VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590018

A Mini-Project Report
On
"TIC TAC TOE"

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE COMPUTER GRAPHICS LABORATORY WITH MINI PROJECT (18CSL67) COURSE OF VI SEMESTER

Submitted by

RAKSHITHA SHANKAR
(1CG19CS090)
NITHYASHREE C
(1CG19CS082)

Guide:
Mr. Kotresh Naik D M.Tech,
Asst.Prof, Dept. of CSE,
CIT, Gubbi.

HOD:
Dr. Shantala C P Ph.D
Vice Principal & Head,
Dept. of CSE,
CIT, Gubbi.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(NAAC Accredited & ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

2021-22

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(NAAC Accredited & ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2021-22

CERTIFICATE

This is to certify that the project entitled **"TIC TAC TOE"** has been successfully carried out by **RAKSHITHA SHANKAR [1CG19CS090]** & **NITHYASHREE C [1CG19CS082]** partial fulfillment for the VI semester during the academic year **2021-21** It is certified that all the corrections / suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the VI semester.

Signature of guide

Mr. Kotresh Naik D M.Tech
Asst.Prof, Dept. of CSE,
CIT, Gubbi.

Signature of HOD

Dr. Shantala C P Ph.D
Vice Principal & Head,
Dept. of CSE,
CIT, Gubbi.

Signature of Principal

Dr. Suresh D S
Principal
CIT, Gubbi.

External Viva

Name of Examiners	**Signature with date**
1. _____	_____
2. _____	_____

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(**NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2021-22

DECLARATION

We, **NITHYASHREE C & RAKSHITHA SHANKAR** student of VI Semester, **BE.,** in Computer Science and Engineering**, C.I.T, Gubbi**, hereby declare that the dissertation work entitled **"TIC TAC TOE"**, embodies the report of our project work carried out independently by us under the guidance of **Mr. Kotresh Naik D**, Assistant Professor, Department CSE, CIT, Gubbi, as partial fulfilment of requirements for the VI Semester during the academic year **2021-22**.We further declare that the project has not been submitted for the award of any other degree.

Place: GUBBI
Date:

RAKSHITHA SHANKAR
USN:1CG19CS090

NITHYASHREE C
USN: 1CG19CS082

ABSTRACT

Computer Graphics has core importance in the development of games. We have seen so many game on this site. Today also we are going to see the interesting loving game. It is simple, easy to play as well quite easy to code it, present to you the **Egg Game Computer graphics with OpenGL.** As name suggest, game is related to egg. We are going to discuss the whole things about it in details.

Tic Tac Toe. Tic-tac-toe also known as noughts and crosses is a paper and pencil game for two players, who take turns marking the spaces in a 3 x 3 grid traditionally. The player who succeeds in placing three of their marks in a horizontal, vertical or diagonal row wins the game .The optimal move for this game can be gained by using minimax algorithm, where the opposition between the utility functions makes the situation adversarial, hence requiring adversarial search supported by minimax algorithm with alpha beta pruning concept in artificial intelligence.

ACKNOWLEDGEMENT

A great deal of time and lot of effort has gone into completing this project report and documenting it. The number of hours spent in getting through various books and other materials related to this topic chosen by me have reaffirmed its power and utility in doing this project.

Several special people have contributed significantly to this effort. First, we are grateful to our institution **Channabasaveshwara Institute of Technology**, Gubbi, which provides me an opportunity in fulfilling my most cherished desire of reaching the goal.

We acknowledge and express my sincere thanks to our beloved Principal and Director(tech) **Dr. Suresh D S** for his many valuable suggestions and continuous encouragement and support in the academic endeavours.

We express our sincere gratitude to **Dr.Shantala C P,** Vice Principal& Head, Department of CSE, for providing his constructive criticisms and suggestions.

Thanks to **Prof. Anil kumar G,** head of the Department of Computer Science and Engineering for his guidance on how to approach an engineering problem and come up with a solution in an organized manner.

We wish to express my deep sense of gratitude to **Kotresh Naik D,** Department of Computer Science and Engineering for all the guidance and who still remains a constant driving force and motivated through innovative ideas with tireless support and advice during the course of project to examine and helpful suggestions offered, which has contributed immeasurably to the quality of the final report.

Project Associate :

NITHYASHREE C [1CG19CS082]

RAKSHITHA SHANKAR[1CG19CS090]

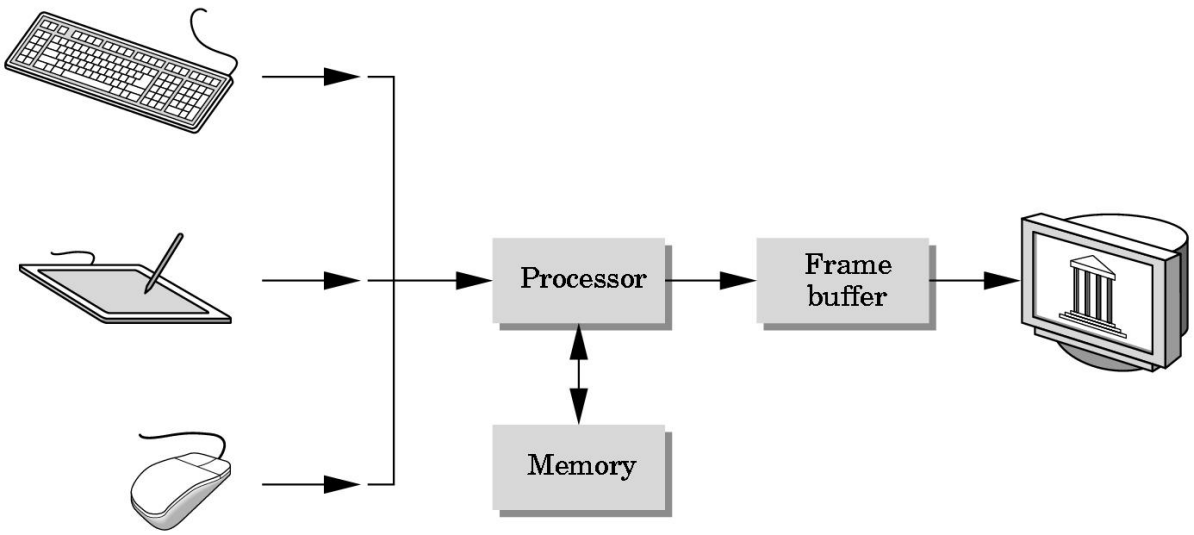
CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics :

- Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D Or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly.
- Computers have become a powerful medium for the rapid and economical production of pictures.
- There is virtually no area in which graphical displays cannot be used to some advantage.
- Graphics provide a so natural means of communicating with the computer thatthey have become widespread.
- Interactive graphics is the most important means of producing pictures since the invention of photography and television .
- We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.



CHAPTER 2

HISTORICAL REVIEW

2.1 History of Computer Graphics:

Computer graphics deals with generating images with the aid of computers. Today, computer graphics is a core technology in digital photography, film, video games, cell phone and computer display, and many specialized applications.

The first cathode ray tube, the Braun tube, was invented in 1897 – it in turn would permit the oscilloscope and the military control panel – the more direct precursors of the field, as they provided the first two-dimensional electronic displays that responded to programmatic or user input. New kinds of displays were needed to process the wealth of information resulting from such projects, leading to the development of computer graphics as a discipline.

In 1996, Krishnamurthy and Levoy invented normal mapping – an improvement on Jim Blinn's bump mapping. By the end of the decade, computers adopted common frameworks for graphics processing such as DirectX and OpenGL. Since then, computer graphics have only become more detailed and realistic, due to more powerful graphics hardware and 3D modeling software.[2]

2.2 History of OpenGL:

By the early 1990s, Silicon Graphics (SGI) was a leader in 3D graphics for workstations. Their IRIS GL API became the industry standard, used more widely than the open standards-based PHIGS. This was because IRIS GL was considered easier to use, and because it supported immediate mode rendering. By contrast, PHIGS was considered difficult to use and outdated in functionality.

SGI's competitors (including Sun Microsystems, Hewlett-Packard and IBM) were also able to bring to market 3D hardware supported by extensions made to the PHIGS standard, which pressured SGI to open source a version of IrisGL as a public standard called **OpenGL**. However, SGI had many customers for whom the change from IrisGL to OpenGL would demand significant investment. Moreover, IrisGL had API functions that were irrelevant to 3D graphics. For example, it included a windowing, keyboard and mouse API, in part because it

was developed before the X Window System and Sun's News And, IrisGL libraries were unsuitable for opening due to licensing and patent issues. These factors required SGI to continue to support the advanced and proprietary Iris Inventor and Iris Performer programming APIs while market support for OpenGL matured.[3]

2.3 Introduction to OpenGL:

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. OpenGL is not a programming language it is an API (Application Programming Interface). The interface consists of many different function calls which can be used for building application programs. OpenGL is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation.

OpenGL is portable to many platforms (windows, MAC, UNIX, LINUX) and callable from many programming languages (C/C++, JAVA, PEARL). OpenGL is primarily concerned with modeling and rendering operations such as, to specify geometric primitives (lines, pixels, polygons…), apply geometric transformations and specify camera light, color, texture information etc.

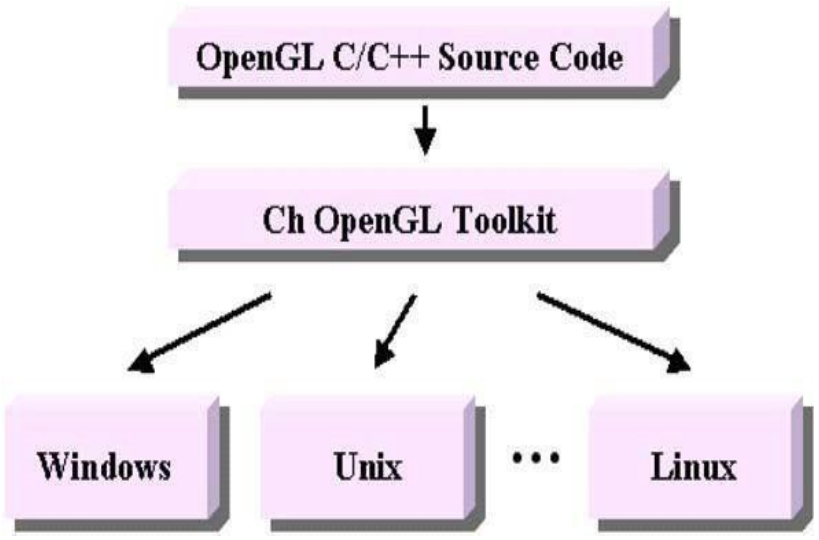


Figure no.:2.3 Displaying Compatibility of OpenGL

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 System Requirement:

The basic requirements for the development of this mini project are as follows:

3.1.1 Hardware Constraints:

- Processor: Pentium PC
- RAM: 8GB
- Hard Disk: 20GB (approx)
- Display: VGA Color Monitor

3.1.2 Software Constraints:

- Software: OpenGL 4.3 or above
- Operating System: Windows 98SE/2000/XP/Vista/UBUNTU
- Compiler: Eclipse/Microsoft Visual studio 2005
- Programming languages: C/C++

3.2 Development Environment:

1.Software – OpenGL:

OpenGL (**Open Graphics Library**) is a cross-language, cross-platform 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.

Silicon Graphics, Inc. (SGI) began developing OpenGL in 1991 and released it on June 30, 1992; applications use it extensively in the fields of computer-aided design (CAD), virtual reality, scientific visualization, information visualization, flight simulation, and video games. Since 2006, OpenGL has been managed by the non-profit technology consortium KhronosGroup.[4]

The OpenGL specification describes an abstract API for drawing 2D and 3D graphics. Although it is possible for the API to be implemented entirely in software, it is designed to be implemented mostly or entirely in hardware. The API is defined as a set of functions which may be called by the client program, alongside a set of named integer constants (for example, the constant GL_TEXTURE_2D, which corresponds to the decimal number 3553).

OpenGL

Original author(s)	Silicon Graphics
Developer(s)	Khronos Group
Initial release	June 30, 1992
Stable release	4.6 / July 31, 2017
Written in	C
Type	3D graphics API
License	Open source license for use of the S.I. This is a Free Software License B closely modeled on BSD, X, and Mozilla licenses. Trademark license for new licensees who want to use the OpenGL trademark and logo and claim conformance.
Website	opengl.org

Fig 3.2.1 OpenGL

2. Development tool – Microsoft Visual Studio:

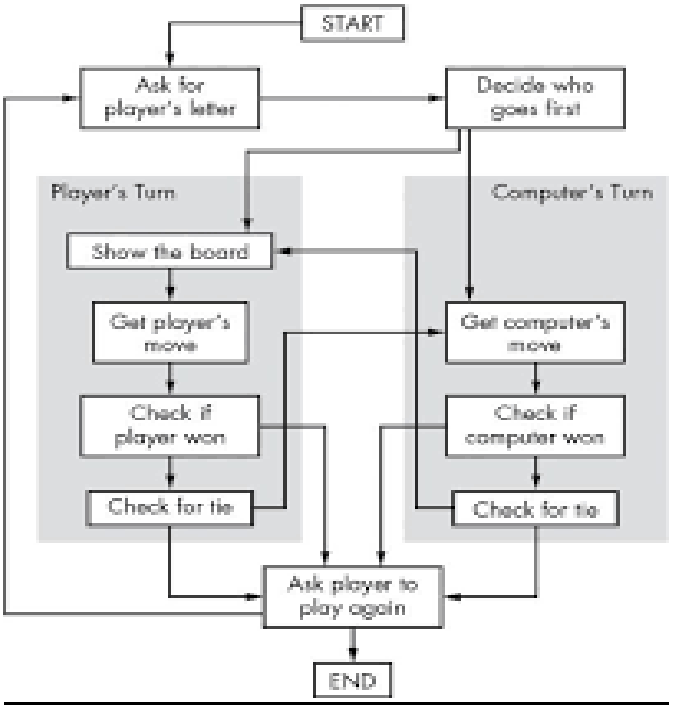
Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services, mobile apps and GUI applications. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Fig 3.2.2 Microsoft Visual Studio

CHAPTER 4

SYSTEM DESIGN

4.1 FLOW CHART :



4.2 SYSTEM DESIGN:

Existing System :

Existing system for a graphics is the TC++ . This system will support only the 2D graphics. 2D graphics package being designed should be easy to use and understand. It should provide various option such as free hand drawing , line drawing , polygon drawing , filled polygons, flood fill, translation , rotation , scaling , clipping etc. Even though these properties were supported, it was difficult to render 2D graphics cannot be . Very difficult to get a 3Dimensional object. Even the effects like lighting , shading cannot be provided. So we go for Eclipse software.

OpenGL is designed as a streamlined.

1. It is a hardware independent interface i.e. it can be implemented on manydifferent hardware platforms.
2. With OpenGL, we can draw a small set of geometric primitives such as points,lines and polygons etc.
3. Its provides double buffering which is vital in providing transformations.
4. It is event driven software.
5. It provides call back function.

4.3 DETAILED DESIGN :

TRANSFORMATION FUNCTIONS :

Matrices allow arbitrary linear transformations to be represented in a consistent format, suitable for computation. This also allows transformations to be concatenated easily (by multiplying their matrices).

Linear transformations are not the only ones that can be represented by matrices. Using homogenous coordinates,both affine transformation and perspective projection on $\mathbf{R}^n$ can be represented as linear transformations on $\mathbf{RP}^{n+1}$ (that is, n+1-dimensional real projective space). For this reason, 4x4 transformation matrices are widely used in 3D computer graphics.

3-by-3 or 4-by-4 transformation matrices containing homogeneous coordinates are oftencalled, somewhat improperly, "*homogeneous transformation matrices*". However, the transformations they represent are, in most cases, definitely non-homogeneous and non-linear (like translation, roto-translation or perspective projection). And even the matrices themselves look rather heterogeneous, i.e. composed of different kinds of elements (see below). Because they are multi-purpose transformation matrices, capable of representing both affine and projective transformations, they might be called "*general transformation matrices*", or, depending on the application, "*affine transformation*" or "*perspective projection*" matrices.

Finding the matrix of a transformation

If one has a linear transformation $T(x)$ in functional form, it is easy to determine the transformation matrix **A** by simply transforming each of the vectors of the standard basis by T and then inserting the results into the columns of a matrix. In other words,

$$\mathbf{A} = \begin{bmatrix} T(\vec{e}_1) & T(\vec{e}_2) & \cdots & T(\vec{e}_n) \end{bmatrix}$$

For example, the function $T(x) = 5x$ is a linear transformation. Applying the above process (suppose that $n = 2$ in this case) reveals that,

$$T(\vec{x}) = 5\vec{x} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \vec{x}$$

Examples in 2D graphics

Most common geometric transformations that keep the origin fixed are linear, including rotation, scaling, shearing, reflection, and orthogonal projection; if an affine transformation is not a pure translation it keeps some point fixed, and that point can be chosen as origin to make the transformation linear. In two dimensions, linear transformations can be represented using a 2×2 transformation matrix.

Rotation :

For rotation by an angle θ anticlockwise about the origin, the functional form is x' = $x\cos\theta - y\sin\theta$ and $y' = x\sin\theta + y\cos\theta$. Written in matrix form, this becomes:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Similarly, for a rotation clockwise about the origin, the functional form is x' = $x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$ and the matrix form is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scaling :

For scaling (that is, enlarging or shrinking), we have $x' = s_x \cdot x$ **and** $y' = s_y \cdot y$.

The matrix form is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

When $s_x s_y = 1$,

Then the matrix is a squeeze mapping andpreserves areas in the plane.

CHAPTER 5
IMPLEMENTATION

5.1 FUNCTIONS USED :

Void glColor3f(float red, float green, float blue);

This function is used to mention the color in which the pixel should appear. The number 3 specifies the number of arguments that the function would take. 'f ' gives the type that is float. The arguments are in the order RGB(Red, Green, Blue). The color of the pixel can be specified as the combination of these 3 primary colors.

Void glClearColor(int red, int green, int blue, int alpha);

This function is used to clear the color of the screen. The 4 values that are passed as arguments for this function are (RED, GREEN, BLUE, ALPHA) where the red green and blue components are taken to set the background color and alpha is a value that specifies depth of the window. It is used for 3D images.

Void glutKeyboardFunc();

*void glutKeyboardFunc(void (*func)(unsigned char key,int x, int y));*

where func is the new keyboard callback function. **glutKeyboardFunc** sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard callback is initially registered, and ASCII key strokes in the window are ignored. Passing NULL to glut Keyboard Func disables the generation of keyboard callbacks.

Void glFlush();

Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. glFlush empties all of these buffers,causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

Void glMatrixMode(GLenum mode);

where *mode* Specifies which matrix stack is the target for subsequent matrixoperations. Three values are accepted: **GL_MODELVIEW**, **GL_PROJECTION**, and **GL_TEXTURE**. The initial value is **GL_MODELVIEW**.

glMatrixMode sets the current matrix mode. Mode can assume one of three values:

GL_MODELVIEW	Applies subsequent matrix operations to the modelview matrixstack.
GL_PROJECTION	Applies subsequent matrix operations to the projection matrixstack.

void glViewport(GLint x, GLint y, GLsizei width, GLsizei height)

where *x, y* Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0, 0).

width, height Specify the width and height of the viewport. When a GL context is first attached to a surface (e.g. window), *width* and *height* are set to the dimensions of that surface. glViewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates. Let (x_{nd}, y_{nd}) be normalized device coordinates. Then the window coordinates (x_w, y_w) are computed as follows:

$$x_w = (x_{nd} + 1) \frac{width}{2} + x$$

$$y_w = (y_{nd} + 1) \frac{height}{2} + y$$

Viewport width and height are silently clamped to a range that depends on the implementation. To query this range, call **glGetInteger** with argument **GL_MAX_VIEWPORT_DIMS.**

void glutInit(int *argcp, char **argv);

glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInitmay cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. Examples of this situation include the failure to connect to the window system, the lack of window system support for OpenGL, and invalid command line options.glutInit also processes command line options, but the specific options parse are window system dependent.

void glutReshapeFunc(void (*func)(int width, int height));

glutReshapeFunc sets the reshape callback for the *current window*. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or whenever an overlay for the window is established. The width and height parameters of the callback specify the new window size in pixels. Before the callback, the *current window* is set to the window that has been reshaped.

If a reshape callback is not registered for a window or NULL is passed to glutReshapeFunc (to deregister a previously registered callback), the default reshape callback is used. This default callback will simplyvoid glutMainLoop(void);

glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never glutPostRedisplay(): glutPostRedisplay, glutPostWindowRedisplay — marks the current or specified window as needing to be redisplayed.

5.1 ALGORITHM

```
int minimax( char _board[9] )
 {
   short int i;
   int bestValue = +INFINITY, index = 0;
   char bestMoves[9] = {0};
   for( i = 0; i < 9; i++ )
   {
    if( _board[i] == empty )
        {
      board[i] = o;
            int value = maxSearch( _board );          if( value < bestValue )
            {
             bestValue = value;
                 index = 0;
                 bestMoves[index] = i;
            }
            else if( value == bestValue )
                 bestMoves[index++] = i;
            _board[i] = empty;}}
   if( index > 0 )
        index = rand() % index;
   return bestMoves[index];
 }
int minSearch( char _board[9] )
 {
   short int i;
   int positionValue = gameState(_board);
   if( positionValue == DRAW ) return 0;
```

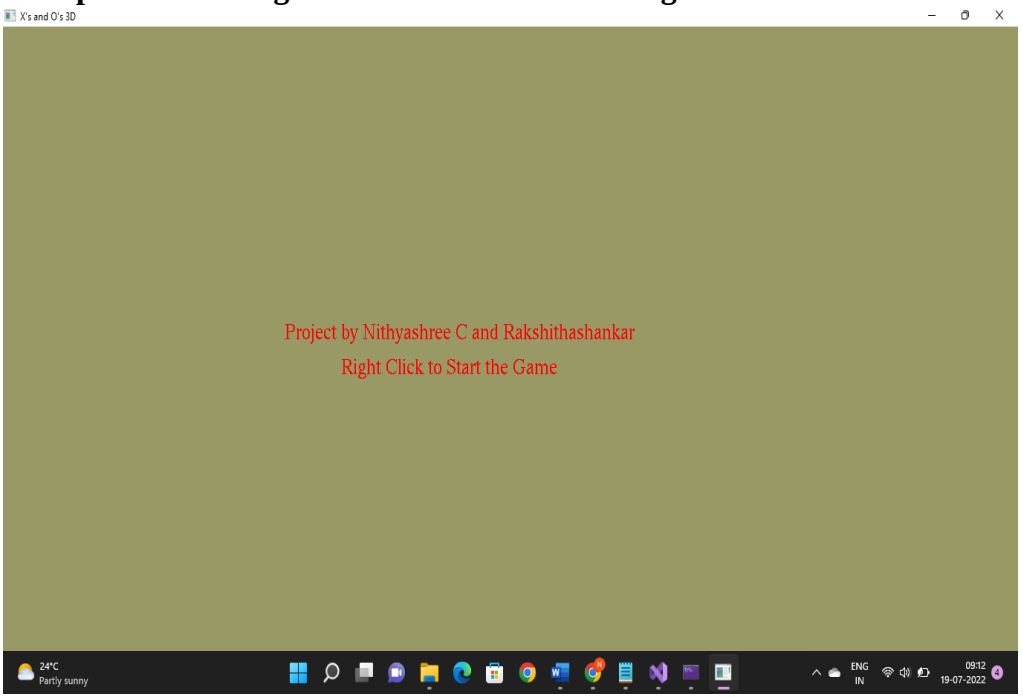
```
        if( positionValue != INPROGRESS ) return positionValue;
        int bestValue = +INFINITY;
        for( i = 0; i < 9; i++ )
        {
           if( _board[i] == empty )
              {
            _board[i] = o;
                    int value = maxSearch( _board );
                    if( value < bestValue )
                          bestValue = value;
                    _board[i] = empty;}}


 return bestValue;
}
int maxSearch( char _board[9] )
{
        short int i;
        int positionValue = gameState(_board);if(
        positionValue == DRAW ) return 0;
          if( positionValue != INPROGRESS ) return positionValue;
        int bestValue = -INFINITY;
        for( i = 0; i < 9; i++ ){
        if( _board[i] == empty )
              {
            board[i] = x;
                  int value = minSearch( _board );
                    if( value > bestValue )
                  bestValue = value;
                  board[i] = empty;
     }}
```

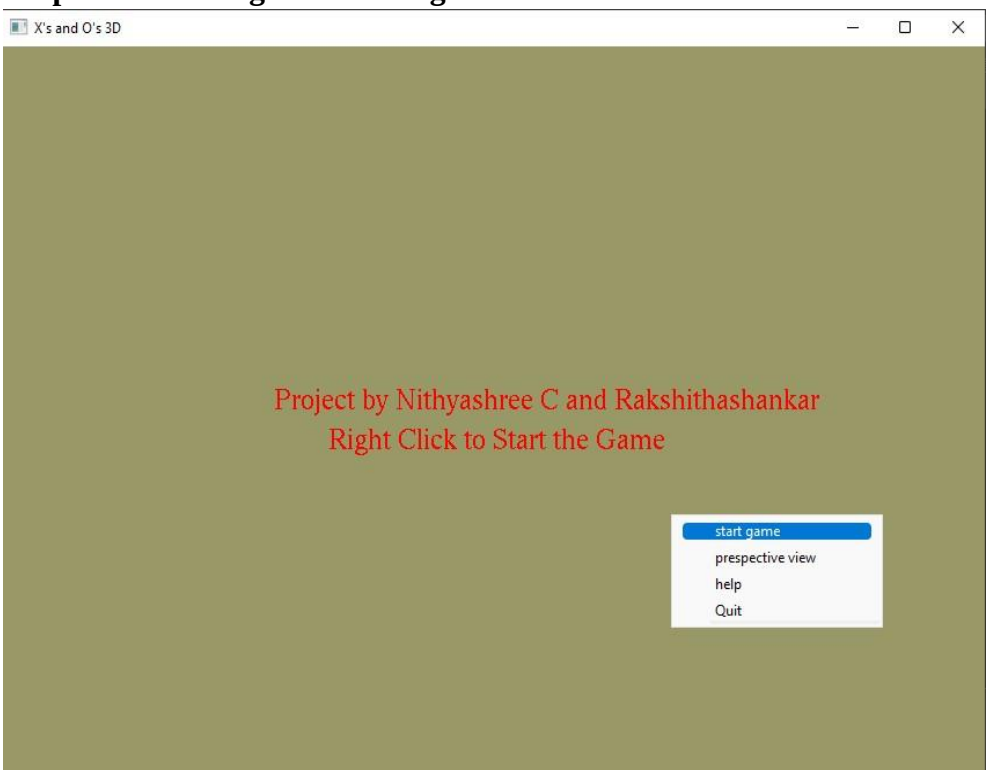
CHAPTER 6

SNAP SHOTS

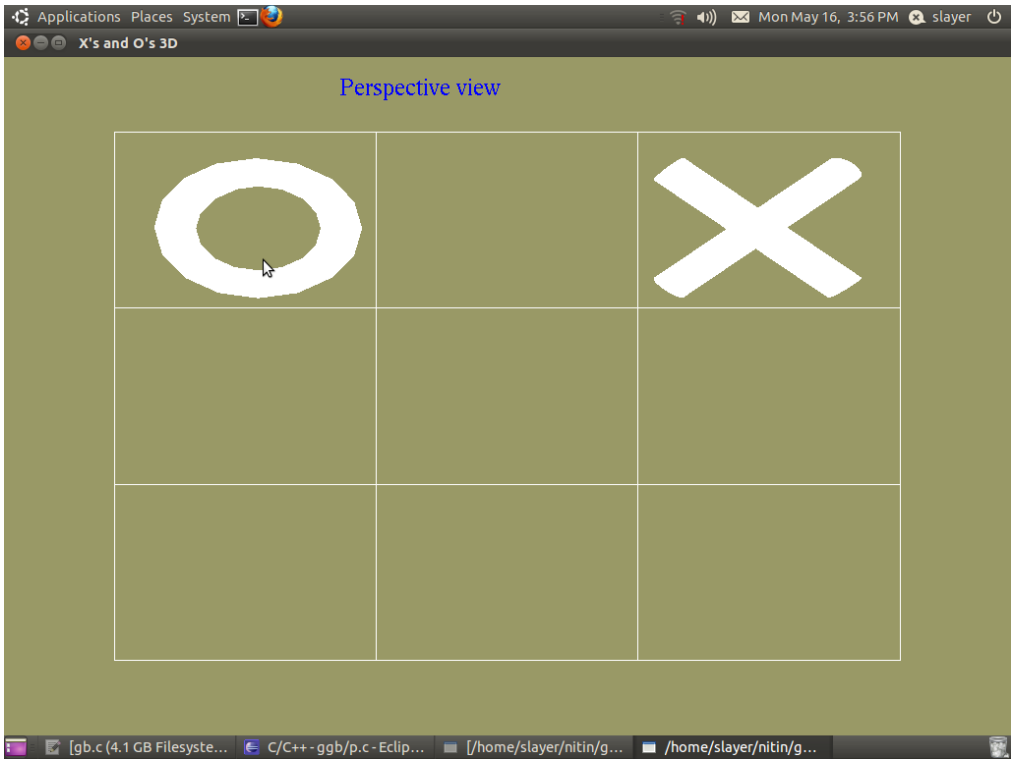
Snap Shot Showing Instructions before Starting:



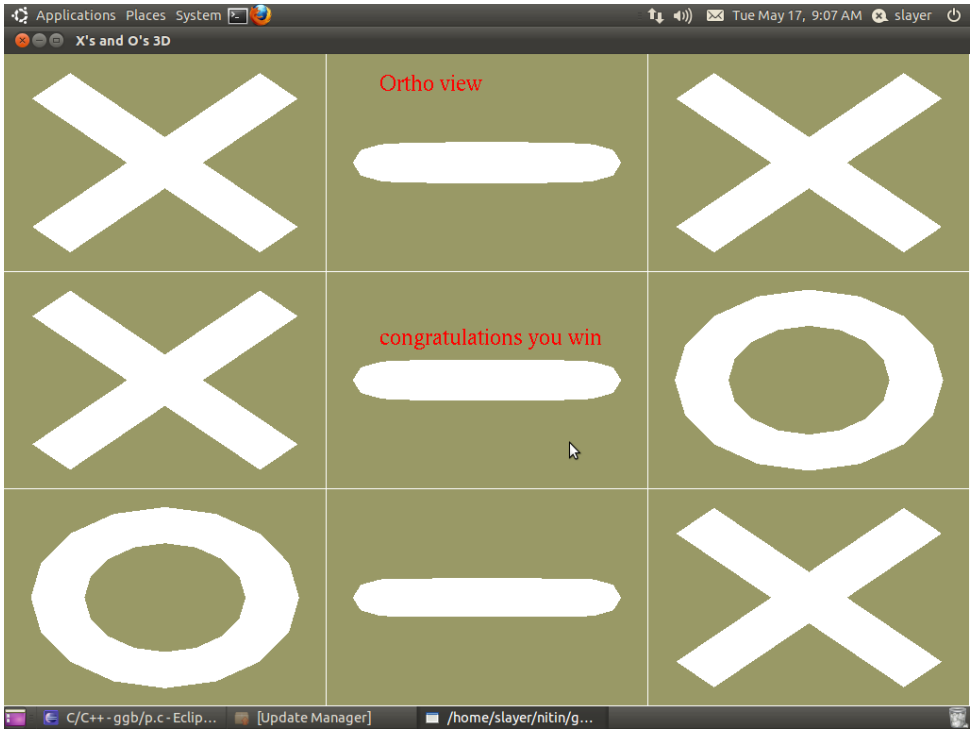
Snap Shot Showing to start the game:



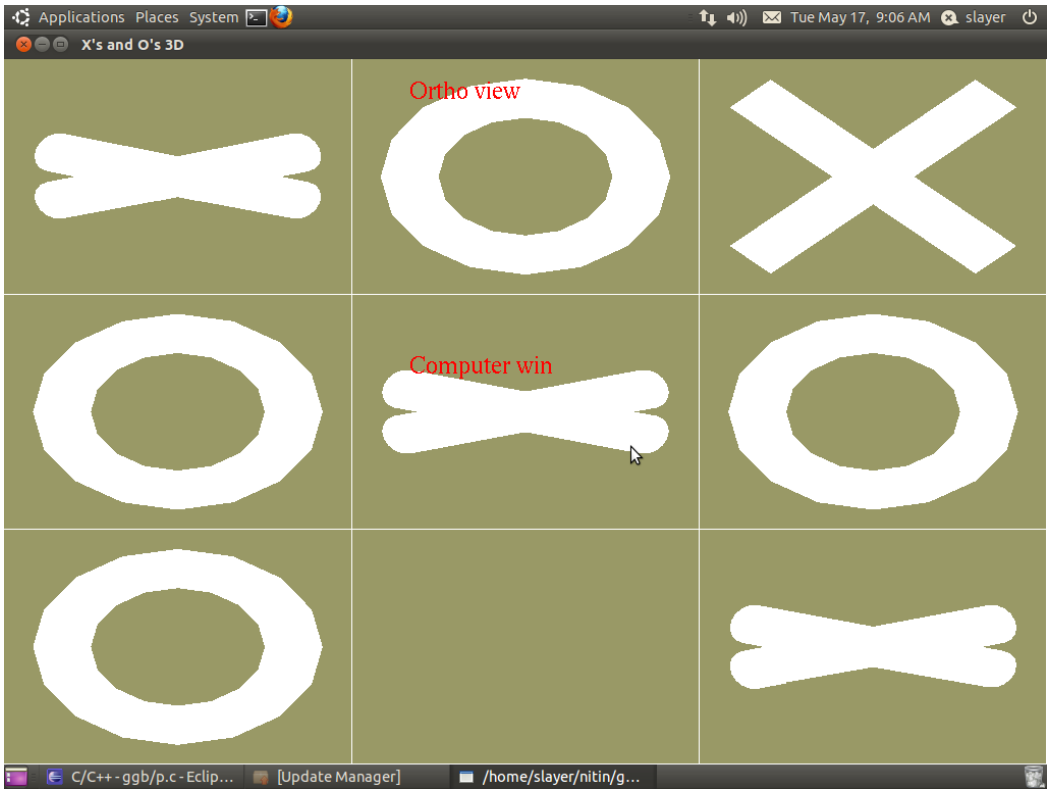
Snap Shot Showing 0's and X's as the input :



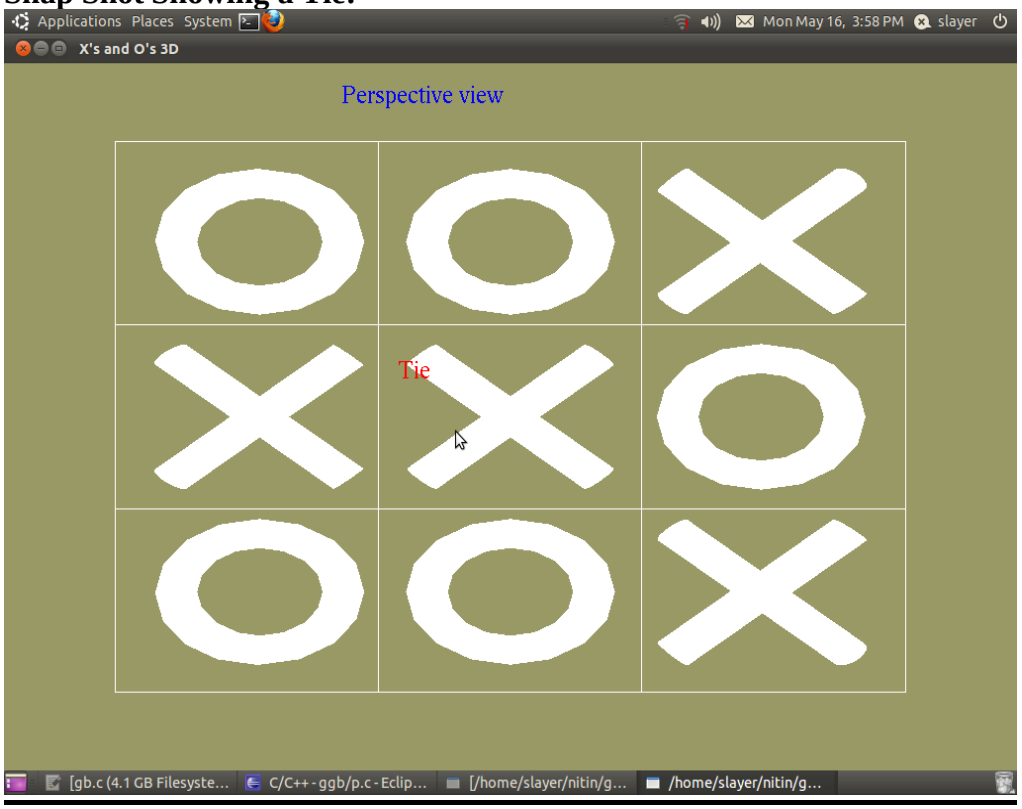
Snap Shot Showing if player wins:



Snap Shot Showing if computer wins:



Snap Shot Showing a Tie:



CHAPTER 7

CONCLUSION AND FUTURE SCOPE

The full designing and creating of **Tic-Tac-Toe** has been executed under ubuntu operating system using Eclipse, this platform provides a and satisfies the basic need of a good compiler.Using GL/glut.h library and built in functions make it easy to design good graphics package such as this simple game.

The following example game is won by the first player, X:

1. **Win**: If the player has two in a row, play the third to get three in a row.
2. **Block**: If the opponent has two in a row, play the third to block them.
3. **Fork**: Create an opportunity where you can win in two ways.

   Block opponent's fork*:*

   - **Option 1**: Create two in a row to force the opponent into defending, as long as
   - it doesn't result in them creating a fork or winning. For example, if "X" has a corner, "O" has the center, and "X" has the opposite corner as well, "O" must not play a corner in order to win. (Playing a corner in this scenario creates a fork for "X" to win.)
   - **Option 2**: If there is a configuration where the opponent can fork, block thatfork.
4. **Center**: Play the center.
5. **Opposite corner**: If the opponent is in the corner, play the opposite corner.
6. **Empty corner**: Play in a corner square.
7. **Empty side**: Play in a middle square on any of the 4 sides.

CHAPTER 7

CONCLUSION

The project is well suited for designing 2D and 3D objects, as well as for carrying out basic graphics functionalities like drawing a simple line, creating a cube, circle, square, erasing and filling them. However, if implemented on large scale with sufficient resources, it has the potential to become a standard stand-alone GUI based application for Windows Operating System. Out of the many features available, the project demonstrates some popular and commonly used features of OpenGL such as Rendering, Transformation, Rotation, Lighting, Scaling etc. These graphic functions will also workin tandem with various rules involved in the game. Since this project works on dynamic values, it can be used for real time computation. The project enabled to work with mid-level. OpenGL complexity and the project demonstrates the scope of OpenGL platform asa premier game developing launch pad. Hence, it has indeed been useful in developing many games. OpenGL in its own right is good for low cost and simple game development. It serves as an important stepping stone for venturing into other fields of Computer Graphics design and applications.

CHAPTER 8

BIBLIOGRAPHY

- Interactive Computer Graphics A Top-Down Approach with OpenGL - Edward Angel, 5th Edition, Addison-Wesley, 2008.
- The Official Guide to Learning OpenGL, by Jackie Neider, Tom Davis, Mason Woo (THE REDBOOK).
- OpenGL Super Bible by Richard S. Wright, Jr. and Michael Sweet.
- http://www.cs.rutgers.edu/~decarlo/428/glman.html online man pages.
- http://www.opengl.org/sdk/docs/man online man pages.
- http://nehe.gamedev.net OpenGL tutorials.
- And GOOGLE.