**CHAPTER 1**

**INTRODUCTION**

This project named “N-QUEEN PROBLEM" uses OpenGL software interface and develops 3D moving Train where the chess board is kept. Simple techniques like rotation, transformation, motion, etc have been used. A solution is proposed for N-Queen problem. The N-queen problem is basically a generalized form of 4-Queen problem. In 4-Queen problem, the goal is to place 4 Queens such that no queen can kill the other using standard chess queen moves.

Computer Graphics is concerned with all aspects of producing pictures or images using a computer- A particular graphics software system called OpenGL, which has become a widely accepted standard for developing graphics applications .The applications of computer graphics in some of the major areas are as follows

l. Display of information.

2. Design.

3. Simulation and Animation.

4. User interfaces.

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that are used to specify the objects and operations needed to produce interactive applications.

Key board

Application Program

Graphics library (API)

Drivers

Mouse

Display

**Fig1.1 Application programmers model of graphics system**

The interface between an application program and a graphics system can be specified through a set of functions the resides in a graphics library .These specification are called the application programmer's interface (API).The application programmer see only the API and is thus shielded from the details of both the hardware and the software implementation of the graphics library. The software drivers are responsible for interpreting the output of the API and converting this data to a form that is understood by the particular hardware.

The objective of this project is to solve the N-Queen problem and demonstrate the user how this program interacts with the system and renders the graphic design of rotating objects with a trail following them.

The other objectives of this project are:

* It makes use of interactive programming.
* It provides features of transformations such as translation, rotation, and scaling.
* It makes use of light source and uses concepts of orthogonal and perspective views.

The application program developed can be used in various fields as follows:

* In Backtracking algorithm, to solve difficult problems.
* A number of real time object models can be designed based on this concept.

This chapter deals with the first phase of development of the project by knowing the drawbacks of the existing computer graphics and proposing a new system.To determine the objectives and scope of the project.

**CHAPTER 2**

**SYSTEM ANALYSIS**

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external behavior of the system. The requirements are specified as below:

**2.1 FUNCTIONAL REQUIREMENTS:**

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases.

The various methods used in this project are as follows:-

**Controls-** The module sets the OpenGL key board operations.

**Display-** The module draws the output on the screen and the functions in it

**keys-** This module specifies the action corresponding to menu entry.

**Mouse Movement-** The module specifies the rotation with the mouse and able to see the menu.

**Idle-** This module is used to display the object more times using some delay t.

**Train-** This module is used to make a moving train and robots with features.

**Mountains-**This module helps the train move.

**2.2 NON-FUNCTIONAL REQUIREMENTS:**

These are constraints on the services or functions offered by the system.

They include timing constraints, constraints on the development process and standards. Nonfunctional requirements often apply to the system as a whole. The Non-Functional Requirements are as Dependability, Availability, Reliability, Safety, Security.

**2.3 EXECUTION IN MS VISUAL STUDIO**

Here, the coding of the project is done in Microsoft Visual Studio which is a commercial integrated development environment (IDE) with OpenGL (Open Graphics Library) which is a standard specification to produce 2D and 3D computer graphics.

The OpenGL Utility Toolkit called GLUT which is a library of utilities for OpenGL programs is also used.

**2.3.1 Microsoft Visual C++**

Microsoft Visual C++ is a commercial integrated development environment (IDE) product engineered by Microsoft for the C, C++ and C#/CLI programming languages. It has tools for developing and debugging C++ code, especially code written for the Microsoft Windows API, OpenGL API, the DirectX API and the Microsoft .NET Framework.

**2.3.2 OpenGL and GLUT**

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, describing a set of functions and the precise behaviors that they must perform. From this specification, hardware vendors create implementations - libraries of functions created to match the functions stated in the OpenGL specification, making use of hardware acceleration where possible. Hardware vendors have to meet specific tests to be able to qualify their implementation as an OpenGL implementation.

GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing

Programs in OpenGL. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms.

**2.4 SOFTWARE REQUIREMENTS**

* OpenGL(Graphics Library Utility Toolkit)
* Microsoft Visual C++ 10.0
* Operating System- Windows XP

**2.5 HARDWARE REQUIREMENTS:**

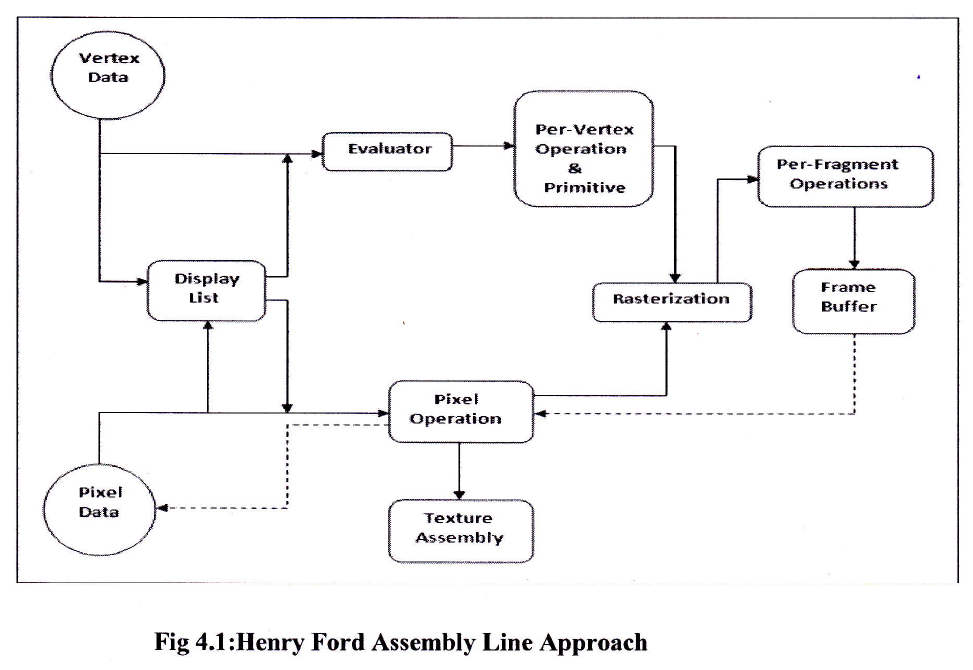
* Hard Disk - 30 GB and Above
* Memory - 256 MB and Above
* Mouse - 3 button mouse
* Keyboard
* Visual Display Unit

**CHAPTER 3**

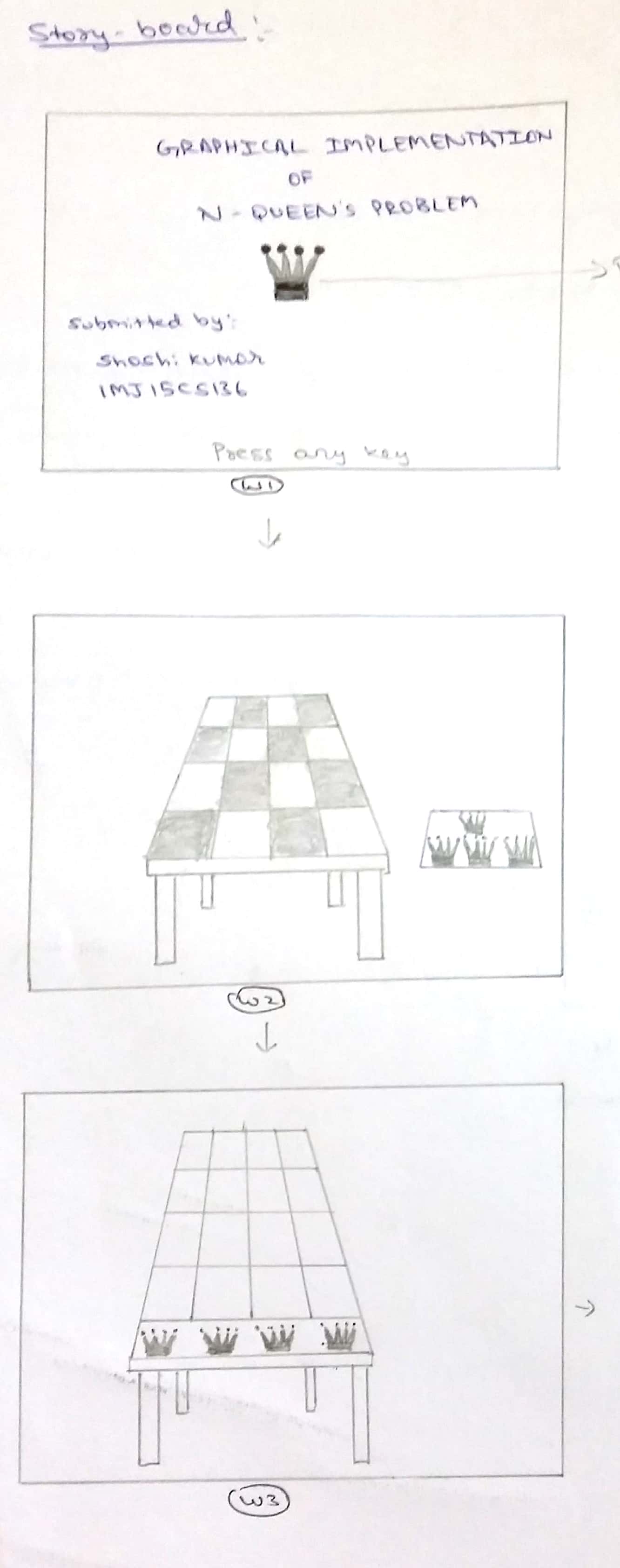
**DESIGN**

Graphics systems used general - purpose computers with the standard von Neumann architecture. Such computers are characterized by a single processing unit that processes a single instruction at a time. Information had to be sent to the display at a rate high enough to avoid flicker on the display. In the early days of computer graphics, computers were so slow that refreshing even simple images, containing a few hundred line segments, would burden an expensive computer.

The following diagram shows the Henry ford assembly line approach which OpenGL takes to process data.

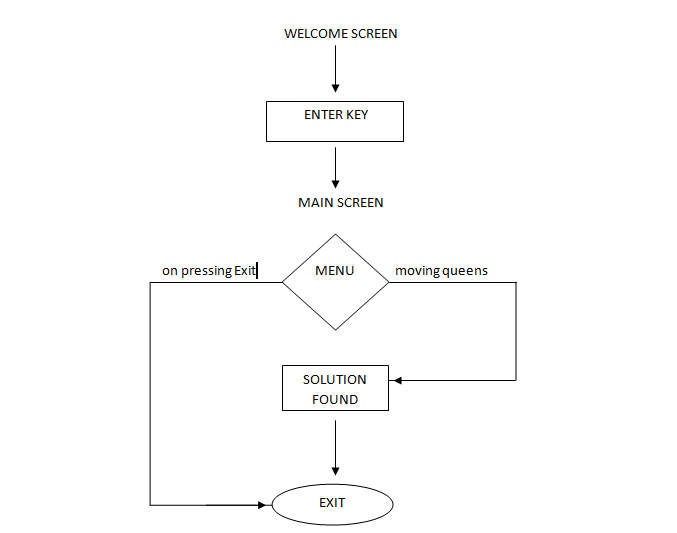


**3.1 Story Board:**

****

**Fig 3.1 : Story board**

**3.2 Flow chart of the system:**

****

**Fig 3.2 : flow chart of the system**

**3.3 Display Processors:**

Built special-purpose graphics systems were concerned primarily with relieving the general purpose computer from the task of refreshing the display continuously. These display processes had conventional architectures but included instructions to display primitives on the CRT. The main advantage of the display processor was that the instructions to generate the image could be assembled once in the host and sent to the display processor, where they were stored in the display processor's own memory as a display list, or display file. The display processor would then repetitively execute the

program in the display list, at a rate sufficient to avoid flickering independently of the host, thus freeing the host for other tasks. This architecture has become closely associated with the client-server architectures.

**3.4 Pipeline Architectures:**

The major advances in graphics architectures closely parallel the advances in work stations. In both cases, the ability to create special-purpose VLSI chops was the key enabling technology development. In addition, the availability of inexpensive solid state memory led to the universality of raster displays. For computer-graphics applications, the most important use of customs VLSI circuits has been in creating pipeline architectures.

**3.5 The graphics pipeline:**

3 major steps in the imaging process:

* **Vertex processing**
* **Clipping and primitive assembly**
* **Fragment processing**

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 ALGORITHM**

1) Start in the leftmost column

2) If all queens are placed

return true

3) Try all rows in the current column. Do following for every tried row.

a) If the queen can be placed safely in this row then mark this [row,

column] as part of the solution and recursively check if placing

queen here leads to a solution.

b) If placing queen in [row, column] leads to a solution then return

true.

c) If placing queen doesn't lead to a solution then umark this [row,

column] (Backtrack) and go to step (a) to try other rows.

3) If all rows have been tried and nothing worked, return false to trigger

backtracking.

**4.2 MODELING**

Opengl objects like Solidcube, Solidsphere, Cone, Polygons, Circles are used for Modeling of the Models used in this project. SolidCylinder, Solidsphere and Cone are usd to model the Queens. SolidCube, Polygons are used to model the Train and the mountains. SolidCubes are used to model the table, chess board, robots, window.

glLines are used to draw the Lines on the chess board when two or more queens are in the position to attack each other. Transformations such as translation, rotation and scaling are used to place the objects in proper positions.

Solidspheres are used to model the sun, moon and the stars in the background.

Many number of polygons are used to model the mountains. Solidsphere with scaling in y and z axis is used to model the clock which is fitted to the wall.

**4.3 INTERACTIONS**

The controls provided in the project are given below:

**MOUSE CONTROLS**

* RIGHT CLICK:
* Queen color ( red , green , blue )
* Queen size ( small, big )
* Theme ( coral, grey-brown , dusk , sandcastle , wheat )
* Window ( close , open )
* Time ( day , night )
* Train speed ( slow , normal , fast )
* Reset ( queen , camera , queen size , theme )
* Help ( controls page )
* Exit

**KEYBOARD CONTROL:**

* arrow up : to move the queens front.
* arrow down : to move the queens back.
* ‘6’ : to view from positive x-axis.
* ‘4’ : to view from negative x-axis.
* ‘8’ : top view.
* ‘2’ : bottom view.
* ‘5’ : zoom in.
* ‘0’ : zoom out.
* Enter key : to get into the Train.

refer to screenshots in Appendix B

**4.4 TRANSFORMATIONS**

Transformations like Translation, Rotation and Scaling are used to place the objects in proper positions. The size of the Queen scan be changed to small or big based on the users choice, Scaling transformations are used in order to change the size of the Queens.

The user can move the Queens in both forward and backward directions in order to place the queens in proper positions such that no queens can attack each other. For the movement of the Queens Translation transformation is being used in positive and negative Z-axis. So that when the user moves the Queens, Translation occurs. Mountains which are placed beyond the train uses Translation transformation for moving. All the mountains are moved in positive X-axis. Windows which are placed in the train can be opened and closed. Translation transformation is used for the opening and closing of the window in positive and negative X-axis.

as can be seen in Fig B.1

The clock placed on the left side of the train wall uses the Rotation transformation. The Solidsphere is Rotated in X and Y axis and placed in the negative X-axiz and positive YZ-axis.

as can be seen in Fig B.3

**4.5 VIEWING**

The user can look the scene from different positions and angles by using the keys in the keyboard. gluLookAt is used to change the viewing positions in the train.

gluLookAt which is consisting of 9 parameters, the first 3 parameters are the eye positions. Eye positions specifies from which position the user is looking into the train.

The user can change the Eye positions by pressing 4,6,8,2,5,0 keys in the keyboard.

The next 3 parameters specifies where the user is Looking at(at position), usually at the origin, so we give the values 0,0,0 for these parameters. The last 3 parameters specifies the camera positions.

Perspective viewing is used in this project for viewing the objects. gluLookAt used is gluLookAt(0.0,6.0,8.0,0.0,0.0,0.0,0.0,1.0,0.0);.

as can be seen in Fig B.5

**4.6 WINDOW MANAGEMENT**

Welcome screen is displayed as soon as the project is started. All the details about the project is given in the welcome window. After pressing the Enter key, the second window which is the main project window will be displayed. Solution for the N-QUEEN PROBLEM will be given in the second window.

Controls window which helps the user to know about the keyboard controls will be displayed after the main window. On successfully exiting the window, the project will be closed and all the windows will be destroyed.

The 3 Windows used in this project are

* Welcome window
* Main window
* Controls window

as can be seen in Fig B.1

**CHAPTER 5**

**CONCLUSION**

In this project, the design and implementation of 3-dimensional graphics Interfacing program has been attempted. We have seen the functions performed by this application and code for the same. In OpenGL, C/C++ provide enormous flexibility in the design and the use of C/C++ graphics programs.

The presence of many in-built OpenGL functions and libraries to take care of many of the functionalities reduces the burden of coding and makes the implementation simpler.

The project started with the designing phase in which we figured the requirements needed the system design, Data flow diagram etc. Then comes the details of the implementation phase where in we have included various functionalities. And now after the testing phase, the project comes to an end.

**APPENDIX A**

**SAMPLE CODE**

void windisplay()

{

w1 += w1s;

if (w1 > 30.0)

{

w1 = 0.0;

}

if (flag < 0)

{

flag = 0;

}

if (flag > 4)

{

flag = 4;

}

if (flag == 3)

{

c1 += 0.009;

if (c1 >3.6)

{

c1 = 3.6;

}

c3 += 0.009;

if (c3 > 3.6)

{

c3 = 3.6;

}

}

if (flag == 4)

{

c3 += 0.009;

if (c3 > 4.5)

{

c3 = 4.5;

}

}

if (close == 1) {

m1 += 0.010;

if (m1 > 3.14)

{

m1 = 3.14;

}

}

else {

m1 -= 0.010;

if (m1 < 0)

{

m1 = 0;

}

}

glutPostRedisplay();

}

//queen1

glPushMatrix();

glTranslatef(-1.4, 0.0, 4.1);

glTranslatef(0.0, 0.0, -c1);

queen();

glPopMatrix();

//Line 1

if (flag == 1)

{

glPushMatrix();

glColor3f(1.0, 0.0, 0.0);

glLineWidth(4);

glBegin(GL\_LINES);

glVertex3f(-1.8, 0.30, 2.4);

glVertex3f(1.7, 0.30, 2.4);

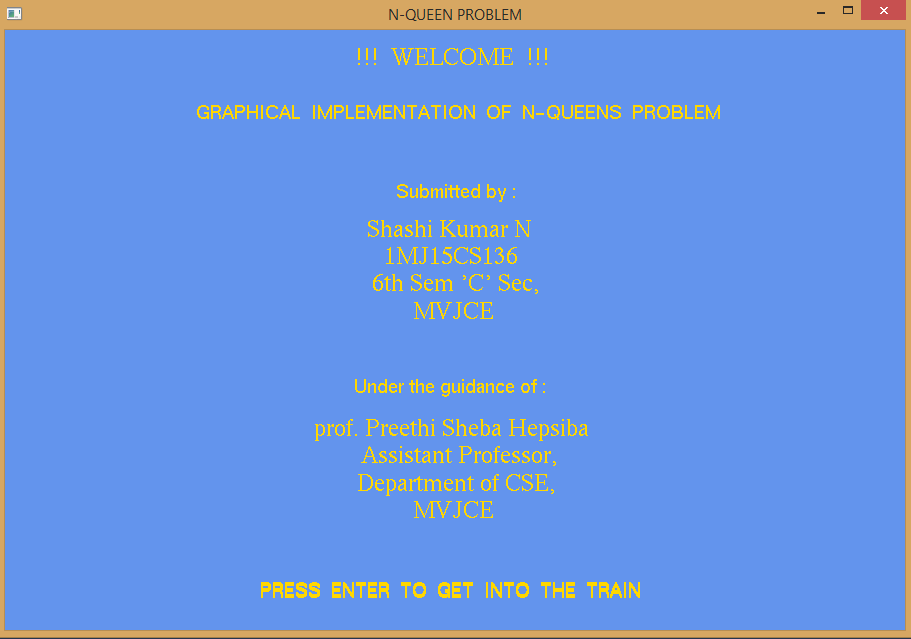
glEnd();

glPopMatrix();

}

**APPENDIX B**

**SCREEN SHOTS**

** Fig B.1: Front Page**



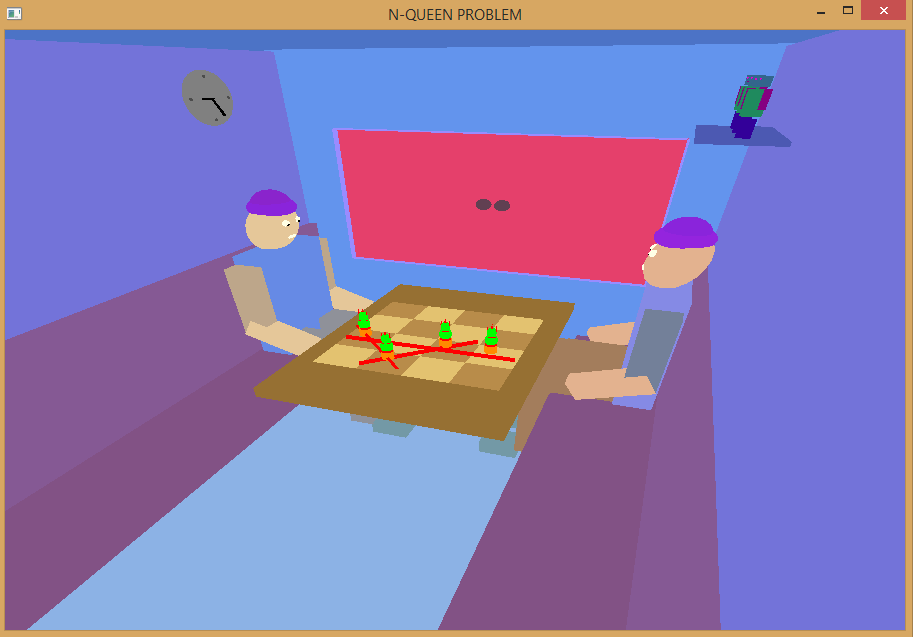
**Fig B.2: Controls**

****

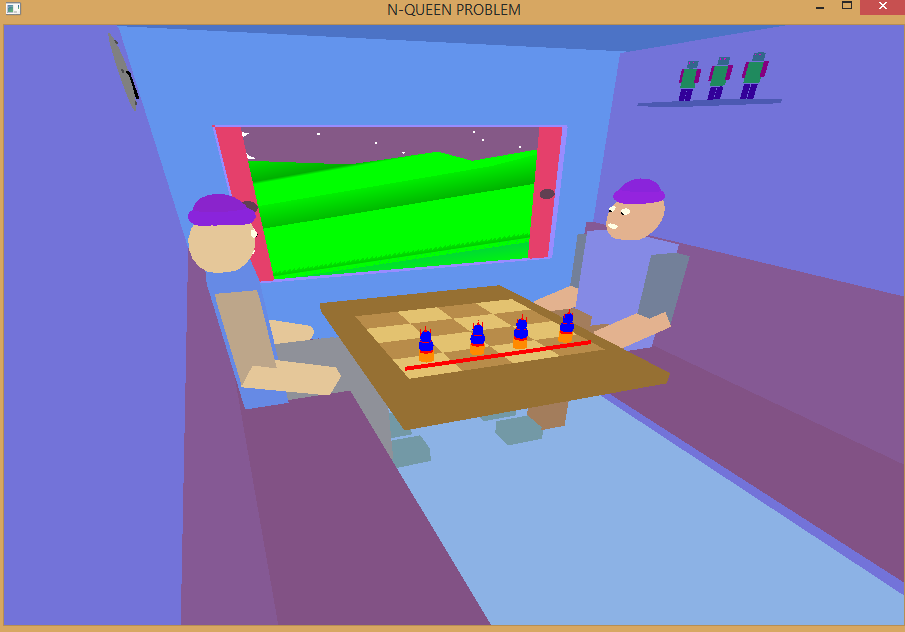
**Fig B.3 : Movement of Queens**



**Fig B.4 : right click after acheiving solution**

****

**Fig B.5 : view from right side with closed window**

****

**Fig B.6 : view from left side with opened window**

**BIBLIOGRAPHY**

* During the course of this project reference to the following books and materials were made:

[1] Edward Angel, *“Interactive Computer Graphics”*, Pearson Publication, 5th edition

[2] F.S. Hill Jr,”Computer Graphics Using OpenGL”,Pearson Publication,2nd edition

[3] James D. Foley,”Computer Graphics”,Addision Wesley,1997

* Internet was extensively browsed for various materials related to this project

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

[5] [http://www.cs.usr.edu](http://www.cs.usr.edu/)

[6] [http://vtupro.com](http://vtupro.com/)