VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Machhe, Belagavi 590018

Computer Graphics Mini Project Report on

“OPENGL ARTS”

Submitted in partial fulfilment of the requirements for the award of the Degree of

Bachelor of Engineering in Computer Science and Engineering

For the academic year

2021-2022

Submitted by

DEVARSHI KOTHARI 1HK19CS045
DIVYANSH SWAMI 1HK19CS046

Under the Guidance of

PROF. TAHIR NAQUASH
Assistant Professor
Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
HKBK COLLEGE OF ENGINEERING
(Approved by AICTE & Affiliated to VTU, Belgaum)
22/1, Opp. Manyata Tech Park, Nagawara, Bangalore-560045
Email: info@hkbk.edu.in URL: www.hkbk.edu.in

HKBK COLLEGE OF ENGINEERING

22/1, Opp. Manyata Tech Park, Nagawara, Bangalore-560045

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled **OpenGL Arts** carried out by Mr. **Devarshi Kothari**, USN **1HK19CS045** and Mr. **Divyansh Swami**, USN **1HK19CS046**, bonafide students of **HKBK College of Engineering** in partial fulfilment for the award of Bachelor of Engineering / Bachelor of Technology in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2021-2022**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report 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.

Project Guide
Professor Tahir Naquash
Assistant Professor
CSE Dept. HKBKCE

HOD
Dr. Ashok Kumar
Department of CSE
HKBKCE

External Viva

Name of examiners

1.__________________

2.__________________

Signature with Date

____

____

HKBK COLLEGE OF ENGINEERING

22/1, Opp. Manyata Tech Park, Nagawara, Bangalore-560045

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We Devarshi Kothari and Divyansh Swami, students of sixth semester BE, **Computer Science & Engineering**, **HKBK College Of Engineering** hereby declare that the project work entitled **“OpenGL Arts“**has been carried out by us at **HKBK College Of Engineering** and submitted in partial fulfilment of the course requirements for the award of the degree of Bachelor of Engineering in **Computer Science and Engineering** of Visvesvaraya Technological University, Belgaum, during the academic year **2021-2022**.

We also declare that, to the best of our knowledge and belief, the work reported here does not from part of any other dissertation on the basis of which a degree or award was conferred on an earlier occasion on this by any other student.

Date:

Place: Nagawara, Bangalore

Devarshi Kothari 1hk19cs045

Divyansh Swami 1hk19cs045

ABSTRACT

OpenGL is a fundamental graphic library for all fields of display procedures, and at its core lies the ever elusive mathematic functions. The Matrix and geometric are what controls the pixels illuminating any images displayed.

This project titled "OpenGL Arts" strives to create great visuals with the help of algorithmic maths. On a Menu driven canvas, many form of arts like fractal, illusions and wire meshes were drew.

The project is undertaken with the vision to spread the beauty of maths and algorithms using computer generated gallery. One of the most important aspect of choosing such problem definition was to learn how to implement and manage a project.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany 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 efforts with success.

We would like to profoundly thank Chairman Mr. CM Faiz and Management of HKBK College of Engineering for providing such a healthy environment for the successful completion of Project Work.

We would like to express our thanks to the Principal Dr. Tabassum Ara, for her encouragement that motivated us for the successful completion of Project Work.

We wish to express our gratitude to Dr., Professor and Head of Department of Computer Science & Engineering for providing such a healthy environment for the successful completion of Project work.

We wish to express our thanks to our guide Prof. Tahir Naquash, Assistant Professor, Department of Computer Science & Engineering for her Expert Guidance, initiative and providing a good working environment and for her constant support and encouragement.

We would also like to thank all other teaching and technical staffs of Department of Computer Science and Engineering, who have directly or indirectly helped us in the completion of this Project Work.

We would like to reach out to all intellectual erudite from whom we referenced crucial elements of the project.

And lastly, we would hereby acknowledge and thank our parents who have been a source of inspiration and also instrumental in the successful completion of this project.

Devarshi Kothari	**1HK19CS045**
Divyansh Swami	**1HK19CS046**

TABLE OF CONTENT

TABLE OF FIGURE

Chapter 1 :

INTRODUCTION

1.1 DEFINITION

The project is an implementation of visual arts using OpenGL in C language. It generates creative drawings.

1.2 OVERVIEW

Although the scope of this project seem dismal, it is undertaken with great contemplation and insights. It act as a gateway to learning ever so elusive algorithms and applied mathematics, it strive to spread the exquisite nature of such by creating easy to analyse graphics.

1.3 PROBLEM DEFINITION

Using numerous library of C and OpenGL implementations, generate visually appeasing arts with an intention to represent mathematics such as fractal.

1.4 PROBLEM EXPLANATION

“The expression or application of human creative skill and imagination, typically in a visual form such as painting or sculpture, producing works to be appreciated primarily for their beauty or emotional power.” is what oxford define art as. In this instance it’s in the form of computer graphics and meant to be appreciated for its intricacy.

Fractals: any of various extremely irregular curves or shapes for which any suitably chosen part is similar in shape to a given larger or smaller part when magnified or reduced to the same size.

Chapter 2 :

SYSTEM DESIGN

2.1 DESIGN CONSIDERATION

C is a computer language known for its resource management efficiency. It's one of the most well used language and is nearer to assembly language than most of its competition.

With the intent to create hundreds of arts, a Modular Programming Approach was undertaken to implement scalability. A main module was created to act as the canvas which can be used to access other modules that defined different drawings.

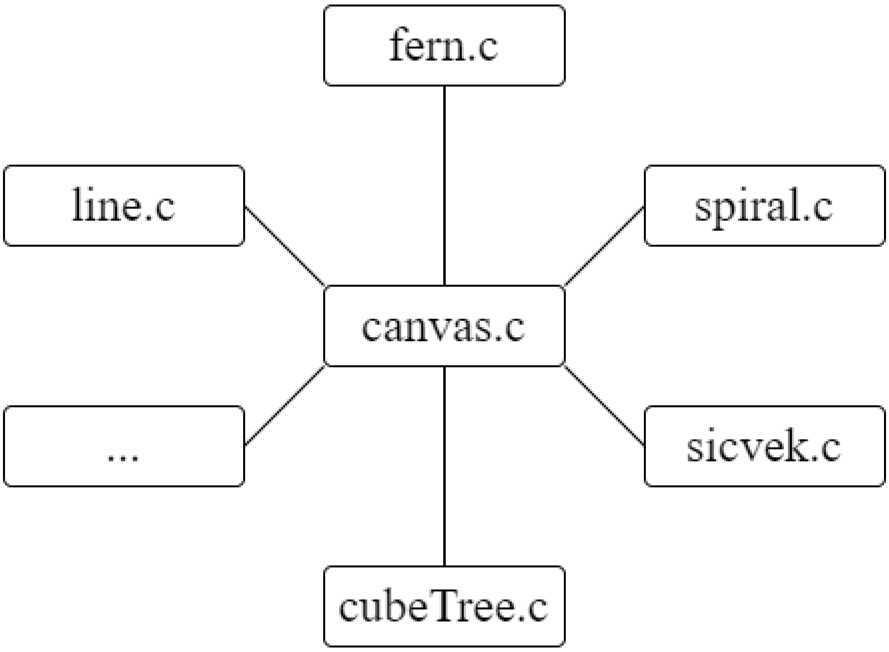
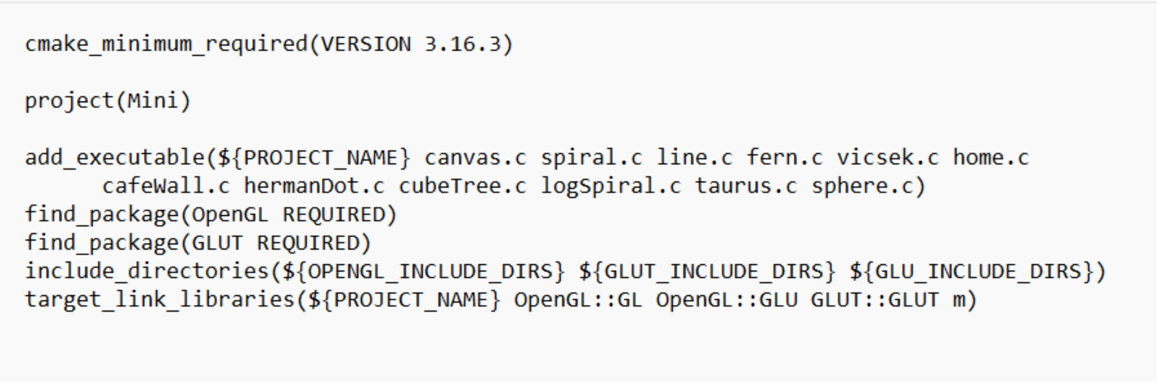


Figure 2-1 Modular Design

2.2 DESIGN ARCHITECTURE

All modules are defined in the project header project.h. Then all modules are connected as executable using cmake during execution.

"*CMake is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice. The suite of CMake tools were created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK and VTK.*"



```
cmake_minimum_required(VERSION 3.16.3)

project(Mini)

add_executable(${PROJECT_NAME} canvas.c spiral.c line.c fern.c vicsek.c home.c
      cafeWall.c hermanDot.c cubeTree.c logSpiral.c taurus.c sphere.c)
find_package(OpenGL REQUIRED)
find_package(GLUT REQUIRED)
include_directories(${OPENGL_INCLUDE_DIRS} ${GLUT_INCLUDE_DIRS} ${GLU_INCLUDE_DIRS})
target_link_libraries(${PROJECT_NAME} OpenGL::GL OpenGL::GLU GLUT::GLUT m)
```

Figure 2-2 Cmake architecture

Chapter 3 :

SYSTEM REQUIREMENT SPECIFICATION

A System Requirements Specification (SRS) (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behaviour of a system or software application.

3.1 FUNCTIONAL REQUIREMENT

- It should be menu driven
- There should be a front page

3.2 NON FUNCTIONAL REQUIREMENT

- Each arts should be modular
- It window should be responsive
- It should be efficient in handling complex functions
- It should be portable and small in size

3.3 SYSTEM REQUIREMENT

3.3.1 Software Requirement

- A Linux powered system
- Glut library functions
- Cmake installed
- Math library

3.3.2 Hardware Requirement

- Processor Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz 2.50 GHz
- Installed RAM8.00 GB (7.83 GB usable)
- System type 64-bit operating system, x64-based processor

Chapter 4 :

IMPLEMENTATION

4.1 CANVAS.C

```
#include "project.h"
#define X 500
#define TOTAL 4
int current=0;
void myinit()
{
glClearColor(0,0,0,1.0);
glColor3f(1.0,0.0,0.0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0,499.0,0.0,499.0,-2000,2000);
}
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        switch(current)
        {
                case 0: home();                 break;
                case 1: line();                 break;
                case 2: spiral();               break;
                case 3: fern();                 break;
                case 4: vicsek();               break;
                case 5: cafeWall();             break;
                case 6: hermanDot();            break;
                case 7: cubeTree();             break;
                case 8: logSpiral();            break;
                case 9: taurus();               break;
                case 10: sphere();              break;
                case 11: exit(0);               break;
                default:                        break;
        }
        glutSwapBuffers();
}
void mymenu(int id)
{
        current=id;
        glutPostRedisplay();
}
void myReshape(int w,int h)
{
        glViewport(0,0,w,h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if(w<=h)
                glOrtho(-X,X,-X*(GLfloat)h/(GLfloat)w,X*(GLfloat)h/(GLfloat)w,-
10.0,10.0);
        else
        glOrtho(-X*(GLfloat)w/(GLfloat)h,X*(GLfloat)w/(GLfloat)h,-X,X,-10.0,10.0);
        glMatrixMode(GL_MODELVIEW);
}
int main(int argc,char** argv)
{
glutInit(&argc,argv);
myinit();
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(500,500);
glutCreateWindow("Arts Using OpenGL");
glutCreateMenu(mymenu);
glutAddMenuEntry("line",1);
glutAddMenuEntry("spiral",2);
glutAddMenuEntry("fern fractal",3);
glutAddMenuEntry("vicsek-5",4);
glutAddMenuEntry("CafeWall",5);
glutAddMenuEntry("hermann dot",6);
glutAddMenuEntry("cubeTree",7);
glutAddMenuEntry("Logarithmic Spiral",8);
glutAddMenuEntry("taurus",9);
glutAddMenuEntry("sphere",10);
glutAddMenuEntry("exit",11);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutReshapeFunc(myReshape);
glutDisplayFunc(display);
glEnable(GL_DEPTH_TEST);
glutMainLoop();
return 0;
}
```

4.2 HOME.C

```
#include "project.h"
void drawstring(float x, float y, const char *s)
{
	unsigned int i;
	glRasterPos2f(x,y);
	for(i=0;i<strlen(s);i++)
		glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,s[i]);
}
void drawstring1(float x, float y, const char *s)
{	unsigned int i;
	glRasterPos2f(x,y);
	for(i=0;i<strlen(s);i++)
		glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,s[i]);
}
void home()
{
	char vtu1[]={"VISVESVARAYA TECHNOLOGICAL "};
	char vtu2[]={"UNIVERSITY"},
			cg[]="Computer Graphics Mini Project on",
			title[]="Arts using OpenGL in C",
			auth1[]="Sonam Dorji		1HK19CS151",
			auth2[]="Sunil Nagar		1HK19CS158",
			guide1[]="Under the guidance of:",
			guide2[]="Prof. Pushpa T.",
			college[]="Cse dept. HKBKCE";
	glColor3f(1,0,0);
	int l=strlen(vtu1);
	drawstring(-(l/2)*29.94011976,300,vtu1);
	l=strlen(vtu2);
	drawstring(-(l/2)*29.94011976,250,vtu2);
	l=strlen(cg);
	glColor3f(0,0,1);
	drawstring1(-(l/2)*18.94011976,150,cg);
	l=strlen(title);
	drawstring1(-(l/2)*18.94011976,100,title);
	l=strlen(auth1);
	glColor3f(1,1,1);
	drawstring1(-(l/2)*18.94011976,0,auth1);
	l=strlen(auth2);
	drawstring1(-(l/2)*18.94011976,-50,auth2);
	l=strlen(guide1);
	glColor3f(1,1,0);
	drawstring1(-(l/2)*18.94011976,-150,guide1);
	l=strlen(guide2);
	drawstring1(-(l/2)*18.94011976,-200,guide2);
	l=strlen(college);
	drawstring1(-(l/2)*20.94011976,-250,college)
}
```

4.3 LINE.C

```
#include "project.h"
void line()
{
        int x=100;
        glBegin(GL_LINES);
        for(int i=0;i<100;i++)
        {
                int x1=rand()%500,y1=rand()%500,x2=rand()%500,y2=rand()%500;
                glColor3f((float)x1/500,(float)y1/500,(float)x2/500);
                glVertex2d(x1,y1);
                glVertex2d(x2,y2);
        }
        for(int i=0;i<100;i++)
        {
                int x1=rand()%500,y1=rand()%500,x2=rand()%500,y2=rand()%500;
                glColor3f((float)x1/500,(float)y1/500,(float)x2/500);
                glVertex2d(-x1,-y1);
                glVertex2d(-x2,-y2);
        }
        for(int i=0;i<100;i++)
        {
                int x1=rand()%500,y1=rand()%500,x2=rand()%500,y2=rand()%500;
                glColor3f((float)x1/500,(float)y1/500,(float)x2/500);
                glVertex2d(-x1,y1);
                glVertex2d(-x2,y2);
        }
        for(int i=0;i<100;i++)
        {       int x1=rand()%500,y1=rand()%500,x2=rand()%500,y2=rand()%500;
                glColor3f((float)x1/500,(float)y1/500,(float)x2/500);
                glVertex2d(x1,-y1);
                glVertex2d(x2,-y2);
        }
        glEnd();
}
```

4.4 TAURUS.C

```
#include "project.h"
void taurus() {
  glClear(GL_COLOR_BUFFER_BIT);

  glColor3f(1.0, 1.0, 1.0);
  glScalef(100,100,0);
  int x=rand()%360;
  glRotatef(x,1,1,0);
  glutWireTorus(2,3,15, 30);
}
```

4.5 SPIRAL.C

```
#include "project.h"
void spiral()
{

        float radius = 1.0f,x,y;
        int a=rand()%500,b=rand()%500,c=rand()%500;
        glBegin(GL_POINTS);
        glColor3f((float)a/500,(float)b/500,(float)c/500);
        for (float angle = 0; angle < 14400; angle += 1)
        {
        x = cos(angle * M_PI / 180) * radius;
        y = sin(angle * M_PI / 180) * radius;
        radius += 0.1f;
        glVertex2f(x, y);
        }
        glEnd();
}
```

4.6 VICSEK.C

```
#include "project.h"

void vicsek()
{
        typedef GLfloat point2[3];
        int points=100000;
   point2 vertices[6]={{0,900},{800,450},{800,-450},{0,-900},{-800,-450},{-800,450}};
   int j, k;
   int rand();
   point2 p ={0.0,50.0};
   glClear(GL_COLOR_BUFFER_BIT);

glColor3f((float)(rand()%1000)/1000,(float)(rand()%1000)/1000,(float)(rand()%1000)/10
00);
   for( k=0; k<points; k++)
   {
        j=rand()%6;

        p[0] = (p[0]+vertices[j][0])*(1.0/3.0);
        p[1] = (p[1]+vertices[j][1])*(1.0/3.0);

        glBegin(GL_POINTS);
           glVertex2fv(p);
        glEnd();
   }
}
```

4.7 FERN.C

```
#include "project.h"

int points=10000;

void fern()
{
          typedef GLfloat point2[2];
          point2 p = {0,0},newPoint;
          double probability[3] = {85, 92, 99};

          glClear ( GL_COLOR_BUFFER_BIT );
          glColor3f(0,1,0);

   for(int i=0; i<points; i++)
          {
            GLfloat prevx=p[0];
            GLfloat randnum=rand()%100 +1;
            if (randnum<probability[0])
      {
          p[0]=p[0]*0.85+0.04*p[1];
          p[1]=prevx*(-0.04)+0.85*p[1]+1.6;
            }
            else if(randnum<probability[1])
            {
          p[0]=0.2*p[0]-0.26*p[1];
          p[1]=0.23*prevx+0.22*p[1]+1.6;
            }
            else if(randnum<probability[2])
            {
          p[0]=-0.15*p[0]+0.28*p[1];
          p[1]=0.26*prevx+0.24*p[1]+0.44;
            }
            else
            {
          p[0]=0.0;
          p[1]=0.16*p[1];
            }
            newPoint[0]=p[0]*50;
            newPoint[1]=p[1]*50-250;
            if(i>100)
            {
                     glBegin(GL_POINTS);
            glVertex2fv(newPoint);
        glEnd();
            }
          }
          glColor3f(0,0,1);
}
```

4.8 CAFEWALL.C

```
#include "project.h"
void squares(int height, int rShift, int j){

      if(j % 2 == 1)
         glColor3f(0.0, 0.0, 0.0);
      else
         glColor3f(1.0, 1.0, 1.0);

      glBegin(GL_QUADS);
         glVertex2i(rShift, height);
         glVertex2i(rShift, height+100);
         glVertex2i(rShift+100, height+100);
         glVertex2i(rShift+100, height);
   }
void cafeWall(void)
   {
      glClear(GL_COLOR_BUFFER_BIT);
      glLineWidth(2);

      int rShift = -500;
      int height = -500;

      for(int i=0; i < 10; i++){
         for(int j=0; j < 11; j++){
            squares(height, rShift, j);
            rShift += 100;
         }

         switch(i){
            case 0:       rShift = -470;  break;
            case 1:       rShift = -440;  break;
            case 2:       rShift = -470;  break;
            case 4:       rShift = -470;  break;
            case 5:       rShift = -440;  break;
            case 6:       rShift = -470;  break;
            case 7:       rShift = -440;  break;
            default:      rShift = -500;         }
         height += 105;
      }
      glEnd();
   }
```

4.9 HERMANDOT.C

```
#include "project.h"
void squares(int height, int rShift);

void hermanDot(void)
   {
      glClear(GL_COLOR_BUFFER_BIT);
      glLineWidth(2);


      int rShift = -500;
      int height = -500;
      for(int i=0; i < 10; i++){
         for(int j=0; j < 10; j++){
            squares(height, rShift);
            rShift += 120;
         }
         rShift = -500;
         height = height + 120;
      }
      glEnd();
   }
```

4.10 LOGSPIRAL.C

```
#include "project.h"
void logSpiral()
{
         float radius = 1.0f,x,y;
         int a=rand()%500,b=rand()%500,c=rand()%500;
         glBegin(GL_POINTS);
         glColor3f((float)a/500,(float)b/500,(float)c/500);
         for (float angle = 0; angle < 14400; angle += 1)
         {
         x = cos(angle)*radius;
         y = sin(angle)*radius;
         radius += 0.1f;
         glVertex2f(x, y);
         }
         glEnd();
}
```

4.11 CUBETREE.C

```
#include "project.h"
void drawtree(int n)
{
   if(n>0)
   {
    glPushMatrix();
    glTranslatef(-0.5,1,0);
    glRotatef(45, 0.0, 0.0, 1.0);
    glScalef(0.707,0.707,0.707);
     drawtree(n-1);
    glPopMatrix();
    glPushMatrix();
    glTranslatef(0.5,1,0);
    glRotatef(-45, 0.0, 0.0, 1.0);
    glScalef(0.707,0.707,0.707);
    drawtree(n-1);
    glPopMatrix();
    glutSolidCube(1);
   }
}

void cubeTree()
{
  int n=10;
  glTranslatef(0,-200,0);
  glColor3f(0, 1.0, 0.8);
  glScalef(100,100,100);
  drawtree(n);
}
```

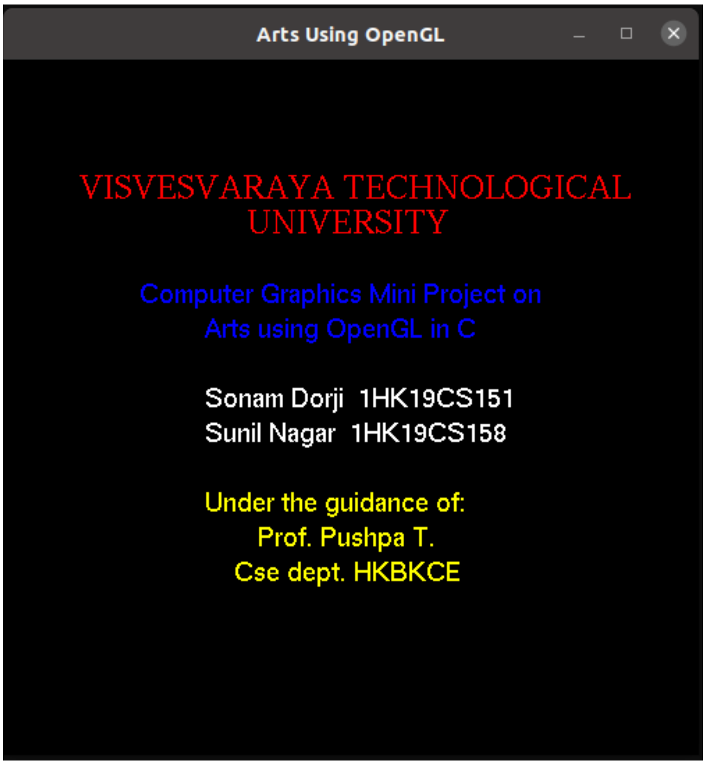
4.12 SPHERE.C

```
#include "project.h"
void sphere() {
  glClear(GL_COLOR_BUFFER_BIT);

  glColor3f(1.0, 1.0, 1.0);
  glScalef(100,100,0);
  int x=rand()%360,y=rand()%360,z=rand()%360;
  glRotatef(x,1,0,0);
  glRotatef(y,0,1,0);
  glRotatef(z,0,0,1);
  glutWireSphere(4, 20,20);
}
```

Chapter 5 :

OUTPUT



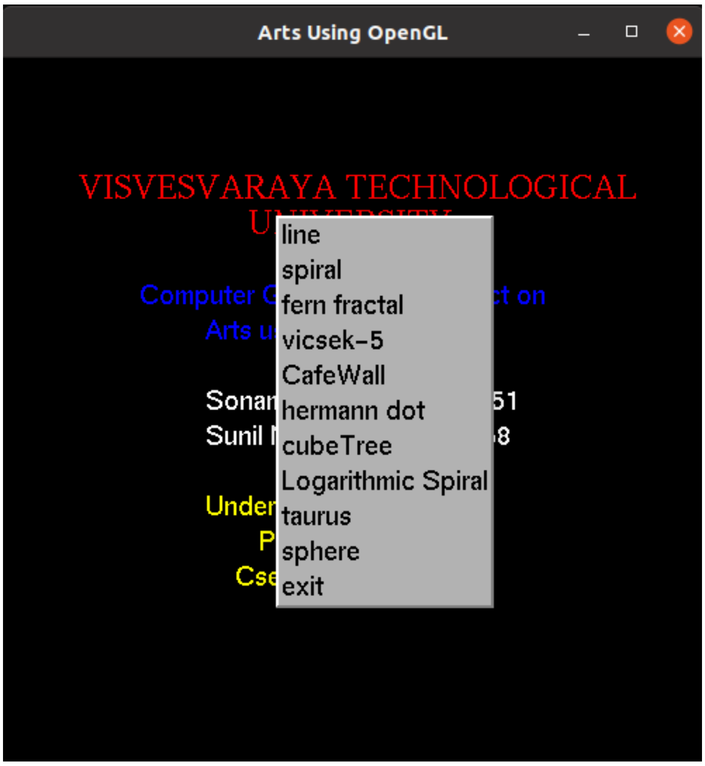
VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Computer Graphics Mini Project on
Arts using OpenGL in C

Sonam Dorji 1HK19CS151
Sunil Nagar 1HK19CS158

Under the guidance of:
Prof. Pushpa T.
Cse dept. HKBKCE

Figure 5-1 Home Display



VISVESVARAYA TECHNOLOGICAL
U

Computer C ct on
Arts u

Sonar 51
Sunil I 8

Under
P
Cse

Figure 5-2 Menu Display

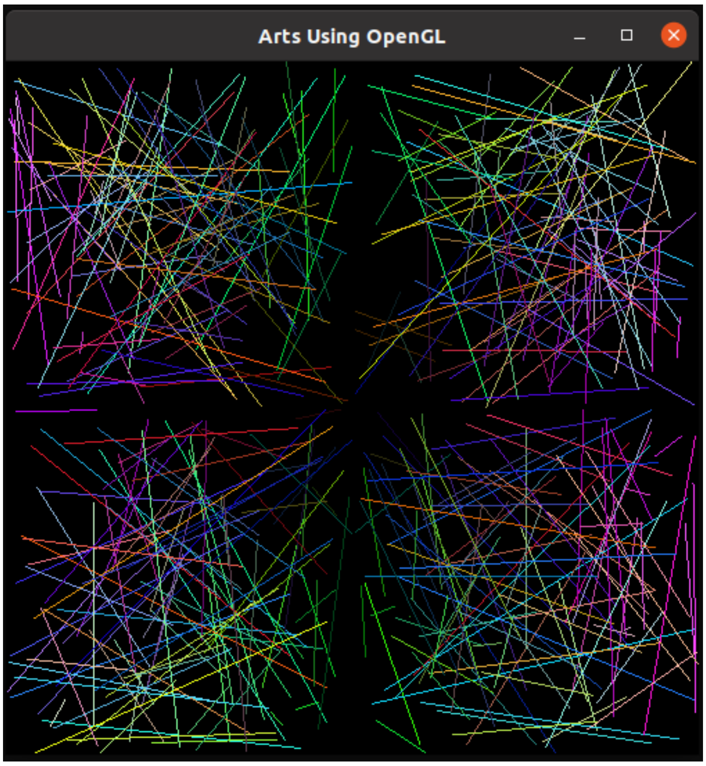


Figure 5-3 Line Art

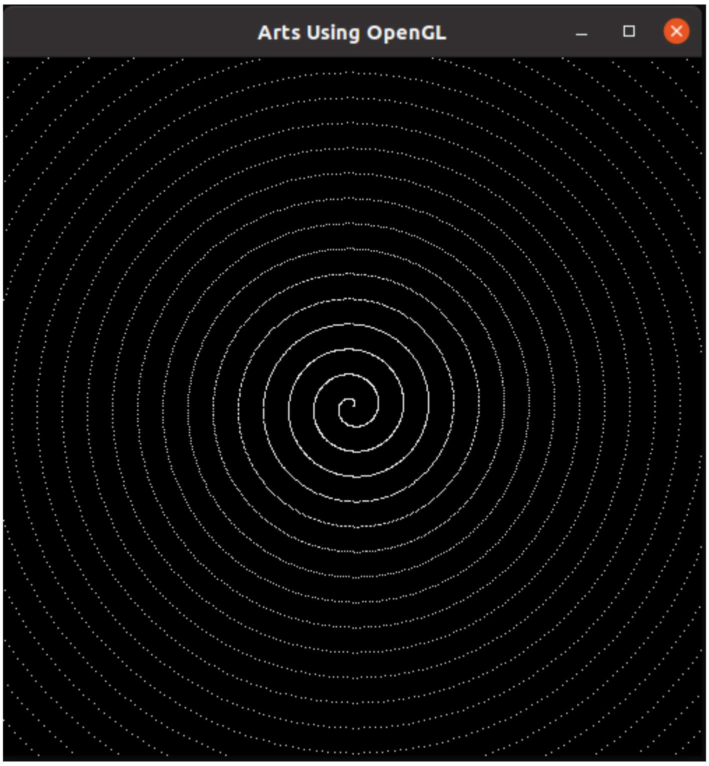


Figure 5-4 Spiral Art

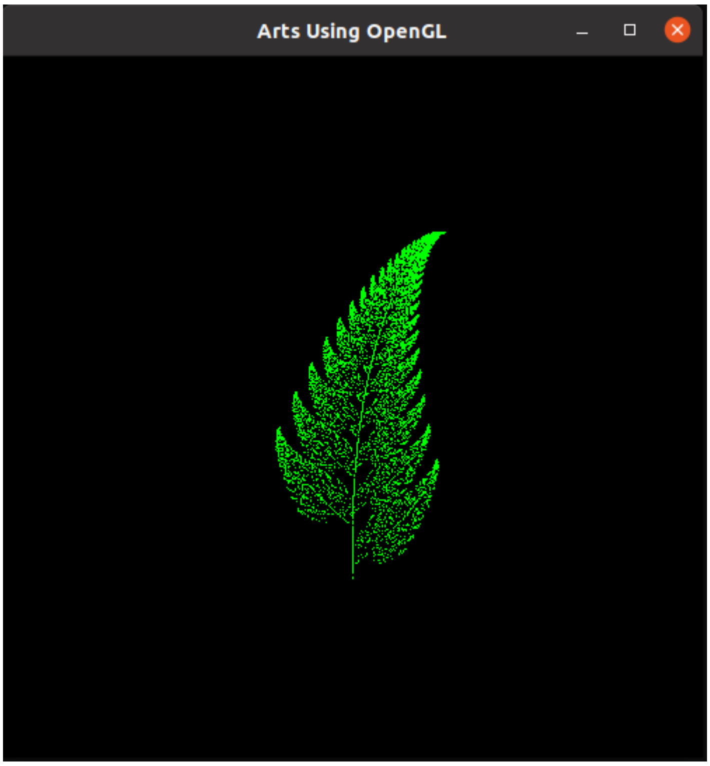


Figure 5-5 Fern Fractal

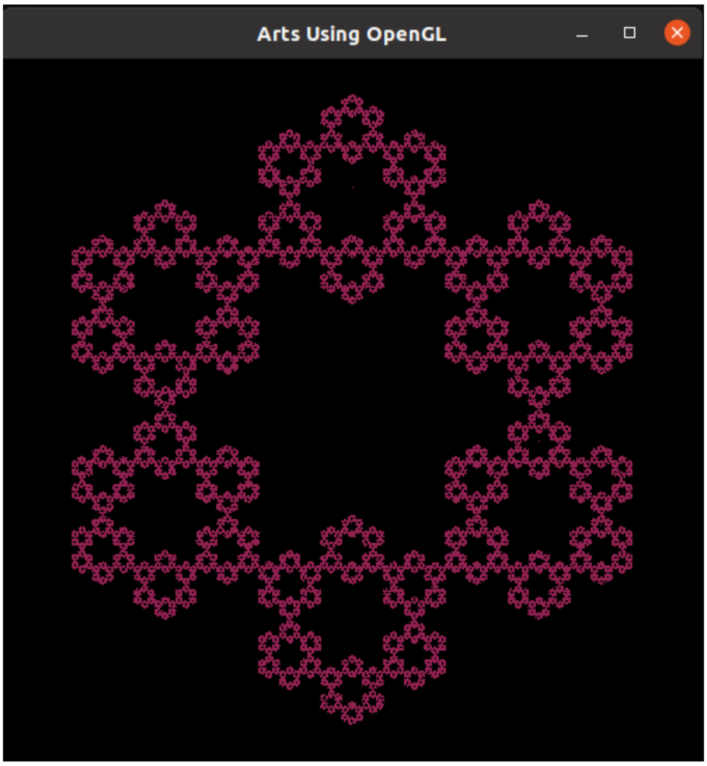


Figure 5-6 Vicsek Fractal

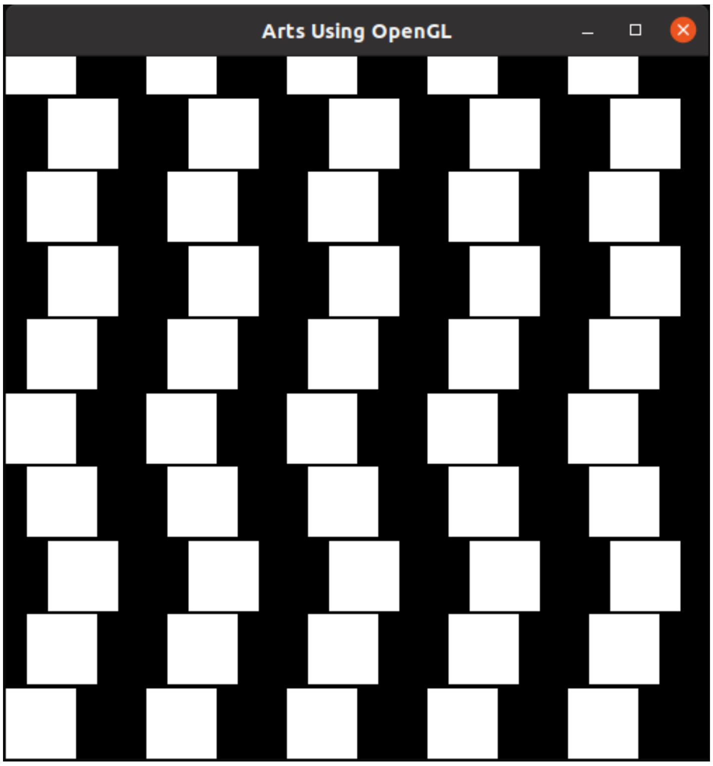


Figure 5-7 Cafe Wall Illusion

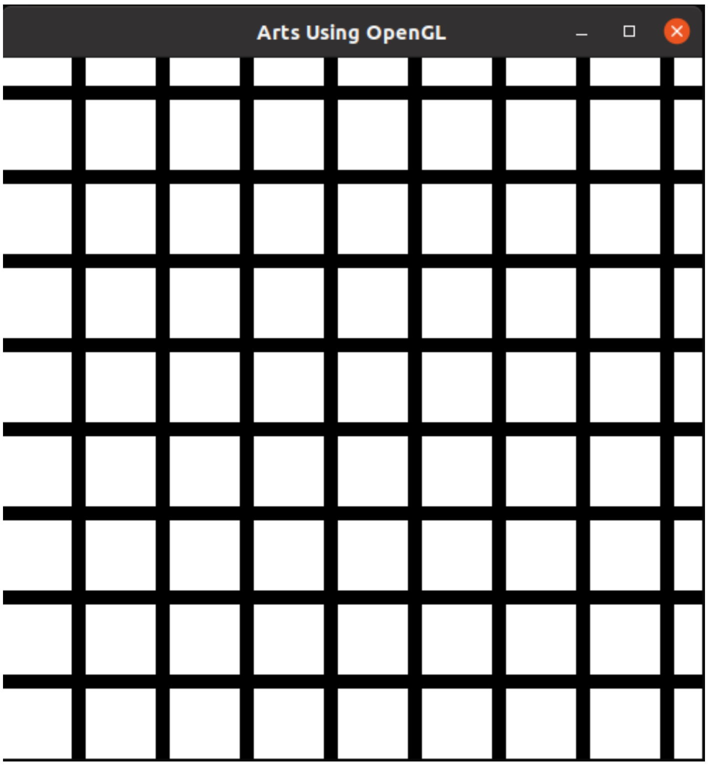


Figure 5-8 Hermann Illusion

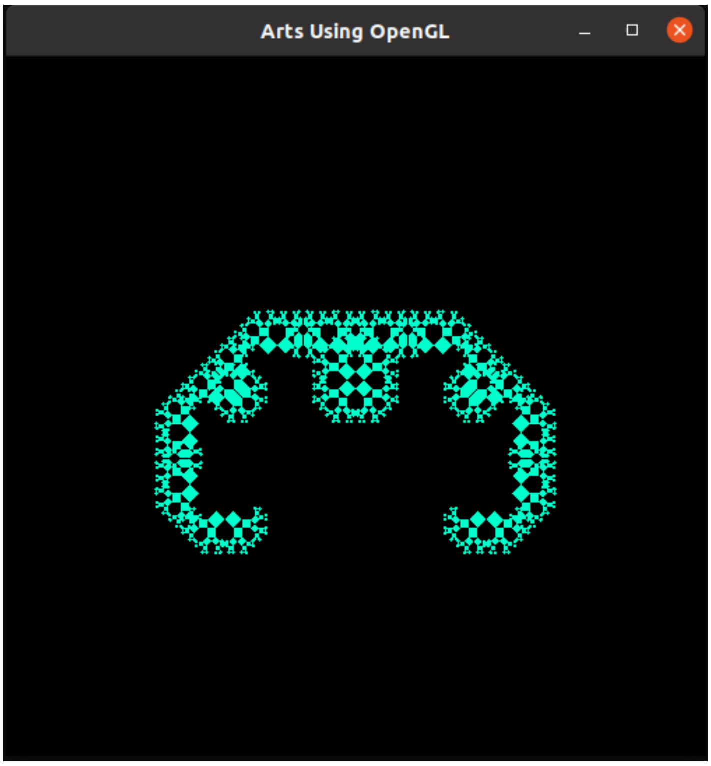


Figure 5-9 Cube Tree Fractal

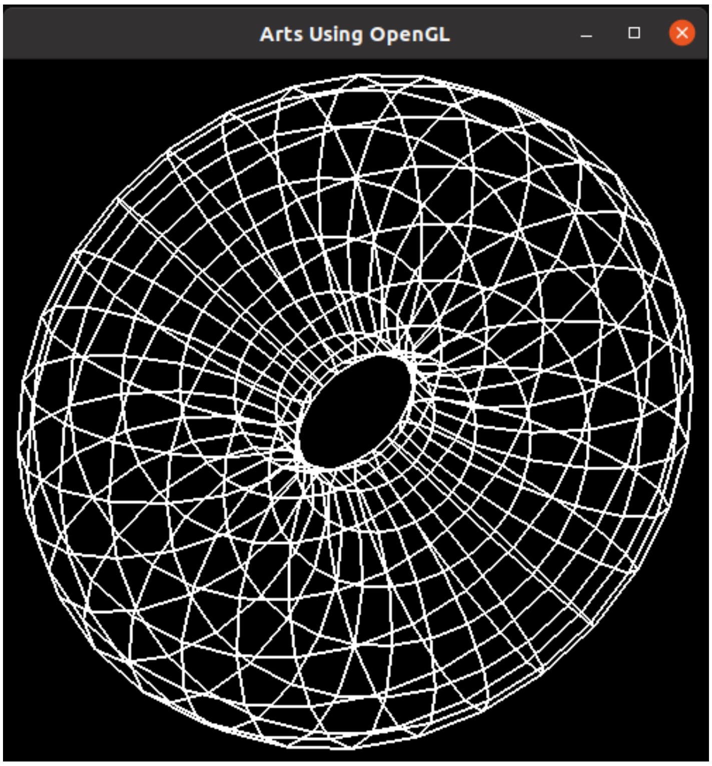


Figure 5-10 Torus Wireframe Art

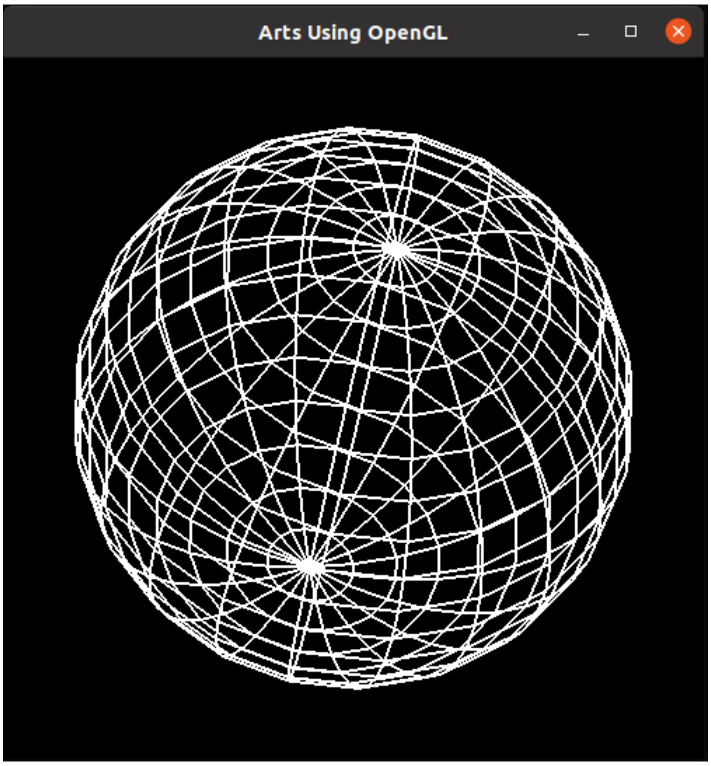


Figure 5-11 Spherical Wireframe Art

CONCLUSION

This project was an effort on spreading interest in Computer graphics and Mathematics. During the making of project, birth of interest in functions such as fractal were apparent. It led to increased proficiency in fundamentals of programming and project development stages.

The project has been born with rigorous body frame and attachable modules. Further upgrades and addition of arts can be easily made without restructuring the canvas. In the duration of the project, lessons in teamwork were many in number. Group discussions were held regularly with regular updates to the project. All such activity were reflected in the git repository of the group.

The Computer Graphic laboratory Mini project “OpenGL Arts” thus came to an end with the fruition of Linux executable application.

REFERENCES

[1] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd / 4th Edition, Pearson Education,2011

[2] Edward Angel: Interactive Computer Graphics- a Top Down approach with OpenGL, 5th edition. Pearson Education, 2008

[3] https://www.khronos.org/opengl/

[4] https://github.com/gogovlas/graphics-fractals.git

[5] https://github.com/UltimaJames/Computer_Graphics_Opengl.git

[6] https://mathworld.wolfram.com/LogarithmicSpiral.html

[7] https://cmake.org/cmake/help/latest/guide/tutorial/index.html