**CHAPTER 1**

**INTRODUCTION**

### OPENGL :

### OpenGL is the abbreviation for Open Graphics Library. It is a software interface for graphics hardware. This interface consists of several hundred functions that allow you, a graphics programmer, to specify the objects and operations needed to produce high-quality color images of two-dimensional and three-dimensional objects. Many of these functions are actually simple variations of each other, so in reality there are about 120 substantially different functions. The main purpose of OpenGL is to render two-dimensional and three-dimensional objects into the frame buffer. These objects are defined as sequences of vertices (that define geometric objects) or pixels (that define images). OpenGL performs several processes on this data to convert it to pixels to form the final desired image in the frame buffer.

### As a software interface for graphics hardware, OpenGL’s aim purpose is to render two- and three-dimensional objects into a frame buffer. OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.

### 1.2 OPENGL FUNDAMENTALS :

This section explains some of the concepts inherent in OpenGl. Primitives and Commands OpenGL draws primitives-points, Line segments of polygons-subject to several selectable modes. You can control modes independently of each other; that is setting one mode doesn’t affect whether other Modes are set Primitives are set. Primitives are specified, modes are set, and other OpenGL Operations are described by issuing commands in the form of function calls. Primitives are defined by a group of one or more vertices. A vertex defines a point, an endpoint of a line , or a corner of a polygon where two edges meet. Data is associated with a vertex, and each vertex and its associated data are processed independently, in order, and in the way. The type of clipping depends on which primitive the group of vertices represents. Commands are always processed in the order in which they are received, although there may be an indeterminate delay before a command takes effect. This means that each primitive is drawn completely before any subsequent command takes effect. It Also means that state-querying commands return data that’s consistent with complete execution of all Previously issued OpenGL commands.

## **HISTORY**

As a result, SGI released the OpenGL standard In the 1980s, developing software that could function with a wide range of graphics hardware was a real challenge. Software developers wrote custom interfaces and drivers for each piece of hardware. This was expensive and resulted in much duplication of effort.

By the early 1990s, Silicon Graphics (SGI) was a leader in 3D graphics for workstations. Their IRIS GL API was considered the state of the art and became the de facto industry standard, overshadowing 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 terms of 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. This in turn caused SGI market share to weaken as more 3D graphics hardware suppliers entered the market. In an effort to influence the market, SGI decided to turn the Iris GL API into an open standard.

SGI considered that the Iris GL API itself wasn't suitable for opening due to licensing and patent issues. Also, the Iris GL had API functions that were not relevant 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 systems were developed.

In addition, SGI had a large number of software customers; by changing to the OpenGL API they planned to keep their customers locked onto SGI (and IBM) hardware for a few years while market support for OpenGL matured. Meanwhile, SGI would continue to try to maintain their customers tied to SGI hardware by developing the advanced and proprietary Iris Inventor and Iris Performer programming APIs.

### THE OPENGL INTERFACE

### Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are

### GL – GRAPHICS LIBRARY

### Functions in the main GL (or OpenGL in Windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in Windows).

### GLU – GRAPHICS UTILITY LIBRARY

### This library uses only GL functions but contain code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begins with the letters glu.

### GLUT – OPENGL UTILITY TOOLKIT

### To interface with the window system and to get input from external devices into our programs we need at least one more library. For the X window System, this library is called GLX, for Windows, it is wgl, and for the Macintosh, it is agl. Rather than using a different library for each system, we use a readily available library called the OpenGL Utility Toolkit (GLUT) , which provides minimum functionality that should be expected in any modern windowing system.

**1.5 OBJECTIVES**

The objectives of this study are summarized below:

* To develop a Open GL software called “PATHFINDING GAME”.
* To build the environment for the player to improve his quick thinking/accuracy.
* To build the basic platform of problem solving for the player.
* To progress the thinking ability of the player to solve the game.
* To navigate through the maze and complete the game within a minute then he wins the game either he loses the game.

**CHAPTER 2**

**LITERATURE SURVEY**

To demonstrate the transformation and lightening, effects, different polygons have Computer graphics started with the display of data on hardcopy plotters and cathode ray tube (CRT) screens soon after the introduction of computers.

Computer graphics today largely interactive, the user controls the contents, structure, and appearance of objects and of displayed images by using input devices, such as keyboard. mouse, or touch-sensitive panel on the screen. Graphics based user interfaces allow millions of new users to control simple, low-cost application programs, such processors, and drawing program spread sheets, word processors, and drawing programs.

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. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. It is also used in video games. where it competes with Direct3D on Microsoft Windows platforms (see Direct3D vs OpenGL). OpenGL is managed by the non-profit technology consortium, the Khronos Group.

OpenGL provides a powerful but primitive set of rendering command, and all higher level drawing must be done in terms of these commands. There are several libraries that allow you to simplify your programming tasks, including the following:

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 and rendering surfaces.

OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, written by Mark Kill guard, to hide the complexities of differing window APIs.

To achieve the objective of the project, information related to the light sources is required with OpenGL we can manipulate the lighting and objects in a scene to create many different kinds of effects. It explains how to control the lighting in a scene, discusses the OpenGL conceptual model of lighting, and describes in detail how to set the numerous illumination parameters to achieve certain effects. This concept is being obtained from to be used. Polygons are typically drawn by filling in all the pixels enclosed within the boundary, but we can also draw them as outlined polygons or simply as points at the vertices. This concept is obtained from.

The properties of a light source like its material, diffuse, emissive, has to mention in the project. So to design the light source and the objects, programming guide of an OpenGL is used.

**CHAPTER 3**

**SYSTEM REQUIREMENTS**

**3.1 HARDWARE REQUIREMENTS :**

* Processor : intel core i3 or above
* RAM : 512 MB or more
* Hard Disk : 5GB or above
* Keyboard : QWERTY Keyboard
* Monitor : 1024 x 768 display resolution

**3.2 SOFTWARE REQUIREMENTS :**

* Operating System : Windows
* Tool Used : CodeBlocks
* Library : OpenGL
* Language : C

# CHAPTER 4

**SYSTEM DESIGN**

**4.1 INITIALIZATION**

* Initialize to interact with the Windows.
* Initialize the display mode that is double buffer and RGB color system.
* Initialize window position and window size.
* Initialize and create the window to display the output.

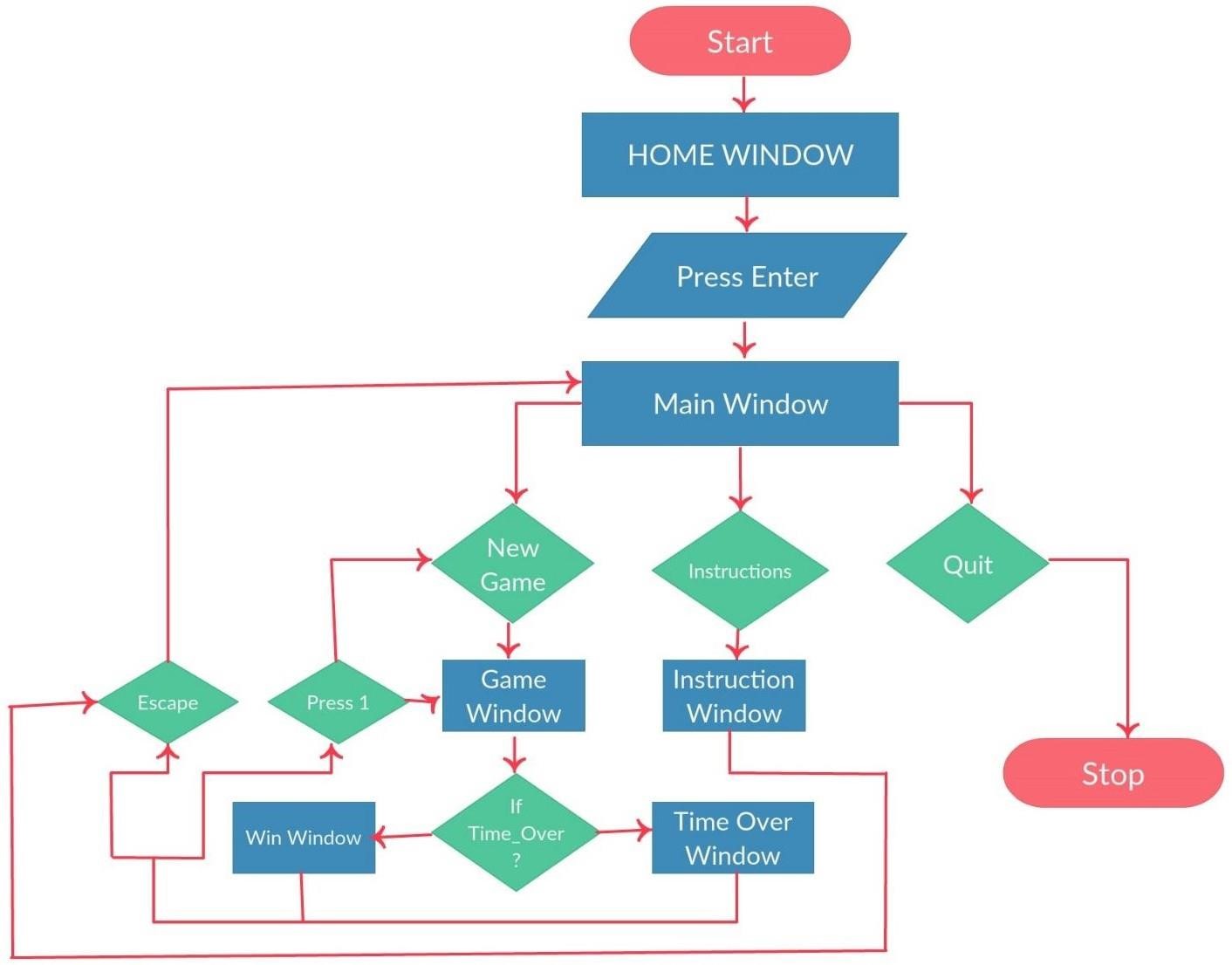
### 4.2 DISPLAY

* Introduction page of “Pathfinding Game”
* Menus are created and depending on the value returned by menus.
* Suitable operations are performed.
* The operations performed are:
* New Game
* Instructions
* Quit

### 4.3 FLOW CHART

When we run the program, home window appears. On clicking ‘Enter’ button Main window is opened. In main window list of options like New Game, Instructions & Quit appears.

By selecting any of these options we can perform the specified operation in the game.



**Fig: 4.3 Flow Chart**

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 OVERVIEW**

This project is a demonstration of “Maze Game”. We have taken the help of built in functions present in the header file. To provide functionality to our project we have written sub functions. These functions provide us the efficient way to design the project. In this chapter we are describing the functionality of our project using these functions.

Keyboard interactions are provided where, when a Enter button is pressed, menu displays and we can select options from menu displayed.

**5.2 USER INTERFACE**

The Project which we have done uses OpenGL functions and is implemented using C. Our Project is to demonstrate MAZE GAME. User can perform operations using keyboard.

**Keyboard interaction**

* Firstly, after compiling we get a Home Page.
* Then we click the Enter button to display the Main window here we get three options in which user has to specify his choices:
* New Game: To start the new game.
* Instructions: It Guides the user how to play the game.
* Exit: Quits the Game.
* As the player clicks 1 i.e. To open the new game.
* Now in game the player uses the arrow key to complete the game.
* Regardless of a win or a lose the player is redirected to pop-up page, where again he has to specify his choice.

**5.3 STRUCTURE**

* void point();
* void point1();
* void point2();
* void output(int x, int y, char \*string);
* void draw\_string(int x, int y, char \*string);
* void frontscreen(void);
* void winscreen();
* void startscreen();
* void instructions();
* void timeover();
* void idle();
* void wall();
* void specialkey(int key, int x, int y);
* void display();
* void keyboard(unsigned char key, int x,int y);
* void mynit();
* void myreshape(int w, int h);
* int main(int argc, char\*\* argv);

**5.4 FUNCTIONS**

A function is a block of code that has a name and it has a property that it is reusable that is it can be executed from as many different points in a c program as required. The partial code of various function that have been used in the program are:

**5.4.1 myInit**

This function is used to initialize the graphics window.glMatrixMode (GL\_PROJECTION), glLoadIdentity() are used to project the output on to the graphics window.

**5.4.2 Display**

If df==10, i.e., it will call the frontscreen(), else if df==0 then startscreen() is called, Now the game has been started with the timer of 60sec displaying the MAZE to be solved by the player

**5.4.3 Wall**

This function is used to display the Wall forming the Maze.

**5.4.4 Point**

This function is used to create a color point in the game to identify the start & end point. In the game starting point is green & end point is red, and the player’s color point is blue which he uses to play the game

**5.4.5 Frontscreen**

This is the function which helps in opening the Home page of the game. This page is linked to all other pages described before. After clicking enter in this page the Main page is opened.

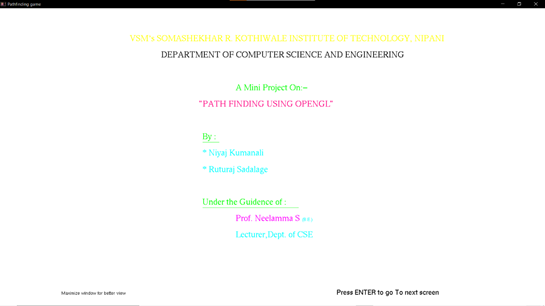
**5.4.6 Idle**

This function is the major criteria of this game as it sets a timer for the player which limits the player to finish his game within 60sec else he loses the game.

**CHAPTER 6**

**SCREENSHOTS**

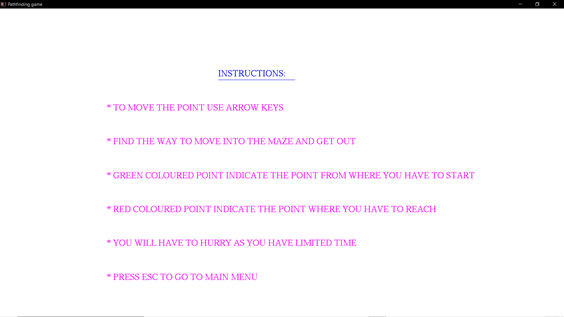
**Front Page:**



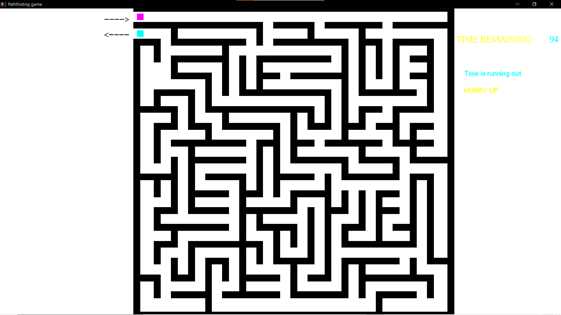
**Main Menu :**

****

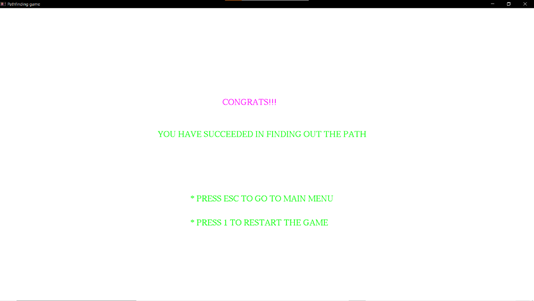
**Instructions:**



**Game Screen :**

****

**Win Screen :**

****

**Time Over (Lost) :**

****

**CHAPTER 7**

**ADVANTAGES**

* Enhance Cognitive Skills. Mazes improve the cognitive skills
* Fine Tune Motor Skills
* Strengthen Visual Skills
* Hand Eye Coordination
* Find Effective Solutions
* Virtue of Patience
* Build Confidence

**CONCLUSION**

Pathfinding Game is designed and implemented using a graphics software system called OpenGL which has became a widely accepted standard for developing graphic application. Using OpenGL functions user can create geometrical objects and can use translation, rotation, scaling with respect to the co-ordinate system. The development of this project has enabled us to improve accuracy, problem solving skills while providing a fun and interactive experience to the player.

# BIBLIOGRAPHY

**BOOKS:**

1. Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd / 4th Edition, Pearson Education,2011
2. Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th edition. Pearson Education, 2008
3. James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL: pearson education
4. Xiang, Plastock : Computer Graphics , sham’s outline series, 2nd edition, TMG.

**List of websites:**

1. <http://www.opengl.org/>
2. <http://www.academictutorials.com/graphics/graphics>[-flood-fill.asp](http://www.academictutorials.com/graphics/graphics-flood-fill.asp)
3. <http://www.glprogramming.com/>
4. <https://www.opengl.org/discussion_boards/showthread.php/167379>[-Making-a-maze- using-arrays](https://www.opengl.org/discussion_boards/showthread.php/167379-Making-a-maze-using-arrays)