VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnanasangama', Belagavi-590014, Karnataka

A COMPUTER GRAPHICS MINI PROJECT REPORT

ON

"TAJ MAHAL SIMULATION"

SUBMITTED IN PARTIAL FULFILLMENT FOR THE ACADEMIC YEAR 2022-23

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Hrishikesh N	**1AH20CS045**
Ayush Thakur	**1AH20CS021**

UNDER THE GUIDANCE OF

Mrs. Ganga

Department of CSE

ACS COLLEGE OF ENGINEERING

#74,kambipura Mysore road, Bengaluru-560074

ACS COLLEGE OF ENGINEERING

#74, Kambipura, Mysore Road, Bengaluru-560074

Department of Computer Science and Engineering

CERTIFICATE

Certificate that the mini project report on the topic "TAJ MAHAL SIMULATION" has been successfully submitted by **Hrishikesh N** (1AH20CS045), **Ayush Thakur** (1AH20CS021). A bonafide students at ACS college of engineering affiliated to Visvesvaraya technological university, Belagavi during the year 2022-2023. It is certified that all corrections/suggestions indicate for Internal assessment have been incorporated in the report submitted to the department. The mini project report has been approved as it satisfies the academic requirements in respect of the mini project work as prescribed in 6th semester.

Signature of the guide	**Signature of HOD**
Ms. Ganga	T. Senthil Kumaran
Assistant Professor, CSE	Professor & HOD, CSE
ACSCE, Bengaluru	ACSCE, Bengaluru

Name of the examiner	**Signature with Date**
1. ________________	________________
2. ________________	________________

ACKNOWLEDGEMENT

We are delighted to thank our honourable Chairman **Sri. A. C. Shanmugam** for extending his support by providing an excellent infrastructure during the tenure of the course. The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned the effort with success.

We are submitting the Mini Project Report on **"TAJ MAHAL SIMULATION"**, as per the scheme of **Visvesvaraya Technological University, Belagavi.**

We express our gratitude to **Dr. M. S. Murali, Principal**, ACS College of Engineering, Bengaluru, for giving us excellent facilities and academic ambience; and for his timely help and inspiration during the tenure of the course.

I express my sincere regards and thanks to **T. Senthil Kumaran, Associate Professor & HOD**, Computer Science and Engineering, ACSCE, Bengaluru for the encouragement and support throughout encouragement the work.

With the profound sense of gratitude, I acknowledge the guidance support and encouragement to my guide **Mrs. Ganga BM, Assistant Professor**, Computer Science and Engineering, ACSCE, Bengaluru.

Hrishikesh N (1AH20CS045)

Ayush Thakur (1AH20CS021)

ABSTRACT

The project "**Simulation of Taj Mahal using OpenGL**" aims to recreate the iconic Taj Mahal, a UNESCO World Heritage Site, using the powerful computer graphics library OpenGL. The Taj Mahal, located in Agra, India, is renowned for its architectural brilliance and serves as a significant symbol of love and grandeur.

This simulation project leverages the capabilities of OpenGL to develop a visually stunning and interactive 3D model of the Taj Mahal. By employing various rendering techniques, such as texture mapping, lighting, and shading, the simulation aims to provide a realistic representation of the monument.

The project involves extensive research into the architectural details, measurements, and historical context of the Taj Mahal. This knowledge is translated into a digital representation, with meticulous attention given to the intricate designs, intricate carvings, and overall structural accuracy.

The implementation of the simulation is carried out using OpenGL, a widely adopted graphics library known for its flexibility and cross-platform compatibility. By utilizing the library's rendering pipeline, the project achieves efficient rendering of the 3D model, ensuring smooth animations and immersive experiences for users.

Overall, the "Simulation of Taj Mahal using OpenGL" project combines art, technology, and historical research to create an immersive and visually appealing experience. By bringing the Taj Mahal to life in a virtual environment, it aims to preserve and promote cultural heritage while offering a platform for exploration and learning.

LIST OF FIGURES

LIST OF ABBREVIATIONS

- OpenGL: Open Graphics Library
- GL: Graphics library
- GLU: OpenGL Utility Library
- GLUT: OpenGL Utility Toolkit

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real-world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

1.2 OpenGL

OpenGL (Open Graphics Library) is a standard specification defining a cross 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 3D scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation. It is also used in video games, where it competes with direct 3D on Microsoft Windows Platforms. OpenGL is managed by the non-profit technology consortium, the Khronos group Inc.

OpenGL serves two main purposes:

- ➢ To hide the complexities of interfacing with different 3D accelerators, by presenting programmer with a single, uniform API
- ➢ To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set.

OpenGL has historically been influential on the development of 3D accelerator, promoting a base level of functionality that is now common in consumer level hardware:

- Rasterized points, lines and polygons are basic primitives.
- A transform and lighting pipeline.
- Z buffering.
- Texture Mapping.
- Alpha Blending.

1.2.1 OpenGL Graphics Architecture:

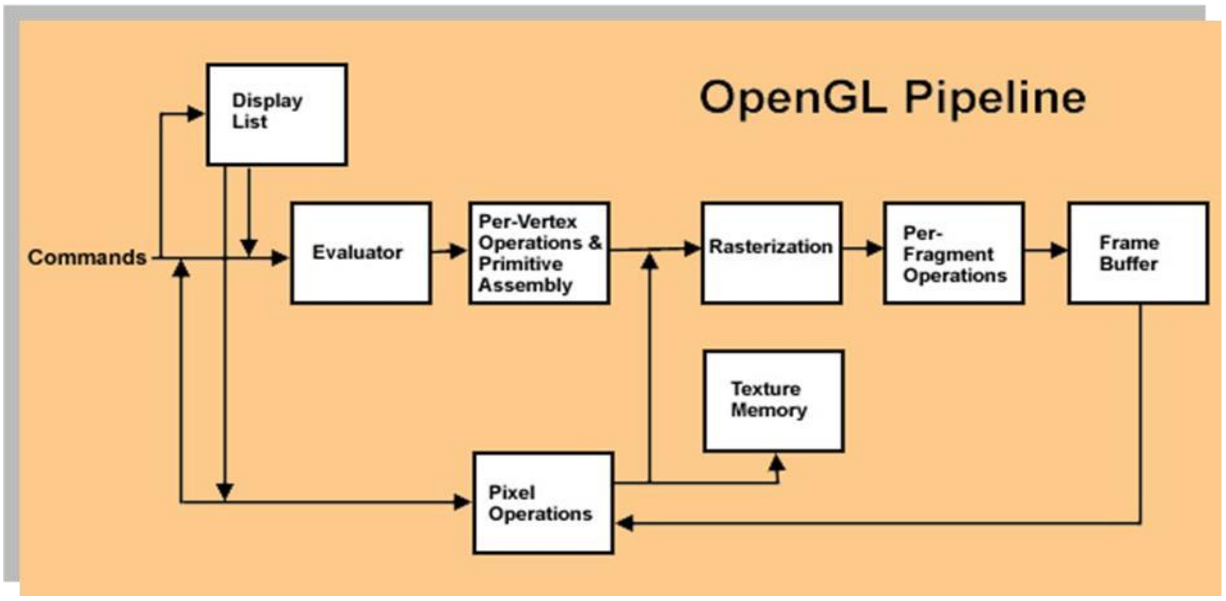

Figure 1.1 OpenGl Graphics Architecture

Display Lists:

All data, whether it describes geometry or pixels, can be saved in a display list for current or later use. 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.

- **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.

- **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.

- **Pixel Operation:**

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.

- **Rasterization:**

Rasterization is the conversion of both geometric and pixel data into fragments. Each fragment square corresponds to a pixel in the framebuffer. Colour and depth values are assigned for each fragment square.

- **Fragment Operations:**

Before values are actually stored into the framebuffer, a series of operations are performed that may alter or even throw out fragments.

1.3 Project Goal

The aim of this project is to show the shadow implementation using OPENGL which include Movement, Light properties also transformation operations like translation, rotation, scaling etc on objects. The package must also have a user-friendly interface.

1.4 Scope

It is developed in ECLIPSE. It has been implemented on Windows platform. The 3-D graphics package designed here provides an interface for the users for handling the display and manipulation of Celestial Exploratory. The Keyboard is the main input device used.

CHAPTER 2

LITERATURE SURVEY

The literature survey aims to provide a comprehensive overview of the Taj Mahal, one of the most iconic and universally recognized monuments in the world. The Taj Mahal, located in Agra, India, is a UNESCO World Heritage Site and a symbol of love and architectural brilliance. This survey examines various aspects of the Taj Mahal, including its historical significance, architectural features, cultural impact, and artistic elements. It also delves into the preservation efforts, controversies surrounding the monument, and its status as a tourist attraction. By reviewing a range of scholarly articles, books, and research papers, this survey offers a holistic understanding of the Taj Mahal and its significance as a cultural and architectural marvel.

It provides a comprehensive overview of the Taj Mahal, exploring its historical, architectural, cultural, and artistic significance. It also sheds light on the preservation efforts, controversies, and tourism-related aspects associated with this magnificent monument. By examining a range of scholarly works, this survey aims to contribute to the existing body of knowledge on the Taj Mahal and inspire further research and exploration of this iconic landmark.

The Taj Mahal is a world-renowned mausoleum located in Agra, Uttar Pradesh, India. It is considered one of the most exquisite examples of Mughal architecture and is recognized as a UNESCO World Heritage Site. The monument was commissioned by the Mughal emperor Shah Jahan in memory of his beloved wife, Mumtaz Mahal, and it stands as a symbol of love and devotion.

Key Facts about the Taj Mahal:

- Construction: The construction of the Taj Mahal began in 1632 and took approximately 22 years to complete. It involved the labor of thousands of skilled artisans and craftsmen from various regions.
- Architecture: The Taj Mahal combines elements of Islamic, Persian, and Indian architectural styles. It features a white marble facade adorned with intricate carvings, calligraphy, and inlaid semi-precious stones. The central dome is surrounded by four minarets, and the complex includes beautiful gardens and reflective pools.
- Symbolism: The Taj Mahal is often described as a symbol of eternal love. It represents the profound grief of Shah Jahan over the loss of his wife, as well as his devotion to her. The monument's symmetrical design and meticulous details reflect the Mughal belief in paradise and the perfection of symmetry.
- Cultural Significance: The Taj Mahal is an integral part of India's rich cultural heritage. It has inspired numerous poets, writers, musicians, and artists across generations. The monument has also been featured in various films, documentaries, and literary works, further amplifying its cultural significance.
- Preservation Efforts: Over the years, the Taj Mahal has faced challenges due to environmental pollution and natural decay. Preservation efforts have been undertaken to protect and restore the monument, including measures to reduce air pollution in the surrounding area.
- Visitor Experience: The Taj Mahal attracts millions of visitors from around the world each year. Visitors can explore the exquisite architecture, stroll through the gardens, and learn about the history and significance of the monument. However, due to its popularity, there are certain visitor regulations in place to ensure its preservation.

- Controversies: The Taj Mahal has been the subject of various controversies, including debates over its origins and claims by some groups questioning its historical significance. However, extensive historical evidence supports the commonly accepted narrative of the monument's construction by Shah Jahan.

Today, the Taj Mahal remains an iconic symbol of love, architectural brilliance, and cultural heritage. Its beauty and grandeur continue to captivate visitors, making it one of the most visited and cherished landmarks in the world.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS:

- 128 MB of RAM, 256 MB recommended.
- 110 MB of hard disk space required; 40 MB additional hard disk space required for installation (150 MB total).

3.2 SOFTWARE REQUIREMENTS:

The Taj Mahal simulator has been designed for Windows. OpenGL libraries are used and hence ECLIPSE is required.

Development Platform	**:** Windows 7 & above
Language	**:** C
Tool	**:** Eclipse
Library	**:** OpenGL

CHAPTER 4

SYSTEM DESIGN

1. **User Interface:**

Implement a user-friendly interface that provides controls for navigation, camera manipulation, and interaction with the virtual environment.

Design an intuitive menu system for accessing different features, options, and information about the Taj Mahal.

2. **Rendering Engine:**

Utilize OpenGL as the primary rendering engine for generating real-time graphics and visual effects.

Implement advanced rendering techniques such as lighting models, shading algorithms, texture mapping, and geometry processing to achieve realistic and immersive visuals.

3. **3D Model Creation:**

Create a high-quality 3D model of the Taj Mahal using modeling software such as Maya, 3ds Max, or Blender.

Ensure accurate representation of architectural details, textures, and colors to maintain the fidelity of the monument.

4. **Texturing and Materials:**

Apply appropriate textures and materials to the 3D model to enhance its visual realism.

Utilize high-resolution textures and normal maps to capture intricate surface details of the Taj Mahal, including marble, inlays, and other decorative elements.

5. **Lighting and Shadows:**

Implement lighting techniques such as global illumination, ambient occlusion, and shadow mapping to create realistic lighting conditions within the virtual environment.

Accurately represent the sun's position and its effect on the Taj Mahal's appearance throughout the day.

6. **Interaction and Exploration:**

Enable user-controlled navigation and camera manipulation to allow users to explore the virtual environment from various angles and perspectives.

Implement interaction mechanisms, such as click or hover-based hotspots, to provide users with information and historical context about different parts of the Taj Mahal.

7. **Optimization and Performance:**

Optimize the rendering pipeline to ensure smooth performance and real-time rendering, even on lower-end hardware.

Employ techniques such as level-of-detail rendering and occlusion culling to manage scene complexity and improve performance.

8. **Educational Content:**

Incorporate informative text, images, and audio narration to educate users about the history, architectural significance, and cultural importance of the Taj Mahal.

Include interactive elements that allow users to learn about specific architectural features, construction techniques, and historical events related to the monument.

9. **Compatibility and Deployment:**

Ensure the project is compatible with a range of hardware configurations and operating systems to maximize accessibility for users.

Package the simulation project for easy deployment, considering factors such as installation, updates, and distribution across different platforms.

10. **Testing and Feedback:**

Conduct rigorous testing to identify and fix any bugs, glitches, or performance issues in the simulation.

Gather user feedback and incorporate improvements based on user experience and suggestions to enhance the overall quality and usability of the project.

CHAPTER 5

SOURCE CODE

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>

#define GL_GLEXT_PROTOTYPES
#ifdef __APPLE__
#include <OpenGL/gl.h>
#include <OpenGl/glu.h>
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#include <GL/gl.h>
#include <GL/glu.h>
#endif


#include <cmath>
#include <queue>
#include <utility>
#include <bits/stdc++.h>
#define PI 3.1415926535898
#define Cos(th) cos(PI/180*(th))
#define Sin(th) sin(PI/180*(th))
#define DEF_D 5

float _angle=44.0f;
//float z=0.0;
float r =(100.0f/600.0f);
float rx=(70.0f/600.0f);
float sx=1.0f,sy=1.0f,sz=1.0f;
float lx = 0.0, lz = -1.0;
float x = 0.0, z = 5.0, xRot, yRot, zRot;
using namespace std;

void drawbase(int xx1,int xx2,int zz1,int zz2)
{
        //glNormal3f(0.0f, 0.0f, 1.0f);
    glVertex3f(xx1*0.75f, 0, 0.75f*zz1);
    glVertex3f(xx1*0.75f, -0.15f, 0.75f*zz1);
    glVertex3f(-0.75f*xx2, -0.15f, 0.75f*zz2);
    glVertex3f(-0.75f*xx2, 0, 0.75f*zz2);



}

void drawBase()
{
    glBegin(GL_QUADS);
   glColor3f(1.0f, 1.0f, 1.0f);
```

```
    //Front
    drawbase(1,1,1,1);

    //Right
    glNormal3f(1.0f, 0.0f, 0.0f);
   drawbase(1,-1,1,-1);

    //Back
    glNormal3f(0.0f, 0.0f, -1.0f);
    drawbase(1,1,-1,-1);
    //Left
    glNormal3f(-1.0f, 0.0f, 0.0f);
    drawbase(-1,1,1,-1);
    glEnd();


}

void drawpoints(float centerx,float centery,float xc,float yc,int zz,int
xx,float cons)
{
        if(zz==1)
        {
                //for(int i=centerx-x;i<=centerx+x;i++)
                glColor3f(0,0,0);
                glBegin(GL_LINES);
                glVertex3f(centerx+xc,centery+yc,cons);
                glVertex3f(centerx-xc,centery+yc,cons);
                glEnd();
        }
        else if(xx==1)
        {
                //for(int i=centerx-x;i<=centerx+x;i++)
                glColor3f(0,0,0);
                glBegin(GL_LINES);
                glVertex3f(cons,centery+yc,centerx+xc);
                glVertex3f(cons,centery+yc,centerx-xc);
                glEnd();

        }



}

void displayCIRCLE(float centerx,float centery,float R ,int zz,int xx,float
cons)
{
  float xc=0,yc=0;
  for(int i=0;i<=90;i++)
  {
    xc=R*Cos(i);
    yc=R*Sin(i);

    drawpoints(centerx,centery,xc,yc,zz,xx,cons);
    }
 // glEnd();
```

```
  }

void draw_door_z(int zz)
{
   glColor3f(0,0,0);

        glBegin(GL_LINES);
   // glNormal3f(0.0f, 0.0f, zz*1.0f);

    for(float i=-1*rx;i<=rx;i+=0.001f)
   { glVertex3f(i, 0, zz*0.5f);
    glVertex3f(i, 0.25f,zz* 0.5f);}

    glEnd();

    glBegin(GL_LINES);
    for(float i=0;i<=0.25f;i+=0.001f)
    {
       glVertex3f(rx,i,zz*0.5f);
       glVertex3f(-1*rx,i,zz*0.5f);
        }
        glEnd();



    displayCIRCLE(0,0.25f,rx,1,0,zz*0.5f);

}


void draw_door_y(int yy)
{
   glColor3f(0,0,0);

        glBegin(GL_LINES);
   // glNormal3f(0.0f, 0.0f, zz*1.0f);
        for(float i=0;i<=0.25f;i+=0.001f){
    glVertex3f(yy*0.5f, i, rx);
    glVertex3f(yy*0.5f,i,-1*rx);}
    glEnd();

    glBegin(GL_LINES);
    for(float i=-1*rx;i<=rx;i+=0.001f){
    glVertex3f(yy*0.5f, 0, i);
     glVertex3f(yy*0.5f, 0.25f,i);}
    glEnd();

    displayCIRCLE(0,0.25f,rx,0,1,yy*0.5f);

}


void draw_window_z(int zz)
{
}

void draw_smallfacey()
{
        glBegin(GL_QUADS);
  // glColor3f(1.0f, 0.0f, 0.0f);
    //Front
    glNormal3f(1.0f, 0.0f, 0.0f);
    glVertex3f(0.5f, 0, r);
    glVertex3f(0.5f,0.35f, r);
    glVertex3f(0.5f, 0.35f, 2*r);
    glVertex3f(0.5f, 0, 2*r);
    glEnd();


}


void draw_smallface()
{
        glBegin(GL_QUADS);
  // glColor3f(1.0f, 0.0f, 0.0f);
    //Front
    glNormal3f(0.0f, 0.0f, 1.0f);
    glVertex3f(r, 0, 0.5f);
    glVertex3f(r,0.35f, 0.5f);
    glVertex3f(2*r, 0.35f, 0.5f);
    glVertex3f(2*r, 0, 0.5f);
    glEnd();


}

void draw_face()
{
        glBegin(GL_QUADS);
  // glColor3f(1.0f, 0.0f, 0.0f);
    //Front
    glNormal3f(0.0f, 0.0f, 1.0f);
    glVertex3f(-1*r, 0, 0.5f);
    glVertex3f(-1*r, 0.45f, 0.5f);
    glVertex3f(r, 0.45f, 0.5f);
    glVertex3f(r, 0, 0.5f);
    glEnd();
}


void draw_facey()
{
        glBegin(GL_QUADS);
  // glColor3f(1.0f, 0.0f, 0.0f);
    //Front
    glNormal3f(1.0f, 0.0f, 0.0f);
    glVertex3f(0.5f, 0, r);
    glVertex3f(0.5f, 0.45f, r);
    glVertex3f(0.5f, 0.45f, -1*r);
    glVertex3f(0.5f, 0, -1*r);
    glEnd();
}

void draw_slnt(int xx,int yy)
{
glBegin(GL_QUADS);
        glVertex3f(2*r*xx, 0, 0.5f*yy);
        glVertex3f(2*r*xx, 0.35f, 0.5f*yy);
        glVertex3f(0.5f*xx, 0.35f, 2*r*yy);
       glVertex3f(0.5f*xx, 0, r*yy);
        glEnd();


}


void draw_cylinder(float r1 , float r2, float h)
{

glRotatef(-90.0f,1,0,0);

GLUquadricObj *quadric=gluNewQuadric();
gluQuadricNormals(quadric, GLU_SMOOTH);
gluQuadricOrientation(quadric,GLU_OUTSIDE);
gluCylinder(quadric,r1,r2,h,50,50);
gluDeleteQuadric(quadric);
}

void draw_hemisphere(float r1,float r2){

glPushMatrix();

GLdouble eqn[4] = {0.0, 1.0, 0.0,r2};
   /*    clip lower half -- y < 0          */
   glClipPlane (GL_CLIP_PLANE0, eqn);
   glEnable (GL_CLIP_PLANE0);
   glRotatef (90.0, 1.0, 0.0, 0.0);
   glutSolidSphere(r1, 40, 40);
   glDisable(GL_CLIP_PLANE0);
glPopMatrix();
 //  glFlush ();

}


void draw()
{
         glRotatef(_angle,0.0f,1.0f,0.0f);
         glScalef(sx,sy,sz);
        glPushMatrix();
        glColor3f(1,0,0);
        draw_face();
        glPopMatrix();

        glPushMatrix();
        glColor3f(0,1,0);
        draw_smallface();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(-0.5f,0,0);
        glColor3f(0,0,1);
        draw_smallface();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(0,0,-1.0f);
        glColor3f(1,0,0);
        draw_face();
        glPopMatrix();


        glPushMatrix();

        glColor3f(0,1,0);
         glTranslatef(0,0,-1.0f);
        draw_smallface();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(-0.5f,0,-1.0f);
        glColor3f(0,0,1);
        draw_smallface();
        glPopMatrix();

        glPushMatrix();
        //glRotatef(-90,0,0,1);
        glColor3f(1,0,0);
        draw_facey();
        glPopMatrix();

        glPushMatrix();
        glColor3f(0,1,0);
        //glRotatef(-90,0,0,1);
        draw_smallfacey();
        glPopMatrix();

        glPushMatrix();
        //glRotatef(-90,0,0,1);
        glTranslatef(0,0,-0.5f);
        //glRotatef(90,0,0,1);
        glColor3f(0,0,1);
        draw_smallfacey();
        glPopMatrix();

        glPushMatrix();
        //glRotatef(-90,0,0,1);
        glColor3f(1,0,0);
         glTranslatef(-1.0f,0,0);
        draw_facey();
        glPopMatrix();
```

```
 glPushMatrix();
 glColor3f(0,1,0);
  glTranslatef(-1.0f,0,0);
 //glRotatef(-90,0,0,1);
 draw_smallfacey();
 glPopMatrix();

 glPushMatrix();
 //glRotatef(-90,0,0,1);
 glTranslatef(-1.0f,0,-0.5f);
 //glRotatef(90,0,0,1);
 glColor3f(0,0,1);
 draw_smallfacey();
 glPopMatrix();


 glPushMatrix();
 glColor3f(1,1,0);
 draw_slnt(1,1);
 glPopMatrix();

 glPushMatrix();
 glColor3f(1,1,0);
 draw_slnt(1,-1);
 glPopMatrix();

 glPushMatrix();
 glColor3f(1,1,0);
 draw_slnt(-1,1);
 glPopMatrix();

 glPushMatrix();
 glColor3f(1,1,0);
 draw_slnt(-1,-1);
 glPopMatrix();

  glPushMatrix();
  drawBase();
  glPopMatrix();

  glPushMatrix();
  glTranslatef(0.7f,0,0.7f);
  glColor3f(1,0,1);
  draw_cylinder(r/4,r/5,0.6f);
  glPopMatrix();

  glPushMatrix();
  glColor3f(0.5,1.0,0);
  glTranslatef(0.7f,0.6f,0.7f);
  draw_cylinder(r/4,r/4,0.035f);
  glPopMatrix();
// glPopMatrix();

  glPushMatrix();
  glColor3f(0.5,1.0,0.5);
  glTranslatef(0.7f,0.65f,0.7f);
  draw_hemisphere(r/4,0.01f);
```

```
glPopMatrix();


 glPushMatrix();
glTranslatef(0.7f,0,-0.7f);
glColor3f(1,0,1);
draw_cylinder(r/4,r/5,0.6f);
glPopMatrix();

 glPushMatrix();
glColor3f(0.5,1.0,0);
glTranslatef(0.7f,0.6f,-0.7f);
draw_cylinder(r/4,r/4,0.035f);
glPopMatrix();

glPushMatrix();
glColor3f(0.5,1.0,0.5);
glTranslatef(0.7f,0.65f,-0.7f);
draw_hemisphere(r/4,0.01f);

glPopMatrix();


 glPushMatrix();
glTranslatef(-0.7f,0,0.7f);
glColor3f(1,0,1);
draw_cylinder(r/4,r/5,0.6f);
glPopMatrix();

 glPushMatrix();
glColor3f(0.5,1.0,0);
glTranslatef(-0.7f,0.6f,0.7f);
draw_cylinder(r/4,r/4,0.035f);
glPopMatrix();

glPushMatrix();
glColor3f(0.5,1.0,0.5);
glTranslatef(-0.7f,0.65f,0.7f);
draw_hemisphere(r/4,0.01f);

glPopMatrix();

 glPushMatrix();
glTranslatef(-0.7f,0,-0.7f);
glColor3f(1,0,1);
draw_cylinder(r/4,r/5,0.6f);
glPopMatrix();

 glPushMatrix();
glColor3f(0.5,1.0,0);
glTranslatef(-0.7f,0.6f,-0.7f);
draw_cylinder(r/4,r/4,0.035f);
glPopMatrix();

glPushMatrix();
glColor3f(0.5,1.0,0.5);
```

```
glTranslatef(-0.7f,0.65f,-0.7f);
draw_hemisphere(r/4,0.01f);

glPopMatrix();


 glPushMatrix();
  glColor3f(0,1,1);
  glTranslatef(0,0.35f,0);
draw_cylinder(r-0.033,r-0.033,0.27f);
glPopMatrix();

glPushMatrix();
glTranslatef(0,0.62f,0);
glTranslatef(0,0.1f,0);
draw_hemisphere(r,0.15f);
glPopMatrix();



glPushMatrix();
glColor3f(0,0.5,0);
glTranslatef(0.3f,0.35f,0.3f);
draw_cylinder(0.045,0.045,0.14f);
glPopMatrix();

glPushMatrix();
glColor3f(1.0,0,1.0);
glTranslatef(0.3f,0.45f,0.3f);
draw_cylinder(0.047,0.047,0.04f);
glPopMatrix();

glPushMatrix();
glColor3f(4,4,4);
glTranslatef(0.3f,0.50f,0.3f);
draw_hemisphere(0.046,0.1f);

glPopMatrix();



glPushMatrix();
glColor3f(0,0.5,0);
glTranslatef(-0.3f,0.35f,0.3f);
draw_cylinder(0.045,0.045,0.14f);
glPopMatrix();

glPushMatrix();
glColor3f(1.0,0,1.0);
glTranslatef(-0.3f,0.45f,0.3f);
draw_cylinder(0.047,0.047,0.04f);
glPopMatrix();

glPushMatrix();
glColor3f(4,4,4);
glTranslatef(-0.3f,0.50f,0.3f);
draw_hemisphere(0.046,0.1f);
```

```
        glPopMatrix();


        glPushMatrix();
        glColor3f(0,0.5,0);
        glTranslatef(0.3f,0.35f,-0.3f);
        draw_cylinder(0.045,0.045,0.14f);
        glPopMatrix();

        glPushMatrix();
        glColor3f(1.0,0,1.0);
        glTranslatef(0.3f,0.45f,-0.3f);
        draw_cylinder(0.047,0.047,0.04f);
        glPopMatrix();

        glPushMatrix();
        glColor3f(4,4,4);
        glTranslatef(0.3f,0.50f,-0.3f);
        draw_hemisphere(0.046,0.1f);

        glPopMatrix();



        glPushMatrix();
        glColor3f(0,0.5,0);
        glTranslatef(-0.3f,0.35f,-0.3f);
        draw_cylinder(0.045,0.045,0.14f);
        glPopMatrix();

        glPushMatrix();
        glColor3f(1.0,0,1.0);
        glTranslatef(-0.3f,0.45f,-0.3f);
        draw_cylinder(0.047,0.047,0.04f);
        glPopMatrix();

        glPushMatrix();
        glColor3f(4,4,4);
        glTranslatef(-0.3f,0.50f,-0.3f);
        draw_hemisphere(0.046,0.1f);

        glPopMatrix();

         glPushMatrix();
        glDisable(GL_LIGHTING);
        glDisable(GL_DEPTH);
        glDisable(GL_LIGHTING);
glDisable(GL_LIGHT0);
   glDisable(GL_LIGHT1);
  glDisable(GL_LIGHT2);
        draw_door_z(1);
        glPopMatrix();

          glPushMatrix();
        glDisable(GL_LIGHTING);
        glDisable(GL_DEPTH);
        draw_door_z(-1);
        glPopMatrix();

         glPushMatrix();
        glDisable(GL_LIGHTING);
        glDisable(GL_DEPTH);
        draw_door_y(1);
        glPopMatrix();

          glPushMatrix();
        glDisable(GL_LIGHTING);
        glDisable(GL_DEPTH);
        draw_door_y(-1);
        glPopMatrix();



}





void display()
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glColor4f(0.0,0.0,1.0,1.0);
        glLoadIdentity();
         gluLookAt( x, 0.0, z,
              x + lx, 0.0, z + lz,
              0.0, 1.0, 0.0);
           //  gluLookAt(0.0f, 0.0f,5.0f , 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f);
          //  glRotatef(_angle,0.0f,1.0f,0.0f);
      //  drawBase();

        glPushMatrix();
         //  glTranslatef(0,0.2f,0);
        //    glColor3f(1.0,0.0,0.0);
         //   draw_cylinder();
      //   draw_gon();
     //    draw_cyl()
     //;

     draw();
    // draw_hemisphere();
      glPopMatrix();


           glFlush();
        glutSwapBuffers();


}


void initRendering() {
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_COLOR_MATERIAL);
    glClearDepth(1.0f);
   glEnable(GL_LIGHTING);
 glEnable(GL_LIGHT0);
    glEnable(GL_LIGHT1);
   glEnable(GL_LIGHT2);
    glEnable(GL_NORMALIZE);
    glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
    GLfloat ambientColor[] = {0.2f, 0.2f, 0.2f, 1.0f}; //Color (0.2, 0.2, 0.2)
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);

    //Add positioned light
    GLfloat lightColor0[] = {0.5f, 0.5f, 0.5f, 1.0f}; //Color (0.5, 0.5, 0.5)
    GLfloat lightPos0[] = {3.0f, 0.0f, 0.3f, 1.0f}; //Positioned at (300, 0,
100)
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);

    GLfloat lightColor1[] = {0.5f, 0.5f, 0.5f, 1.0f}; //Color (0.5, 0.5, 0.5)
    GLfloat lightPos1[] = {-5.0f, 0.0f, 8.0f, 1.0f}; //Positioned at (300, 0,
100)
    glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
    glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);

    GLfloat lightColor2[] = {0.5f, 0.5f, 0.5f, 1.0f}; //Color (0.5, 0.5, 0.5)
    GLfloat lightPos2[] = {3.0f, 0.0f, 0.5f, 1.0f}; //Positioned at (300, 0,
100)
    glLightfv(GL_LIGHT2, GL_DIFFUSE, lightColor2);
    glLightfv(GL_LIGHT2, GL_POSITION, lightPos2);
}


float frc=0.5f;
void specialkeyboard(int key, int xx, int yy){
    float frc = 0.1;
    switch(key){
        case(GLUT_KEY_LEFT):
            _angle -= 0.05;
            lx = sin(_angle);
            lz = -cos(_angle);
            break;

        case(GLUT_KEY_RIGHT):
            _angle += 0.05;
            lx = sin(_angle);
            lz = -cos(_angle);
            break;

        case(GLUT_KEY_UP):
            x += lx*frc;
            z += lz*frc;
            break;

        case(GLUT_KEY_DOWN):
            x -= lx*frc;
            z -= lz*frc;
            break;
        default: break;
    }
    glutPostRedisplay();
}

void handleResize(int w, int h) {
   // glViewport(0, 0, w, h);
   double w2h = (h>0) ? (double)w/h:1;
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
   // glOrtho(-dim*w2h,+dim*w2h, -dim,+dim, -dim,+dim);
    glLoadIdentity();
    gluPerspective(45.0f, (double)w / (double)h, 1.0f, 200.0f);
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
//    glMatrixMode(GL_PROJECTION);
//    glLoadIdentity();
//    glOrtho( -500, 500, -500, 500,-500 , 500);
    //glDepthFunc(GL_LEQUAL);    // Set the type of depth-test
    //    glShadeModel(GL_SMOOTH)



   // gluPerspective(45.0, 2, 1.0, 800.0);
}

void windowMenu(int value)
{
  specialkeyboard((unsigned char)value, 0, 0);
}

int main(int argc, char** argv)
{

               glutInit(&argc,argv);
               glutInitDisplayMode(GLUT_RGBA|GLUT_DOUBLE|GLUT_DEPTH);
               glutInitWindowSize(600,600);
               glutCreateWindow("Taj Mahal");
               initRendering();
               glutDisplayFunc(display);
               glutReshapeFunc(handleResize);

    glutSpecialFunc(specialkeyboard);
    glutCreateMenu(windowMenu);
    glutAddMenuEntry("INcrease angle ",GLUT_KEY_LEFT);
     glutAddMenuEntry("Decrease angle ",GLUT_KEY_RIGHT);
      glutAttachMenu(GLUT_RIGHT_BUTTON);


               glutMainLoop();
               return 0;
}
```

CHAPTER 6

SNAPSHOTS

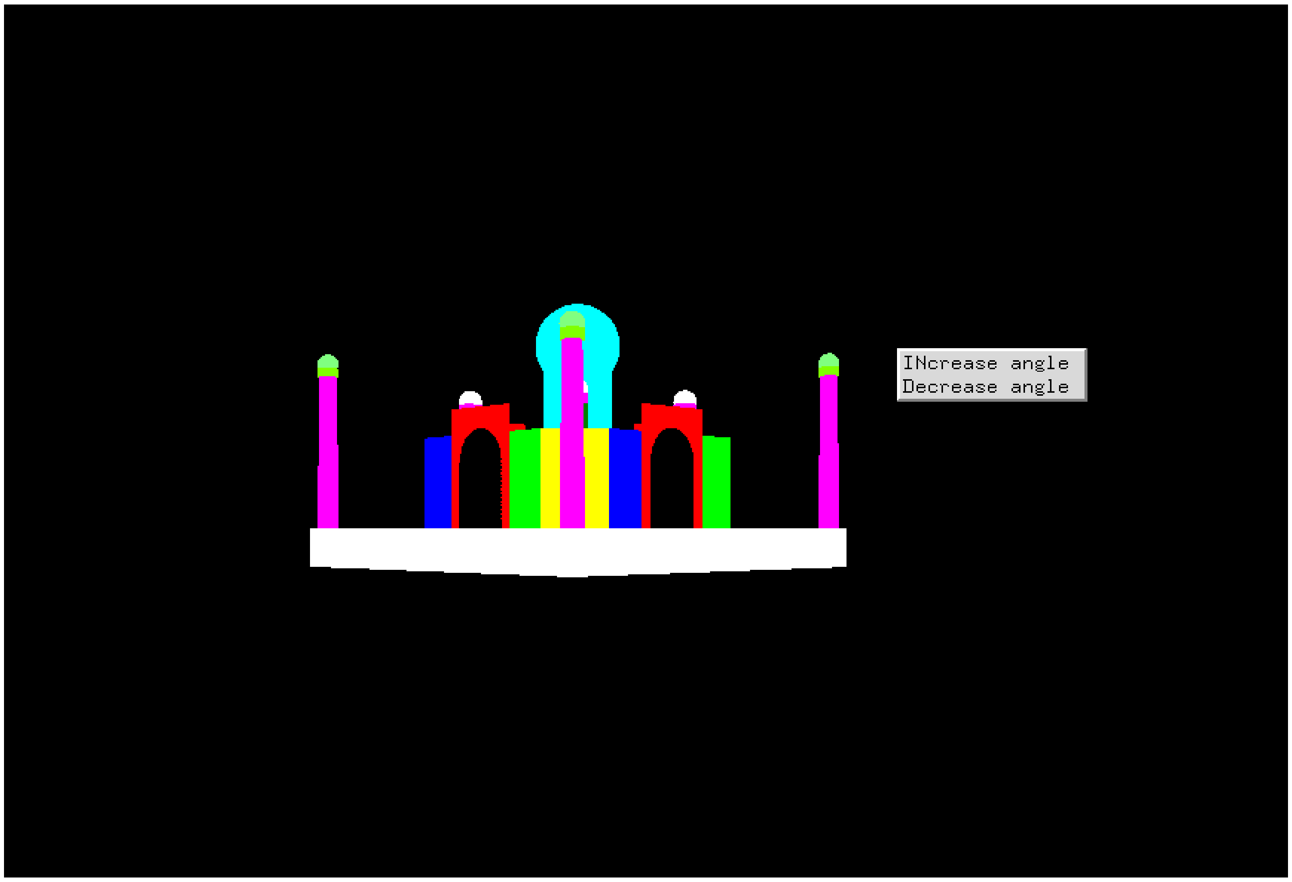

Fig. 6.1

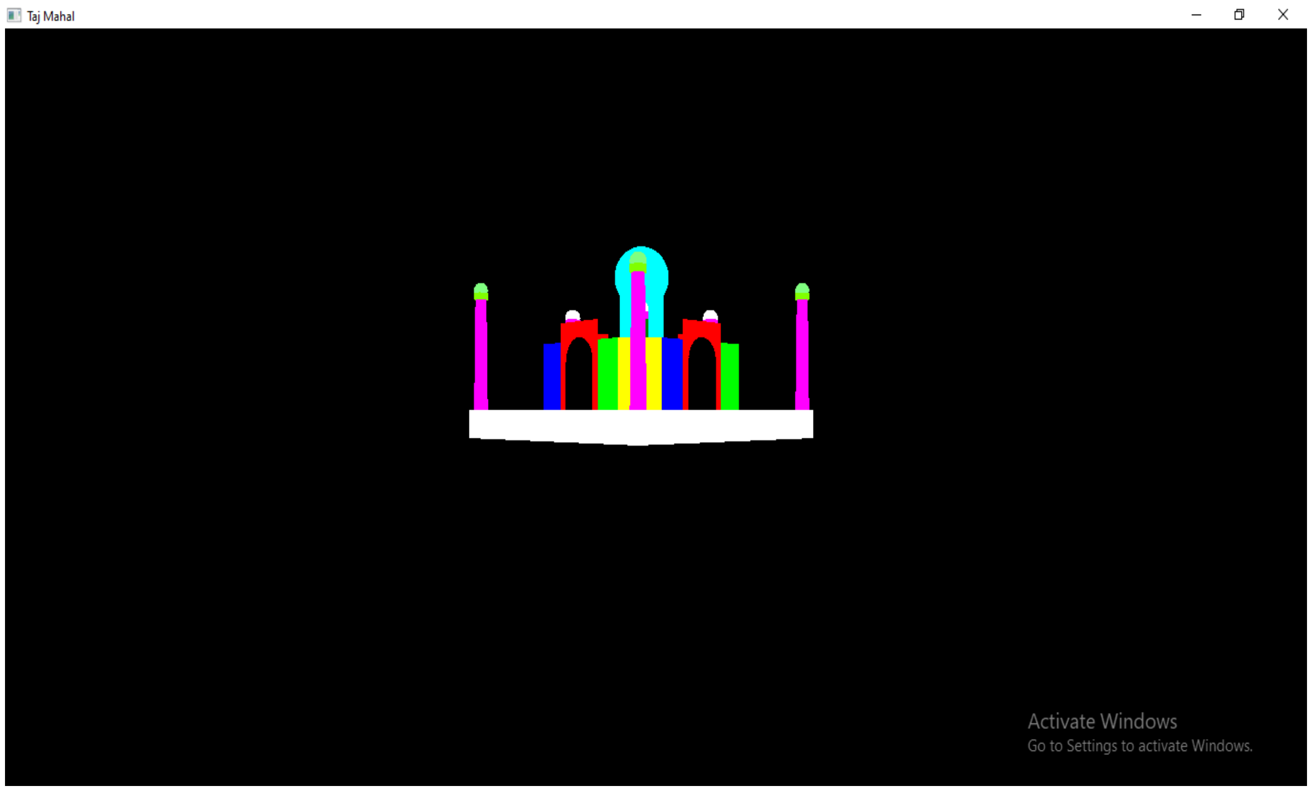
Taj Mahal
Activate Windows
Go to Settings to activate Windows.

Fig. 6.2

CHAPTER 7

CONCLUSION

The code we have implemented for our project is working well to the best of our knowledge.

Throughout the development process, various OpenGL features and techniques were utilized to achieve realistic rendering and visually captivating results.

The project's primary objective was to enable users to explore and interact with the digital rendition of the Taj Mahal. By implementing user controls, such as navigation and camera manipulation, visitors can freely roam the virtual environment, admire the intricate architectural details, and gain a deeper appreciation for this architectural masterpiece.

The project has demonstrated the capabilities of OpenGL as a powerful tool for computer graphics and virtual reality applications. Through careful design and meticulous attention to detail, the developers have created a visually stunning and immersive simulation that transports users to the magnificence of the Taj Mahal.

While the project has achieved its goals in creating a captivating simulation, there are always opportunities for further enhancements. Future iterations could include additional interactivity, such as allowing users to manipulate specific elements or learn about the construction techniques used in building the Taj Mahal. Furthermore, optimizations could be made to improve performance and extend the project's compatibility to a broader range of devices.

In conclusion, the Taj Mahal Simulation project serves as an impressive example of the potential of computer graphics and OpenGL in recreating historical landmarks in a virtual environment. By offering an immersive and educational experience, it allows users to appreciate the architectural marvel of the Taj Mahal from anywhere in the world.

REFERENCES

1. **Edward Angel**: Interactive Computer Graphics: A Top-Down Approach with 5th Edition, Addison-wesley,2008.

2. James D.foley, Andries Van Dam, Steven K.Feiner, John F.Hughes, **“Computer Graphics Principle & Practice**” Published by Pearson Education Pte.ltd, 2nd Edition.

3. Yashavanth Kanetkar,” **Graphics under opengl**”, Published by BPB Publications.

4. Athul P.Godse, **“Computer Graphics”**, Technical Publications Pune, 2nd Revised Edition.