# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**JnanaSangama, Belgaum-590014**


**A CG mini project report**

**On**

## "3D GEARS AND TRANSFORMATIONS"

**Submitted in Partial fulfillment of the Requirements for the VI Semester of the Degree of**

**Bachelor of Engineering**

**In**

**Computer Science & Engineering**

**By**

**AKSHAY AMRUT MORAB(1CE17CS008)**

**BHUVANESHWARI M(1CE17CS024)**

**Under the Guidance of**

**Mr. Ramesh B**

**Asst. Professor, Dept. of CSE**


## CITY ENGINEERING COLLEGE

**Doddakallasandra, Kanakapura Road,**

**Bengaluru-560061**

# CITY ENGINEERING COLLEGE

**Doddakallasandra, Kanakapura Road, Bengaluru-560061**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## CERTIFICATE

Certified that the CG Mini Project work entitled **"3D GEARS AND TRANSFORMATIONS"** has been carried out by **AKSHAY AMRUT MORAB (1CE17CS008) and BHUVANESHWARI M (1CE17CS024),** bonafide students of City Engineering College in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveshvaraya Technological University, Belgaum during the year **2019-2020**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The CG Mini Project Report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

| | | |
|---|---|---|
| **Mr.Ramesh B** | **Mr. B Vivekavardhana Reddy** | **Dr. V. S Rama Murthy** |
| Asst.Prof, Dept.of CSE | Head, Dept. of CSE | Principal |

External Viva

| Name of the examiners | Signature with date |
|---|---|
| 1. | |
| 2. | |

# ACKNOWLEDGEMENT

While presenting this CG Mini Project on "**3D Gears And Transformations",** we feel that it is our duty to acknowledge the help rendered to us by various persons.

Firstly we thank God for showering his blessings on us. We are grateful to our institution City Engineering College for providing us a congenial atmosphere to carry out the project successfully.

We would like to express our heartfelt gratitude to **Dr. V S Ramamurthy,**
Principal, CEC, Bangalore, for extending his support**.**

We would also like to express our heartfelt gratitude to **Prof. Vivekavardhana Reddy**, HOD, Computer Science and Engineering whose guidance and support was truly invaluable**.**

We are very grateful to our guide, **Mr. Ramesh B**, Asst. Prof., Department of Computer Science, for his able guidance and valuable advice at every stage of our project which helped me in the successful completion of our project.

We would also have indebted to our Parent and Friends for their continued moral and material support throughout the course of project and helping me in finalize the presentation.

Our hearty thanks to all those have contributed bits, bytes and words to accomplish this Project.

**AKSHAY AMRUT MORAB(1CE17CS008)**

**BHUVANESHWARI M(1CE17CS024)**

# ABSTRACT

The project is on 3D Gears and transformation where gears are on the real view and the transformations are made as the given keys are pressed from the keyboard. It gives the movement of the given gear which rotates in one of the direction and the direction of rotation can be changed by the function rotate as used in the code for the transformation of the gear.

The mouse event is generated when left buttons button of the mouse is pressed for more options in the transformation is needed to be done in it, a menu is popped up on the screen. The special thing about this project is that options can be seen by pressing a keyboard key in the middle of output to see the options or the actions to be taken on the output i.e. the options will be displayed on the screen.

Here keyboard is used as an input device, the keyboard events are generated by the keyboard in the window and one of the operations is done by pressing mouse button to change the color of the gears while in operation. The glut function glutKeyboardFunc(key) is used for keyboard function and glutMouse(mouse) is used for the mouse function.

# CONTENTS

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION TO COMPUTER GRAPHICS

[2] Computer Graphics is concerned with all aspects of producing pictures or images using a computer. Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention
of photography and television.

**Applications of Computer Graphics**

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

**The Graphics Architecture**

Graphics Architecture can be made up of seven components:

1. Display processors
2. Pipeline architectures
3. The graphics pipeline
4. Vertex processing
5. Clipping and primitive assembly
6. Rasterization
7. Fragment processing

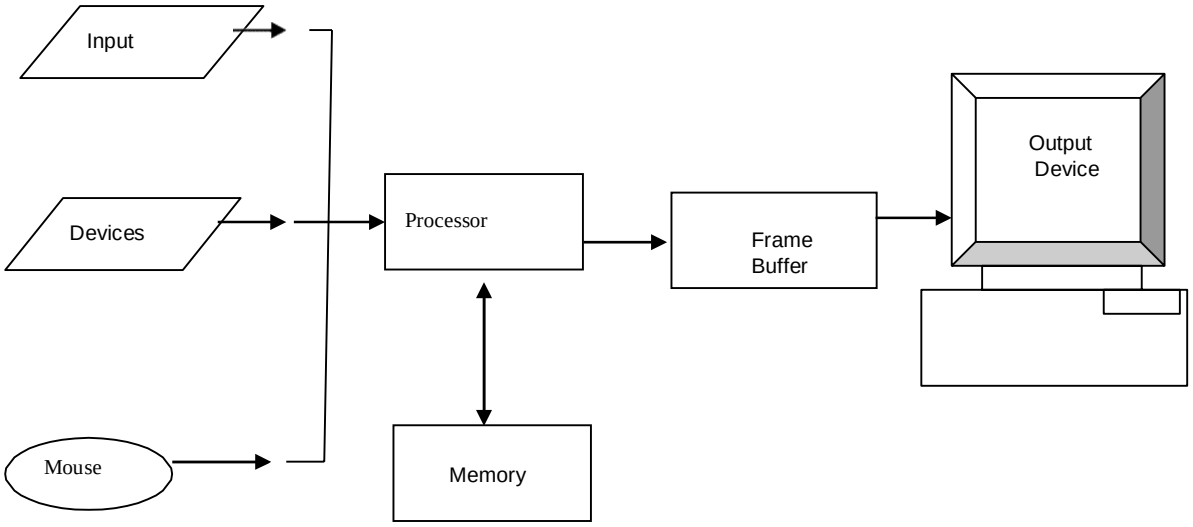


**Figure 1.1: Components of Graphics Architecture and their working**

## 1.2 INTRODUCTION TO OPENGL

[1]OpenGL is software used to implement computer graphics. The structure of OpenGL is similar to that of most modern APIs including Java 3D and DirectX. OpenGL is easy to learn, compared with other.

APIs are nevertheless powerful. It supports the simple 2D and 3D programs. It also supports the advanced rendering techniques. OpenGL API explains following 3 components

1. Graphics functions
2. Graphics pipeline and state machines
3. The OpenGL interfaces

There are so many polygon types in OpenGL like triangles, quadrilaterals, strips and fans. There are 2 control functions, which will explain OpenGL through,

1. Interaction with window system
2. Aspect ratio and viewports

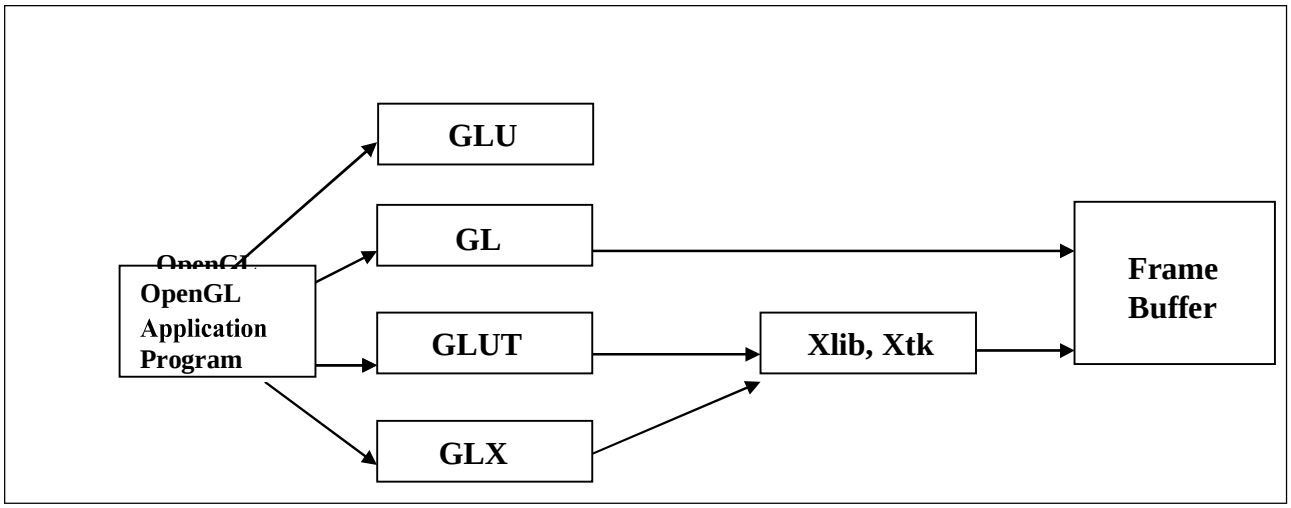


**Figure 1.2: OpenGL Library organization**

Most implementations of OpenGL have a similar order of operations, a series of processing stages called the OpenGL rendering pipeline. This ordering, as shown in Figure 1.2, isnot a strict rule of how OpenGL is implemented but provides a reliable guide for predicting what OpenGL will do. The following diagram shows the assembly line approach, which OpenGL takes to process data. Geometric data (vertices, lines, and polygons) follow the path through the row of boxes that includes evaluators and per-vertex operations, while pixel data (pixels, images, and bitmaps) are treated differently for part of the process. Both types of data undergo the same final steps before the final pixel data is written into the frame buffer.

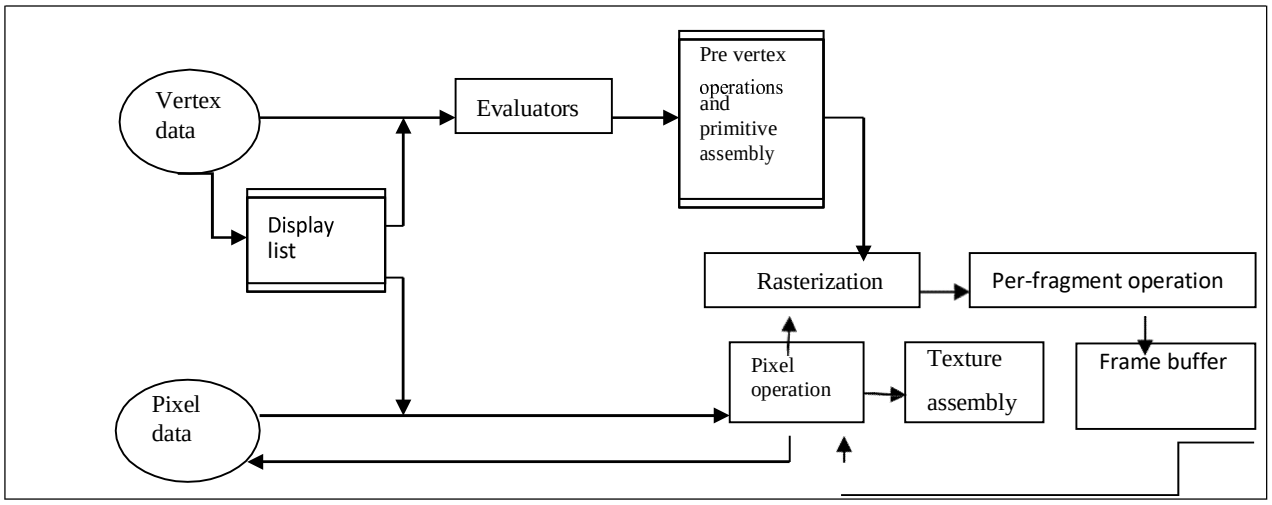


**Figure 1.3: OpenGL Order of Operations**

# CHAPTER 2

# REQUIREMENTS SPECIFICATION

## 2.1 SOFTWARE REQUIREMENTS

- Operating system – Windows10
- Microsoft Visual Studio2017
- Code Blocks
- OPENGL library files – GL, GLU,GLUT
- Language used is C/C++

## 2.2 HARDWARE REQUIREMENTS

- Processor – Pentium Pro
- Memory - 128MBRAM
- 20GB Hard Disk Drive
- Mouse or other pointing device
- Keyboard
- Display device

# CHAPTER 3

# SYSTEM DEFINITION

## 3.1 PROJECT DESCRIPTION

The working of the 3D gears is simulated by using OpenGL library functions. The linear motion of the gears and how the linear motion is converted into rotary motion is shown. The gear can be rotated in 3-dimensions which demonstrates the 3-dimensional rotation of an object about an axis. The shading effect is also demonstrated using some functions. The motion of the gears also demonstrates the use of animation in OpenGL. The translation and scaling of the gears is also seen using the functions.

## 3.2 FUNCTIONS USED:

[4] The user-defined functions implemented in this simulation are:

1. **static void gear(GLfloat inner_radius, GLfloat outer_radius, GLfloat width, GLint teeth, GLfloat tooth_depth)::** This function is used to draw gears . It takes parameters inner radius, outer radius, width ,teeth, tooth_depth.
2. **void text(char str[])::** This function used to display various texts in the screen. It takes string as parameter.
3. **void help()::** This function is used to create help menu for user who does not know anything about the project.
4. **Void Welcome()::** This function is used to produce welcome screen.
5. **Void wdraw(void)::** This function is used to perform rotation, scaling and translation operations recursively.
6. **Void demo()::** This function is used to produce the demo screen.
7. **Void draw1(void)::** This function is used to perform rotation, scaling and translation operations recursively for the 1st gear.
8. **Static void draw2(void)::** This function is used to perform rotation, scaling and translation operations recursively for the 2nd gear.
9. **Static void draw3(void)::** This function is used to perform rotation, scaling and translation operations recursively for the 3rd gear.
10. **Static void idle(void)::** This function is used for the construction of gears.

11. **Static void key(unsigned char key, int x, int y)::**This function is used for recognizing key pressed by the user.
12. **static void special(int k, int x, int y)::** This function is used for changing the view of the angle.
13. **static void reshape(int width, int height)::** This function is used for displaying the object after modifying its size and shape.
14. **static void init(void)::**The gear function is called through this function and the corresponding gears are created.
15. **void mouse(int btn, int state, int x, int y)::** This is the mouse function which is called whenever a mouse event is detected. It takes four arguments : btn for checking the button pressed, x and y to note the position i.e. co-ordinates where it is pressed.
16. **int main(int argc, char *argv[])::**This is the main function. In this function we initialize OpenGl, window size & window position. We also create the main window, create the sub window , register the keyboard and mouse functions, register the display function for the main window and the subwindow.

## 3.1 DATA FLOW DIAGRAM

The data flow diagram represents how the control is passed throughout the program

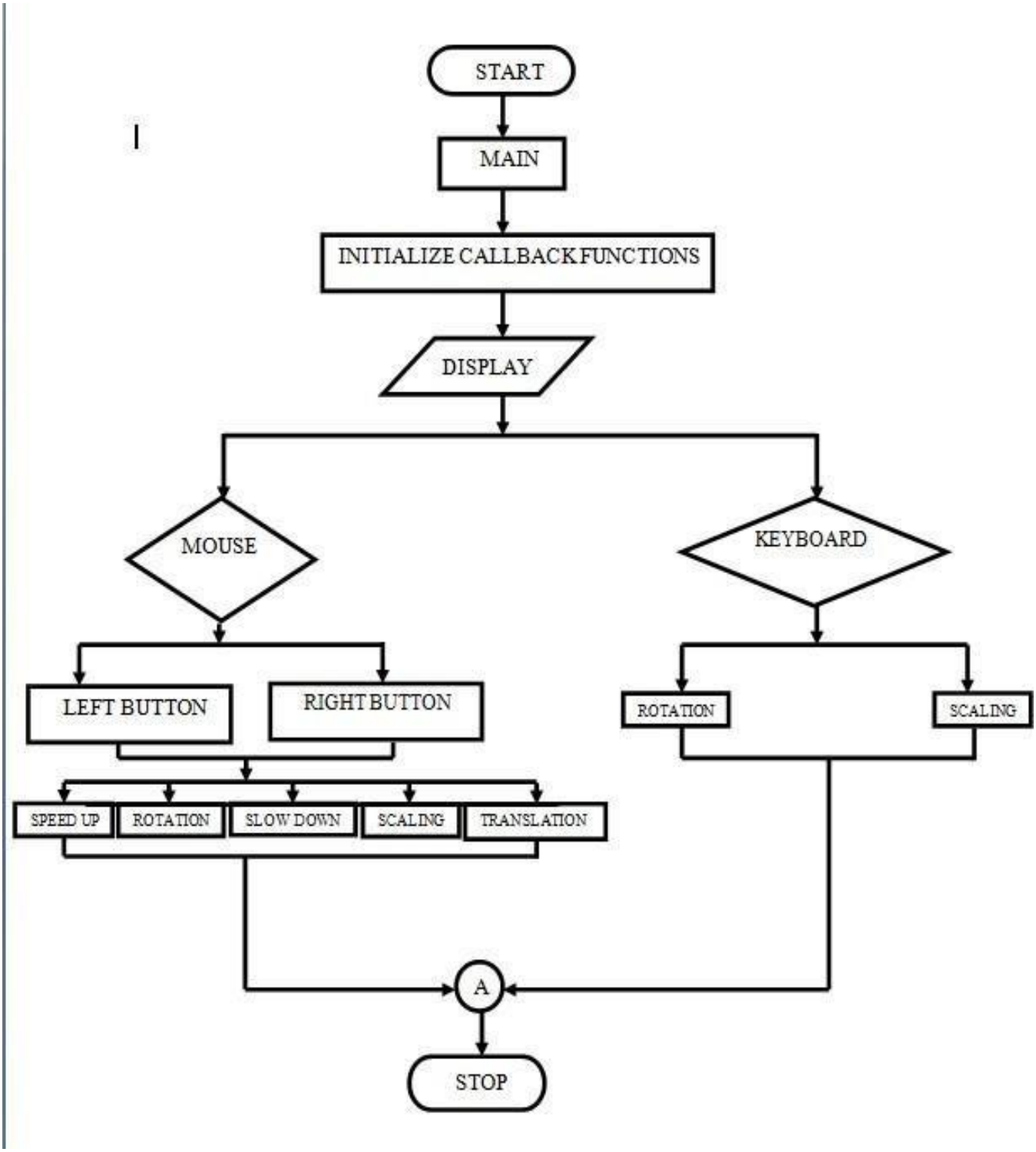


**Figure 3.1: Data and Control Flow Diagram [1]**

# CHAPTER 4

## IMPLEMENTATION

### 4.1 SOURCE CODE

```
#include<math.h>
#include<stdlib.h>
#include<GL/glut.h>
#include<stdio.h>
#include<windows.h>
#include<string.h>
#define M_PI 3.14159265
static void gear(GLfloat inner_radius, GLfloat outer_radius, GLfloat width, GLint teeth, GLfloat
tooth_depth)
{
        GLint i;
        GLfloat r0, r1, r2;
        GLfloat angle, da;
        GLfloat u, v, len;
        /******Shading******/
        glShadeModel(GL_FLAT);
        r0 = inner_radius;
        r1 = outer_radius - tooth_depth / 2.0;
        r2 = outer_radius + tooth_depth / 2.0;
        da = 2.0 * M_PI / teeth / 4.0;
        glNormal3f(0.0, 0.0, 1.0);
        /******** draw front face *********/
        glBegin(GL_QUAD_STRIP);
        for (i = 0; i <= teeth; i++)
        {
                angle = i * 2.0 * M_PI / teeth;
```

```
        glVertex3f(r0 * cos(angle), r0 * sin(angle), width * 0.5);
        glVertex3f(r1 * cos(angle), r1 * sin(angle), width * 0.5);
        glVertex3f(r0 * cos(angle), r0 * sin(angle), width * 0.5);
        glVertex3f(r1 * cos(angle + 3 * da), r1 * sin(angle + 3 * da), width * 0.5);
}
glEnd();

/******** draw front sides of teeth **********/
glBegin(GL_QUADS);
da = 2.0 * M_PI / teeth / 4.0;
for (i = 0; i <= teeth; i++)
{
        angle = i * 2.0 * M_PI / teeth;
        glVertex3f(r1 * cos(angle), r1 * sin(angle), width * 0.5);
        glVertex3f(r2 * cos(angle + da), r2 * sin(angle + da), width * 0.5);
        glVertex3f(r2 * cos(angle + 2 * da), r2 * sin(angle + 2 * da), width * 0.5);
        glVertex3f(r1 * cos(angle + 3 * da), r1 * sin(angle + 3 * da), width * 0.5);
}
glEnd();
glNormal3f(0.0, 0.0, -1.0);

/******** draw back face **********/
glBegin(GL_QUAD_STRIP);
for (i = 0; i <= teeth; i++)
{
        angle = i * 2.0 * M_PI / teeth;
        glVertex3f(r1 * cos(angle), r1 * sin(angle), -width * 0.5);
        glVertex3f(r0 * cos(angle), r0 * sin(angle), -width * 0.5);
        glVertex3f(r1 * cos(angle + 3 * da), r1 * sin(angle + 3 * da), -width * 0.5);
        glVertex3f(r0 * cos(angle), r0 * sin(angle), -width *  0.5);
}
glEnd();
```

```
/******** draw back sides of teeth ********/
glBegin(GL_QUADS);
da = 2.0 * M_PI / teeth / 4.0;
for (i = 0; i < teeth; i++)
{
        angle = i * 2.0 * M_PI / teeth;
        glVertex3f(r1 * cos(angle + 3 * da), r1 * sin(angle + 3 * da), -width * 0.5);
        glVertex3f(r2 * cos(angle + 2 * da), r2 * sin(angle + 2 * da), -width * 0.5);
        glVertex3f(r2 * cos(angle + da), r2 * sin(angle + da), -width * 0.5);
        glVertex3f(r1 * cos(angle), r1 * sin(angle), -width * 0.5);
}
glEnd();

/******** draw outward faces of teeth *********/
glBegin(GL_QUAD_STRIP);
for (i = 0; i < teeth; i++)
{
        angle = i * 2.0 * M_PI / teeth;
        glVertex3f(r1 * cos(angle), r1 * sin(angle), width * 0.5);
        glVertex3f(r1 * cos(angle), r1 * sin(angle), -width * 0.5);
        u = r2 * cos(angle + da) - r1 * cos(angle);
        v = r2 * sin(angle + da) - r1 * sin(angle);
        len = sqrt(u * u + v * v);
        u /= len;
        v /= len;
        glNormal3f(v, -u, 0.0);
        glVertex3f(r2 * cos(angle + da), r2 * sin(angle + da), width * 0.5);
        glVertex3f(r2 * cos(angle + da), r2 * sin(angle + da), -width * 0.5);
        glNormal3f(cos(angle), sin(angle), 0.0);
        glVertex3f(r2 * cos(angle + 2 * da), r2 * sin(angle + 2 * da), width * 0.5);
        glVertex3f(r2 * cos(angle + 2 * da), r2 * sin(angle + 2 * da), -width * 0.5);
        u = r1 * cos(angle + 3 * da) - r2 * cos(angle + 2 * da);
```

```
            v = r1 * sin(angle + 3 * da) - r2 * sin(angle + 2 * da);
            glNormal3f(v, -u, 0.0);
            glVertex3f(r1 * cos(angle + 3 * da), r1 * sin(angle + 3 * da), width * 0.5);
            glVertex3f(r1 * cos(angle + 3 * da), r1 * sin(angle + 3 * da), -width * 0.5);
            glNormal3f(cos(angle), sin(angle), 0.0);
        }
        angle = 0;
        glVertex3f(r1 * cos(angle), r1 * sin(angle), width * 0.5);
        glVertex3f(r1 * cos(angle), r1 * sin(angle), -width * 0.5);
        glEnd();
        angle = i * 2.0 * M_PI / teeth;

        /******* draw inside radius cylinder**********/
        glBegin(GL_QUAD_STRIP);
        for (i = 0; i <= teeth; i++)
        {
            angle = i * 2.0 * M_PI / teeth;
            glNormal3f(-cos(angle), -sin(angle), 0.0);
            glVertex3f(r0 * cos(angle), r0 * sin(angle), -width * 0.5);
            glVertex3f(r0 * cos(angle), r0 * sin(angle), width * 0.5);
        }
        glEnd();
}
static GLfloat view_rotx = 20.0, view_roty = 30.0, view_rotz = 0.0;
static GLfloat sx = 1.0, sy = 1.0, sz = 1.0, tx = 1.0, ty = 1.0, tz =1.0;
static GLint gear1, gear2,  gear3;
static GLfloat angle = 0.0;
static GLuint limit;
static GLuint count = 1;
static GLenum flag = 0;

void text(char str[])
```

```
{
        int i;
        for (i = 0; i < (int)strlen(str); i++)
        {
                glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, str[i]);
        }
}
void text1(char str[])
{
        int i;
        for (i = 0; i < (int)strlen(str); i++)
        {
                glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, str[i]);
        }
}

void text2(char str[])
{
        int i;
        for (i = 0; i < (int)strlen(str); i++)
        {
                glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, str[i]);
        }
}

void text3(char str[])
{
        int i;
        for (i = 0; i < (int)strlen(str); i++)
        {
                glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, str[i]);
        }
```

```
}

void text4(char str[])
{
        int i;
        for (i = 0; i < (int)strlen(str); i++)
        {
                glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, str[i]);
        }
}

void help()
{
        char a1[] = "While Executing Press and Hold \"H\" to See This Menu and Release to Continue";
        char b1[] = "Q: Quit";
        char c1[] = "D: Demo Menu";
        char d1[] = "M: Main Menu";
        char e1[] = "E: Execute";
        char f1[] = "g. Scaling Control";
        char g1[] = "f. Rotation Control(Z-axis)";
        char h1[] = "e. Rotation Control(Y-axis)";
        char i1[] = "d. Rotation Control(X-axis)";
        char j1[] = "c. Translation Control";
        char ab[] = "KEYBOARD OPTIONS :: ";
        char ac[] = "_____";
        char ad[] = "S: Positive Scaling";
        char ae[] = "s: Negative scaling";
        char af[] = "T: Positive Translation";
        char ag[] = "t: Negative Translation";
        char ah[] = "Z: Positive Z-Axis Rotation";
        char ai[] = "z: Negative Z-Axis Rotation";
        char one[] = "UP: Positive X-Axis Rotation";
```

```
char two[] = "DOWN: Negative X-Axis Rotation";
char two1[] = "RIGHT: Positive Y-Axis Rotation";
char two2[] = "LEFT: Negative Y-Axis Rotation";
char two3[] = "Gear Count Control ::";
char a[] = "_____";
char b[] = "1--> 1 Gear";
char c[] = "2--> 2 Gears";
char d[] = "MOUSE OPTIONS :: ";
char e[] = "3--> 3 Gears";
char f[] = "_____";
char g[] = "1. LEFT CLICK: Gears in RGB Color System";
char h[] = "2. MIDDLE CLICK: Gears in CMY Color System";
char i[] = "3. RIGHT CLICK: Menu Options to Change Speed";
char j[] = "a. Gear Speed Controls";
char k[] = "b. KeyBoard Directions";
glColor3f(1.0, 0.0, 0.0);
glRasterPos3f(-7.5, 4.5, -1.0);
text2(ab);
glRasterPos3f(-7.6, 4.4, -1.0);
text2(ac);
glRasterPos3f(-7.5, 4, -1.0);
text1(ad);
glRasterPos3f(-7.5, 3.5, -1.0);
text1(ae);
glRasterPos3f(-7.5, 3.0, -1.0);
text1(af);
glRasterPos3f(-7.5, 2.5, -1.0);
text1(ag);
glRasterPos3f(-7.5, 2.0, -1.0);
text1(ah);

glRasterPos3f(-7.5, 1.5, -1.0);
```

```
text1(ai);
glRasterPos3f(-7.5, 1.0, -1.0);
text1(one);
glRasterPos3f(-7.5, 0.5, -1.0);
text1(two);
glRasterPos3f(-7.5, -0.0, -1.0);
text1(two1);
glRasterPos3f(-7.5, -0.5, -1.0);
text1(two2);
glRasterPos3f(-7.5, -2.0, -1.0);
text2(two3);
glRasterPos3f(-7.6, -2.1, -1.0);
text2(a);
glRasterPos3f(-7.5, -2.5, -1.0);
text1(b);
glRasterPos3f(-7.5, -3, -1.0);
text1(c);
glRasterPos3f(-7.5, -3.5, -1.0);
text1(e);
glRasterPos3f(-0.5, 4.5, -1.0);
text2(d);
glRasterPos3f(-0.6, 4.4, -1.0);
text2(f);
glRasterPos3f(-0.5, 4, -1.0);
text1(g);
glRasterPos3f(-0.5, 3.5, -1.0);
text1(h);
glRasterPos3f(-0.5, 3, -1.0);
text1(i);
glRasterPos3f(0.5, 2.50, -1.0);
text3(j);
glRasterPos3f(0.5, 2.10, -1.0);
```

```
        text3(k);
        glRasterPos3f(0.5, 1.8, -1.0);
        text3(j1);
        glRasterPos3f(0.5, 1.4, -1.0);
        text3(i1);
        glRasterPos3f(0.5, 1.0, -1.0);
        text3(h1);
        glRasterPos3f(0.5, .5, -1.0);
        text3(g1);
        glRasterPos3f(0.5, .1, -1.0);
        text3(f1);
        glRasterPos3f(5, -2, -1.0);
        text(e1);
        glRasterPos3f(5, -2.5, -1.0);
        text(d1);
        glRasterPos3f(5, -3.0, -1.0);
        text(c1);
        glRasterPos3f(5, -3.5, -1.0);
        text(b1);
        glRasterPos3f(-5.5, -5.5, -1.0);
        text(a1);
        glFlush();
        glutSwapBuffers();
}

void cover()
{
        char aa[] = "Asst. Professor";
        char ab[] = "Mr. Ramesh B";
        char ac[] = "City Engineering College"; char ad[] = " 3D
        GEARS & TRANSFORMATIONS ";
        char ae[] = "                                                ";
```

```
char af[] = "USING OPENGL PACKAGE";
char ag[] = "Press D for Demo-Menu";
char ah[] = "Submitted by : ";
char name[] = "Akshay
Bhuvaneshwari M ";
char usn[]:
"1ce17cs008",”1ce17cs024”;
char guide[] = "Under the guidance of : ";
glColor3f(0.0, 1.0, 0.0);
glRasterPos3f(-6.5, 6.0, -1.0);
text(ac);
/* glRasterPos3f(-7.0,5,-1.0);
text("VI SEMESTER -- COMPUTER SCIENCE AND ENGINEERING");
glRasterPos3f(-5.0,4.0,0.0);
text("GRAPHICAL IMPLEMENTATION OF"); */
glRasterPos3f(-5.0, 3.0, -1.0);
text(ad);
glRasterPos3f(-5.0, 2.9, -1.0);
glColor3f(0.0, 0.0, 1.0);
text(ae);
glRasterPos3f(-4.0, 2.0, -1.0);
text(af);
glRasterPos3f(-4.0, 1, -1.0);
text2(ag);
glRasterPos3f(-10, -2.5, -1.0);
text(ah);
glRasterPos3f(-10, -3.0, -1.0);
text(name);
glRasterPos3f(-10, -3.5, -1.0);
text(usn);
glColor3f(0.0, 1.0, 0.0);
glRasterPos3f(4.5, -2.5, -1.0);
```

```
text2(" teacher"); glRasterPos3f(4.5, -
3.5, -1.0);
        text2("Senior Lecturer, Dept of CSE\n");*/
        glRasterPos3f(4.5, -3.0, -1.0);
        text2(ab);
        glRasterPos3f(4.5, -3.5, -1.0);
        text2(aa);
        /*glRasterPos3f(4.5, -4, -1.0);
        text2("");*/
        glFlush();
        glutSwapBuffers();
}

void wdraw(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glPushMatrix();
        glRotatef(view_rotx, 1.0, 0.0, 0.0);
        glRotatef(view_roty, 0.0, 1.0, 0.0);
        glRotatef(view_rotz, 0.0, 0.0, 1.0);
        glTranslatef(tx, ty, tz);
        glPushMatrix();
        glTranslatef(-3.0, -2.0, 0.0);
        glRotatef(angle, 0.0, 0.0, 1.0);
        glScalef(sx, sy, sz);
        glCallList(gear1);
        glPopMatrix();
        glPopMatrix();
        glFlush();
        glutSwapBuffers();

        count++;
```

```
        if (count == limit)
        {
                exit(0);
        }
}

void demo()
{
        char a[] = "PRESS 'Q' TO QUIT";
        char b[] = "PRESS 'M' FOR MAIN MENU";
        char c[] = "PRESS 'H' FOR HELP MENU";
        char d[] = "PRESS 'E' FOR ExecutionGears";
        char e[] = "__________________";
        char f[] = "USAGE OPTIONS : ";
        char g[] = "WELCOME TO DEMO-MENU";
        glColor3f(0.0, 1.0, 0.0);
        glRasterPos3f(-4.0, 2.8, -1.0);
        text(g);
        glRasterPos3f(-4.0, 1.0, -1.0);
        text1(f);
        glRasterPos3f(-4.1, 0.9, -1.0);
        text1(e);
        glRasterPos3f(-4.5, 0.2, -1.0);
        text2(d);
        glRasterPos3f(-4.5, -0.6, -1.0);
        text2(c);
        glRasterPos3f(-4.5, -1.4, -1.0);
        text2(b);
        glRasterPos3f(-4.5, -2.2, -1.0);
        text2(a);
        glFlush();
        glutSwapBuffers();
```

```
}


void draw(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        if (flag == 0)
        {
                cover();
        }
        else if (flag == 1)
        {
                demo();
        }
        else if (flag == 2)
        {
                help();
        }
        else if (flag == 3)
        {
                wdraw();
        }
        glFlush();
        glutPostRedisplay();
}

static void draw2(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glPushMatrix();
        glRotatef(view_rotx, 1.0, 0.0, 0.0);
        glRotatef(view_roty, 0.0, 1.0, 0.0);
```

```
        glRotatef(view_rotz, 0.0, 0.0, 1.0);
        glTranslatef(tx, ty, tz);
        glPushMatrix();
        glTranslatef(-3.0, -2.0, 0.0);
        glRotatef(angle, 0.0, 0.0, 1.0);
        glScalef(sx, sy, sz);
        glCallList(gear1);
        glPopMatrix();
        glPushMatrix();
        glTranslatef(3.1, -2.0, 0.0);
        glRotatef(-2.0 * angle - 9.0, 0.0, 0.0, 1.0);
        glScalef(sx, sy, sz);
        glCallList(gear2);
        glPopMatrix();
        glPopMatrix();
        glFlush();
        glutSwapBuffers();
        count++;
        if (count == limit)
        {
                exit(0);
        }
}

static void draw3(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glPushMatrix();
        glRotatef(view_rotx, 1.0, 0.0, 0.0);
        glRotatef(view_roty, 0.0, 1.0, 0.0);
        glRotatef(view_rotz, 0.0, 0.0, 1.0);
        glTranslatef(tx, ty, tz);
```

```
        glPushMatrix();
        glTranslatef(-3.0, -2.0, 0.0);
        glRotatef(angle, 0.0, 0.0, 1.0);
        glScalef(sx, sy, sz);
        glCallList(gear1);
        glPopMatrix();
        glPushMatrix();
        glTranslatef(3.1, -2.0, 0.0);
        glRotatef(-2.0 * angle - 9.0, 0.0, 0.0, 1.0);
        glScalef(sx, sy, sz);
        glCallList(gear2);
        glPopMatrix();
        glPushMatrix();
        glTranslatef(-3.1, 4.2, 0.0);
        glRotatef(-2.0 * angle - 25.0, 0.0, 0.0, 1.0);
        glScalef(sx, sy, sz);
        glCallList(gear3);
        glPopMatrix();
        glPopMatrix();
        glFlush();
        glutSwapBuffers();
        count++;
        if (count == limit)
        {
                exit(0);
        }
}

static void idle(void)
{
        angle += 0.25;
        glutPostRedisplay();
```

```
}

static void idle1(void)
{
        angle += 0.5;
        glutPostRedisplay();
}

static void idle2(void)
{
        angle += 0.1;
        glutPostRedisplay();
}

static void idle3(void)
{
        angle -= 0.5;
        glutPostRedisplay();
}

static void idle4(void)
{
        angle -= 0.1;
        glutPostRedisplay();
}

/******** change view angle, exit upon pressing any key **********//* ARGSUSED1 */
static void key(unsigned char key, int x, int y)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        switch (key)
```

```
        {
        case'M':flag = 0; draw(); break;
        case'D':flag = 1; draw(); break;
        case'H':flag = 2; draw(); break;
        case'E': flag = 3; glutSetWindowTitle("Gears"); glutPostRedisplay(); break;
        case's':sz += 0.125; glutSetWindowTitle("Gears(Positive Scaling)"); break;
        case'S':sz -= 0.125; glutSetWindowTitle("Gears(Negative Scaling)"); break;
        case't':tx += 0.5; ty+= 0.5; tz += 0.5; glutSetWindowTitle("Gears(Translation along all Axes in
                Positive Direction)"); break;
        case'T':tx -= 1.0; ty -= 1.0; tz -= 1.0; glutSetWindowTitle("Gears(Translationalong all Axes in
                Negative Direction)"); break;
        case'z':view_rotz += 5.0; glutSetWindowTitle("Gears(Rotation about Z axis)"); break;
        case'Z':view_rotz -= 5.0; glutSetWindowTitle("Gears(Rotation about Z axis)"); break;
        case'f': { glutIdleFunc(idle1); } break;
        case'1': {wdraw(); glutDisplayFunc(wdraw); glutSetWindowTitle("1 Gear"); } break;
        case'2': {draw2(); glutDisplayFunc(draw2); glutSetWindowTitle("2 Gears"); } break;
        case'3': {draw3(); glutDisplayFunc(draw3); glutSetWindowTitle("3 Gears"); } break;
        case'l': {glutIdleFunc(idle2); } break;
        case'Q': exit(0);
        }
        glutPostRedisplay();
}

/******* change view angle ********//* ARGSUSED1 */
static void special(int k, int x, int y)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        switch (k)
        {
        case GLUT_KEY_UP:
                view_rotx += 5.0; glutSetWindowTitle("Gears(Rotation about X axis)");
                break;
```

```
        case GLUT_KEY_DOWN:
                view_rotx -= 5.0; glutSetWindowTitle("Gears(Rotation about X axis)");
                break;
        case GLUT_KEY_LEFT:
                view_roty += 5.0; glutSetWindowTitle("Gears(Rotation about Y axis)");
                break;
        case GLUT_KEY_RIGHT:
                view_roty -= 5.0; glutSetWindowTitle("Gears(Rotation about Y axis)");
                break;
        default:return;
        }
        glutPostRedisplay();
}
/******* new window size or exposure ********/
static void reshape(int width, int height)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        GLfloat h = (GLfloat)height / (GLfloat)width;
        glViewport(0, 0, (GLint)width, (GLint)height);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glFrustum(-1.0, 1.0, -h, h, 5.0, 60.0);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glTranslatef(0.0, 0.0, -50.0);
}

static void init(void)
{
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
        static GLfloat pos[4] = { 2.0, 2.0, 10.0, 0.0 };
        static GLfloat red[4] = { 0.8, 0.1, 0.0, 1.0 };
        static GLfloat green[4] = { 0.0, 0.8, 0.2, 1.0 };
        static GLfloat blue[4] = { 0.2, 0.2, 1.0, 1.0 };
        glClearColor(1.0, 1.0, 1.0, 1.0);
        glLightfv(GL_LIGHT0, GL_POSITION, pos);
        glEnable(GL_CULL_FACE);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        /******* make the gears ********/
        gear1 = glGenLists(1);
        glNewList(gear1, GL_COMPILE);
        glMaterialfv(GL_FRONT, GL_AMBIENT, red);
        gear(1.0, 4.0, 1.0, 20, 0.7);
        glEndList();
        gear2 = glGenLists(1);
        glNewList(gear2, GL_COMPILE);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, green);
        gear(0.5, 2.0, 2.0, 10, 0.7);
        glEndList();
        gear3 = glGenLists(1);
        glNewList(gear3, GL_COMPILE);
        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, blue);
        gear(1.3, 2.0, 0.5, 10, 0.7);
        glEndList();
        glEnable(GL_NORMALIZE);
}
void demo_menu(int id)
{
        switch (id)
        {
        case 7: exit(0);
```

```
            break;
        }
        glutPostRedisplay();
}

void func1(int id)
{
        switch (id)
        {
        case 1: glutIdleFunc(idle);
                break;
        case 2: glutIdleFunc(idle1);
                break;
        case 3: glutIdleFunc(idle2);
                break;
        case 4: glutIdleFunc(idle3);
                break;
        case 5: glutIdleFunc(idle4);
                break;
        }
        glutPostRedisplay();
}

void func2(int id)
{
        switch (id)
        {
        case 1: tx += 1.0;
                break;
        case 2: tx += 2.0;
                break;
        case 3: tx -= 1.0;
```

```
            break;
        case 4: tx -= 2.0;
            break;
        case 5: ty += 1.0;
            break;
        case 6: ty += 2.0;
            break;
        case 7: ty -= 1.0;
            break;
        case 8: ty -= 2.0;
            break;
        case 9: tz += 1.0;
            break;
        case 10:tz += 2.0;
            break;
        case 11:tz -= 1.0;
            break;
        case 12:tz -= 2.0;
            break;
        }
        glutPostRedisplay();
}

void func3(int id)
{
        switch (id)
        {
        case 1: view_rotx += 5.0;
            break;
        case 2: view_rotx += 10.0;
            break;
        case 3: view_rotx -= 5.0;
```

```
                break;
        case 4: view_rotx -= 5.0;
                break;
        }
        glutPostRedisplay();
}

void func4(int id)
{
        switch (id)
        {
        case 1: view_roty += 5.0;
                break;
        case 2: view_roty += 10.0;
                break;
        case 3: view_roty -= 5.0;
                break;
        case 4: view_roty -= 5.0;
                break;
        }
        glutPostRedisplay();
}

void func5(int id)
{
        switch (id)
        {
        case 1: view_rotz += 5.0;
                break;
        case 2: view_rotz += 10.0;
                break;
        case 3: view_rotz -= 5.0;
```

```
                break;
        case 4: view_rotz -= 5.0;
                break;
        }
        glutPostRedisplay();
}

void func6(int id)
{
        switch (id)
        {
        case 1: sz += 2.0;
                break;
        case 2: sz -= 2.0;
                break;
        }
        glutPostRedisplay();
}

void null_select(int mode)
{ }

Void mouse(int btn, int state, int x, int y)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
        {
                static GLfloat pos[4] = { 2.0, 2.0, 5.0, 2.0 };
                static GLfloat red[4] = { 1.0, 0.0, 0.0, 0.0 };
                static GLfloat blue[4] = { 0.0, 0.0, 1.0, 0.0 };
                static GLfloat green[4] = { 0.0, 1.0, 0.0, 0.0 };
                glClearColor(1.0, 1.0, 1.0, 1.0);
```

```
        glLightfv(GL_LIGHT0, GL_POSITION, pos);
        glEnable(GL_CULL_FACE);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);

        /******* make the gears ********/
        gear1 = glGenLists(1);
        glNewList(gear1, GL_COMPILE);
        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, red);
        gear(1.0, 4.0, 1.0, 20, 0.7);
        //glutWireSphere(1.0, 20, 16);
        glEndList();
        gear2 = glGenLists(1);
        glNewList(gear2, GL_COMPILE);
        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, blue);
        gear(0.5, 2.0, 2.0, 10, 0.7);
        //glutWireSphere(1.0, 20, 16);
        glEndList();
        gear3 = glGenLists(1);
        glNewList(gear3, GL_COMPILE);
        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, green);
        gear(0.8, 2.0, 0.5, 10, 0.7);
        //glutWireSphere(0.8, 20, 16);
        glEndList();
        glEnable(GL_NORMALIZE);
}

if (btn == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN)
{
        /*static GLfloat pos[4]={2.0, 2.0, 5.0, 2.0};
        static GLfloat red[4]={1.0, 1.0, 0.0, 1.0};
        static GLfloat blue[4]={0.0, 1.0, 1.0, 1.0};
```

```
static GLfloat green[4]={0.5, 0.5, 0.5, 1.0};*/
static GLfloat pos[4] = { 2.0, 2.0, 5.0, 2.0 };
static GLfloat cyan[4] = { 0.0, 1.0, 1.0, 1.0 };
static GLfloat magenta[4] = { 1.0, 0.0, 1.0, 1.0 };
static GLfloat yellow[4] = { 1.0, 1.0, 0.0, 1.0 };
glClearColor(1.0, 1.0, 1.0, 1.0);
glLightfv(GL_LIGHT0, GL_POSITION, pos);
glEnable(GL_CULL_FACE);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_DEPTH_TEST);
/******* make the gears ********/
gear1 = glGenLists(1);
glNewList(gear1, GL_COMPILE);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, cyan);
gear(1.0, 4.0, 1.0, 20, 0.7);
//glutWireSphere(1.0, 20, 16);
glEndList();
gear2 = glGenLists(1);
glNewList(gear2, GL_COMPILE);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, magenta);
gear(0.5, 2.0, 2.0, 10, 0.7);
//glutWireSphere(1.0, 20, 16);
glEndList();
gear3 = glGenLists(1);
glNewList(gear3, GL_COMPILE);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, yellow);
gear(0.8, 2.0, 0.5, 10, 0.7);
//glutWireSphere(0.8, 20, 16);
glEndList();

glEnable(GL_NORMALIZE);
```

```
        }
}

int main(int argc, char *argv[])
{
        int func, menu, t1, r1, r2, r3, s1, s2;

        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        if (argc > 1)
        {
                limit = atoi(argv[1]) + 1;
        }/******* do 'n' frames then exit *******/
        else
        {
                limit = 0;
        }
        glutInitWindowSize(800, 600);
        glutCreateWindow("3D GEARS & TRANSFORMATIONS");
        init();
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        menu = glutCreateMenu(null_select);
        glutAddMenuEntry("Rotation about X axis     : up,down", 0);
        glutAddMenuEntry("Rotation about Y axis     : right,left", 0);
        glutAddMenuEntry("Rotation about Z axis     : z,Z", 0);
        glutAddMenuEntry("Scale                     : s,S", 0);
        glutAddMenuEntry("Translate along all Axes : t,T", 0);
        glutAddMenuEntry("Continue                  : c", 0);
        func = glutCreateMenu(func1);
        glutAddMenuEntry("Normal", 1);
        glutAddMenuEntry("High Forward", 2);
        glutAddMenuEntry("Low Forward", 3);
```

```
glutAddMenuEntry("High Backward",4);
glutAddMenuEntry("Low Backward", 5);
t1 = glutCreateMenu(func2);
glutAddMenuEntry("1 unit1(X-Axis)",1);
glutAddMenuEntry("2 units(X-Axis)",2);
glutAddMenuEntry("-1 unit1(X-Axis)",3);
glutAddMenuEntry("-2 units(X-Axis)",4);
glutAddMenuEntry("1 unit1(Y-Axis)",5);
glutAddMenuEntry("2 units(Y-Axis)",6);
glutAddMenuEntry("-1 unit1(Y-Axis)",7);
glutAddMenuEntry("-2 units(Y-Axis)",8);
glutAddMenuEntry("1 unit1(Z-Axis)", 9);
glutAddMenuEntry("2 units(Z-Axis)", 10);
glutAddMenuEntry("-1 unit1(Z-Axis)", 11);
glutAddMenuEntry("-2 units(Z-Axis)", 12);
r1 = glutCreateMenu(func3);
glutAddMenuEntry("+5 degrees", 1);
glutAddMenuEntry("+10 degrees", 2);
glutAddMenuEntry("-5 degrees", 3);
glutAddMenuEntry("-10 degrees", 4);
r2 = glutCreateMenu(func4);
glutAddMenuEntry("+5 degrees", 1);
glutAddMenuEntry("+10 degrees", 2);
glutAddMenuEntry("-5 degrees", 3);
glutAddMenuEntry("-10 degrees", 4);
r3 = glutCreateMenu(func5);
glutAddMenuEntry("+5 degrees", 1);
glutAddMenuEntry("+10 degrees", 2);
glutAddMenuEntry("-5 degrees", 3);
glutAddMenuEntry("-10 degrees", 4);

s1 = glutCreateMenu(func6);
```

```
        glutAddMenuEntry("+2 units", 1);
        glutAddMenuEntry("-2 units", 2);
        glutCreateMenu(demo_menu);
        glutAddSubMenu("Speed", func);
        glutAddSubMenu("Keyboard Directions", menu);
        glutAddSubMenu("Translate By: ", t1);
        glutAddSubMenu("Rotate X-Axis by: ", r1);
        glutAddSubMenu("Rotate Y-Axis by: ", r2);
        glutAddSubMenu("Rotate Z-Axis by: ", r3);
        glutAddSubMenu("Scale By: ", s1);
        glutAddMenuEntry("Quit, Press q", 7);
        glutAttachMenu(GLUT_RIGHT_BUTTON);
        glutDisplayFunc(draw);
        glutReshapeFunc(reshape);
        glutKeyboardFunc(key);
        glutSpecialFunc(special);
        glutMouseFunc(mouse);
        glutIdleFunc(idle);
        glEnable(GL_DEPTH_TEST);
        glutMainLoop();
        return 0;
}
```

# CHAPTER 5

# TESTING AND RESULTS

## 5.1 DIFFERENT TYPES OF TESTING

| Sl. No. | Type of testing | Description |
|---|---|---|
| 1. | Unit Testing | Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components. |
| 2. | Module Testing | A module is a collection of dependent components such as an object class, an abstract data type or some looser collection of procedures and functions. A module related components, so can be tested without other system modules. |
| 3. | System Testing | This is concerned with finding errors that result from unanticipated interaction between sub-system interface problems. |
| 4. | Acceptance Testing | The system is tested with data supplied by the system customer rather than simulated test data. |

## 5.2 TEST CASES

| Sl No | Test Input | Expected Results | Observed Results | Remarks |
|---|---|---|---|---|
| 1 | Press D for Demo menu | Demo Menu to be displayed | Menu displayed | pass |
| 2 | Press D for Demo menu | Demo Menu to be displayed | Menu not displayed | fail |
| 3 | Press E for Execution of gears | Working of gears must be displayed | Working of gears displayed | pass |
| 4 | Press H for Help menu | Description of keys and their functions for the execution of gears is to be displayed. | Description of keys and their functions for the execution of gears is displayed. | pass |
| 5 | Press M for Main menu | Home Screen should be displayed | Home Screen is displayed | pass |
| 6 | Press Q for quit | Exit | Exit | pass |

# CHAPTER 6

## SNAPSHOTS

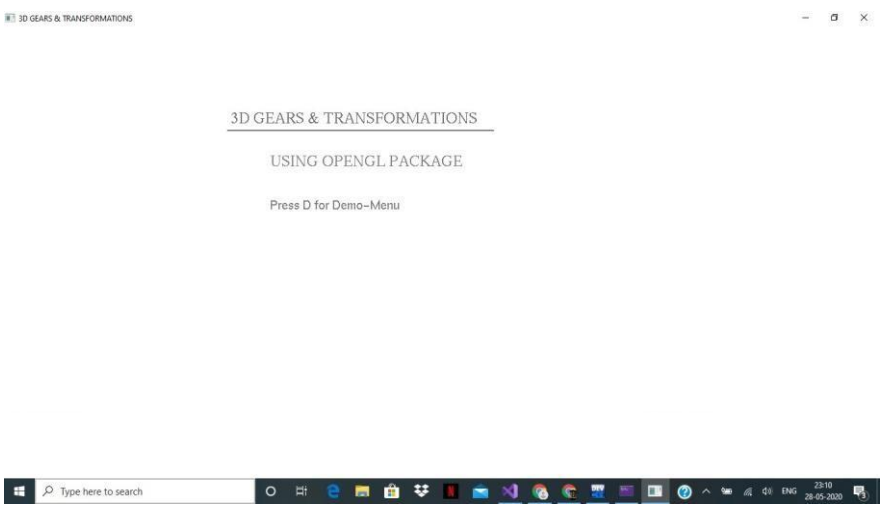


**Fig 6.1 Home Screen**

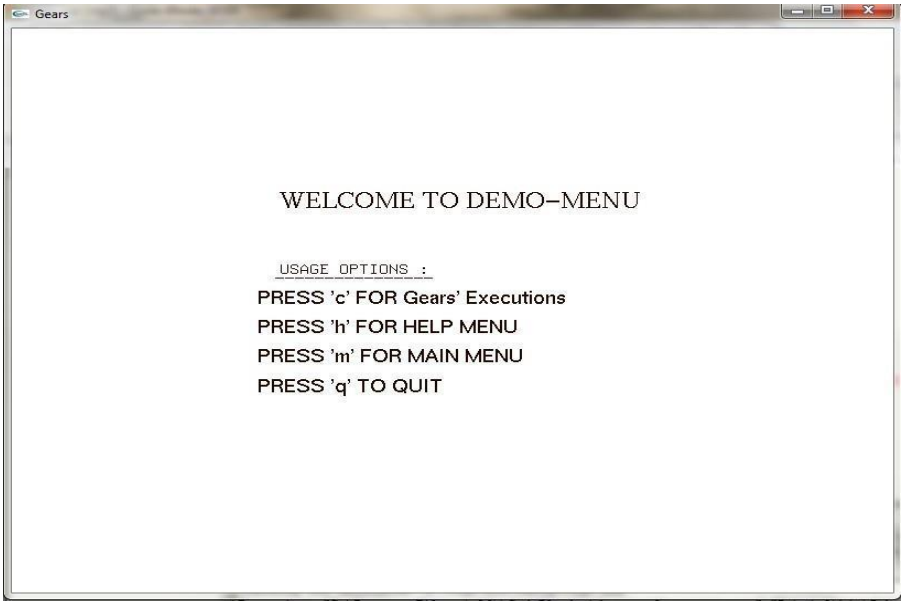


**Fig 6.2 Demo menu**

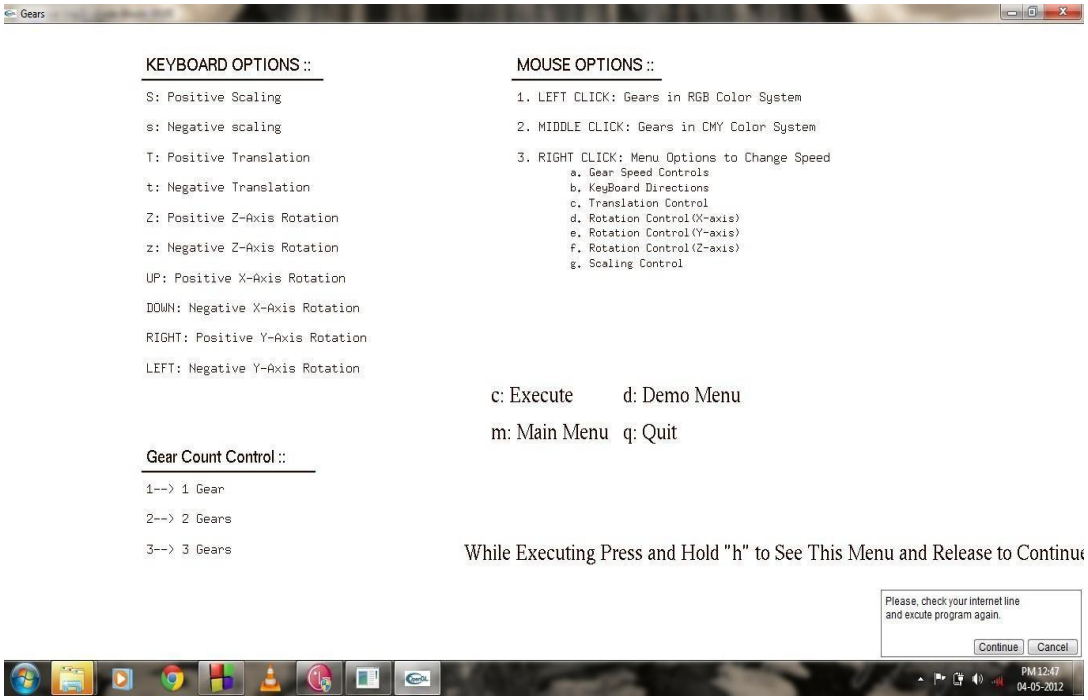


**Fig 6.3 Help menu**

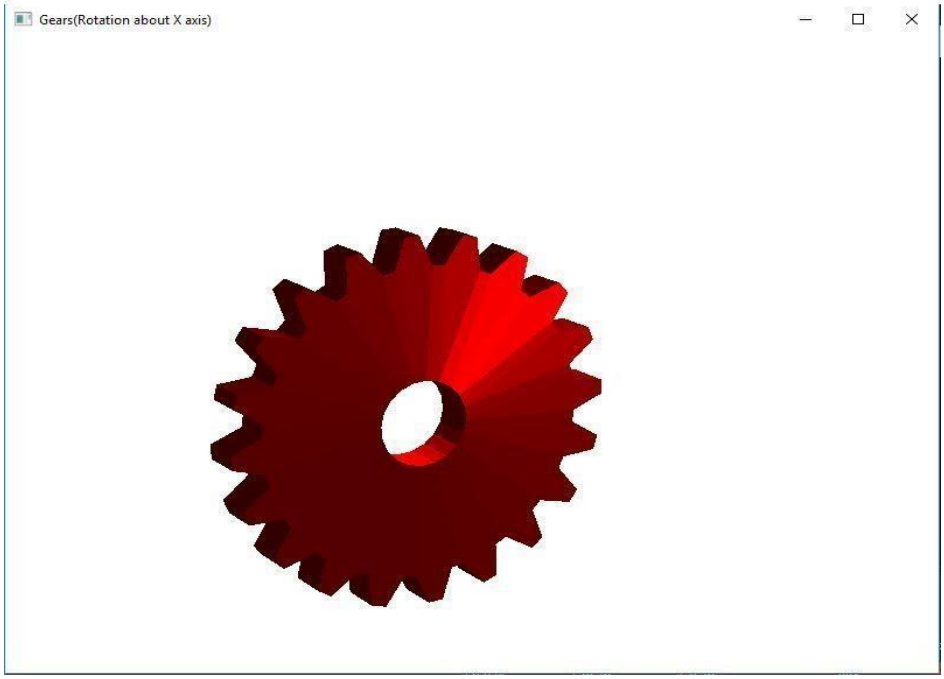


**Figure 6.4: Single Gear**

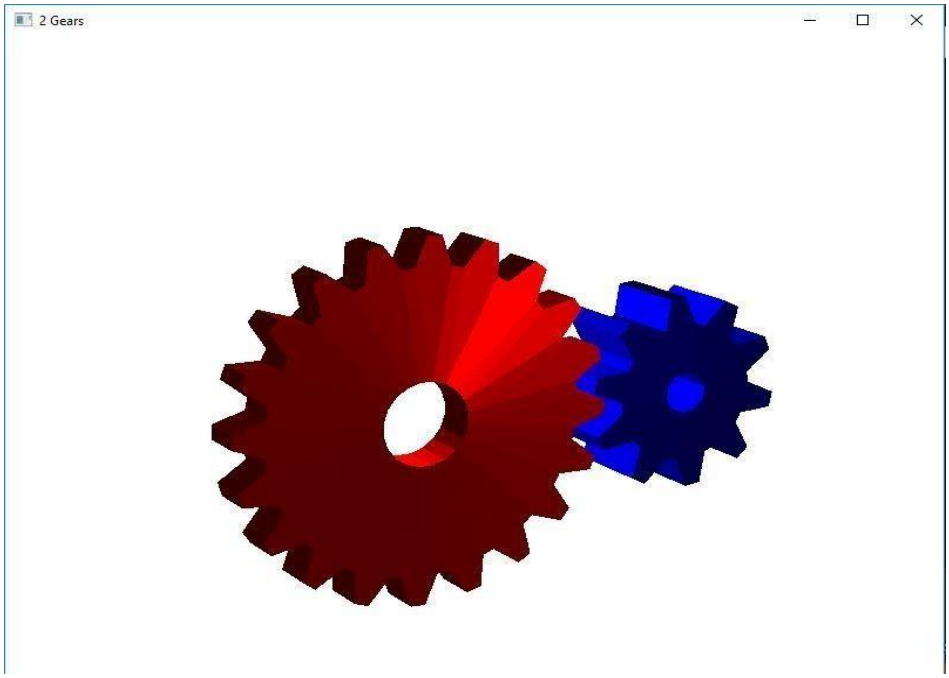


**Figure 6.5: Double Gears**

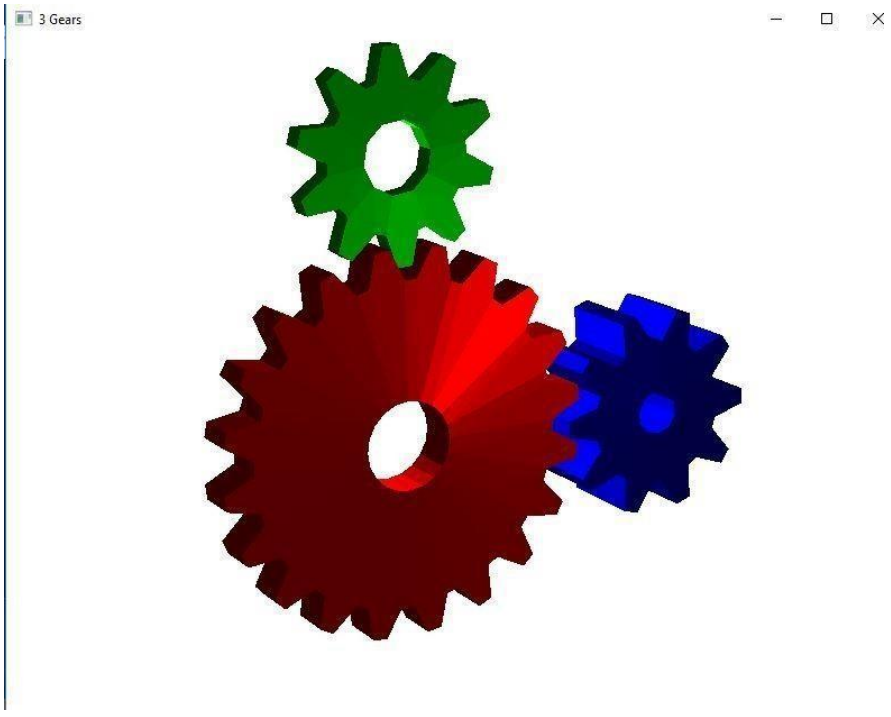


**Figure 6.6: Triple Gears**

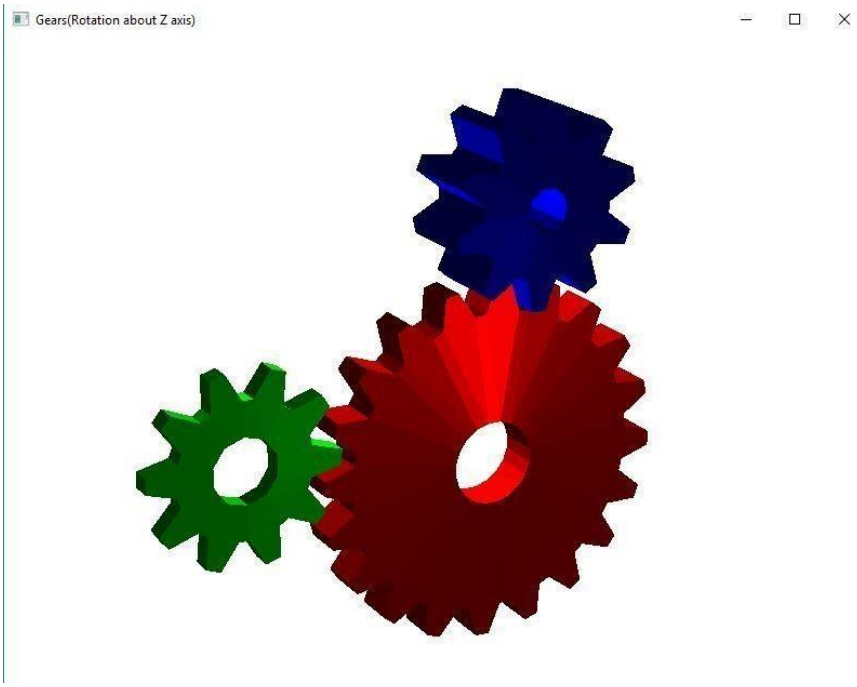


**Figure 6.7: Rotation about Z axis**

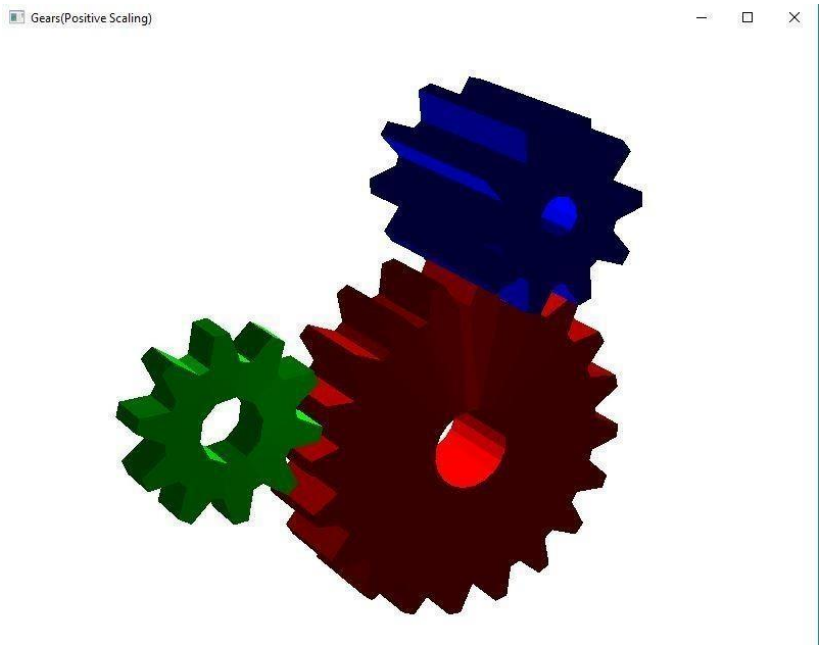


**Figure 6.8: Positive Scaling of gears**

# CONCLUSION

The project "**3D GEARS & TRANSFORMATIONS**" was designed and implemented to show the rotation, transformation and scaling of a gear. In this project rotation, translation and scaling about 3 axes i.e. X, Y and Z axis have been successfully implemented. The speed of the rotation of all the three gears can be altered at different levels. Shading of the gear is also implemented successfully. In this project ,the further enhancement for it could be adding of textures to gears, increasing number of gears up to 'n' number, transparency to gears could be provided, lighting effect can be improved using different light source and still better code could be provided for drawing gears which will improve the codeefficiency

# BIBLIOGRAPHY

## 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].www.opengl.org

[4].https://www.learnopengl.com

[5].https://www.khronos.org

# DECLARATION

We student of 6th semester BE, Computer Science and Engineering College hereby declare that project work entitled "3D Gears and Transformations" has been carried out by us at City Engineering College, Bengaluru and submitted in partial fulfillment of the course requirement for the award of the degree of **Bachelor of Engineering in Computer Science and Engineering of Vivesvaraya Technological University, Belgaum**, during the academic year 2019-2020.

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

Date**:**

Place: Bangalore

**AKSHAY AMRUT MORAB**

**(1CE17CS008)**

**BHUVANESHWARI M**

**(1CE17CS024)**