VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI

Mini Project Report on

" PERPLEXITY"

A SIMPLE MAZE GAME

Submitted in the partial fulfillment for the requirements of Computer Graphics & Visualization Laboratory of 6th semester CSE requirement in the form of the Mini Project work

Submitted By

KEVIN BARNES	**USN:1BY20CS085**
KEVIN MATHEW	**USN:1BY20CS086**
NISAR AHMED P	**USN: 1BY21CS412**

Under the guidance of

Mr. SHANKAR R
Assistant Professor
Dept. of CSE

Dr. SUNANDA DIXIT
Associate Professor
Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

YELAHANKA, BENGALURU - 560064.

2022-2023

BMS INSTITUTE OF TECHNOLOGY &MANAGEMENT

YELAHANKA, BENGALURU – 560064

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project work entitled **"A SIMPLE MAZE GAME"** is a bonafide work carried out by **Kevin Barnes(1BY20CS085) ,Kevin Mathew(1BY20CS086) and Nisar Ahmed P(1BY21CS412)** in partial fulfillment for *Mini Project* during the year 2022-2023. It is hereby certified that this project covers the concepts of *Computer Graphics & Visualization*. It is also certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report.

Signature of the Guide with date
Mr. SHANKAR R
Assistant Professor
CSE, BMSIT&M

Signature of the Guide with date
Dr. SUNANDA DIXIT
Associate Professor
CSE, BMSIT&M

Signature of HOD with date
Dr. Thippeswamy G
Prof & Head
CSE, BMSIT&M

INSTITUTE VISION

To emerge as one of the finest technical institutions of higher learning, to develop engineering professionals who are technically competent, ethical and environment friendly for betterment of the society.

INSTITUTE MISSION

Accomplish stimulating learning environment through high quality academic instruction, innovation and industry-institute interface.

DEPARTMENT VISION

To develop technical professionals acquainted with recent trends and technologies of computer science to serve as valuable resource for the nation/society.

DEPARTMENT MISSION

Facilitating and exposing the students to various learning opportunities through dedicated academic teaching, guidance and monitoring.

PROGRAM EDUCATIONAL OBJECTIVES

1. Lead a successful career by designing, analysing and solving various problems in the field of Computer Science & Engineering.
2. Pursue higher studies for enduring edification.
3. Exhibit professional and team building attitude along with effective communication.
4. Identify and provide solutions for sustainable environmental development.

ACKNOWLEDGEMENT

We are happy to present this project after completing it successfully. This project would not have been possible without the guidance, assistance and suggestions of many individuals. We wouldlike to express our deep sense of gratitude and indebtedness to each and every one who hashelped us make this project a success.

We heartily thank our Principal, **Dr. MOHAN BABU G N**, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We heartily thank our Professor and Head of the Department, **Dr. THIPPESWAMY G**, Department of Computer Science and Engineering, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We gracefully thank our Project Guide, **Mr. SHANKAR R**, Assistant Professor and **Dr. SUNANDA DIXIT,** Associate Professor, Department of Computer Science and Engineering for his intangible support and for being constant backbone for our project.

Special thanks to all the staff members of Computer Science Department for their help and kind co-operation.

Lastly, we thank our parents and friends for the support and encouragement given throughout in completing this precious work successfully.

KEVIN BARNES(1BY20CS085)

KEVIN MATHEW(1BY20CS086)

NISAR AHMED P(1BY21CS412)

ABSTRACT

The main aim of **A SIMPLE MAZE GAME** mini project is to provide an immersive and interactive experience for players as they navigate through a challenging maze environment.

The maze game utilizes 2D graphics and features a first-person perspective, allowing players to explore the maze from their character's viewpoint. OpenGL's capabilities are leveraged to render the maze and its surrounding environment, providing realistic lighting, textures, and smooth animations.The game includes various gameplay elements to engage players. The objective is to guide the character through the maze, overcoming obstacles and avoiding dead-ends to reach the exit point. Players must rely on their problem-solving skills, spatial awareness, and agility to successfully complete the maze within the shortest time possible.To enhance the gaming experience, sound effects and background music are incorporated using OpenGL's audio capabilities. These auditory cues provide feedback and add to the overall immersion and excitement of the gameplay.

The maze game is developed with simplicity in mind, making it accessible to a wide range of players. The controls are intuitive and responsive, allowing players to navigate the maze smoothly. The game supports different difficulty levels, providing options for both casual and more experienced players.

TABLE OF CONTENTS

CHAPTER 1

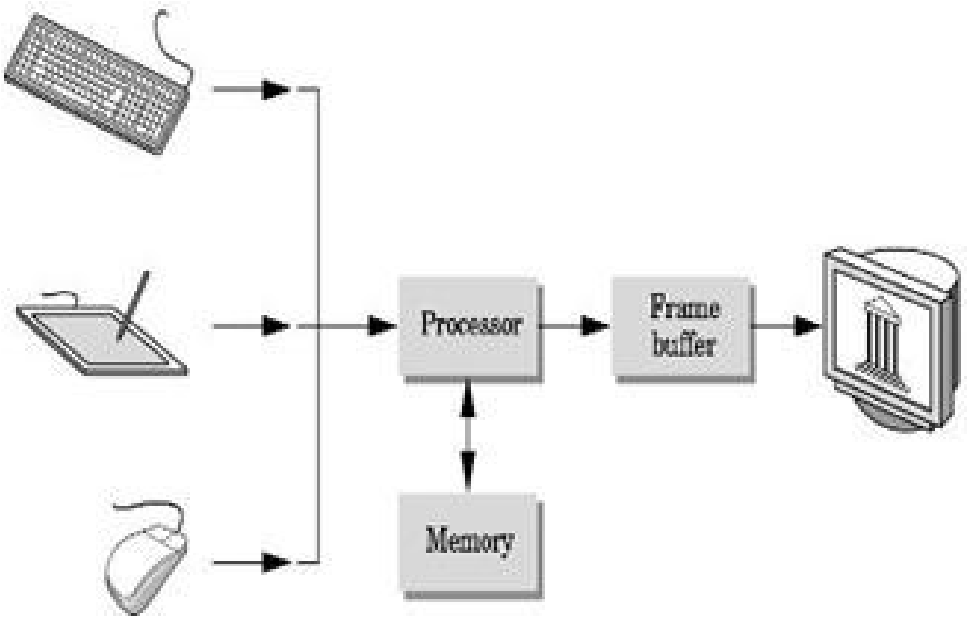
INTRODUCTION

1.1 Computer Graphics

Computer graphics are graphics created using computers and, more generally, the representation and manipulation of image data by a computer hardware and software. The development of computer graphics, or simply referred to as CG, has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized the animation and video game industry. 2D computer graphics are digital images—mostly from two-dimensional models, such as 2D geometric models, text (vector array), and 2D data. 3D computer graphics in contrast to 2D computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering images.

The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touchscreen. Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics.

The advantages of the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently.In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations. OpenGL is widely used in various fields, including video games, computer-aided design (CAD), scientific visualization, virtual reality, and more. It provides a powerful and flexible framework for creating high-performance graphics applications, enabling developers to leverage the capabilities of modern graphics hardware.



BASIC GRAPHICS SYTEM

1.2 OpenGL

OpenGL is the most extensively documented 3D graphics API (Application Program Interface) to date. Information regarding OpenGL is all over the Web and in print. It is impossible to exhaustively list all sources of OpenGL information. OpenGL programs are typically written in C and C++. One can also program OpenGL from Delphi (a Pascal-like language), Basic, Fortran, Ada, and other languages. To compile and link OpenGL programs, one will need OpenGL header files. To run OpenGL programs one may need shared or dynamically loaded OpenGL libraries, or a vendor-specific OpenGL Installable Client Driver (ICD).

OpenGL is a low-level graphics library specification. It makes available to the programmer a small set of geometric primitives - points, lines, polygons, images, and bitmaps. OpenGL provides a set of commands that allow the specification of geometric objects in two or three dimensions, using the provided primitives, together with commands that control how these objects are rendered (drawn).

1.3 GLUT

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres, and cylinders. GLUT even has some limited support for creating pop-up menus. The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is crossplatform) and to make learning OpenGL easier. All GLUT functions start with the glut prefix (for example, glutPostRedisplay marks the current window as needing to be redrawn).

GLUT's ease of use makes it an accessible choice for beginners in computer graphics programming.It is widely used and has a large community of users and resources available for support.GLUT has a straightforward API and can be easily integrated into existing OpenGL projects.It offers a range of utility functions that aid in the creation of interactive and visually appealing applications.GLUT's simplicity and cross-platform compatibility make it a useful tool for learning and prototyping in computer graphics. GLUT includes timer functions for scheduling tasks and updating animations at specified intervals.

1.4 GRAPHICS FUNCTIONS

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

- **Primitive function** 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** If primitives are the what of an API – the primitive objects that can be displayedthen attributes are the how. That is, the attributes govern the way the primitive appears on the display. Attribute functions allow us to perform operations ranging from choosing the color with which we display a line segment, to picking a pattern with which to fill inside of a polygon.

- **Viewing functions** The viewing functions allow us to specify various views, although APIs differ in the degree of flexibility they provide in choosing a view.

- **Transformation functions** One of the characteristics of a good API is that it provides the user with a set of transformations functions such as rotation, translation and scaling.

- **Input functions** For interactive applications, an API must provide a set of input functions, to allow users to deal with the diverse forms of input that characterize modern graphics systems. We need functions to deal with devices such as keyboards, mice and data tablets.

- **Control functions** These functions enable us to communicate with the window system, to initialize our programs, and to deal with any errors that take place during the execution of our programs.

- **Query functions** If we are to write device independent programs, we should expect the implementation of the API to take care of the differences between devices, such as how many colors are supported or the size of the display. Such information of the particular implementation should be provides through a set of query functions.

CHAPTER 2

LITERATURE SURVEY

1."OpenGL SuperBible: Comprehensive Tutorial and Reference" by Graham Sellers, Richard S. Wright Jr., and Nicholas Haemel: This book provides an in-depth introduction to OpenGL programming and covers various topics, including 2D and 3D graphics, shaders, and rendering techniques. It serves as a valuable resource for understanding the fundamentals of OpenGL necessary for developing a maze game.

2."Interactive Computer Graphics: A Top-Down Approach with WebGL" by Edward Angel and Dave Shreiner: This textbook focuses on interactive computer graphics using WebGL, a JavaScript API based on OpenGL ES. It covers essential concepts such as transformations, shading, texture mapping, and geometric modeling. The book provides practical examples and exercises that can be adapted to develop a maze game using OpenGL.

3."Computer Graphics with OpenGL" by Donald D. Hearn, M. Pauline Baker, and Warren Carithers: This book offers a comprehensive introduction to computer graphics using OpenGL. The book also includes a chapter on game development, which can provide insights into implementing game logic and controls for a maze game.

4."Beginning OpenGL Game Programming" by Dave Astle and Kevin Hawkins: This book focuses specifically on game development using OpenGL. It covers essential topics such as input handling, collision detection, and game physics. While it may not directly address maze game development, it provides valuable insights into the overall game development process that can be applied to creating a maze game.

4."Game Programming Patterns" by Robert Nystrom: This book explores various design patterns and principles used in game development. It covers topics such as the game loop, state management, and entity-component systems. Understanding these patterns can help in structuring the codebase and implementing game mechanics efficiently for a maze game.

Online tutorials and resources: Numerous online tutorials and resources are available that provide step-by-step guides on maze game development using OpenGL. Websites such as LearnOpenGL (https://learnopengl.com/) and OpenGL-Tutorial (http://www.opengl-tutorial.org/) offer comprehensive tutorials on graphics programming with OpenGL, which can be adapted to create a maze game.

CHAPTER 3

PROJECT IN DETAIL

3.1 Problem Statement

Computer graphics is no longer a rarity. It is an integral part of all computer user interfaces, and is indispensable for visualizing 2D; 3D and higher dimensional objects .Creating 3D objects, rotations and any other manipulations are laborious process with graphics implementation using text editor. OpenGL provides more features for developing 3D objects with few lines by built in functions. The geometric objects are the building blocks of any individual .Thereby developing, manipulating, applying any transformation, rotation, scaling on them is the major task of any image development.

So in this project the main problem statement is to design a game with help of computer graphics that is using Open GL. We will be implementing the **A simple maze game** using the concepts of Open GL

3.2 Brief Introduction

"Perplexity" is a simple maze game that challenges players to navigate through a series of intricate mazes, testing their problem-solving skills and spatial awareness. The game aims to provide an engaging and immersive experience by presenting players with increasingly complex mazes to explore.

In Perplexity, players assume the role of a character trapped within a labyrinthine world. Their objective is to find the exit point of each maze, which is often hidden behind multiple paths, dead-ends, and obstacles. As players progress through the game, the mazes become more challenging, requiring careful observation and decision-making to reach the goal.To enhance the gameplay experience, Perplexity incorporates visually appealing graphics using the OpenGL library. Realistic lighting, textures, and smooth animations are utilized to create an immersive environment for players to navigate. The game also includes sound effects and background music to provide audio feedback and enhance the overall atmosphere.The controls in Perplexity are designed to be intuitive and responsive. Players can typically move their character using keyboard inputs or mouse controls, allowing them to explore the maze from a first-person perspective. The game may also include additional gameplay elements

such as collectible items, keys, or timed challenges to add variety and depth to the gameplay.Perplexity offers different difficulty levels to cater to a wide range of players. Beginners can enjoy simpler mazes with fewer obstacles, while experienced players can test their skills with more intricate and challenging layouts. The game may also include a scoring system or time tracking feature to encourage replayability and competition among players.

Overall, Perplexity provides a captivating and immersive maze-solving experience. By combining engaging gameplay mechanics, visually appealing graphics, and an element of challenge, the game aims to entertain and captivate players as they navigate through the perplexing mazes in search of the exit.

3.3 Scope and Motivation

The scope of "Perplexity" encompasses the development of a standalone, single-player maze game with a focus on simplicity and engaging gameplay. The game will feature a variety of mazes, each with increasing difficulty levels, and will be developed using the OpenGL graphics library. The game will primarily target desktop platforms, such as Windows, macOS, and Linux.

The motivation behind creating "Perplexity" as a simple maze game lies in providing players with an enjoyable and immersive gaming experience. The maze genre has long been popular among gamers due to its challenging nature and ability to test problem-solving skills. The motivation can be further elaborated as follows:

Entertainment and Engagement: "Perplexity" aims to captivate players by offering a game that is easy to pick up and play, providing a source of entertainment and engagement. The challenging mazes, along with visually appealing graphics and sound effects, seek to create an immersive experience for players.

Stimulating Problem Solving: The maze-solving element in "Perplexity" stimulates players' problem-solving skills and critical thinking. Players will need to analyze the maze layout, plan their routes, and make strategic decisions to navigate through the intricate pathways and reach the exit.

Relaxation and Stress Relief: Maze games often provide a relaxing and immersive experience, allowing players to escape from the real world for a while. By offering a simple and intuitive gameplay mechanic, "Perplexity" aims to provide a calming and enjoyable experience that helps players unwind.

Replayability and Competition: By incorporating different difficulty levels, collectibles, and time-tracking features, "Perplexity" seeks to offer replayability and encourage friendly competition among players. Players can challenge themselves to improve their maze-solving skills and compete for high scores or faster completion times.

Showcase OpenGL Capabilities: Developing "Perplexity" using the OpenGL graphics library allows for the utilization of its powerful rendering capabilities, lighting effects, and smooth animations. The game can serve as a demonstration of the potential and versatility of OpenGL in creating visually appealing and interactive gaming experiences.

Overall, the motivation behind "Perplexity" is to create a simple maze game that provides entertainment, engages players in problem-solving, and offers a relaxing and immersive gaming experience. By leveraging the capabilities of the OpenGL library, the game aims to deliver visually appealing graphics and intuitive gameplay that captivate players and encourage replayability.

3.4 Proposed System

The proposed system for a simple maze game using OpenGL involves the development of a maze environment with interactive controls, graphics rendering, and gameplay mechanics. The system aims to provide an immersive and engaging experience for players. Here is an overview of the proposed system components:

Graphics Rendering:

- ✓ Utilize the OpenGL library for rendering 2D graphics.
- ✓ Implement lighting effects to create a realistic and visually appealing environment.
- ✓ Apply textures to walls, floors, and other elements to enhance the visual experience.
- ✓ Implement smooth animations for character movement and camera control.

Maze Generation:

- ✓ Generate random mazes using algorithms like depth-first search or Prim's algorithm.
- ✓ Define maze structure using data structures such as grids or graphs.
- ✓ Ensure that the maze has a clear starting point and an exit point.

Character Control:

- ✓ Implement intuitive controls for character movement using keyboard or mouse inputs.
- ✓ Allow the player to navigate through the maze in a first-person perspective.

- ✓ Ensure responsive and smooth character movement within the maze.

Difficulty Levels:

- ✓ Implement different difficulty levels, varying maze complexity and size.
- ✓ Allow players to choose the desired difficulty level before starting the game.

User Interface:

- ✓ Design an intuitive user interface for game menus, settings, and in-game displays.
- ✓ Include options for adjusting graphics settings and controls.

By implementing these components, the proposed system can create a simple maze game using OpenGL that provides an enjoyable and immersive gaming experience. Players will navigate through the maze, overcoming obstacles, and strive to complete the maze in the shortest time possible.

3.5 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.

CHAPTER 4

SYSTEM 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 myinit()
 {
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity(); glPointSize(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(185,160,"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);
 sprintf(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 called, 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,GLfloat x2,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.

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, and the player's color point is blue which he uses to play the game

CHAPTER 5

REQUIREMENT SPECIFICATION

5.1 Software Requirements

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

2. Compiler used: VC++ 6.0 compiler

3. Language used: Visual C++

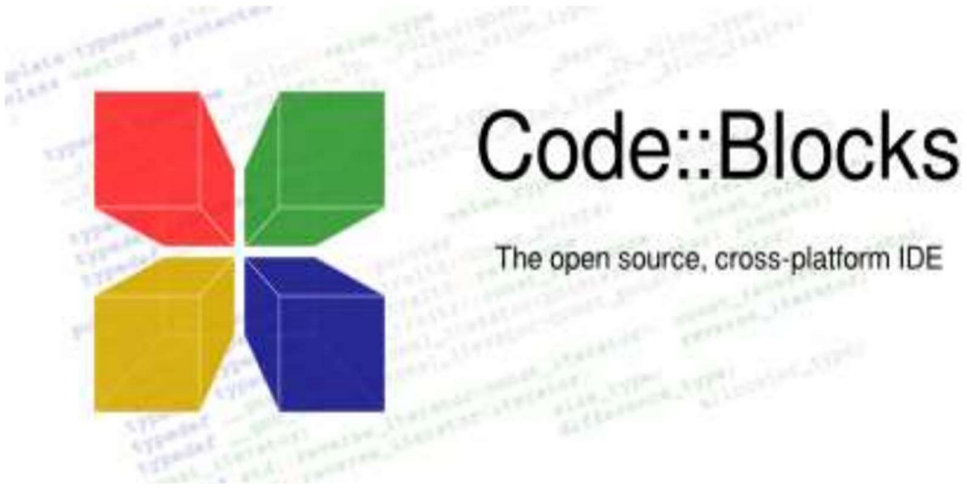
5.2 Hardware Requirements

1. Processor: Intel® CoreTM i3-32 bit

2. Processor Speed: 2.9 GHz

3. RAM Size: 8GB DDR3

4. Graphics – 2GB 5. Cache Memory: 2MB



CHAPTER 6

DESIGN AND IMPLEMENTATION

6.1 Design

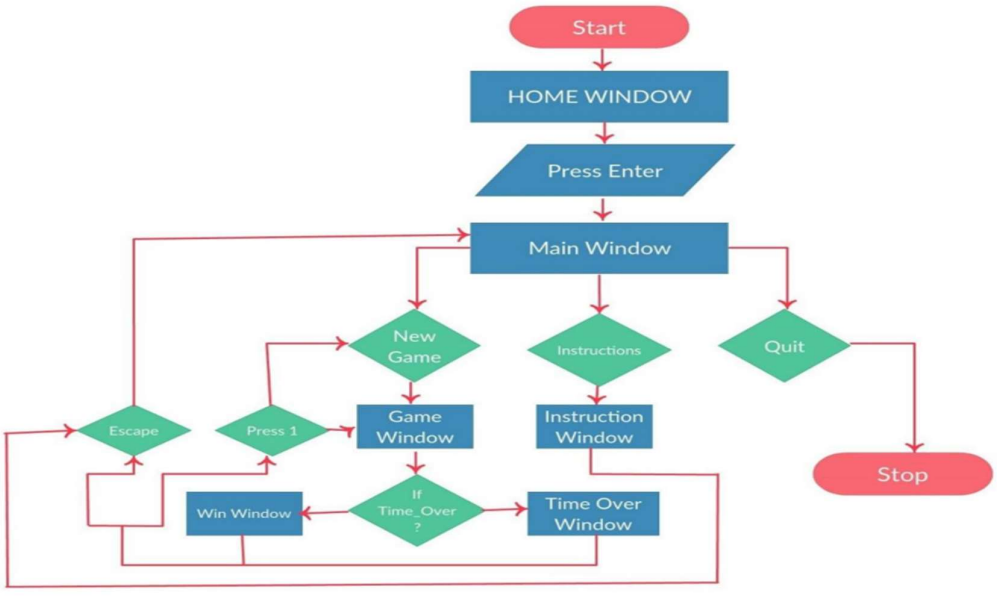
Initialization

- ✓ 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.

Display

- ✓ Introduction page of "PREPLEXITY"
- ✓ Menus are created and depending on the value returned by menus.
- ✓ Suitable operations are performed.
- ✓ The operations performed are:
    - New Game
    - Instructions
    - Quit

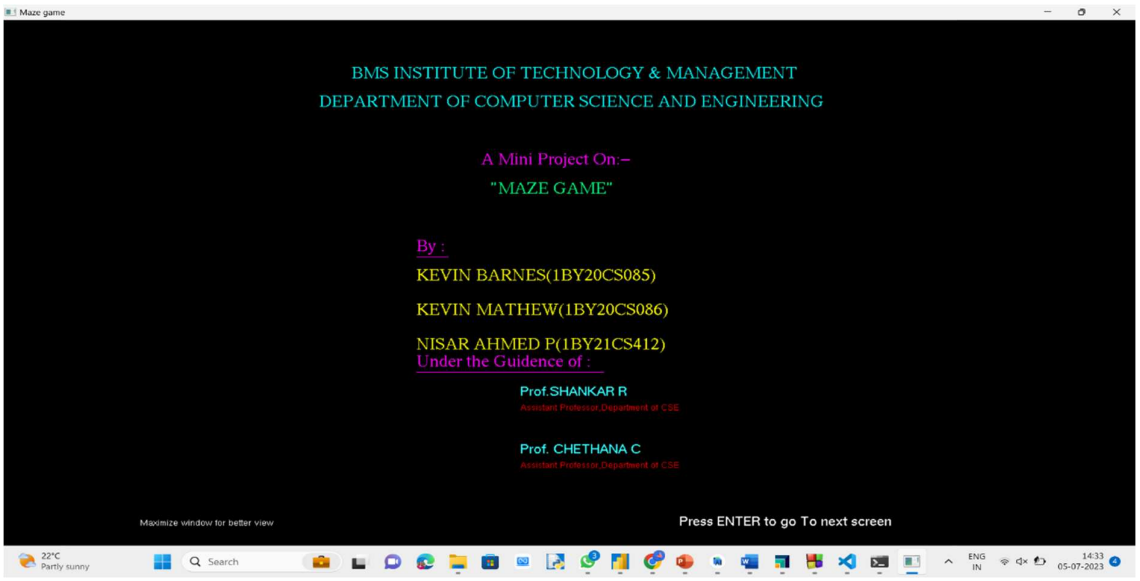
FlowChart



6.2 Implementation

This section speaks about the implementation of the project via the snapshots. It gives the detailed description of the way in which various scenes and frames are implemented.

1.After running the program

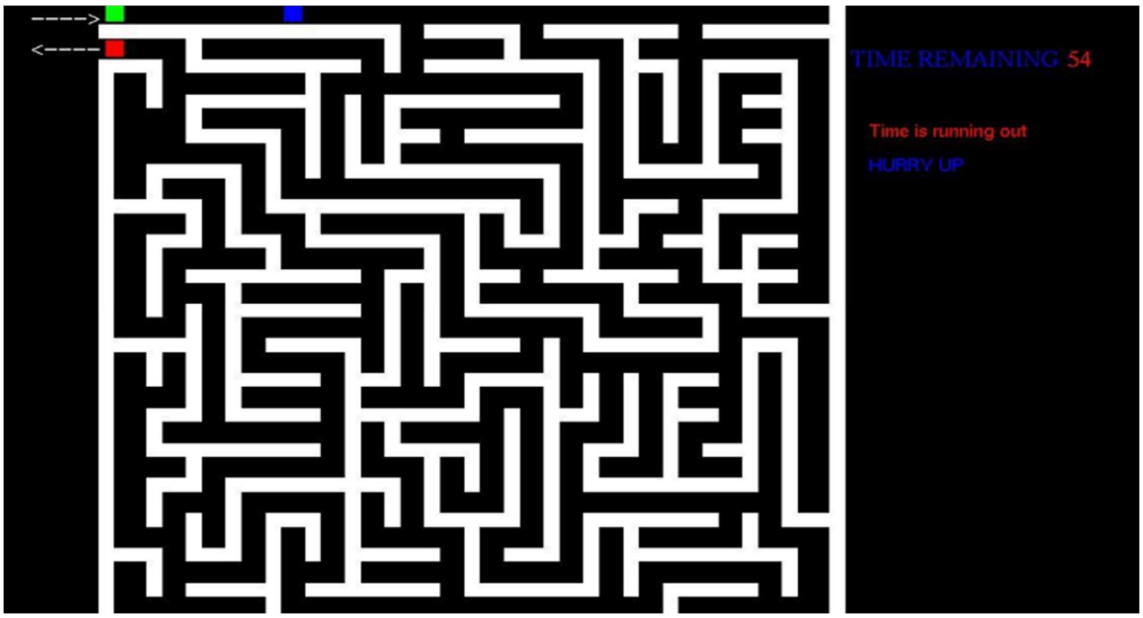


The above snapshot shows the screen displayed when the program gets Executed.

2.Game Menu

The above snapshot shows the Game Menu 1st one to start the game , 2nd option guides the player how to use the game & 3rd option Exits the game.

3.The Game



The above snapshot our Maze Game with green mark as the starting point & red mark is the end point & the player uses the blue block

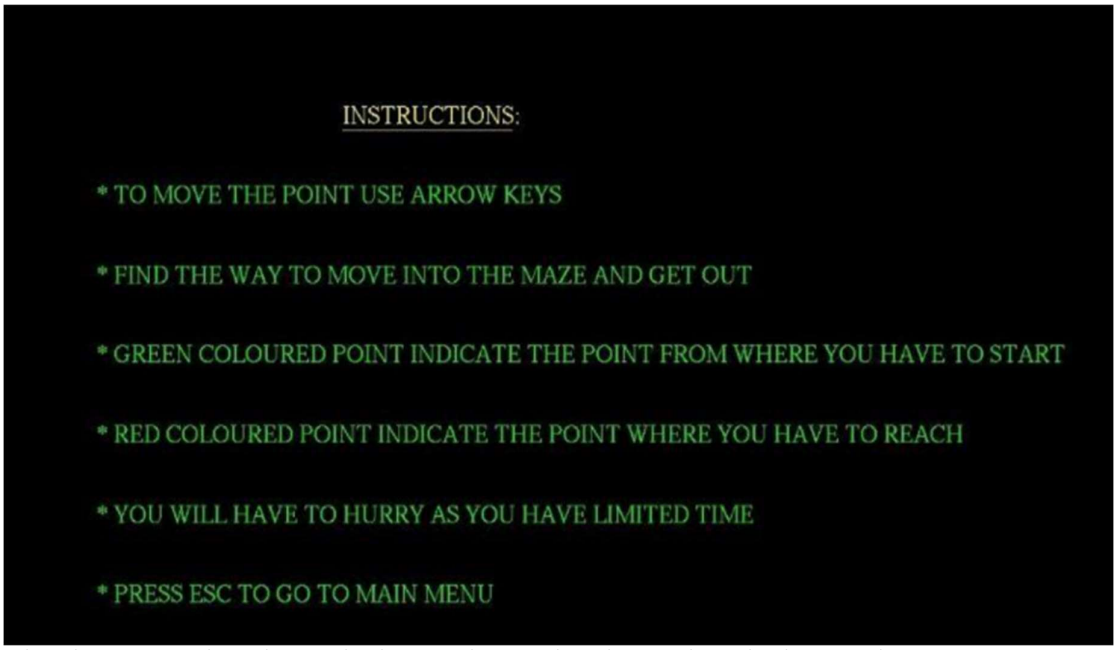
.4.Win Screen

The above snapshot shows the win screen as the player has finished the game within 60 Sec.

5.Game Over(Lost!!!!)

The above snapshot shows the lost screen as the player has not finished the game within 60 Sec.

6.Instructions



The above snapshot shows the instruction to the player , how he has to play game

CONCLUSION

PERPLEXITY 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

3. http://www.glprogramming.com/

4. https://www.opengl.org/discussion_boards/showthread.php/167379-Making-a-mazeusing-arrays