CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

A maze is a kind of games where a player moves in pathways with many branches to find a way out/certain targets. Maze and labyrinth are two different things. The labyrinth is not as complicated maze: labyrinth's path that a player can tread has no branching at all. But usually, people consider maze and labyrinth are the same. Of the diverse types of mazes, the one commonly encountered by people is the perfect maze type. Perfect here means that the maze does not have not only loops/circular paths, but also insulated parts which no other parts can access to. Figure 1 shows an example of a perfect maze layout. Maze-making process is the process of designing the position of paths and walls in the maze. There are many methods/algorithms to generate a maze, but essentially these methods can be categorized into two groups: step-by-step creating a path in the maze that is wholly covered by walls or building walls on the empty field/space. In both methods, random values are used in determining the next path or wall which is going to be built. Because of the randomness and the steps taken in the methods, computers are an appropriate tool for building mazes, since computers not only have the abilities in creating and using random values easily but also can execute certain steps of algorithm repeatedly without being exhausted.

Therefore are many methods to create a perfect maze layout. Depth First Search, Prim, Eller, Kruskal, and Aldous-Broder are some of them. Each has advantages and disadvantages . Many maze games have been created and explored: building dynamic maze in applet environment, making a social maze game to encourage training for multiple sclerosis patients, creating a maze based mobile application for learning foreign language, using 3D OpenGL ES gravity maze game to develop and apply built-in sensors in Android mobile phones, constructing maze games using an Android game framework as part of teaching programming for college students, and many other research that have a similar topic. Mobile devices have been growing rapidly during the last decade. Computational power of mobile devices has increased considerably. This capability is needed by the operating system and various applications (e.g. 3D real-time games) to provide a rich and attractive user experience. Android, as one

of many operating systems used in mobile devices, has diverse game applications built on it. Several of them explore the serious side of game such as first aid education for Autism Spectrum Disorder people, and teaching forestry lessons in a quiz game. In gaming, maze-making process is part of Procedural Content Generation field, since it creates the level/world to be played on the fly and in random manner by following a procedure/algorithm with no human intervention. One of the advantages of this technique, compared to the manual making technique, is that the game makers do not need to create detailed assets for each part of the game. Game manufacturers just need to write a procedure for making the level once, and then the procedure is the one that will be making different kind of levels in relatively unlimited quantities. Each level generated can also be made such that the difficulty is different: either increasing or decreasing. In this study, an app/maze game that can generate maze layouts dynamically on Android mobile devices was built. What is meant by the dynamic here is that the maze will be re-created each time the player enters the game levels. The process of making paths and walls of the maze occurs randomly, so the chance of a player getting the exact same maze in succession is small. The method used is the Growing Tree method. This application is also used to measure the level of performance the mobile device has in solving problem.

CHAPTER 2

SYSTEM REQUIREMENTS

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™ i3-32 bit
2. Processor Speed: 2.9 GHz
3. RAM Size: 8GB DDR3
4. Graphics – 2GB
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.
- 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.

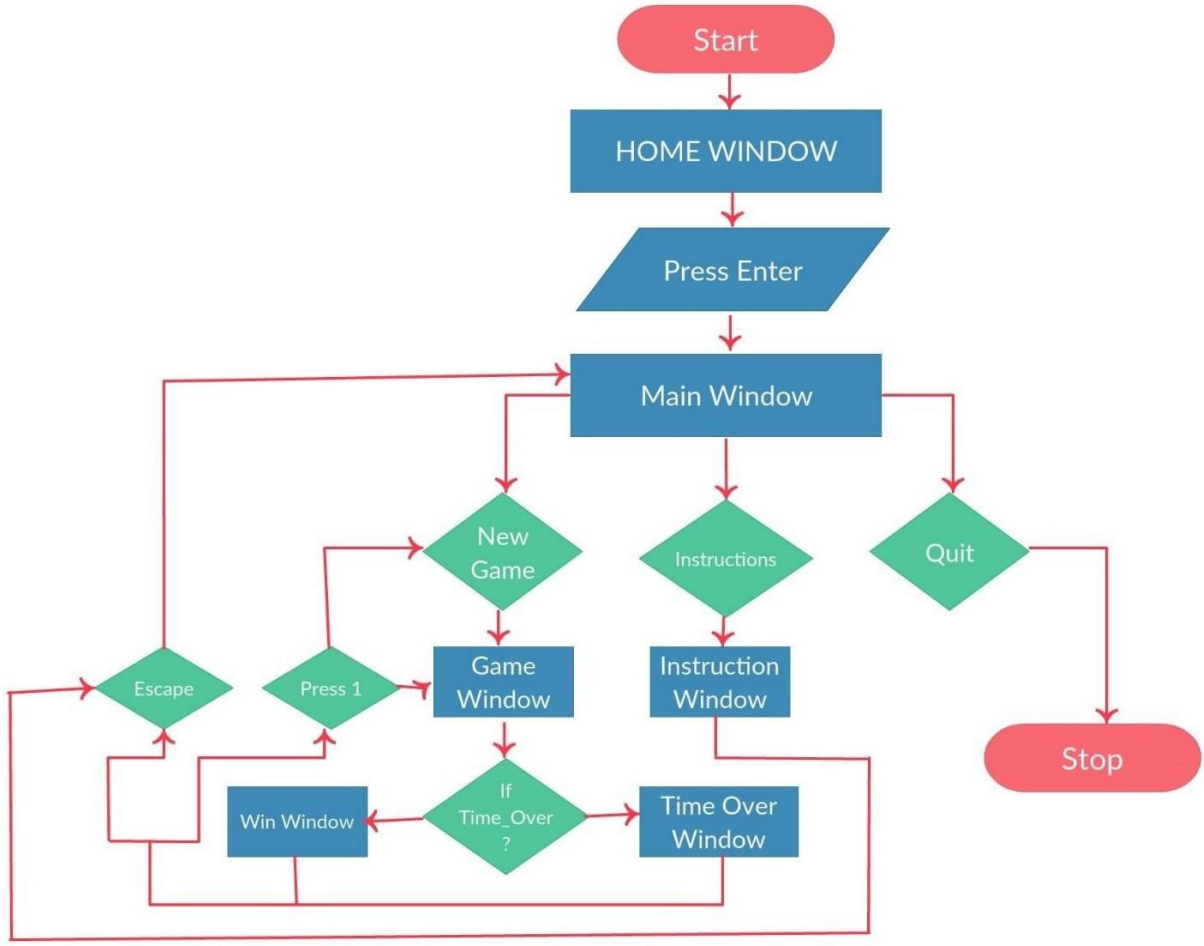

Fig: 3.3 Flow Chart

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 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.

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 *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 myinit();
- void myreshape(int w,int h);
- int main(int argc,char** argv);

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 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.

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, and 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,"Lecturer,Dept. of CSE");
        glFlush();
}
```

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.

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

1. After running the program

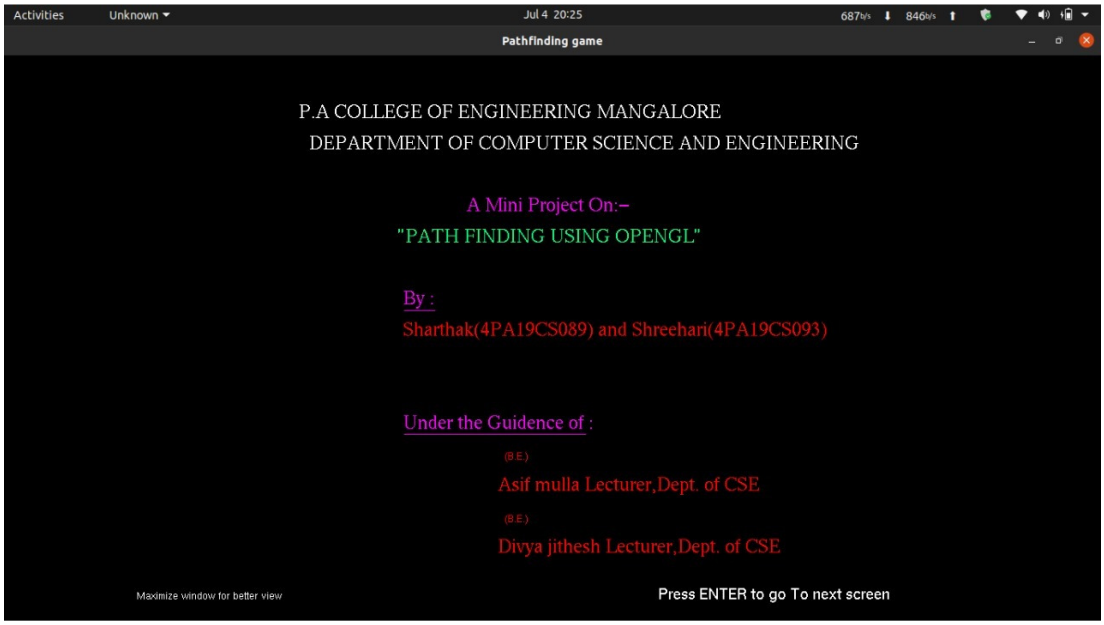

Fig 5.1: Home page

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

2. Game Menu

Fig 5.2: Main Window

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

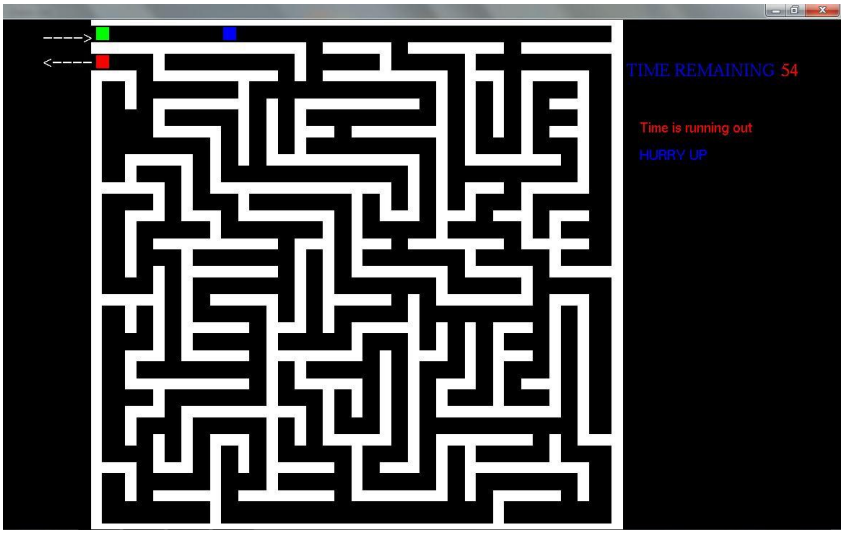

Fig 5.3: Game Page

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

Fig 5.4: Win Page.

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

5. Game Over (Lost!!)

Fig 5.5: Game Over Page

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

6. Instructions

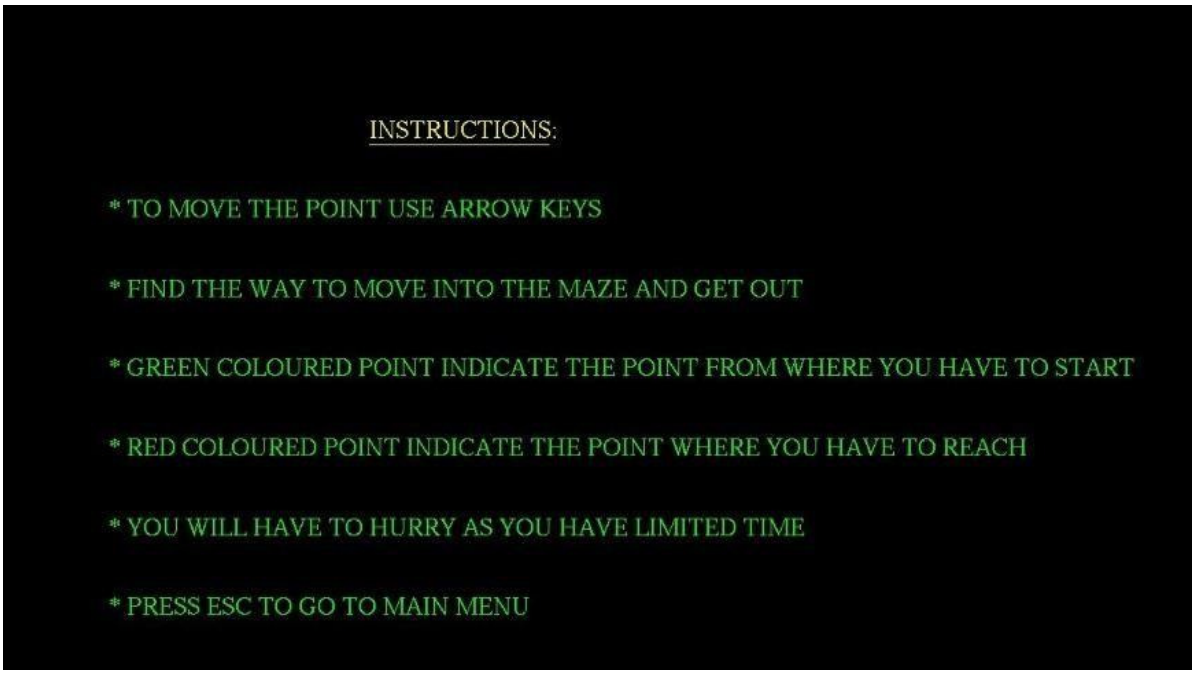

Fig 5.6: Instruction Page

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

CHAPTER 6

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.

REFERENCES

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-maze-using-arrays