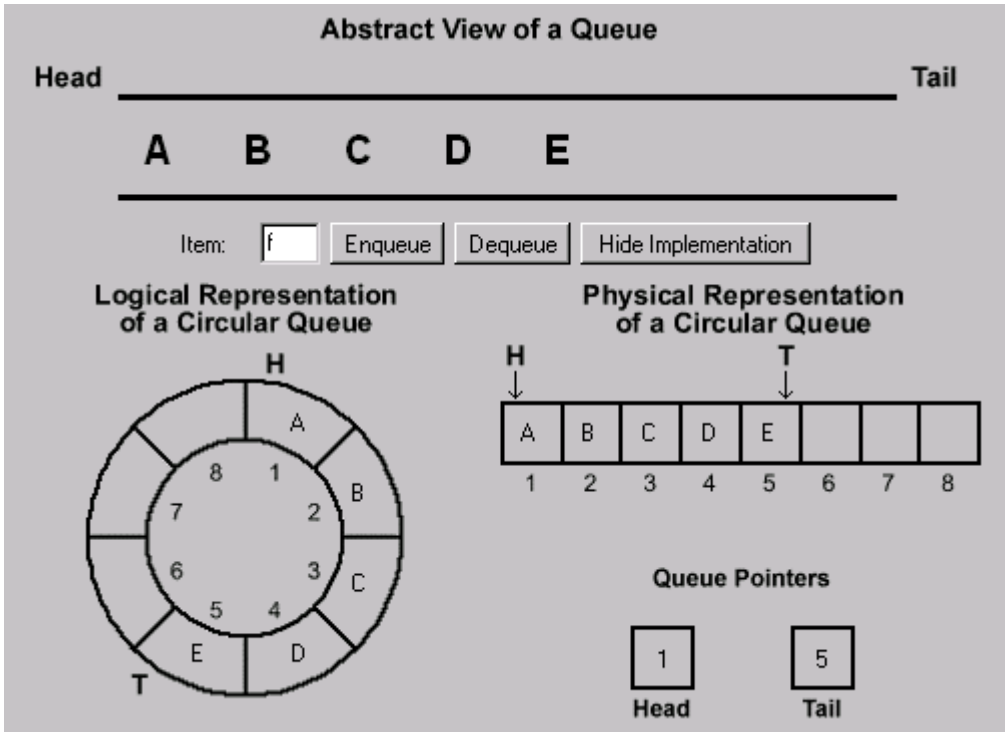


Figure 4.17 Queue Applet

Link: http://courses.cs.vt.edu/csonline/DataStructures/Lessons/QueuesImplementationView/applet.html

4.3.6 Two-Dimensional Arrays Animation

This animation introduces the concept of a two-dimensional array and describes how such arrays are typically represented in memory. The animation presents a segment of nine linear memory cells and shows how these are mapped to a 3x3 matrix using row major order and column major order (see Figure 4.18). Finally, the animation illustrates one application of the 3x3 array by showing how this data structure can be used to represent a tic-tac-toe board. As various X's and O's appear in the matrix, a corresponding X or O appears in the representation of linear memory below the matrix.

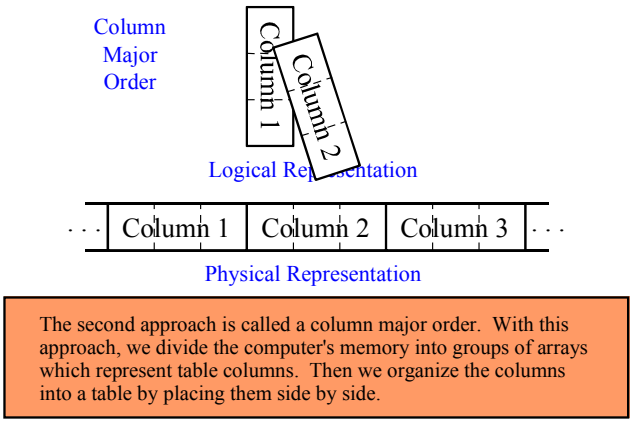


Figure 4.18 Two-Dimensional Arrays Animation

Link: http://courses.cs.vt.edu/csonline/DataStructures/Lessons/2DArrays/index.html

4.3.7 Graph Animation

This animation introduces the concept of general graphs and shows how they can be implemented using an adjacency matrix to represent the graph. The animation simulates graphs varying from two to six vertices in size. Students select the size of the graph from a drop-down list in the instructions box and then connect various vertices by clicking on them with the mouse (see Figure 4.19). Adding or removing an edge from the graph will automatically update the adjacency matrix at the bottom right of the animation.

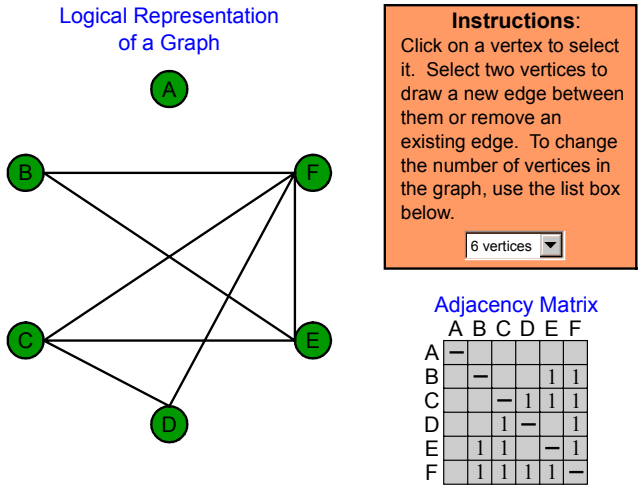


Figure 4.19 Graph Animation

Link: http://courses.cs.vt.edu/csonline/DataStructures/Lessons/Graphs/index.html

4.3.8 Bag Abstract Data Type Applet

This applet simulates the behavior of a bag data structure using an analogy to a grocery bag and fruit. Using the buttons on the right side of the applet (see Figure 4.20), students can perform the following operations on the bag:

- Put an item into the bag,
- Get an item from the bag,
- Ask if the bag is empty,
- Ask if the bag is full,
- Grab a random item from the bag,
- Check if the bag has a certain item,
- Determine the total number of items in the bag, and
- Empty the bag.

The purpose of the applet is to demonstrate the concept of an abstract data type. Therefore, the applet focuses on the behavior of a bag and not its implementation. The review questions for this lesson direct students to manipulate the bag using the various operations and then to describe the effect the operations on the bag.

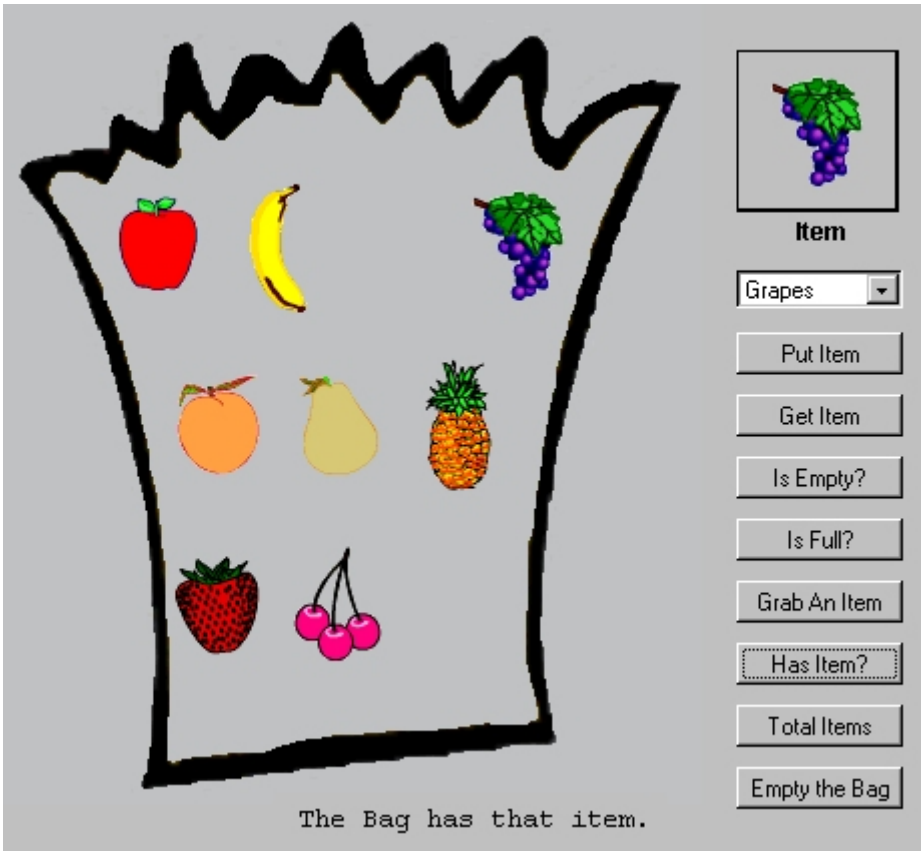


Figure 4.20 Bag Applet

Link: http://courses.cs.vt.edu/csonline/DataStructures/Lessons/AbstractDataTypes/index.html

4.4 MACHINE ARCHITECTURE

The Machine Architecture Module consists of 6 lessons that introduce the topic of machine architecture by explaining data storage in computers, illustrating various gates and circuits, and discussing the central processing unit and machine code. The learning objectives for the module are as follows:

- Explain the various ways data is represented in computer memory.
- Reproduce the truth tables for the AND, OR, and NOT gates,
- Trace the logic of circuits composed of a few simple gates,
- Describe the behavior of the following circuits: decoder, latch, and adder, and
- Write simple programs in machine code.

Table 4.4 lists each of the lessons in this module and briefly describes the content of each lesson. The right column of the table indicates any interactive components that are embedded in the lesson.

Table 4.4 Lessons in the Machine Architecture Module

Lessons	*Interactive Components* †
Introduction to Machine Architecture – discussion of computers as a universal machine	
Data Storage – discussion of bits, floating point numbers, and computer memory	• Data Representation (J)
Gates – introduction to basic gates (AND, OR, NOT) and truth tables	• Simcir (J)
Circuits – introduction to decoders, latches, and adders	• Simcir (J)
The Central Processing Unit – discussion of the CPU and machine language with two simple programs	• Sum Program (F) • Count Program (F)
Summary – review of the main ideas in Machine Architecture	

† (F) – Flash animation (J) – Java applet

4.4.1 Data Representation Applet

This applet shows students the machine representations of three types of data: 2's complement integers, floating point numbers, and ASCII characters. Students select the type of data they wish to store, enter the value for the data, and then view its machine representation (see Figure 4.21). The purpose of this applet is to complement the lesson discussion of data representation.

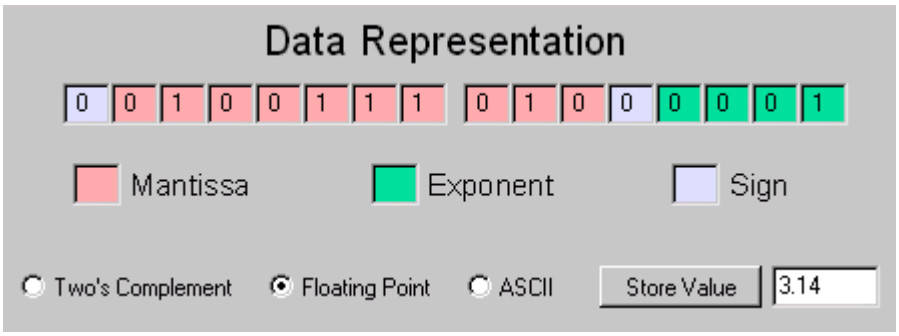


Figure 4.21 Data Representation Applet

Link: http://courses.cs.vt.edu/csonline/MachineArchitecture/Lessons/DataStorage/index.html

4.4.2 Simcir Applet

This applet allows students to build and test simple circuits from a variety of gates. The applet also allows students to experiment with the behavior of various gates as they are introduced in the lesson. The screen shot below (see Figure 4.22) shows a 4-bit decoder in Simcir built from NOT gates and AND gates. Students can interact with the decoder by using the mouse to change the states of the toggle switches. Students can also build their own circuits by dragging gates and other components from the tool pallet on the left and connecting them with mouse clicks. Simcir was reused by permission [Arase 1999] and included in the Machine Architecture module without any modifications.

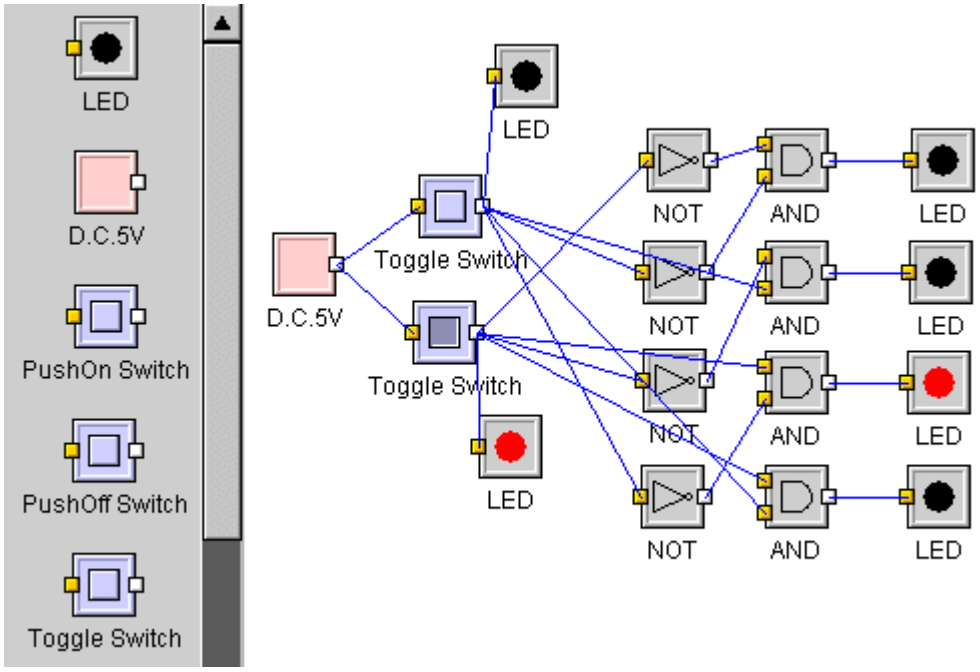


Figure 4.22 Simcir Applet

Link: http://courses.cs.vt.edu/csonline/MachineArchitecture/Lessons/Gates/index.html

4.4.3 Latch Animation

This animation simulates the behavior of a simple latch built from two NAND gates. The latch animation illustrates a circuit that has the ability to remember certain states. By pressing the "Remember 1" button, the latch will eventually show a 1 on the output line (see Figure 4.23). The flow of current through the circuit is simulated by red 1's and 0's moving along the wires. When a pair of numbers enters one of the gates, the

appropriate line of the truth table is highlighted, and a single corresponding number travels down the output line of the gate. Both buttons act as toggle switches that momentarily change the value of their output from 1 to 0.

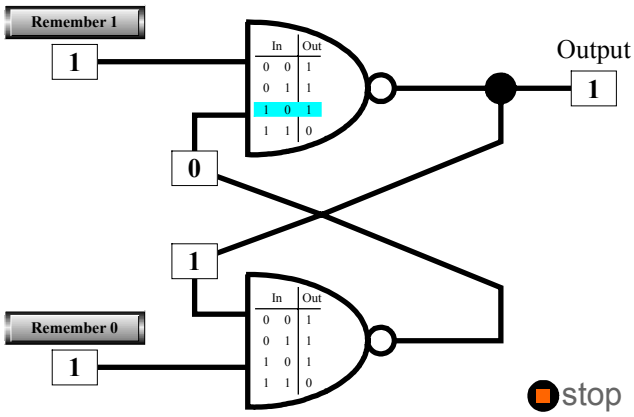


Figure 4.23 Latch Animation

Link: http://courses.cs.vt.edu/csonline/MachineArchitecture/Lessons/Circuits/index.html

4.4.4 Sum Program/Count Program Animations

These animations simulate two simple assembly code programs running on a microprocessor. The microprocessor consists of an instruction register, program counter, decoder, multiplexer, ALU, accumulator, and 15 memory cells (RAM). These components are connected by a data bus, an address bus, and a control bus. Students can view either assembly code or machine code while the animation is running. The Sum Program animation (see Figure 4.24) loads two numbers from memory, adds them, and stores the result back into memory. The Count Program animation demonstrates a simple loop that counts up from zero to five.

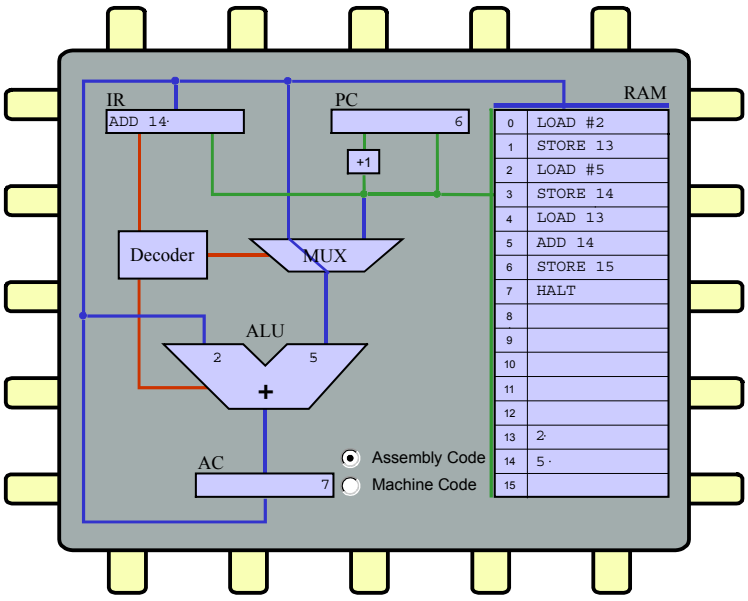


Figure 4.24 Sum Program Animation

Link: http://courses.cs.vt.edu/csonline/MachineArchitecture/Lessons/CPU/index.html

4.5 NUMBER SYSTEMS

The Number Systems Module consists of 21 lessons that introduce the topic of number systems with a focus on binary numbers and binary arithmetic. The module also includes a brief explanation of octal and hexadecimal numbers. The learning objectives for the module are as follows:

- Convert between binary and decimal numbers,
- Add, subtract, multiply, and divide binary numbers,
- Represent signed binary numbers with 1's and 2's complements, and
- Add and subtract signed binary numbers.

Table 4.5 lists each of the lessons in this module and briefly describes the content of each lesson. The right column of the table indicates any interactive components that are embedded in the lesson.

All of the interactive components in this module are Flash animations that illustrate techniques for using binary numbers. These components are divided into three sections: conversion, binary arithmetic, and complements. Each animation is divided into logical scenes that correspond to the steps of the mathematical problem being illustrated. Students can choose to repeat a scene or continue to the next scene using the controls that

appear at the bottom of the animation (see Figure 4.25). Each animation also includes narration that explains the steps of the problem as the animation progresses.

Figure 4.25 Animation controls

4.5.1 Conversion

Three animations in this module deal with the topic of converting numbers between the binary and decimal number systems. Each of these animations illustrates a particular technique for performing the conversion.

4.5.1.1 Binary to Decimal Conversion Animation

The purpose of this animation is to explain how to convert a binary number to its decimal equivalent. The animation solves the problem of converting 11101.01_2 to 29.25_{10} by writing the binary position values above each digit, converting these to decimal, and then summing the position values that correspond to a "1" digit in the binary number. The screen shot below (see Figure 4.26) shows the final step of the conversion in which the position values are summed.

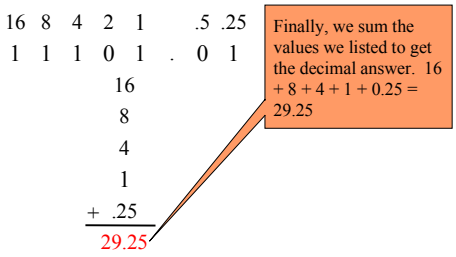


Figure 4.26 Binary to Decimal Conversion Animation

Link: http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/BinaryToDecimalConversion/index.html

Table 4.5 Lessons in the Number Systems Module

Lessons	*Interactive Components* †
Introduction to Binary Numbers – discussion of the relevance of binary numbers to digital computers	
Understanding Number Systems – illustration of the decimal number system	
Binary to Decimal Conversion – illustration of converting binary numbers to decimal	• Binary to Decimal Conversion (F)
Decimal to Binary Conversion – illustration of converting decimal numbers to binary	• Decimal to Binary Conversion (F)
Converting Fractions – illustration of converting binary fractions to decimal	• Conversion with Fractions (F)
Binary Arithmetic – comparison of binary and decimal arithmetic	
Binary Addition – discussion of four basic rules of binary addition	
Adding Two Binary Numbers – illustration of binary addition	• Adding Two Binary Numbers (F)
Adding Multiple Binary Numbers – illustration of adding a column of binary numbers	• Adding Multiple Binary Numbers (F)
Adding Binary Fractions – illustration of adding binary numbers with fractions	
Binary Subtraction – discussion and illustration of the four basic rules of binary subtraction	• Binary Subtraction (F)
Binary Multiplication – discussion and illustration of binary multiplication	• Binary Multiplication (F)
Binary Division – discussion and illustration of binary division	• Binary Division (F)
Signed numbers – discussion of signed magnitude representation for binary numbers	
One's complement – explanation of one's complement	
Two's complement – explanation of two's complement	
Subtraction with Signed Numbers – introduction to subtracting signed numbers	
Subtraction with One's complement – illustration of subtraction with one's complement	Subtraction with One's Complement (F)
Subtraction with Two's complement – illustration of subtraction with two's complement	Subtraction with Two's Complement (F)
Hexadecimal and Octal Numbers – application of number system principles to hexadecimal and octal numbers	
Comparing Number Systems – converting between binary, octal, and hexadecimal numbers	
Summary – review of the main ideas in the Number Systems	

† (F) – Flash animation (J) – Java applet

4.5.1.2 Decimal to Binary Conversion Animation

The purpose of this animation is to explain how to convert a decimal number to its binary equivalent. The animation solves the problem of converting 11_{10} to 1011_2 by the method of repeated division. After each division by two, the remainder is moved to the bottom of the animation to form the final answer. The screen shot below (see Figure 4.27) shows the first division in the conversion.

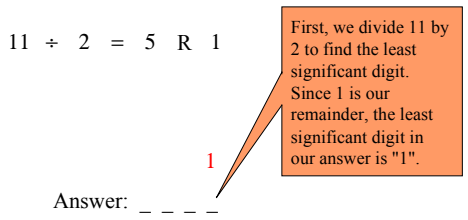


Figure 4.27 Decimal to Binary Conversion Animation

Link: http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/DecimalToBinaryConversion/index.html

4.5.1.3 Decimal to Binary Conversion with Fractions

The purpose of this animation is to explain how to convert a decimal fraction to its binary equivalent. The animation solves the problem of converting 0.035_{10} to 0.011_2 by the method of repeated multiplication. After each multiplication by two, the most significant digit is moved to the bottom of the animation to form the final answer. The screen shot below (see Figure 4.28) shows the second multiplication in the conversion.

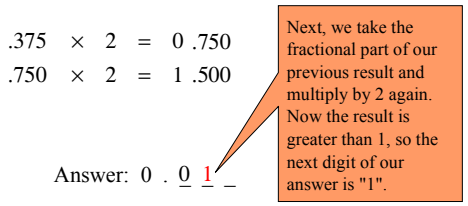


Figure 4.28 Decimal to Binary Conversion with Fractions Animation

Link: http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/DecimalToBinaryConversionWithFractions/index.html

4.5.2 Binary Arithmetic

Five animations in this module deal with the topic of binary arithmetic. These cover the four main arithmetic operations: addition, subtraction, multiplication, and division. Since the concept of carries is often confusing to students learning binary addition, one animation is devoted to illustrating how this is done when adding multiple binary numbers.

4.5.2.1 Adding Two Binary Numbers Animation

This animation shows how to add the numbers 110_2 and 1111_2 using the four basic rules of binary addition. These rules are discussed in the lesson text before the animation. Notice in the scene shot below (see Figure 4.29) that cueing is used to direct the student's attention by graying out parts of the problem that are not immediately relevant.

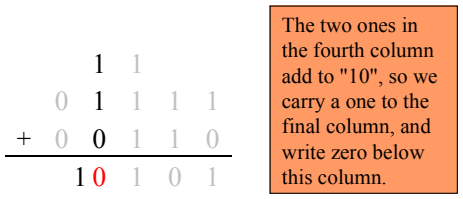


Figure 4.29 Adding Two Binary Numbers Animation

Link: http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/AddingTwoBinaryNumbers/index.html

4.5.2.2 Adding Multiple Binary Numbers Animation

This animation teaches a technique for adding multiple binary numbers by showing students how to add 111_2, 110_2, 1101_2, 101_2, and 1110_2 together. When adding many binary numbers, students need to keep track of the number of carries to mark in the next column. The animation shows them how this can be done by crossing out pairs of ones in the current column and marking a carry for each pair in the adjacent column. The screen shot below (see Figure 4.30) shows this technique.

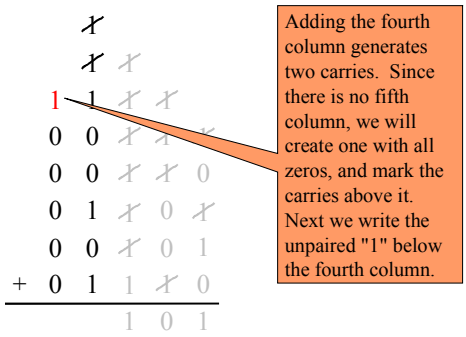


Figure 4.30 Adding Multiple Binary Numbers Animation

Link: http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/AddingMultipleBinaryNumbers/index.html

4.5.2.3 Binary Subtraction Animation

This animation shows how to subtract the number 1011_2 from 1111_2 using the four basic rules of binary subtraction. These rules define what is generally thought of as the Borrow Method for subtraction. The rules are discussed in the lesson text before the animation. The screen shot below (see Figure 4.31) shows a borrow from the left-most column in order to complete the subtraction in the middle column.

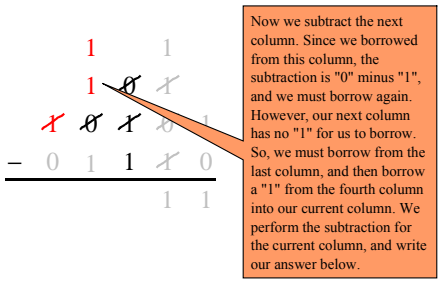


Figure 4.31 Binary Subtraction Animation

Link: http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/Subtraction/index.html

4.5.2.4 Binary Multiplication Animation

This animation shows how to multiply the numbers 1111_2 and 1011_2. Since multiplication is really just repeated addition, this animation builds on the technique introduced in the Adding Multiple Binary Numbers animation. The screen shot below (see Figure 4.32) shows how a zero multiplier digit results in a row of zeros.

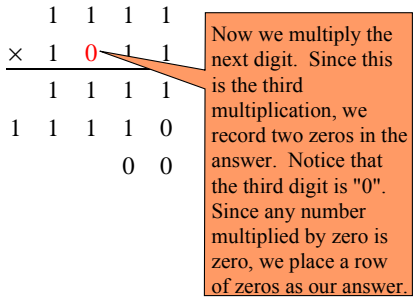


Figure 4.32 Binary Multiplication Animation

Link: http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/Multiplication/index.html

4.5.2.5 Binary Division Animation

This animation shows how to divide the number 1000_2 by 110_2 using the method of long division. Since division is really just repeated subtraction, this animation builds on the Borrow Method technique introduced in the Binary Subtraction animation. The screen shot below (see Figure 4.33) shows the completed long division problem.

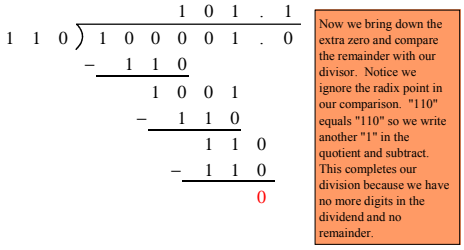


Figure 4.33 Binary Division Animation

Link: http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/Division/index.html

4.5.3 Complements

Two animations in this module deal with the topic of complements. These animations illustrate how complements are used to perform binary subtraction in microprocessors.

4.5.3.1 Binary Subtraction with 1's Complement Animation

This animation illustrates how to subtract binary numbers by converting the numbers to 1's complement representation and adding them. The animation uses the example problem of subtracting the number 1001_2 from 1101_2. As the animation

progresses, the decimal values are displayed to the right side of the problem (see Figure 4.34). This allows students to see how subtraction with complements is similar to adding negative numbers in decimal.

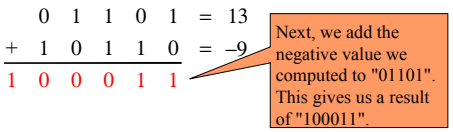


Figure 4.34 Binary Subtraction with 1's Complement Animation

Link: http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/SubtractionWithOnesComplement/index.html

4.5.3.2 Binary Subtraction with 2's Complement Animation

This animation illustrates how to subtract binary numbers by converting the numbers to 2's complement representation and adding them. The animation uses the same example problem as the animation above so that students can contrast the 1's complement and 2's complement techniques. The screen shot below (see Figure 4.35) shows how the 2's complement number is formed.

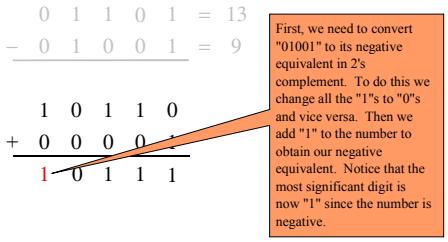


Figure 4.35 Binary Subtraction with 2's Complement Animation

Link: http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/SubtractionWithTwosComplement/index.html

4.6 OPERATING SYSTEMS

The Operating Systems Module consists of 9 lessons that introduce this topic by discussing the main function of operating systems and investigating several key parts of operating systems such as memory and file managers. The learning objectives for the module are as follows:

- Understand the purpose of the operating system,
- Distinguish between a resource, a program, and a process,
- Recognize critical resources and explain the behavior of semaphores,
- Describe various memory page replacement algorithms, and
- Describe how files are stored in secondary storage

Table 4.6 lists each of the lessons in this module and briefly describes the content of each lesson. The right column of the table indicates any interactive components that are embedded in the lesson.

Table 4.6 Lessons in the Operating Systems Module

Lessons	*Interactive Components* †
Introduction to Operating Systems – discussion of various types of operating systems	
Resources – discussion of the problem of resource allocation in computers	• Nursery Game (J)
Processes – introduction to processes and process scheduling	• Process State Diagram (F) • Process Scheduling Simulation (J)
Synchronization – definition and illustration of critical resources and sharing	• Mutex Demonstration (J) • Bounded Buffer Demonstration (J)
Deadlock – discussion of the conditions that must exist for deadlock to occur	• Dining Philosophers (J)
Memory Allocation – discussion of best fit, worst fit, and first fit memory allocation strategies	• Memory Allocation (J)
Virtual Memory – discussion of five virtual memory paging policies	• Virtual Memory Simulation (J) • Simulation of Page Replacement (J)
File Management – discussion of contiguous, linked, and indexed allocation methods	• File System Allocation (J)
Summary – review of the main ideas in Operating Systems	

† (F) – Flash animation (J) – Java applet

4.6.1 Nursery Game Applet

This animation demonstrates the allocation of limited computer resources to processes in an operating system. To do this, it draws an analogy between an operating system full of processes and a nursery full of babies (see Figure 4.36). In the applet, babies represent processes while nurses, toys, and blankets represent resources. The students themselves represent the operating system which must allocate resources to the babies with the goal of maximizing the number of babies who are happy. Babies who are unhappy signal their distress by crying and requesting a certain resource (i.e., a blanket,

toy, or nurse) which is indicated by a blinking red light. Students must then allocate the desired resource to the baby to appease him. Since resources are limited, students cannot always keep the nursery quiet. The intention of this applet is to give the students some idea of how difficult resource allocation can be when resources are limited.

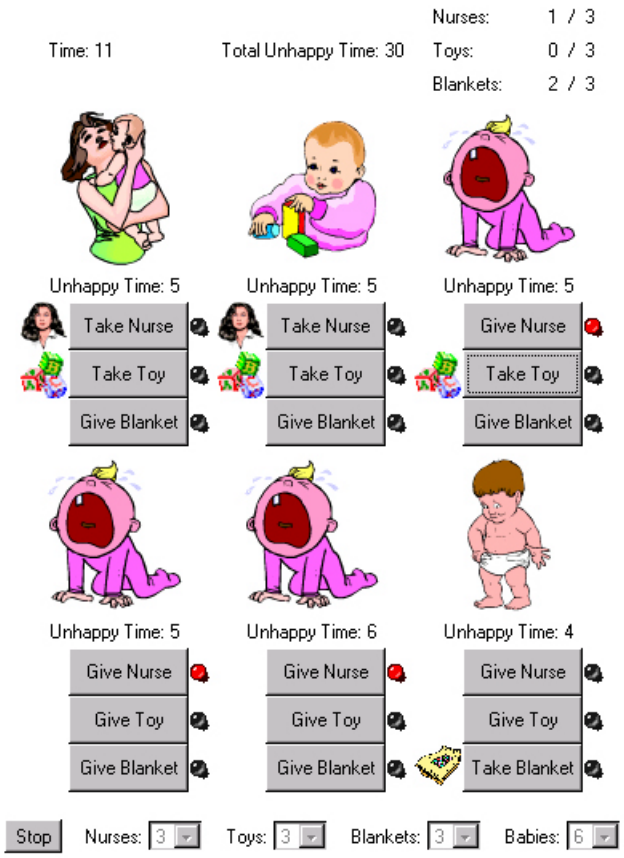


Figure 4.36 Nursery Game Applet

Link: http://courses.cs.vt.edu/csonline/OS/Lessons/Resources/index.html

4.6.2 Process State Diagram

This animation demonstrates the classic Ready-Running-Waiting process state diagram. It shows how processes transition from one state to another throughout their life cycle and includes the role of the operating system in overseeing these transitions. The animation controls at the bottom right corner allow students to pause or step through the simulation (see Figure 4.37).

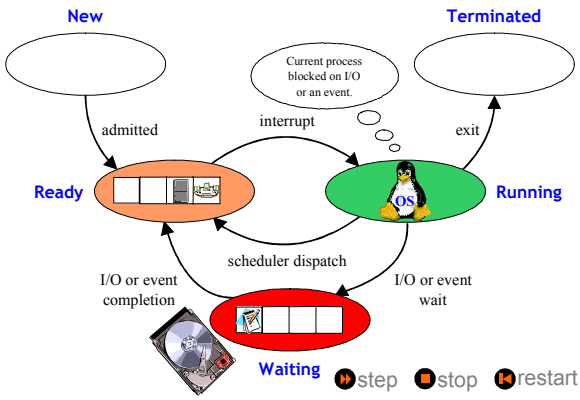


Figure 4.37 Process State Diagram Animation

Link: http://courses.cs.vt.edu/csonline/OS/Lessons/Processes/index.html

4.6.3 Process Scheduling Simulation Applet

This applet allows students to enter the name, arrival time, and service time of various processes and choose a scheduling algorithm. The applet then runs the processes using that algorithm, plots the running time for each process (see Figure 4.38), and displays a table of statistics with the arrival time, service time, finish time, and turnaround time for each process. The scheduling algorithms are First Come First Serve, Round Robin, and Shortest Process Next. This applet was reused with permission [Tran 1998] and included in this module with some modifications to the interface.

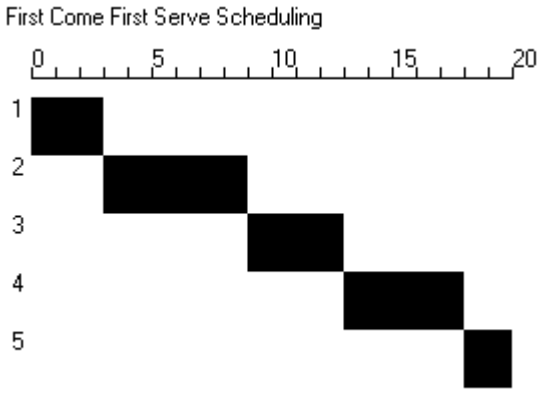


Figure 4.38 Process Scheduling Simulation Applet

Link: http://courses.cs.vt.edu/csonline/OS/Lessons/Processes/index.html

4.6.4 Mutex Demonstration Applet

This applet demonstrates how a semaphore is used in an operating system to prevent multiple processes from accessing a critical section simultaneously. Processes are represented by blue disks that slowly paint themselves on the screen in a counter-clockwise fashion. Each process has a light blue critical section that it does not enter unless Mutex is "1" (see Figure 4.39). When any process is in its critical section, Mutex is set to "0" and all other processes block upon reaching their critical sections. When a process exits its critical section, Mutex is set to "1" again and the next process can unblock and enter its critical section. Note that Mutex must initially be set to "1" or none of the processes can enter their critical section. While the simulation is running, students can pause or run each process and adjust the size of a process's critical section. This applet was reused with permission [Magee and Kramer 1999] and included without modification in this module.

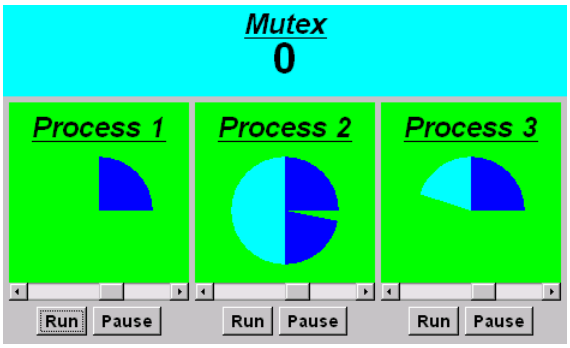


Figure 4.39 Mutex Demonstration Applet

Link: http://courses.cs.vt.edu/csonline/OS/Lessons/Synchronization/index.html

4.6.5 Bounded Buffer Demonstration Applet

This applet demonstrates synchronization in the classic Bounded Buffer problem. Processes are again represented by animated disks (see Figure 4.40). The producer process adds one letter to the center buffer with each revolution, and the consumer process removes one letter with each revolution. The producer blocks if the buffer becomes full, and the consumer blocks if the buffer becomes empty. Students can run and pause both processes to observe the behavior of the system. This applet was reused

with permission [Magee and Kramer 1999] and included without modification in this module.

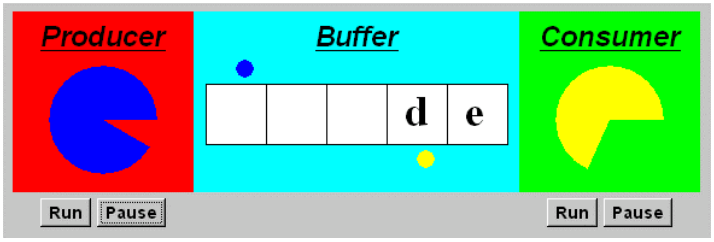


Figure 4.40 Bounded Buffer Demonstration Applet

Link: http://courses.cs.vt.edu/csonline/OS/Lessons/Synchronization/index.html

4.6.6 Dining Philosophers Applet

This applet demonstrates the problem of deadlock with the classic Dining Philosopher problem. The five smiley faces represent the philosophers and the five black dots represent forks (see Figure 4.41). Each philosopher must have two forks to begin eating, and they only take forks to their immediate left or right. When each philosopher has exactly one fork, the system deadlocks since they are all waiting for another fork. The applet operates in two modes. The first mode uses a naïve allocation method that is susceptible to deadlock. The second mode uses a deadlock avoidance method by dictating how some of the philosophers may take their forks. This applet was reused with permission [Magee and Kramer 1999] and included without modification in this module.

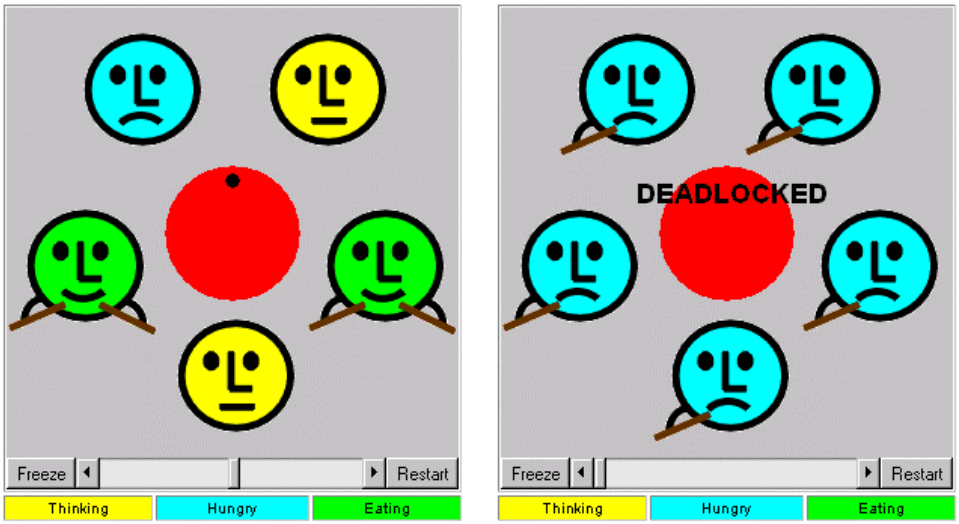


Figure 4.41 Dining Philosophers Applet

Link: http://courses.cs.vt.edu/csonline/OS/Lessons/Deadlock/index.html

4.6.7 Memory Allocation Applet

This applet demonstrates three memory allocation strategies: best fit, worst fit, and first fit. Students begin the simulation by setting the size of the simulation memory and the number of processes in the operating system. The applet then has each process make memory requests and uses the three allocation strategies to manage the memory in the system. Each strategy is represented as a green column that corresponds to main memory. Active processes are shown as light blue bands (see Figure 4.42). This representation is particularly helpful for highlighting the problems associated with memory fragmentation. The applet was reused by permission [Gokey 1997] and included in this module with some minor interface changes.

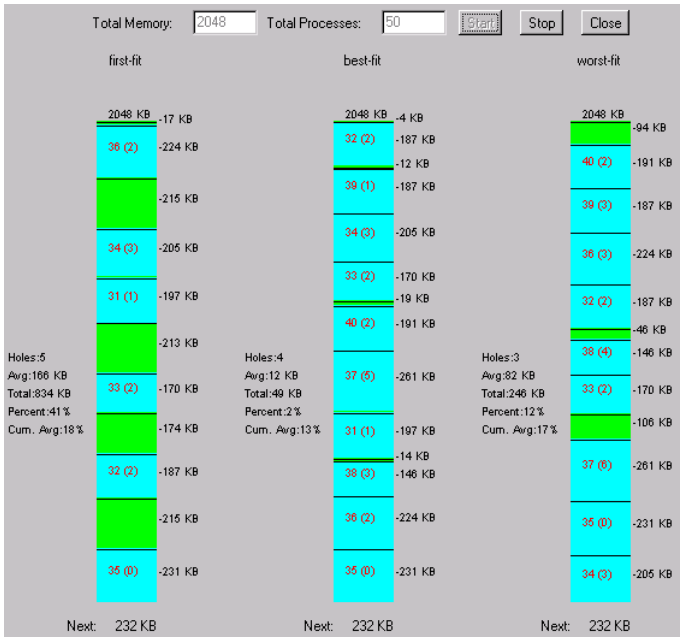


Figure 4.42 Memory Allocation Applet

Link: http://courses.cs.vt.edu/csonline/OS/Lessons/MemoryAllocation/index.html

4.6.8 Virtual Memory Simulation Applet

This applet displays an animated simulation of processes requesting pages in memory. The eight rectangles in the middle of the applet represent the physical memory pages, and the sixteen rectangles on the left represent the virtual memory pages (see Figure 4.43). Since the physical memory is limited, pages must be swapped to virtual memory when a process requests a page that is not currently loaded. Physical memory pages are connected by lines to their locations in virtual memory. When a process's memory request causes a page fault, the process is displayed in red, and the appropriate virtual memory page is loaded. The applet was reused with permission [Coutinho 1996] and included with some minor modifications to the interface.

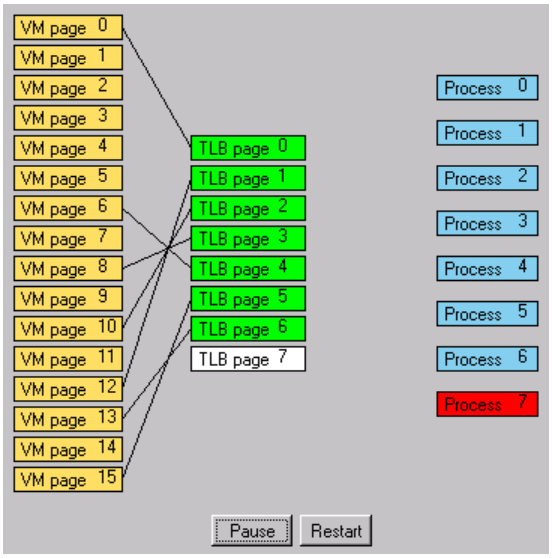


Figure 4.43 Virtual Memory Simulation Applet

Link: http://courses.cs.vt.edu/csonline/OS/Lessons/VirtualMemory/index.html

4.6.9 Simulation of Page Replacement Algorithms Applet

This applet simulates the performance of five virtual memory page replacement algorithms: First Come First Serve, Least Recently Used, Least Frequently Used, Second Chance, and Random. Students can specify the order of page references via an input string or allow the applet to randomly choose pages. During the simulation, the applet calculates the number of page faults and the page fault rate for each algorithm (see Figure 4.44). The applet was reused by permission [Song 1997] and included in this module with minor modifications.

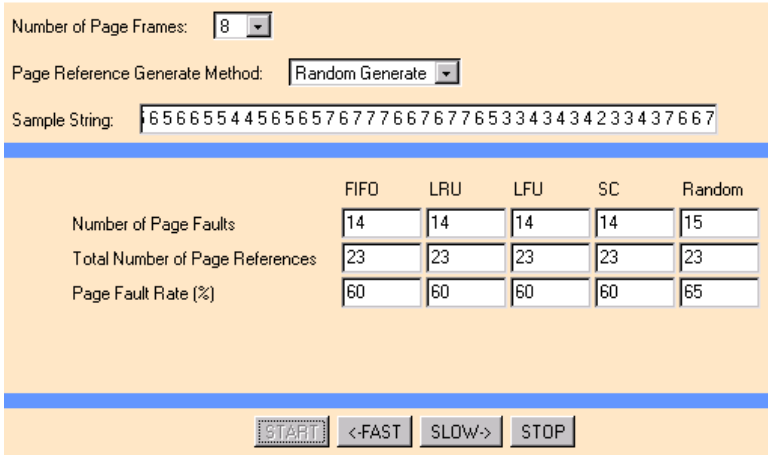


Figure 4.44 Simulation of Page Replacement Algorithms Applet

Link: http://courses.cs.vt.edu/csonline/OS/Lessons/VirtualMemory/index.html

4.6.10 File System Allocation Applets

These three applets illustrate three methods for allocating blocks in a file system: the contiguous method, the linked method, and the indexed method. The applets display the free list, file allocation table, and hard disk sectors graphically so that students can visualize the file allocation process (see Figure 4.45). Through the applets' interfaces, students can allocate and deallocate files, view a directory list of the current files, or reset the applet. These applets were reused by permission [Moosani 1998] and included in this module with various modifications and corrections.

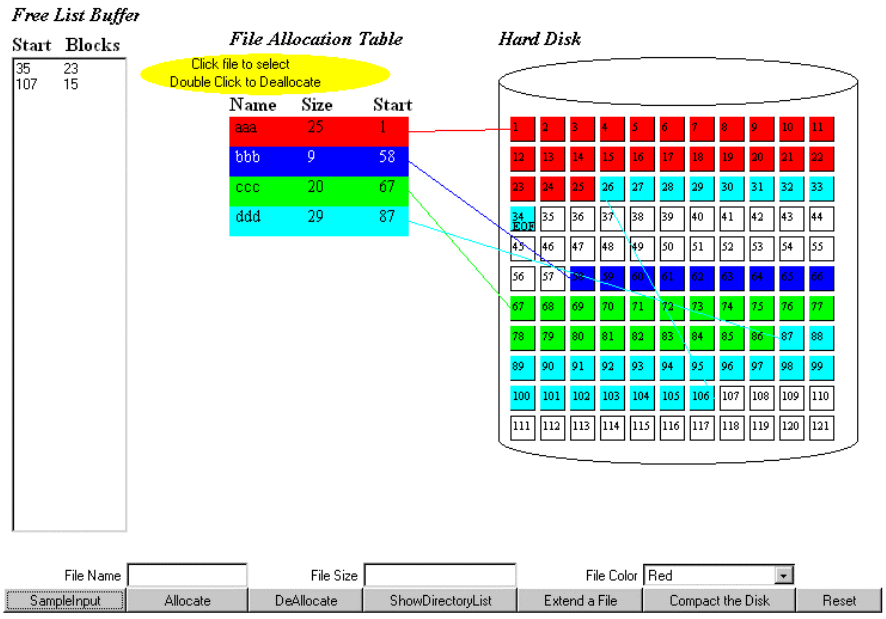


Figure 4.45 File System Allocation Applet

Link: http://courses.cs.vt.edu/csonline/OS/Lessons/FileManagement/index.html

4.7 PROGRAMMING LANGUAGES

The Programming Languages Module consists of 14 lessons that introduce the topic of programming languages by discussing five important concepts in procedural languages: identifiers, expressions, control structures, input/output, and abstraction. The final lesson illustrates these concepts with an example program implementing the Selection Sort algorithm. The learning objectives for the module are as follows:

- Identify appropriate control structures,
- Use subroutines to organize programs,
- Trace the execution of simple programs, and
- Write simple programs in pseudocode.

Table 4.7 lists each of the lessons in this module and briefly describes the content of each lesson. The right column of the table indicates any interactive components that are embedded in the lesson.

Table 4.7 Lessons in the Programming Languages Module

Lessons	*Interactive Components* †
Introduction to Programming Languages – discussion of the evolution of procedural programming languages	• Code Representations (F)
Identifiers – definition of identifiers, variables, and constants	
Assignment – introduction to the assignment operation and its syntax	• Variables and Assignment (F) • Simple Assignment Machine (J)
Expressions – definition and illustration of expressions	
Boolean Expressions – illustration of Boolean operators and truth tables	
Data Types – comparison of five primitive data types	• Data Types Animation (F)
Control Structures – introduction to program flow of control and four control structures	
Selection – introduction to case logic and simple programs with conditionals	• Selection Exercises (F)
Loops – introduction to loop logic and a simple loop program	• Loops Demonstration (F)
Subprograms – definition and illustration of procedures and functions	• Call Power (F) • Trace Power (F)
Parameters – comparison of pass-by-reference and pass-by-value parameters	• Trace Power (F) • Parameter Passing (F)
Input/Output – discussion of user input/output and file input/output	
Programs – illustration of a procedural program using Selection Sort	• Selection Sort (J)
Summary – review of the main ideas in Programming Languages	

† (F) – Flash animation (J) – Java applet

4.7.1 Code Representations Animation

This animation displays the plain English, pseudocode, assembly code, and machine code representations of the Simple Tax Program, a basic program to calculate sales tax. By clicking on the left column of buttons (see Figure 4.46), students can flip through the different representations of the program and compare them.

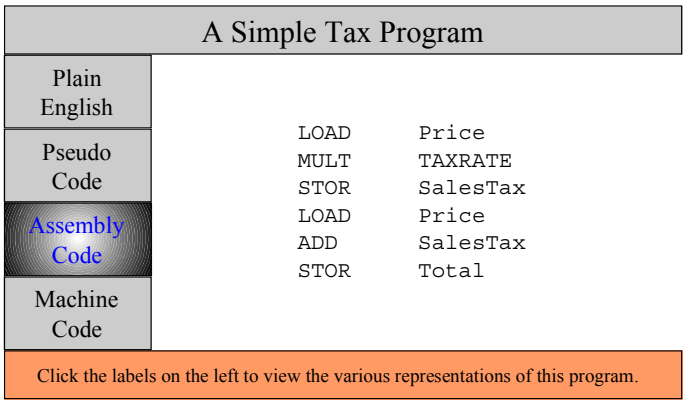


Figure 4.46 Code Representations Animation

Link: http://courses.cs.vt.edu/csonline/ProgrammingLanguages/Lessons/Introduction/index.html

4.7.2 Variables and Assignment Animation

This animation illustrates how variable names are associated with memory locations in a computer. It also uses the Simple Tax Program presented in the previous animation to demonstrate how statements in a procedural programming language cause values to be assigned to memory cells and cause those values to change. The screen shot below (see Figure 4.47) shows a cross section of computer memory along with the Simple Tax Program. As each statement of the program is executed, the values assigned to various identifiers appear in the designated memory locations.

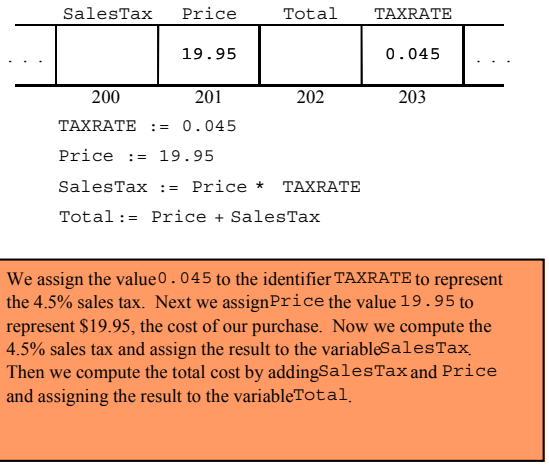


Figure 4.47 Variables and Assignment Animation

Link: http://courses.cs.vt.edu/csonline/ProgrammingLanguages/Lessons/Assignment/index.html

4.7.3 Simple Assignment Machine Applet

This applet allows students to construct simple programs consisting of assignment statements and basic mathematical operations. Students can create both variables and constants and watch the values of the variables change in memory as each statement executes. By working through the five accompanying exercises, students learn the appropriate use of variables and constants, the value of using named constants, how assignment statements are constructed, and how these statements work to modify variables. The first exercise (see Figure 4.48) requires students to code the Simple Tax Program that they have seen in the previous lessons.

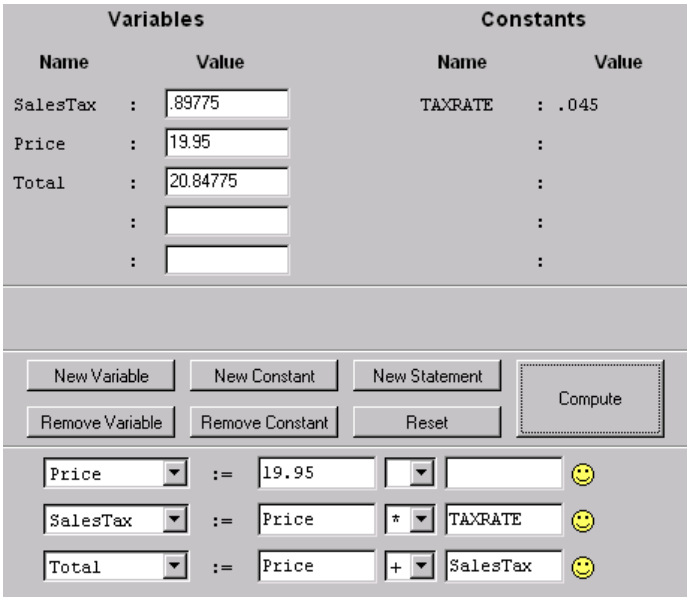


Figure 4.48 Simple Assignment Machine Applet

Link: http://courses.cs.vt.edu/csonline/ProgrammingLanguages/Questions/Assignment/applet.html

4.7.4 Data Types Animation

This animation describes five primitive data types that are found in procedural programming languages: integers, reals, characters, Booleans, and arrays. Each type corresponds to one scene in the animation. Students view the scenes by clicking on one of the data type buttons at the bottom of the animation (see Figure 4.49). In addition to providing a brief explanation of each type, each scene also gives example values of the

type, a pseudocode declaration for the type, and some example operations that can be performed on the type.

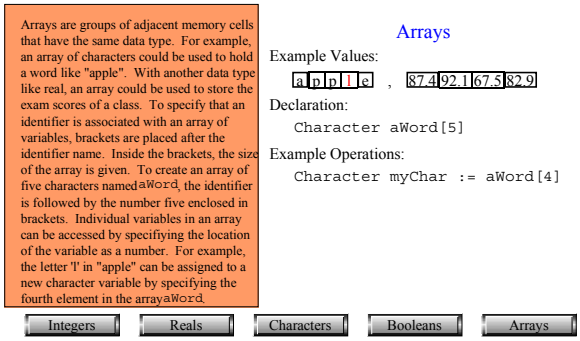


Figure 4.49 Data Types Animation

Link: http://courses.cs.vt.edu/csonline/ProgrammingLanguages/Lessons/DataTypes/index.html

4.7.5 Selection Exercises

These animations test for understanding of conditional logic in the form of if-else clauses. Each animation consists of a set of light bulbs and switches that are interrelated by some unknown logic (see Figure 4.50). Students must experiment with the system by changing the position of the switches on the left side and testing each new state to determine the relationship between the switch positions and the flow of electricity to the light bulbs. Students can check their answers by filling in a JavaScript form below the animation. They may also elect to give up and request the correct answer or continue to the next exercise.

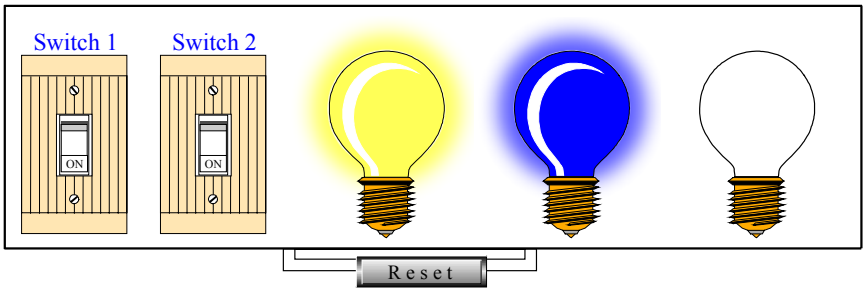


Figure 4.50 Selection Exercises

Link: http://courses.cs.vt.edu/csonline/ProgrammingLanguages/Lessons/Selection/index.html

4.7.6 Loops Demonstration Animation

This animation demonstrates how the loop construct works in procedural programming languages. The animation traces through a program designed to compute the average of a list of five grades. The left side of the animation displays the program in pseudocode with a blue bar indicating the line that is currently executing (see Figure 4.51). The current value of each identifier is displayed above the blue bar. The right side shows the variables and constants used in the program. As the animation progresses, these values change according to the program execution on the left. Students can step through the program or execute it entirely by using the controls at the bottom of the animation.

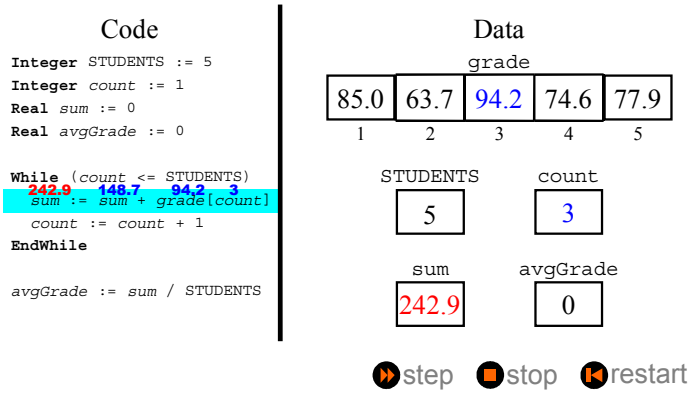


Figure 4.51 Loops Demonstration Animation

Link: http://courses.cs.vt.edu/csonline/ProgrammingLanguages/Lessons/Loops/index.html

4.7.7 Call/Trace Power Animations

These animations present the concept of a subprogram as an abstract "black box" that performs a particular function. In this case, the black box is called "Power" and performs exponentiation. In the Call Power animation, a student enters x and y values and presses the "Call Power" button to begin the calculation (see Figure 4.52). However, the student never sees the implementation of the function. Instead, the x and y values simply enter the left side of the black box and the correct answer exits the right side. In the Trace Power animation, the student follows the x and y values inside the black box

and traces the actual code that performs exponentiation. This trace is similar to the one found in the Loops Demonstration animation above.

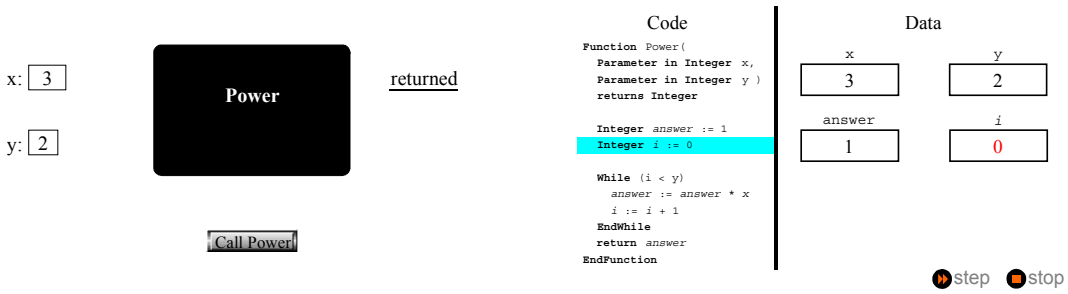


Figure 4.52 Call/Trace Power Animations

Link: http://courses.cs.vt.edu/csonline/ProgrammingLanguages/Lessons/Subprograms/index.html

4.7.8 Parameter Passing Animation

This animation illustrates the differences between the pass-by-reference and pass-by-value methods of passing parameters to a subprogram. During the animation two calls are made to a subprogram called Swap (see Figure 4.53). The subprogram takes two parameters and exchanges the values of these parameters. The first call to the subprogram shows that passing parameters by reference results in the expected behavior. The second call to the subprogram shows that passing parameters by value will not exchange the original values even though the Swap subprogram executes.

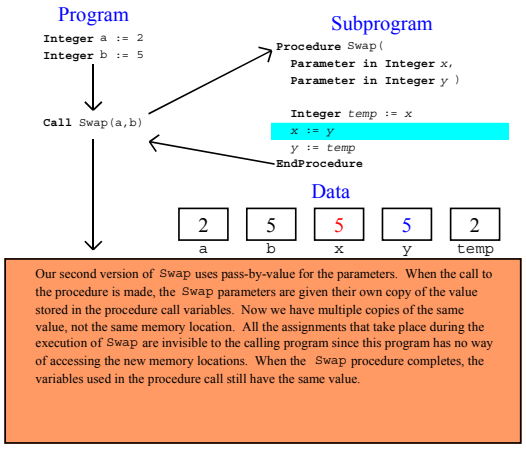


Figure 4.53 Parameter Passing Animation

Link: http://courses.cs.vt.edu/csonline/ProgrammingLanguages/Lessons/Parameters/index.html

4.7.9 Selection Sort Applet

This applet allows students to step through the entire Selection Sort program and watch the values of the program variables change in memory as it executes. The main window of the applet displays the program code with the current line of code highlighted (see Figure 4.54). Students can enter up to 10 numbers for the applet to sort. This is done through a dialog box that appears when the program trace reaches the line of code for input. During the trace, the Variables window shows the current values of all the program variables and highlights values accessed by the current line of code. The Numbers window represents the array of numbers to be sorted. As the sort progresses, students can watch these numbers shift till the array reaches its final sorted order.

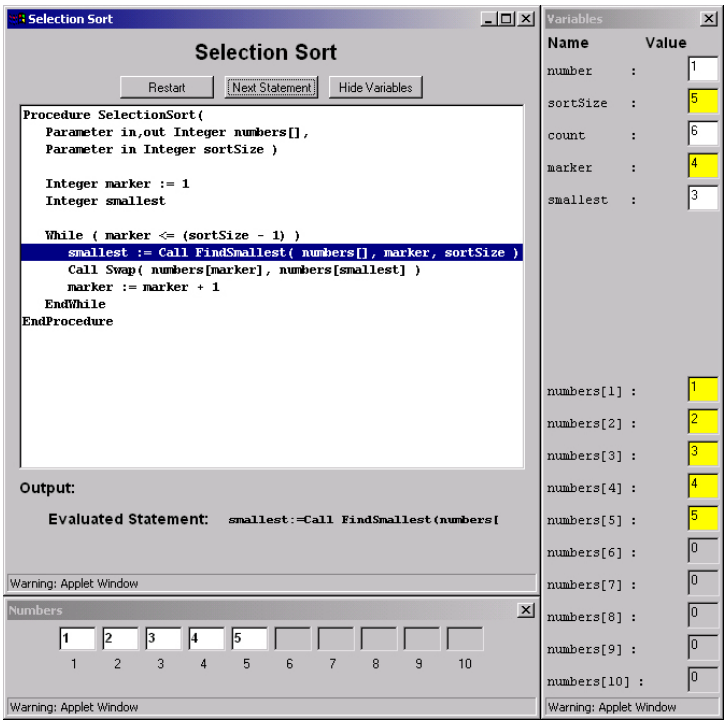


Figure 4.54 Selection Sort Applet

Link: http://courses.cs.vt.edu/csonline/ProgrammingLanguages/Lessons/Programs/index.html

4.8 SOFTWARE ENGINEERING

The Software Engineering Module consists of 10 lessons that introduce the topic of software engineering by comparing two important models of the software life cycle, the Waterfall Model and the Spiral Model, and two paradigms for designing software, the

Procedural Paradigm and the Object-Oriented Paradigm (OOP). The learning objectives for the module are as follows:

- Recognize the challenge of writing correct software,
- Understand and reproduce the phases of the software life cycle,
- Compare the procedural paradigm with the OOP, and
- Understand the principles of the OOP.

Table 4.8 lists each of the lessons in this module and briefly describes the content of each lesson. The right column of the table indicates any interactive components that are embedded in the lesson.

Table 4.8 Lessons in the Software Engineering Module

Lessons	*Interactive Components* †
Introduction to Software Engineering – illustration of the current challenges to Software Engineering	• Software Engineering Quiz (J)
Software Life Cycle Models – discussion of the six processes in the software life cycle	
The Waterfall Model – explanation of this model as a series of processes and products	• Waterfall Model (F) • Waterfall Model Quiz (F)
The Spiral Model – explanation of this model as a risk-driven, prototyping development approach	• Spiral Model (F)
Software Quality Characteristics – discussion of six important software quality characteristics	
Procedural Paradigm – discussion of the main characteristics of the procedural paradigm	• Selection Sort (J)
Object Oriented Paradigm – introduction to objects, messages, and encapsulation	• Queue (J) • Bag Abstract Data Type (J)
Classes and Inheritance – introduction to classes, instantiation, and inheritance	• Inheritance (F)
Comparison of Paradigms – assessment of paradigms according to six software quality characteristics	
Summary – review of the main ideas in Software Engineering	

† (F) – Flash animation (J) – Java applet

4.8.1 Software Engineering Quiz Applet

This applet tests students' knowledge of some surprising software engineering facts by asking a series of nine questions. Many of the answers are unintuitive and highlight the current challenges in software engineering. For each question, students can

select from four possible answers using the mouse and submit their answer by clicking the button at the bottom of the applet (see Figure 4.55).

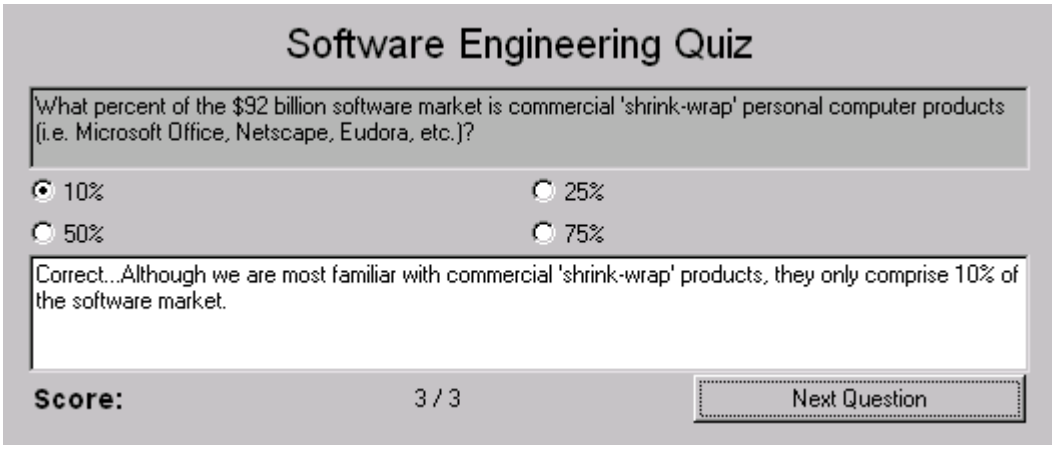


Figure 4.55 Software Engineering Quiz Applet

Link: http://courses.cs.vt.edu/csonline/SE/Lessons/Introduction/index.html

4.8.2 Waterfall Model Animation

This animation explains the various processes of the Waterfall Model, the classic software life cycle model. After a brief introductory sequence in which the meaning of the various parts of the Waterfall Model diagram are explained, students can click on a process to view a scene that explains how that process works. The screen shot below (see Figure 4.56) shows the scene that explains the details of the programming process. Students may also choose to view all the scenes explaining the Waterfall model in sequence. The animation emphasizes the iterative nature and product-driven approach of the Waterfall Model.

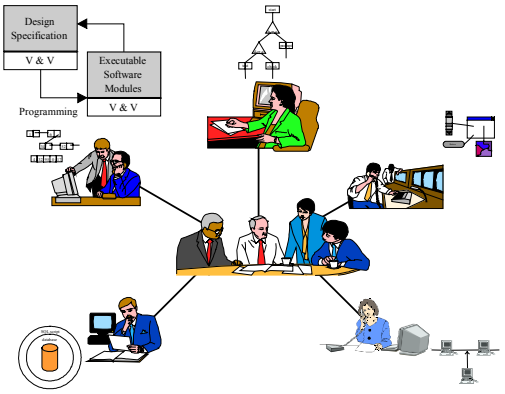


Figure 4.56 Waterfall Model Animation

Link: http://courses.cs.vt.edu/csonline/SE/Lessons/Waterfall/index.html

4.8.3 Waterfall Model Review Quiz

This animation allows students to test their knowledge of the Waterfall Model by labeling a skeleton diagram of the model (see Figure 4.57). Students use the mouse to drag labels to the correct location on the diagram. Using the "Check Answers" button, students can then test their answers to see if they correctly labeled the diagram or if some of the labels are out of place.

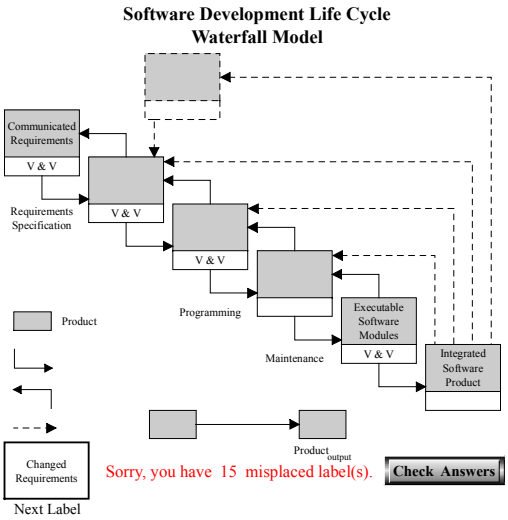


Figure 4.57 Waterfall Model Review Quiz

Link: http://courses.cs.vt.edu/csonline/SE/Lessons/Waterfall/index.html

4.8.4 The Spiral Model Animation

This animation explains the underlying concepts of the Spiral Model life cycle. The animation begins by describing the four steps that make up a single loop of the spiral. With each consecutive loop, these steps are repeated resulting in new prototypes and repeated project risk assessment. The screen shot below (see Figure 4.58)shows several prototypes that result from development loops in the spiral. The last scene of the animation shows how the final steps of the Spiral Model correspond to the final three processes of the Waterfall Model: programming, integration, and delivery.

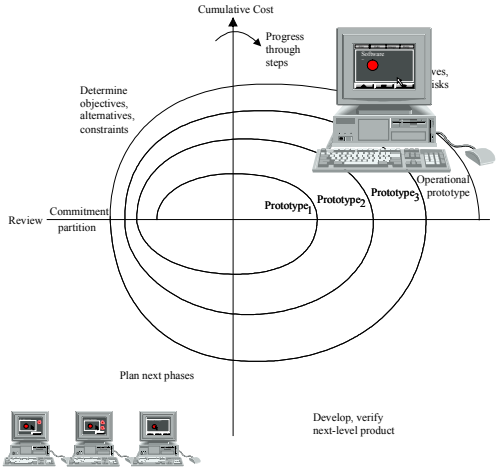


Figure 4.58 Spiral Model Animation

Link: http://courses.cs.vt.edu/csonline/SE/Lessons/Spiral/index.html

4.8.5 Selection Sort Applet

This applet is the same as the one described in the preceding section on Programming Languages. The applet is included in this module to illustrate one of the primary characteristics of software designed with the procedural paradigm: sequential logic. Students are asked to trace the logic of the Selection Sort through its various routines using the applet.

Link: http://courses.cs.vt.edu/csonline/SE/Lessons/Procedural/index.html

4.8.6 Abstract Data Type Applets

These two applets are the Queue Applet and the Bag Applet that were described in the Data Structures section. They are included in this module to show how objects can

be viewed as abstract data types that support encapsulation and provide a public interface for performing operations.

Link: http://courses.cs.vt.edu/csonline/SE/Lessons/OOP/index.html

4.8.7 Inheritance Animation

This animation explains the concept of inheritance among classes in the Object-Oriented Paradigm. The animation considers the problem of modeling shapes in a drawing program and shows how various types of shape objects can be organized into an inheritance hierarchy based on their shared characteristics. The screen shot below (see Figure 4.59) shows the inheritance hierarchy and attributes for various shapes.

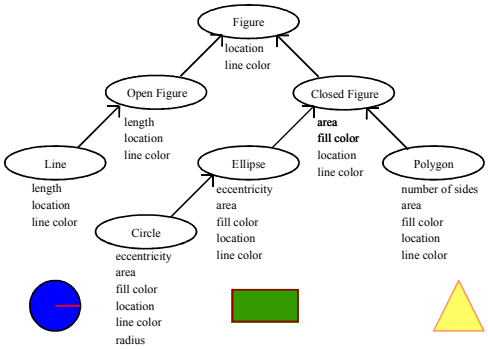


Figure 4.59 Inheritance Animation

Link: http://courses.cs.vt.edu/csonline/SE/Lessons/Classes/index.html

5 CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

5.1 CONCLUSIONS

In conclusion, we would like to say a word about the goal of this project to improve the effectiveness of learning. Although the project did not include a formal assessment of the online modules, we believe the material does improve the effectiveness of learning for the following three reasons: (1) accessibility of materials, (2) level of student engagement, (3) informal student responses.

The first reason we believe the modules improve the effectiveness of learning is their accessibility. Since the eight learning modules are always accessible to students via the WWW, they can review information and learn at their own pace outside of class. Consider the problem of teaching binary arithmetic. During a lecture session, the professor has a limited amount of time in which to explain and illustrate the rules of binary arithmetic. Our experience shows that many students will not understand the example problems that the professor works in class. Often these students will ask for the problem to be repeated, a request that may not be practical given the time constrains. The module on binary arithmetic provides a practical solution to this problem. This module includes five animations illustrating the rules of binary arithmetic, and each animation is divided into logical sections that the student can repeat. Thus, what was unreasonable in class is now very reasonable through animation technology.

The second reason we believe the modules improve the effectiveness of learning is the opportunity for greater student engagement during study. Many of the lessons in the modules present the material with interactive components that are more engaging than traditional textbook reading. In the introduction, we cited the example of the ELIZA applet and its contribution to understanding artificial intelligence in computers. In addition to this, the AI module also includes an eight-puzzle applet, a checkers applet, and a handwriting recognition applet. As students interact with these applets, they develop their own intuition about the meaning of artificial intelligence. They also encounter unanswered questions that can become the motivation for further inquiry. For

example, Figure 5.1 below shows the results of three attempts to recognize the numbers "6" and "1" using the handwriting recognition applet. Clearly, the results of the second attempt are wrong, but the logical question that follows is, "What exactly went wrong?" A comparison of the second and third attempts shows that the recognition algorithm is very sensitive to small bends. Thus, the tiny bends at the top of the two "1" digits make a significant difference in the outcome. This discovery leads to many more interesting questions such as "Why is the algorithm so sensitive to bends?" and "How can the algorithm be improved to avoid this problem?". This type of inquiry is difficult to motivate from a textbook or lecture since students cannot immediately test their hypotheses. The interactive components such as the handwriting recognition applet encourage students to explore and discover on their own, and we feel this contribution improves the effectiveness of learning.

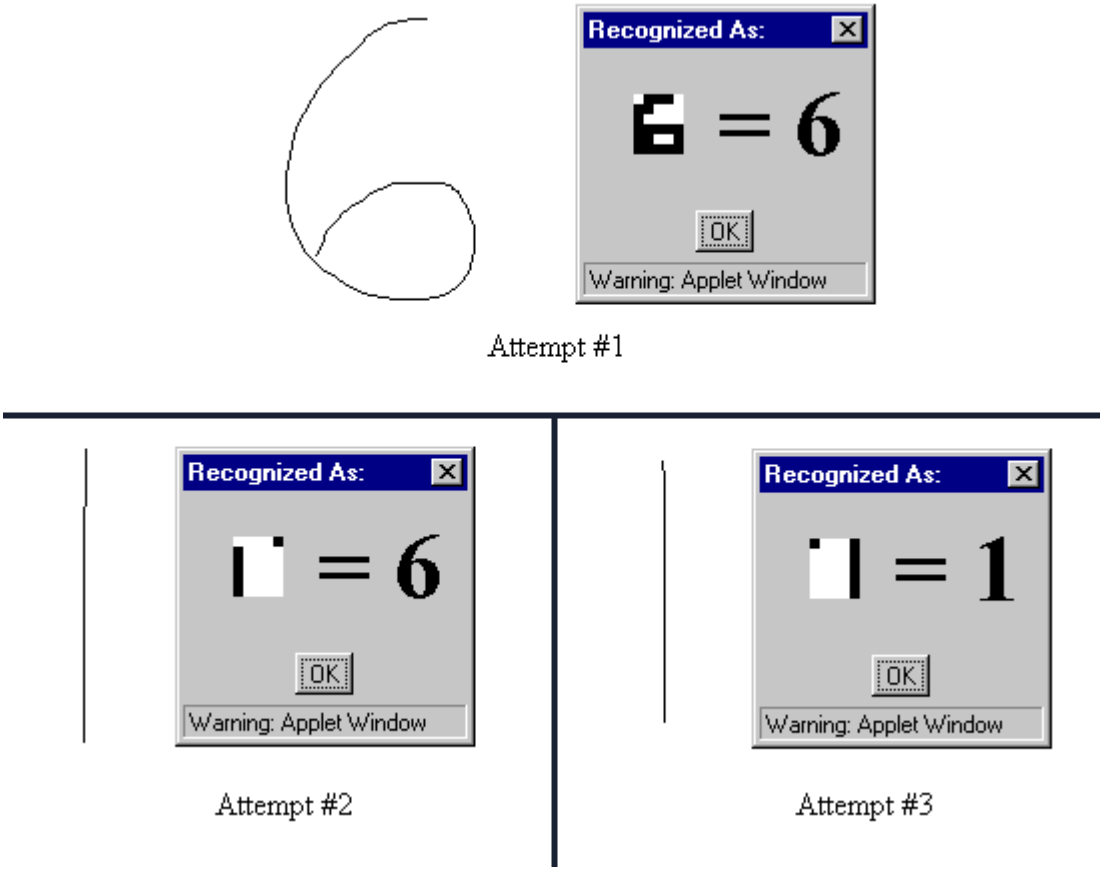


Figure 5.1 Handwriting Recognition Examples

Another important aspect of student engagement is the use of multiple senses in the learning process. In summarizing their review of video media, Wetzel *et al*. [1994] conclude,

> The studies reviewed generally support the idea of a benefit to simultaneous verbal and visual information when they are carefully matched. Combining visual and verbal information in video presentations generally leads to either equal or better learning compared to when these sources are given alone.

Similarly, Rieber [2000] cites a study [Mayer and Anderson 1991] in which students were given the same animation to watch with and without narration. He summarizes the results by saying,

> Students given the animation along with the narration significantly outperformed students who either in isolation watched the animation or heard the narration or who heard the narration right before seeing the animation on the problem-solving tasks. Even more important, the animation without the verbal description was completely ineffective, as students in this treatment compared equally with students provided no instruction at all.

These results explain our decision to include narration in many of the animations. The use of narration seems particularly important, however, in the animations on binary arithmetic. By explaining each arithmetic technique with narration, students are free to focus their eyes on the animated illustration rather than explanatory text.

The third reason we believe the modules improve the effectiveness of learning is the informal responses we have received from students using the modules. Fifty four undergraduate students enrolled in Intermediate Software Design and Engineering were asked for their opinion of the SE module which was accessible from the class web site. The students unanimously agreed that the lessons and animations explaining the Waterfall Model and the Spiral Model were more helpful than reading the textbook alone.

5.2 RECOMMENDATIONS FOR FUTURE WORK

Finally, we present several suggestions for future work on the learning modules. These suggestions fall into three categories: extension work, revision work, and assessment work.

Many possibilities exist for the extension of the current modules. First, the modules can be extended through the addition of new modules that cover other topics in Computer Science such as Human-Computer Interaction and Numerical Analysis. The

existing modules can also be enhanced through the addition of new lessons and interactive components. Second, many of the interactive components can themselves be enhanced. One example is the Memory Allocation Applet in the Operating Systems module. One logical extension to this applet would be the addition of a pause button to the interface so that students can pause the simulation and compare the current state of the three memory representations. Third, the review questions can be expanded to include "concept extension" questions that require the student to think beyond the scope of the material presented. Consider the question of comparators for sorting. In the Algorithms module, a numerical comparator is assumed for all the sorting examples. However, what would need to change in the sorting algorithms if the goal was to sort names rather than numbers? This questions embodies the principle of concept extension. The question directs students to apply the knowledge they have learned to a similar yet different problem.

Some revision work is also needed to maintain the consistency of the lessons and to improve the clarity of the lessons. The first type of revision is simply a necessary evil for any web-based entity. Over time the links within the lessons will break and need updating. In addition, the facts about the current state of computing will become obsolete and need revision. The second type of revision is more subtle and difficult. This work is not simply the correction of errors but rather the arduous task of discerning how much detail to abstract away when explaining new concepts to students. For example, the lesson on identifiers in the Programming Languages module concludes with the following statement:

> One last note about identifiers: In most programming languages, identifiers are required to conform to a certain format. For example, the identifiers in this lesson all began with letters and were composed only of letters and numbers. None of the identifiers included spaces, and constants were written in UPPERCASE letters. This is a typical format for identifiers, and we will use this format in the rest of the lessons.

While this word on the formatting of identifiers is certainly true, it is hardly exhaustive. The question then becomes, "How much detail should the lesson present?" Where is the balance between completeness and conciseness, accuracy and abstraction? Clearly, this type of revision work requires a great deal of careful thought.

Finally, the opportunity exists for a formal assessment of the learning modules to determine the various ways that student learning was influenced. Such an assessment could include more detailed student critiques of the material specifying which lessons and interactive components were particularly helpful and which were not helpful. This information would be a valuable guide to help with the revision of the existing modules and the extension of the work with future modules.

BIBLIOGRAPHY

Arase, K. (1999), "Simcir the Circuit Simulator,"
http://www.tt.rim.or.jp/~kazz/simcir/simcir.html

Arny, T. (2000), "Explorations: An Introduction to Astronomy,"
http://www.mhhe.com/physsci/astronomy/arny/student/anims/index.mhtml

Bergin, J. (1997), "Web Based Visualization," http://sol.pace.edu/webvis/

Brookshear, J. G. (1997), *Computer Science: An Overview*, Fifth Edition, Addison-Wesley, Reading, MA.

Brummund, P. (1997), "The Complete Collection of Algorithm Animations,"
http://www.cs.hope.edu/~alganim/ccaa/ccaa.html

Chang, R. (2000), "Essential Chemistry Flash Animations,"
http://www.mhhe.com/physsci/chemistry/essentialchemistry/flash/flash.mhtml

Coutinho, M. (1996), "Virtual Memory Simulation Applet,"
http://www.isi.edu/people/coutinho

CQU (2000), "Operating Systems 85349,"
http://www.infocom.cqu.edu.au/Units/win2000/85349/Resources/Animations/

CSTC (1999), "The Computing Science Teaching Center," http://www.cstc.org/

Decker, R. and S. Hirshfield (1998), "The Analytical Engine Online,"
http://www.brookscole.com/compsci/aeonline/course/

Denning, P. and D. Menascé (1998), "The HyperLearning Center Workbenches,"
http://cne.gmu.edu/workbenches/index.html

Enger, E. and B. Smith (2001), "Environmental Science,"
http://www.mhhe.com/biosci/pae/environmentalscience/enger7e/enger_animation-quizzes.mhtml

Fabio, F. (1997), "Checkers Applet,"
http://www.geocities.com/SiliconValley/8381/checkers_en.html

Goerlich, R. (1996), "ELIZA Applet," http://www.cyberloft.com/bgoerlic/index.htm

Gokey, C. (1997), "Fragmentation Example,"
http://asia.cs.bowiestate.edu/~cgokey/fragment/example.html

JERIC (2000), "ACM Journal on Educational Resources in Computing," http://fox.cs.vt.edu/JERIC/index.html

JARS (2001), "Java Applet Rating Service," http://www.jars.com/

Magee, J. and J. Kramer, (1999), "Concurrency: State Models & Java Programs," http://www-dse.doc.ic.ac.uk/~jnm/book/index.html

Mayer, R. E. and R. B. Anderson (1991), "Animations Need Narrations: An Experimental Test of a Dual-Coding Hypothesis," *Journal of Educational Psychology 83*, 4, 484-490.

Mitchell, B. (1998), "Java Handwriting Recognition Applet," http://members.aol.com/Trane64/java/JRec.html

Moosani, H. (1998), "Graphical Simulation of Disk Block Allocation Methods," http://unity.njit.edu/students/fall98/haritha/frame_page.htm

Naps, T., J. Bergin, R. Jiménez-Peris, M. McNally, M. Patiño-Martínez, V. Proulx and J. Tarhio (1997), "Using the WWW as the delivery mechanism for interactive, visualization-based instructional modules," In *Supplemental Proceedings of the Conference on Integrating Technology into Computer Science Education: Working Group Reports and Supplemental Proceedings (ITiCSE-WGRSP '97)*, Association for Computing Machinery, pp. 13-26.

Palakal, M. (1997), "Personally Active Computing Exploration Resource," http://klingon.cs.iupui.edu/~pacer/index.html

Pilgrim, C., Y. Leung and D. Grant (1997), "Cost Effective Multimedia Courseware Development," In *Proceedings of the Conference on Integrating Technology into Computer Science Education (ITiCSE '97)*, Association for Computing Machinery, pp. 45-50.

Rieber, L. P. (1990a), "Animation in Computer-Based Instruction," *Educational Technology Research and Development 38*, 1, 77-86.

Rieber, L. P. (1990b), "Using Animation in Science Instruction with Young Children," *Journal of Educational Psychology 82*, 1, 135-140.

Rieber, L. P. (1991a), "Effects of Visual Grouping Strategies of Computer Animated Presentations on Selective Attention in Science," *Educational Technology Research and Development 39*, 4, 5-15.

Rieber, L. P. (2000), *Computers, Graphics, and Learning*, http://www.nowhereroad.com/cgl/index.html

Schank, R. (1994), "Active Learning through Multimedia," *IEEE Multimedia 1*, 1, 69-78.

Shaffer, C. A. (2000), "Statistics Activity-Based Learning Environment," http://simon.cs.vt.edu/SoSci/converted/index.html

Song, Q. (1997), "Simulating Memory Page Replacement," http://iris.nyit.edu/~qsong/

Tran, Q. (1998), "Algorithm Animation Applet," http://www.utdallas.edu/~ilyen/animation/cpu/program/prog.html

Tsichritzis, D. (1999), "Reengineering the University," *Communications of the ACM 42*, 6, 93-100.

Wetzel, C., P. Radtke and H. Stern (1994), *Instructional Effectiveness of Video Media*, Lawrence Erlbaum Associates, Hillsdale, NJ.

VITA

William Shadwell Gilley, the son of William F. and Pamela M. Gilley, was born on August 29, 1976, in Lynchburg, Virginia. He graduated from Timberlake Christian High School in May 1994. He received an Academic Merit Scholarship to attend Central Virginia Community College and graduated from this institution in May 1996, with his A.A. degree in Business Administration. Three years later he graduated from Virginia Tech with his B.S. degree in Computer Science and minors in Mathematics and Professional Writing and Language. This thesis completes his M.S. degree in Computer Science from Virginia Tech.