THE MULTIMEDIA ANIMATION FOR LEARNING SPLAY TREES

By

YINGJIE DONG

Bachelor of Arts
Hebei Teacher's University
Shijiazhuang, Hebei
People's Republic of China
1990

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1998

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1999

THE MULTIMEDIA ANIMATION FOR

LEARNING SPLAY TREES

Thesis Approved:

Jacques E. LaFrance

Thesis Advisor

J Chandler

Blayne E. Mayfield

Wayne B. Powell

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to extend my sincere appreciation to my committee members for their guidance and support. I am particularly grateful to my advisor, Dr. Jacques LaFrance, for his time and efforts invested in my study and his sincere friendship with me. My thanks also go to Dr. John P. Chandler and Dr. H. K. Dai for their great help in my study and their time of serving as members of the committee.

My special thanks go to my husband, Yimin Yang, for his great love and continuous support in the whole journey of my graduate study. My special thanks are also extended to my parents, Zhilin Zhang and Wenli Dong, who continuously encouraged and supported me in every aspect of my life's endeavors. Their great love helped me overcome all the difficulties I have met in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

With the rapid development of technology, great changes have occurred in teaching and training methods. Computers are being more and more widely used in schools all over the world due to the decrease in computer costs and the development of more, better, and more diversified learning materials designed for personal computers (courseware) (Salomon, 1989). Multimedia was born with the increasing power of the computers and the compact-disc technologies. Visualization can help people understand complicated models and algorithms easily and is becoming more and more important in education, science, business industry, and military.

The uses of multimedia are endless. Multimedia can be used in marketing and advertising, staff and program development training presentations, and sports. For staff development, multimedia can make the training become more economical. Multimedia can also be revised and upgraded at a low cost and the trainees retain more knowledge with the multimedia training (Villamil and Molina, 1997).

Numerous evaluations conducted during the past ten years indicated the power and efficiency of interactive media as an ideal training tool. It was reported that the training times had been improved by between 40 and 60 percent by the use of interactive video (Feldman, 1994).

Multimedia is considered as a combination of at least three of the following media: text, graphics, still pictures, animation, video, and audio(Tucker, 1997). When the texts, graphics, and the sound are used together, much more attention can be paid by the human being because both left and right brain will be actively involved in processing the information. Hence, multimedia can promote a better understanding of the abstract concepts.

Data structures are widely used in the computer programs. Arra (1992) summarized that a data structure is the mathematical model of the data in an Abstract Data Type (ADT) which is a mathematical abstraction with data and a collection of operations defined on that data. When we mention the ADT, two things come up to mind. That is data and operations applied to that data.

The tree-based data structures occupy a very important status among the data structures and are frequently used in the computer programming. The splay tree is a unique tree-based data structure. It helps make future access cheaper on all the nodes by restructuring the depth of these nodes. It involves the rotations under zig-zig and zig-zag cases.

Statement of the Problem

The data structure plays a critical role in doing the computer programming. In order to meet the requirements of the computer jobs which are needed a lot by the current labor market, the computer science students must understand and grasp the concept and operations of the data structures very well. However, the current data structure class is limited to the traditional teaching method. The computer-based training has not been widely used in the class due to the lack of the software which gives the graphical representations of the data structures. Actually, the operation of the data structures involves a lot of node movement and node traversal. It would be much easier for the students to study the data structures if they could view the graphical representations of the structure and its various operations. The splay tree operation involves accessing the identified node and moving it to the root of the tree by doing the splaying. A series of node movement occurs during the operation. Hence, it would be much more vivid by using the multimedia animation to show the concept and operations of the splay tree than the other data structures. The software designed specifically for the splay tree will definitely help the students understand this kind of data structure easily.

Purpose of the Study

The purpose of this study was to develop a flexible, interactive, and user-friendly educational software to study one of the tree-based data structures – splay tree. The bottom-up algorithm would be applied to the operations of the splay tree in the software to be developed. The software would work as an individual teaching module and help the students study the splay tree through the visualization of the data structure. The software was developed by using Macromedia Director 6 PC version and was run under Windows 95, 98, Windows NT authoring environments as well as Mac OS on both 68k and power PC systems. The simulator can also be put on the internet with shockwave plug-in as a tool for long-distance learning.

Outline of the Work

The study background, problem statement, and the purpose of the study were described in Chapter I. In Chapter II, the literature of the data structures, the technology-based training, and the related works would be reviewed. Chapter III would address the design and implementation issues. The conclusions of the study and the recommendations for future study would be given in Chapter IV.

CHAPTER II

REVIEW OF THE LITERATURE

Introduction

This study focused on the development of the educational software which showed the operations of the selected data structure – splay tree by using the multimedia animation. Three areas were examined in the review of literature: data structures, technology-based training, and the related works.

Data Structures

Baron and Shapiro (1980) defined the data structures as "a structure whose elements are items of data, and whose organization is determined both by the relationships between the data items and by the access functions that are used to store and retrieve them" (p. 1). They emphasized that data structures were widely used in all areas of computer science, from application programming to theory, from microprocessors to large-scale computer systems, and from hardware to operating systems.

Reingold and Hansen (1983) summarized that a data structure consists of three components including: a) a series of function definitions; b) a storage structure which specifies classes of values, collections of variables, and relations between variables to implement the functions; and c) a set of algorithms.

Aho, Hopcroft, and Ullman (1983) put forward that there exists difference among "data type", "data structure", and "abstract data type". They defined the data type of a variable as the set of values that the variable may assume. An abstract data type was considered as a mathematical model with a collection of operations defined on that model, while data structures were defined as collections of variables, probably of several different data types, connected in various ways.

Stubbs and Webre (1987) described the four basic structural relationships as follows:

> A structure in which there is no relationship among the elements other than their belonging to the set of elements comprising the data structure is closely related to the mathematical notion of a set. Structures in which each element is related to one other element (a one-to-one relationship) are said to be linear. Those in which the relationships are one-to-many are called tree or hierarchical structures. Sets of elements in which the relationships are many-to-many are said to have a graph (in the graph theory sense) or network structure (p. 30 – 31).

Stubbs and Webre (1987) displayed the basic structural relationships in the following figure:

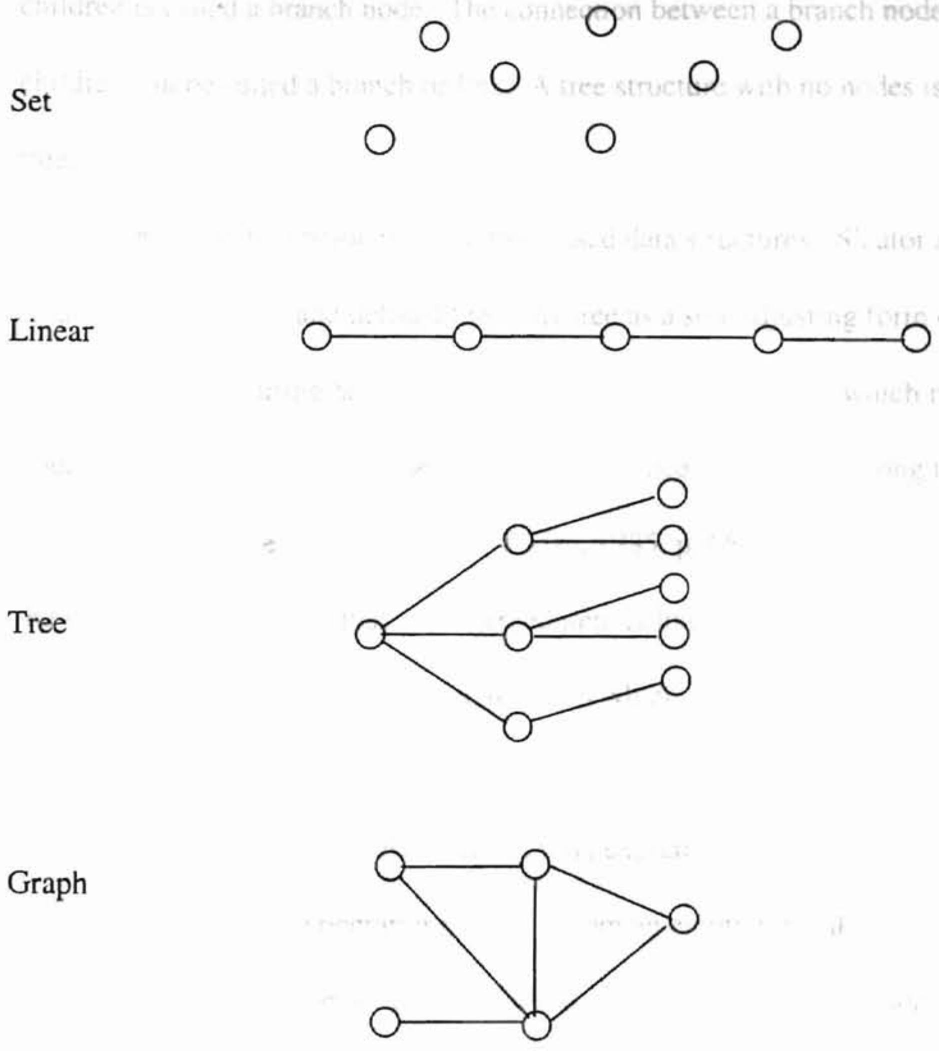


Figure 1: Basic structural relationships

The tree-based data structures occupy an important role in data structures. According to Baron and Shapiro (1980), a tree is used to represent a hierarchical relationship among items of data. This hierarchical structure is made up of cells named nodes. The distinguished node at the top level (level 0) of the hierarchy is called the root node. Each node of the tree is the parent of zero or more child nodes associated with the parent. The children of one parent node are called siblings and are in one level below the level of their parent. A node with no children is called a leaf node, while a node that has children is called a branch node. The connection between a branch node and one of its children can be called a branch or link. A tree structure with no nodes is called a null tree.

The splay tree belongs to the tree-based data structures. Sleator and Tarjan (1985) created the splay tree and defined the splay tree as a self-adjusting form of binary search tree. "The restructuring heuristic used in splay trees is splaying, which moves a specified node to the root of the tree by performing a sequence of rotations along the (original) path from the node to the root" (Sleator & Tarjan, 1985, p. 653). The splay tree is based on the plausible assumption that the accessed item is likely to be accessed again soon. Ramaiyer (1998) described in his dissertation which was put on the web site that the splay trees are self-adjusting binary trees and form a simple and very interesting class of "balanced" binary search trees. Weiss (1997) explained that the splay tree ensures that any M consecutive tree operations starting from an empty tree take at most O(MlogN) time. N refers to the number of the nodes in the tree. The basic implementation method for the splay tree is that once a node is accessed, it is pushed to the root by a series of rotations which include zig, zig-zig, and zig-zag rotations. Different from AVL tree, the splay tree does not need maintain the height or balance information. The splaying not only moves the accessed node to the root, but also roughly halves the depth of most nodes on the access path.

There are several variations of splay trees including the bottom-up and top-down variations. This study utilizes the bottom-up method to develop the software. The splaying rotations involve three cases including zig, zig-zig and zig-zag cases. Weiss

(1997) described the zig-zig and zig-zag cases and their corresponding rotation methods as follows (p. 126):

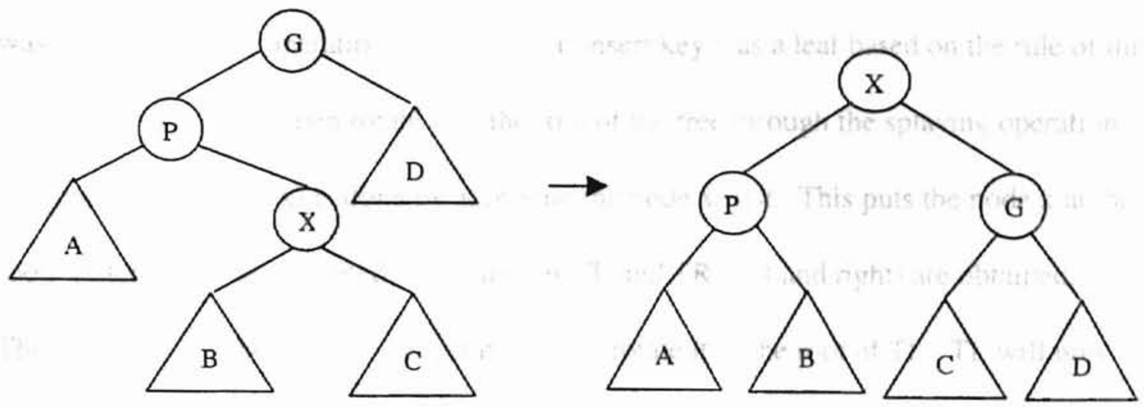


Figure 2: Zig-zag

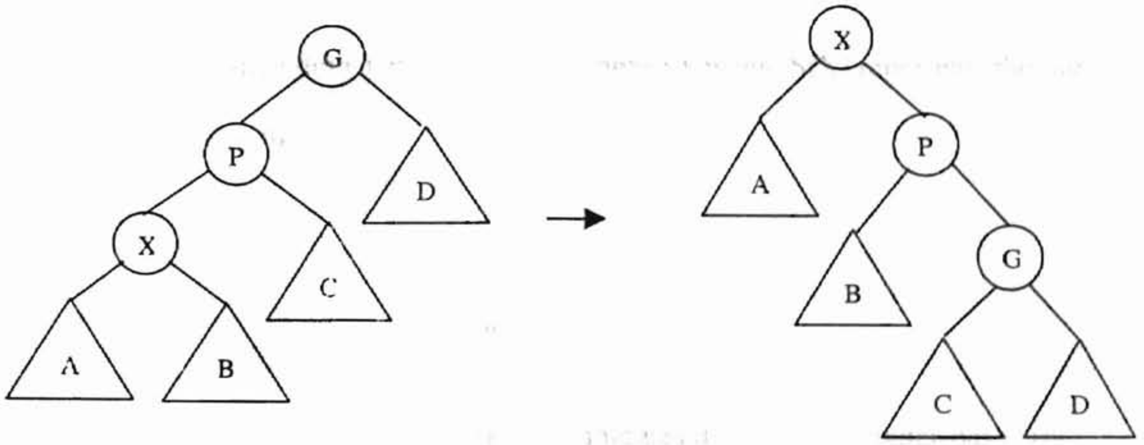


Figure 3: Zig-zig

Obviously, under the zig-zag case, a standard AVL double rotation is performed, while the zig-zig case is two consecutive single rotations.

According to Moret & Shapiro (1991), the splaying is the only operation that is directly performed on the tree. The operation splay (x) is to promote the node with key x to the root of the tree by a sequence of rotations after this node is found. The operations of find, insert, and delete are all defined based on the splaying operation. The operation find (x) is nothing but splay (x), followed by a check to see whether or not the desired key was in the tree. The operation insert (x) is to insert key x as a leaf based on the rule of the binary search tree and then rotate it to the root of the tree through the splaying operation. The operation delete (x) is done by accessing the node x first. This puts the node x at the root. After the node is deleted, two subtrees TL and TR (left and right) are obtained. Then we can find the largest element in TL and rotate it to the root of TL. TL will now have a root with no right child. We can finish the deletion by making TR the right child.

Weiss (1997) described the amortized time to splay a tree with root T at node X is at most $3(R(T)-R(X))+1 = O(\log N)$. R(T) is the rank of root T of the tree, while R(X) is the rank of the accessed node X. $R(T) = \log S(T)$, where S(T) represents the number of descendants of T (including T itself). $R(X) = \log S(X)$, where S(X) represents the number of descendants of X (including X itself).

Technology-Based Training

The successful use of established technologies including computer-based training and interactive video, as well as the emerging multimedia and communication technologies are promoting performance improvement in training and development and making a positive contribution to the teaching-learning transactions. "The term 'technology-based training' is applied to training or learning which is undertaken using computer and/or communications technologies to enable learning to take place" (Tucker, 1997, p. 3).

An Overview of Various Educational Technologies

Technology-based training involves computer-based training, interactive audio, interactive video, digital video interactive, expert systems/artificial intelligence, multimedia, CD-ROM, CD-ROM(XA), compact disc interactive, digital video disc, simulation, virtual reality, video conferencing, desk-to-desk conferencing, satellite broadcasting, networks/internet/intranet, and electronic performance support systems (Tucker, 1997). Some selected technologies are reviewed in this study.

According to Tucker (1997), the training delivered, tested or managed by a computer is called the computer-based training (CBT). The learning becomes more individualized and self-paced by using the computer-based training.

With the advent of CD-ROM and compression techniques, interactive audio became a workable methodology. The capacity of storing more data in both main memory and hard disk together with the advent of sound processing cards brought a new era. The sound processing card made the student's voice be able to be recorded for the purpose of comparison and hence promoted the learning. Besides language training, interactive audio has also played its role in the courses covering human resource skills, such as interviewing (Tucker, 1997).

Copeland (1989) defined interactive video as "the presentation of video and audio information according to the response input made by the viewer" (p. 111). One of several devices such as video disc, videotape or compact disc can be the source of video. The video disc that could store both video and sound brought the possibility for true interactive video. Interactive video has allowed users to simulate complicated technical skills.

Digital video interactive (DVI) is a different approach to the medium of interactive video and uses digital video. It compresses the video and audio and stores the compressed data on a CD-ROM, and then decompresses the data for delivery on the screen. With DVI, it is possible to run the video at controlled variable speeds (Tucker, 1997).

It is said that artificial intelligence and expert system would revolutionize technology-based training. But this did not happen because artificial intelligence required a huge amount of processing power and the programming of a course was expensive and very time consuming. One advantage of this type of system was that it was possible for learners to learn by discovery once the knowledge was in the system. The artificial intelligence/expert system has been utilized by some subject domains, such as, the field of medicine.

The simulation technology can be used as: "a) software products, such as word processing systems, and databases; b) form filling; c) office procedures; d) industrial processes; and e) train and road driving" (Tucker, 1997, p. 16).

Virtual reality is a stage up from simulation. It is a practical training option. In some cases that involves the simulation of precise instrument or weapons such as laser guns, virtual reality can be used in a training role. Aircraft simulators belong to virtual reality (Tucker, 1997).

Video conferencing uses video cameras, television monitors and multiple-channel telephone links. It is used for training and basic conferencing. The video conferencing can make the remote lecturer become available to an audience across a wide geographical spread. It can also help the tutor in one location be able to communicate with students in a number of locations. The students can interact with each other and hence effective learning is promoted (Tucker, 1997).

Galbreath (1997) defined the internet as "a network of networks with a universal addressing scheme allowing real-time, computer-to-computer, location-independent communication and information exchange" (p. 39). Today, the internet is used for various applications including electronic mail (E-mail), integrated messaging, voice and video traffic, electronic commerce, and world wide web. One growing use of the internet is to provide education and training. The internet web presents visually compelling content to train and educate people across the world. The trainers in the companies are using listservs, newsgroups, and on-line courses. Colleges and universities including the Open University and Mind Extension University are utilizing the internet to deliver education. According to Starr and Milheim (1996), many internet programs have already been conducted in primary and secondary education.

Carr (1992) defined the electronic performance support system (EPSS) as "a computer-based system that uses knowledge-based systems, hypertext, on-line reference, extensive databases, and allied technologies to provide support to performers on the job, where they need it, when they need it, in the form most useful to them" (p. 45).

Compact disc read-only memory (CD-ROM) is not a delivery method. Instead, it is a storage device. CD-ROM is a very good storage medium and is widely applied in training (Tucker, 1997). With the introduction of the CD-ROM, multimedia became available for the mass market.

Multimedia

Feldman (1994) defined multimedia as follows:

> Multimedia is the seamless integration of data, text, images of all kinds and sound within a single, digital information environment (p. 4).

Rosenborg and the others (1993) described today's common definition of multimedia as the integration of sound, animation, still images, hypertext, or video used in conjunction with computing technology.

Galbreath (1997) put forward that there were basically two types of applications for multimedia systems including symmetrical and asymmetrical. Symmetrical multimedia refers to those applications that require real-time communications such as video conferencing/video telephone. Asymmetrical multimedia communications refers to those applications that are store-and-forward in nature such as video-on-demand, and CD-ROM.

Rathbone (1995) summarized the elements of a multimedia program as described in table 1.

Table 1: Elements of a Multimedia Program

This Part of Multimedia	Allows for This
Text	This part of multimedia – displaying words on the screen – is the base layer of almost all programs. Text is till a quick way to spread information, so programs will always use it.
Pictures	Multimedia computers can display photograph-quality pictures on the monitor. Seeing a shiny garden slug on the screen carries a lot more impact than just reading about one.
Movies	With a multimedia program or part, your computer can turn into a TV set, letting you watch Bewitched reruns. Or the computer can store snippets of your own home movies onto disks, letting you mail baby-food movies to the relatives on a floppy.
Animation	Sometimes animation (cartoons) can express a point better than movies. Without animation, for example, nobody could show footage of dinosaurs biting each other. Some industry folks use the word animation to describe any type of moving picture, including movies.
Sound	A biting dinosaur isn't much unless you can hear the bones crunch, as well.
Increased control	Best yet, multimedia lets you jump around. Bored with the biting dinosaurs? Click on a button and switch to the flying pterodactyls, instead. Unlike a normal, television-style movie, a computerized multimedia program lets you skip past the boring parts and watch the fun stuff, over and over. Plus, there's no delay while the VCR rewinds; multimedia programs can jump quickly to different areas.

According to Villamil and Molina (1997), the educational multimedia programs include training programs, edutainment, games, cyberArt, magazine and newspapers, multimedia interactive kiosks. The training programs using the multimedia technology on the market covers from foreign language training programs, software applications training programs, programming languages, diets and nutrition, to self-help programs, cooking programs, and music education, etc.

Related Works

Several decades ago, it was hard to implement the computer technology-based teaching and training due to the high cost of the graphics hardware. Since the 1960s, people started to utilize visualization as a method to understand programs and algorithms. But at the beginning, the visualization was mainly implemented by using the films. A review of literature shows that Knowlton (1996) produced the first computer-generated movie named "L6: Bell Telephone Laboratory Low Level Linked List Language". The movie described how the list processing language works at the assembly level.

Several other important films were also produced following the first computer-generated movie. Hopgood's film about hashing algorithms (1974) was one of these films. This movie displayed a hash table, a graph showing the number of probes to finish the insertion of an item, and the maximum number of collisions in the course of item insertion. The actions based on different types of input were shown by using the hash table and the graph. Booth's (1975) PQ-trees described the actions of several algorithms on PQ-trees. Baecker's well-known work "sorting out sorting" (1981) is a good algorithm animation film which illustrated the visualization of a number of different sorting algorithms. Each sorting method was explained and the comparison on the performance of all the nine sorting methods was conducted based on sample data. The film was also accompanied with a narrative through a sound track. It was considered that Baecker's system was the first known system to aim at algorithm animation (Arra, 1992).

In the 1980s, there occurred several algorithm animation systems used for education. The well-known ones are BALSA-I (Brown University Algorithm Simulator and Animator) and BALSA-II. Brown (1988) stated that "the user can watch execution of an algorithm through various views. Each view is displayed in a window on the screen." BALSA-I (Brown, 1988) was developed in the early 1980s and used in the electronic classroom at Brown University. It focused more on algorithm animation and involved good interactivity and some Smalltalk techniques including popup menus, overlapping windows and the shape changes of the cursors. BALSA-II was later developed based on BALSA-I. It utilized several Macintosh user interface functions including zoom in/out and dialog boxes and displayed the detailed and overall views of an object simultaneously (Brown, 1988). These systems require internal Macintosh coding to create new animation view and the animation must be executed on a Macintosh. This constraint blocked the systems from being widely used.

Lee (1988) designed a system which provided the graphical representations of a variety of data structures and allowed the user to execute and study the step-by-step movement of an operation on a particular data structure. The display system was implemented for VT 100 type terminals. It can be executed using one-screen or two-screen modes.

Stasko (1990) introduced an algorithm animation framework named "TANGO" which helped simplify the animation design. The framework was based on four abstract data types: locations, images, paths, and transitions. Stasko (1997) did some research examining algorithm animations as a learning tool in computer science education. The project named "Evaluating Algorithm Animations as Learning Aids" and sponsored by Stasko (1997) sought to demonstrate whether algorithm animation could provide an effective pedagogical tool in algorithm instruction.

Shimomura and Isoda (1991) designed a system called 'Visualization and Interactive Programming Support Debugging System" (VIPS). The system presented linked-list visualization for debugging and used UNIX's debugger – DBX to execute the program to be debugged. It displayed linked list as syntax trees.

Arra (1992) developed a prototype to demonstrate data structure animation. The prototype includes a simple user-friendly Graphical User Interface (GUI) and a library that aids the dynamic animation of data structures. To run the package, the user needs Microsoft window 3.0 or later.

Shen (1994) displayed the dynamic behavior of the algorithms of several tree-based data structures including AVL tree, red-black tree, B-tree, and splay tree. This system was developed under the X window environment. According to Xu (1997), this system can not accept the input data from the users for visualization.

Vikas (1996) developed a graphical tool for the visualization of a number of sorting method. It was coded in C++ programming language and was designed to run on the sequent symmetry S/81 computer running the Dynix/ptx operating system.

Xu (1997) designed and implemented a simulator to help the user execute and visualize the operations on three ADTs including the linked list, stack, and the queue. It focused on the display of various operations associated with the three ADTs. The simulator is run on the Microsoft window 95 operating system.

Shen (1998) developed an animated presentation of the concept and algorithms of binary search trees. The simulator displayed various operations of the binary search tree including insertion, deletion, and inorder, preorder, and postorder traversal. It was implemented with Director and runs on both IBM PC and Macintosh systems.

A search of the Internet has found a complete collection of algorithm animations including the sorting, tree, geometric, graph, data structures and miscellaneous algorithms. The animations were developed by using JAVA programming language and can be accessed in the following web site: http://www.cs.hope.edu/~alganim/ccaa/algo.html. For the tree algorithm animation, several research institutes posted the animation of the algorithms. Among them, the University of Toronto did the B and B+ trees; Michael Bahl at the University of Hartford did the binary search tree; Georgia Institute of Technology did the binomial heap; Lawrence University did the heap insertion; and the University of Southern California posted the red-black tree, splay tree, and treaps (randomized search trees). However, these animations did not show the step-by-step movement of the operations of various algorithms. They just displayed the final results. No narratives accompanied the algorithm animation. It is necessary to develop an interactive educational software which works as an independent teaching module and shows the step-by-step movement of various operations of the selected algorithms through the multimedia animation.

CHAPTER III

DESIGN AND IMPLEMENTATION

Introduction

This chapter discussed the design and implementation issues of this study. Five areas were covered in this chapter including Implementation Software and Hardware, General Design of the Educational Software, Implementation Details, the Characteristics of the Software Design, Demonstration of the Animation with a Sample Node, and the Amortized Analysis. The area of Implementation Details covered the descriptions of the stage, cast members, frame and frame script which were used in the animation.

Implementation Software and Hardware

The educational software introduced in the study was developed by using Macromedia Director 6 PC version and run under Windows 95, 98, Windows NT authoring environment as well as MacOS on both 68K and power PC systems.

> Director lets you create, combine, and display various images on the computer, and allows your users to control what displays and when. It does this with a degree of detail and precision that other programs can't begin to match (Fisher, 1995, p. 3).

The Macromedia Director developed by Macromedia Inc, the leader in digital arts, multimedia and web publishing software, is the most powerful authoring tool for multimedia productions. The Director provides good interactivity and combines video with text and graphics. According to Fisher (1995), on one hand, Director was a tremendously powerful animation program; on the other hand, Director was an extremely powerful integration tool which introduced you a great way to combine disparate graphical elements, such as QuickTime movies, video clips, and sound effects coming from external applications or special hardware. Director 6 is the new and upgraded version of the Macromedia Director. Comparing with previous versions, Director 6 added several new things that are used to enhance the easiness and better effects of the visualization. It is an ideal tool to develop the flexible and interactive graphical representations for learning the selected data structures. The movie developed by the Director can also be put on the internet with shockwave plug-in for Director, which can definitely promote the long-distance learning.

Lingo is the scripting language that comes with Director. It is used to write scripts or instructions to create powerful interactive multimedia projects. Lingo adds powerful functionality to Director and creates animations that would be difficult to accomplish by only using the score. With the commands and functions of Lingo, the developer can create unforgettable non-linear presentations and applications (Bennett, 1997).

General Design of the Educational Software

The educational software developed in this study works as an individual teaching module for learning the selected tree-based data structure - splay tree. The animator aims to help the students study the splay tree in an effective and efficient way through visualizing the animation of the splay tree's operations. The software was designed to emphasize the splaying operation which is directly performed upon the splay tree. During the animation of the splaying operation, the number of the splaying steps for each accessed node was dynamically calculated through the "pointer change" box which was shown at the left corner of the stage. The software was designed to contain the following three parts:

Part I: Concepts and algorithms of the splay tree.

Part II: The animation of splay tree's basic operations.

Part III: Test.

Text was mainly used for part I and part III to explain the basic concepts, algorithms and the splaying steps of each accessed node and to help the students evaluate their learning from the splay tree algorithm animation. Static displays were used for these two parts. Part I consisted of several screens of different text descriptions and some "Next" buttons which were used to forward the screen one by one. Part III gave the students a chance to evaluate their mastery of the class. After finishing the test, the students can look at the correct answer by clicking "Answer" button shown at the bottom of the stage.

Part II demonstrated the animation of the splay tree's algorithms, operations, and the splaying steps of each accessed node. It is the core part of the whole software. The combination of the static displays and dynamic displays were adopted in this part. The static display was used to show the text of data, and the image of data structures. The dynamic displays were emphasized in this part to demonstrate the behavior of the operations and the changes in the data structure.

The software aimed to show the basic operations of the splay tree through the graphical representations and display the splaying steps of each accessed node during the animation. A splay tree with more depth should be used in order to show all the possibilities of the splaying operations. However, the limited stage size of the Director constricted the building of a deep splay tree. If a fixed tree can be used in the animation, it can be designed to reach the maximum depth of the splay tree which would be shown at the stage. So, a sample tree was developed as the original tree to occur at the initial stage based on the considerations of effectively demonstrating different cases of the splaying rotations and meanwhile taking care of the stage limitation of the Director.

The original tree contained both internal nodes and the corresponding subtrees which were respectively represented with the circle and the triangle. The user could select each of the internal nodes occurred in the original tree to see the splaying steps of this node through the visualized animation. To select the accessed node, the user need type the key of the node into the input box which was designed at the left corner of the stage and then click "SPLAY" button which was placed next to the input box. A subtree can't be selected.

The color of the stage was designed as blue, while the original color of the nodes was made yellow. When a node to be accessed was selected, its color was changed from the original yellow to the red color to obtain more attention from the users and hence promote the ideal learning effects. The red and yellow colors were chosen because of their sharp comparison effect. During the step-by-step node movements, the two different colors effectively and vividly reflected the data structure changes occurred in the tree.

Based on the algorithms of the splay tree operations, the accessed node was promoted to the root step by step through a sequence of rotations. To optimize the visualization design, smooth and continuous node movements were emphasized during the whole animation period. The whole splaying animation was synchronized by the sound which explained various operations of the splaying and described what happened during the animation. After the accessed node was splayed to the root of the tree, the sound was again used to instruct the students to click "Continue" button to select next node to be splayed. In this way, the students could select different nodes which represent different cases and study the corresponding splaying steps through the visualized animation. The number of the splaying steps for each accessed node was dynamically calculated and displayed at the stage during the whole animation. After finish studying this part, the students could choose the "Test" button to go to the test and its corresponding answers.

The software utilized a lot of director functions and lingo scripts. Concerning the Director, totally 194 cast members, 119 frames, and 120 channels were used in the software. For the lingo scripting language, both cast script and frame script were used in the software.

The software used several elements which were described by Macromedia (1997) as follows:

a. Stage: It is one of the most common elements of Director's user interface. It is the place where the movie takes place.

b. Cast Window: It is used to store all the media elements or cast members that will be used in the Director movie. It is the off-stage area where the cast members stay to wait for the use by the movie. Any element utilized by the movie must be in the cast.

c. Score Window: It is used to instruct the cast members when to enter, how to act, what to say, and when to exit the stage. The score consists of the frames and channels that intersect to form the shape of cells.

d. Channel: Each row in the score is a channel. There are 120 numbered channels in the score window. Each cell in a numbered channel contains a sprite. A sprite is a representation of a cast member that is placed on the stage.

e. Frame: Each column in the score is a single frame. Each frame contains the sprites which are located in the different numbered channel. Director animates the sprites over a series of frames. The frame controls the sequential appearance of the sprites based on the order of the numbered frame.

f. Paint Window: It is the place to create new graphics or edit the things imported into the movie. The paint window has a text window where text can be entered as much as you do with a word processor.

g. Lingo Script: Lingo is the English-based scripting language built for Director. It is used to write scripts or instructions to control where the playback head goes, allow users to do what they want to do, and create animations that would be difficult to implement using the score alone. So, lingo gives more functionality to Director.

Generally, four steps were undertaken for the design of the animation. At the first step, the sprites needed by the stage at different moments were identified; At the second step, the cast members were designed and created based on the sprites needed; At the third step, the score window was designed to locate the cast members to their corresponding frames; At the fourth step, various lingo scripts were developed to control the sprites' behaviors at the stage to accomplish the specific animation.

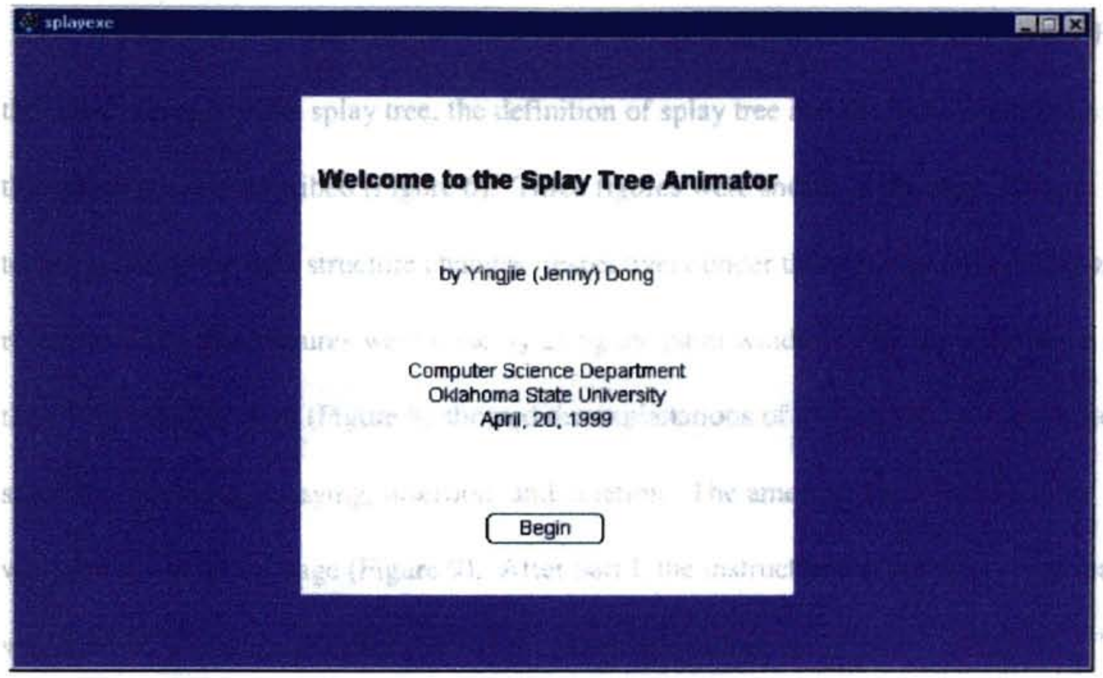
Implementation Details

The software used many Director functions and lingo scripts. Totally, 194 cast members, 119 frames, and 120 channels were used during the whole animation. The implementation details of the stage, cast members, frame and frame script utilized in the software would be discussed in this section.

Stage

The stage was the place where the animation took place. It was open from the start of the movie. The stage used in this software was designed as blue color. The maximum stage size was set with 752 width and 440 height.

The first stage shown in the software was the "Welcome" window (Figure 4). The stage was accompanied by the sound which instructed the user to click "Begin" button to tart the animator. Then the next stage (Figure 5) descried the objectives of the software and the three parts included in the software.

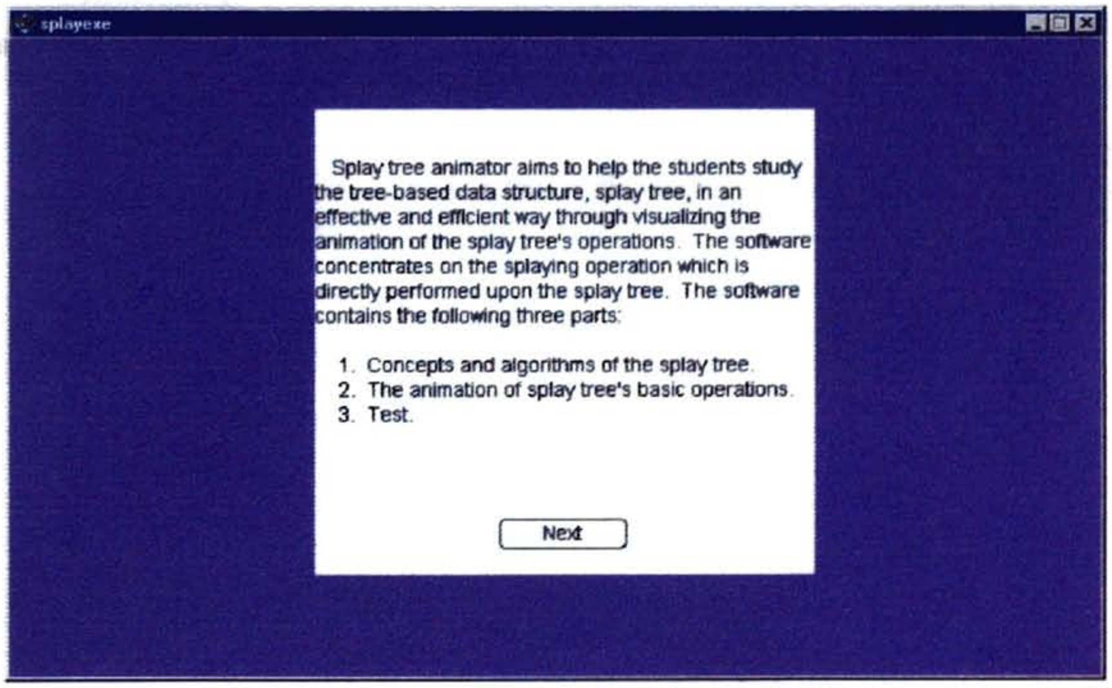


Welcome to the Splay Tree Animator

by Yingjie (Jenny) Dong

Computer Science Department
Oklahoma State University
April, 20, 1999

Figure 4: "Welcome" window

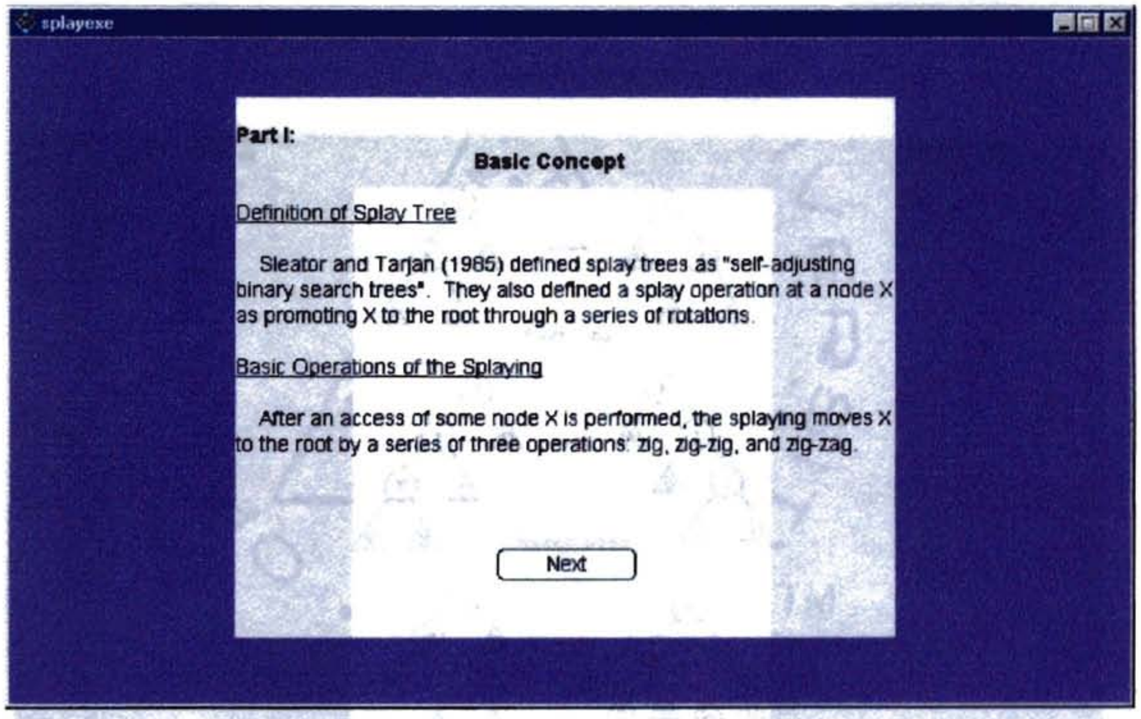


Splay tree animator aims to help the students study the tree-based data structure, splay tree, in an effective and efficient way through visualizing the animation of the splay tree's operations. The software concentrates on the splaying operation which is directly performed upon the splay tree. The software contains the following three parts:

1. Concepts and algorithms of the splay tree.
2. The animation of splay tree's basic operations.
3. Test.

Figure 5: Software component

Part I of the software discussed the concepts and algorithms of the splay tree. For the basic concept of the splay tree, the definition of splay tree and the basic operations of the splaying were described (Figure 6). Three figures were shown at the stage (Figure 7) to demonstrate the data structure changes respectively under the zip, zig-zig, and zig-zag rotations. The three figures were done by using the paint window. For the algorithms of the splay tree, the stage (Figure 8) showed the explanations of various operations of the splay tree including splaying, insertion, and deletion. The amortized analysis formula was also given at the stage (Figure 9). After part I, the instruction on how to use animator was described (Figure 10). For each of the above-mentioned stages, there was a "next" button placed at the bottom of the stage to forward the stage one by one. After the "Next" button on the instruction stage was clicked, the software went to its second part, the animation part.



Part I:

Basic Concept

Definition of Splay Tree

Sleator and Tarjan (1985) defined splay trees as "self-adjusting binary search trees". They also defined a splay operation at a node X as promoting X to the root through a series of rotations.

Basic Operations of the Splaying

After an access of some node X is performed, the splaying moves X to the root by a series of three operations: zig, zig-zig, and zig-zag.

Figure 6: Basic concept of splay tree

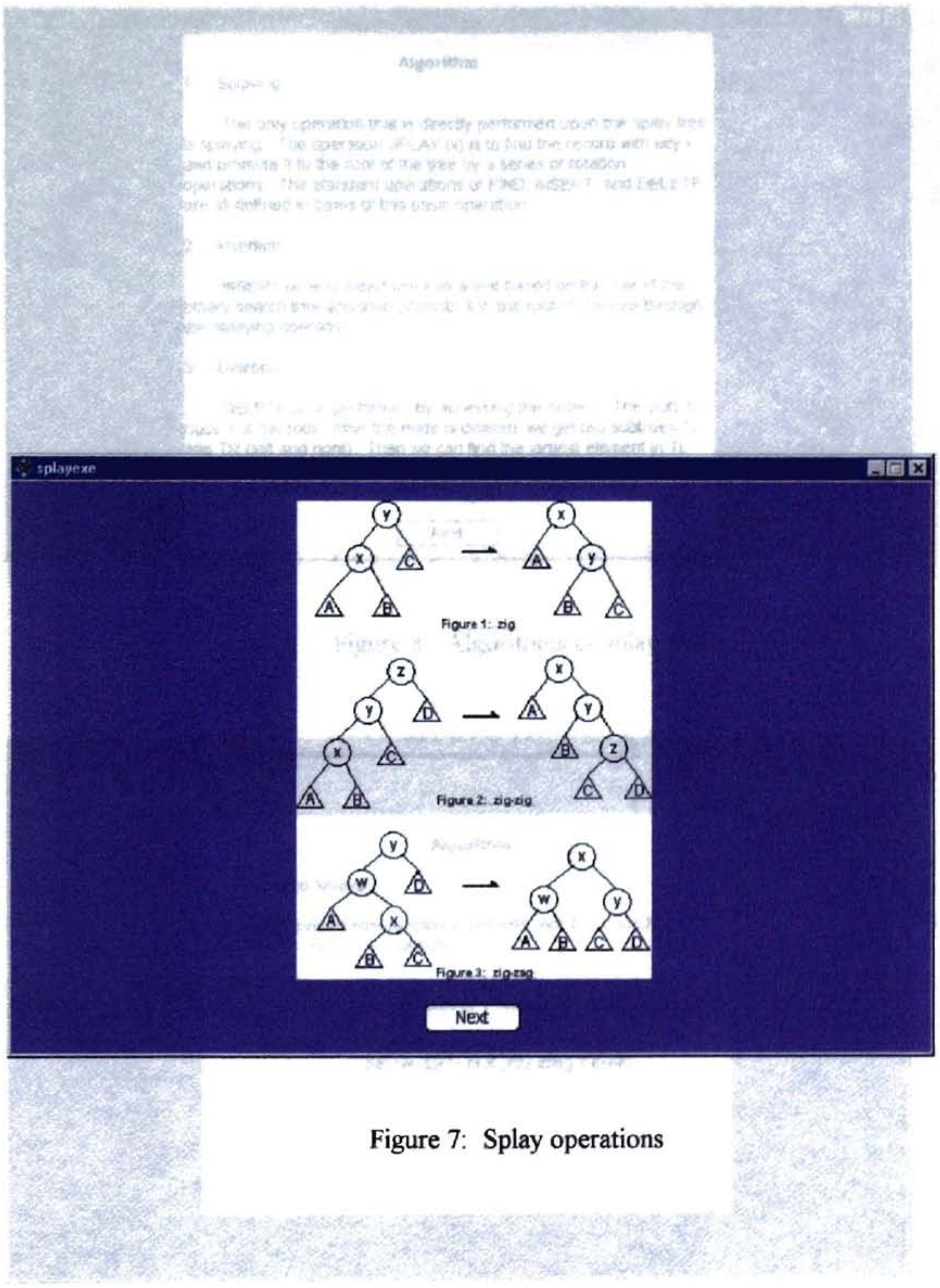
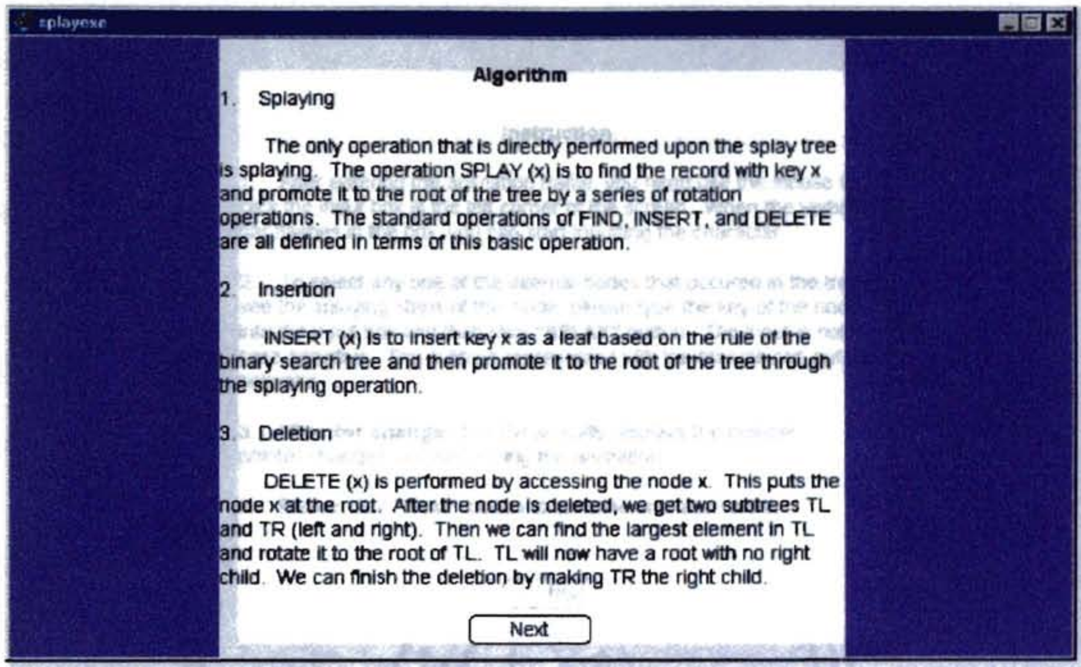


Figure 7: Splay operations



Algorithm

1. Splaying

The only operation that is directly performed upon the splay tree is splaying. The operation SPLAY (x) is to find the record with key x and promote it to the root of the tree by a series of rotation operations. The standard operations of FIND, INSERT, and DELETE are all defined in terms of this basic operation.

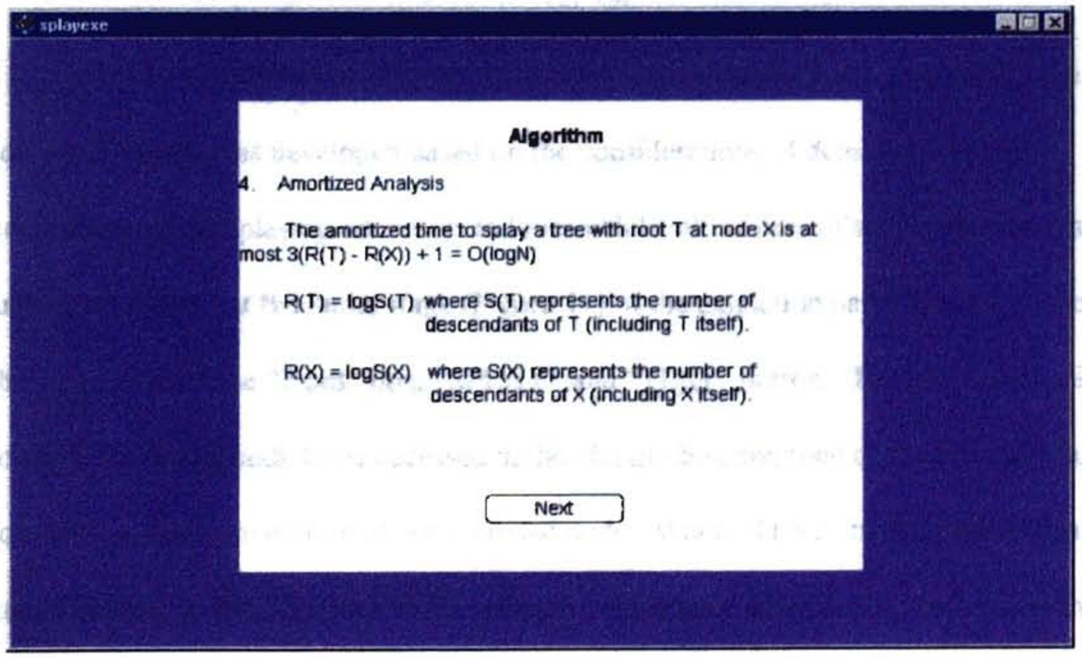
2. Insertion

INSERT (x) is to insert key x as a leaf based on the rule of the binary search tree and then promote it to the root of the tree through the splaying operation.

3. Deletion

DELETE (x) is performed by accessing the node x. This puts the node x at the root. After the node is deleted, we get two subtrees TL and TR (left and right). Then we can find the largest element in TL and rotate it to the root of TL. TL will now have a root with no right child. We can finish the deletion by making TR the right child.

Figure 8: Algorithms of splay tree



Algorithm

4. Amortized Analysis

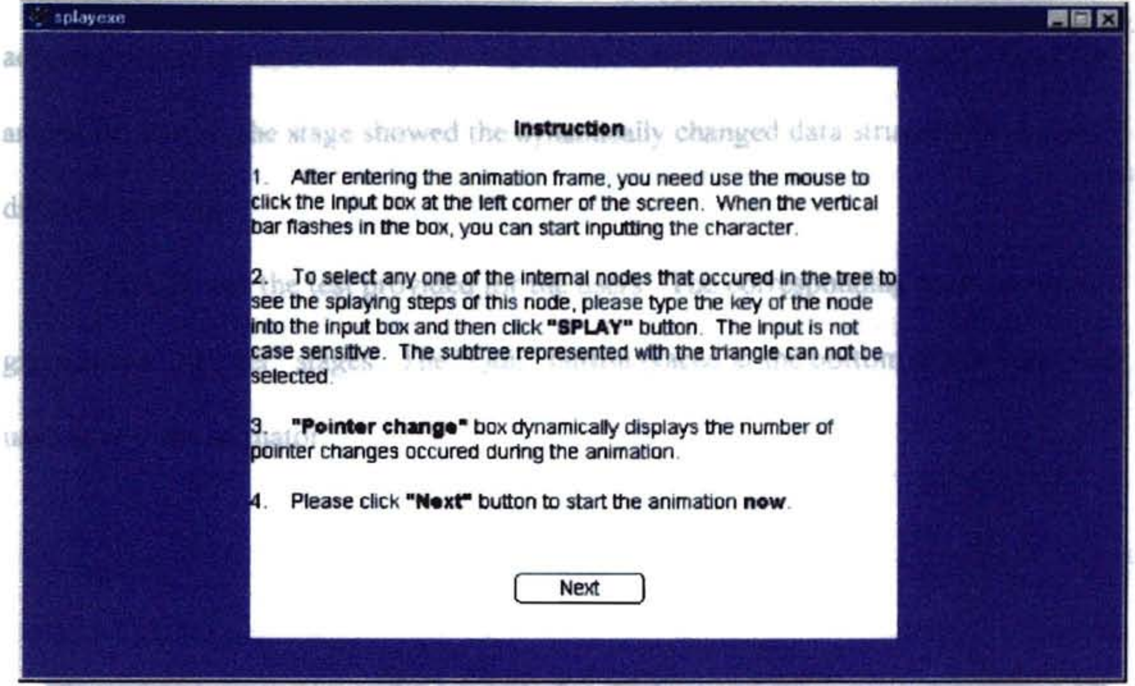
The amortized time to splay a tree with root T at node X is at most 3(R(T) - R(X)) + 1 = O(logN)

R(T) = logS(T) where S(T) represents the number of descendants of T (including T itself).

R(X) = logS(X) where S(X) represents the number of descendants of X (including X itself).

Figure 9: Algorithm continued

The original color of the nodes was designed as yellow, while the color of the node to be



Instruction

1. After entering the animation frame, you need use the mouse to click the input box at the left corner of the screen. When the vertical bar flashes in the box, you can start inputting the character.

2. To select any one of the internal nodes that occured in the tree to see the splaying steps of this node, please type the key of the node into the input box and then click **"SPLAY"** button. The input is not case sensitive. The subtree represented with the triangle can not be selected.

3. **"Pointer change"** box dynamically displays the number of pointer changes occured during the animation.

4. Please click **"Next"** button to start the animation **now**.

Figure 10: Instruction window

Part II, the animation of splay tree's splaying operation, is the core part of the software. A tree was developed based on the considerations of demonstrating all possibilities of the splaying operation and meanwhile taking care of the limited stage size and was displayed at the initial stage (Figure 11) of the animation part. The left corner of the stage showed the "Input" box, "SPLAY" and "TEST" button. The users were asked to enter the key of node to be accessed in the "Input" box and then click "SPLAY" button to see the splaying operation of the accessed node. When "TEST" button was clicked, the animator went to the third part of the software which gave several questions to test the users' learning results. To show a tree of sufficient depth at the stage, the node of the tree was designed to its minimum size that is large enough to encompass the value of the node.

The original color of the nodes was designed as yellow, while the color of the node to be accessed was made as red. The key value of each node was written in black. Once the animation started, the stage showed the dynamically changed data structures at each different moment.

Part III was the test provided for the users. The corresponding answers were given at the "Answer" stages. The "Quit" button placed at the bottom of the stage was used to exit the animator.

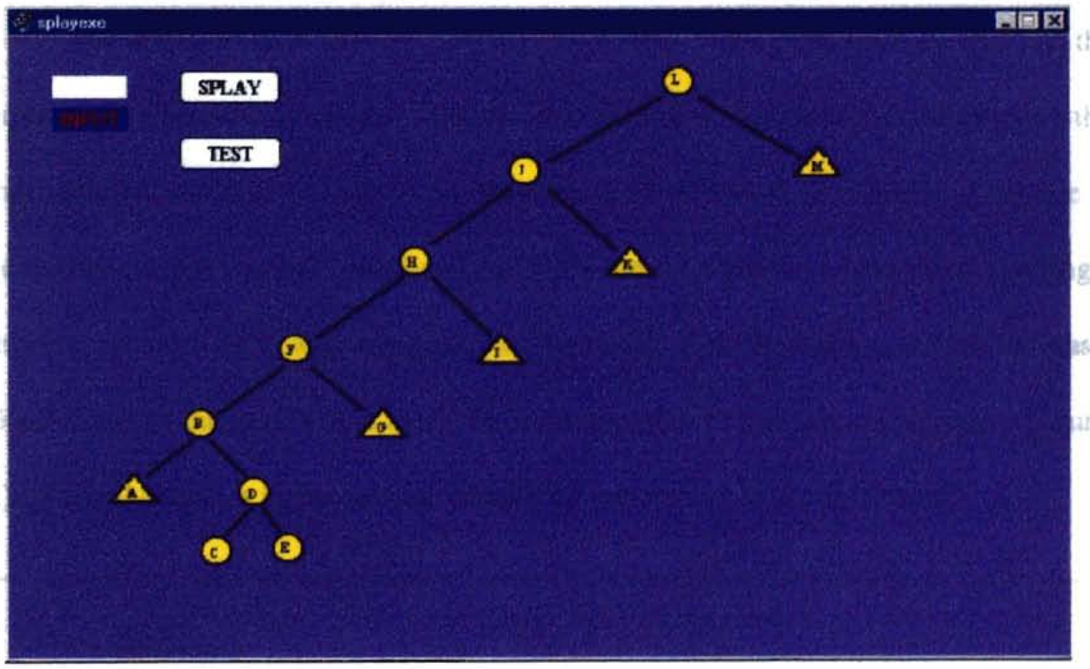


Figure 11: Splay tree

<u>Cast Members</u>

The cast members referred to any element utilized by the movie. They were located in the cast window of the Director. There could be various types of cast members including graphic, text, sound, digital video, palette, lingo, and behavior inspector cast members. The thumbnail in the bottom-right corner of the cast member indicated the type of the cast member.

This software used 194 cast members. Various types of cast members were covered in the software including text, sound, paint, lingo, behavior inspector, button, and shape cast members.

The descriptions of the concepts and algorithms of the splay tree, the test, and the corresponding answers belonged to the text cast members. The sound accompanying the whole animation added the sound cast members to the software. The nodes of the tree and the figures given at both part I and part III of the software were developed by using the paint window. They were the paint cast members. The internal score behavior was asked to control the loop within a certain frame by using its corresponding script (Figure 12). Several behavior inspector cast members were created in the software.

```
On exitFrame
        Go to the frame
End
```

Figure 12: Score script

In the software, there were several "Next" buttons to forward the stage one by one. There were also the "SPLAY" button used to start the splaying animation, the "TEST" button to change to the "test" frame, the "Answer" button to go to the "answer" frame, and "Quit" button to exit the animator. Those buttons formed the type of button cast members. Attached to each button, there was the corresponding script to help accomplish the frame change. The selected scripts attached to certain buttons were given as follows:

```
On mouseUp
        Go to the frame + 1
End
```

Figure 13: "Next" button script

```
On mouseUp
        Go to the frame – 1
End
```

Figure 14: "Back" button script

```
on mouseUp
        set Token to field "Word"
        put "" into field "Word"

        if Token="E" then
                go to "Esplay"
        end if
        if Token="C" then
                go to "Csplay"
        end if

        if Token="D" then
                go to "Dsplay"
         end if

         if Token="B" then
                go to "Bsplay"
        end if

        if Token="F" then
                go to "Fsplay"
        end if

        if Token="H" then
                go to "Hsplay"
        end if

        if Token="J" then
                go to "Jsplay"
        end if

        if Token="L" then
                go to "Lsplay"
        end if
end
```

Figure 15: "SPLAY" button script

```
On mouseUp
        Go to "Test"
End
```

Figure 16: "Test" button script

```
On mouseUp
        Go to "Answer"
End
```

Figure 17: "Answer" button script

```
On mouseUp
        puppetSound "thank"
        updateStage
        startTimer
        repeat while the timer < 4 * 60
                nothing
        end repeat
        quit
End
```

Figure 18: "Quit" button script

Each edge between the two nodes belonged to the shape cast member. Several shape cast members were used in the software.

Several lingo scripts were used to implement the animation of the splay tree algorithm and control where the playback head went. The selected scripts written for the software would be given in Appendix.

Frame and Frame Script

Each column in the score has a single frame. The sprites were displayed at the stage frame by frame. So, the frame controlled the time sequence of the objects displayed in the movie. The frame script was used to implement what would be done at that frame.

There were 119 frames used in the software. Basically, the sprites were placed in the different frames based on the order in which the sprites occurred at the stage. The first frame showed the "Welcome" window. Once the "Begin" button was clicked, the playback head went to the second frame. Then the objects at the second frame occurred at the stage. The contents of part I of the software were placed in the corresponding frame based on their appearance order. Then the playback head went to the animation frame. The displaying of the original tree was placed from frame 25 to 30. The attached frame scripts were shown in Figure 19 and Figure 20. Frames 35 to 40 encompassed the animation of node E's splaying operation with the marker "Esplay" set above frame 35. In the same way, frames 45 to 50, frames 55 to 60, frames 65 to 70, frames 75 to 80, frames 85 to 90, frames 95 to 100, and frames 105 to 110 respectively recorded the splaying animation of the accessed node C, D, B, F, H, J, and L with the separate "Csplay", "Dsplay", "Bsplay", "Fsplay", "Hsplay", "Jsplay", and "Lsplay markers set above at each corresponding starting frame. Each node's splaying animation was accomplished by using different frame scripts. The selected frame scripts would be described at the Appendix. The contents of the test were placed at frame 112, while the corresponding answers were given at frame 117, 118, and 119. The frame scripts were also used to keep the playback head to loop within the certain frame.

```
on exitFrame
        repeat with i = 103 to 115
                puppetSprite i, FALSE
        end repeat

        set the visible of sprite 118 to TRUE
        set the visible of sprite 117 to TRUE

        -- initialize all base nodes and edges' visible to FALSE
        repeat with i = 1 to 102
                puppetSprite i, TRUE
        end repeat
        repeat with i = 1 to 102
                set the visible of sprite i to FALSE
        end repeat
        -- set the edges in the original tree to be visible.
        set the visible of sprite 53 to TRUE
        set the visible of sprite 54 to TRUE
        set the visible of sprite 55 to TRUE
        set the visible of sprite 56 to TRUE
        set the visible of sprite 59 to TRUE
        set the visible of sprite 60 to TRUE
        set the visible of sprite 67 to TRUE
        set the visible of sprite 68 to TRUE
        set the visible of sprite 83 to TRUE
        set the visible of sprite 84 to TRUE
        set the visible of sprite 93 to TRUE
        set the visible of sprite 94 to TRUE
end
```

Figure 19: Frame 25's script

```
On exitFrame
        Go to the frame
End
```

Figure 20: Frame 30's script

The Characteristics of the Software Design

The software design done in the study owned the following characteristics:

a. The software was designed and developed by using the extensive combination of Director functions and lingo scripts. There were 194 cast members, 119 frames, and 120 channels used in the software. Several cast scripts and frame scripts were written to control the position of the playback head and implement the animation.

b. The software was interactive and user-friendly. Hence, it was easy to use.

c. The software was easy to be maintained. The codes developed in the software was easy to debug and hence easy to improve. For the animation of the data structure changes, one major function was developed to be used as the basic function for various cases of splaying operation. This function was called throughout the lingo scripts developed respectively for each accessed node.

d. Groups of node movements were designed and implemented by using the technique of setting certain nodes and their edges visible or invisible based on the time sequence. With this technique, the subtrees were moved to become another node's children as a group during the animation. Hence, the data structure changes were reflected more effectively and vividly in the animation.

There were 8 internal nodes designed in the tree which could be selected by the users to see the corresponding splaying operations. The key values of these nodes were respectively "L", "J", "H", "F", "B", "D", "C", and "E". The node "D" was chosen in this section as a sample node to demonstrate the animation of the splaying operation of the accessed node.

a. The value "D" was entered into the "Input" box by the user to see its splaying operation (Figure 21).

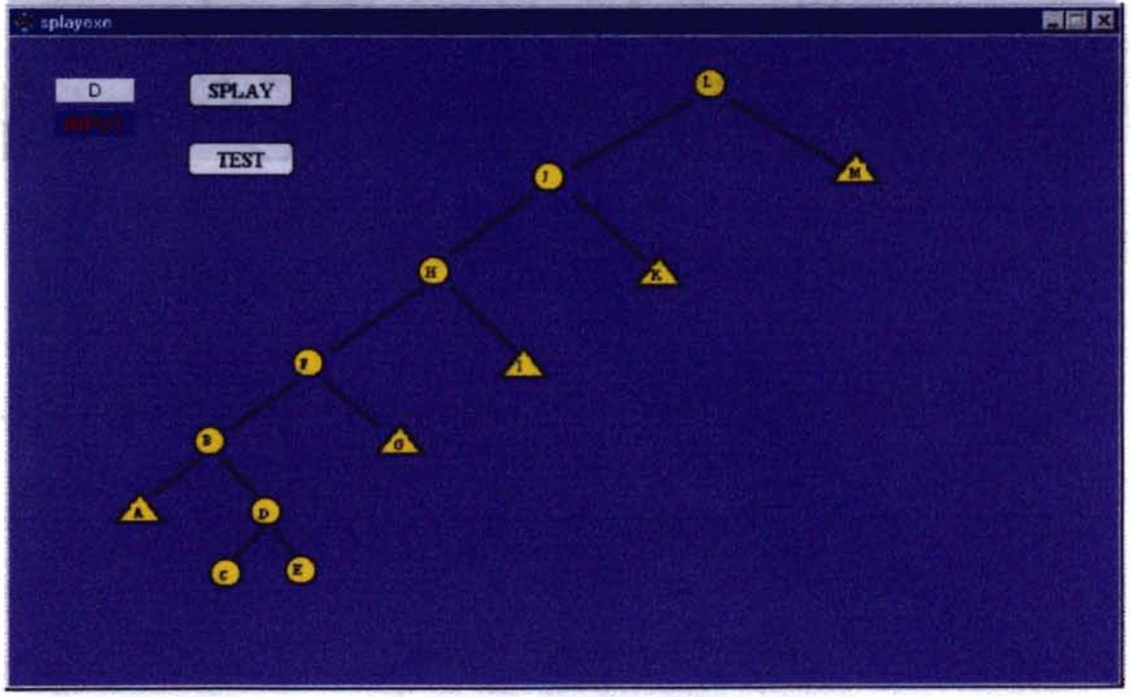


Figure 21: Selection of "D" node

b. The accessed node "D" was marked as red color (Figure 22).

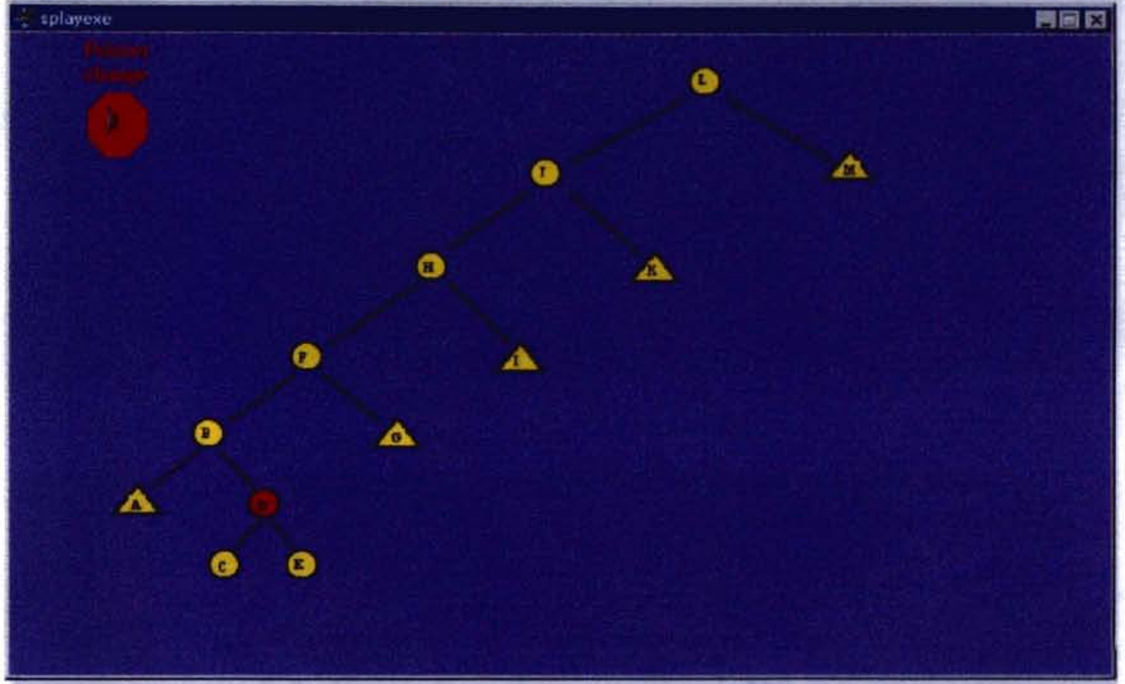


Figure 22: Red "D"

c. The zig-zag rotation was done among the accessed node "D", its parent "B", and its grandparent "F" (Figure 23).

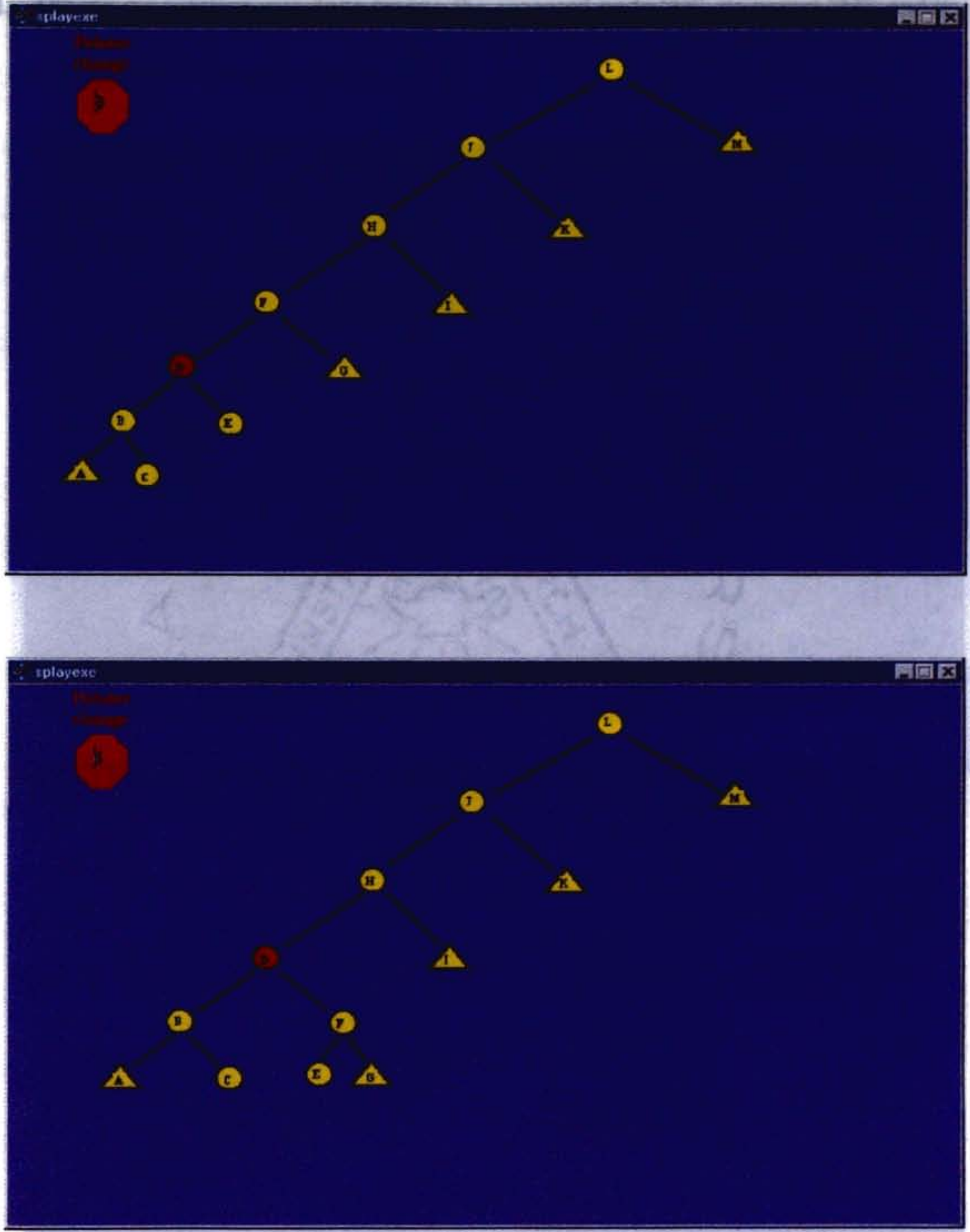


Figure 23: Zig-zag rotation

Figure 24: Zig-zig rotation

d. The zig-zig rotation was done among the accessed node "D", its parent "H", and its grandparent "J" (Figure 24).

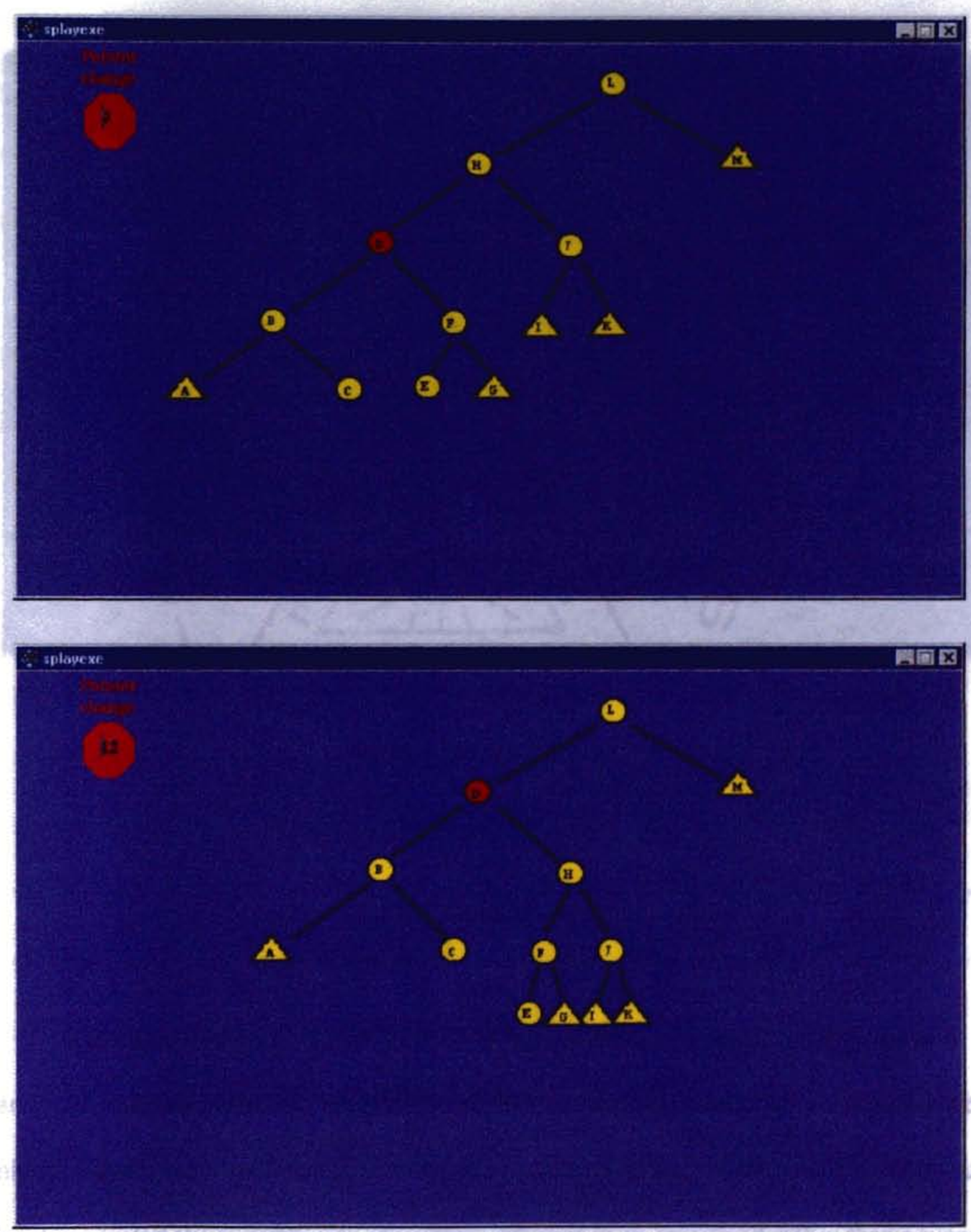


Figure 24: Zig-zig rotation

e.	The zig rotation was done between the accessed node "D" and its parent "L" which was the original root of the tree.

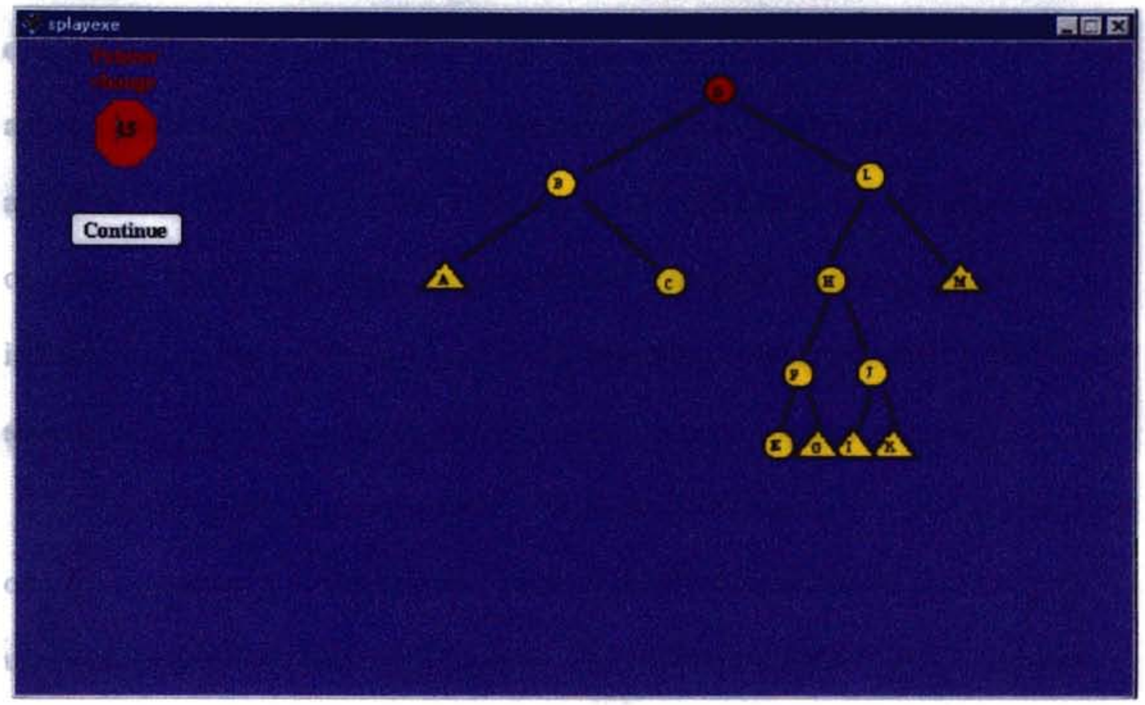


Figure 25: Zig rotation

Amortized Analysis

The amortized running time analysis is the technique in complexity analysis of a variety of data structures. Tarjan (1985) put forward that the amortized running time not only provided a more exact method to measure the running time of known algorithms but also it suggested the possible new algorithms efficient in an amortized rather than a worst-case sense.

Tarjan (1985) defined "amortize" as "to average the running times of operations in a sequence over the sequence" (p. 306). The author asserted that the amortized analysis which averages the running time per operation over a (worst-case) sequence of operations could yield a both realistic and robust answer comparing with the worst-case analysis and the average-case analysis. Moret and Shapiro (1991) also stated the amortized complexity as a technique for dealing with the complexity of sequences of operations. "While both worst- and average-case complexity measures apply to individual operations, amortized complexity applies only to sequences of operations and says nothing about the actual running time of any particular operation" (p. 94).

Tarjan (1985) used two ways to analyze the amortized running time of operations on a data structure. The first was the "banker's view" of amortization. The computer user was regarded as a customer of a bank. The credits were defined as the amortized time of the operation. A certain number of credits were allocated to each operation. These credits were then deposited into or withdrawn from the account based on the operation sequence. The final account balance was then used to obtain the upper and lower bounds of performance of that data structure. The second view of amortization was that of physicist. A potential function Φ was defined to map any configuration D of the data structure into a real number $\Phi(D)$ called the potential of D.

Having the idea of explicitly seeking only amortized efficiency instead of worst-case efficiency, Sleator suggested the possibility of simplifying dynamic trees. Based on this idea, Sleator and Tarjan came up with a new data structure, splay tree, which was a self-adjusting binary search tree that was as efficient as balanced trees but only in the amortized sense (Tarjan, 1987).

Sleator and Tarjan (1985) used the potential function to analyze the amortized complexity of splaying. The authors defined a potential function Φ to map each possible configuration of the data structure into a real number called the potential of the given configuration of the data structure. The amortized time a of an operation is:

$$a = t + \Phi' - \Phi$$

where t = the actual time of the operation

Φ = data structure configuration before the operation

Φ' = data structure configuration after the operation

With this definition, the total time of sequence of m operations can be obtained by

$$\sum_{i=1}^{m} t_i = \sum_{i=1}^{m} (a_i + \Phi_{i-1} - \Phi_i) = \sum_{i=1}^{m} a_i + \Phi_0 - \Phi_m$$

Hence, the total amortized time of sequence of m operations is:

$$\sum_{i=1}^{m} a_i = \sum_{i=1}^{m} t_i + \Phi_m - \Phi_0$$

where a_i = amortized time of operation i

t_i = actual time of operation I

Φ_0 =the initial data structure potential

Φ_m =the potential after operation I, for $i \geq 1$

The equation indicated that the total amortized time equals the total actual time of the m operations plus the net increase in potential from the initial to the final configuration. The initial potential is zero. If the final potential is no less than the initial potential, then the total amortized time serves as an upper bound of the total actual time.

For the splay tree, a potential function Φ over all nodes i in the tree T is defined

as: $\Phi(T) = \sum_{i \in T} \log S(i)$

Where S(i) represents the number of descendants of i including i itself

R(i) = logS(i), R(i) represents the rank of node i.

Hence, $\Phi(T) = \sum_{i \in T} R(i)$

Sleator and Tarjan (1985) used the number of rotations done as the measure of the running time of a splaying operation. If there are no rotations, one is counted for the splaying. Based on the analysis of the three basic cases of the splaying operation, Sleator and Tarjan (1985) derived the following lemma:

The amortized time to splay a tree with root t at node x is at most 3(r(t) – r(x)) + 1 = O(log(S(t)/S(x))) (p. 658).

The lemma was proven through three cases of the splaying operation.

Let s = size before a splay

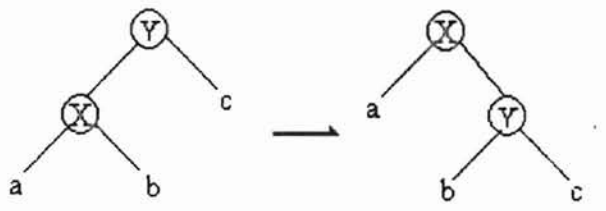
r = rank before a splay

s' = size after a splay

r' = rank after a splay

$a_m = t_m + \Phi(D_m) - \Phi(D_{m-1})$

case 1: (zig) one rotation



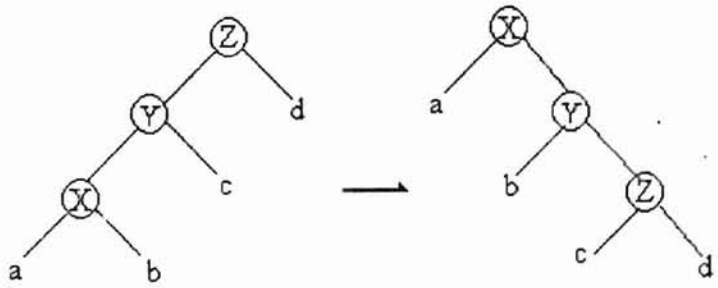
Amortized time of this step is:

$1 + [r'(x) + r'(y)] - [r(x) + r(y)]$

$\le 1 + r'(x) - r(x)$ since $r(y) \ge r'(y)$

$\le 1 + 3(r'(x) - r(x))$ since $r'(x) \ge r(x)$

Case 2: (zig-zig) Two rotations



Amortized time of this step is:

$2 + [\,r'(x) + r'(y) + r'(z)] - [\,r(x) + r(y) + r(z)]$

$= 2 + r'(y) + r'(z) - r(x) - r(y)$ since $r'(x) = r'(z)$

$\leqslant 2 + r'(x) + r'(z) - 2r(x)$ since $r'(x) \geqslant r'(y)$ and $r(y) \geqslant r(x)$

$\leqslant 3(\,r'(x) - r(x)\,)$ This is the claim.

Now we need to prove the claim $2 + r'(x) + r'(z) - 2r(x) \leqslant 3(\,r'(x) - r(x)\,)$.

Simplifying the above equation, we get: $r(x) + r'(z) - 2r'(x) \leqslant -2$ (need to prove)

From the zig-zig figure, we can see

$$s(x) + s'(z) < s'(x)$$

then,

$$\frac{s(x)}{s'(x)} + \frac{s'(z)}{s'(x)} < 1$$

Based on the lemma (weiss, 1997):

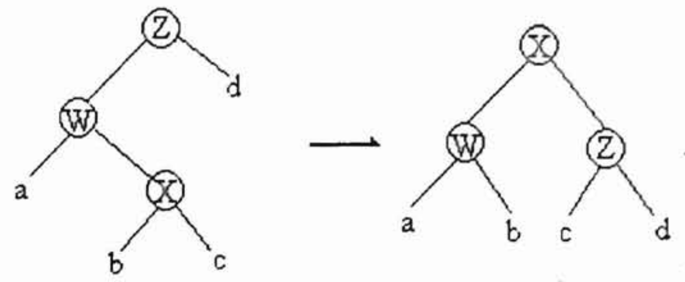
If $a + b \leqslant c$ ($a > 0, b > 0$), then $\log(a) + \log(b) \leqslant 2\log(c) - 2$

We obtained

$$\log\frac{s(x)}{s'(x)} + \log\frac{s'(z)}{s'(x)} \leq 2\log 1 - 2$$
$$\log s(x) - \log s'(x) + \log s'(z) - \log s'(x) \leq -2$$
$$r(x) - r'(x) + r'(z) - r'(x) \leq -2$$
$$r(x) + r'(z) - 2r'(x) \leq -2$$

Hence, the claim is true.

Case 3: (zig-zag) two rotations



Amortized time of this step is:

$$2 + [\ r'(x) + r'(w) + r'(z)] - [\ r(x) + r(w) + r(z)]$$

$\leqslant 2 + r'(w) + r'(z) - 2\ r(x)$ since $r'(x) = r(z)$ and $r(x) \leqslant r(w)$

$\leqslant 2\ (\ r'(x) - r(x)\ )$

$< 3\ (\ r'(x) - r(x)\ )$

CHAPTER IV

SUMMARY AND FUTURE WORK

Summary

The educational software in this study was developed under Windows 95 operating system with the authoring tool of multimedia production Director 6.0 and its scripting language Lingo.

The software developed in this study worked as an individual teaching module to help the students study one of the tree-based data structures, splay tree, in an effective and efficient way through visualizing the animation of the splay tree's basic operations. The software contained the following three parts:

Part I: Concepts and algorithms of the splay tree.

Part II: The animation of the splaying operation.

Part III: Test.

Part I described the definition of splay tree, various cases of the splaying, and the algorithms of the splay tree including the splaying, insertion, deletion, and the amortized analysis formula. Part II is the core part of the software, which demonstrated the splaying operation through the graphical representations and displayed each splaying step of each accessed node during the animation. Smooth and continuous node movements, color, and the sound were emphasized in the whole animation period to pursue the ideal teaching result. Part III was the test to help the users evaluate their mastery of the class. The corresponding answers were also given in the software.

Finally, the study gave a brief description on the concept of the amortized analysis and the amortized complexity of the splaying which was analyzed by using a potential function.

Future Work

The splay tree animator used a fixed tree to demonstrate the algorithms of the splay tree in order to show all the possibilities of the splaying operation and meanwhile meet the requirement of the limited stage size of the Director. A more effective way could be developed in future to overcome the difficulty of stage limitation.

The software developed in the study could be put on the Internet with Shockwave plug-in to act as the long-distance learning tool.

There are some other important data structures in computer science. The animation of those data structures' algorithms could be done in future.

SELECTED REFERENCES

Aho, A. V. , Hopcroft, J. E., & Ullman, J. D. (1983). *Data Structures and Algorithms.* Reading, Massachusetts: Addison-Wesley Publishing Company.

Arra, S. K. (1992). *Object-Oriented Data Structure Animation.* Unpublished master's thesis, Oklahoma State University at Stillwater.

Baron, R. J. & Shapiro, L. G. (1980). *Data Structures and Their Implementation.* New York: Van Nostrand Reinhold Company.

Bennett, J. (1997). *Director 6 and Lingo Interactive.* Berkeley, CA: Macromedia Press.

Booth, K. (1975). *PQ-Trees.* 16mm color silent film, 12 minutes.

Brown, M. H. (1988). *Algorithm Animation.* MA: The MIT Press.

Carr, C. (1992). "Performance Support System: A New Horizon for Expert Systems". *AI Expert,* 7(5), 44-49.

Copeland, P. (1989). Interactive Video. In M. Eraut, *The International Encyclopedia of Educational Technology.* Oxford: Pergamon Press.

Feldman T. (1994). *Multimedia.* London: British Library Board.

Fisher, S. (1995). *Macromedia Director: Your Personal Consultant.* Emeryville, CA: Ziff-Davis Press.

Galbreath, J. (1997). "The Internet: Past, Present, and Future". *Educational Technology, 37*(6), 39-45.

http://www.cs.hope.edu/~alganim/ccaa/algo.html

Knowlton, K. C. (1966). *L6: Bell Telephone Laboratories Low-Level Linked List Language.* Two black and white sound films. Murray Hill, NJ: Bell Telephone Laboratories.

Lee, W. (1988). *An Implementation of a Data Structures Display System.* Unpublished master's thesis, Oklahoma State University at Stillwater.

Moret, B.M.E. & Shapiro, H.D. (1991). *Algorithms from P to NP.* CA: The Benjamin/Cummings Publishing Company, Inc.

Ramaiyer, K. (1998). Web site: Http://www.cs.jhu.edu/~kumar.

Rathbone, A. (1995). *Multimedia & CD-ROMs for Dummies.* Foster City, CA: IDG Books Worldwide, Inc.

Reingold, E. M. & Hansen, W. J. (1983). *Data Structures.* Boston: Little, Brown and Company.

Rosenborg, V., Green. B., Hester, J., Knowles, W., & Wirsching, M. (1993). *A Guide to Multimedia.* Indiana: New Riders Publishing.

Salomon, G. (1989). Computers in the Curriculum. In M. Eraut, *The International Encyclopedia of Educational Technology.* Oxford: Pergamon Press.

Shen, B. (1998). *Instructional Module for Teaching about Binary Search Trees.* Unpublished master's thesis, Oklahoma State University at Stillwater.

Shen, H. (1994). *A Visual Aid for the Learning of Tree-Based Data Structure.* Unpublished master's thesis, Oklahoma State University at Stillwater.

Shimomura, T. & Isoda, S. (1991). "Linked-List Visualization for Debugging". *IEEE Software, 17*, 44-51.

Sleator, D.D. & Tarjan, R.E. (1985). "Self-Adjusting Binary Search Trees." *Journal of the Association for Computing Machinery, 32*(3), 652-686.

Starr, R. M. & Milheim, W. D. (1996). "Educational Uses of the Internet: An Exploratory Survey". *Educational Technology, 36*(5), 19-22.

Stasko, J. T. (1990). "Tango: A Framework and System for Algorithm Animation". *IEEE Computer, 23*(2), 27-39.

Stasko, J. T. (1997). http://www.cc.gatech.edu/~cilla/cs7100/project01/proposal.html

Stubbs, D. F. & Webre, N. W. (1987). *Data Structures with Abstract Data Types and Modula-2*. Pacific Grove, California: Brooks/Cole Publishing Company.

Tarjan, R. E. (1985). "Amortized Computational Complexity". *SIAM Journal on Algebraic and Discrete Methods, 6*(2), 306 – 318.

Tarjan, R. E. (1987). "Algorithm Design". *Communications of the ACM, 30*(3), 205 – 212.

Tucker, B. (1997). *Handbook of Technology-Based Training*. Vermont: Gower Publishing Limited.

Vikas, M. (1996). *Visualization of Sorting Algorithms*. Unpublished master's thesis, Oklahoma State University at Stillwater.

Villamil, J. & Molina, L. (1997). *Multimedia: An Introduction*. Indianapolis: Que Education and Training, Macmillan Computer Publishing.

Weiss, M. A. (1997). *Data Structures and Algorithm Analysis in C*. Menlo Park, California: Addison-Wesley Longman, Inc.

Xu, C. (1997). *Multimedia Visualization of Abstract Data Type*. Unpublished master's thesis, Oklahoma State University at Stillwater.

APPENDIX

Selected Lingo Scripts of the Software

Main Script

```
on startMovie
  put "" into field "Word"
end startMovie

on movenode number, x1, x2, y1, y2
  set stepx to float((x2-x1)/100.0)
  set stepy to float((y2-y1)/100.0)
  repeat with i=1 to 100
    set x1 to x1 + stepx
    set the locH of sprite number to x1
    set y1 to y1 + stepy
    set the locV of sprite number to y1
    updateStage
  end repeat
  startTimer
  repeat while the timer < 30
    nothing
  end repeat
end
```

Script of Splaying E

```
on exitFrame
  --puppetSprite 111, TRUE
  repeat with i = 103 to 115
    puppetSprite i, TRUE
  end repeat
  set the castNum of sprite 111 to the number of cast "Red E"
  updateStage

  puppetSound "begin"
  updateStage
  startTimer
  repeat while the timer < 6*60
    nothing
  end repeat
  puppetSound "zigzig"
  updateStage

  startTimer
  repeat while the timer < 10*60
```

```
 nothing
end repeat

movenode (108, the locH of sprite 108, the locH of sprite 17,¬
          the locV of sprite 108, the locV of sprite 17)
--move A leftdown
movenode (107, the locH of sprite 107, the locH of sprite 18,¬
          the locV of sprite 107, the locV of sprite 18)
--move B leftdown
set the visible of sprite 91 to TRUE
updateStage
startTimer
repeat while the timer < 2*60
 nothing
end repeat

movenode (110, the locH of sprite 110, the locH of sprite 26,¬
          the locV of sprite 110, the locV of sprite 26)
--move C left and become B's right child
set the visible of sprite 93 to False
set the visible of sprite 92 to True
updateStage
startTimer
repeat while the timer < 2*60
 nothing
end repeat

movenode (109, the locH of sprite 109, the locH of sprite 27,¬
          the locV of sprite 109, the locV of sprite 27)
--move D leftup
movenode (111, the locH of sprite 111, the locH of sprite 29,¬
          the locV of sprite 111, the locV of sprite 29)
--move E leftup and single rotation finished
set the visible of sprite 94 to FALSE
updateStage
startTimer
repeat while the timer < 2*60
 nothing
end repeat

movenode (108, the locH of sprite 108, the locH of sprite 1,¬
          the locV of sprite 108, the locV of sprite 1)
--move A leftdown
movenode (107, the locH of sprite 107, the locH of sprite 17,¬
          the locV of sprite 107, the locV of sprite 17)
```

```
--move B leftdown

movenode (110, the locH of sprite 110, the locH of sprite 2,¬
        the locV of sprite 110, the locV of sprite 2)
--move C to B's right child
set the visible of sprite 99 to true
set the visible of sprite 100 to true
set the visible of sprite 92 to FALSE
updateStage
startTimer
repeat while the timer < 2*60
  nothing
end repeat

movenode (109, the locH of sprite 109, the locH of sprite 18,¬
        the locV of sprite 109, the locV of sprite 18)
--move D leftdown

movenode (111, the locH of sprite 111, the locH of sprite 27,¬
        the locV of sprite 111, the locV of sprite 27)
--move E leftup and zig-zig rotation among E,D,B finished
set the visible of sprite 84 to FALSE
updateStage

puppetSound "zigzig"
updateStage

startTimer
repeat while the timer < 10*60
  nothing
end repeat

--Next zig-zig rotation among E,F, and H will be done
movenode (114, the locH of sprite 114, the locH of sprite 41,¬
        the locV of sprite 114, the locV of sprite 41)
--move I rightdown
movenode (105, the locH of sprite 105, the locH of sprite 46,¬
        the locV of sprite 105, the locV of sprite 46)
--move H rightdown
set the visible of sprite 70 to TRUE
updateStage
startTimer
repeat while the timer < 2*60
  nothing
end repeat
```

```
movenode (115, the locH of sprite 115, the locH of sprite 48,¬
         the locV of sprite 115, the locV of sprite 48)
--move G right and become H's left child
set the visible of sprite 69 to TRUE
set the visible of sprite 68 to FALSE
updateStage
startTimer
repeat while the timer < 2*60
  nothing
end repeat
movenode (106, the locH of sprite 106, the locH of sprite 35,¬
         the locV of sprite 106, the locV of sprite 35)
--move F rightup and finished single rotation of F and H
movenode(111, the locH of sprite 111, the locH of sprite 19,¬
         the locV of sprite 111, the locV of sprite 19)
--move E rightup
movenode(109, the locH of sprite 109, the locH of sprite 27,¬
         the locV of sprite 109, the locV of sprite 27)
--move D rightup
movenode(107, the locH of sprite 107, the locH of sprite 18,¬
         the locV of sprite 107, the locV of sprite 18)
--move B rightup
movenode(108, the locH of sprite 108, the locH of sprite 17,¬
         the locV of sprite 108, the locV of sprite 17)
--move A rightup
--set the visible of sprite 99 to FALSE
--updateStage
--startTimer
--repeat while the timer < 2*60
--  nothing
--end repeat
movenode(110, the locH of sprite 110, the locH of sprite 26,¬
         the locV of sprite 110, the locV of sprite 26)
--move C as B's right child.
--All E's children finished following E
set the visible of sprite 99 to FALSE
set the visible of sprite 92 to TRUE
set the visible of sprite 100 to FALSE
updateStage
startTimer
repeat while the timer < 2*60
  nothing
end repeat
```

```
movenode(114, the locH of sprite 114, the locH of sprite 42,¬
        the locV of sprite 114, the locV of sprite 42)
--move I rightdown
movenode(115, the locH of sprite 115, the locH of sprite 45,¬
        the locV of sprite 115, the locV of sprite 45)
--move G as H's left child

movenode(105, the locH of sprite 105, the locH of sprite 41,¬
        the locV of sprite 105, the locV of sprite 41)
--move H rightdown
set the visible of sprite 69 to FALSE
set the visible of sprite 89 to TRUE
set the visible of sprite 90 to TRUE
updateStage
startTimer
repeat while the timer < 2*60
 nothing
end repeat

movenode(106, the locH of sprite 106, the locH of sprite 46,¬
        the locV of sprite 106, the locV of sprite 46)
--move F rightdown
movenode(111, the locH of sprite 111, the locH of sprite 35,¬
        the locV of sprite 111, the locV of sprite 35)
--move E rightup to the point of zig-zig rotation
movenode(109, the locH of sprite 109, the locH of sprite 19,¬
        the locV of sprite 109, the locV of sprite 19)
--move D rightup
movenode(107, the locH of sprite 107, the locH of sprite 27,¬
        the locV of sprite 107, the locV of sprite 27)
--move B rightup
movenode(108, the locH of sprite 108, the locH of sprite 18,¬
        the locV of sprite 108, the locV of sprite 18)
--A follows B as B's left child

movenode(110, the locH of sprite 110, the locH of sprite 29,¬
        the locV of sprite 110, the locV of sprite 29)
set the visible of sprite 91 to FALSE
set the visible of sprite 92 to FALSE
set the visible of sprite 84 to TRUE
updateStage

--C follows B as B's right child
--E's children finished following E's movement
--zig-zig rotation among E,F,and H is finished
```

```
puppetSound "zigzig"
updateStage

startTimer
repeat while the timer < 10*60
  nothing
end repeat

--Next zig-zig rotation among E,J, and L will be done
movenode(112, the locH of sprite 112, the locH of sprite 13,¬
          the locV of sprite 112, the locV of sprite 13)
--move M rightdown
movenode(103, the locH of sprite 103, the locH of sprite 10,¬
          the locV of sprite 103, the locV of sprite 10)
--move L rightdown
set the visible of sprite 58 to TRUE
updateStage
startTimer
repeat while the timer < 2*60
  nothing
end repeat

movenode(113, the locH of sprite 113, the locH of sprite 24,¬
          the locV of sprite 113, the locV of sprite 24)
--move K right to make it as L's left child
set the visible of sprite 56 to FALSE
set the visible of sprite 57 to TRUE
updateStage
startTimer
repeat while the timer < 2*60
  nothing
end repeat
movenode(104, the locH of sprite 104, the locH of sprite 16,¬
          the locV of sprite 104, the locV of sprite 16)
--move J rightup
movenode(111, the locH of sprite 111, the locH of sprite 20,¬
          the locV of sprite 111, the locV of sprite 20)
--move E rightup
movenode(106, the locH of sprite 106, the locH of sprite 22,¬
          the locV of sprite 106, the locV of sprite 22)
--move F to follow E
movenode(105, the locH of sprite 105, the locH of sprite 7,¬
          the locV of sprite 105, the locV of sprite 7)
--move H to follow E
```

```
movenode(114, the locH of sprite 114, the locH of sprite 12,¬
        the locV of sprite 114, the locV of sprite 12)
--move I to follow its parent H
movenode(115, the locH of sprite 115, the locH of sprite 38,¬
        the locV of sprite 115, the locV of sprite 38)
--move G to follow H
set the visible of sprite 56 to TRUE
set the visible of sprite 62 to TRUE
set the visible of sprite 73 to TRUE
set the visible of sprite 74 to TRUE
updateStage
set the visible of sprite 60 to FALSE
set the visible of sprite 70 to FALSE
set the visible of sprite 89 to FALSE
set the visible of sprite 90 to FALSE
updateStage
startTimer
repeat while the timer < 2*60
 nothing
end repeat

movenode(109, the locH of sprite 109, the locH of sprite 35,¬
        the locV of sprite 109, the locV of sprite 35)
--move D rightup to follow its parent E
movenode(107, the locH of sprite 107, the locH of sprite 19,¬
        the locV of sprite 107, the locV of sprite 19)
--move B rightup to follow its parent D
movenode(108, the locH of sprite 108, the locH of sprite 27,¬
        the locV of sprite 108, the locV of sprite 27)
--move A rightup to follow its parent B

movenode(110, the locH of sprite 110, the locH of sprite 47,¬
        the locV of sprite 110, the locV of sprite 47)
--move C to follow its parent B
set the visible of sprite 83 to FALSE
set the visible of sprite 84 to FALSE
set the visible of sprite 68 to TRUE
updateStage
startTimer
repeat while the timer < 2*60
 nothing
end repeat

movenode(112, the locH of sprite 112, the locH of sprite 51,¬
        the locV of sprite 112, the locV of sprite 51)
```

```
--move M rightdown
movenode(103, the locH of sprite 103, the locH of sprite 13,¬
        the locV of sprite 103, the locV of sprite 13)
--move L rightdown
movenode(113, the locH of sprite 113, the locH of sprite 28,¬
        the locV of sprite 113, the locV of sprite 28)
--move K to follow its parent L
set the visible of sprite 66 to TRUE
set the visible of sprite 65 to TRUE
set the visible of sprite 57 to FALSE
updateStage
startTimer
repeat while the timer < 2*60
  nothing
end repeat

movenode(104, the locH of sprite 104, the locH of sprite 10,¬
        the locV of sprite 104, the locV of sprite 10)
--move J rightdown
movenode(111, the locH of sprite 111, the locH of sprite 16,¬
        the locV of sprite 111, the locV of sprite 16)
--move E to root
movenode(106, the locH of sprite 106, the locH of sprite 24,¬
        the locV of sprite 106, the locV of sprite 24)
--move F to make it become J's left child
movenode(105, the locH of sprite 105, the locH of sprite 9,¬
        the locV of sprite 105, the locV of sprite 9)
--move H to follow its parent F
movenode(114, the locH of sprite 114, the locH of sprite 21,¬
        the locV of sprite 114, the locV of sprite 21)
--move I to follow its parent H
movenode(115, the locH of sprite 115, the locH of sprite 25,¬
        the locV of sprite 115, the locV of sprite 25)
--move G to follow its parent H
set the visible of sprite 56 to FALSE
set the visible of sprite 62 to FALSE
set the visible of sprite 73 to FALSE
set the visible of sprite 74 to FALSE
updateStage
set the visible of sprite 57 to TRUE
set the visible of sprite 64 to TRUE
set the visible of sprite 77 to TRUE
set the visible of sprite 78 to TRUE
updateStage
startTimer
```

```
  repeat while the timer < 2*60
   nothing
  end repeat

  movenode(109, the locH of sprite 109, the locH of sprite 20,¬
          the locV of sprite 109, the locV of sprite 20)
  --move D to follow its parent E
  movenode(107, the locH of sprite 107, the locH of sprite 35,¬
          the locV of sprite 107, the locV of sprite 35)
  --move B to follow its parent D
  movenode(108, the locH of sprite 108, the locH of sprite 19,¬
          the locV of sprite 108, the locV of sprite 19)
  --move A to follow its parent B

  movenode(110, the locH of sprite 110, the locH of sprite 46,¬
          the locV of sprite 110, the locV of sprite 46)
  set the visible of sprite 67 to FALSE
  set the visible of sprite 68 to FALSE
  set the visible of sprite 60 to TRUE
  updateStage
  --move C to follow its parent B

  puppetSound "root"
  updateStage

end
```

2

VITA

YINGJIE DONG

Candidate for the Degree of

Master of Science

Thesis: THE MULTIMEDIA ANIMATION FOR LEARNING SPLAY TREES

Major Field: Computer Science

Biographical:

Personal Data: Born in Hebei Province, P.R. China, October 15, 1968, the daughter of Zhilin Zhang and Wenli Dong.

Education: Graduated from the Cangzhou High School, Cangzhou, Hebei, P.R.China, in June 1986; received Bachelor of Arts Degree in English Literature and Language from Hebei Teacher's University in June 1990; received Master of Science Degree in Occupational and Adult Education at Oklahoma State University in May, 1998; completed requirements for the Master of Science degree in Computer Science at Oklahoma State University in July, 1999.

Professional Experience: Project Officer, Ministry of Labor of P.R.China, Beijing, 1994-1996; Research Associate, the 4th Research Institute, Beijing, 1992-1994; Instructor, Shijiazhuang Education Institute, Shijiazhuang, P.R.China, 1990-1992.