VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELGAUM

A MINI PROJECT REPORT ON

"Analog clock using OpenGL"

Submitted in partial fulfilment of the requirements

For the award of the 6th sem of

BACHELOR OF ENGINEERING

COMPUTER SCIENCE ENGINEERING

SUBMITTED BY:

AKASH RP **[1VE19CS013]**

AMBARISH BN **[1VE19CS015]**

Under the guidance of

Mr. ABHILASH DC

Asst. Prof, Dept. of CSE

SRI VENKATESHWARA COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BANGALORE-562157

SRI VENKATESHWARA COLLEGE OF ENGINEERING

Kempegowda International Airport Bengaluru, Road, Kempegowda Int'l Airport Rd, Vidya Nagar, Central Telecom Society, Bengaluru, Karnataka 562157

Accredited by NBA*
SVCE BENGALURU
SRI VENKATESHWARA COLLEGE OF ENGINEERING
Affiliated to VTU, Approved by AICTE, Recognised by UGC u/s 2(f) & 12(B)

CERTIFICATE

This is to certify that Mini-project work entitled **"ANALOG CLOCK USING OPENGL"** submitted in partial fulfilment of the requirement for VI semester Bachelor of Engineering in Computer Science & Engineering prescribed by the **Visvesvaraya Technological University, Belgaum** is a result of the bonafide work carried out by **AKASH RP [1VE19CS013]** and **AMBARISH BN [1VE19CS016]** during the academic year **2021-22.** It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in theReport deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

.....	.....
Signature of Course Teacher	**Signature of HOD**
Mr. Abhilash DC	**Dr. S C Lingareddy**

Name of the examiners:	**Signature with date**
1:	**.....**
2:	**.....**

ACKNOWLEDGEMENT

At the various stages in doing this project, Many number of people have given us invaluable comments on the manuscript. We take this opportunity to express our deepest gratitude and appreciation to all those who helped us directly or indirectly towards the successful completion of this project.

I would like to express my immense gratitude to the Principal, For his help and inspiration during the tenure of the course.

I also extend my sincere thanks to the HOD of the Computer Science and Engineering department for his inspiration during the completion of the project.

In this regard, I owe a heartfelt gratitude to my Guide in the Computer Science and Engineering Department for the timely advice on the project and regular assistance throughout the project work. I would also like to thank my lecturers for their cooperation.

Finally, I thank our parents and friends for their moral support.

Akash RP (1VE19CS013)

Ambarish (1VE19CS015)

ABSTRACT

OpenGL is an emerging graphics standard that provides advanced rendering features while maintaining a simple programming model. Because OpenGL is rendering-only, it can be incorporated into any window system or can be used without a window system. An OpenGL implementation can efficiently accommodate almost any level of graphics hardware,from a basic frame buffer to the most sophisticated graphics subsystems. It is therefore a good choice for use in interactive 3D and 2D graphic applications.

We describe how these and other considerations have governed the selection and presentation of graphical operators in OpenGL. Complex operations have been eschewed in favour of simple, direct control over the fundamentals of 2D and 3D graphics. Higher level graphical functions may, however, be built from OpenGL's low-level operators, as the operators have been designed with such layering in mind.

In our project we are implementing a menu driven program animating the Analog clocks of different time zones using OpenGL implementation. In this project, we will tell how to make a clock structure by using OpenGL functions to display time and then use a menu driven program to display the time of your preferred zones. This program currently allows you to see the times from 6 different cities namely **New Delhi, New York, London, Tokyo, Sydney** and **Moscow**. This package is based on the OpenGL library functions. Our program will give the Analog clock with system time. The programming language used here is C++. We used various glut functions to create an environment in which we could perform such actions with our object.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

Computer graphics are graphics created using computers and more generally the representation and manipulation of image data by a computer. The term computer graphics has been used in a broad sense to describe “almost everything on computers that is not text or sound”.

The development of computer graphics has made computers easier to interact with, 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 revolutionised animation, movies and the video game industry.

The various applications of computer graphics are:

1. Display of information
2. Design
3. Simulation and animation
4. User interfaces

OpenGL (Open Graphics Library) is a standard specification defining a ross-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 visualisation, information visualisation, and flight simulation. It is also used in video games, where it competes with Direct3D on micros windows platforms. OpenGL is managed by the non-profit technology consortium, The Khronos Group.

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry’s most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.

OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other

powerful visualisation functions. Developers can leverage the power of OpenGLacross all popular desktop and workstation platforms, ensuring wide application deployment. It is the most widely adopted graphics standard.

Any visual computing application requiring maximum performance from 3D animation to CAD to visual simulation can exploit high-quality, high-performance OpenGL capabilities. These capabilities allow developers in diverse markets such as broadcasting, CAD/CAM/CAE, entertainment, medical imaging, and virtual reality to produce and display incredibly compelling 2D and 3D graphics.

OpenGL routines simplify the development of graphics software-from rendering a simple geometric point, line, or filled polygon to the creation of the most complex lighted and texture-mapped NURBS curved surface. OpenGL gives software developers access to geometric and image primitives, display lists, modelling transformation lighting.

OpenGL is an API similar to direct3D, However, OpenGL is not supported by Microsoft and is a cross-language, cross-platform API used for writing applications that produce 2D and 3D graphics. Over 300 functions are implemented in this API which can help you draw complex 3D scenes from primitive data such as polygonsand triangles. Mesa 3D is an open-source library API which is also compatible with the OpenGL protocol.

The project is to demonstrate a working analog clock using animation. We are implementing it using primitives available in the OpenGL library and combining them together in a required manner. It highlights the key features of the data structures and its high efficiency that is obtained on its usage in the application program. It illustrates the different call back functions that provide an easier way to accomplish our project in an effective manner.

1.2 Computer Graphics Overview

- Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D or 3D pattern recognition abilities allow us to perceive and process pictorial data rapidly.
- Computers have become a powerful medium for the rapid and economical production of pictures.
- Graphics provide a natural means of communicating with the computer that they have become widespread.
- Interactive graphics is the most important means of producing pictures since the invention of photography and television.
- We can make pictures of not only the real-world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.
- A computer graphics system is a computer system with all the components of the general-purpose computer system. There are five major elements in a system: input devices, processor, memory, frame buffer, output devices.

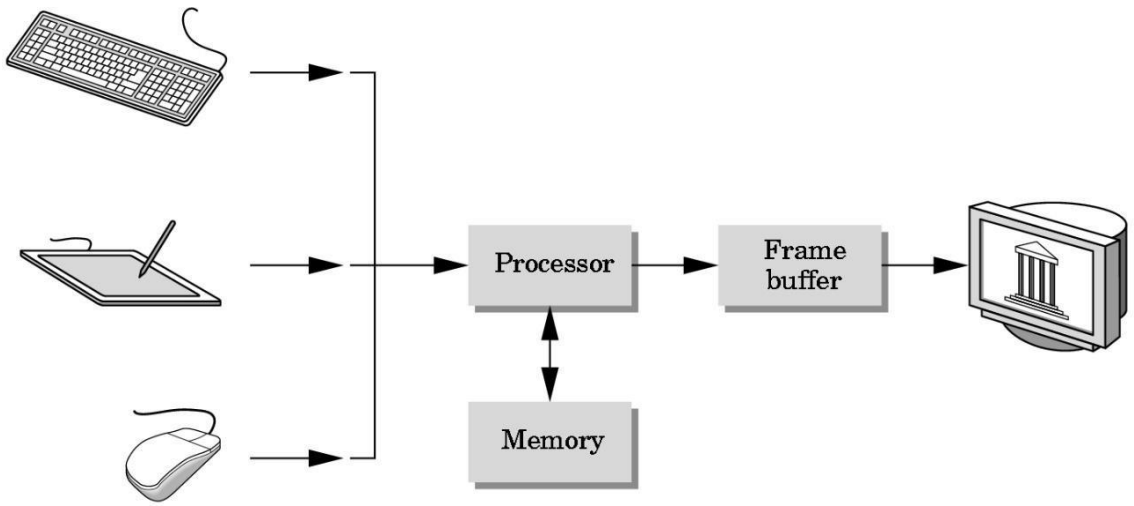

Fig 1.1: Graphics system.

1.3 OpenGL Technology

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.

OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

OpenGL Available Everywhere: Supported on all UNIX® workstations, and shipped standard with every Windows 95/98/2000/NT and MacOS PC, no other graphics API operates on a wider range of hardware platforms and software environments.

OpenGL runs on every major operating system including Mac OS, OS/2, UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, Open Step, and BeOS; it also works with every major windowing system, including Win32, MacOS, Presentation Manager, and X-Window System. OpenGL is callable from Ada, C, C++, FORTRAN, Python, Perl and Java and offers complete independence from network protocols and topologies.

1.4 The OpenGL Interface

Our application will be designed to access OpenGL directly through functions in three libraries namely: gl, glu, glut.

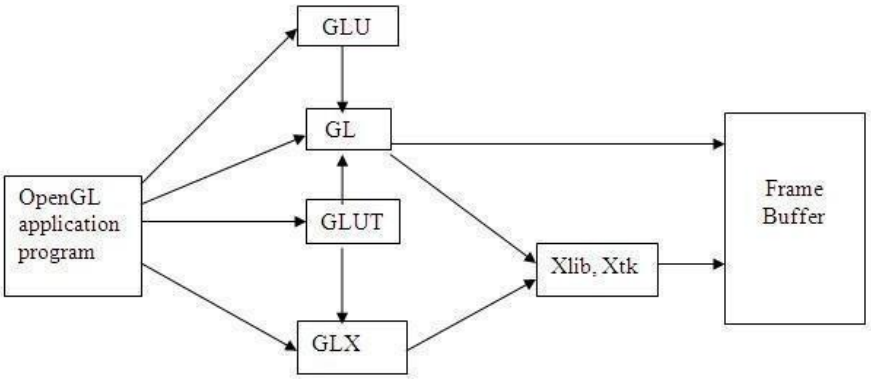

Fig 1.2: OpenGL interface.

Here, we are going to develop the basic structure of the Analog clock in computer graphics. In this, we will tell how to make a clock structure by using OpenGL functions to display time. This package is based on the OpenGL library functions. Our program will give the Analog clock with system time. The programming language used here is C++.

The main theme of this project is as follows:

- It is Interactive 2D graphics.
- Menu driven code to choose a time zone.
- The clock will be displayed.
- Time will be displayed based on the city you choose.

This project currently allows you to see the times from 6 different cities namely **New Delhi, New York, London, Tokyo, Sydney** and **Moscow** . The system time is taken and using the system time the times of the other cities are calculated.

1.5 Report Organization

The chapters included in this report give the detailed idea about the project and these are:

Chapter-1: This chapter consists of the introduction of concepts of Computer Graphics used in this project. It also includes the list of chapters included in the project.

Chapter-2: It specifies the requirements of the software and hardware for the program to run. It also describes the environment where this project can work.

Chapter-3: This chapter includes system design of the program.

Chapter-4: This chapter gives the overview of how the project is implemented and how the program will work with respect to the functions called iterations. It specifies the libraries used in the program. It also gives the functions that form the program with its description in the form of comments.

Chapter-5: This chapter has snapshots of the program output i.e. the view of the output on executing the program.

Chapter-6: The last chapter consists of the conclusion part of the report along with the future work of the project which gives information about the future use or modifications of the project.

CHAPTER 2

REQUIREMENT SPECIFICATION

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 behaviour of the system.

2.1 Functional Requirements

Functional Requirements defines the internal working of the software, i.e., the calculations, technical details, data manipulation and processing and other specific functionality that show how the cases are to be satisfied and how they are supported by non-functional requirements, which impose constraints on the design or the implementation.

The following must be taken care of:

- The ability to perform correct operation when the corresponding keys are pressed.
- When the corresponding menu is selected, the corresponding option should be performed.

2.2 Non-functional Requirements

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of the system, rather than specific behaviours. This should be contrasted with functional requirements that specify specific behaviour or functions. Typical non-functional requirements are reliability and scalability. Non-functional requirements are “constraints”, “quality attributes” and “quality of service requirements”.

2.2.1 Types of non-functional requirements

Volume: The volume of a system (the total space occupied) varies depending on how the component assemblies are arranged and connected.

Reliability: System reliability depends on component reliability but unexpected interactions can cause new types of failure and therefore affect the reliability of the system.

Security: The security of the system (it’s ability to resist attacks) is a complex property that cannot be easily measured. Attacks may be devised that were not anticipated by the system designers and so may default built-in safeguards.

Reparability: This property reflects how easy it is to fix a problem with the system once it has been discovered. It depends on being able to diagnose the problem, access the components that are faulty and modify or replace these components.

Usability: This property reflects how easy it is to use the system. It depends on the technical system components, its operators and its operating environment.

2.3 Software Requirements

1. OPERATING SYSTEM : Windows 98, Windows XP, Windows 7
2. FRONT END : Microsoft Visual C++ Version 6.0.
3. CODING LANGUAGE : C++.

2.4 Hardware Requirements

1. SYSTEM : Pentium IV 2.4 GHz or Above
2. HARD DISK : 40 GB, 80 GB, 160 GB or Above
3. RAM : 256 MB, 512 MB, 1 GB or Above

CHAPTER 3

DESIGN

Requirement analysis encompasses all the tasks that go into the instigation, scoping and definition of a new or altered system. Requirement analysis is an important part of the design process. Here we identify the needs or requirements. Once the requirements have been identified the solution for the requirements can be designed.

Design a project with specific goals, tasks and outcomes. The more specific, the better; the more closely aligned with traditional instructional objectives, the better. Avoid “sister school” and “pen pal” projects.

3.1 High Level Design

High Level Design (HLD) is the overall system design - covering the system architecture and database design. It describes the relation between various modules and functions of the system. data flow, flow charts and data structures are covered under HLD.

A series of processing stages is called the OpenGL rendering pipeline. This ordering is not a strict rule of how OpenGL is implemented but provides a reliable guide for predicting what OpenGL will do.

Geometric data (vertices, lines, and polygons) follow the path through the row of boxes that includes evaluators and per-vertex operations, while pixel data (pixels, images, and bitmaps) are treated differently for part of the process. Both types of data undergo the same final steps (rasterization and per-fragment operations) before the final pixel data is written into the frame buffer.

The different modules used in our project are as follows:

`RenderString()` : This function is used to call `glColor3ub()`, `glRasterPos2f()` and `glutBitmapString()` and to initialize them.

`circle()` : This function is used to draw the circle required for the clock.

`draw()` : This function is used to draw interior of the clock, fill in colour, to generate numbers starting from 1-12 and to finally generate hour, minute, and second arm.

This function also displays the name of the place on the clock whose time is displayed.

`resize()` : This function contains `glViewport()` , `glMatrixMode()`, `gluOrtho2D()` to set and adjust the viewport of the animation.

`sub()` : This function is used to subtract the time difference between IST and other time zones.

`add()` : This function is used to add the time difference between IST and other time zones.

`move()` : This function is used to calculate the movement of the three different arms present in the clock. The `add()` and `sub()` functions are used here to calculate the various hours and minutes of different time zones.

`main()` : This function used to set the prerequisites of the project.It is also used to take the input from the user for the menu driven segment of the code.

3.1 Low Level Design

Low Level Design (LLD) is like detailing the HLD. It defines the actual logic for each and every component of the system. Class diagrams with all the methods and relations between classes come under LLD. Program specs are covered under LLD.

CHAPTER 4

IMPLEMENTATION

Code::Blocks is a free, Open-source cross-platform IDE that supports multiple compilers including GCC, Clang and Visual C++. It is developed in C++ using wxWidgets as the GUI toolkit. Using a plugin architecture, It's capabilities and features are defined by the provided plugins. Currently, Code::Blocks is oriented towards C, C++, and Fortran. It has a custom build system and optional Make support.

Code::Blocks is being developed for Windows and Linux and has been ported to FreeBSD, OpenBSD and Solaris. The latest binary provided for macOS version is 13.12 released on 2013/12/26 (compatible with Mac OS X 10.6 and later), but more recent versions can be compiled and MacPorts supplies version 17.12..

After releasing two release candidate versions, 1.0rc1 on July 25, 2005 and 1.0rc2 on October 25, 2005, instead of making a final release, The project developers started adding many new features, with the final release being repeatedly postponed. Instead, there were nightly builds of the latest SVN version made available on a daily basis.

The first stable release was on February 28, 2008, with the version number changed to 8.02. The versioning scheme was changed to that of Ubuntu, with the major and minor number representing the year and month of the release. Version 20.03 is the latest stable release; however for the most up-to-date version the user can download the relatively stable nightly build or download the source code from SVN.

In April 2020, A critical software vulnerability was found in the Code::Blocks IDE, identified by CVE-2020-10814.

Jennic Limited distributes a version of Code::Blocks customized to work with its microcontrollers.

The IDE features syntax highlighting and code folding (through its Scintilla editor component), C++ code completion, class browser, a hex editor and many other utilities. Opened files are organized into tabs. The code editor supports font and font size selection andpersonalized syntax highlighting colours.

Code::Blocks supports multiple compilers, including GCC, MinGW, Digital Mars, Microsoft Visual C++, Borland C++, LLVM Clang, Watcom, LCC and the Intel C++ compiler.

Although the IDE was designed for the C++ language, there is some support for other languages, including Fortran and D. A plug-in system is included to support other programming languages.

The Code::Blocks debugger has full breakpoint support. It also allows the user to debug their program by having access to the local function symbol and argument display, user-defined watches, call stack, disassembly, custom memory dump, thread switching, CPU registers and GNU Debugger Interface.

Source code

```
#include <iostream>
#include <stdio.h>
#include <GL/gl.h>
#include <GL/glut.h>
#include <GL/freeglut.h>
#include <math.h>
#include  <time.h>
#include  <string>
#include <cstring>

const GLfloat tam_x = 50.0f;
const GLfloat tam_y = 50.0f;

const GLint sy = 30;
const GLint my = 25;
const GLint hy = 20;

int hour;
int minute;
int second;
int c;

void RenderString(float x, float y, void *font, const unsigned char
*str) {
      char *c;
      glColor3ub(183, 28, 28);
      glRasterPos2f(x, y);
      glutBitmapString(font, str);
}

void circle(GLfloat xc, GLfloat yc, GLfloat raio, bool fill) {
      const GLfloat c = 3.14169f / 180.0f;
      GLint i;

      glBegin(fill ? GL_TRIANGLE_FAN : GL_LINE_LOOP);

      for (i = 0; i <= 360; i += 2) {
          float a = i * c;
          glVertex2f(xc + sin(a) * raio, yc + cos(a) * raio);
      }
```

```
	glEnd();
}

void draw(void) {
	glClear(GL_COLOR_BUFFER_BIT);

	glColor3ub(249, 168, 37);
	circle(0, 0, tam_x, true);

	glColor3f(0.0f, 0.0f, 0.0f);
	unsigned char time_1[4] = "1";
	unsigned char time_2[4] = "2";
	unsigned char time_3[4] = "3";
	unsigned char time_4[4] = "4";
	unsigned char time_5[4] = "5";
	unsigned char time_6[4] = "6";
	unsigned char time_7[4] = "7";
	unsigned char time_8[4] = "8";
	unsigned char time_9[4] = "9";
	unsigned char time_10[4] = "10";
	unsigned char time_11[4] = "11";
	unsigned char time_12[4] = "12";

	RenderString(-2, 40, GLUT_BITMAP_TIMES_ROMAN_24, time_12);
	RenderString(0, -40, GLUT_BITMAP_TIMES_ROMAN_24, time_6);
	RenderString(40, 0, GLUT_BITMAP_TIMES_ROMAN_24, time_3);
	RenderString(-40, 0, GLUT_BITMAP_TIMES_ROMAN_24, time_9);

	RenderString(-35, 20, GLUT_BITMAP_TIMES_ROMAN_24, time_10);
	RenderString(35, 20, GLUT_BITMAP_TIMES_ROMAN_24, time_2);
	RenderString(35, -20, GLUT_BITMAP_TIMES_ROMAN_24, time_4);
	RenderString(-35, -20, GLUT_BITMAP_TIMES_ROMAN_24, time_8);

	RenderString(-20, 35, GLUT_BITMAP_TIMES_ROMAN_24, time_11);
	RenderString(20, 35, GLUT_BITMAP_TIMES_ROMAN_24, time_1);
	RenderString(20, -35, GLUT_BITMAP_TIMES_ROMAN_24, time_5);
	RenderString(-20, -35, GLUT_BITMAP_TIMES_ROMAN_24, time_7);

	glColor3ub(0, 0, 0);
	glRasterPos2f(-6, -20);
	const unsigned char ND[]="NEW DELHI";
	const unsigned char NY[]="NEW YORK";
	const unsigned char LD[]="LONDON";
```

```
const unsigned char TK[]="TOKYO";
const unsigned char SY[]="SYDNEY";
const unsigned char MW[]="MOSCOW";
const unsigned char DE[]="UTC";
const unsigned char ST[]="Stay tuned for more time zones.";

switch(c) {
case 1:
    glRasterPos2f(-7, -20);
    glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, ND);
    break;
case 2:
    glRasterPos2f(-7, -20);
    glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, NY);
    break;
case 3:
    glRasterPos2f(-5, -20);
    glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, LD);
    break;
case 4:
    glRasterPos2f(-4, -20);
    glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, TK);
    break;
case 5:
    glRasterPos2f(-5, -20);
    glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, SY);
    break;
case 6:
    glRasterPos2f(-5, -20);
    glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, MW);
    break;
default:
    glRasterPos2f(-3, -20);
    glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, DE);
    glRasterPos2f(-75, 40);
    glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, ST);
}

const unsigned char AK[]="Made by:  AKASH RP";
const unsigned char AM[]="       AMBARISH BN";
glRasterPos2f(50, -40);
glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, AL);
glRasterPos2f(56, -45);
```

```
    glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, AN);

    // Second
    float angle_s = second * 6;
    glRotatef(-angle_s, 0.0f, 0.0f, 1.0f);

    glBegin(GL_LINES);
    glColor3f(1.0f, 1.0f, 1.0f);
    glVertex2i(0, 0);
    glVertex2i(0, sy);
    glEnd();

    glLoadIdentity();

    //minute
    float angle_m = minute * 6;

    glRotatef(-angle_m, 0.0f, 0.0f, 1.0f);

    glLineWidth(5);
    glBegin(GL_LINES);
    glColor3f(0.0f, 0.0f, 0.0f);
    glVertex2i(0, 0);
    glVertex2i(0, my);
    glEnd();

    glLoadIdentity();

    //hour
    float angle_h = (hour + minute / 60.0) * 30;

    glRotatef(-angle_h, 0.0f, 0.0f, 1.0f);

    glBegin(GL_LINES);
    glColor3f(0.0f, 0.0f, 0.0f);
    glVertex2i(0, 0);
    glVertex2i(0, hy);
    glEnd();



    glLoadIdentity();
```

```
	glFlush();
}

void resize(GLsizei width, GLsizei height) {

	glViewport(0, 0, width, height);

	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();

	if (width <= height)
		gluOrtho2D(-tam_x, tam_x, -tam_y * height / width, tam_y
* height / width);
	else
		gluOrtho2D(-tam_x * width / height, tam_x * width /
height, -tam_y, tam_y);

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

void add(int hz,int mz) {
	hour=hour+hz;
	minute=minute+mz;
	if(minute>=60) {
		hour=hour + (minute)/60;
		minute=minute%60;
	}
}

void sub(int hz,int mz) {
	int m,h;

	if(minute< mz) {
	   --hour;
	   minute+= 60;
	}

	m = minute-mz;
	h = hour - hz;
	hour=h;
	minute=m;
}
```

```
void move(int n) {
      time_t agora = time(0);
      struct tm *datahour = localtime(&agora);

      hour = datahour->tm_hour;
      minute = datahour->tm_min;
      second = datahour->tm_sec;

      switch(c) {
      case 1:
          break;
      case 2:
          sub(9,30);
          break;
      case 3:
          sub(4,30);
          break;
      case 4:
          add(3,30);
          break;
      case 5:
          add(4,30);
          break;
      case 6:
          sub(2,30);
          break;
      default:
          sub(5,30);
      }

      glutPostRedisplay();
      glutTimerFunc(1000, move, 0);
}

void initialize(void) {
      glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
}

int main(int argc, char **argv) {
      printf("Welcome to The Analog clock\n");
      printf("Choose any 1 time zones from\n");
      printf("\t1.New Delhi\n");
```

```
        printf("\t2.New York\n");
        printf("\t3.London\n");
        printf("\t4.Tokyo\n");
        printf("\t5.Sydney\n");
        printf("\t6.Moscow\n");
        printf("Enter your choice: ");
        scanf("%d",&c);
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(1300, 1300);
        glutCreateWindow("Clock");
        glutDisplayFunc(draw);
        glutReshapeFunc(resize);
        glutTimerFunc(1000, move, 0);
        initialize();
        glutMainLoop();
        return 0;
}
```

CHAPTER 5

SNAPSHOTS

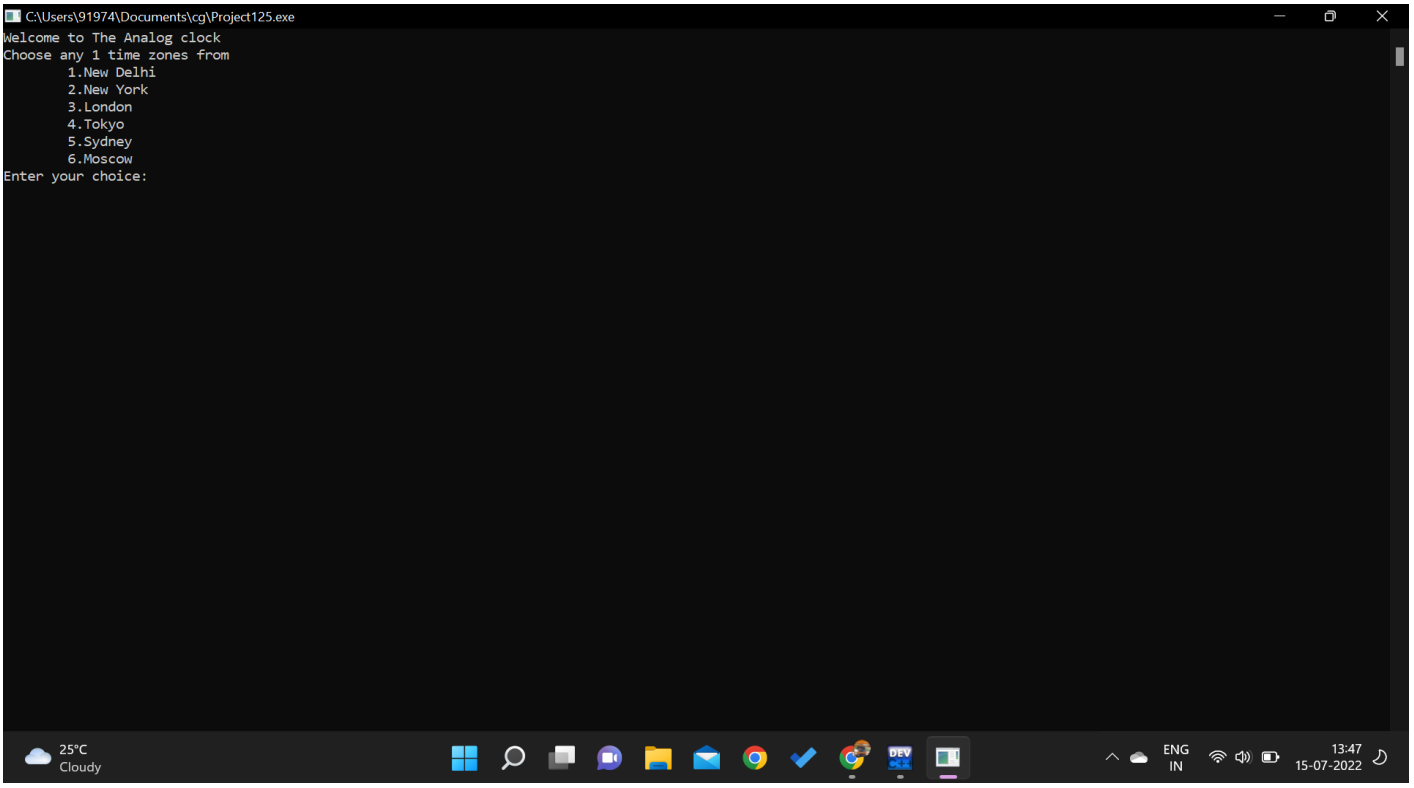

Fig 5.1: Console window of Menu driven choices.

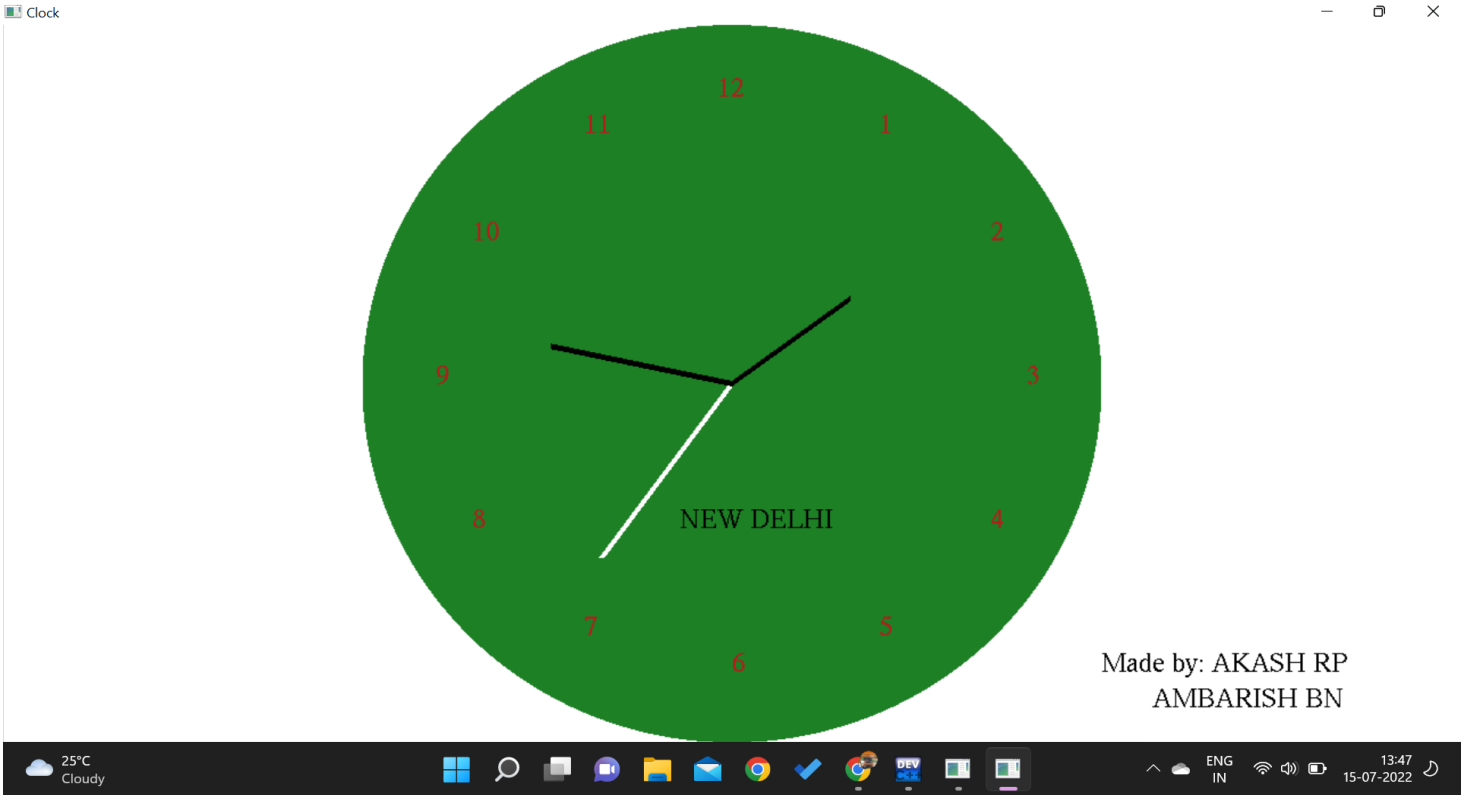

Fig 5.2: Choice 1: New Delhi

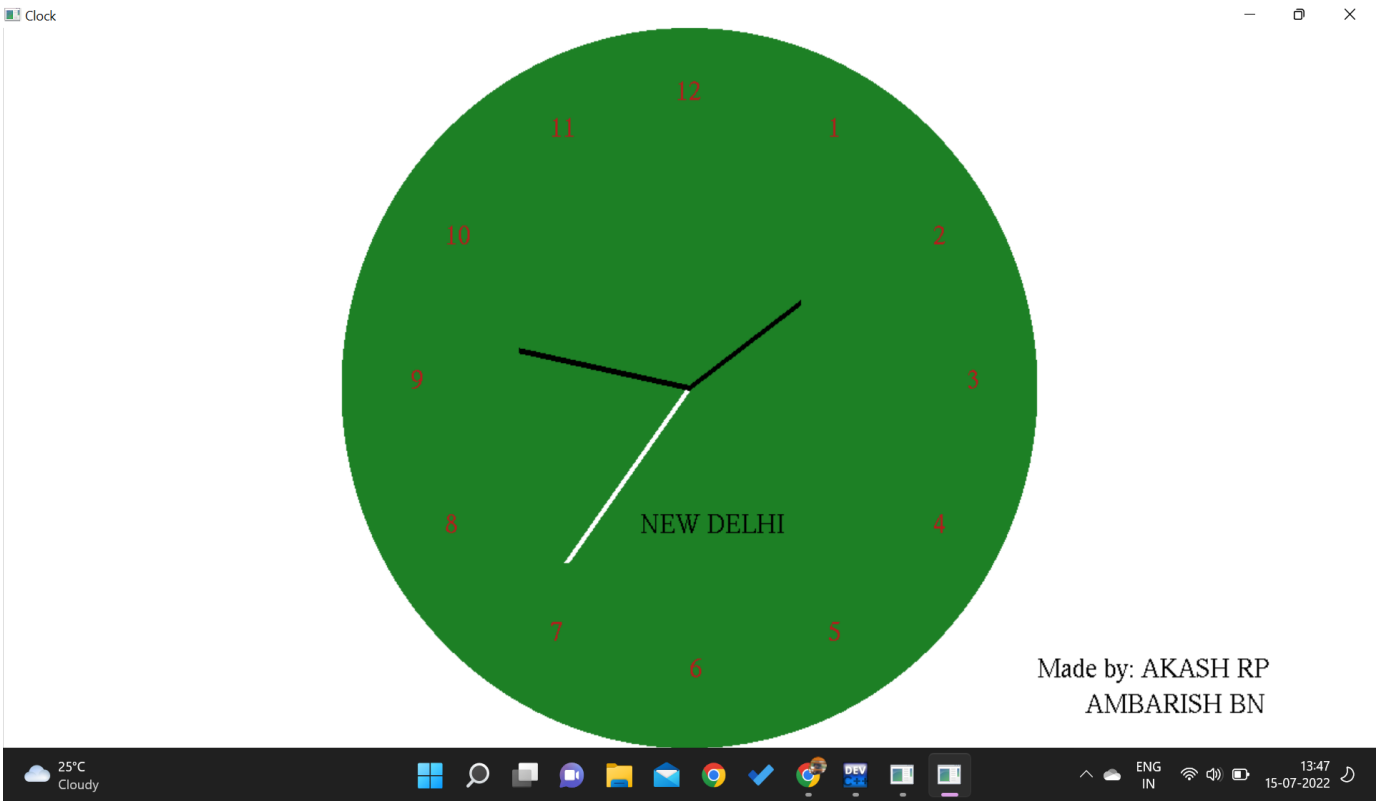

Fig 5.3: Choice 2: New York

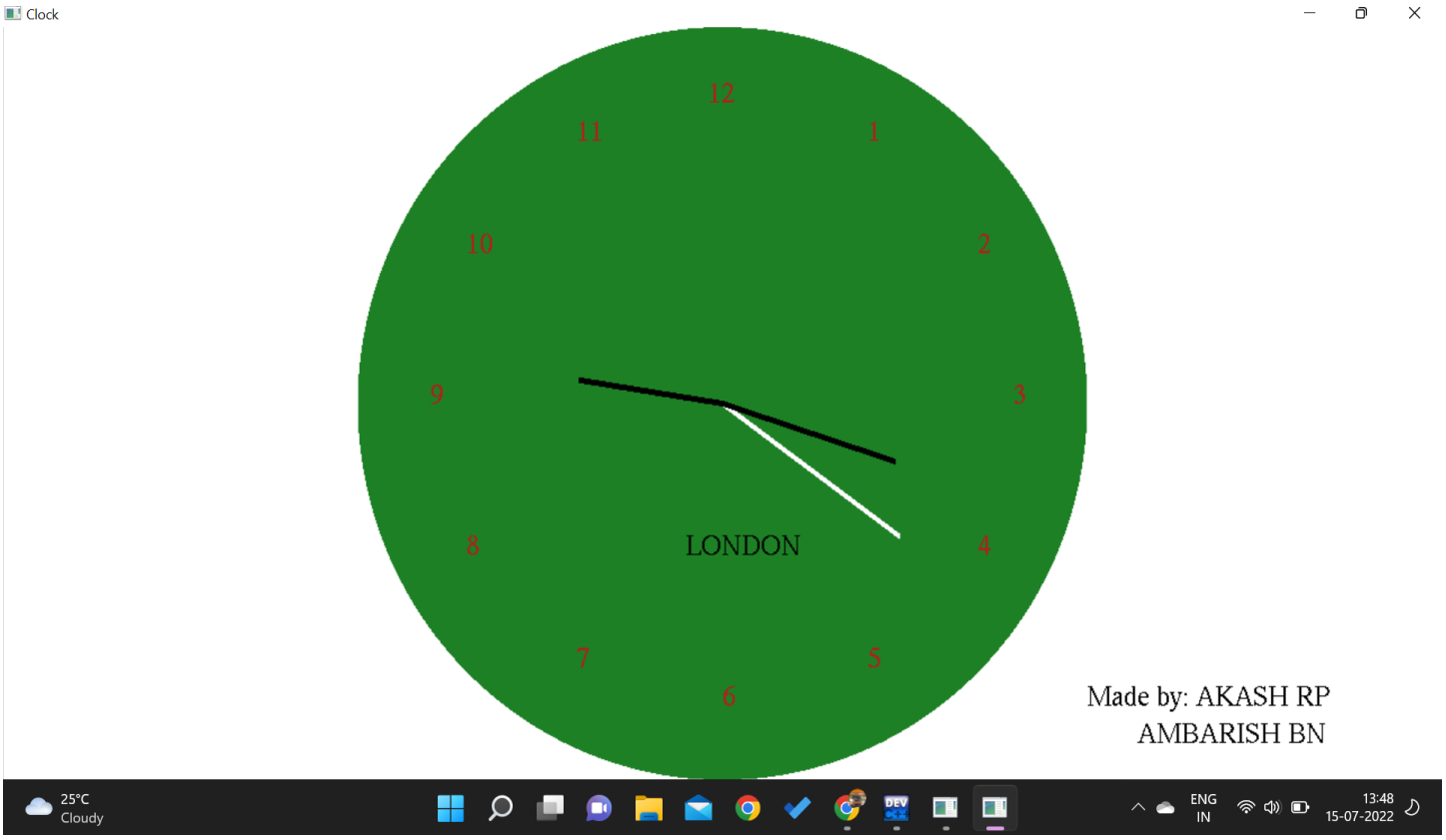

Fig 5.4: Choice 3: London

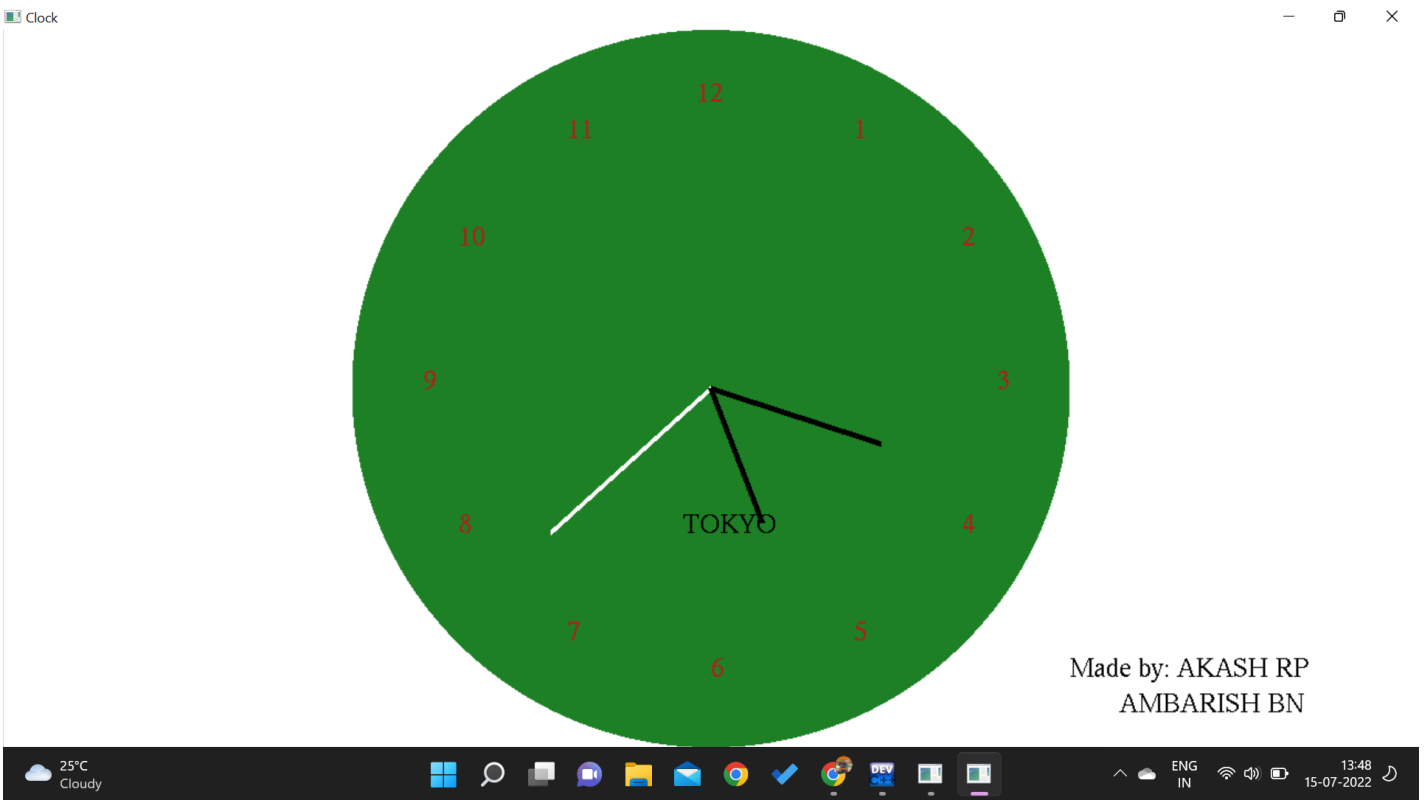

Fig 5.5: Choice 4: Tokyo

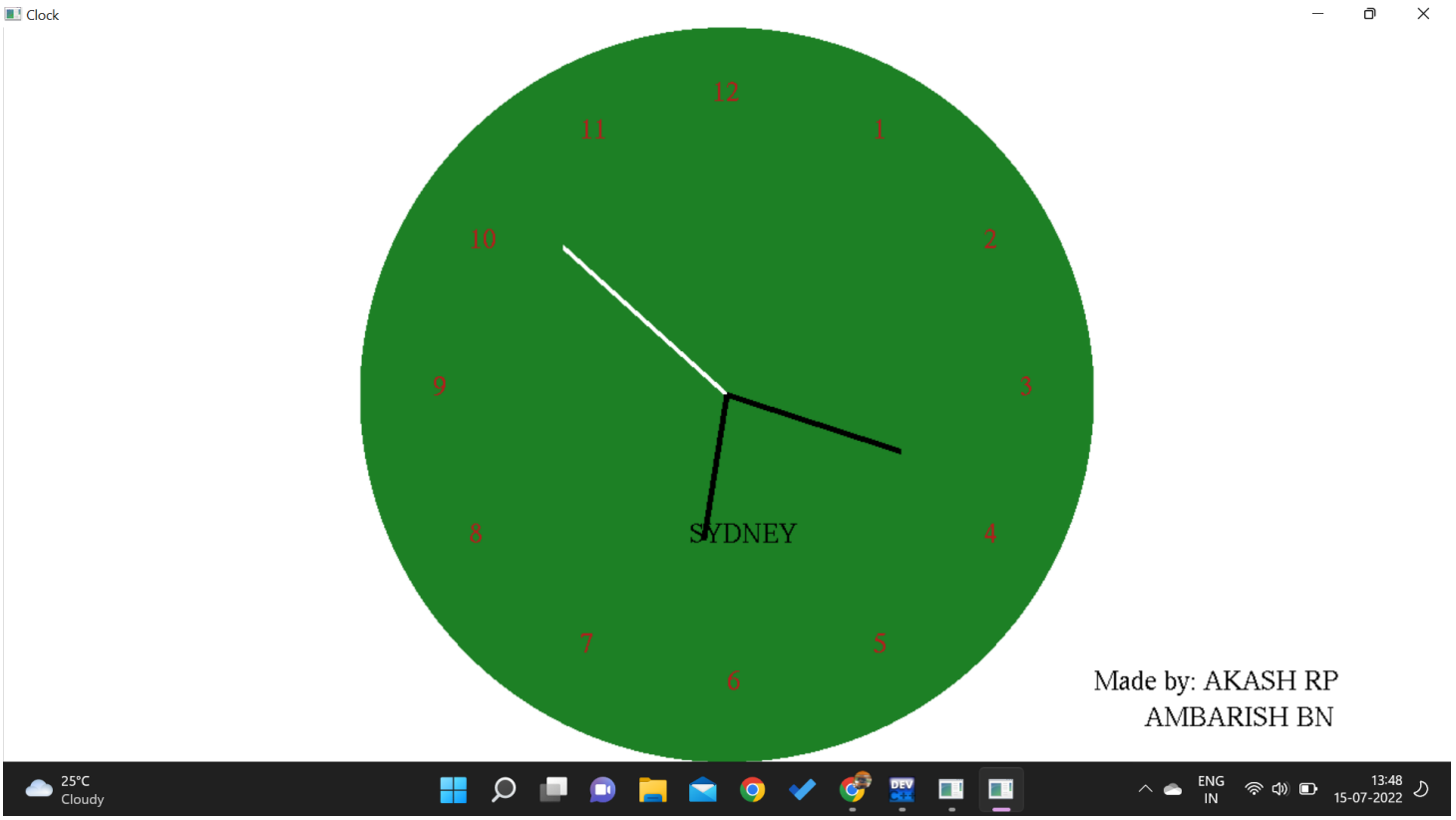

Fig 5.6: Choice 5: Sydney

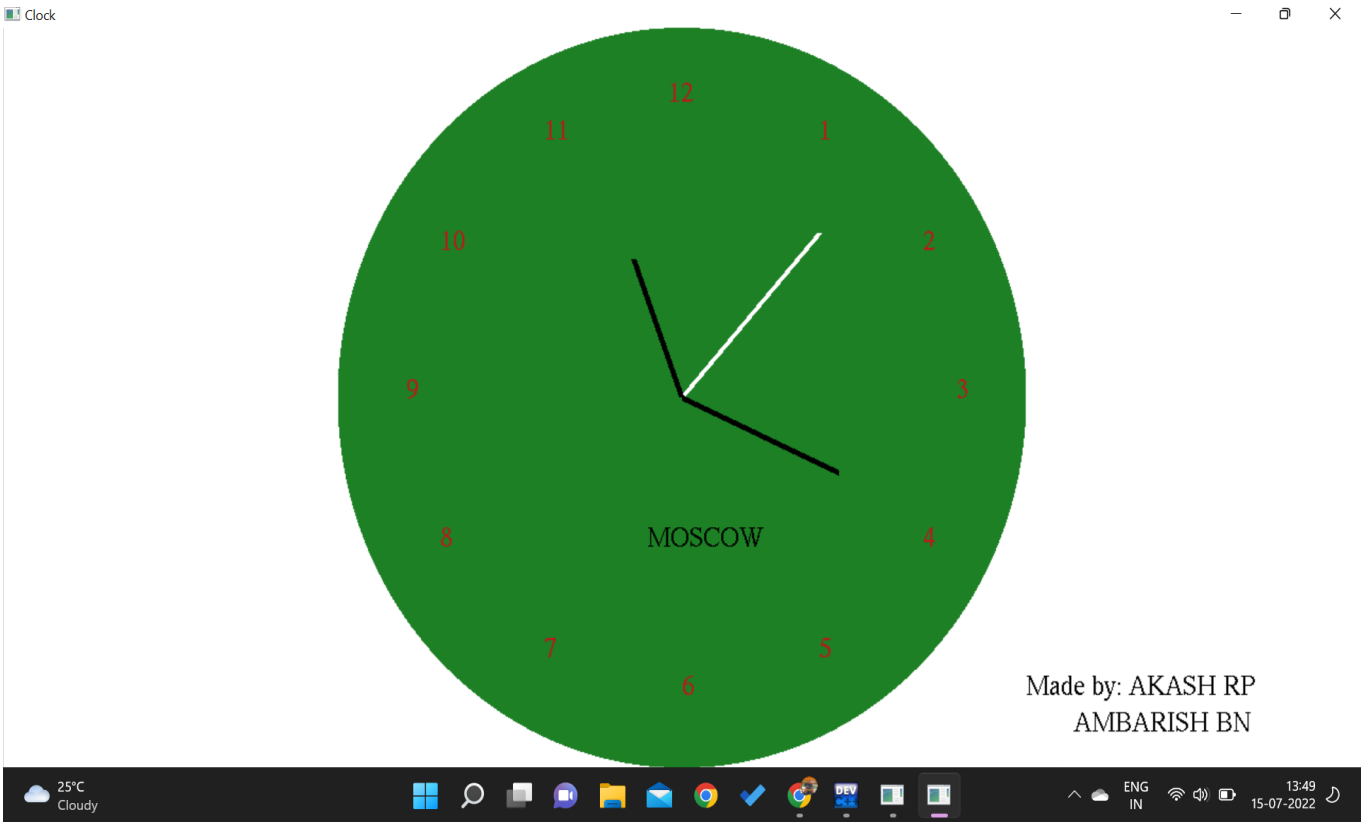

Fig 5.7: Choice 6: Moscow

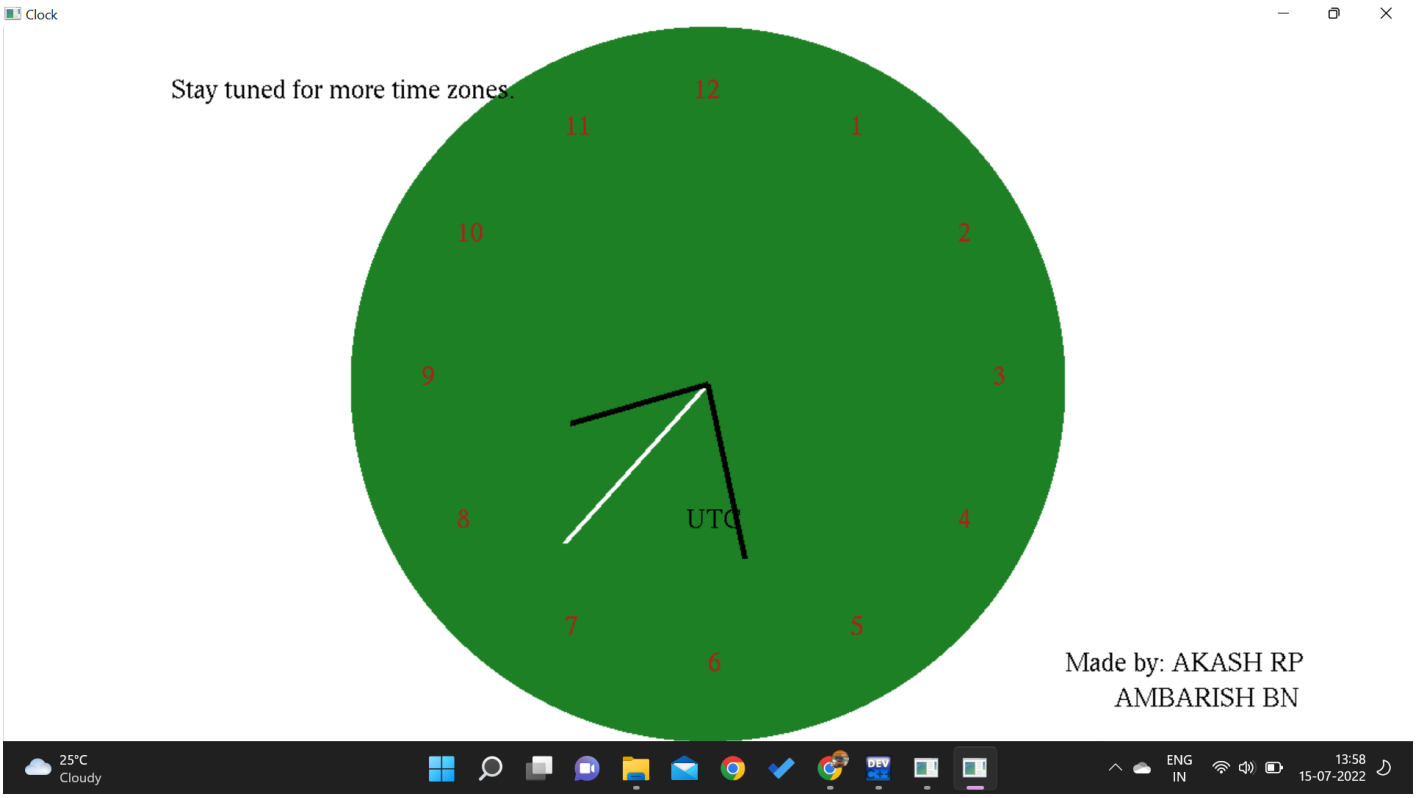

Fig 5.8: Wrong Choice: Display UTC

CHAPTER-6

CONCLUSION AND FUTURE ENHANCEMENT

An attempt has been made to develop an OpenGL package which meets necessary requirements of the user successfully. Since it is user friendly, it enables the user to interact efficiently and easily.

The development of the Mini Project has given us a good exposure to OpenGL by which we have learnt some of the techniques which help in the development of animated pictures, gaming.

Hence it is helpful for us to take up this field as our career too and develop some other features in OpenGL and provide as a token of contribution to the graphics world.

This project was done to have an idea about Computer Graphics using OpenGL. This project helped us to understand the concepts behind OpenGL and its programming. It also helped us to implement concepts in our project such as translation, drawing objects on the window screen using points, lines, polygons and lighting effects on the objects. This has given us a brief insight as to how programs, involving graphics, are written using OpenGL.

This project currently displays times from 6 different time zones namely New Delhi, New York, London, Tokyo, Sydney and Moscow. We can enhance this project by adding more time zones from all over the world. We can further add more faces to the clock to make it look glamorous.