VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590014



A VIth Semester Mini-Project Report on

# “Maze Game”

Submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Saraswati Hunnur 2AG20CS062

Sukanya Benawadi 2AG20CS077

Under the Guidance of

Prof. Pavan K

Assistant Professor, Dept. of CSE, AITM, Belagavi.



ANGADI INSTITUTE OF TECHNOLOGY &MANAGEMENT

BELAGAVI-590009 2022-2023

ANGADI INSTITUTE OF TECHNOLOGY &MANAGEMENT

BELAGAVI -590009

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Certificate

This is to certify that Project entitled “Maze Game” is work carried out by Saraswati Hunnur (2AG20CS062) and Sukanya Benawadi (2AG20CS077) in partial fulfillment of the requirements for the award of the degree of Bachelor of Computer Science & Engineering under Visvesvaraya Technological University, Belagaviduring the year 2022-2023. It is certified that all the correction/suggestion indicated for internal assessment have been incorporated in the report. The Management System 6th Semester MiniProject report has been approved as it satisfies the academic requirements in respect of miniproject work prescribed for the Bachelor of Engineering degree.

Signature of the Guide Signature of the HOD Signature of the Principal

Prof. Pavan K Prof.Dhanashree Kulkarni Dr.Anand Deshpande

Assistant Professor, Assistant Professor and Head Principal and Director,

Dept. of CSE, AITM Dept. of CSE,AITM AITM, Belagavi

Name of the Examiner: Signature with date:

1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## DECLARATION

We **Saraswati Hunnur (2AG20CS062), Sukanya Benawadi (2AG20CS077)**, studying in the 6th semester of Bachelor of Engineering in Computer Science and Engineering at Angadi Institute of Technology and Management, Belagavi, hereby declare that this mini project work entitled “**Maze Game**” which is being submitted by us in the partial fulfilment for the award of the degree of Bachelor of Engineering in Computer Science and Engineering from Visvesvaraya Technological University, Belagavi is an authentic record of us carried out during the academic year 2022-2023 under the guidance of Prof. Pavan K, Department of Computer Science and Engineering, Angadi Institute of Technology and Management, Belagavi.

We further undertake that the matter embodied in the dissertation has not been submitted previously for the award of any degree or diploma by us to any other university or institution.

Place: Belagavi **Saraswati Hunnur**

**Date:**  **Sukanya Benawadi**

## ACKNOWLEDGEMENT

It is our proud privilege and duty to acknowledge the kind of help and guidance received from several people in preparation of this report. It would not have been possible to prepare this report in this form without their valuable help, cooperation and guidance.

First and foremost, we wish to record our sincere gratitude to **Management of Angadi Institute of Technology and Management, Belagavi** and to our beloved Principal **Dr.Anand Deshpande**, Angadi Institute of Technology and Management, Belagavi for his constant support and encouragement in preparation of this report and for making available library and laboratory facilities needed to prepare this report.

Our sincere thanks to our HOD **Prof.Dhanashri *Kulakarni*,** Department of Computer Science and Engineering, Angadi Institute of Technology and Management, Belagavi for her valuable suggestions and guidance throughout the period of this report.

We express our sincere gratitude to our guide **Prof. Pavan K**,Department of Computer Science and Engineering, Angadi Institute of Technology and Management, Belagavi for guiding us in investigations for this seminar and in carrying out experimental work. Our numerous discussions with him were extremely helpful. I hold him in esteem for guidance, encouragement and inspiration received from him.

The mini -project on “**Maze Game**” was very helpful to us in giving the necessary background information and inspiration in choosing this topic for the project.

Last but not the least, we wish to thank our **parents** for financing our studies in this college as Last well as for constantly encouraging us to learn engineering. Their personal sacrifice in providing this opportunity to learn engineering is gratefully acknowledged.

**Saraswati Hunnur,**

**Sukanya Benawadi**

## ABSTRACT

“PERPLEXITY” is a 2D Maze game. The game is created using OpenGL where player tries to solve the maze. The objective of the game is to navigate through the maze and complete the game within a minute then he wins the game either he loses the game

**CONTENTS**

**TOPIC PAGE NO.**

**Chapter 1: INTRODUCTION 1-11**

* 1. OpenGL
  2. History
  3. Features of openGL
  4. Basic OpenGL Operation
  5. OpenGL Interface
  6. Graphics Functions
  7. Data Types
  8. Objectives

**Chapter 2: SYSTEM REQUIREMENT 12**

2.1 Software Requirements

2.2 Hardware Requirements

**Chapter 3: SYSTEM DESIGN 13-14**

* 1. Initialization
  2. Display
  3. Flowchart

**Chapter 4: IMPLEMENTATION 15-21**

1. 4.1 Overview
2. 4.2 User Interface
3. 4.3 Structure
4. 4.4 Analysis

**Chapter 5: SNAPSHOTS 22**

**LIST OF FIGURES**

**SI.NO. Fig. No.**  **NAME Page No.**

1. 1.4 OpenGL Block Diagram 4

2. 1.5 Library Organization 7

3. 1.7 Data Types List 10

4. 3.3 Flow Chart 14

5. 5.1 Home page 22

6. 5.2 Main Window 23

7. 5.3 Game Page 23

8. 5.4 Win Page 24

9. 5.5 Game Over Page 24

10. 5.6 Instruction Page 25

Chapter 1

## INTRODUCTION

1.1 OPENGL

OpenGL is the abbreviation for Open Graphics Library. It is a software interface for

graphics hardware. This interface consistes of several hundred function 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 function .The main purpose of OpenGL is to render two-dimensional objects into the frame buffer. These objects are defined as sequence 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 perform several processing steps on this data to convert it to pixels to from the final desired image in the frame buffer.

**openGL Fundamentals**

This section explains some of the concepts inherent in openGL

Primitives and Commands

OpenGL draws primitives-points,Line segements of polygoans-subject to several selectable modes.

You cPan 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.

rimitives 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 date 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 returen data that’s consistent with complete execution of all Previously issued OpenGL commands.

**1.2 HISTORY**

As a result, SGI released the OpenGL standard In the 1980s,developing software that could function with 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 lweaken 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.

**1.3** **FEATURES OF OPENGL**

* **Industry standard**

An independent consortium, the OpenGL Architecture Review Board, guides the OpenGL specification. With broad industry support, OpenGl is the only truly open, vendor-neutral, multiplatform graphics standard.

* **Stable**

OpenGL implementations have been available for more than seven years on a wide variety of platforms. Additions to the specification are well controlled, and proposed updates are announced in time for developers to adopt change. Backward compatibility requirements ensure that existing applications do not become obsolete.

* **Reliable and portable**

All OpenGL applications produce consistent visual display results on any OpenGL API-compliant hardware, regardless of operating system or windowing system.

* **Evolving**

Because of its through and forward-looking design, OpenGL allows new hardware innovations to be accessible through the API via the OpenGL extension mechanism. In this way, innovations appear in the API in a timely fashion, letting application developers and hardware vendors incorporate new features into their normal product release cycles.

* **Scalable**

OpenGL API-based application can run on systems ranging from consumer electronics to PCs, workstations, and supercomputers. As a result, applications can scale to any class of machine that the developer chooses to target.

* **Easy to use**

OpenGL is well structured with an intuitive design and logical commands. Efficient OpenGL routines

typically result in application with fewer lines of code than those that Commands are always processed

in the order in which they are received, although there may be an indeterminate delay before a command tasks effect.

This means that each primitive is drawn completely before any subsequent command takes effect.

It Also means that state-querying command return data that’s consistent with complete execution of

All Previously issued OpenGL commands.

**1.4 BASIC OPENGL** **OPERATION**

The following diagram illustrates how openGL processes data. As shown, commands enter from

the left and proceed through a processing pipeline. Some commands specify geometric objects to be drawn,

and others control how the objects are handled during various processing stages.

Display list

Frame buffer

Per-fragment operations

Rasterization

Pre-vertex operations & primitive assembly

ass

ass

ass

Evaluator

Commands

Texture memory

Pixel operations

**Fig: 1.4 OpenGL Block Diagram**

**1.5 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 Window) library have names that begin with

the letters g1 and are stored in a library usually referred to as GL (or OpenGL in Window).

* **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

implementation; functions in the GLU library begin with the letters g1u.

* **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 wg1, and for the Macintosh, it is ag1.Rather than using a different library for each system, we

use a readily available library called the OpenGL Utility (GLUT), which provides minimum functionality that should be expected in any modern windowing system.

* **Display list**

Rather than having all commands proceed immediately through the pipeline, you can choose to

accumulate some of them in a display list for processing later.

* **Evaluator**

The evaluator stage of processing provides an efficient way to approximate curve and surface

geometry by evaluating polynomial commands of input values.

* **Pre-vertex operations and primitive assembly**

OpenGL process geometric primitives-points, line segments, and polygons all of which are described

by vertices. Vertices are transformed , and primitives are clipped to the viewport in preparation for

rasterization.

* **Rasterization**

The rasterization stage products a series of frame-buffer addresses and associated values using

a two-dimensional description of a point ,line segment, or polygon. Each so produced is fed into the

last stage ,per-fragment operations.

* **Per-fragment operations**

Theseare the final operations performed on the data before it is stored as pixels in the frame

buffer Pre-fragment operations include conditional update to frame buffer based on incoming and

previously stored z values (for z buffering) and blending of incoming pixel colors, as well as making

and other logical operations on pixel values.

* **Pixel Operation**

Input data can be in the form of pixels rather than vertices. Such data which might describe an

Image for texture mapping skips the first stage of processing and instead processed as pixels in the pixel operation stage.

* **Texture memory**

Make up programs generated using other graphics libraries packages. In addition, OpenGL

drives encapsulate information about the underlying hardware, freeing the application developer from

having to design for specific hardware features.

* **Well-document**

Numerous books have been published about OpenGL, and a great deal of sample code is

readily available, making information about OpenGL inexpensive and easy to obtain.

Frame

buffer

GL

GLU

GLX

GLUT

OpenGL Application Program

Xlib

**Fig 1.5: Library Operation**

The above figure shows the organization of the libraries for an X Window System environment.

**1.6 GRAPHICS FUNCTIONS**

Our basic model of a graphic package is a black box, a term that engineers use to denote a system whose properties are described only by its input and outputs; we may know nothing about its internal workings. OpenGL functions can be classified into seven major groups:

* **Primitive functions**

The primitive functions define the low-level objects or atomic entities that our system can display. Depending on the API, the primitives can include points, lines, polygons, pixels, text, and

Various types of curves and surfaces.

* **Attribute functions**

The result of pixel operation stage is either stored as texture memory for use in rasterization

Stage or rasterised and resulting fragment merged into the frame buffer just as they were generated

As they were generated from the geometric data.

**1.7 DATA TYPES**

OpenGL supports different data types. A list data types supported by OpenGL is given in the following

Table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SI no. | Suffix | Data type | C type | OpenGL type |
| 1. | B | 8 bit int | Signed int | GLbyte |
| 2. | S | 1 bit int | Short | GLshort |
| 3. | I | 32 bit int | Long | GLint, GLsizei |
| 4. | F | 32 bit float | Float | GLfloat, GLclampf |
| 5. | D | 64 bit float | Double | GLdouble,GLclamped |
| 6. | Ub | 8 bit unsigned | Unsigned char | GLubyte, GLboolean |
| 7. | Us | 16 bit unsigned | Unsigned short | GLushort |
| 8. | Ui | 32 bit unsigned | Unsigned int | GLuint, GLenum,GLbitfield |

**1.8 OBJECTIVES**

The objectives of this study are summarized below:

* To develop a OpenGL software called “PERPLEXITIY”.
* 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.

**CHAPTER 2**

**SYSTEM REQUIREENTS**

**2.1 SOFTWARE REQUIREMENTS**

1. Operating System: Microsoft Windows XP, Microsoft Windows 7

2. Compiler used: VC++6.0 compiler

3. Language used: Visual C++

**2.2 HARDWARE REQUIREMENTS**

1. Processor: Intel Core TM i3-32 bit

2. Processor Speed: 2.9 GHz

3. RAM Size: 8GB DDR3

4. Graphics – 2 GB

5. Cache Memory: 2MB

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 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.

**3.2 DISPLAY**

* Introduction page of “PREPLEXITY”
* Menus are created and depending on the value returned by menus.
* To navigate through the maze and complete the game within a minute then he wins the game either he loses the game.
* Suitable operations are performed.
* The operations performed are:
* New Game
* Instructions
* Quit

**3.3 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.

Start

HOME WINDOW

Press Enter

Main Window

Instructions

Quit

New Game

Escape

Stop

Press 1

Game Window

Instruction Window

II Time-Over?

Time Over Window

Win Window

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 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 jof 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.

**4.2 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.

**4.3 STRUCTURE**

* Void point();
* Void point1();
* Void point2();
* Void output(int x,int y,char

**4.4 ANALYSIS**

**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:

**4.4.1 myinit**

Void muinit()

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();glPointSizel(18.0);

glMatrixMode(GL\_MODELVIEW);

glClearColor(0.0,0.0,0.0,0.0);

}

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

**4.4.2 Display**

Void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

if(df==10)

frontscreen();

else if(df==0)

startscreen();

else if(df==1){

output(-21,172,”---->”);

output(-21,163,”<----“);

glColor3f(0.0,0.0,1.0);output(185,260,”TIME REMAINING:”);

drawstring(190,130,” HURRY

UP”,GLUT\_BITMAP\_HELVETICA\_18);

glColor3f(1,0,0);

drawstring(190,140,”Time is running

out”,GLUT\_BITMAP\_HELVETICA\_18); sprint(t,”%d”,60-count);

output(240,160,t);glutPostRedisplay();point();point1();

point2();

//line();

glColor3f(1.0,1.0,1.0);wall(-4,-4,0,-4,0,162,-4,162);

wall(8,162,8,158,0,158,0,162);

glutPostRedisplay();

}

else if(df==2)

instructions();

else if(df==3)

exit(1);

else if(df==4)

timeover();

else if(df==5)

winscreen();

glFlush();

}

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

4.4.3 Wall

Void wall(GLfloat x1,GLfloat y1,GLfloatx2,GLfloat y2,GLfloat x3,GLfloat y3,GLfloat x4,GLfloat y4){

glBegin(GL\_POLYGON);

glVertex3f(x1,y1,0);

glVertex3f(x2,y2,0);

glVertex3f(x3,y3,0);

glVertex3f(x4,y4,0);

glEnd();

}

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

**4.4.4 Point**

Void point(){

glColor3f(0.0,0.0,1.0);

glBegin(GL\_POINTS);

glVertex2f(px,py);glEnd();

}

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, land the player’s color point is blue which he uses to play the game.

**4.4.5 Frontscreen**

void frontscreen(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();glColor3f(1,1,1);

drawstring(120,5,” Press ENTER to go To next screen”, GLUT\_BITMAP\_HELVETICA\_18);

...............

...............

Drawstring(72,30,”(B.E.)”,GLUT\_BITMAP\_HELVETICA\_

12);output(70,20,”Lecture,Dept.of CSE”);glFlush();

}

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

**4.4.6 Idle**

Void idle()

{

If(df==1)

{

end=clock();

count=(end-start)/CLOCKS\_PER\_SEC;

if(count==60)

df=4;

else if(count<60)&&((px>=0&&px<=4)&&(py>=162&&py<=168)))

df=5;}

glutPostRedisplay();

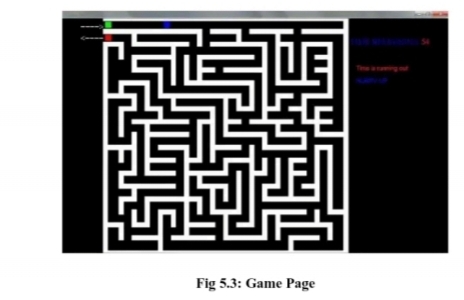
}

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 5**

**SNAPSHOTS**

****









**CONCLUSION**

**PERPLEXITY** is designed and implemented using a graphics software system called

**OpenGL** which has became a widely accepted standard for developing graphics 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.

**REFERENCE**

**BOOG REPERENCE:**

1. Donald Heam & Pauline Baker. Computer Graphics with openGL Version,3rd/ 4th Edition,

Pearson Education,2011

1. Edward Angel: Interactive Computer Graphics- A top Down approach with openGL, 5th

Edition. Pearson Education, 2008

1. James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with

openGL: pearson education

1. Xiang, Plastock: Computer Graphics, sham’s outline series,2nd edition, TMG.

**Web Reference:**

* <http://www.opengl.org/>
* <http://www.academictutprials.com/graphics-flood-fill.asp>
* <http://www.glprogramming.com/>
* <http://www.opengl.org/discussion_board/showthread.php/167379-Making-a-maze-using-array>