VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum: 590018

A Mini Project Report (18CSL67)

on

"ACHARYA BOYS' HOSTEL ROOM"

A mini project report submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering

Submitted by

Abhinav Sinha (1AY19CS001)

Mridul Mondal (1AY19CS060)

Under the guidance of

Prof. Varalakshmi B D

Prof. Sujatha T

Prof. Reshma

Department of Computer Science &Engineering

Acharya Institute of Technology

Department of Computer Science & Engineering

Soladevanahalli, Bangalore-560107

ACHARYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belgaum)
Soladevanahalli, Bangalore – 560 107

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

Certified that the Computer Graphics Mini Project entitled **"ACHARYA BOYS' HOSTEL ROOM "** is a bonafide work carried out by **Mr. Abhinav Sinha (1AY19CS001) & Mr. Mridul Mondal (1AY19CS060)** in partial fulfillment for the award of degree of **Bachelor of Engineering in Computer Science & Engineering of the Visvesvaraya Technological University**, Belgaum during the academic year **2021-2022.** It is certified that all corrections/ suggestions indicated for internal assessments have been incorporated in the Report deposited in the departmental library. The Mini project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the **Bachelor of Engineering Degree**.

Signature of Guides **Signature of H.O.D**

Name of the Examiners **Signature with date**

1.

ACKNOWLEDGEMENT

We express our gratitude to our institution and management for providing us with good infrastructure, laboratory facilities and inspiring staff, and whose gratitude was of immense help in completion of this Computer Graphics and Visualization mini project successfully.

We express our sincere gratitude to our principal, **Dr. Rajath Hegde** and Vice Principal **Prof. Marigowda C K** for providing us the required environment in completion of this Computer Graphics and Visualization mini project

Our sincere thanks to **Dr. Ajith Padyana,** Head of the Department, Computer Science and Engineering, Acharya Institute of Technology for his valuable support and for rendering us the resources for this Computer Graphics and Visualization mini project.

We heartily thank**, Prof. Varalakshmi B D, Prof. Sujatha T and Prof. Reshma** Assistant Professors, Department of Computer Science and Engineering, Acharya Institute of Technology who guided us with valuable suggestions at every stage in completing this Computer Graphics and Visualization mini project.

Our gratitude thanks should be rendered to many people who helped us in all possible ways.
A warm thanks to all the faculty of Department of Computer Science and Engineering, who have helped us with their views and encouraging ideas.

ABSTRACT

The project is on **3D ACHARYA BOY'S HOSTEL ROOM** using OpenGL functions. It is a user interactive program where the user can view the required display on the screen by making use of the input devices such as keyboard and mouse. The project mainly consists of a bed, clock, air-conditioner, table, chair, table lamp & ceiling bulb which can be moved and stop as per the user wish. It is even more exciting to implement this room through a code written in OpenGL and allowing it to serve as a basis for explaining most of the concepts that can be driven through it. There is room whose clock stopped working on pressing 5 on keyboard its battery gets changed and it start to work, there is AC in room which can be switched on/off as per wish, we have also incorporated mood lighting in the room.

CONTENTS

List of Figures

List of Tables

Chapter 1
INTRODUCTION

1.1 Introduction to Computer Graphics

Computer Graphics involves technology to access. The Process transforms and presents information in a visual form. The role of computer graphics insensible. In today life, computer graphics has now become a common element in user interfaces, T.V. commercial motion pictures.

Computer Graphics is the creation of pictures with the help of a computer. The end product of the computer graphics is a picture it may be a business graph, drawing, and engineering..

In computer graphics, two or three-dimensional pictures can be created that are used for research. Many hardware devices algorithm has been developing for improving the speed of picture generation with the passes of time. It includes the creation storage of models and image of objects. These models for various fields like engineering, mathematical and so on.

Today computer graphics is entirely different from the earlier one. It is not possible. It is an interactive user can control the structure of an object of various input devices

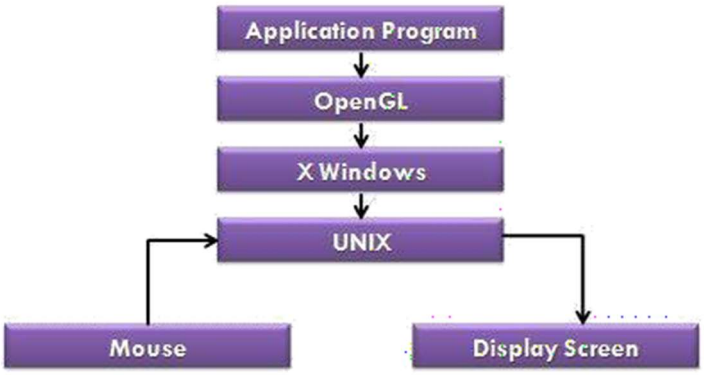


FIG:1.1 OVERVIEW OF COMPUTER GRAPHICS

1.1.1 Definition of Computer Graphics

It is the use of computers to create and manipulate pictures on a display device. It comprises of software techniques to create, store, modify, represents pictures.

1.1.2 Why Computer Graphics used?

Suppose a shoe manufacturing company want to show the sale of shoes for five years. For this vast amount of information is to store. So a lot of time and memory will be needed. This method will be tough to understand by a common man. In this situation graphics is a better alternative. Graphics tools are charts and graphs.

Interactive computer graphics work using the concept of two-way communication between computer users. The computer will receive signals from the input device, and the picture is modified accordingly. Picture will be changed quickly when we apply command.

1.1.3 Application of Computer Graphics

Computer graphics deals with creation, manipulation and storage of different type of images and objects.

Some of the applications of computer graphics are :

1. **Computer Art:** Using computer graphics we can create fine and commercial art which include animation packages, paint packages. These packages provide facilities for designing object shapes and specifying object motion. Cartoon drawing, paintings, logo design can also be done.
2. **Computer Aided Drawing:** Designing of buildings, automobile, aircraft is done with the help of computer aided drawing, this helps in providing minute details to the drawing and producing more accurate and sharp drawings with better specifications.
3. **Presentation Graphics:** For the preparation of reports or summarising the financial, statistical, mathematical, scientific, economic data for research reports, managerial reports, moreover creation of bar graphs, pie charts, time chart, can be done using the tools present in computer graphics.
4. **Entertainment:** Computer graphics finds a major part of its utility in the movie industry and game industry. Used for creating motion pictures , music video, television shows, cartoon animation films. In the game industry where focus and interactivity are the key players, computer graphics helps in providing such features in the efficient way.
5. **Education:** Computer generated models are extremely useful for teaching huge number of concepts and fundamentals in an easy to understand and learn manner.

Using computer graphics many educational models can be created through which more interest can be generated among the students regarding the subject.

6. **Training:** Specialized system for training like simulators can be used for training the candidates in a way that can be grasped in a short span of time with better understanding. Creation of training modules using computer graphics is simple and very useful
7. **Visualization:** Today the need of visualize things have increased drastically, the need of visualization can be seen in many advance technologies , data visualization helps in finding insights of the data , to check and study the behaviour of processes around us we need appropriate visualization which can be achieved through proper usage of computer graphics
8. **Image Processing:** Various kinds of photographs or images require editing in order to be used in different places. Processing of existing images into refined ones for better interpretation is one of the many applications of computer graphics.
9. **Machine Drawing:** Computer graphics is very frequently used for designing, modifying and creation of various parts of machine and the whole machine itself, the main reason behind using computer graphics for this purpose is the precision and clarity we get from such drawing is ultimate and extremely desired for the safe manufacturing of machine using these drawings.

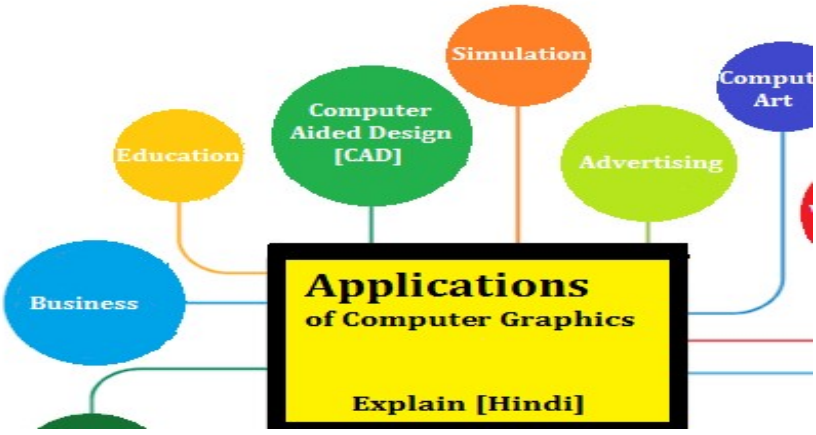


FIG:1.2 APPLICATION OF COMPUTER GRAPHICS

1.2 OpenGL Concepts

1.2.1 Interface

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. Functions in the main GL library have names that begins with *gl* and are stored in a library usually referred to as GL. The second is the **OpenGL Utility Library(GLU)**. The library uses only GL functions but contains code for creating common objects and simplifying viewing.

All functions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with the letter *glu*. Rather than using a different library for each system we use a readily available library called OpenGL Utility Toolkit(GLUT),which provides the minimum functionality that should be expected in any modern windowing system.

1.2.2 Overview

- OpenGL(Open Graphics Library) is the interface between a graphics program and graphics hardware. *It is streamlined*. In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons.
- OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc.
- It is system-independent. It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.
- It is a state machine. At any moment during the execution of a program there is a current model transformation.
- It is a rendering pipeline. The rendering pipeline consists of the following steps:
    * Defines objects mathematically.
    * Arranges objects in space relative to a viewpoint.
    * Calculates the color of the objects.

1.2.3 OpenGL Architecture

1) Pipeline Architectures

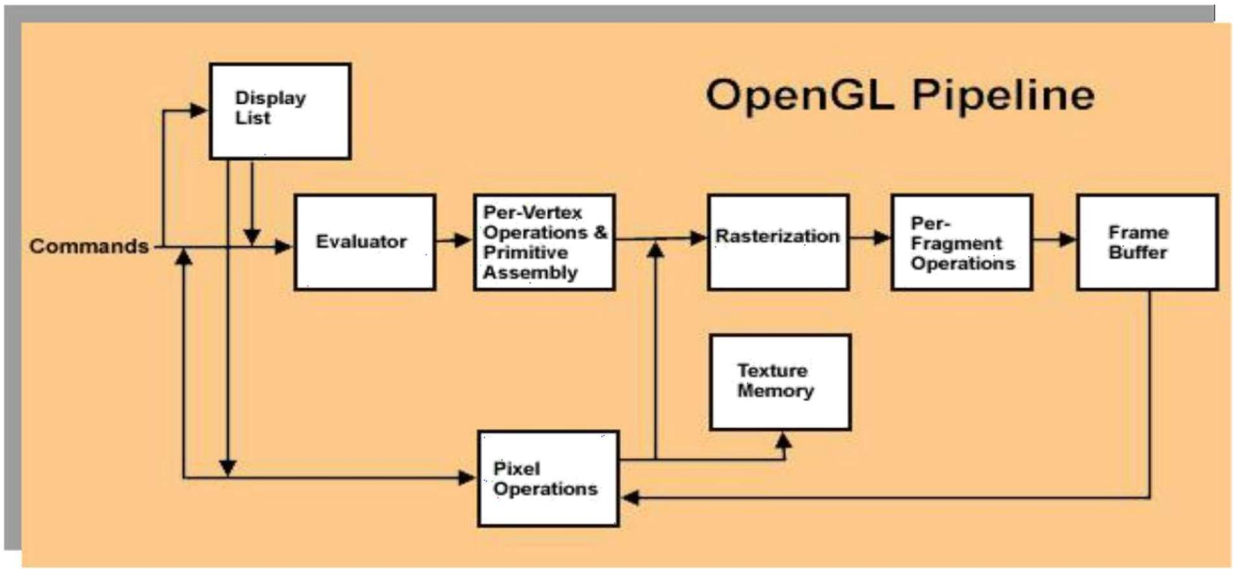


FIG:1.3 OPENGL PIPELINE ARCHITECTURE

2) OpenGL Engine And Drivers

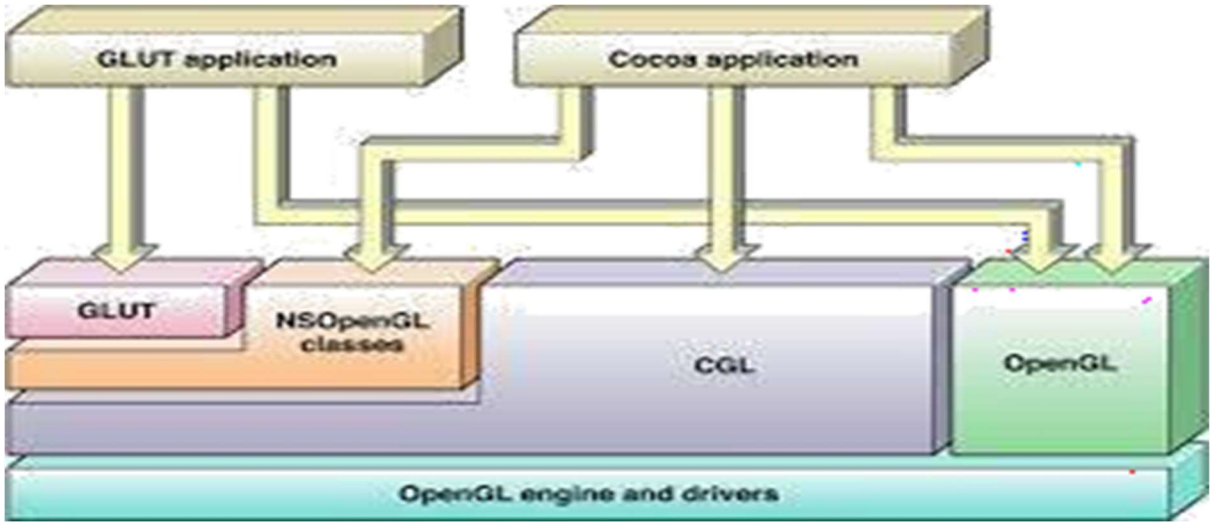
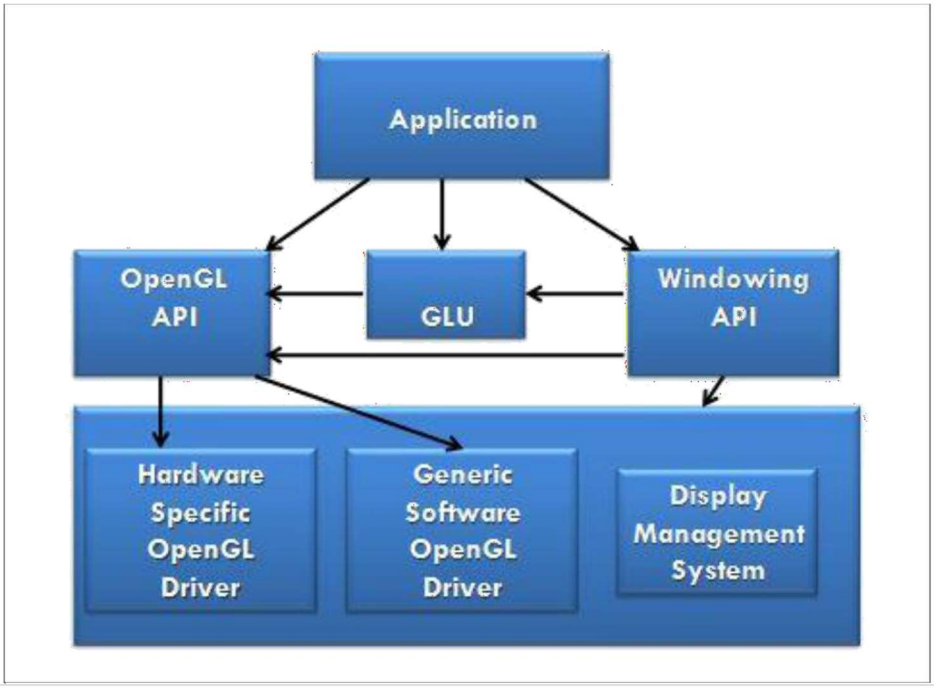


FIG:1.4 OPENGL ENGINE AND DRIVERS

3) Application Development-API's



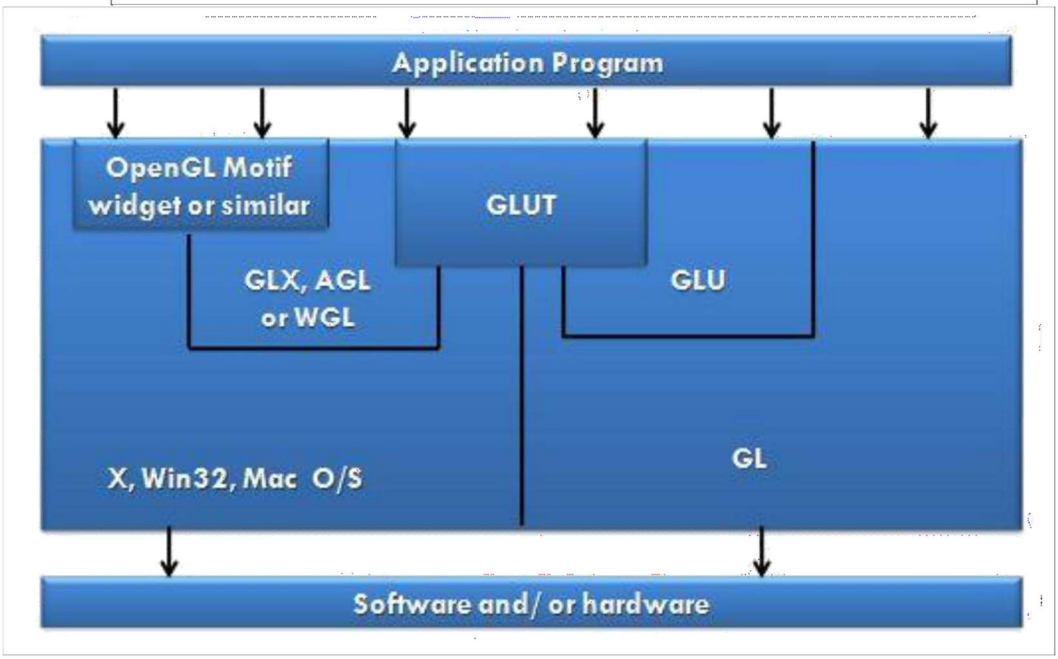


FIG:1.5 APPLICATIONS DEVELOPMENT(API'S)

Generally, applications which require more user interface support will use a library designed to support those types of features (i.e. buttons, menu and scroll bars, etc.) such as Motif or the Win32 API.

Prototype applications, or ones which do not require all the bells and whistles of a full GUI, may

choose to use GLUT instead because of its simplified programming model and window system independence.

Display Lists:

All data, whether it describes geometry or pixels, can be saved in a *display list for* current or later use. (1 alternative to retaining data in a displaylist is processing the data immediately - also known as *immediate mode*.) When a display list is executed, the retained data is sent from the display list just as if it were sent by the application in immediate mode.

Evaluators*:*

All geometric primitives are eventually described by vertices. Parametric curves and surfaces may be initially described by control points and polynomial functions called basis functions. Evaluators provide a method to derive the vertices used to represent the surface from the control points. The method is a polynomial mapping, which can produce surface normal, texture coordinates, colors, and spatial coordinate values from the control points.

Per-Vertex Operations

For vertex data, next is the "per-vertex operations" stage, which converts. The vertices into primitives. Some vertex data are transformed by 4 x 4 floating-point matrices. Spatial coordinates are projected from a position in the 3D world to a position on your screen.

If advanced features are enabled, this stage is even busier. If texturing is used, texture coordinates may be generated and transformed here. If lighting is enabled, the lighting calculations

are performed using the transformed vertex, surface normal, light source position, material properties, and other lighting information to produce a color value.

Primitive Assembly

Clipping, a major part of primitive assembly, is the elimination of portions of geometry which fall outside a half-space, defined by a plane. Point clipping simply passes or rejects vertices; line or polygon clipping can add additional vertices depending upon how the line or polygon is clipped.

In some cases, this is followed by perspective division, which makes distant geometric

objects appear smaller than closer objects. Then viewport and depth (z coordinate) operations are applied. If culling is enabled and the primitive is a polygon, it then may be rejected by a

culling test. Depending upon the polygon mode, a polygon may be drawn as points or lines. The results or this stage are complete geometric primitives, which are the transformed and clipped vertices with related color, depth, and sometimes texture-coordinate values and guidelines for the rasterization step.

Pixel Operations

While geometric data takes one path through the OpenGL rendering pipeline, pixel data takes a different route. Pixels from an array in system memory are first unpacked from one of a variety of formats into the proper number of components. Next the data is scaled, biased, and processed by a pixel map. The results are clamped and then either written into texture memory or sent to the rasterization step. If pixel data is read from the frame buffer, pixel-transfer operations (scale, bias, mapping, and clamping) are performed. Then these results are packed into an appropriate format and returned to an array in system memory. There are special pixel copy operations to copy data in the frame buffer to other parts of the frame buffer or to the texture memory. A single pass is made through the pixel transfer operations before the data is written to the texture memory or back to the frame buffer.

Texture Assembly

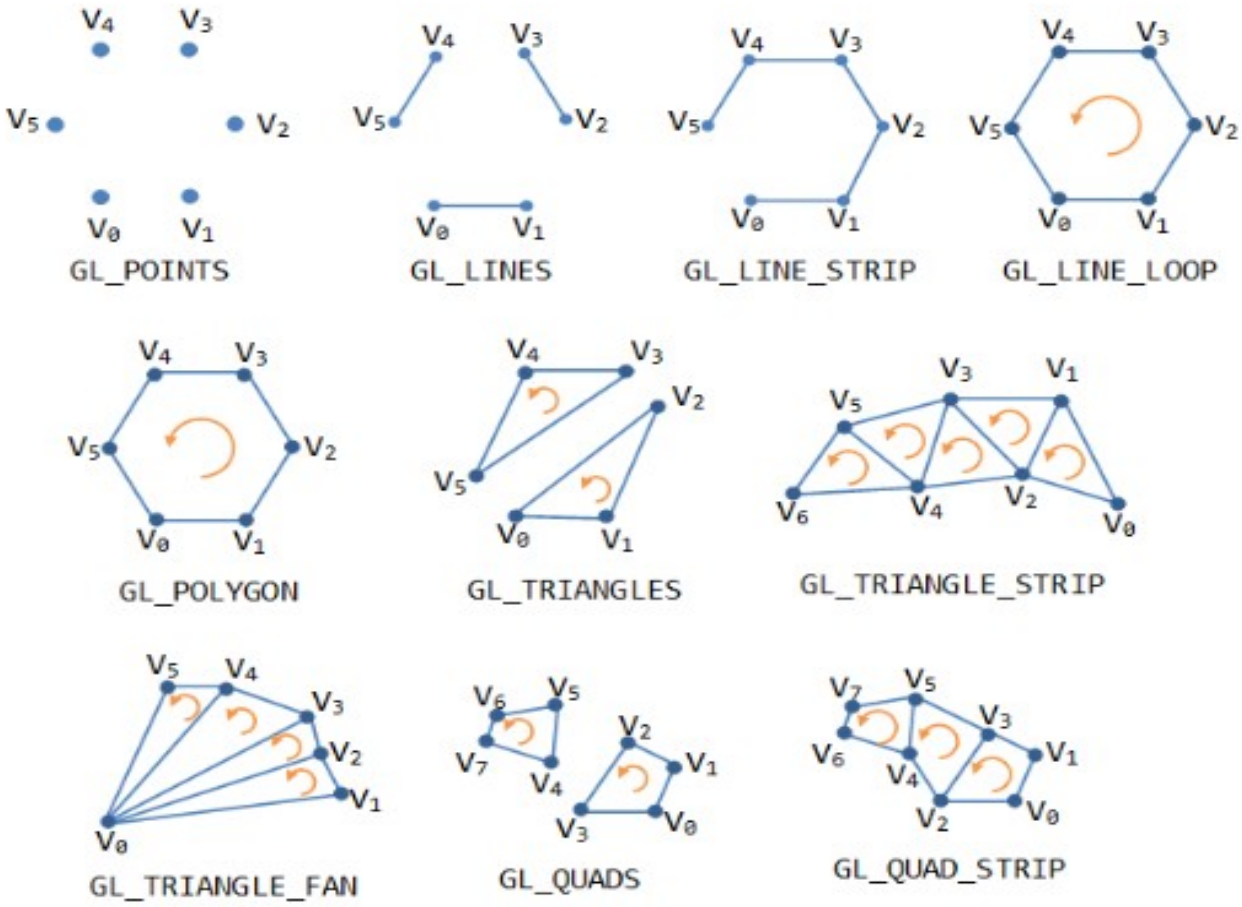
An OpenGL application may wish to apply texture images onto geometric objects to make them look more realistic. If several texture images are used, it's wise to put them into texture objects so that you can easily switch among them. Some OpenGL implementations may have special resources to accelerate texture performance. There may be specialized, high-performance texture memory. If this memory is available, the texture objects may be prioritized to control the use of this limited and valuable resource.

Rasterization

Rasterization is the conversion of both geometric and pixel data into fragments. Each

fragment square corresponds to a pixel in the frame buffer. Line and polygon stipples, line width, point size, shading model, and coverage calculations to support ant aliasing are taken into consideration as vertices are connected into lines or the interior pixels are calculated for a filled polygon. Color and depth values are assigned for each fragment square.

1.6 OpenGL Primitive



Chapter 2

REQUIREMENT SPECIFICATION

2.1 Software Requirement Specification

This section attempts to bring out the requirements and specifications as given out by the Visvesvaraya Technological University for the completion of the package. Minimum requirements expected are cursor movement, editing picture objects like point, line, circle, ellipse and polygons. Transformations on objects/selected area should be possible. User should be able to open the package do the required operations and exit from the environment.

2.2 External Interface Requirements

- **User Interface:**

  The interface for the 2D package requires for the user to have a mouse connected, and the corresponding drivers software and header files installed. For the convenience of the user, there are menus and sub -menus displayed on the screen.

- **Menus:**

  The Menus consists of various operations related to drawing on the area specified. It also has the 'clear' option for clearing the screen and also changes the background color.

- **Hardware Interface:**

  The standard output device, as mentioned earlier has been assumed to be a color monitor. It is quite essential for any graphics package to have this, as provision of color options to the user is a must. The mouse, the main input device, has to be functional. A keyboard is also required. .

  Apart from these hardware requirements, there should be sufficient hard disk space and primary memory available for proper working of the package.

2.2.1 Hardware Requirements

System	:	Intel
Frequency	:	3.0 GHz
RAM	:	8 GB
Disk Capacity	:	500 GB

2.2.2 Software Requirements:

Operating System	:	WINDOWS 10/11
Compiler, IDE	:	Ubuntu/Linux

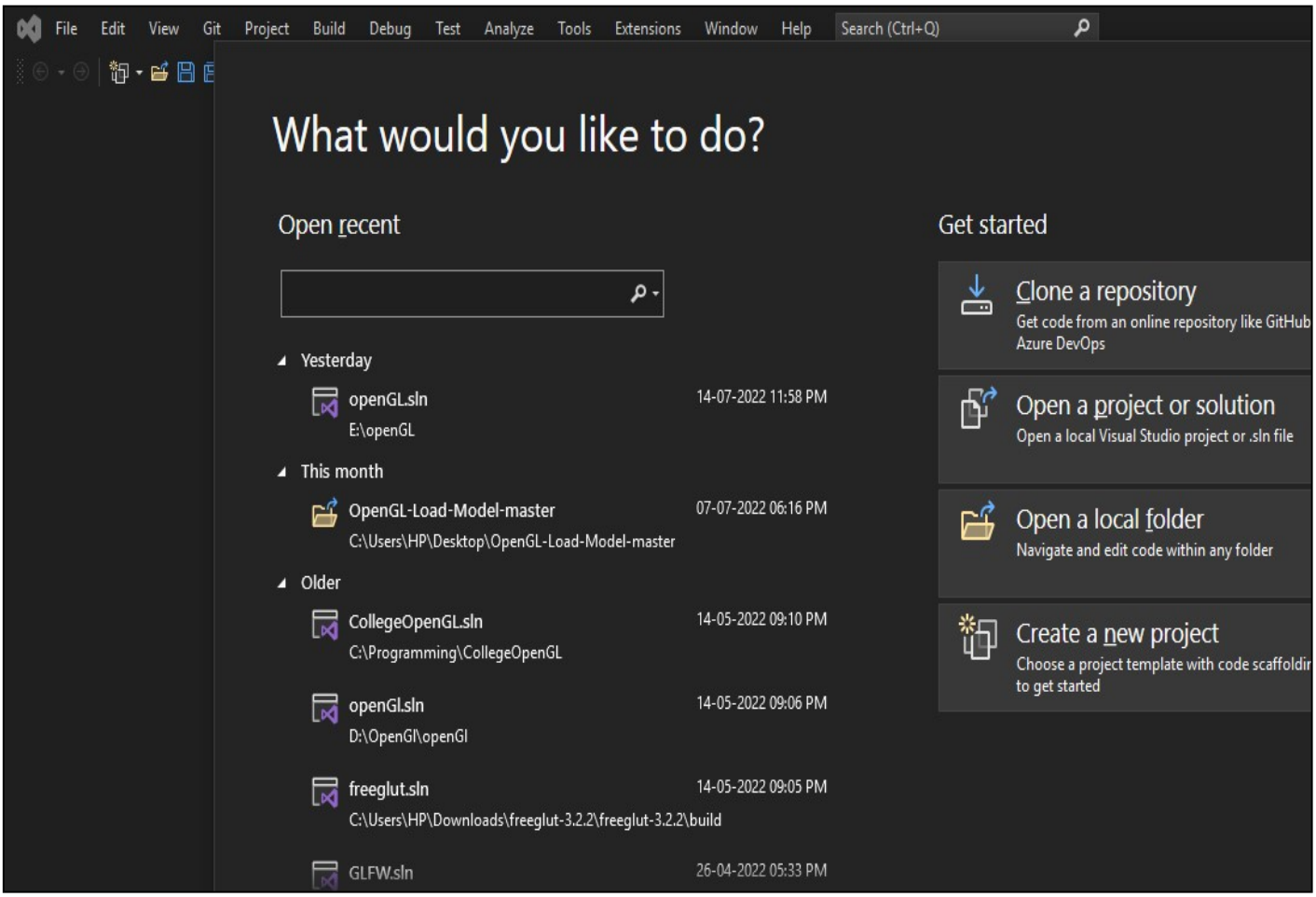


FIG:1.2.3.4 IDE

Chapter 3

ABOUT THE PROJECT

3.1 Introduction

Computer graphics today is largely interactive: 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 touch-sensitive panel on the screen. Because of the close relationship between the input devices and the display, the handling of such devices is included in the study of computer graphics. OpenGL rendering **uses the 3D View's drawing for quick preview renders**. This allows you to inspect your animatic (for object movements, alternate angles, etc.). This can also be used to preview your animations – in the event your scene is too complex for your system to play back in real-time in the 3D View.

3.2 Objectives

Over time, the race between DirectX and OpenGL has caused these technologies to become more accessible to developers, along with better documentation and an easier process of getting started with them. This project a 3D modeling application that generates and renders 3D models from user inputs.

1. Point Primitive:

   There is only one kind of point primitive:

   - **GL_POINTS**: This will cause OpenGL to interpret each individual vertex in the stream as a point. Points that have a Texture mapped onto them are often called "point sprites".

2. Line Primitive:

   There are 3 kinds of line primitives, based on different interpretations of a vertex stream.

   - **GL_LINES**: Vertices 0 and 1 are considered a line. Vertices 2 and 3 are considered a line.

And so on. If the user specifies a non-even number of vertices, then the extra vertex is ignored.

- **GL_LINE_STRIP**: The adjacent vertices are considered lines. Thus, if you pass *n* vertices, you will get *n*-1 lines. If the user only specifies 1 vertex, the drawing command is ignored.
- **GL_LINE_LOOP**: As line strips, except that the first and last vertices are also used as a line. Thus, you get *n* lines for *n* input vertices. If the user only specifies 1 vertex, the drawing command is ignored. The line between the first and last vertices happens after all of the previous lines in the sequence.

3. Triangle Primitives:

   A triangle is a primitive formed by 3 vertices. It is the 2D shape with the smallest number of vertices, so renderers are typically designed to render them. Since it is created from only 3 vertices, it is also guaranteed to be planar. There are 3 kinds of triangle primitives, based again on different interpretations of the vertex stream:

   - **GL_TRIANGLES**: Vertices 0, 1, and 2 form a triangle. Vertices 3, 4, and 5 form a triangle. And so on.
   - **GL_TRIANGLE_STRIP**: Every group of 3 adjacent vertices forms a triangle. The face direction of the strip is determined by the winding of the first triangle. Each successive triangle will have its effective face order reversed, so the system compensates for that by testing it in the opposite way. A vertex stream of *n* length will generate *n*-2 triangles.
   - **GL_TRIANGLE_FAN**: The first vertex is always held fixed. From there on, every group of 2 adjacent vertices form a triangle with the first.

Header files

The headers that are used are as follows:

- #include<GL/glut.h>: To include glut library files.
- #include<stdio.h>: To include standard input and output files.
- #include<math.h>: To include various mathematical functions.
- #include<GL/glu.h>: To include utility functions.
- #include<string.h>: To include various string built-in functions.

Functions:

- **void glScalef(TYPE sx, TYPE sy, TYPE sz)** alters the current matrix by a scaling of (sx, sy, sz). TYPE here is GLfloat. Here in the above considered example we use scaling to minimize the length of the curve at each iteration. Forthis curve we use the scale factor to be 3 units because we substitute a line by 4 lines in each iteration.

- **void glTranslatef(TYPE x, TYPE y, TYPE z)** alters the current matrix by a displacement of (x, y, z). TYPE here is GLfloat. We need to translate to display the new position of the line from the old position and also to go out to thebeginning of the next side while drawing.**void glLoadIdentity()** sets the current transformation matrix to an identity matrix.

- **void glPushMatrix()** pushes to the matrix stack corresponding to the current matrix mode.

- **void glPopMatrix()** pops from the matrix stack corresponding to the current matrix mode.

- **void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top)**

defines a two-dimensional viewing rectangle in the plane z=0.

- **void glutKeyboardFunc(myKey)** refers to the keyboard callback function. Here keyboard interface is given to quit, the user can quit by pressing 'q' and to see next example of the implementation, the user should press 'n'.

- **void glutSwapBuffers()** swaps the front and back buffers.

  User defined functions are used to color the curves in a standard cycle rainbow manner which becomes veryeasy for the user to identify the levels of recursion for the curves.

- **void glutInit(int *argc, char**argv)** Initializes GLUT< the arguments from main are passed in and can by the application.

- **void glutCreateWindow(char *title)** Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

- **void glutInitWindowSize(int width,int heights)** Specifies the initial height and width of the window in pixels.

- **void glutInitWindowPosition(int x,int y)** Specifies the initial position of the top-left corner of the window in pixels.

- **void glViewport(int x,int y,GLsizei width,GLsizei height)** Specifies a width * height viewport in pixels whose lower-left corner is at (x,y) measured fromhe origin of the window.

- **void glutMainLoop()** Cause the program to enter an event –processing loop.it should be the statement in main.

- **void glutPostRedisplay()** Requests that the display callback be executed after the current callback returns.

- **void gluLookAt(GLdouble eyex,GLdouble eyey, GLdouble eyez, GLdouble atx, GLdouble aty, GLdoubleatz, GLdouble upx, GLdouble upy, GLdouble upz)** Postmultiplies the current matrix determined by the viewer at the eye point looking at the pointwith specified up direction.

3.3 Built-in Functions

Function	Description
glAccum	Operates on the accumulation buffer.
glAddSwapHintRectWIN	Specifies a set of rectangles that are to be copied by SwapBuffers.
glAlphaFunc	Enables your application to set the alpha test function.
glAreTexturesResident	Determines whether specified texture objects are resident in texture memory.
glArrayElement	Specifies the array elements used to render a vertex.
glBegin, glEnd	Delimit the vertices of a primitive or a group of like primitives.

glClearColor	Specifies clear values for the color buffers.
glClearDepth	Specifies the clear value for the depth buffer.
glClearIndex	Specifies the clear value for the color-index buffers.
glClearStencil	Specifies the clear value for the stencil buffer.
glClipPlane	Specifies a plane against which all geometry is clipped.
glColor functions	Set the current color.
glColorMask	Enables and disables writing of frame-buffer color components.
glColorMaterial	Causes a material color to track the current color.
glColorPointer	Defines an array of colors.
glColorTableEXT	Specifies the format and size of a palette for targeted paletted textures.
glColorSubTableEXT	Specifies a portion of the targeted texture's palette to be replaced.
glCopyPixels	Copies pixels in the framebuffer.
glCopyTexImage1D	Copies pixels from the framebuffer into a one-

Chapter 4

RESULTS

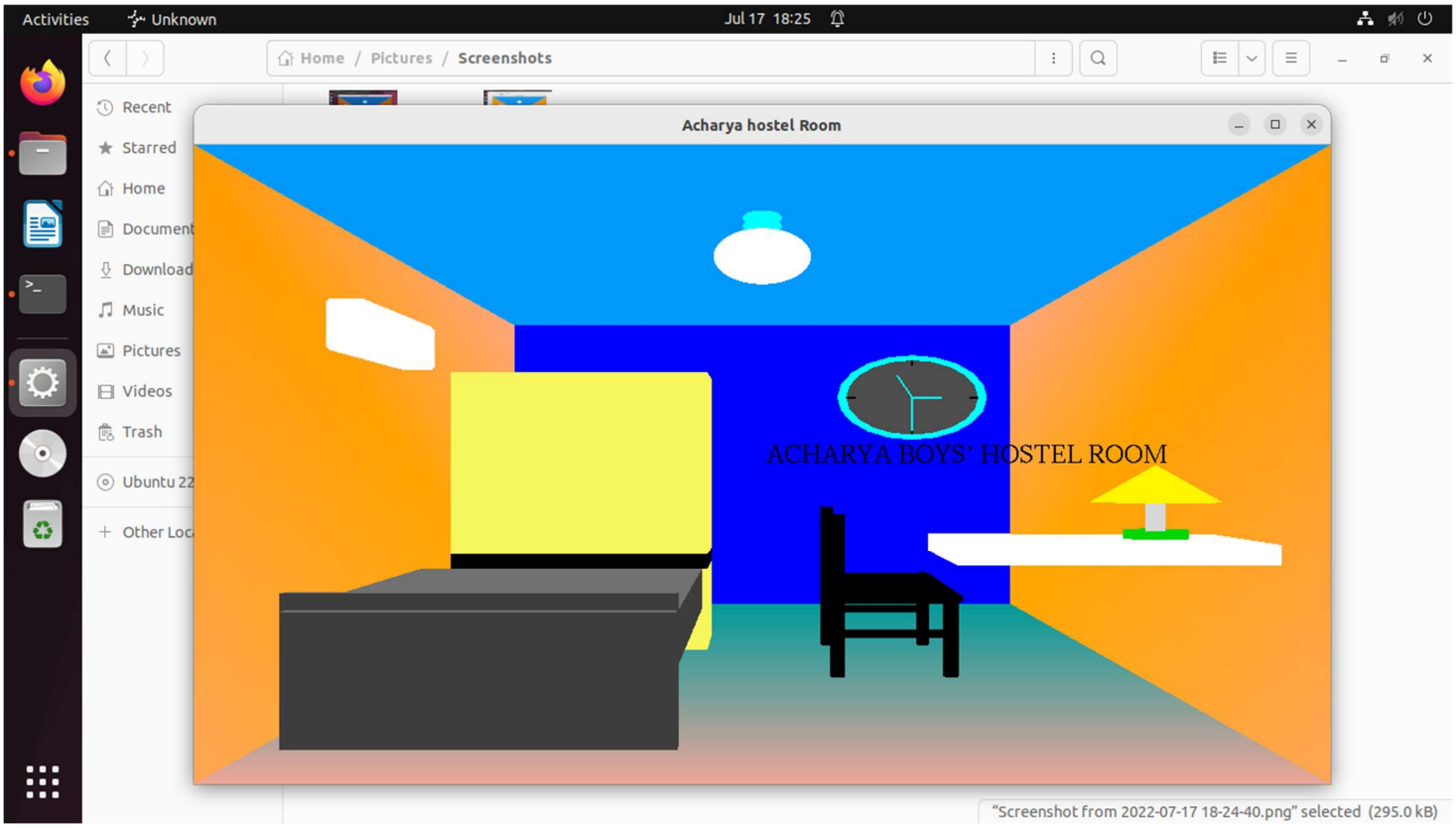


FIG 4.1 IDLE ROOM

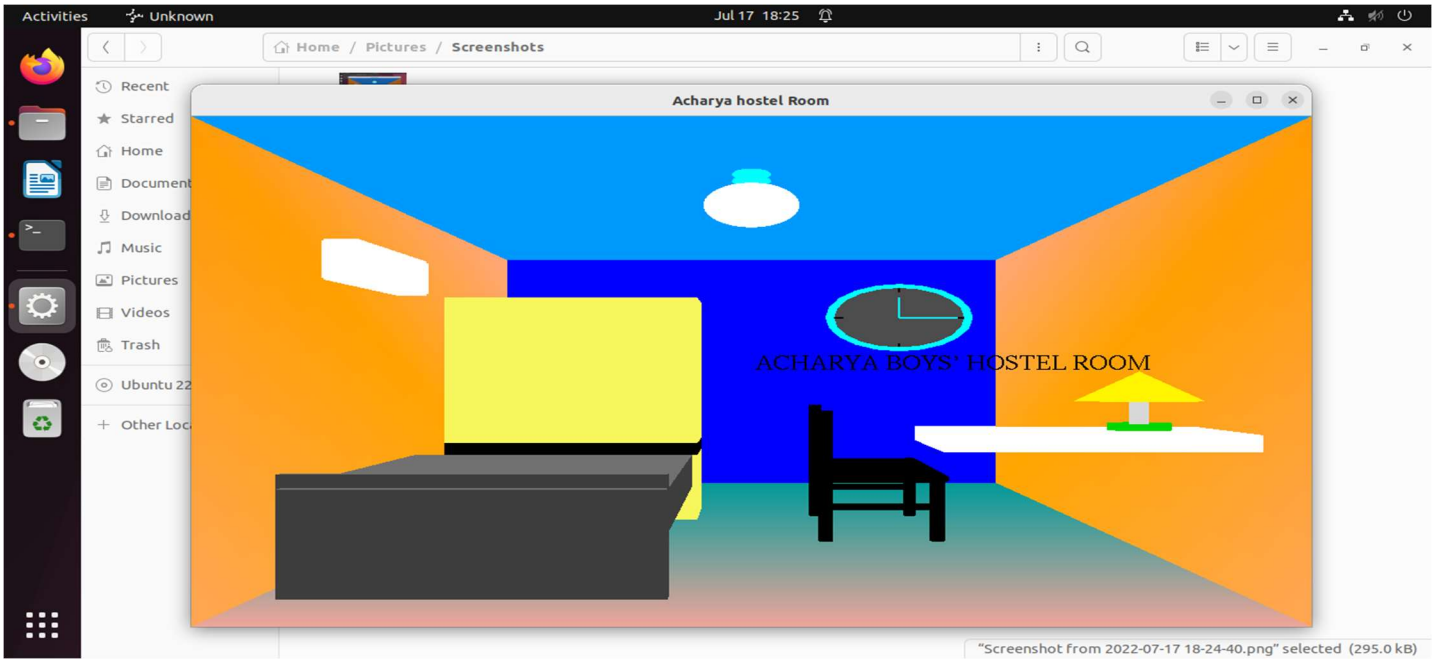


FIG 4.2 MOVING CLOCK

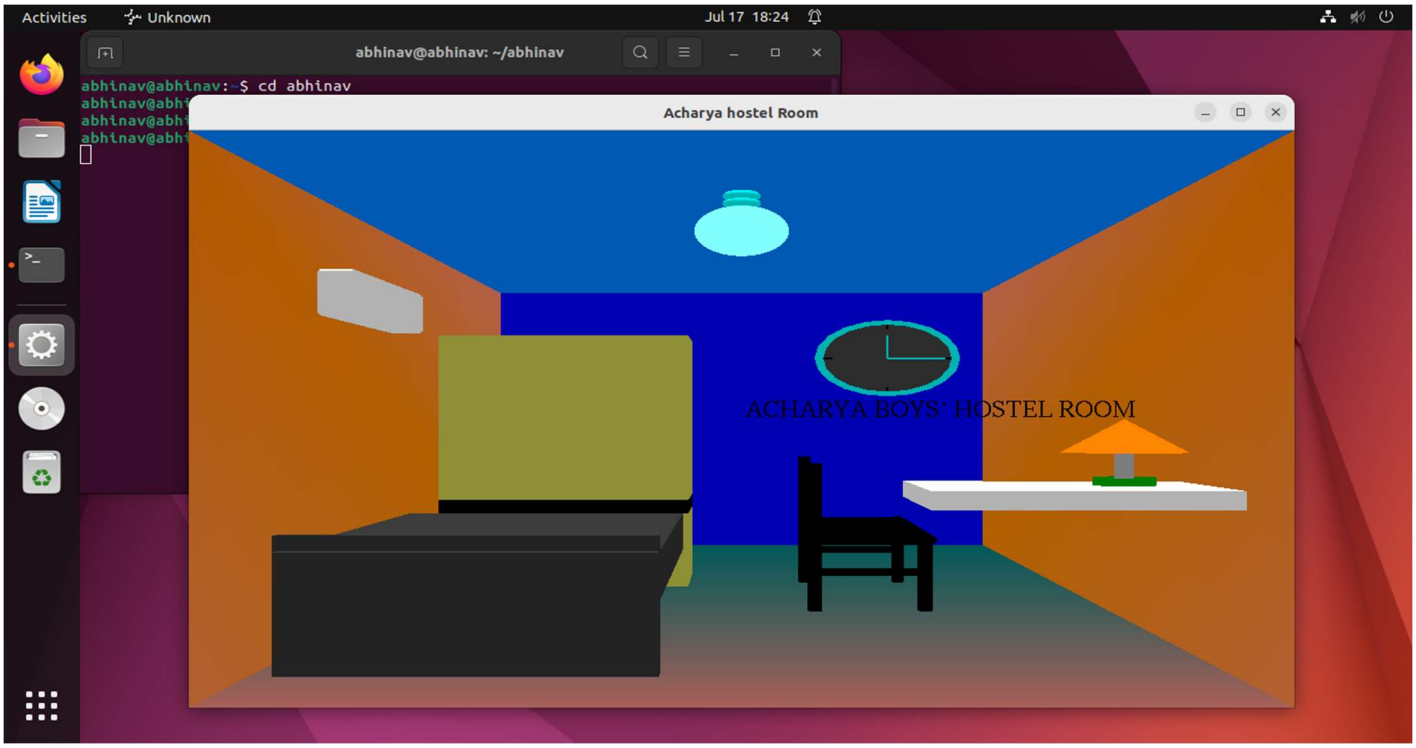


FIG 4.3 DIMMED LIGHT

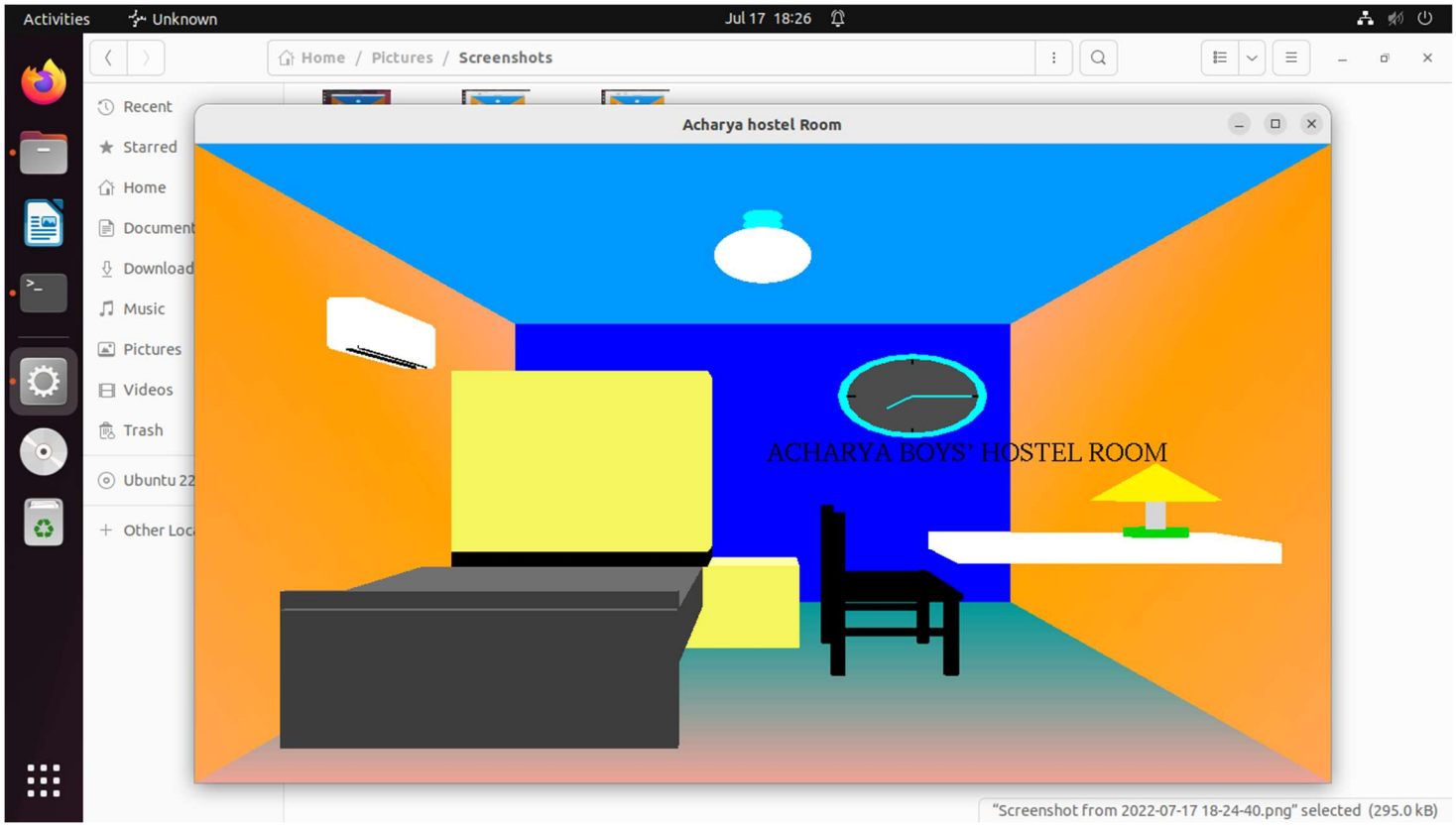


FIG 4.4 AC ON, CHAIR MOVED & DRAWER OPEN