

AN INSTRUCTIONAL MODULE FOR TEACHING
ABOUT BINARY SEARCH TREES

BY

BIN SHEN

Bachelor of Engineering

Wuhan University of Water Transportation
Engineering

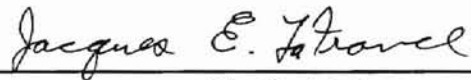
Wuhan, HuBei, P.R.China

1988

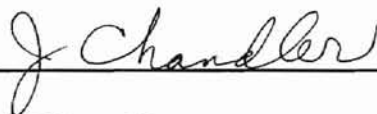
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1998

AN INSTRUCTIONAL MODULE FOR TEACHING
ABOUT BINARY SEARCH TREES

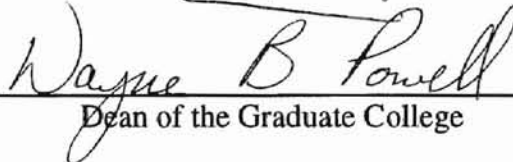
Thesis Approved:



Thesis Advisor







Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my utmost appreciation to my advisor Dr. Jacques LaFrance for his constructive guidance, supervision and inspiration. Without his understanding and support, I could not have accomplished this project. I also want to extend sincere thanks to Dr. J. P. Chandler, my thesis committee member, for his warm and constant guidance in addition to valuable suggestions and comments on this thesis. I would also like to express my deep gratitude to Dr. K. M. George for his assistance and for being my thesis committee.

My special thanks go to my father, Ze-Yuan Shen and to my mother, Su-qun Wang for their love and understanding. My deep appreciation is extended to my brother Ming Shen for his encouragement and support.

TABLE OF CONTENTS

Chapter
Page

1. INTRODUCTION.....	1
1.1. Background	1
1.2. Problem Statement.....	3
1.3. Purpose of the Study.....	3
1.4. Outline of Work.....	4
2. REVIEW OF THE LITERATURE.....	5
2.1. History of Educational computing.....	5
2.2. Related Work.....	7
3. METHODOLOGY	10
3.1. Software and Hardware Used in the Project.....	10
3.2. General Design of the Education Software.....	10
3.3. Design of the Cast Member.....	13
3.4. Design of the Frame.....	16
3.5. Navigation and Animation for Frames.....	20
4. CONCLUSIONS.....	35
REFERENCE.....	36

LIST OF TABLES

Table	Page
1. Names and Contents of the Sound Cast.....	15
2. Names and functions of the lists in this software.....	25

LIST OF FIGURES

Figure	Page
1. Components in the Design.....	12
2. Frame 1 - a Textual Frame in the Movie.....	17
3. Frame 8 - the Animation Frame in the Movie.....	19
4. Movie Script.....	21
5. Navigation from the current frame to next frame.....	22
6. Frame Script Creates the First Node of the Movie.....	22
7. Parent Script.....	23
8. The Script for insertion operation.....	27
9. Deleting a Node on the Leaf.....	29
10. Deleting a Node with Only One Subtree.....	30
11. Deleting a Node with Two Subtrees.....	31
12. Script for Inorder traversal.....	32

1. INTRODUCTION

1.1. Background

Data Structures, the study of methods of organizing large amounts of data, algorithm analysis and the estimation of the running time of algorithm [18], is one of the most important courses for students in a computer science department. By analyzing an algorithm before it is actually coded, students can decide if a particular solution will be feasible. In a word, it is very important for students to master this course completely. Actually, it is not easy for many students to learn this course, especially for students who have no background in computer science. Generally, trees are very useful abstractions in data structures. Students sometimes find it hard to learn the programming of tree data structure algorithms. Most textbooks of data structures explain tree structures in textual format with static illustrations, and graphs and algorithms' explanation are not clear enough. Actually, it is far from easy for some students to study data structures of trees and deeply understand their basic concepts and algorithms. The reason is that they can not visualize what really happens in the construction and development of the tree algorithms, such as insertion and deletion operations. Static drawings just show the results of procedures. Tree structures are too abstract for students to understand quickly and totally.

Using the computer as a tool is a good idea. Actually, computer-assisted instruction (CAI) has already proven itself effective in improving students achievement

[12]. Algorithm animation, the visualization of the fundamental operations of running program, has proven to be a powerful tool in the teaching of algorithms [10]. There is some educational software to show tree algorithms. It seems that all this software is not produced as complete instruction units.

My project will be made as an instructional unit (also called class). It will include explanations of concepts and algorithms, interactive animation of algorithms and a test. This computer software assumes the role of teacher and presents the material in a learning format. The student gradually moves from one step to the next, usually at his or her own pace. The student is actively involved, by seeing and listening to the text explanation and algorithms' animation, and finally by taking a test. To help the student comprehend the task of an algorithm to maintain a data structure, interactive animation of these data structures' algorithms will allow the students to explore different scenarios in the construction of the data structures and better understand what their programs need to do. There are many tools that can create an animated presentation of a data structure. Director is ideal tool for creating simulation, visualization and interactive presentation. This thesis will develop animated presentation of the construction of a binary search tree by using Director 6.0.

1.2. Problem statement

The labor market in computer technology and application is increasing rapidly, more and more engineers, programmers and specialists in system management will be needed. Education in computer science is a bottleneck. Many people are changing to study computer, and some have no background in computer science. All of these situations lead to a challenge for education in computer science. One problem is that many students need to be trained but fewer teachers are available, another is to find a good way of computer learning for different levels of learner. Using the computer as a tool for learning has been adapted and proven in enhancing student performance [7]. As mentioned earlier, most data structures textbooks explain tree structures in text, and traditional education like lecture in class is hard for students to understand well and rapidly. This results in mistakes and difficulties in programming. Much instructional software for trees can help students to learn tree data structures. It seems, however, that they are not practical and can't serve as the role of teacher. In order to improve students' performance of learning binary search trees, a complete instructional unit will be developed by using Macromedia Director 6.0.

1.3. Purpose of the Study

The purpose of the thesis is to develop a practical animated presentation of the concept and algorithms of binary search trees. It can create simulation, visualization and

interactive presentation during the whole class. It includes presentation of concepts and algorithms, animation of algorithms (including sound synchronized with animation), and testing with some problems about binary search trees. Answers will be given in the last part, so students can evaluate their mastery of this class.

1.4. Outline of Work

This thesis is organized as follows. In this chapter the background of study, problem statement and purpose of the study are addressed. In Chapter 2, history of the instruction software and some related work are reviewed. In Chapter 3, design of this instruction software is introduced. Finally, conclusions of the thesis and suggestions for future study are given in Chapter 4.

2. LITERATURE REVIEW

2.1. History of Educational Computing

There have been many changes in the way educators have used computing in their schools over the last two decades brought about, to a large extent, by an increase in availability and sophistication of both computer and software. Unfortunately, there is still a strong emphasis on drill and practice and reluctance to experiment with more creative approaches to educational computing.

Suppes' paper on using computers for problem solving and computer programming is intriguing because Suppes used a PDP-10 Mainframe computer with terminals in the children's homes to teach LOGO, BASIC and logic [16]. The two revolutions in the computer industry, growth in the power of personal computers and an exponential growth in the quality and variety of computer software, have brought the computer closer to the day-to-day problems and needs of children and they have provided them with computing power of such vastly different kinds than practiced by Suppes more than two decades ago [17].

Bork recommends guidelines for writing teaching dialogues for educational software and for increasing pupil interaction with the computer [2]. When Bork counsels on the importance of graphics, we realize that software has made gigantic leaps and bounds in the last few years.

Moursund [11] pointed out that it is important to increase teachers' level of computer-education literacy and underscored the changes in computer literacy. BASIC programming and some use of software is less importance now because access to computers has increased so dramatically.

With today's approaches to educational computing with improved software and hardware, it is easy to image and design infinitely more creative curriculum opportunities. These opportunities are ideal for more personalized, flexible and responsible teaching/learning environments which characterize so many of our special education classrooms.

Using multimedia technology to develop practical educational software is becoming popular today because it can provide interactive animation. From the angle of cognitive psychology, seeing is believing, and furthermore an important part of perception is the information-gathering responses [15]. Animation, orienting our eyes to a stimulus and tracking its movement with our eyes, is one of the most important responses for perception. It is much better than receiving information by an abstract representation [19]. Algorithm animation is concerned with illustrating the behavior of a program by visualizing the fundamental operations of the program as it runs[4]. Such displays have proven to be quite useful for education [5]. It is valuable to develop a practical instructional software for trees algorithm by using multimedia.

2.2. Related Work

This part reviews previous and current work that is related to my research. In the early '60s, graphics hardware was a scarce and expensive commodity. It was difficult for users to use it for education. The way to solve this problem in the mid-'70s was to record the computer-generated images on film.

The first computer-generated movie concerning computers was produced by Knowlton at Bell labs in 1966. The movie, L6: Bell Telephone laboratories Low-Level Linked list Language [9], portrayed the workings of an assembly-level Link-processing language. Hopgood's movie on hashing algorithms [8] in 1974 was the first movie whose purpose was to portray an algorithm. In the mid-70s, much work was done at the University of Toronto under Baecker's direction in the area of algorithm animation [1]. Baecker's stellar Sorting Out Sorting film illustrated a number of different sorting algorithms running on both small and large data sets.

The basic limitation of these films are, of course, that they do not allow viewers to experiment with the model being displayed. Viewers must watch the films in the exact form in which they were produced, with the exact parameters and the exact data.

Obviously, the interactive instruction software is more suitable for the learner. Today the microcomputer serves only one person, it can act almost instantaneously on any request that a person might give it. With the earlier constraints of cost and undependability

eliminated by the microcomputer, it is no surprise that interest in computer-assisted instruction has once again exploded.

Harbison developed Modula-3, incorporating interactive design to make it would easy for non-experts to implement additional algorithm animations of their choice [6].

Brown improved on Web-base algorithm animations by providing a framework that makes it easy to construct new animations, including those that involve multiple views[3].

Shen developed TBDSV System for animation of trees data structure under the X window environment [13]. Several trees like AVL-tree, B-tree, Red-Black tree and splay tree are discussed. It is a very good approach to teaching tree structures.

Dr. Daniel Sleator developed a demonstration of top-down splaying. This program allows one to see how a splay tree evolves [14]. It will ask the user to choose how many nodes wanted in the tree. It will construct a maximally skewed tree with that number of nodes, and draw it. One can specify a node to be displayed, and can insert and delete nodes. Many tree structures are presented by some animation can be found at the following Web site: <http://langevin.usc.edu/BST/>.

There are many educational programs for the algorithms of trees structures. However, it seems that all of this software is not developed as a complete instruction unit. For the binary search tree, some important operations like inorder, preorder and postorder are not included in this software. Some software is not interactive, and some is not

synchronized with sound. I think it is useful to develop software as a complete instructional unit (also called a class). It includes presentation of concepts, animation of algorithm and testing. It will let students grasp the BST by learning this class.

Using Director to create a movie, the essential elements worked with are the stage, the score, the cast window and cast members.

The stage is the backdrop for all Director movies.

The score records the state of all the elements in the movie over time. It is the primary tool for creating and editing movies. Actually, what we see on the stage at a certain time is the frame which contains information about every cast member active at a specific time.

A cast member is any media element that is part of a movie. Everything seen or heard in a movie is a cast member. Cast members include not just the graphics displayed and animated on the screen, but text, sound effects and music, color palettes, buttons and the lingo scripts that provide interactive control of a director movie.

The cast window displays the members of the current cast. As many windows as needed can be opened to display the different casts in the movie.

The first step in the development of the movie is to design the basic elements in the movie: the cast members. The second step is to use the score to make frames to locate cast members in the movie. The third step is to design the interactive function between frames and specific animation on every frame. Actually, the first and the second steps are the design of the interface. The third step is the design of visualization and the algorithms. Figure 1 shows the components in the design.

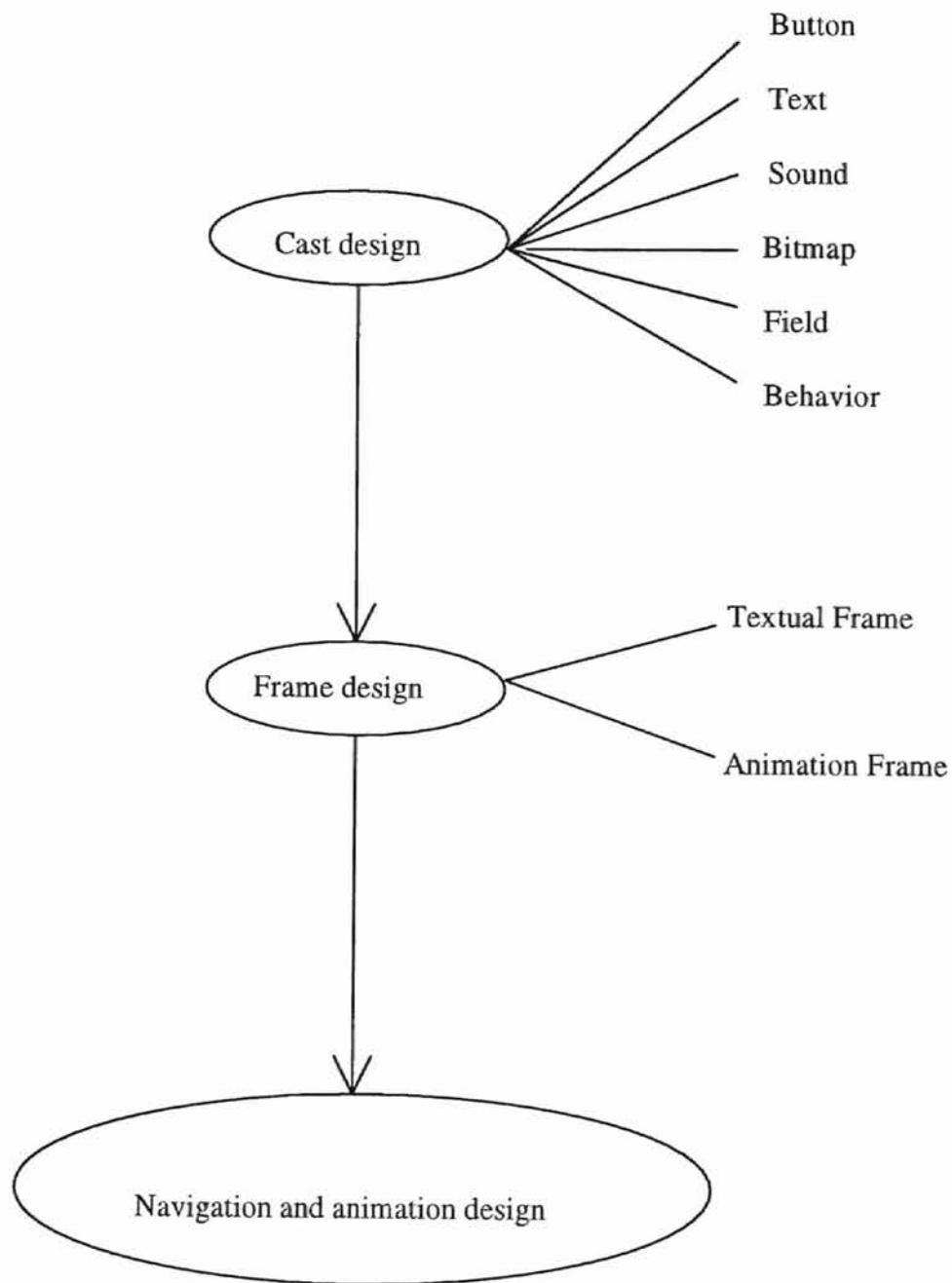


Figure 1. Components in the design.

3.3. Design of the cast member

The cast members we need to use in the movie are as follows:

1) Button

For the transition from one frame to another or performance of different operations, buttons are to be used. From Lingo one can easily monitor button cast members and control how they respond. In the movie, “next” buttons, a “answer” button and a “test” button are to be used to change frames, and buttons like “insertion”, “deletion”, “preorder”, “inorder”, “postorder”, “soundset” and “quit” are to be used for corresponding operations.

2) Text

Text casts are to be used to explain the content of the program, menu of the program, the basic concepts, algorithms, and the test.

3) Bitmap

Bitmap cast members can be created in the paint window. When a cast member is placed on the stage, Director creates a new instance of the cast member called a sprite. In the movie, like the sprites of the nodes, the sprite of “node left” and other drawings on the animation frame are created from bitmap casts.

4) Behavior

Behaviors are special cast members that define operations or procedures. In order to move the playback Head step by step, several behavior cast members are to be created. These behaviors have the movie wait and loop when the playback enter a frame.

5) Field

Working with fields is almost identical to working with text cast members. Unlike text cast members, fields can be edited while a movie plays. In the movie, the input box is the field where the user can input new node value. Fields also can be controlled by Lingo in ways not possible with text cast members. Value slots of each node are this kind of field casts.

6) Sound

Sound casts are to be used to play audio tracks during animation to explain algorithms clearly. All operations in the movie are to be synchronized by the specific sound cast. Table1 show the sound cast names and contents.

Table 1

Names and Contents of the Sound Cast

Sound cast name	Sound content
insertion	a value to be inserted is compared with each node down path in the tree to find the correct position and added to the last node on the path
leaf delete	leaf node will be deleted directly
two subtrees	a deleted node which has left and right subtree will be deleted and replaced by the smallest node of right subtree
right subtree	a deleted node with only a right subtree will be deleted and its right subtree will replaced it in the tree
left subtree	a deleted node with only a left subtree will be deleted and its left subtree will replaced it in the tree
inorder	an inorder traversal processes its left subtree first, the processes the tree's root, finally processes its right subtree
preorder sound	a preorder traversal processes a tree's root first, then processes its left branch in preorder, finally processes its right branch in preorder
postorder sound	a postorder traversal processes its left subtree in postorder, then processes its right branches in postorder, finally processes the root of the tree

3.4. Design of the frame

Frames represent a single step in the movie, like the frames in a traditional film. In this movie, every frame has more than one cast member in it. Sprites are objects that control when, where, and how cast member media appear in a movie. Sprites appear on the stage in order according to their channel number. A sprite in the lower channel number appears on the top of a sprite in higher channel number.

1) Textual explanation frames

Several frames are designed to explain the concept of BST, the algorithms of BST, the animation menu and the test. Each frame includes both a text cast and a button cast. The button sprite has a high channel number. Figure 2 shows a textual frame in the movie.

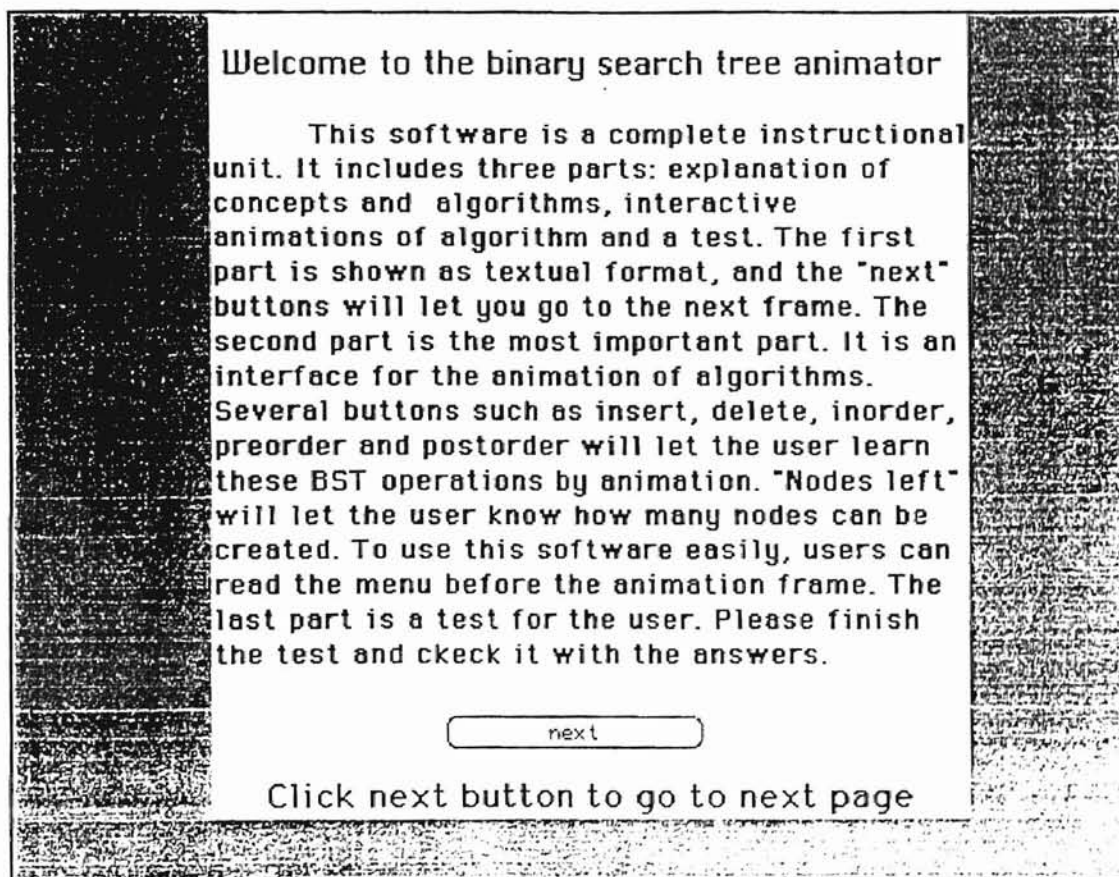


Figure 2. Frame 1 - a textual frame in the movie.

2) Animation frame

This frame is the most important frame in the movie. It include all kinds of casts which have been created. The following visualization and algorithm design features are to be made for this frame. The input field box is located in the upper left of the stage. All the operation buttons are located at the bottom of the stage. Figure 3 shows the animation frame.

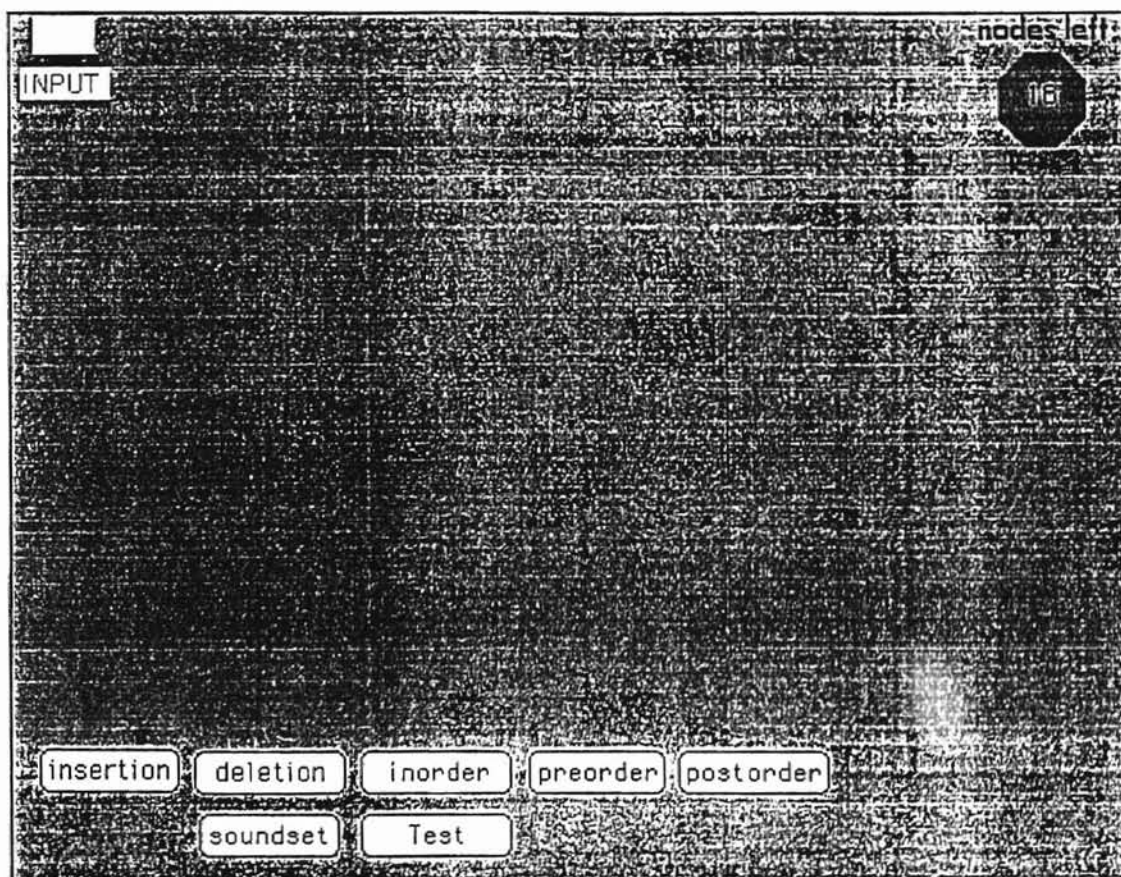


Figure 3. Frame 8 - the animation frame in the movie.

3.5. Navigation and animation for frames

In Director's interface, dragging an existing behavior from a cast window to a sprite or frame attaches the behavior to the object in the movie. Lingo is the Director's scripting language which enhances a movie's possibilities. It can combine animation and sound in ways that the score alone can't, and let the user precisely control text sound, images and digital video. Scripts are combinations of words that convey information and instructions for a movie. In this movie, movie script, frame script and parent script are to be used. Lists have many uses in Director. Structuring records and tracking sets of objects are two common ones. Several lists are to be used in the movie design.

1. Movie script design

Movie scripts are available to the entire movie. In the movie script, some important designs are declaring global variable, initializing of the first node location and puppeting channels under the control of Lingo. Figure 4 is a part of a movie script.

```

on startMovie
    global root, maxNodes, firstNode, nodelist, valuelist, valuelistx, valuelisty,
    valuelist1, nodelist1, preshowlist, templist, postshowlist, showlist, postlist
    --declare all global variables

    set maxNodes to 22
    --the number of node can be created
    set firstNode to 5
    --the sprite number of the firstNode

    repeat with k = firstNode to maxNodes + firstNode - 1
        --nodes 0 - 21 are sprites 5-26
        puppetSprite k, TRUE
    end repeat
    repeat with k = maxNodes + firstNode to maxNodes*2 + firstNode + 1
        --values of nodes are sprites 27 - 48
        puppetSprite k, TRUE
    end repeat
    --make channels under control of Lingo

    set the locH of sprite firstNode to 300
    set the locV of sprite firstNode to 40
    set the locH of sprite firstNode + maxNodes to 300
    set the locV of sprite firstNode + maxNodes to 40
    --set original location of the first node

    .
    .
    .
    .
    .
end startMovie

```

Figure 4. Movie script.

2. Frame scripts design

Frame scripts which attached to the script channel determine what happens when the playback head enters, exits or is in the frame that the script is attached to. Navigation lets users explore the movie the way they want, playing parts of the movie that offer additional information, and staying in or repeating the parts of the movie. Figure 5 shows the frame has the movie go forward when the user clicks the sprite, so the user can look the frames one by one. Figure 6 shows the first node created by the frame script.

```
on exitFrame  
    go to the frame  
end
```

Figure 5. Navigation from the current frame to next frame.

```
on enterFrame  
    global root, nodeList, maxNodes  
    set root to birth(script "Node Parent", 0, 0, 0)  
    set the text of cast 30 to " "  
    add nodeList, root  
    put maxNodes - count(nodeList) into field "Node left"  
    updateStage  
end
```

Figure 6. Frame script creates the first node of the movie.

3. Parent script design

The benefit of parent scripts comes from Lingo's capability of creating multiple copies or instances of the script's content. Each instance of the parent script is a child object.

A parent script contains a set of handles and sometimes an additional statement that declares which variables are property variables. The property variables are values that each child objects maintains. Figure 7 shows implementation of a parent script.

```
property number, Boxes, Values, spriteNum, valueNum
--property variables which each child object can maintain its own values

on birth me, nodeNumber, cloneNum, cloneNode
    global nodeList, naxNodes, firstNode

    .
    .
    set number to nodenumber
    set spriteNum to nodeNumber + firstNode
    set valueNum to nodeNumber + 30
    set Boxes to birth(script "Box Parent", count(nodeList), cloneNum)
    --create a circle sprite
    set Values to birth(script "values Parent", count(nodeList), cloneNum)
    --create a value sprite
    return me

on insertToken me, Token

--handler for creating new node
```

Figure 7. Parent script

4. Lists used in design

Lists in this software design play an important role, structuring records, tracking sets of objects, and setting up the parameters used to define behaviors. In addition, Lingo can create, sort, add to, reorder, or substitute a list's contents.

Director provides two types of list:

- 1) Linear lists, in which each element is a single value.
- 2) Property lists, in which each element contains two values separated by a colon. The first value is the property. The second value is the value associated with that property.

Two kinds of lists are to be used in this software. The following Table 2 shows all the names and functions of the lists in this software.

Table 2

Names and Functions of the Lists in this Software

List Name	List Type	Function of lists
nodelist	linear list []	undeleted newnode list, include all created nodes
nodelist1	linear list []	current newnode list, not include the deleted nodes
valuelist	property list[:]	string of every node and its associated sprite number
valuelistx	property list[:]	horizontal position value of every node and its associated string
valuelisty	property list[:]	vertical position value of every node and its associated string
valuelist1	linear list []	ordered string list by entering sequence
preshowlist	linear list []	save the sprite numbers of nodes on the stage by preorder
templist	linear list []	list for storing the data temporary
showlist	linear list []	save the sprite numbers of nodes on the stage by inorder
postshowlist	linear list []	list for storing the data temporary
postlist	linear list []	save the sprite numbers of nodes on the stage by postorder

5. Algorithm design

It is necessary to decompose an algorithm into a set of functions that best represents the BST's distinct behaviors. For the software of BST, this will include

insertion, deletion, and inorder, preorder and postorder traversals. Subsequently, the algorithms for every situation are coded.

1) Insertion

For the insertion function the following points need to be considered:

- a) a parent script needs to be called to create a new node,
- b) compute the x and y value of the location for the new node, then move the new node to the computed location
- c) insert sound script to synchronize with the period of animation
- d) store the all information in the lists for the new node

Figure 8 show the script for the insertion operation.

```

On insertToken me, Token
.
.
set newToken to Token
if newToken <> "" then
set newNode to birth(script "Node Parent", count(nodeList), number, me)
--create a new node
puppetSound "insertion"
updateStage
--sound synchronized with animation
.
.
if collide then
    if Token < getPropAt(valuelist, j) then
        -- insert value less than node value, move down along edge of the left subtree
        set t to 2
        set M to  $-4/(2*k)$ 
        moveedge newNode, t, M
    else
        --insert value larger than nodevalue, move down along edge of the right subtree
        set t to 2
        set M to  $4/(2*k)$ 
        moveedge newNode, t, M
    end if
.
.
append(nodeList, newNode)
append(nodelist1, newNode)
addProp(valuelist, newToken, the spriteNum of newNode)
addProp(valuelistx, newToken, the locH of sprite the spriteNum of newNode)
addProp(valuelisty, newToken, the LocH of sprite the spriteNum of newNode)
append(valuelist1, newToken)
sort valuelistx
--store information of newNode in the lists
.

```

Figure 8. The script for insertion operation.

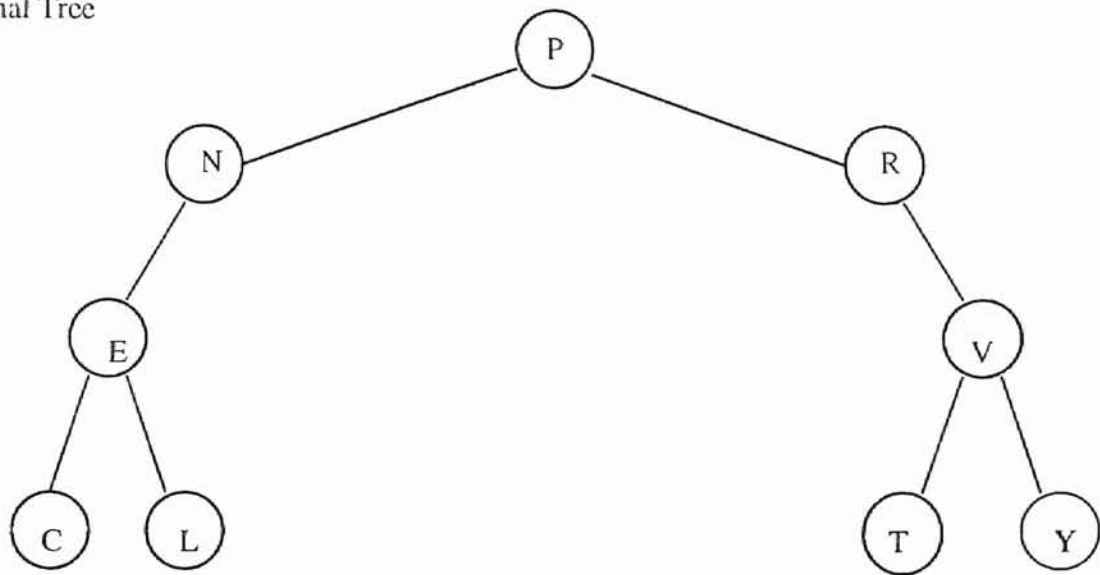
2) deletion

For the deletion function, three situations need to be considered:

- a) If the deleted node of a BST has a left child but no right child, remove the node from the tree by making the node's parent point to the node's left child.
- b) If the deleted node of a BST has a right child but no left child, remove the node from the tree by making the node's parent point to the node's right child.
- c) If the deleted node of a BST has both right and left subtree, the node will be replaced by the smallest node of the right subtree.
- d) If the node is leaf, it can be disconnected from its parent node.

The coding of the algorithms for deletion must be precise in accordance with the every condition. Different sound scripts will be added to synchronize animation for different situations. Lists for storage of the tree information will be changed for every condition. Figures 9,10,11 show the different situations for deletion.

Original Tree



a) Delete BST(T, 'c')
(a leaf)

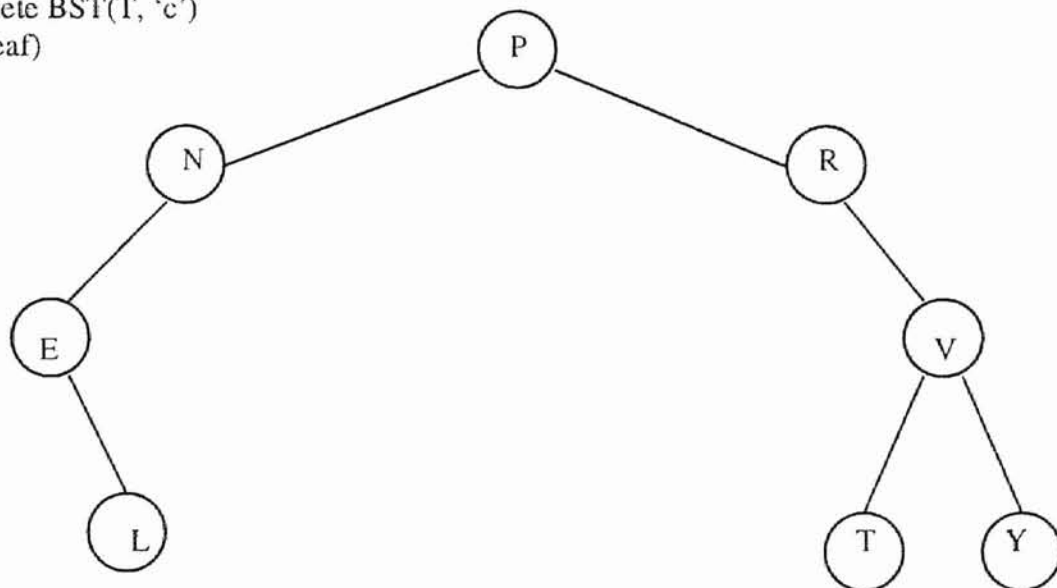
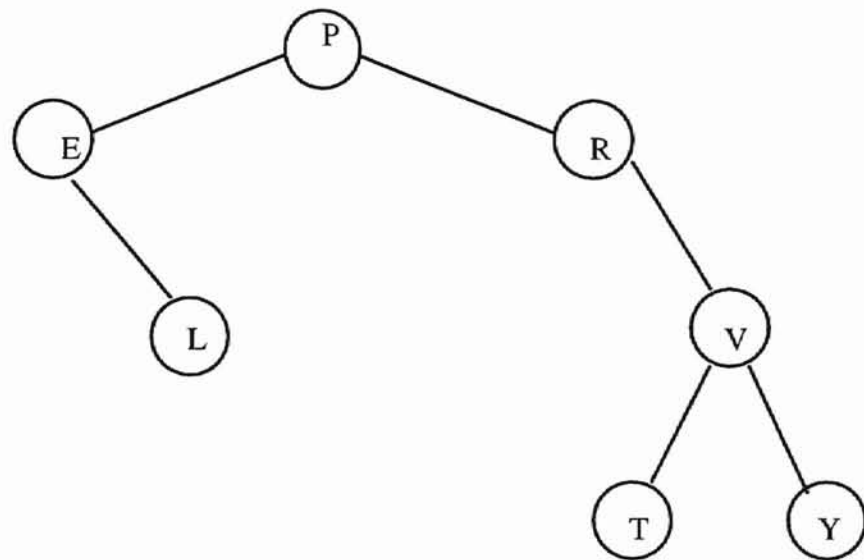


Figure 9. Deleting a node on the leaf .

b) Delete BST(T, 'N')

(left child only)



c) Delete BST(T, 'R')

(right child only)

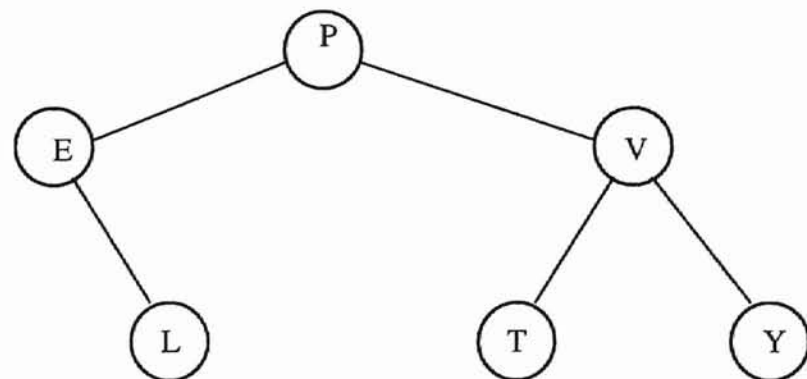


Figure 10. Deleting a node with only one subtree.

d) Delete BST(T, 'P')

(both children)

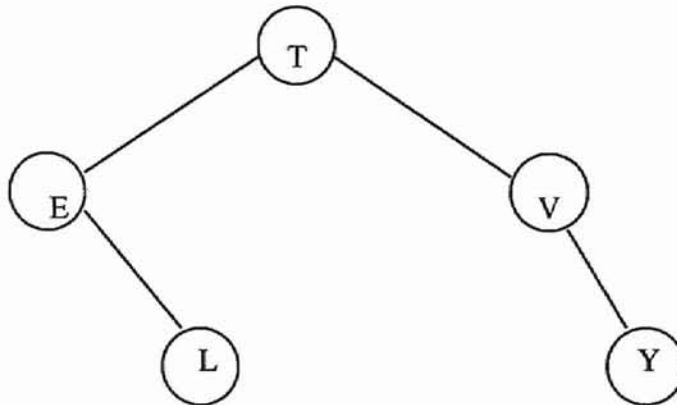


Figure 11. Deleting a node with two subtrees

3) Preorder, inorder and postorder

For the traversal function inorder, the sprite number of the every node will be stored in the list called showlist by inorder sequence. Nodes will undergo changes in color by the sequence in which the function changes the forecolor value for the sprite. The animation will be accompanied by a sound track for explanation. The design of the preorder and postorder traversals is the same. The following paragraph explains the design of the inorder function.

An inorder traversal processes the nodes from left to right. The valuelistx is for storing an x value and it is corresponding sprite number for every node in the ordered list.

It is sorted after every operation. The showlist is used to save the information in valuelistx.

When the color of the node with sprite is changed by the list order, the visualization of inorder will be illustrated. Figure 12 shows the script of the inorder function.

```
.  
.   
repeat with k=1 to count(valuelistx)  
  set showToken=getPropAt(valuelistx, k)  
  set showSprite=getaProp(valuelist, showToken)  
  --get the sprite number by order  
  append(showlist, showSprite)  
  --add the ordered sprite into showlist  
end repeat  
  
repeat with j=1 to count(showlist)  
  set showNum=getAt(showlist, j)  
  set originColor to the foecolor of sprite showNum  
  --record the original color  
  
  set the foreColor of sprite showNum to 190  
  updateStage  
  --change the node color  
  
  startTimer  
  repeat while the timer < 2*60  
    nothing  
  --delay the time  
end repeat  
  
.  
.
```

Figure 12. Script for inorder traversal.

6. Visualization Design

Visualization design is an important part in this instructional software. The following requirements are to be considered:

- 1) Easy shape recognition: It must be easy for viewer to associate the shape or color with node it represents.
- 2) Smooth movement: Image movement must be smooth and continuous during the specific operation.
- 3) Sound synchronization: The speed of movement should be adjusted, so the sound will be synchronized with the animation
- 4) Moving path calculation: Moving path needs to be compute so that the movement of the node will represent the specific algorithm. For the insertion of a newnode, the node will start at the root of tree. Comparing its value with values of the nodes on the moving path, it will move along the edge of the subtree until it reaches the suitable location.
- 5) Wrong input control: The design needs to consider the following wrong operations:
 - e) the node is off the stage
 - f) deleted node is not an element of the current tree
 - g) inserted node already exists
- 1) Visualization of tree's edge: Edges of the tree will appear or disappear according to different operations.

7. Summary of navigation and animation design

Actually, the different design described above is not completely independent. All these designs are correlated with each other. It is necessary to decompose the general design into a set of parts to simplify the problem. When the specific part is to be designed, many factors need to be considered. For example, when we design the insert algorithm, obviously, we need to consider the visualization, utilization of lists, synchronization of the sound and so on. The procedure of the design is developed and improved step by step.

RERERENCE

1. Baecker, Ronald M. "Two System Which Produce Animated Representations of the Execution of Computer Programs" ACM SIGCSE Bulletin, 7, 1, 158-167, Feb. 1975.
2. Bork, Alfred "Interactive Learning" American Journal of Physics, 47 (1), Jan. 1979.
3. Marc H. Brown and Marc A. Najork. "Collaborative Active Textbooks: A Web-Base Algorithm Animation System for an Electronic Classroom" SRC Research Report, #142, DEC Systems Research Center, Palo Alto, CA, May, 1996.
4. Marc H Brown and Marc A. Najork. "Distributed Active Oblets" SRC Research Report, #141, DEC Systems Research Center, Palo Alto, CA, May, 1996.
5. Buisberg, Robert A. "Animated Graphical Interfaces Using Temporal Constraints" ACM CHI, 131-136, 1986.
6. Harbison, Sam. Modula-3. Prentice-Hall, Englewood Cliffs, NJ, 1992.

7. Harper, Dennis O. RUN: Computer Education Monterey, Calif.: Brooks/Cole Pub. Co., c1983.
8. Hopgood, F. Robert A. "Computer Animation Used as a Tool in Teaching Computer Science", Proc. 1974 IFIP Congress, 1974.
9. Knowlton, Kenneth C. L6: Bell Telephone Laboratories Low-Level Linked List Language, two 16mm black and white sound films, Bell Telephone laboratories, Murray Hill, NJ, 1966.
10. Lawrence, Andren "Empirically Evaluating the Use of Animation to Teach Algorithms" IEEE Symp. In Visual Languages, pp48-54, Oct. 1994.
11. Moursand, D. Introduction to Computer in Education for Elementary and Middle School Teachers Oregon: ICCE, 1983.
12. Orwig, Gray W. Creating Computer Program for Learning Reston, Virginia: Reston Publi. Co. A Prentice-Hall Co., 1983.
13. Shen, H. C. A Visual Aid for Learning of Tree-Based Structure, M.S. thesis,

Computer Science Department, Oklahoma State University, 1994.

14. Sleator, Daniel. Web site: [Http://www.cs.cmu.edu/Sleator](http://www.cs.cmu.edu/Sleator), maintained by School of Computer Science, at Carnegie Mellon University. 1997.

15. Solso, Robert L. Cognitive Psychology Chicago: Harcourt Brace Jovanovich, Inc, 1979.

16. Suppes, Patrick "Impact of Computers on Curriculum in the Schools and Universities", Scientific American, Vol 215, pp. 206-220, Sept. 1966.

17. Taylor, R. P.(Ed) The Computer in the School: Tutor, Tool, Tutee, New York: Teacher's College Press, 1980.

18. Weiss, Mark Allen. Data Structures and Algorithm Analysis in C, Redwood City, Calif.: Benjamin/Cummings Publishing Company, Inc., 1993.

19. Wickelgren, Wayne A. Cognitive Psychology Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1979.

VITA

Bin Shen

Candidate for the Degree of

Master of Science

Thesis: AN INSTRUCTIONAL MODULE FOR TEACHING ABOUT BINARY
SEARCH TREES

Major Field: Computer Science

Biographical:

Personal Data: Born in WuHan, P. R. China, on May 14, 1966, the son of Ze-Yuan Shen and Su-qun Wang.

Education: Graduated from WuHan Middle School in WuHan, July 1984. Received Bachelor of Engineering from WuHan University of Water Transportation Engineering. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December 1997.

Experience: Employed with Daya Bay Nuclear Power Plants as a Engineer from May 1990 to July 1992. Employed with WEI-DA Industrial Company as an engineer from August 1992 to April 1995.