**1. INTRODUCTION**

**1.1 Computer graphics introduction**

Computer graphics is subfield of computer science and is concerned with digitally synthesizing and manipulating visual content, although the term refers to three-dimensional. Computer graphics, it also encompasses two-dimensional graphics and image processing. Computer graphics is often differentiated from the field of visualization, although two have same similarities. Graphs are visual presentation on some surface like wall, canvas, computer screen. Graphics often combine text, illustration and color.

Computer graphics started with a display of data on hardcopy plotters and cathode ray tube (CRT) screen soon after the introduction of computers. It has grown to include the creation, storage and manipulation of models and image of objects. These models come from a diverse and expanding set of fields, and include physical, engineering, architectural, and even conceptual structures. Computer graphics today is largely interactive. The user controls the contents, structure and appearance of objects and of their displayed images by using inputs input devices, such as keyboard, mouse, or touch-sensitive panel on screen.

Graphical interfaces have replaced textual interfaces as the standard means of user-computer interaction. Graphics has also become a key technology for communicating ideas, data and trends in most areas of commerce, science, engineering and education. Much of the task of creating effective graphic communication lies in modeling the objects whose images we want to reproduce.

**1.2 Aim**

Tree traversal(also known as tree search) is a form of graph traversal and refers to the process of visiting(checking and/or updating) each node in a tree data structure,exactly once. Such traversals are classified in the order in which the nodes are visited. There are three types of traversals depending upon the order of tracing namely inorder, preorder and postorder.

**1.3 Motivation**

* Nodes and the connections form a tree.
* A tree whose elements have at most two children is called a binary tree.
* Given a binary tree with set of nodes and connections between them, find the order of traversal based on the prescribed order of tracing in inorder, preorder and postorder traversals.

**1.4 Layout of project**

The rest of this report is organized as follows .Chapter 1 provides an introduction on Binary tree traversal Chapter 2 looks at analysis and design part of Binary tree traversals. Chapter 3 covers the hardware and software requirements of openGL. Chapter 4 goes through the steps needed to develop computer graphics applications. Chapter 5 is about the results and snapshots of the project model. Chapter 6 is the conclusion. Chapter 7 contains case studies contributed by members of the computer graphics contribution.

**2. ANALYSIS AND DESIGN**

**2.1 Analysis**

Circles are used to represent the nodes .Midpoint circle drawing algorithm is used to draw the circles with specified radius. Bitmap character function is used to display the characters. Numbers from 1 to 7 are used to label the nodes. Lines are used to represent the relationship between the nodes .Different modes of traversals namely inorder, preorder and postorder are implemented using two methods of demonstration. First mode of demonstration being the interactive mode ,traces the path of traversal on pressing number keys from 1 to 7.Keyboard functions are used to demonstrate this.The line changes the color to blue and the nodes change color to green on being visited.Second mode of demonstration is through animation where a point is moved based on the order of traversal.

In each mode of demonstration, order of traversal being left -> root -> right for inorder traversal, root -> left -> right for preorder traversal and left -> right -> root for postorder traversal are printed. Special functions are used to go and forth from windows.

**2.2 Design**

START

HOME PAGE

DESCRIPTION

Press next Press next

ANIMATION

KEYPRESS

MODE OF TRAVERSAL

MODE OF TRAVERSAL

EXIT

POSTORDER

PREORDER

INORDER

INORDER

PREORDER

POSTORDER

Fig 2.2:Flow diagram of project

**3. SOFTWARE AND HARDWARE REQUIREMENTS**

**3.1 Hardware Requirements**

The minimum/recommended hardware configure required for developing the proposed software is given below:

* PENTIUM-2 and compatible system.
* 512 RAM
* Approximately 170 MB free space in the hard disk.
* Hard disk access time must be less than 19 milliseconds.

**3.2 Software Requirements**

* OpenGL
* WINDOWS XP/ME/9X
* Microsoft Visual Studio
* Linux(Ubuntu 14.0)
* gcc compiler

**3.2.1 Dev c++**

Dev c++ is a free full-featured integrated development environment (IDE) distributed under the GNU general public license for programming in c and c++, it is written in Delphi.It is bundled with and uses, the minGW or TDM\_GCC 64bit port of the GCC as its compiler. Dev c++ can also be used in combination with Cygwin or any other GCC based compiler.Dev c++ is generally considered windows only program ,but there are attempts to create a Linux version. Header files and path delimiters are switchable between platforms.

**3.2.1.1 The OpenGL Utility Library (GLU).**

It consists of a number of functions that use the base OpenGL library to provide higher-level drawing routines from the more primitive routines that OpenGL provides .It is usually distributed with the base OpenGL package. GLU is not implemented in the embedded version of the OpenGL package. OpenGL package OpenGL ES.

Among these features are mapping between screen and world coordinates, generation of texture mipmaps, drawing of quadratic surfaces, NURBS, tessellation of polygonal primitives, interpretation of OpenGL error codes, an extended range of transformation routines for setting up viewing volumes and simple positioning of the camera, generally in more human friendly terms than the routines presented by OpenGL. It also provides additional primitives for use in OpenGL applications, including spheres, cylinders and disks.

All GLU functions start with the glut prefix. An example function is gluortho2D which defines a two dimensional orthographic projection matrix. The GLU specification was last updated in 1998,and it depends on features which were deprecated with the release of OpenGL 3.1 in 2009.Specifications for GLU are still available.

* Write a 16 bit DOS program and use assembly language for graphics.
* Make calls to the window API
* Make calls to the OpenGL subsystem
* Use a graphics library

**3.2.1.2 OpenGL Subsystem.**

OpenGL provides a powerful but primitive set of rendering commands , and all higher-level drawing must be done in terms of these commands. Also OpenGL programs have to use the underlying mechanisms of the windowing system. A number of libraries exist to simplify the programming tasks, including the following.

* The OpenGL Utility Library (GLU) contains several routines that use lower level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections, performing polygon tessellation, and rendering surfaces. The library is provided as part of every OpenGL implementation.
* The OpenGL Utility Toolkit (GLUT) is a window system-independent toolkit. GLUT routines use the prefix glut.

**4.IMPLEMENTATION**

**4.1 FUNCTIONS USED**

**1. glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);**

This function specifies whether to use a RGB or color-index color(if you are working in color-index mode, you want to load certain colors into the color map use glutSetColor() to do this)model, you can also specify whether you want a SINGLE or DOUBLE buffered window. Finally you can use this routine to indicate that you want the window to have am associated depth, stencil and/or accumulation buffer. For example if you want the window with double buffering, RGB color mode and DEPTH buffer you might call.

**2**. **glutBitmapCharacter(font,string[size]) ;**

It renders the character in the named bitmap font and advances the current raster position.

**3. glutInitWindowSize(1356,695 );**

This command specifies the initial height and width of the window in pixels.

**4. glutInitWindowPosition(0,0);**

This command is used to set the position of the display window with respect to upper left corner.

**5.** **glutCreateWindow("BINARY TREE");**

This command is used to create a display window with the name passed as parameter enclosed in double inverted commas.

**6. glutMouseFunc(draw);**

This command is used to invoke draw function when specified mouse button is pressed.

**7. glutDisplayFunc(display);**

This command is used to assign the created window on the screen.

**8. glutKeyboardFunc(keypress);**

This command is used to invoke function keypress when any key is pressed from the keyboard.

**9. glutSpecialFunc(fkeys);**

This command is used to invoke function fkeys when special keys like F1 F2.. are pressed from the keyboard.

**10.glutMainLoop();**

This command is used to retain the picture for certain time. It makes the program to enter into infinite loop.

**4.2 Modules**

**4.2.1Function used to draw binary tree**

void in()

{

fun1();

fun2();

fun3();

fun4();

fun5();

fun6();

fun7();

glColor3f(0,206,209);

glLineWidth(3);

glBegin(GL\_LINES);

glVertex2i(350,575);

glVertex2i(220,525);

glVertex2i(220,475);

glVertex2i(120,425);

glVertex2i(220,475);

glVertex2i(320,425);

glVertex2i(350,575);

glVertex2i(480,525);

glVertex2i(480,475);

glVertex2i(380,425);

glVertex2i(480,475);

glVertex2i(580,425);

glEnd();

}

**4.2.2 Function to move the point**

int px,py;

int qx=0,qy=0;

void translate()

{

in();

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,0,0);

glPointSize(8);

glBegin(GL\_POINTS);

glVertex2i(px+qx,py+qy);

glEnd();

in();

glFlush();

Sleep(100);

}

**4.2.3 Function to trace path in inorder using animation.**

void animinorder()

{

px=0;

py=0;

qx=0;

qy=0;

{

glFlush();

in();

px=120,py=425;

if(px<220&&py<475)

{

for(i=1;i<12;i++)

{

translate();

qx=qx+8;

qy=qy+5;

}

}

if(px>220&&py>475)

px=220,py=475;

for(i=1;i<10;i++)

{

translate();

qx=qx+12;

qy=qy-5;

}

if(px>320&&py>425)

px=320,py=425;

for(i=1;i<21;i++)

{

translate();

qx=qx+1;

qy=qy+7;

}

if(px>320&&py>425)

px=375,py=575;

for(i=1;i<27;i++)

{

translate();

qx=qx+1;

qy=qy-6;

}

if(px>375&&py>430)

px=375,py=425;

for(i=1;i<10;i++)

{

translate();

qx=qx+12;

qy=qy+7;

}

if(px>580&&py>425)

px=580,py=425;

for(i=1;i<18;i++)

{

translate();

qx=qx+7;

qy=qy-5;

}

glFlush();

in1();

glutSwapBuffers();

glColor3f(1,1,0);

x=200;

y=200;

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,1);

x=201,y=670;

glColor3f(1,0,0);

for(m=0;m<30;m++)

{

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,inorder1[m]);

x=x+20;

}

x=180,y=150;

glColor3f(1,0,0);

for(m=0;m<30;m++)

{

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,inorder[m]);

x=x+20;

}

llinne();

details2();

px=350,py=575;

}

glFlush();

}

**4.2.4 Function to show order of traversal of inorder using keypress**

void keypress(GLubyte k,GLint x,GLint y)

{x=201,y=670;

if(flag==1)

{

glColor3f(1,0,0);

for(m=0;m<30;m++)

{

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,inorder1[m]);

x=x+20;

}

details();

switch(k)

{

case '1':in();

fu3();

int x=200,y=180;

glColor3f(1,1,0);

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'3');

x=x+10;

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'>');

llinne();

x=180;

y=230;

glColor3f(1,0,0);

for(m=0;m<30;m++)

{

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15,order[m]);

x=x+20;

}

glLineWidth(3);

glColor3f(1,1,0);

glBegin(GL\_LINES);

glVertex2i(180,220);

glVertex2i(535,220);

glEnd();

break;

case '2': fu2();

x=225,y=180;

glColor3f(1,1,0);

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'4');

x=x+10;

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'>');

llinne();

break;

case '3': fu4();

x=250,y=180;

glColor3f(1,1,0);

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'5');

x=x+10;

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'>');

llinne();

break;

case '4': fu1();

x=275,y=180;

glColor3f(1,1,0);

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'6');

x=x+10;

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'>');

llinne();

break;

case '5': fu5();

x=300,y=180;

glColor3f(1,1,0);

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'7');

x=x+10;

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'>');

llinne();

break;

case '6': fu7();

x=325,y=180;

glColor3f(1,1,0);

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'8');

x=x+10;

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'>');

llinne();

break;

case '7': fu6();

x=350,y=180;

glColor3f(1,1,0);

glRasterPos2i(x,y);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'9');

llinne();

break;

case'0':

glClear(GL\_COLOR\_BUFFER\_BIT);

glutPostRedisplay();

break;

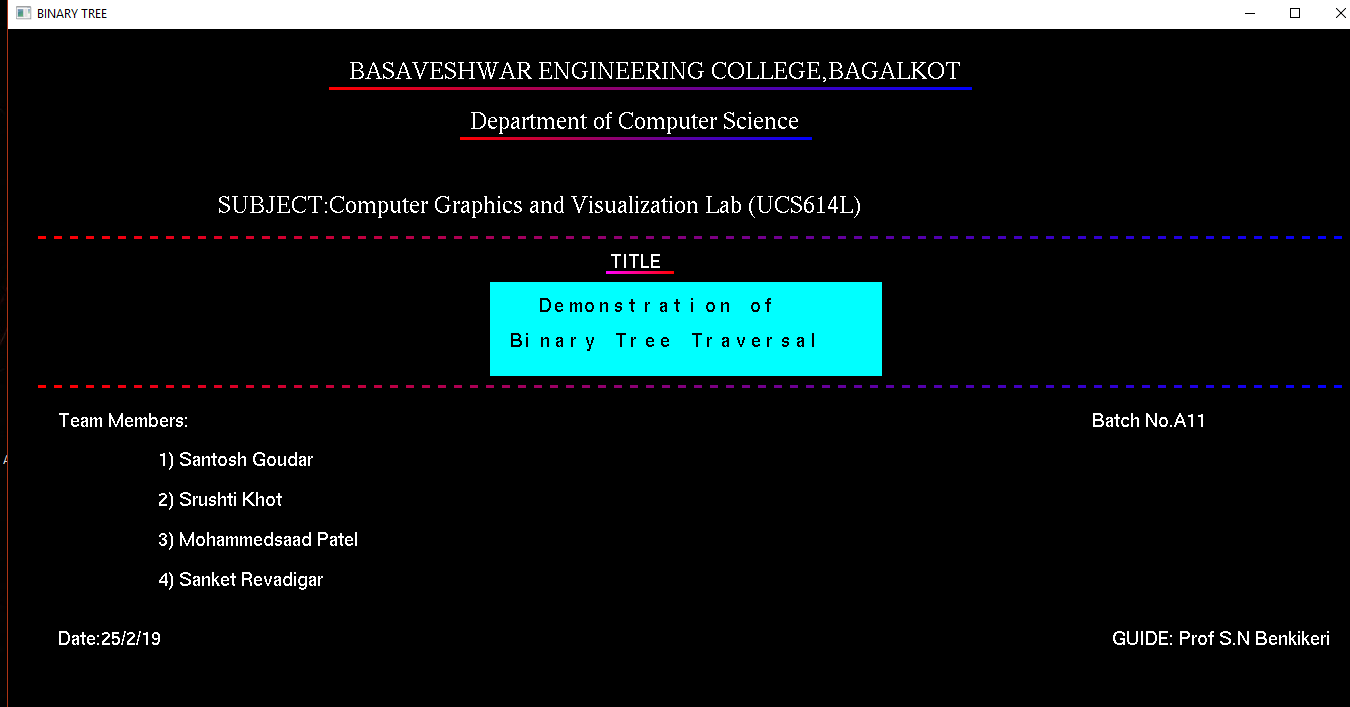
}

}

**5. RESULT AND SNAPSHOTS**

**1. Home Page**

This page contains title of the project and names of the group members.

****

Press on the blue box to move to description.

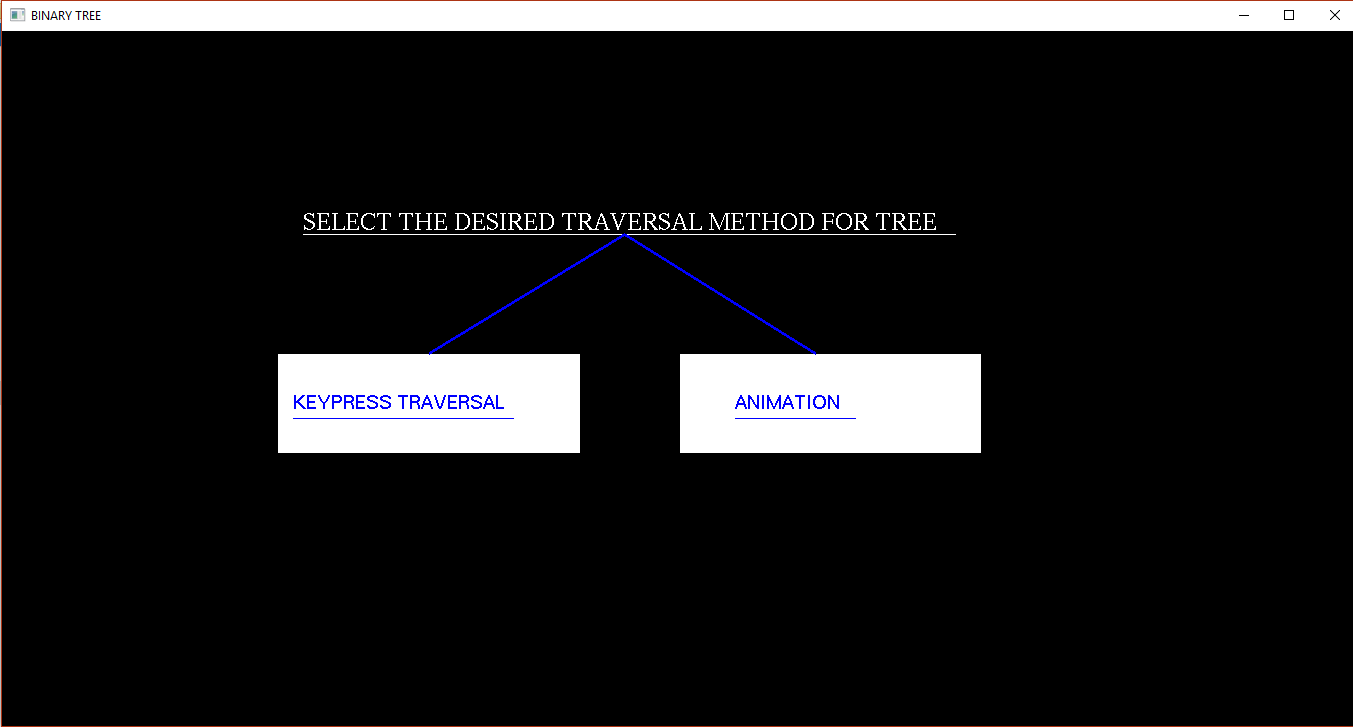
**2. Description**



Press on next to move to select mode of demonstration. Either through key press or through animation.

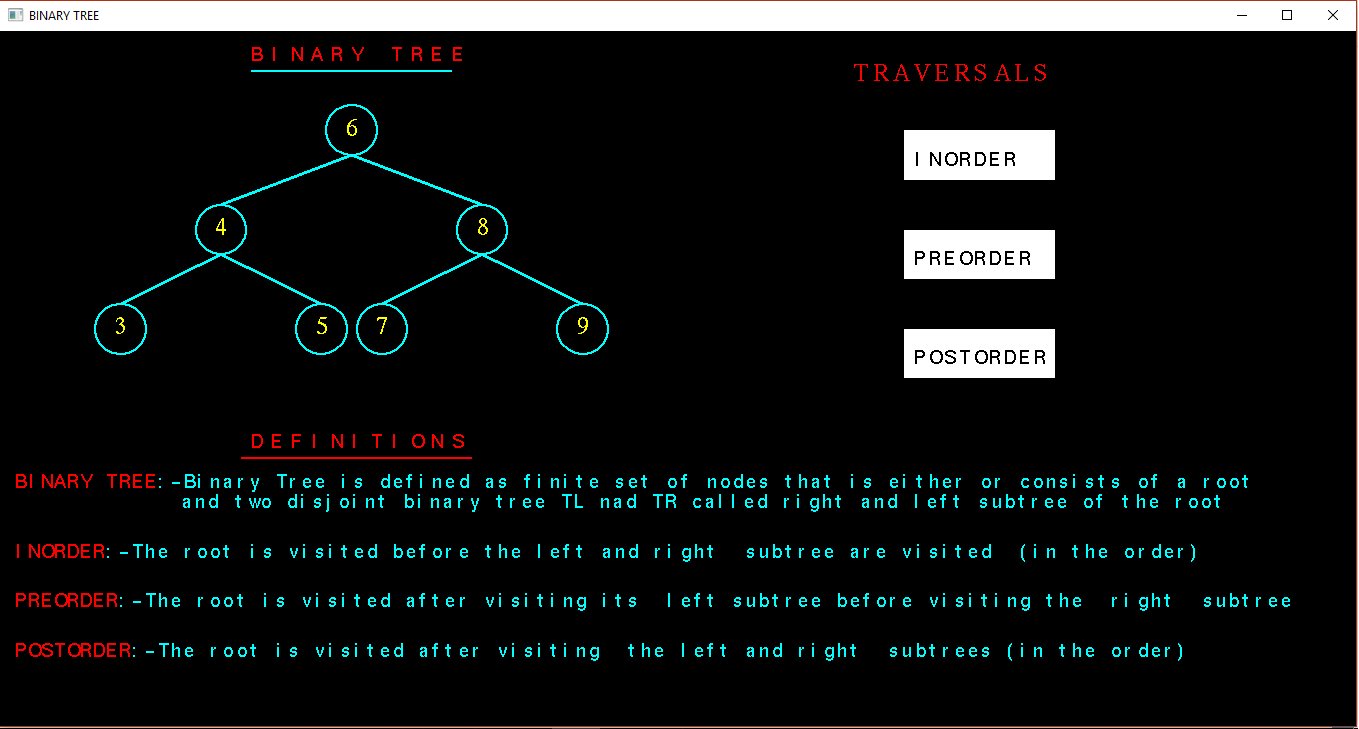
**3. Mode of Demonstration**

This window allows user to select mode of demonstration.



Select your choice.

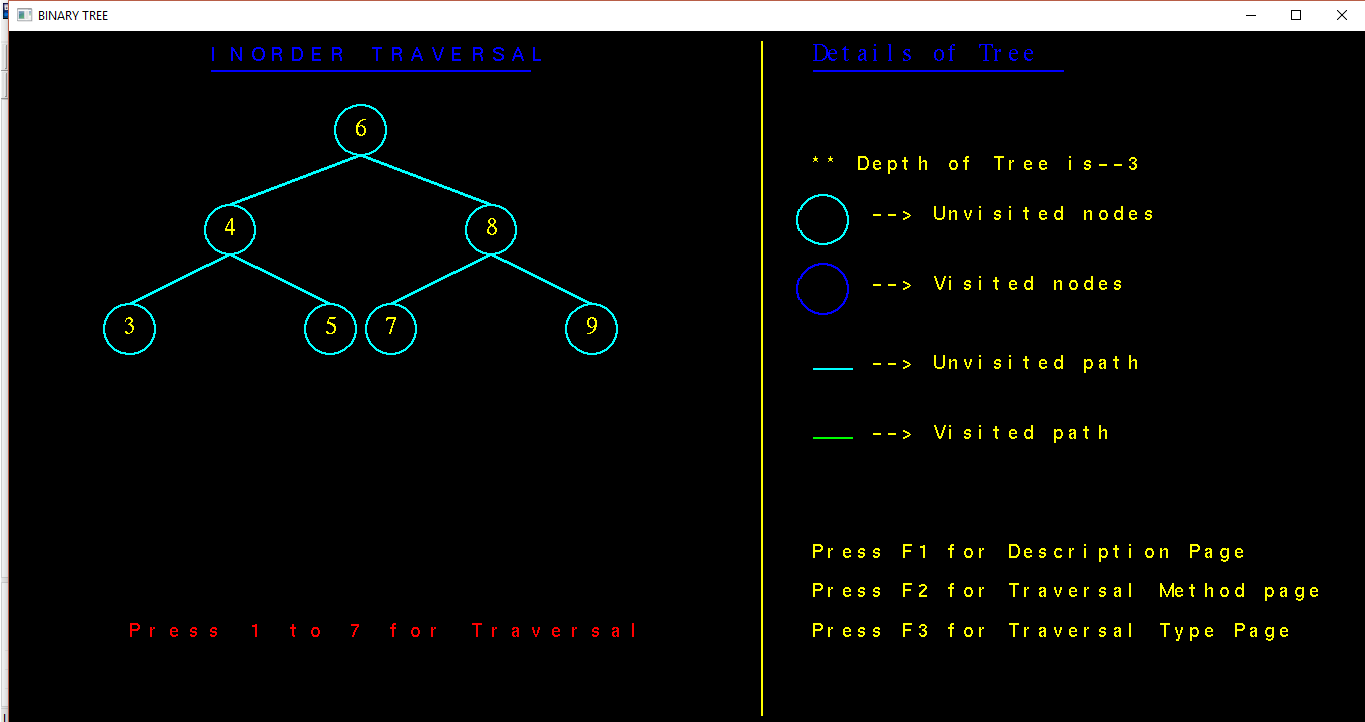
**4. Keypress traversal**



Select the type of traversal. Either Inorder, Preorder or Postorder.

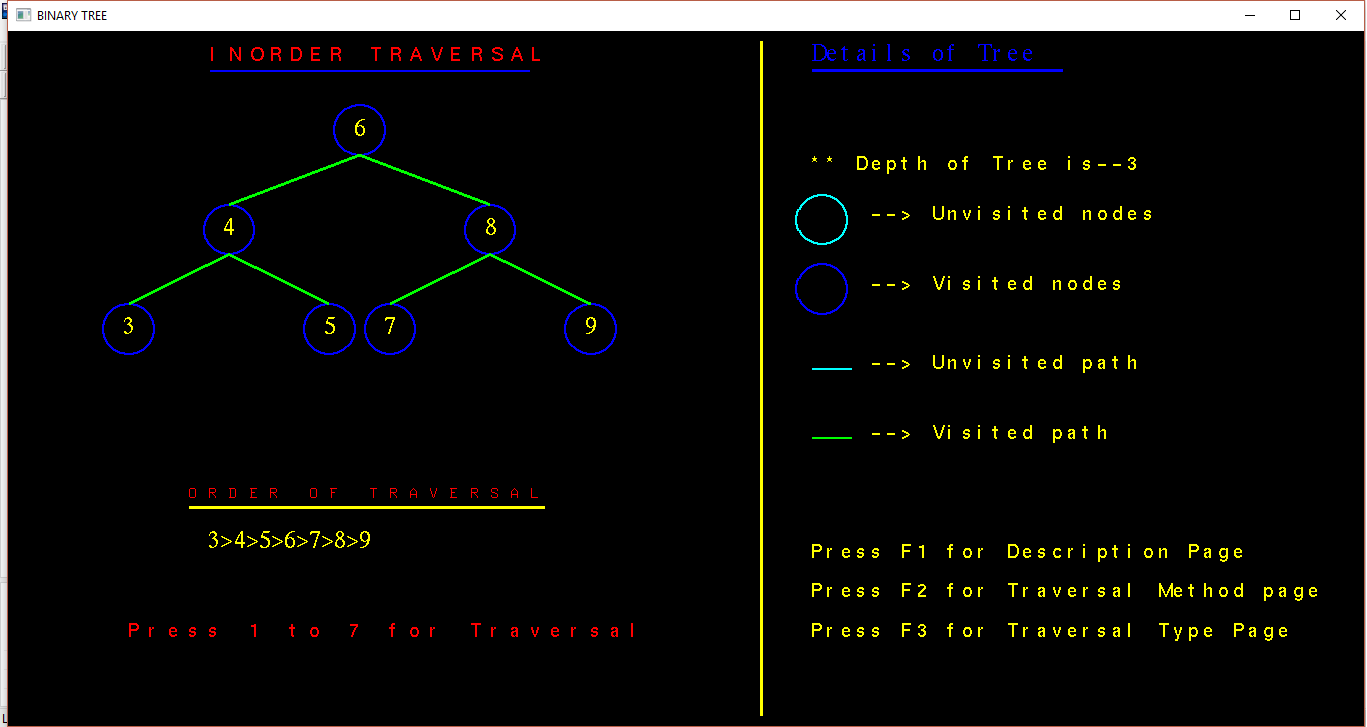
**4.1 Inorder (using keypress)**

**4.1.1 Before traversal**

****

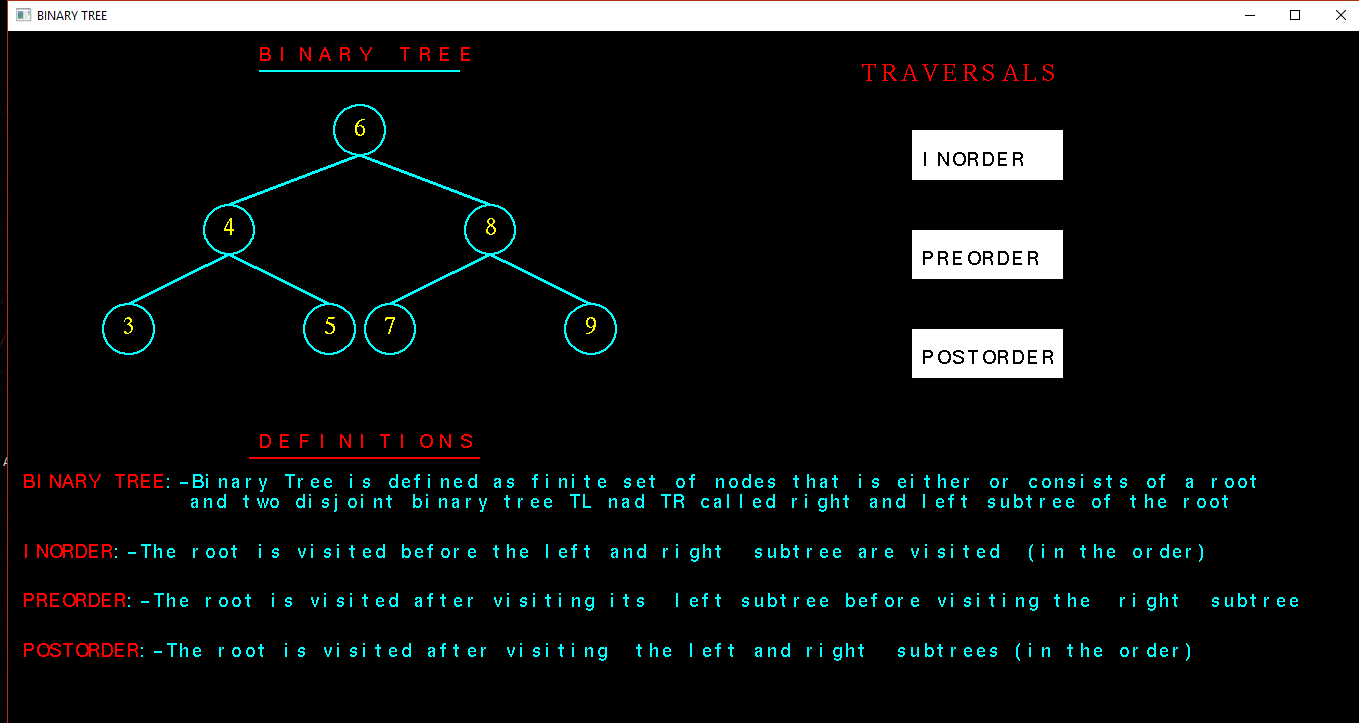
Demonstration of Inorder traversal. This appears before the traversal is done. Order of traversal is displayed by pressing numbers from 1-7.

**4.1.2 After traversal**

****

This snapshot demonstrates the traversed path and displays the order of traversal.

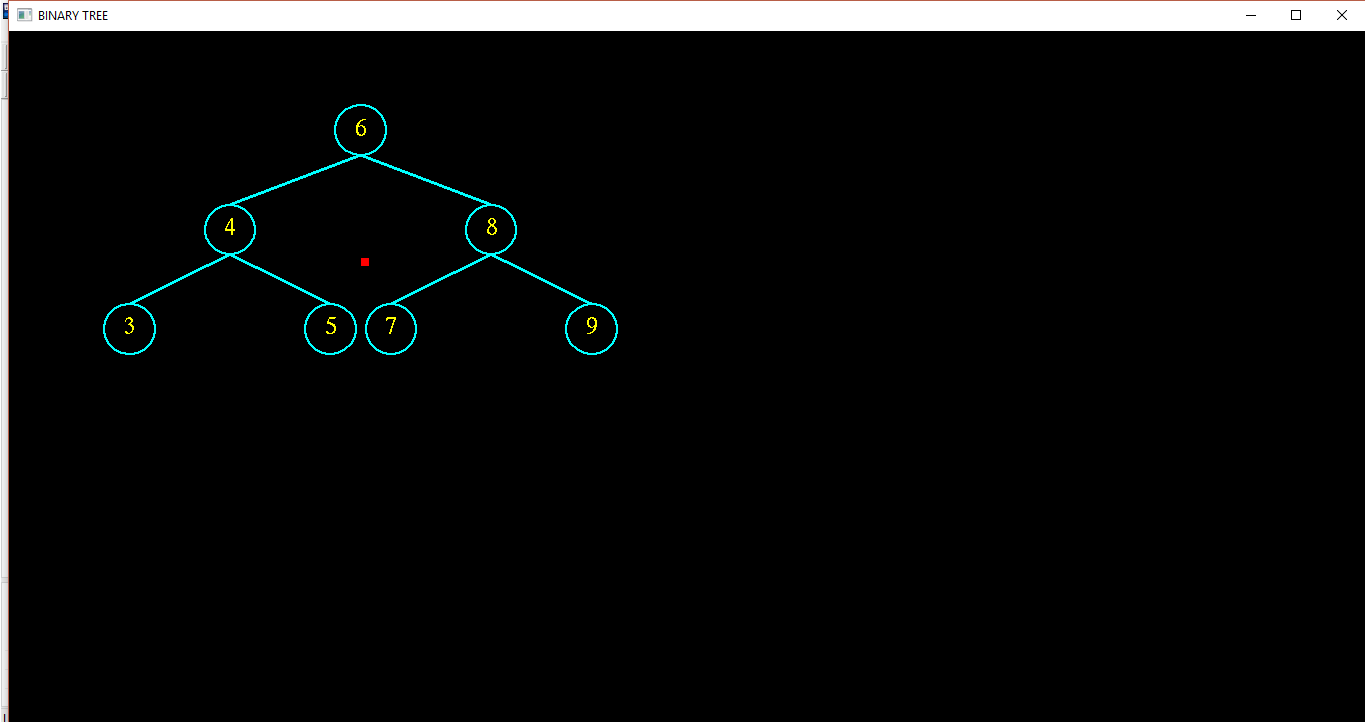
**5. Animation**



Select which type of traversal to be viewed from the options provided by clicking on it with left mouse button.

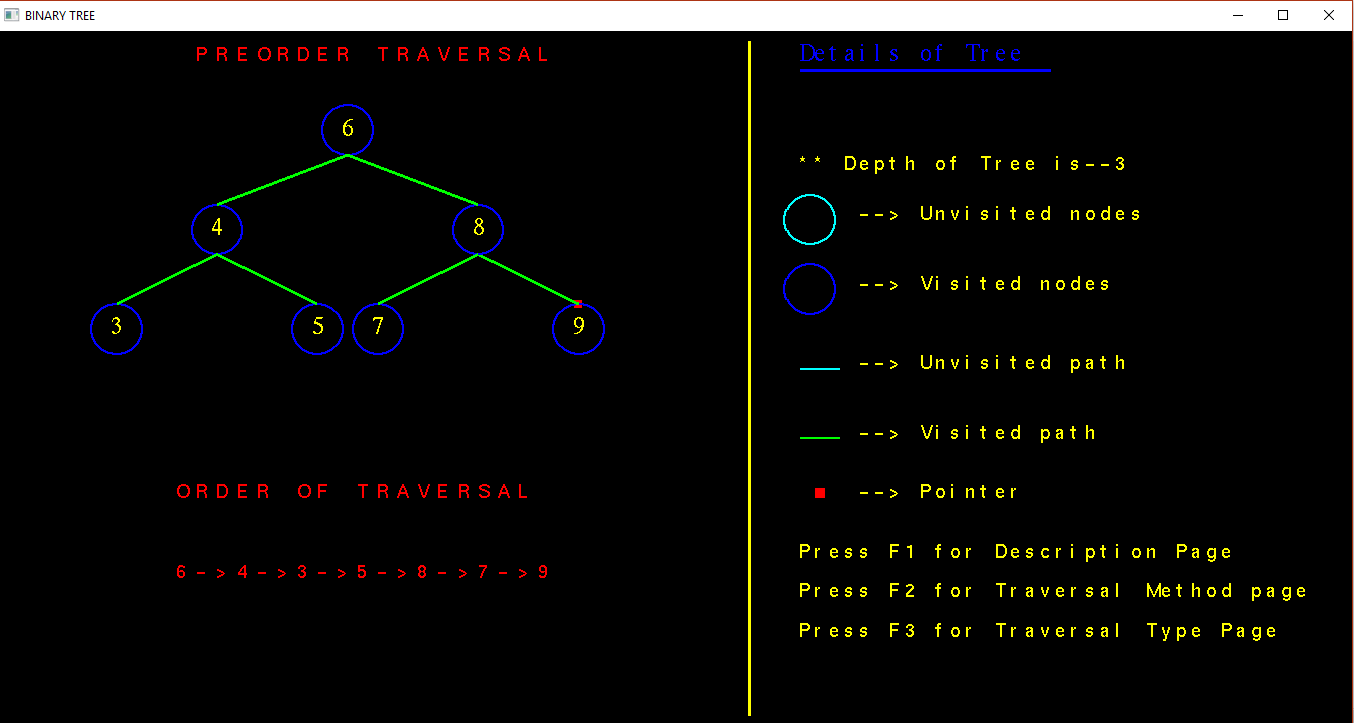
**5.1 Inorder**

**5.1.1 During traversal**

****

The order in which the path is traversed is thrown moving the box from one node to another node.In order the order od tracing is left->root->right.

**4.1.2 After traversal**

****

After traversal all the nodes and paths change their color

**6. CONCLUSION AND FUTURE SCOPE:**

Designing and implementing project in graphics is a great experience. We understood analyzed about the concept of OpenGL which is very useful indeveloping any graphics application.

Tree-Traversal technique is an animated demonstration usually used by student. By this project student can understand Tree-Traversal technique easily.

This application is like open source where anyone can design and add his own codes. Even more features and good graphics can be add to this application to make it even more interesting and provide good graphical user interface. Developing this application incompatible with mobile devices will provide even. More compatible and easy use. Designing this type of application is very interesting.

**REFERENCES**

[1].Ariponnammal, S. and Natarajan, S. (1994) ‘Transport Phonomena of Sm Sel – X Asx’, Pramana – Journal of Physics Vol.42, No.1, pp.421-425.

[2]. Barnard, R.W. and Kellogg, C. (1980) ‘Applications of Convolution Operators to Problems in Univalent Function Theory’, Michigan Mach, J., Vol.27, pp.81–94.

[3]. Shin, K.G. and Mckay, N.D. (1984) ‘Open Loop Minimum Time Control of Mechanical Manipulations and its Applications’, Proc.Amer.Contr.Conf., San Diego, CA, pp. 1231-1236.

[4]. Computer graphics 4th edition Donald Hearn,M.Pauline Baker.

[5]. [www.opengl.org](http://www.opengl.org).