VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi

A Mini-Project Report on

"VISUALIZATION OF BINARY SEARCH TREE USING OPENGL"

Computer Graphics & Visualization Laboratory with Mini Project

15CSL68

Submitted by

Prajwal Bharadwaj N **(4VV16CS079)**

Prajwal S **(4VV16CS082)**

Under the Guidance of

Pavan Kumar S P
Assistant Professor

Gururaj H L
Assistant Professor

VVCE

Department of Computer Science & Engineering

VIDYAVARDHAKA COLLEGE OF ENGINEERING
MYSURU-570 002

2018-19

Vidyavardhaka College of Engineering

Department of Computer Science and Engineering

Mysuru, Karnataka 570 002

(Affiliated to Visvesvaraya Technological University, Belagavi)

Certificate

This is to certify that the Mini-project work entitled **"Visualization of Binary Search Tree using OpenGL"**, is a bonafide work carried out by **Prajwal Bharadwaj N (4VV16CS079) and Prajwal S (4VV16CS082)** having completed the Computer Graphics & Visualization with Mini-project (15CSL68) during the year 2018-2019.

Guide	**Dr. Ravikumar V**
Assistant Professor,	Professor & Head,
Dept. CS & E,	Dept. CS & E,

Dr. B Sadashive Gowda
Principal, VVCE

External Viva

Name of the Examiners	Signature with date
1.	
2.	

ABSTRACT

This project is about visualizing Binary Search Tree. It is implemented using different primitives available in OpenGL library and combining them together in a required manner. It highlights the key features of Binary Search Tree. This project consists of Binary Search Tree which is constructed by using the concept of Linked Lists. Through this Project one can visualize the working of various features of a Binary Search Tree. Computer Graphics allows the user to visualize the concept of Binary Search Tree very easily.The project illustrates the role of different callback functions that provides easier way to accomplish the project in an effective manner. The project has been implemented by efficiently using the data structure to obtain the optimized results and also various functions and features that are made available by the OpenGL software package have been utilized effectively.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This chapter gives brief introduction about computer graphics, the features of glew and glut and binary search tree.

1.1 COMPUTER GRAPHICS

Computers have become a powerful tool for the rapid and economical production of pictures. There is virtually no area in which graphical displays cannot be used to some advantage, and so it is not surprising to find the use of computer graphics so widespread. Although early applications in engineering and science had to rely on expensive and cumbersome equipment, advances in computer technology have made interactive computer graphics a practical tool. [8] Computer graphics are pictures and films created using computers. Usually, the term refers to computer-generated image data created with the help of specialized graphical hardware and software.

1.2 OPENGL

Open Graphics Library (OpenGL) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering. Modern OpenGL has raised the entry threshold for computer graphics programming.[2]

1.3 GLUT

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres and the Utah teapot. GLUT also has some limited support for creating pop-up menus.

1.4 FREEGLUT

FreeGLUT is a free-software/open-source alternative to the OpenGL Utility Toolkit (GLUT) library. GLUT (and hence FreeGLUT) allows the user to create and manage windows containing OpenGL contexts on a wide range of platforms and also read the mouse, keyboard and joystick functions. FreeGLUT is intended to be a full replacement for GLUT, and has only a few differences.

1.5 GLEW

The OpenGL Extension Wrangler Library (GLEW) is a cross-platform open-source C/C++ extension loading library. GLEW provides efficient run-time mechanisms for determining which OpenGL extensions are supported on the target platform. OpenGL core and extension functionality are exposed in a single header file.

1.6 MICROSOFT VISUAL STUDIO WITH VISUAL C++

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Microsoft Visual C++ is Microsoft's partial implementation of the C and full implementation C++ compiler and associated languages-services and specific tools for integration with the Visual Studio IDE. It can compile either in C mode or C++ mode.

1.7 BINARY SEARCH TREE

There are many storage structures available to store data in memory of many forms. These structures can be array, class, linked list with its various forms, Tree, Binary Tree, Binary Search Tree (BST), etc. These can be differentiated in two major forms. First one uses continuous memory allocation and the second one can occupy any free memory block by pointed by the other memory locations. The Binary Search Tree is more efficient than the other mentioned data structures [1]. Binary tree is a graph, without cycle, that is frequently used in computer science for fast data access and retrieval [6]. A random binary search tree is defined as the random binary tree obtained by consecutive insertion of x1,x2,..,xn to an initially empty tree, where x1,x2,..,xn is either an independently and identically distributed sequence of random variables with a fixed density or an equivalent random permutation of l,2, ..., n [7]. Binary Search Tree (BST) is one of the most widely used techniques for searching in non-linear data structure [3]. Working with large BSTs can become complicated and inefficient unless a programmer can visualize them [4].

CHAPTER 2

BINARY SEARCH TREE

This chapter explains about the features of Binary Search Tree that have been implemented in the project.

2.1 PROBLEM DEFINITION

Visualization of Binary Search Tree using openGL for the following features

1. Insertion of nodes
2. Deletion of nodes
3. Searching a node

2.2 BINARY SEARCH TREE

It is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys lesser than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- The left and right subtree each must also be a binary search tree.

We are implementing the BST using Linked List and Arrays. We use separate functions for each of the following features. Nodes are represented using circles and the connections are shown using lines. We translate a node to the required position when necessary and also rotate the lines.

2.3 INSERTION OF A NODE

A new node is always inserted at leaf. We start searching a node from root till we hit a leaf node. Once a leaf node is found, the new node is added as a child of the leaf node.

2.4 DELETION OF A NODE

When we delete a node, three possibilities arise.

- Node to be deleted is leaf: Simply remove from the tree.
- Node to be deleted has only one child: Copy the child to the node and delete the child.
- Node to be deleted has two children: Find inorder successor of the node. Copy contents of the inorder successor to the node and delete the inorder successor. Note that inorder predecessor can also be used.

2.5 SEARCHING A NODE

To search a given key in Binary Search Tree, we first compare it with root, if the key is present at root, we return root. If key is greater than root’s key, we recur for right subtree of root node. Otherwise we recur for left subtree.

CHAPTER 3

METHODOLOGY

This section gives information regarding the user defined functions and GLUT functions used in the project.

3.1 Pseudo Code

```
class binary_search_tree {

    binary_search_tree() // Constructor

    get_minimum() // To get the minimum valued node

    draw_node() // To display a node for the BST

    draw_arrow() // To draw arrows between nodes

    insert() // To insert a new node for the BST

    search() // To search for a node in the BST

    remove() // To delete a node from the BST

    pre_order() // To traverse the BST

    draw_tree() // To display the BST

    accept_tree() // To input insertion elements

}
```

```
class option_box {

        option_box() // constructor

        draw_box() // to draw a box for options

        clicked() // to check whether the option is clicked or not

}

//Other functions

count_digit() // to count the number of digits

to_string() // to convert numbers to string

display_string() // to display string on output window

clear_input_region() // to clear specific area on screen

init() // initialization specifications

display_options() // to display available options

display() // display specifications

mouse() // mouse function

keyboard() // keyboard function

main() // main function
```

3.2 GLUT functions

glBegin(): Initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL_POINTS, GL_LINES and GL_POLYGON.

glEnd(): Terminates a list of vertices

glutInit(&argc,argv): Initializes GLUT and processes any command line arguments(for x, this would be options like display and geometry). glutInit() should be called before any other GLUT routine.

glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB): Specifies whether to use an RGBA or color_index model. You can also specify whether you want a single or double buffered window. if you are working in color_index mode,you will want to load certain colors into the color map use glutSetColor() to do this. Finaly, you can use this routine to indicate you want the window to have an associated depth, stencil, and/or accumulation buffer.

glutCreatewindow("Binary Search Tree"): Creates a window with an openGL context.it returns a unique identifier for the new window. Until glutMainLoop() is called the window is not displayed.

glutInitWindowSize(x_size,y_size): Specifies the size in pixels of your window.

glutInitWindowPosition(int x,int y): Specifies the screen location for the upper_left corner of your window.

glutDisplayFunc(display): It is the first and most important event callback function you will see. Whenever GLUT determines the contents of window, sometimes you will have to call glutPostRedisplay(void), which gives glutMainLoop() a nudge to call the registered display callback at its next opportunity.

Chapter 4

IMPLEMENTATION

This chapter tells us the procedure to be followed in order to execute the project and implement the program.

4.1 Steps Involved

1. Click on the Run Button in visual studio after the project is opened

2. Enter the elements to be inserted in the black screen and hit 0 when over

3. Click anywhere on the output window to see the tree

4. Click and hold on the buttons available for the features

5. Enter the element and press Enter. Based on the option selected

    a. Insert: The entered element will be added to the tree as a new node

    b. Delete: The entered element will be deleted from the tree

    c. Search: The entered element will be searched and highlighted in the tree

6. To repeat any of the features repeat steps 4 and 5 accordingly

7. Close the window for stopping the execution of project

The above steps are shown in the form of a flowchart as shown in Fig 4.2.1

4.2 Flowchart

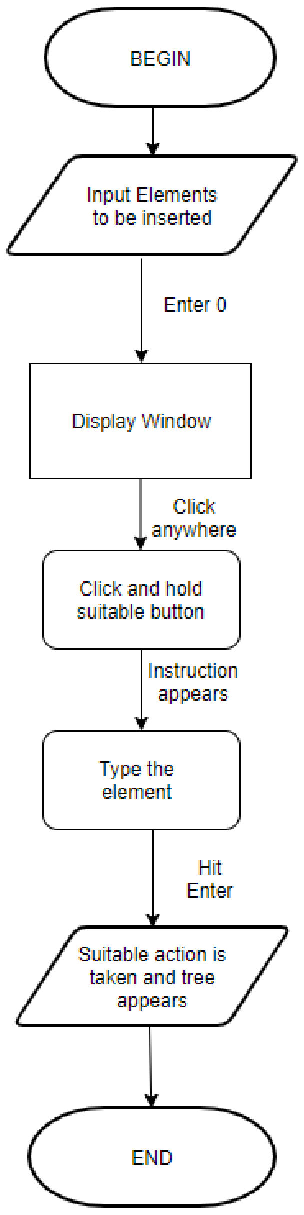

BEGIN
Input Elements to be inserted
Enter 0
Display Window
Click anywhere
Click and hold suitable button
Instruction appears
Type the element
Hit Enter
Suitable action is taken and tree appears
END

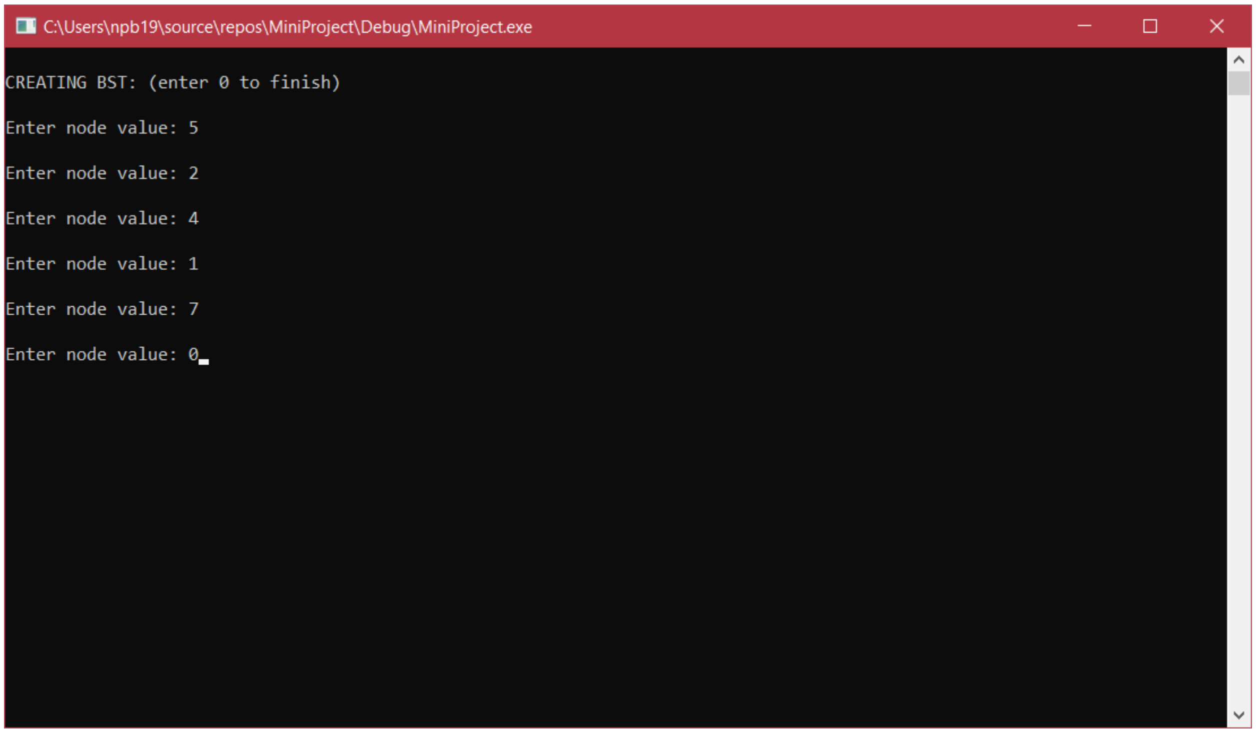
Fig 4.2.1 Flowchart

Chapter 5

RESULTS

This section includes the snapshots of the possible test cases of the project and its details.

5.1 Snapshots


C:\Users\npb19\source\repos\MiniProject\Debug\MiniProject.exe

```
CREATING BST: (enter 0 to finish)

Enter node value: 5

Enter node value: 2

Enter node value: 4

Enter node value: 1

Enter node value: 7

Enter node value: 0
```

Fig 5.1.1 Input Screen

The user can input the elements for insertion as shown in Fig 5.1.1.

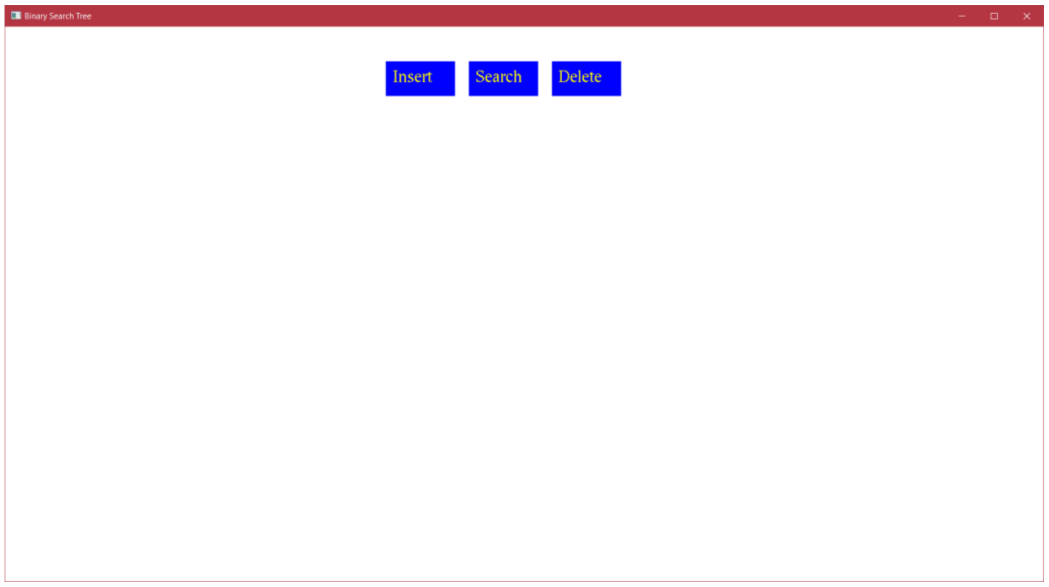


Fig 5.1.2 Output Window

The user is redirected to the window shown in Fig 5.1.2 after he enters the elements and enters 0.

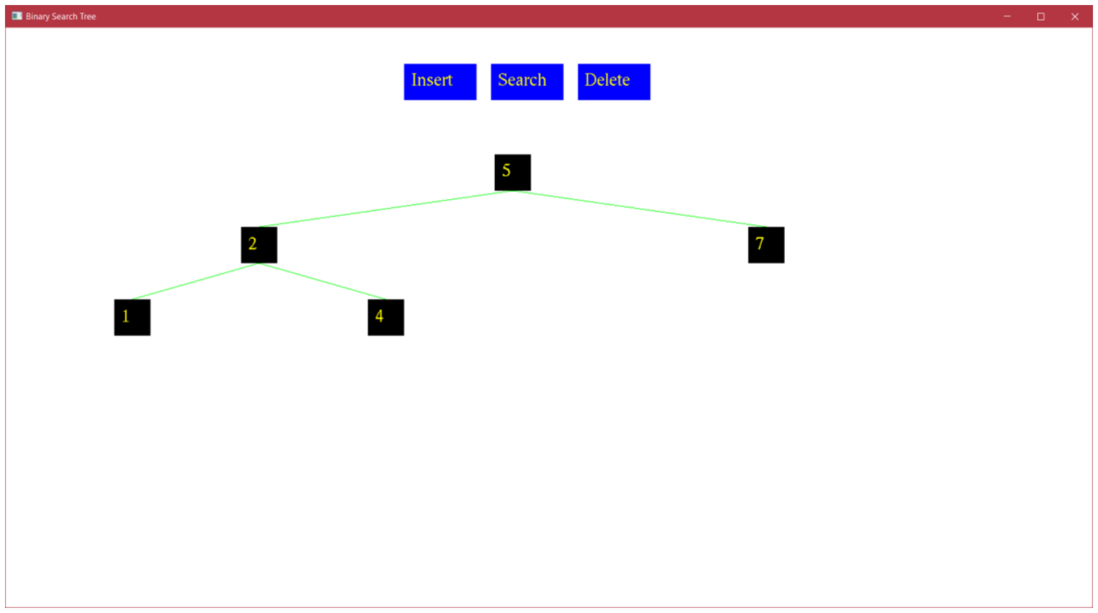


Fig 5.1.3 Display of BST

The BST is displayed as shown in Fig 5.1.3 in the output window.

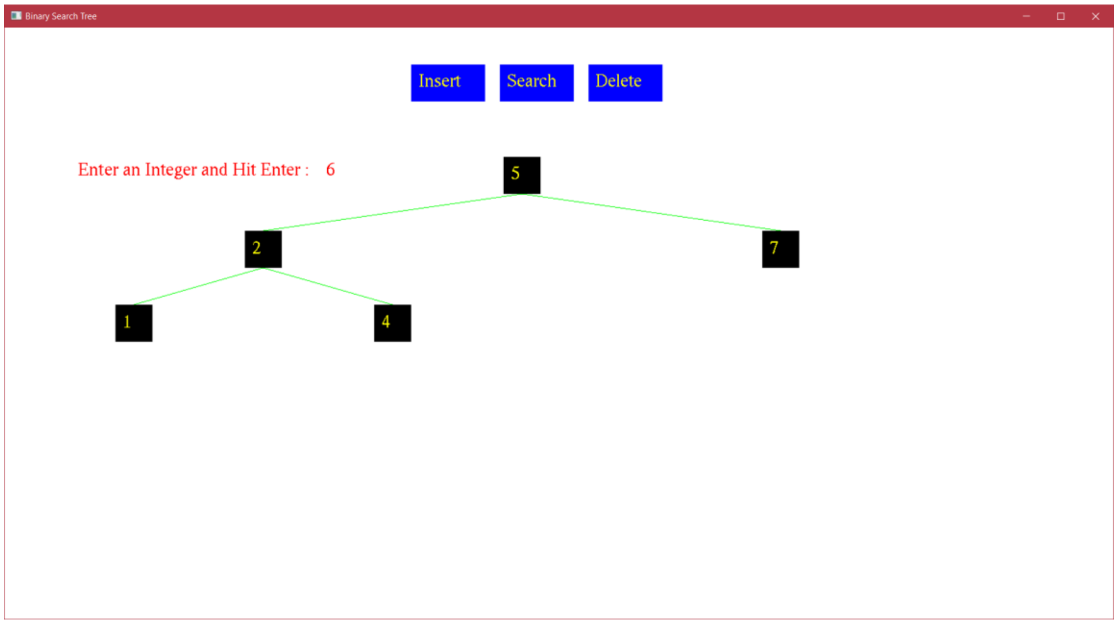


Fig 5.1.4 Prompt message for option clicked

A prompt message is shown for the user when he clicks on the options given in order to insert, delete or search an element as shown in Fig 5.1.4.

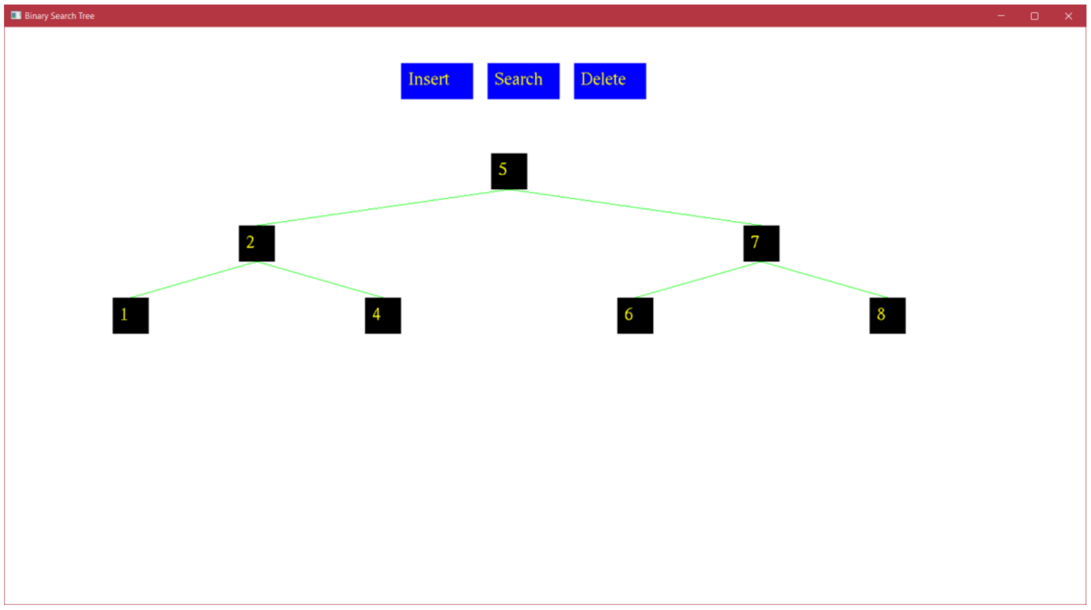


Fig 5.1.5 Insertion of a node

The insertion of nodes takes place as shown in Fig 5.1.5

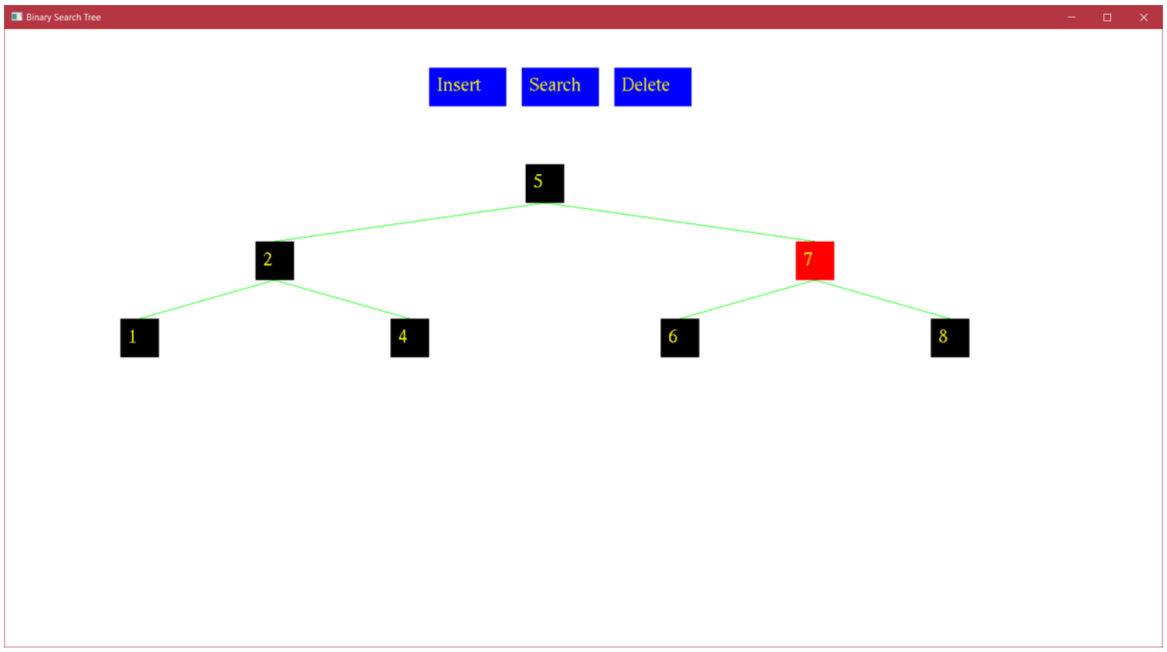


Fig 5.1.6 Searching a node

Searching of the specified node takes place as shown in Fig 5.1.6

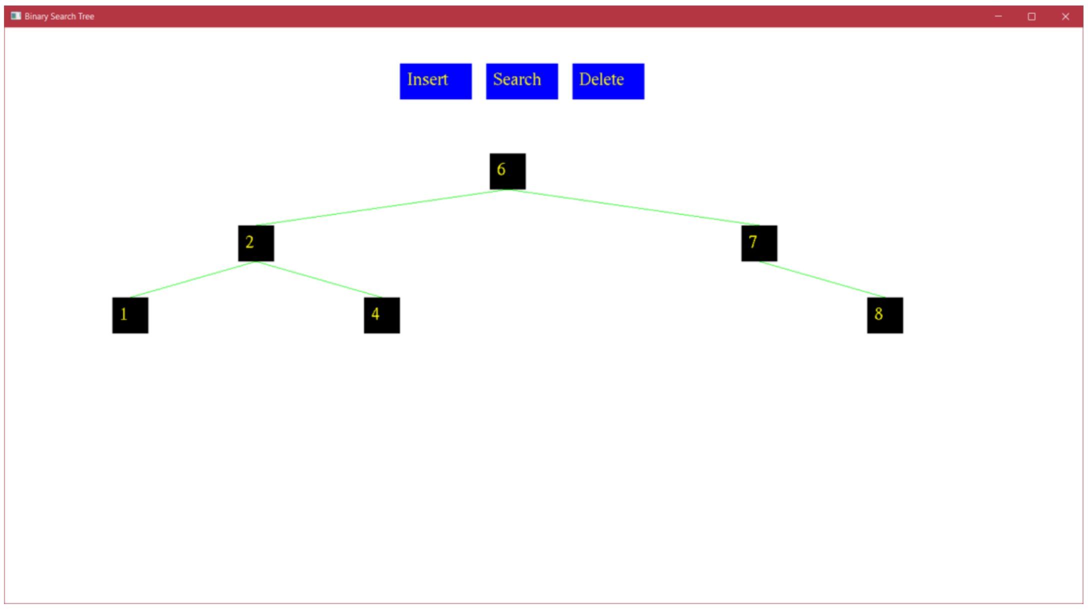
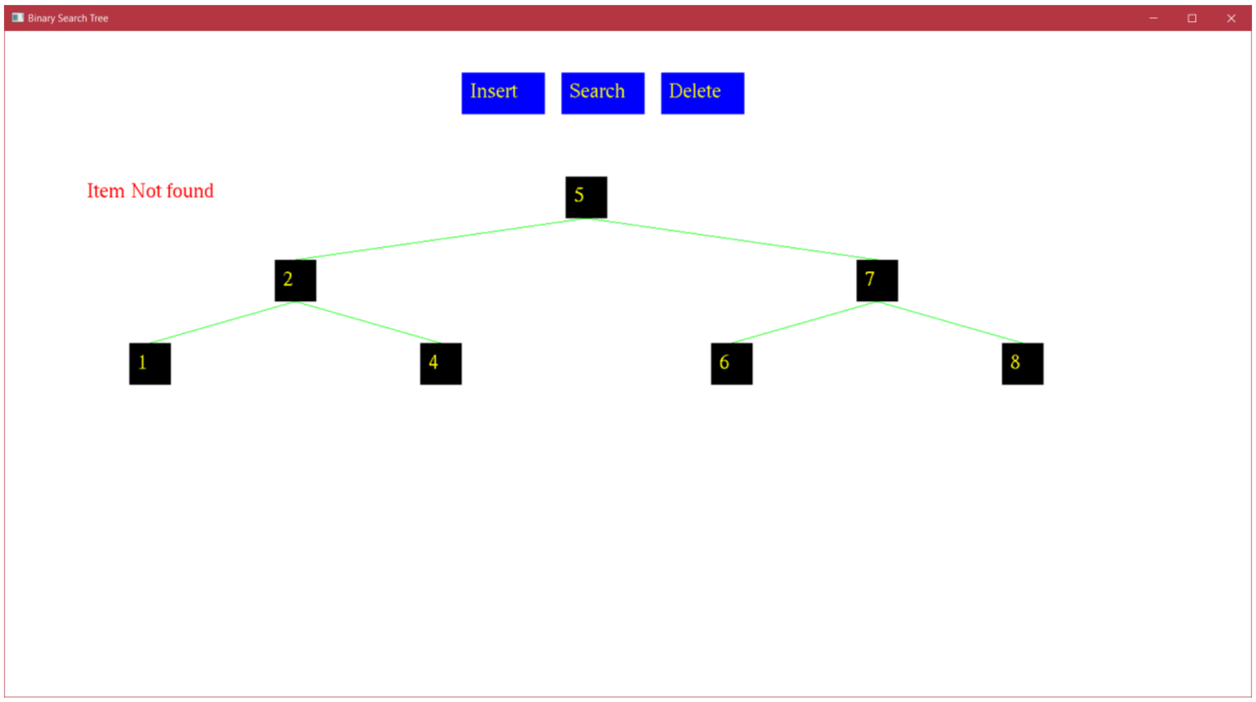


Fig 5.1.7 Deletion of a node

Deletion of the specified node takes place as shown in Fig 5.1.7



Item Not found

Fig 5.1.8 Prompt message for invalid item

The prompt message is displayed as shown in Fig 5.1.8 when the user searches or deletes a node which is not present in the BST.

Chapter 6

CONCLUSION

The BINARY SEARCH TREE Graphics package has been developed Using OpenGL. The illustration of graphical principles and OpenGL features are included and application program is efficiently developed. The aim in developing this program was to design a simple program using Open GL application software by applying the skills we learnt in class, and in doing so, to understand the algorithms and the techniques underlying interactive graphics better. The designed program will incorporate all the basic properties that a simple program must possess. The program is user friendly as the only skill required in executing this program is the knowledge of graphics. The main idea of the program is to visualize the features of Binary Search Tree effectively through computer graphics.

REFERENCES

[1] R. Chinnaiyan and A. Kumar, "Construction of estimated level based balanced binary search tree," 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, 2017, pp. 344-348

[2] G. Reina, T. Müller and T. Ertl, "Incorporating Modern OpenGL into Computer Graphics Education," in IEEE Computer Graphics and Applications, vol. 34, no. 4, pp. 16-21, July-Aug. 2014.

[3] Inayat-ur-Rehman, S. Khan and M. S. H. Khayal, "A Survey on Maintaining Binary Search Tree in Optimal Shape," 2009 International Conference on Information Management and Engineering, Kuala Lumpur, 2009, pp. 365-369.

[4] V. Borovskiy, J. Müller, M. Schapranow and A. Zeier, "Binary search tree visualization algorithm," 2009 16th International Conference on Industrial Engineering and Engineering Management, Beijing, 2009, pp. 108-112.

[5] B. K. Shrivastava, G. Khataniar and D. Goswami, "Binary Search Tree: An Efficient Overlay Structure to Support Range Query," 27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07), Toronto, Ont., 2007, pp. 77-77.

[6] P. Vinod, S. Pushpa and C. Maple, "Maintaining a Random Binary Search Tree Dynamically," Tenth International Conference on Information Visualisation (IV'06), London, England, 2006, pp. 483-488.

[7] Limin Xiang, Kazuo Ushijiam, Jianjun Zhao, Tianqing Zhang and Changjie Tang, "O(1) time algorithm on BSR for constructing a random binary search tree," Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, Chengdu, China, 2003, pp. 599-602.

[8] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd / 4th Edition, Pearson Education,2011

[9] Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL,5th edition. Pearson Education, 2008