**VISVESVARAYA TECHNOLOGICAL UNIVERSITY JNAMA SANGAMA,**

**BELGAVI – 590018**



**Mini project Report**

On

**“Implementation of Analog Clock using C and OpenGL”**

Submitted in the partial fulfillment of Sixth Semester CG Laboratory with Mini Project Work

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

By

**Imran Hussain 1ME20CS026 Shivam Yadav 1ME20CS039** Under the guidance of

**Mrs. Prabha Naik**

Associate Professor, CSE Dept.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**M S ENGINEERING COLLEGE**

NAAC Accredited, Affiliated to VTU, Belagavi, Approved by AICTE New Delhi, Navarathna Road, Bengaluru- Agrahara, off Intl. Airport. 562110

2022-23

Jnana Sangama, Belagavi-590018

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



## CERTIFICATE

This is to certify that the mini project work entitled **“Implementation of analog clock using C and OpenGL”** carried out by **Imran Hussain** (1ME20CS026) and **Shivam Yadav** (1ME20CS039), are bonafide students of **M S ENGINEERING COLLEGE** submitted in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University, Belagavi,** during the year **2022-2023.** It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report, deposited in the department library. This mini project work report has been approved as it satisfies the academic requirements in respect of project work prescribed for **Bachelor of Engineering Degree.**

**Signature of Guide Signature of HOD**

**Mrs. Prabha Naik Dr. Malatesh S.H,**

**Associate Professor, Dept. of CSE HOD, Dept. of CSE**

|  |  |  |
| --- | --- | --- |
| **Internal Examiner** | **Name** | **Signature** |
| **External Examiner** | **Name** | **Signature** |
|  |  |  |
| **Exam Date -** |  |  |

## ABSTRACT

Main aim of our work is to illustrate the concepts and usage of pre-built functions in OpenGL. A clock or watch is called "analog" when it has moving hands and hours marked from 1 to 12 to show you the time. Analog clocks usually indicate time using angles.

The most common clock face uses a fixed numbered dial or dials and moving hand or hands. It usually has a circular scale of 12 hours, which can also serve as a scale of 60 minutes, and 60 seconds if the clock has a second hand.

Many other styles and designs have been used throughout the years, including dials divided into 6, 8, 10, and 24 hours. The only other widely used clock face today is the 24 hour analog dial , because of the use of 24 hour time in military organizations and timetables.

My main goal for this project was to develop a model for analog clock identification, and to implement it with a program that would be able to receive an image of a clock, and return the time displayed after its analysis. I strived to write a program that would reach an acceptable success rate, and would work on a broad range of images.

## ACKNOWLEDGEMENT

A project work is of immense sheer size and it cannot be proficient by an individual all by them, ultimately, we are gratifying to a number of individuals whose qualified guidance, and assistance and encouragement have made it a pleasant venture to undertake this project work.

We are grateful to my institution, M S Engineering College with its ideals and inspiration for having provided us with the facilities, which has made this project work a success.

It is our pleasure to tender our heartfelt thanks to our College Trustees for their vision behind, towards the successful completion of our course

We would like to express our gratitude to Dr. P Mahadevaswamy, Principal MSEC, who is the source of inspiration as well providing an amiable atmosphere to work in.

Further, we would like to express our kind gratitude towards, Dr. Malatesh S.H, HOD, Dept. of CSE, and the whole department for providing us kindly environment for the successful completion of the project work.

We also extend our sincere thanks to our project guide, Mrs. Prabha Naik, Associate Prof. Dept. of CSE, for the timely suggestions and cooperation throughout our dissertation.

It’s our duty to thank one and all faculties of CSE Department, who have directly or indirectly supported to accomplish the project work successfully.

We would also like to thank our friends, who really helped us to complete our project work successfully.

**IMRAN HUSSAIN 1ME20CS026 SHIVAM YADAV 1ME20CS039**

# Table of Contents

**Title Page no.**

CHAPTER 1: INTRODUCTION 1

* 1. [MOTIVATION 1](#_TOC_250013)
  2. [SCOPE 1](#_TOC_250012)
  3. [PROBLEM STATEMENT 1](#_TOC_250011)
  4. [OBJECTIVES 2](#_TOC_250010)
  5. OPENGL FUNCTION 2

[CHAPTER 2: SYSTEM ANALYSIS 3](#_TOC_250009)

* 1. [EXISTING SYSTEM 3](#_TOC_250008)
     1. [FEATURES 3](#_TOC_250007)
     2. [LIMITATIONS 4](#_TOC_250006)
  2. [PROPOSED SYSTEM 4](#_TOC_250005)
     1. [FEATURES 4](#_TOC_250004)
     2. [ADVANTAGES 4](#_TOC_250003)

CHAPTER 3: SYSTEM REQUIREMENT SPECIFICATIONS 5

* 1. SOFTWARE REQUIRMENTS 5
  2. [HARDWARE REQUIREMENTS 5](#_TOC_250002)
  3. FUNCTIONAL REQUIRMENTS 5
  4. NON-FUNCTIONAL REQUIRMENTS 6

CHAPTER 4: SYSTEM DESIGN 7

* 1. FUNCTION USED 7
  2. IMPORTANT OPENGL FUNCTIONS 9

[CHAPTER 5: SNAPSHOTS 13](#_TOC_250001)

* 1. ORTHO VIEW 13
  2. PERSPECTIVE VIEW 13
  3. LIGHTING OFF 14
  4. LIGHTING ON 14

[CHAPTER 6: CONCLUSION 15](#_TOC_250000)

BIBLIOGRAPHY 16

CHAPTER-1

# INTRODUCTION

Clocks are in homes, offices and many other places; smaller ones (watches) are carried on the wrist or in a pockets, larger ones are in public places, e.g. a railway station or church. A small clock is often shown in the corner of the computer displays, mobile phones and many MP3 players.

Our project is Analog clock which depict a wall clock which is completely satisfied completely satisfied i.e., it shows time in the form of hours, minutes and seconds while we run the program.

not only that we can hear the hand movement sound of hour, minute and second hands i.e., tick tick sound done by a clock and there will be two clock of this type in which one show the system time and the other show the time of the other country as per the choice. Both clocks are similar in looking but different in time

### MOTIVATION

Simulation of any process helps in better understanding of it. This project will demonstrate the internal working of a Analog clock and help us to find the time correctly. This system can be used for educational purposes to teach and explain the importance Analog clock.

### SCOPE

The scope of the analog clock project is that a person should be able to know the exact time of the system in the form of hours, minutes and seconds and also should be able to know the other country time of his choice in the given list in the form of hours, minutes and seconds and should be able to know the time difference between the two times.

### 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 individual Thereby developing. manipulating, applying any transformation, rotation, scaling on them is the major task of any image development. Thereby we have put our tiny effort to develop 3D objects and perform different operations on them by using OpenGL utilities.

### OBJECTIVES

* + - Developing a package using computer graphics with OpenGL.
    - Migration from text editor to OpenGL.
    - To show that implementation of translation is easier with OpenGL.
    - Implementing certain technical concepts like translation, motion , and use of idle function.
    - How to use lightning effects used to produce computer animation.
    - Creation of primitives, i.e. polygons
* Providing human interaction through mouse and keyboard.

### OPENGL FUNCTIONS

OpenGL draws primitives—points, line segments, or polygons—subject to several selectable modes. You can control modes independently of each other; that is, setting one mode doesn't affect whether other modes are set.

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects.

A vertex defines a point, an endpoint of a line, or a corner of a polygon where two edges meet. Data is associated with a vertex, and each vertex and its associated data are processed independently, in order, and in the same way.

The type of clipping depends on which primitive the group of vertices represents. Commands are always processed in the order in which they are received, although there may be an indeterminate delay before a command takes effect.

This means that each primitive is drawn completely before any subsequent command takes effect. It also means that state-querying commands return data that's consistent with complete execution of all previously issued OpenGL commands.

## CHAPTER-2 SYSTEM ANALYSIS

Systems analysis is "the process of studying a procedure or business to identify its goal and purposes and create systems and procedures that will efficiently achieve them". Another view sees system analysis as a problem-solving technique that breaks down a system into its component pieces, and how well those parts work and interact to accomplish their purpose. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose

### EXISTING SYSTEM

In the existing system, most of the records are maintained on paper. It becomes very inconvenient to modify the data. In the existing system, here is a possibility that the same data in different registers may have different values which means the entries of the same datado not match. This inconsistent state does not supply the concrete information which poses a problem in the case information related to particular search record. Our project is very useful. Existing system for a graphics is the TC++. This system will support only the 2D graphics. 2D graphics package being designed should be easy to use and understand. It should provide various options such as free hand drawing, line drawing, polygon drawing, filled polygons, flood fill, translation, rotation, scaling, clipping etc. Even though these properties were supported, it was difficult to render 2D graphics cannot be very difficult to get a 3-Dimensional object. Even the effects like lighting, shading cannot be provided. So, we go for Microsoft Visual Studio software.

### FEATURES

* Relies on paper-based records
* Manual data entry and modification
* Inconsistent data entries across registers
* Lack of concrete and reliable information
* Limited 2D graphics capabilities
* Difficulties in creating 3D objects
* Absence of user-friendly interface

### LIMITATIONS

* + - * Takes a lot of effort to create even a basic animation.
      * It is static output only shows the time.

### PROPOSED SYSTEM

The system based on OpenGL enables the development of graphics applications with 2D and 3D rendering. It provides cross-platform compatibility and supports visually appealing graphics, geometric transformations, lighting and shading effects, texturing, and flexible customization.

With a simple interface, developers can create graphics with impressive visual effects and easy modification.

### FEATURES

The OpenGL Code we presenting have many of the good features. Some of the features are listed below

* It is the Basic opponent of AI.
* It is a simple interface.
* It has nice graphics and nice visual effects too.
* It is easy to alter the graphics.

•

### ADVANTAGES

* + - * It shows the exact time.
      * This can be viewed so colorful.
      * It gains the attention of the viewer
      * It also shows the Date and the Day

## CHAPTER-3

**SYSTEM REQUIREMENT SPECIFICATION**

A System Requirements Specification is a structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify business problems and propose solutions.

### SOFTWARE REQUIREMENTS

Software requirements for a system are the description of what the system should do, the service or services that it provides and the constraints on its operation.

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

### HARDWARE REQUIREMENTS

The hardware requirements are the requirements of a hardware device. Most hardware only has operating system requirements or compatibility.

1 Operating System: Microsoft Windows XP, Microsoft Windows 7

1. Compiler used: VC++ 6.0 compiler
2. Language used: Visual C++

### FUNCTIONAL REQUIREMENTS

Functional requirements are the desired operations of a program, or system as defined in software development and systems engineering. The systems in systems engineering can be either software electronic hardware or combination software-driven electronics.

* + 1. OPENGL APIs:

If we want to have a control on the flow of program and if we want to interact with the window system then we use OpenGL API'S. Vertices are represented in the same manner internally, whether they are specified as two-dimensional or three-dimensional entities, everything that we do are here will be equally valid in three dimensions. Although OpenGL is easy to learn, compared with other APIs, it is nevertheless powerful. It supports the simple three dimensional

* + 1. GL/GLUT.H:

We use a readily available library called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system.The application program uses only GLUT functions d can be recompiled with the GLUT library for other window system. OpenGL make heavy use of macros to increase code readability and avoid the use of magic numbers. In most implementation, one of the include lines.

### NON-FUNCTOINAL REQUIREMENTS

Non-Functional Requirements are the constraints or the requirements imposed on the system. They specify the quality attribute of the software. Non-Functional Requirements deal with issues like scalability, maintainability, performance, portability, security, reliability, and many more. Non-Functional Requirements address vital issues of quality for software systems. If NFRs not addressed properly, the results can include:

* Users, clients, and developers are unsatisfied.
* Inconsistent software.
* Time and cost overrun to fix the software which was prepared without keeping NFRS in mind.

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs.

Also known as system qualities, nonfunctional requirements are just as critical as functional Epics, Capabilities, Features, and Stories. They ensure the usability and effectiveness of the entire system. Failing to meet any one of them can result in systems that fail to satisfy internal business, user, or market needs, or that do not fulfill mandatory requirements imposed by regulatory or standards agencies. In some cases, non-compliance can cause significant legal issues (privacy, security, safety, to name a few)

## CHAPTER 4

### SYSTEM DESIGN

* 1. **FUNCTIONS USED**

The program contains many functions listed below.

1.) **Sprint(int x, int y, const char\* st):**

This function is used to display text on the screen at the specified coordinates (x, y). It takes a character string (st) as input and iterates over each character, using **glutBitmapCharacter** to render each character on the screen.

2.) **TimeEvent(int te)**

* + - This function is a timer event callback that updates the clock's time by incrementing the angle variables (rx, ry, rz, angle) responsible for the rotation of the clock hands. It also triggers a redisplay of the scene and sets the timer function to be called again after a certain time interval (100 milliseconds in this case)

3.) **init(void)**

This function initializes the OpenGL settings and sets up the lighting for the scene. It configures the clear color, shading model, enables depth testing, sets the lighting parameters, and creates the GLU (OpenGL Utility Library) objects for drawing cylinders and disks.

4.) **Draw\_gear(void)**

This function is responsible for drawing the gear-like structure on the clock. It uses GLU functions to draw a cylinder and two disks to create the base of the gear, and then it positions and rotates cubes to form the teeth of the gear.

5.) **Draw\_clock(GLfloat cx, GLfloat cy, GLfloat cz)**

This function draws the entire clock on the screen. It takes the (cx, cy, cz) coordinates to specify the position of the clock in the 3D scene. It uses GLU functions and GLUT solids to draw the clock's components, including the wireframe cube, clock face, hour hand, minute hand, second hand, and hour and second markers.

6.) **display(void)**

This function is the display callback that is responsible for rendering the scene. It first obtains the current local time using time and localtime functions, and then it clears the color and depth buffers. It sets the projection and modelview matrices based on the view state (perspective or orthographic) and enables or disables lighting based on the light state. Finally, it calls Draw\_clock to draw the clock and swaps the buffers to display the rendered scene.

7.)**reshape(int w, int h)**

This function is the reshape callback that is called when the window is resized. It adjusts the viewport and the projection matrix to match the new window dimensions.

8.) **keyboard(unsigned char key, int x, int y)**

This function is the keyboard callback that handles keyboard input. It responds to specific keys: 'L' toggles the light state (on/off), 'V' toggles the view state (perspective/orthographic), and 'ESC' key exits the program.

9.)**main(int argc, char\*\* argv)**

The main function initializes GLUT, creates the window, sets the window properties, and registers the callback functions. It then enters the GLUT main loop, which handles the event processing and rendering.

## Important OpenGL functions

1. **glClearColor: Sets the color that will be used to clear the color buffer when `glClear` is called, allowing you to define the background color of your OpenGL scene.**
2. **glShadeModel: Specifies the shading model for rendering objects, choosing between flat shading (one color per face) or smooth shading (interpolation of colors across vertices).**
3. **glEnable: Enables specific OpenGL capabilities, like depth testing or lighting, enabling you to utilize advanced rendering techniques in your scene.**
4. **glDisable: Disables specific OpenGL capabilities, such as lighting, allowing you to selectively turn off certain features in your rendering pipeline.**
5. **glLightfv: Sets the properties of a light source, including its ambient, diffuse, or specular intensity and color, influencing the lighting of your scene.**
6. **glMatrixMode: Specifies which matrix stack is the target for subsequent matrix operations, such as modelview or projection matrices, allowing you to control how transformations are applied.**
7. **glLoadIdentity: Resets the current matrix to the identity matrix, providing a clean slate for transformations and ensuring no prior transformations affect subsequent rendering.**
8. **glOrtho: Defines a 2D orthographic projection matrix, specifying the coordinate system and mapping of objects from 3D space to the 2D screen, useful for 2D**

**rendering.**

1. **gluNewQuadric: Creates a new quadric object for rendering quadric shapes like spheres or cylinders, providing a convenient interface for generating complex geometric primitives.**
2. **gluQuadricDrawStyle: Specifies the drawing style for the quadric object, allowing you to render it as a solid shape or wireframe, providing flexibility in visual .**
3. **gluQuadricOrientation: Specifies the orientation of the quadric object, determining whether it is oriented inside or outside, affecting the direction of normal vectors used in lighting calculations.**
4. **gluQuadricTexture: Enables or disables texturing for the quadric object, allowing you to apply textures to the rendered quadric shapes.**
5. **gluDisk: Draws a disk or partial disk using the current quadric object, creating a flat circular shape with optional inner and outer radii.**
6. **gluCylinder: Draws a cylinder using the current quadric object, rendering a solid or hollow tube with specified radius and height.**
7. **glutBitmapCharacter: Renders a bitmap character from the specified font, allowing you to display individual characters on the screen.**
8. **glutWireCube: Renders a wireframe cube, drawing the outlines of a cube without filling its faces.**
9. **glutSolidCube: Renders a solid cube, drawing a cube with filled faces.**
10. **gluPerspective: Defines a perspective projection matrix, specifying the viewing**

**frustum parameters, such as field of view, aspect ratio, and near/far clipping planes.**

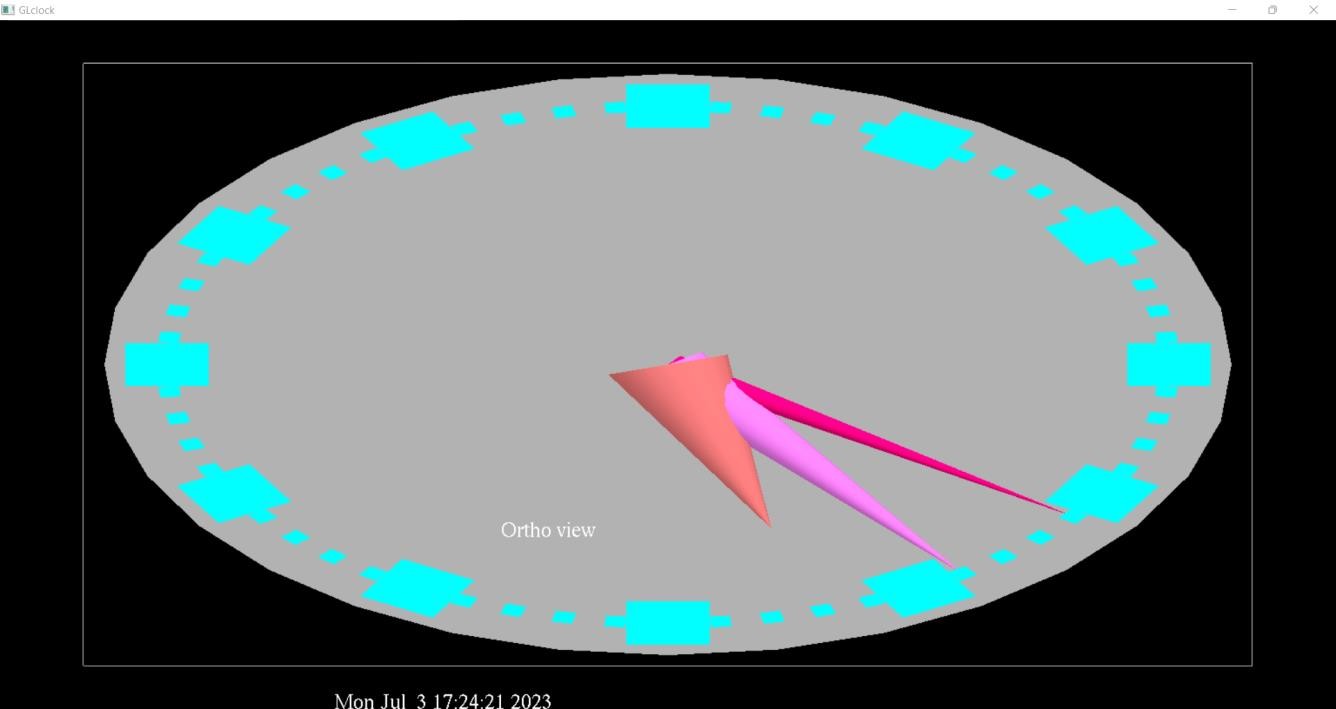
1. **gluLookAt: Defines a viewing transformation matrix, positioning the camera in 3D space and specifying the viewing direction and up vector.**
2. **glViewport: Sets the viewport transformation, specifying the mapping of normalized device coordinates to the window or screen coordinates.**
3. **glutSwapBuffers: Swaps the front and back buffers, displaying the rendered image on the screen, useful in double-buffered rendering.**
4. **glutDisplayFunc: Sets the display callback function, specifying the function to be called for rendering the scene whenever the window needs to be redrawn.**
5. **glutReshapeFunc: Sets the reshape callback function, defining the function to be called whenever the window is resized, allowing you to update the viewport or adjust the projection matrix.**
6. **glutKeyboardFunc: Sets the keyboard callback function, allowing you to define a function that will be called whenever a keyboard key is pressed or released.**
7. **glutTimerFunc: Registers a timer callback function, enabling you to schedule a function to be called after a specified time interval.**
8. **glutInit: Initializes the GLUT library, setting up the necessary environment and internal data structures before creating windows and handling events.**
9. **glutInitDisplayMode: Sets the initial display mode, specifying the properties of the window and frame buffer, such as color depth, double buffering, and stencil buffer.**
10. **glutInitWindowSize: Sets the initial window size in pixels, defining the dimensions of**

**the window that will be created.**

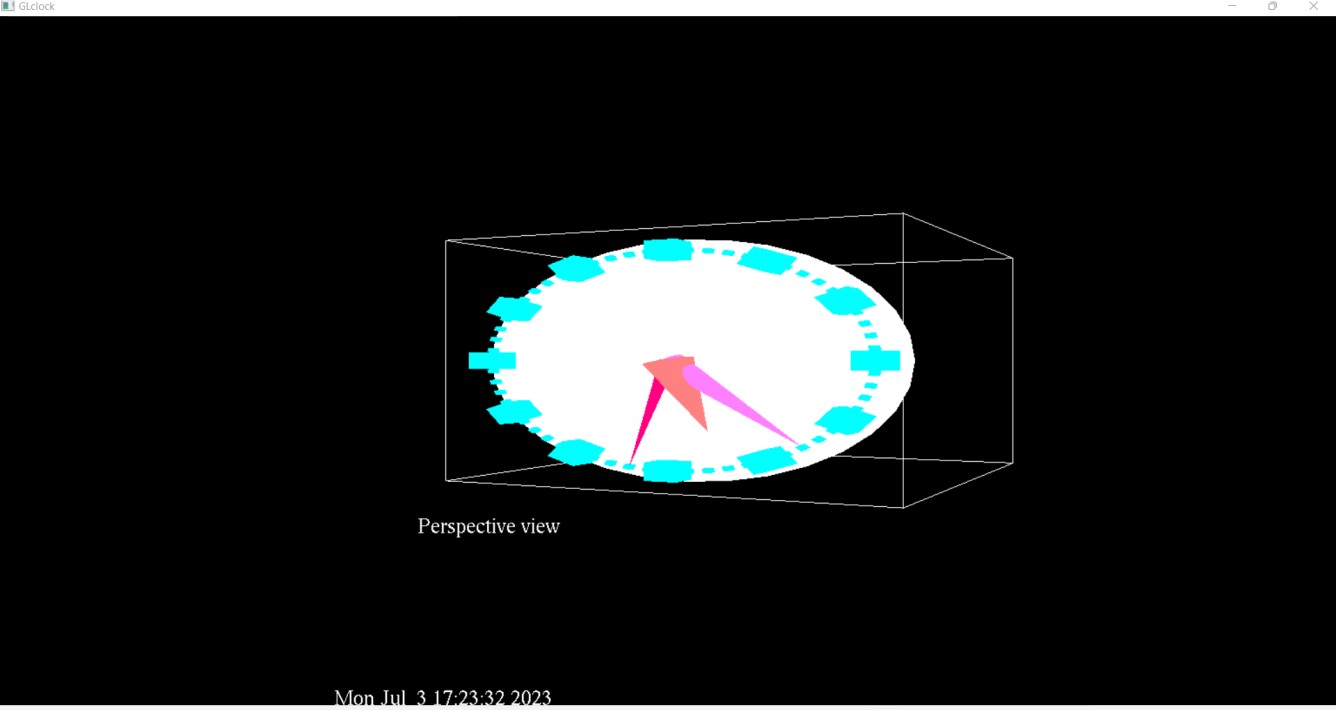
1. **glutInitWindowPosition: Sets the initial window position in screen coordinates, determining where the window will appear on the screen.**
2. **glutCreateWindow: Creates a window with the specified title, generating a new window and associating it with the current OpenGL context.**
3. **glutSetWindowTitle: Sets the window title, allowing you to change the text displayed in the title bar of the window.**
4. **glutMainLoop: Enters the GLUT event processing loop, starting the main program loop where GLUT will handle events, update the window, and execute registered callback functions.**

**CHAPTER 5 SNAPSHOTS**

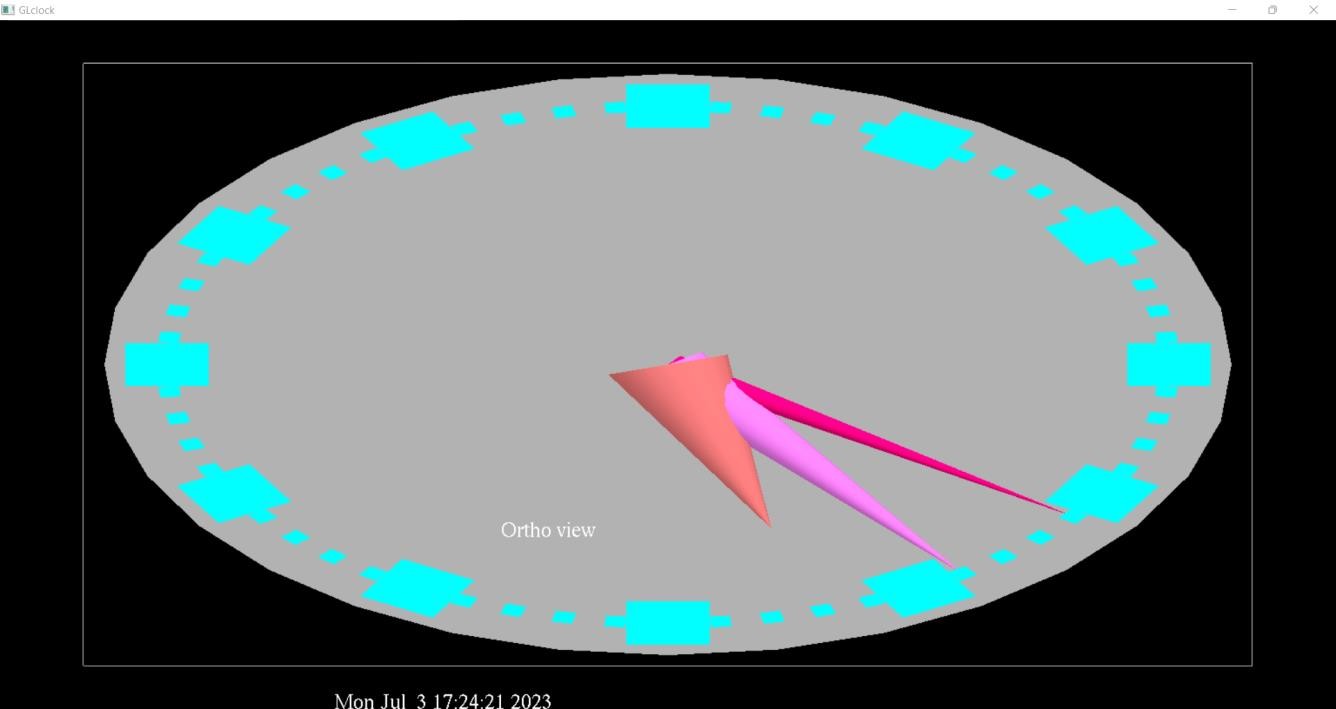
## 5.1

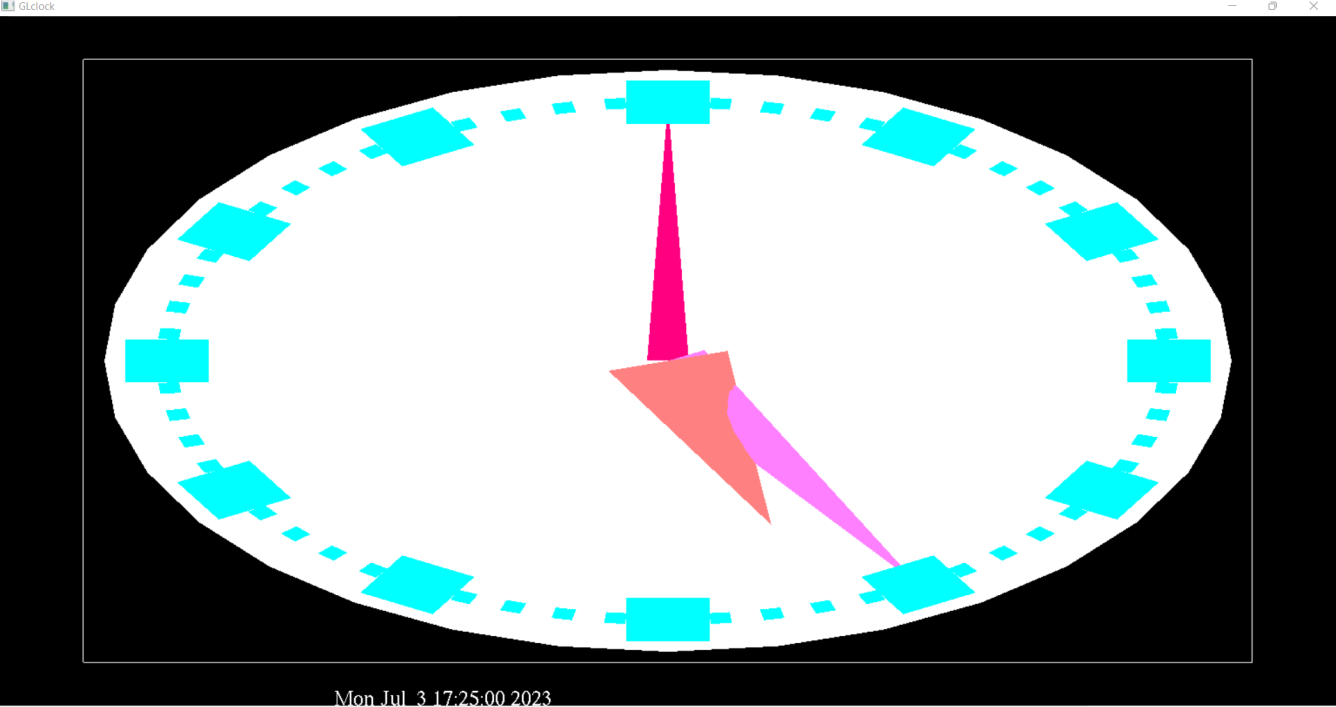


**5.2**



## 5.3



**5.4**

## CHAPTER 6 CONCLUSION

It has been an interesting journey through the development of this project. At the beginning we used our limited knowledge to implement only the basic features. However, through the months of development, new issues and bugs led to new ideas which led to newer methods of implementation which, in turn, led to us learning even more features of the OpenGL and apply more creative and efficient ways to perform the older functions. New methods helped us in adding flexibility to the various parts of the program, making the further addition of newer features easier and less time consuming which, again, led to the possibility of adding even more features. This sequential chain reaction of progress and ideas has enabled to learn so much through the months of working on this project and we have done our best to add as many features as we could and provide a user interface that is easy and intuitive to use.

Before concluding, it is worth mentioning that this project would never have been possible without the tremendous amount of encouragement by the staff and guides of our department.

We are content with the outcome of this project and are hopeful that it meets the requirements expected and we wish that it may inspire others to be creative and critical in the field of computer graphics and have a newfound appreciation for the Open Graphics Library.

## CHAPTER 7 BIBLIOGRAPHY

[https://www.codeproject.co](https://www.codeproject.com/) [m](https://www.codeproject.com/) [https://www.geeksforgeeks](https://www.geeksforgeeks.org/)

[.org](https://www.geeksforgeeks.org/) https://stackoverflow.com [https://www.javapoint.com](https://www.javapoint.com/) [https://github.com](https://github.com/) <https://processing.org/>