Department Of Mathematics & Computing

ASSIGNMENT

NAME: **Newton Mallick**

SUBJECT: **Computer Graphics**

Submitted To: **Dr Badam Singh Kushvah**

Serial No:**25**

Admission No: **17JE003104**

OBJECTIVE 1

This project aims to design न्यूटन (NEWTON in Devnagri) in the 3-D name using Bezier Curve in OpenGL.

METHODOLOGY

1. *name()*

Create a function name to construct न्यूटन (NEWTON in Devnagri) in the 3D plane using the concept of Bezier curve.

Create a cuboid box to use as background

Then the axis is uplifted using to `(00,-30,15)` *`to glTranslatef()`*.

Then the curve & straight line are drawn using the Bezier Curve. Lines with light shade joint the cuboid and the न्यूटन. The front is made in black to make it more visually stunning, but the cuboid is made different.

2. **myinit()**

   Initial windows setting when the program is executed and called by GLUT.

3. ***resize()***

   Function to resize the viewport and projection matrix according to input width and height. Called by GLUT.

4. ***display()***

   Use to apply the function on the object to change its orientation.Call the function name().

5. ***keyInput()***

   Consist of function activates when specific keys pressed, changes in the angle of the object or close the window.

6. **specialKeyInput()**

   Allows moving the object in the 3D coordinate system using arrow keys to move left, right, left & right, and p/P to move in and out of the screen.

7. **main()**

   Main which call all the function and is in the loop.

CODE

```
#include<iostream>

#include<math.h>

#include<GL/glut.h>


#define mode GL_LINE_LOOP

using namespace std;


float Xangle=0,Yangle=0,Zangle=0;

float xx=0,yy=0,zz=0;

float angle=0;

static bool isAnimate=0;

static int animatePeriod=100;

void myinit()

{

   glClearColor(1,1,1,0);

   glEnable(GL_DEPTH_TEST);

   glMatrixMode(GL_PROJECTION);

   gluPerspective(60, 1, 1, 900);

   //glOrth2D(0,100,0,100);

}


void resize(int w,int h)

{

   glViewport(0,0,w,h);

   glMatrixMode(GL_PROJECTION);

   glLoadIdentity();

   glFrustum(-7.0,2.0,-5.0,5.0,5.0,200.0);
   glMatrixMode(GL_MODELVIEW);
}

void name()
{
   //background
   glColor3f(0,0.5,1);
   glPointSize(7) ;
   glBegin(GL_POINTS);
   int x1=-40,y1=0,z1=0,x2=30,y2=0,z2=0;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
           glVertex2f(px,py);// -------
   }
   glEnd();

   glBegin(GL_POINTS);
   x1=30 , y1=0 , z1=0 , x2=30 , y2=70 , z2=0;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
           glVertex2f(px,py);// -------
   }
   glEnd();
   glBegin(GL_POINTS);
   x1=30 , y1=70 , z1=0 , x2=-40 , y2=70 , z2=0;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
           glVertex2f(px,py);// -------
   }
   glEnd();

   glBegin(GL_POINTS);
   x1=-40,y1=0,x2=-40,y2=70;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
           glVertex2f(px,py);// -------
   }
   glEnd();

   glColor3f(0,1,0) ;
   glBegin(GL_POINTS);
   x1 = 30,y1 = 0,z1 = 0,x2 = 30,y2 = 0,z2 = -20;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
   float pz=z1*(1-t)+z2*(t);
           glVertex3f(px,py,pz);// -------
   }
   glEnd();

   glBegin(GL_POINTS);
   x1 = 30,y1 = 70,z1 = 0,x2 = 30,y2 = 70 , z2 = -20;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);

   float pz=z1*(1-t)+z2*(t);

           glVertex3f(px,py,pz);// -------

   }

   glEnd();


   glBegin(GL_POINTS);

   x1 = -40,y1 = 0,z1 = 0,x2 = -40,y2 = 0 , z2 = -20;

   for(float t=0;t<=1;t+=0.001){

   float px=x1*(1-t)+x2*(t);

   float py=y1*(1-t)+y2*(t);

   float pz=z1*(1-t)+z2*(t);

           glVertex3f(px,py,pz);// -------

   }

   glEnd();


   glBegin(GL_POINTS);

   x1 = -40,y1 = 70,z1 = 0,x2 = -40,y2 = 70 , z2 = -20;

   for(float t=0;t<=1;t+=0.001){

   float px=x1*(1-t)+x2*(t);

   float py=y1*(1-t)+y2*(t);

   float pz=z1*(1-t)+z2*(t);

           glVertex3f(px,py,pz);// -------

   }

   glEnd();


   glColor3f(0.5,0.4,0.3) ;

   glBegin(GL_POINTS);

   x1 = 30,y1 = 0,z1 = -20,x2 = -40,y2 = 0 , z2 = -20;
```

```
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1 = 30,y1 = 0,z1 = -20,x2 = 30,y2 = 70 , z2 = -20;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1 = 30,y1 = 70,z1 = -20,x2 = -40 ,y2 = 70 , z2 = -20;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
```

```
x1 = -40,y1 = 70,z1 = -20,x2 = -40,y2 = 0 , z2 = -20;

for(float t=0;t<=1;t+=0.001){

float px=x1*(1-t)+x2*(t);

float py=y1*(1-t)+y2*(t);

float pz=z1*(1-t)+z2*(t);

        glVertex3f(px,py,pz);// -------

}

glEnd();



///////////////////////NAME/////////////////////////////

glTranslatef(00,-30,15);

glColor3f(0.2,0.2,0.2) ;

glPointSize(3) ;


glBegin(GL_POINTS);

x1=-23,y1=80,x2=25,y2=80;

for(float t=0;t<=1;t+=0.001){

float px=x1*(1-t)+x2*(t);

float py=y1*(1-t)+y2*(t);

        glVertex2f(px,py);// -------

}

glEnd();


glBegin(GL_POINTS);

x1=-10,y1=80,x2=-10,y2=50;

for(float t=0;t<=1;t+=0.001){

float px=x1*(1-t)+x2*(t);

float py=y1*(1-t)+y2*(t);
```

```
        glVertex2f(px,py);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-15,y1=65,x2=-23,y2=65;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
        glVertex2f(px,py);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-23,y1=65,x2=-19,y2=60;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
        glVertex2f(px,py);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-19,y1=65,x2=-19,y2=60;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
        glVertex2f(px,py);// -------
}
```

```
   glEnd();


   glBegin(GL_POINTS);
   x1=-5,y1=45,x2=-10,y2=50;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
           glVertex2f(px,py);// -------
   }
   glEnd();


   glBegin(GL_POINTS);
   int a1=-20,a2= 80 , b1=-14,b2=75 , c1=-14,c2=69 , d1=-15,d2=65 ;
   for(float t=0;t<=1;t+=0.001)
   {
       float px = a1*pow((1-t),3) + 3*b1*t*(1-t)*(1-t) + 3*c1*t*t*(1-t) +
d1*t*t*t ;
       float py = a2*pow((1-t),3) + 3*b2*t*(1-t)*(1-t) + 3*c2*t*t*(1-t) +
d2*t*t*t ;
       glVertex2f(px,py) ;
   }
   glEnd();


   glBegin(GL_POINTS);
   x1 = -10,y1=48 ;
   for(int k=300,r=2;k<800;k++)
   {
       double x,y;
        x=x1+r*sin(k);
        y=y1+r*cos(k);
       glVertex2f(x,y);


   }
   glEnd();


   glBegin(GL_POINTS);
   a1=-15,a2= 65 , b1=-19,b2=63 , c1=-13,c2=50 , d1=-10,d2=55 ;
   for(float t=0;t<=1;t+=0.001)
   {
       float px = a1*pow((1-t),3) + 3*b1*t*(1-t)*(1-t) + 3*c1*t*t*(1-t) +
d1*t*t*t ;
       float py = a2*pow((1-t),3) + 3*b2*t*(1-t)*(1-t) + 3*c2*t*t*(1-t) +
d2*t*t*t ;
       glVertex2f(px,py) ;
   }
   glEnd();


   glBegin(GL_POINTS);
   x1=8,y1=80,x2=8,y2=65;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
           glVertex2f(px,py);// -------
   }
   glEnd();


   glBegin(GL_POINTS);
   a1=8,a2= 65 , b1=-5,b2=60 , c1=2,c2=30 , d1=12,d2=48 ;
   for(float t=0;t<=1;t+=0.001)

   {

       float px = a1*pow((1-t),3) + 3*b1*t*(1-t)*(1-t) + 3*c1*t*t*(1-t) +
d1*t*t*t ;

       float py = a2*pow((1-t),3) + 3*b2*t*(1-t)*(1-t) + 3*c2*t*t*(1-t) +
d2*t*t*t ;

       glVertex2f(px,py) ;

   }

   glEnd();


   glBegin(GL_POINTS);

   x1=20,y1=80,x2=20,y2=45;

   for(float t=0;t<=1;t+=0.001){

   float px=x1*(1-t)+x2*(t);

   float py=y1*(1-t)+y2*(t);

           glVertex2f(px,py);// -------

   }

   glEnd();


   glBegin(GL_POINTS);

   x1=12,y1=65,x2=20,y2=65;

   for(float t=0;t<=1;t+=0.001){

   float px=x1*(1-t)+x2*(t);

   float py=y1*(1-t)+y2*(t);

           glVertex2f(px,py);// -------

   }

   glEnd();


   glBegin(GL_POINTS);
   x1=15,y1=65,x2=15,y2=55;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
           glVertex2f(px,py);// -------
   }
   glEnd();

   glBegin(GL_POINTS);
   x1=12,y1=65,x2=15,y2=55;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
           glVertex2f(px,py);// -------
   }
   glEnd();
   /////////////////////////
///////////////////////////////////GAP//////////////...


   glColor3f(0.8,0.8,0.8) ;
   glBegin(GL_POINTS);
   x1=-23,y1=80,z1=0,x2=-23,y2=80,z2=-15;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
   float pz=z1*(1-t)+z2*(t);
           glVertex3f(px,py,pz);// -------
   }
   glEnd();
```

```
glBegin(GL_POINTS);
x1=25,y1=80,z1=0,x2=25,y2=80,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-23,y1=80,z1=0,x2=-23,y2=80,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-10,y1=80,z1=0,x2=-10,y2=80,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
```

```
glEnd();


glBegin(GL_POINTS);

x1=-10,y1=50,z1=0,x2=-10,y2=50,z2=-15;

for(float t=0;t<=1;t+=0.001){

float px=x1*(1-t)+x2*(t);

float py=y1*(1-t)+y2*(t);

float pz=z1*(1-t)+z2*(t);

        glVertex3f(px,py,pz);// -------

}

glEnd();


glBegin(GL_POINTS);

x1=-23,y1=65,z1=0,x2=-23,y2=65,z2=-15;

for(float t=0;t<=1;t+=0.001){

float px=x1*(1-t)+x2*(t);

float py=y1*(1-t)+y2*(t);

float pz=z1*(1-t)+z2*(t);

        glVertex3f(px,py,pz);// -------

}

glEnd();


glBegin(GL_POINTS);

x1=-15,y1=65,z1=0,x2=-15,y2=65,z2=-15;

for(float t=0;t<=1;t+=0.001){

float px=x1*(1-t)+x2*(t);

float py=y1*(1-t)+y2*(t);

float pz=z1*(1-t)+z2*(t);

        glVertex3f(px,py,pz);// -------
```

```
}
glEnd();

glBegin(GL_POINTS);
x1=-19 ,y1=60 ,z1=0,x2= -19,y2=60 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-19 ,y1=65 ,z1=0,x2=-19 ,y2=65 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-10 ,y1=53 ,z1=0,x2=-10 ,y2=53 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
```

```
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-10 ,y1=50 ,z1=0,x2=-10 ,y2=50 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=12 ,y1=48 ,z1=0,x2=12 ,y2=48 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-12 ,y1=48 ,z1=0,x2=-12 ,y2=48 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-8 ,y1=48 ,z1=0,x2=-8 ,y2=48 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-10 ,y1=46 ,z1=0,x2=-10 ,y2=46 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=8 ,y1=80 ,z1=0,x2=8 ,y2=80 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=8 ,y1=65 ,z1=0,x2=8 ,y2=65 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=-5 ,y1=45 ,z1=0,x2=-5 ,y2=45 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=20 ,y1=45 ,z1=0,x2=20 ,y2=45 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
```

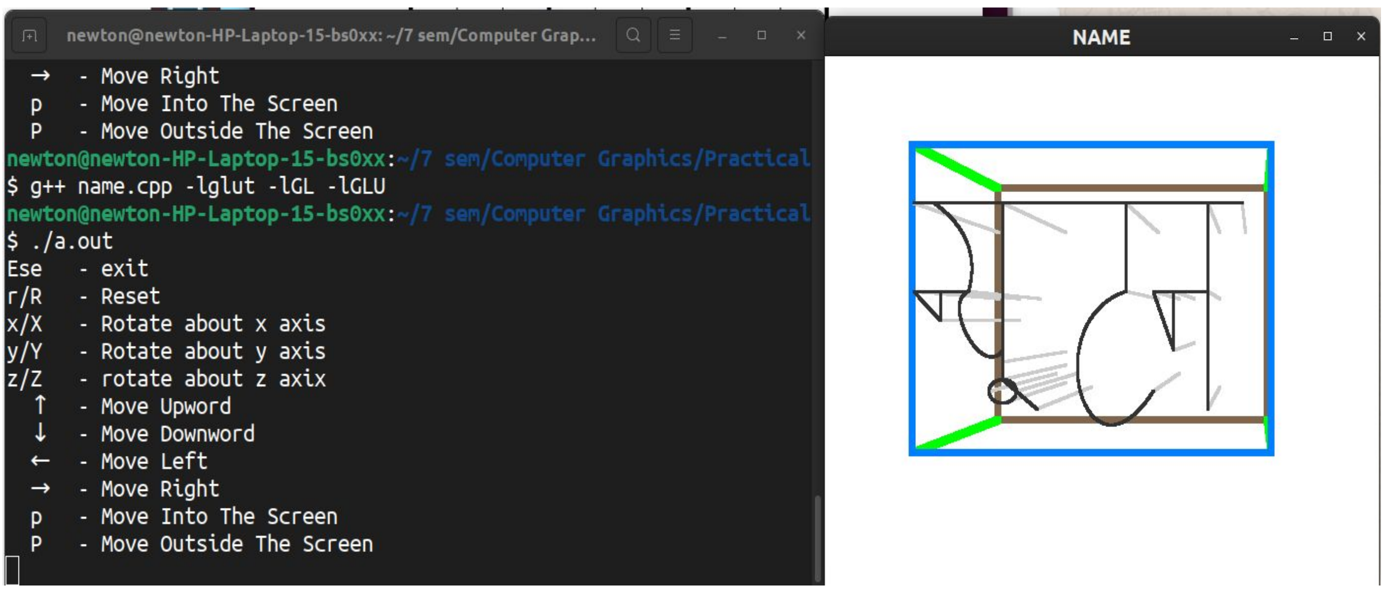
```
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();
glBegin(GL_POINTS);
x1=20 ,y1=80 ,z1=0,x2=20 ,y2=80 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=20 ,y1=65 ,z1=0,x2=20 ,y2=65 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
float px=x1*(1-t)+x2*(t);
float py=y1*(1-t)+y2*(t);
float pz=z1*(1-t)+z2*(t);
        glVertex3f(px,py,pz);// -------
}
glEnd();

glBegin(GL_POINTS);
x1=15 ,y1=65 ,z1=0,x2=15 ,y2=65 ,z2=-15;
for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
   float pz=z1*(1-t)+z2*(t);
           glVertex3f(px,py,pz);// -------
   }
   glEnd();

   glBegin(GL_POINTS);
   x1=12 ,y1=65 ,z1=0,x2=12 ,y2=65 ,z2=-15;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
   float pz=z1*(1-t)+z2*(t);
           glVertex3f(px,py,pz);// -------
   }
   glEnd();
   glBegin(GL_POINTS);
   x1=15 ,y1=55 ,z1=0,x2=15 ,y2=55 ,z2=-15;
   for(float t=0;t<=1;t+=0.001){
   float px=x1*(1-t)+x2*(t);
   float py=y1*(1-t)+y2*(t);
   float pz=z1*(1-t)+z2*(t);
           glVertex3f(px,py,pz);// -------
   }
   glEnd();
}
void display()
{
   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();

   glTranslatef(-20.0+xx,-30.0+yy,-60.0+zz);

   glRotatef(Xangle,1.0,0.0,0.0);

   glRotatef(Yangle,0.0,1.0,0.0);

   glRotatef(Zangle,0.0,0.0,1.0);

   glColor3f(1,0,0);

   name();

   glutSwapBuffers();

}

void keyInput(unsigned char key,int x,int y)

{

   switch(key)

   {

       case 'R':

       case 'r':

           Xangle=0,Yangle=0,Zangle=0,xx=0,yy=0,zz=0,angle=0;

           glutPostRedisplay();

           break;

       case 27:

           exit(0);

           break;

       case 'x':

           Xangle+=5.0;

           if(Xangle>360.0) Xangle-=360.0;

           glutPostRedisplay();

           break;

       case 'X':

           Xangle-=5.0;

           if(Xangle<0.0) Xangle+=360.0;
```

```
        glutPostRedisplay();

        break;

    case 'y':

        Yangle+=5.0;

        if(Yangle>360.0) Yangle-=360.0;

        glutPostRedisplay();

        break;

    case 'Y':

        Yangle-=5.0;

        if(Yangle<0.0) Yangle+=360.0;

        glutPostRedisplay();

        break;

    case 'z':

        Zangle+=5.0;

        if(Zangle>360.0) Zangle-=360.0;

        glutPostRedisplay();

        break;

    case 'Z':

        Zangle-=5.0;

        if(Zangle<0.0) Zangle+=360.0;

        glutPostRedisplay();

        break;

    case 'p':

        zz=zz-1;

        if(zz<=-45) zz=-44;

        glutPostRedisplay();

        break;

    case 'P':

        zz=zz+1;
```

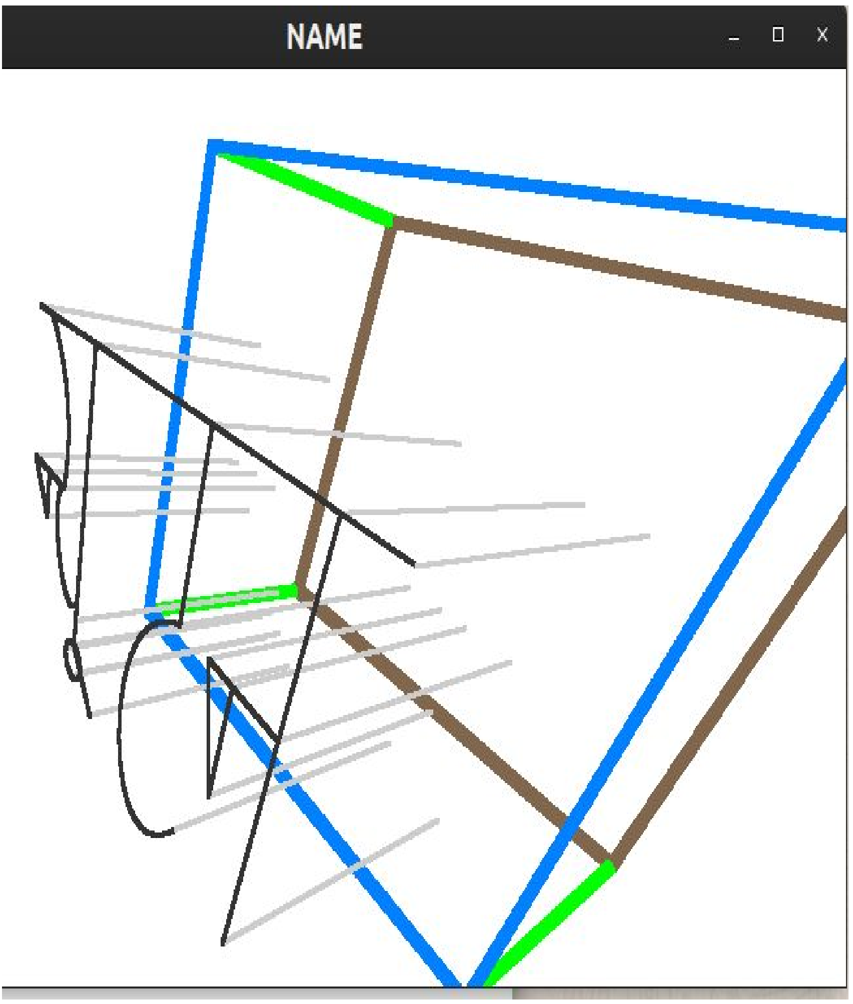
```
            if(zz>=60) zz=59;
            glutPostRedisplay();
            break;
        default:
            break;
    }
}
void specialKeyInput(int key,int x,int y)
{
    switch(key)
    {
        case GLUT_KEY_UP:
                yy=yy+1;
                glutPostRedisplay();
                break;
        case GLUT_KEY_DOWN:
                yy=yy-1;
                glutPostRedisplay();
                break;
        case GLUT_KEY_LEFT:
                xx=xx-1;
                glutPostRedisplay();
                break;
        case GLUT_KEY_RIGHT:
                xx=xx+1;
                glutPostRedisplay();
                break;
    }
}
```

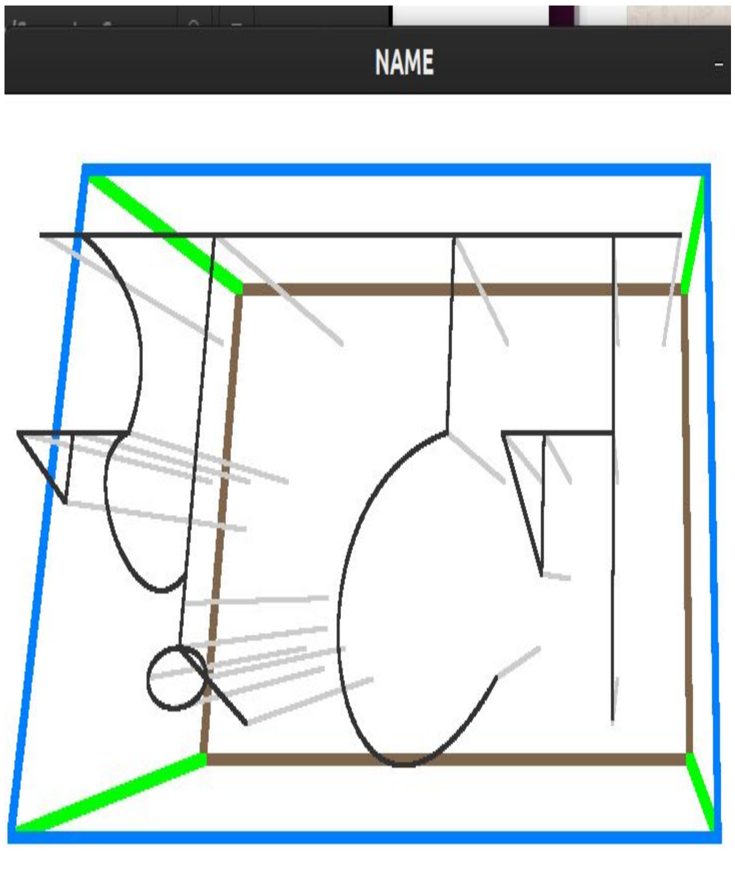
```
void PrintFun()
{
  cout<<"Ese    - exit\nr/R    - Reset\nx/X    - Rotate about x axis\n
      y/Y    - Rotate about y axis\nz/Z    - rotate about z axix\n
        ↑    - Move Upword\n  ↓    - Move Downword\n  ←    - Move Left\n
        →    - Move Right\n  p    - Move Into The Screen\n
        P    - Move Outside The Screen\n"  ;
}
int main(int argc, char** argv)
{
  PrintFun() ;
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
   glutInitWindowPosition(50,50);
   glutInitWindowSize(800,500);
   glutCreateWindow("NAME");
   myinit();
   glutDisplayFunc(display);
   glutReshapeFunc(resize);
   glutKeyboardFunc(keyInput);
   glutSpecialFunc(specialKeyInput);
   glutMainLoop();
   return 0;
}
```

Output



```
  →   - Move Right
  p   - Move Into The Screen
  P   - Move Outside The Screen
newton@newton-HP-Laptop-15-bs0xx:~/7 sem/Computer Graphics/Practical
$ g++ name.cpp -lglut -lGL -lGLU
newton@newton-HP-Laptop-15-bs0xx:~/7 sem/Computer Graphics/Practical
$ ./a.out
Ese   - exit
r/R   - Reset
x/X   - Rotate about x axis
y/Y   - Rotate about y axis
z/Z   - rotate about z axix
  ↑   - Move Upword
  ↓   - Move Downword
  ←   - Move Left
  →   - Move Right
  p   - Move Into The Screen
  P   - Move Outside The Screen
```





Objective 2

Make a 3D WindMill with animated (rotaring) wings using Opengl.

Methodology

1. ***windmill()***

   Create a function to construct a windmill with animated (rotating) wings.

   Rotation of wings is achieved by an increment of variable *angle* by 10 degrees when bool variable *isAnimate* is TRUE.

2. ***display()***

   Use to control the orientation of the windmill. Call the function windmill()

3. ***animate()***

   The function which controls the animation of wings

4. ***resize()***

   Function to resize the viewport and projection matrix according to input width and height. Called by GLUT.

5. ***keyInput()***

   Consist of function activates when specific keys pressed, changes in the angle of the object or close the window or activate the wing.

6. ***specialKeyInput()***

   Allows moving the object in the 3D coordinate system using arrow keys to move left, right, left & right, and p/P to move in and out of the screen.

7. ***PrintFun()***

   A menu function to tell the user to tell the functionality to the user

8. ***main()***

   The main function will call all the functions and is in the loop.

CODE

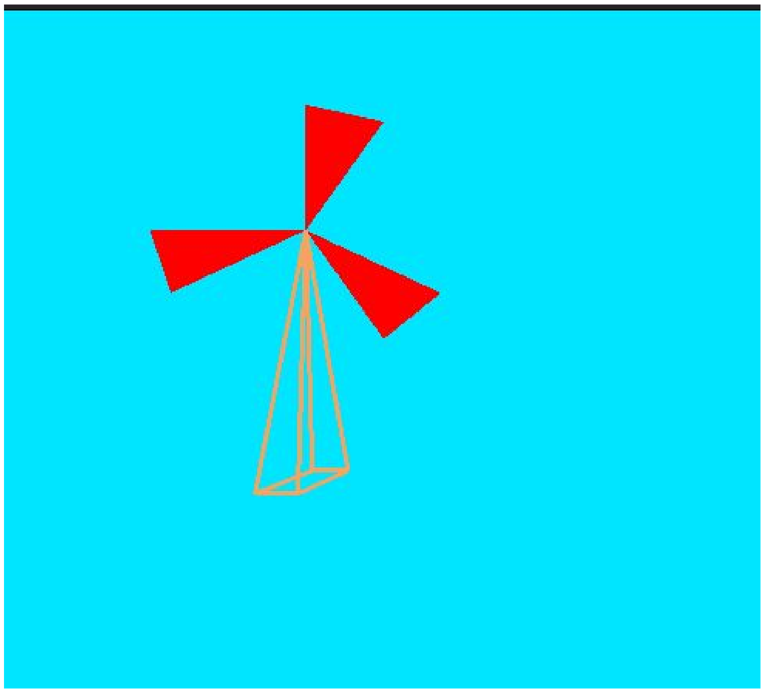
```
#include<iostream>

#include<math.h>

#include<GL/glut.h>

#define PI 3.141592653589793

#define mode GL_LINE_LOOP

using namespace std;


float Xangle=0,Yangle=0,Zangle=0;

float xx=0,yy=0,zz=0;

float angle=0;


GLfloat fSize[2] ;

void windmill();

static bool isAnimate=0;

static int animatePeriod=100;


void animate(int value)

{

    if(isAnimate)

    {

        angle+=10;
```

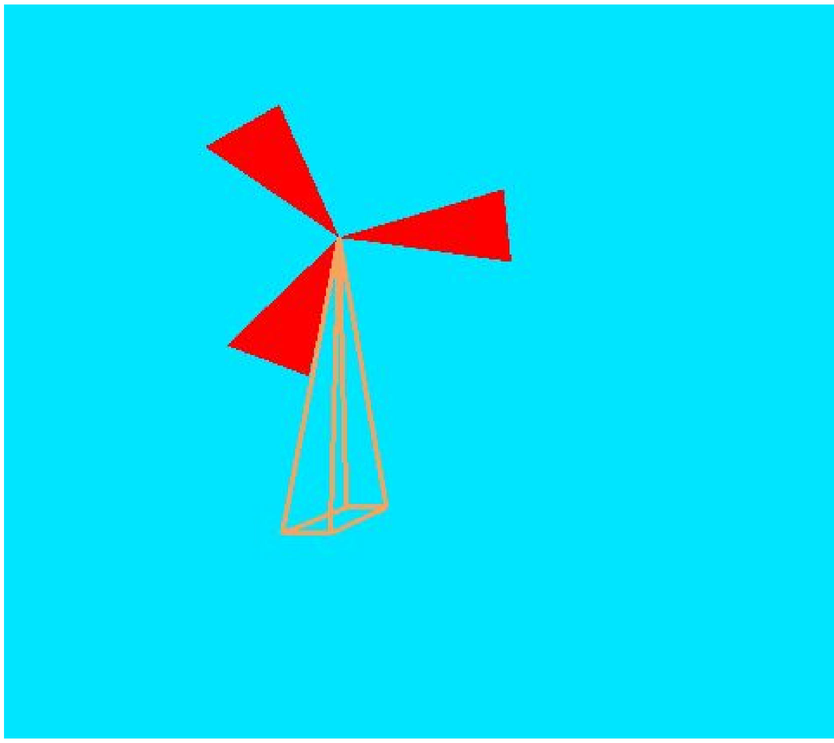
```
       if(angle>=360)  angle-=360;


       glutPostRedisplay();

       glutTimerFunc(animatePeriod,animate,1);

   }

}


void myinit()

{

   glClearColor(0.0f, 0.9f, 1.0f,1.0f);

   glEnable(GL_DEPTH_TEST);

}


void resize(int w,int h)

{

   glViewport(0,0,w,h);

   glMatrixMode(GL_PROJECTION);

   glLoadIdentity();

   glFrustum(-7.0,2.0,-5.0,5.0,5.0,200.0);

   glMatrixMode(GL_MODELVIEW);

}

void display()

{

   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

   glLoadIdentity();

   glTranslatef(-20.0+xx,-20.0+yy,-50.0+zz);

   glRotatef(Xangle,1.0,0.0,0.0);

   glRotatef(Yangle,0.0,1.0,0.0);
   glRotatef(Zangle,0.0,0.0,1.0);

   windmill();

   glutSwapBuffers();

}

void windmill()

{

   glGetFloatv(GL_LINE_WIDTH_RANGE,fSize) ;

   glLineWidth(fSize[0]+2.0f);

   glColor3f(0.956862,0.643137,0.376470);

   glBegin(mode);

   glVertex3f(-20,0,0);

   glVertex3f(-15,0,0);

   glVertex3f(-17.5,40,-5);

   glEnd();

   glBegin(mode);

   glVertex3f(-20,0,0);

   glVertex3f(-20,0,-10);

   glVertex3f(-17.5,40,-5);

   glEnd();

   glBegin(mode);

   glVertex3f(-15,0,0);

   glVertex3f(-15,0,-10);

   glVertex3f(-17.5,40,-5);

   glEnd();

   glBegin(mode);

   glVertex3f(-20,0,-10);

   glVertex3f(-15,0,-10);

   glVertex3f(-17.5,40,-5);
   glEnd();

   glBegin(GL_POLYGON);

   glVertex3f(-17.5,40,-5);

   glVertex3f(-17.5,40,-4);

   glEnd();

   glLineWidth(fSize[0]+4.0f);

   glPointSize(7);

   float radius=20;

   glColor3f(1,0,0);

   for(float i=1;i<=3;i+=1)

   {

       float ang=angle+120*i;

       glBegin(GL_TRIANGLE_STRIP);

       glVertex3f(-17.5,40,-5);

      glVertex3f(-17.5+cos(ang*PI/180.0f)*radius

     ,40+sin(ang*PI/180.0f)*radius

     ,-5);

     glVertex3f(-17.5+cos((ang+30)*PI/180.0f)*radius,

     40+sin((ang+30)*PI/180.0f)*radius

     ,-5);

   glEnd();

   }

}


void keyInput(unsigned char key,int x,int y)

{

   switch(key)

   {
    case 27:

        exit(0);

        break;

    case 'R':

    case 'r':

        Xangle=0,Yangle=0,Zangle=0;

        xx=0,yy=0,zz=0;

        angle=0;

        glutPostRedisplay();

        break;

    case ' ':

        if(isAnimate) isAnimate=0;

        else{

            isAnimate=1;

            animate(1);

        }

        break;

    case 'x':

        Xangle+=5.0;

        if(Xangle>360.0) Xangle-=360.0;

        glutPostRedisplay();

        break;

    case 'X':

        Xangle-=5.0;

        if(Xangle<0.0) Xangle+=360.0;

        glutPostRedisplay();

        break;

    case 'y':
        Yangle+=5.0;

        if(Yangle>360.0) Yangle-=360.0;

        glutPostRedisplay();

        break;

    case 'Y':

        Yangle-=5.0;

        if(Yangle<0.0) Yangle+=360.0;

        glutPostRedisplay();

        break;

    case 'z':

        Zangle+=5.0;

        if(Zangle>360.0) Zangle-=360.0;

        glutPostRedisplay();

        break;

    case 'Z':

        Zangle-=5.0;

        if(Zangle<0.0) Zangle+=360.0;

        glutPostRedisplay();

        break;

    case 'p':

        zz=zz-1;

        if(zz<=-45) zz=-44;

        glutPostRedisplay();

        break;

    case 'P':

        zz=zz+1;

        if(zz>=60) zz=59;

        glutPostRedisplay();
```

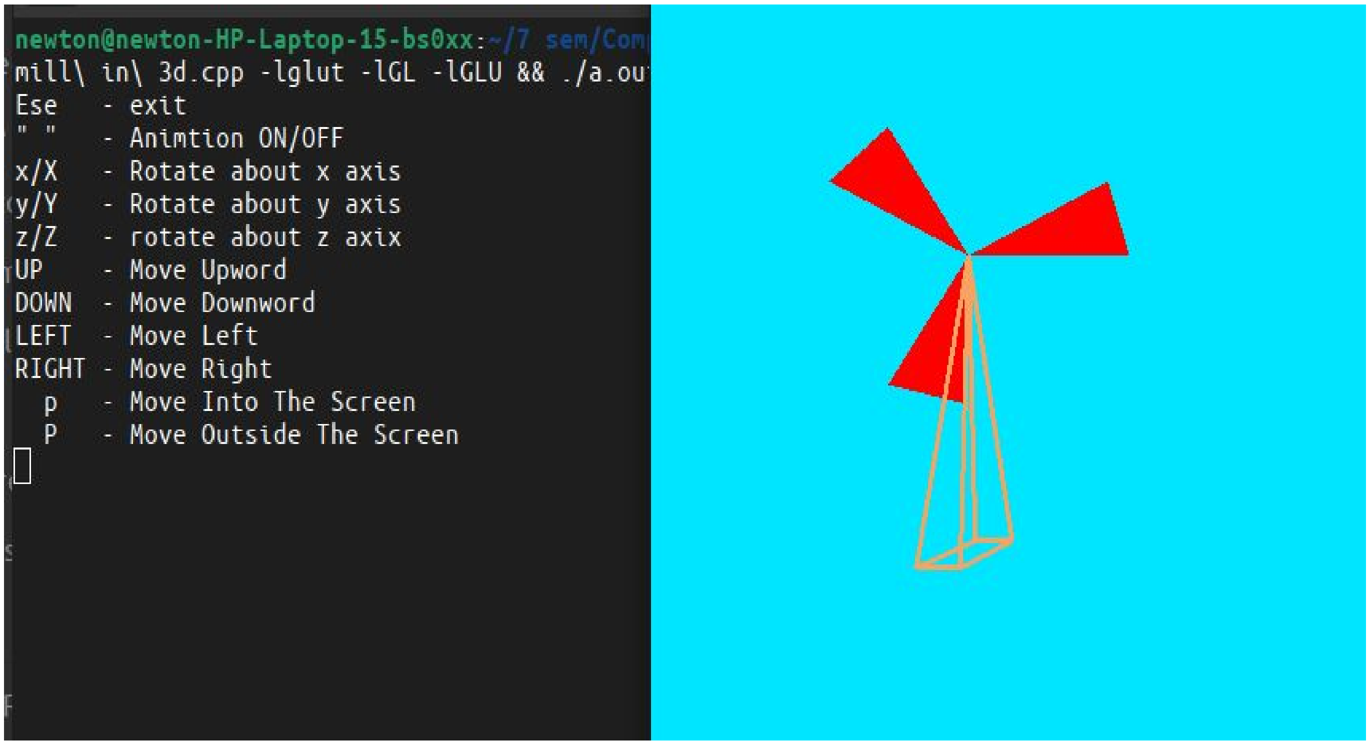
```
            break;
        default:
            break;
    }
}
void specialKeyInput(int key,int x,int y)
{
   switch(key)
   {
       case GLUT_KEY_UP:
               yy=yy+1;
               glutPostRedisplay();
               break;
       case GLUT_KEY_DOWN:
               yy=yy-1;
               glutPostRedisplay();
               break;
       case GLUT_KEY_LEFT:
               xx=xx-1;
               glutPostRedisplay();
               break;
       case GLUT_KEY_RIGHT:
               xx=xx+1;
               glutPostRedisplay();
               break;
   }
}
void PrintFun()
```

```
{
  cout<<"Ese    - exit\n\" \"    - Animtion ON/OFF\n
x/X    - Rotate about x axis\ny/Y    - Rotate about y axis\n
z/Z    - rotate about z axix\nUP     - Move Upword\n
DOWN   - Move Downword\nLEFT  - Move Left\nRIGHT - Move Right\n
  p    - Move Into The Screen\n  P    - Move Outside The Screen\n"  ;
}
int main(int argc, char** argv)
{
  PrintFun() ;
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
   glutInitWindowPosition(50,50);
   glutInitWindowSize(500,500);
   glutCreateWindow("Wind Mill");
   myinit();
   glutDisplayFunc(display);
   glutReshapeFunc(resize);
   glutKeyboardFunc(keyInput);
   glutSpecialFunc(specialKeyInput);
   glutMainLoop();
   return 0;
}
```

OUTPUT







```
newton@newton-HP-Laptop-15-bs0xx:~/7 sem/Com
mill\ in\ 3d.cpp -lglut -lGL -lGLU && ./a.ou
Ese    - exit
" "    - Animtion ON/OFF
x/X    - Rotate about x axis
y/Y    - Rotate about y axis
z/Z    - rotate about z axix
UP     - Move Upword
DOWN   - Move Downword
LEFT   - Move Left
RIGHT  - Move Right
  p    - Move Into The Screen
  P    - Move Outside The Screen
```

OBJECTIVE 3

The project aims to create a Clock - Tower using Opengl.

Methodology

The program divided into four parts, which are as follows:-

1. ***Design the Clock - Tower***

   The clock designed in modules each are created by shifting and rotating the axis which are

   a. Tower()
   b. Stairs()
   c. Land()
   d. Clock1()
   e. Clock2()
   f. Clock3()
   g. Clock4()

   All four clocks are in Syncronosty and set at four adjacent sides.

2. ***Lighting***

   A light source fixed to give the shadow effect.

3. ***Control***

   Control different parameters that change the observation

   a. My mouse()

      The scene can be clicked and drag using the mouse to change the view.

   b. mymotion()

      Change scale when the wheel on the mouse is revolving.

c. mykey()

Rotate, Scale or Reset the state of Clock Tower

4. ***Execution***

a. PrintFun()

Print the menu.

b. main()

Call the function.

CODE

```
#include <GL/glut.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

#include<iostream>

#define XFORM_NONE    0

#define XFORM_ROTATE  1

#define XFORM_SCALE 2

using namespace std;

struct tm *newtime;

time_t ltime;

float x_angle = 0.0, y_angle = 0.0, scale_size = 1;

int xform_mode = 0, ani = 0, timer = 75, release_x, release_y, press_x,
press_y;


void initLighting() {

   GLfloat lightColor0[] = { 0.6f, 0.6f, 0.6f, 1.0f };

   GLfloat lightPos0[] = { -0.5f, 0.8f, 1.0f, 0.0f };
   GLfloat ambientColor[] = { 0.4f, 0.4f, 0.4f, 1.0f };

   glEnable(GL_DEPTH_TEST);

   glEnable(GL_COLOR_MATERIAL);

   glEnable(GL_LIGHTING);

   glEnable(GL_LIGHT0);

   glEnable(GL_NORMALIZE);

   glShadeModel(GL_SMOOTH);

   glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);

   glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);

   glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);

}

void Tower()

{

   glPushMatrix();

   glRotatef(45, 0, 1, 0);

   glTranslatef(0, 22.7, 0);

   glScalef(4.5, 4, 4.5);

   glColor3f(0.82, 0, 0.05);

   glutSolidOctahedron();

   glPopMatrix();

   //Tower

   glPushMatrix();

   glTranslatef(0, 11, 0);

   glScalef(5, 25, 5);

   glColor3f(0.65, 0.32, 0.05);

   glutSolidCube(1.0);

   glPopMatrix();

   //Cuts

   glPushMatrix();
   glColor3f(0.27, 0.13, 0);
   glTranslatef(0, 15, 0);
   glScaled(5.5, 0.5, 5.5);
   glutSolidCube(1);
   glPopMatrix();
   glPushMatrix();
   glColor3f(0.27, 0.13, 0);
   glTranslatef(0, 20, 0);
   glScaled(5.5, 0.5, 5.5);
   glutSolidCube(1);
   glPopMatrix();
   //Door
   glPushMatrix();
   glColor3f(0, 0, 0);
   glTranslatef(-0., 1.0, 2.5);
   glScalef(20, 24, 0.1);
   glutSolidCube(0.1);
   glPopMatrix();
   glPushMatrix();
   glColor3f(0, 0, 0);
   glTranslatef(-0., 1.8, 2.5);
   glScalef(10.1, 24, 0.01);
   glutSolidSphere(0.1, 222, 22);
   glPopMatrix();
}
//Stairs
void Stairs() {
   glPushMatrix();
       glColor3f(0.5, 0.5, 0.5);
   glPushMatrix();
       glTranslatef(0, -0.4, 0);
       glScaled(10, 0.5, 10);
       glutSolidCube(1);
   glPopMatrix();
   glPushMatrix();
       glTranslatef(0, -0.9, 0);
       glScaled(15, 0.5, 15);
       glutSolidCube(1);
   glPopMatrix();
   glPushMatrix();
       glTranslatef(0, -1.3, 0);
       glScaled(20, 0.5, 20);
       glutSolidCube(1);
   glPopMatrix();
   glPopMatrix();
}
//Land
void Land() {
   glPushMatrix();
   glTranslatef(0, -1.5, 0);
   glScalef(500, 0.1, 500);
   glColor3f(0, 0.56, 0);
   glutSolidCube(1.0);
   glPopMatrix();
}
//Clock
void Clock1()
{
   glPushMatrix();

   glScaled(0.2, 0.2, 0.2);

   glTranslatef(0, 88, 13.55);

   glRotatef(180, 1.0, 0.0, 0.0);

   //Clock Frame

   glPushMatrix();

   glColor3f(0, 0, 0);

   glTranslatef(0.0, 0.0, 0.0);

   glScaled(1, 1, 0.01);

   glutSolidTorus(0.5, 7.5, 22, 55);

   glPopMatrix();

   //Clock Background

   glPushMatrix();

   glTranslatef(0, 0, 1.0);

   glColor3f(1.0, 1.0, 1.0);

   glScaled(7, 7, 0.01);

   glutSolidSphere(1, 22, 22);

   glPopMatrix();

   glPushMatrix();// Draw hour hand

   glColor3f(1.0, 0.5, 0.5);

   glRotatef((360 / 12) * newtime->tm_hour + (360 / 60) * (60 /
(newtime->tm_min + 1)), 0.0, 0.0, 1.0);

   glScaled(0.6, 4, 0.2);

   glTranslated(0, -0.45, 0);

   glutSolidCube(1);

   glPopMatrix();

   glPushMatrix();// Draw minute hand

   glColor3f(1.0, 0.5, 1.0);

   glRotatef((360 / 60) * newtime->tm_min, 0.0, 0.0, 1.0);
```

```
glScaled(0.4, 5, 0.2);

glTranslated(0, -0.45, 0);

glutSolidCube(1);

glPopMatrix();

glPushMatrix();// Draw second hand

glColor3f(1.0, 0.0, 0.5);

glRotatef((360 / 60) *newtime->tm_sec, 0.0, 0.0, 1.0);

glScaled(6, 0.2, 0.2);

glTranslated(-0.45, 0, 0);

glutSolidCube(1);

glPopMatrix();

for (int hour_ticks = 0; hour_ticks < 12; hour_ticks++)

{

    glPushMatrix();// Draw next arm axis.

    glColor3f(0.0, 0, 0); // give it a color

    glTranslatef(0.0, 0.0, 0.0);

    glRotatef((360 / 12) * hour_ticks, 0.0, 0.0, 1.0);

    glTranslatef(6.0, 0.0, 0.0);

    glutSolidCube(1.0);

    glPopMatrix();

}

for (int sec_ticks = 0; sec_ticks < 60; sec_ticks++)

{

    glPushMatrix();

    glTranslatef(0.0, 0.0, 0.0);

    glRotatef((360 / 60) * sec_ticks, 0.0, 0.0, 1.0);

    glTranslatef(6.0, 0.0, 0.0);

    glutSolidCube(0.25);

    glPopMatrix();
```

```
   }
   glPopMatrix();
}


void Clock2()
{
   glPushMatrix();
   glScaled(0.2, 0.2, 0.2);
   //glTranslatef(0, 88, 13.55);
   glTranslatef(13.55,88,0) ;
   glRotatef(-90,  0.0,1.0, 0.0);
   glRotatef(180,0,0,1) ;
   //Clock Frame
   glPushMatrix();
   glColor3f(0, 0, 0);
   glTranslatef(0.0, 0.0, 0.0);
   glScaled(1, 1, 0.01);
   glutSolidTorus(0.5, 7.5, 22, 55);
   glPopMatrix();
   //Clock Background
   glPushMatrix();
   glTranslatef(0, 0, 1.0);
   glColor3f(1.0, 1.0, 1.0);
   glScaled(7, 7, 0.01);
   glutSolidSphere(1, 22, 22);
   glPopMatrix();
   glPushMatrix();// Draw hour hand
   glColor3f(1.0, 0.5, 0.5);
   glRotatef((360 / 12) * newtime->tm_hour + (360 / 60) * (60 /
(newtime->tm_min + 1)), 0.0, 0.0, 1.0);

   glScaled(0.6, 4, 0.2);

   glTranslated(0, -0.45, 0);

   glutSolidCube(1);

   glPopMatrix();

   glPushMatrix();// Draw minute hand

   glColor3f(1.0, 0.5, 1.0);

   glRotatef((360 / 60) * newtime->tm_min, 0.0, 0.0, 1.0);

   glScaled(0.4, 5, 0.2);

   glTranslated(0, -0.45, 0);

   glutSolidCube(1);

   glPopMatrix();

   glPushMatrix();// Draw second hand

   glColor3f(1.0, 0.0, 0.5);

   glRotatef((360 / 60) *newtime->tm_sec, 0.0, 0.0, 1.0);

   glScaled(6, 0.2, 0.2);

   glTranslated(-0.45, 0, 0);

   glutSolidCube(1);

   glPopMatrix();


   for (int hour_ticks = 0; hour_ticks < 12; hour_ticks++)

   {

       glPushMatrix();// Draw next arm axis.

       glColor3f(0.0, 0, 0); // give it a color

       glTranslatef(0.0, 0.0, 0.0);

       glRotatef((360 / 12) * hour_ticks, 0.0, 0.0, 1.0);

       glTranslatef(6.0, 0.0, 0.0);

       glutSolidCube(1.0);
```

```
        glPopMatrix();
    }
    for (int sec_ticks = 0; sec_ticks < 60; sec_ticks++)
    {
        glPushMatrix();
        glTranslatef(0.0, 0.0, 0.0);
        glRotatef((360 / 60) * sec_ticks, 0.0, 0.0, 1.0);
        glTranslatef(6.0, 0.0, 0.0);
        glutSolidCube(0.25);
        glPopMatrix();
    }
    glPopMatrix();
}

void Clock3()
{
    glPushMatrix();
    glScaled(0.2, 0.2, 0.2);
    //glTranslatef(0, 88, 13.55);
    glTranslatef(0, 88, -13.55);
    glRotatef(180, 0.0, 0, 1.0);
    //Clock Frame
    glPushMatrix();
    glColor3f(0, 0, 0);
    glTranslatef(0.0, 0.0, 0.0);
    glScaled(1, 1, 0.01);
    glutSolidTorus(0.5, 7.5, 22, 55);
    glPopMatrix();
```

```
  //Clock Background
  glPushMatrix();
  glTranslatef(0, 0, 1.0);
  glColor3f(1.0, 1.0, 1.0);
  glScaled(7, 7, 0.01);
  glutSolidSphere(1, 22, 22);
  glPopMatrix();

  glPushMatrix();// Draw hour hand
  glColor3f(1.0, 0.5, 0.5);
  glRotatef((360 / 12) * newtime->tm_hour + (360 / 60) * (60 /
(newtime->tm_min + 1)), 0.0, 0.0, 1.0);
  glScaled(0.6, 4, 0.2);
  glTranslated(0, -0.45, 0);
  glutSolidCube(1);
  glPopMatrix();
  glPushMatrix();// Draw minute hand
  glColor3f(1.0, 0.5, 1.0);
  glRotatef((360 / 60) * newtime->tm_min, 0.0, 0.0, 1.0);
  glScaled(0.4, 5, 0.2);
  glTranslated(0, -0.45, 0);
  glutSolidCube(1);
  glPopMatrix();

  glPushMatrix();// Draw second hand
  glColor3f(1.0, 0.0, 0.5);
  glRotatef((360 / 60) *newtime->tm_sec, 0.0, 0.0, 1.0);
  glScaled(6, 0.2, 0.2);
```

```
   glTranslated(-0.45, 0, 0);

   glutSolidCube(1);

   glPopMatrix();

   for (int hour_ticks = 0; hour_ticks < 12; hour_ticks++)

   {

       glPushMatrix();// Draw next arm axis.

       glColor3f(0.0, 0, 0); // give it a color

       glTranslatef(0.0, 0.0, 0.0);

       glRotatef((360 / 12) * hour_ticks, 0.0, 0.0, 1.0);

       glTranslatef(6.0, 0.0, 0.0);

       glutSolidCube(1.0);

       glPopMatrix();

   }

   for (int sec_ticks = 0; sec_ticks < 60; sec_ticks++)

   {

       glPushMatrix();

       glTranslatef(0.0, 0.0, 0.0);

       glRotatef((360 / 60) * sec_ticks, 0.0, 0.0, 1.0);

       glTranslatef(6.0, 0.0, 0.0);

       glutSolidCube(0.25);

       glPopMatrix();

   }

   glPopMatrix();

}

void Clock4()

{

   glPushMatrix();

   glScaled(0.2, 0.2, 0.2);

   glTranslatef(-13.55, 88, 0);
   glRotatef(90, 0, 1.0, 0.0);

   glRotatef(180,0,0,1);

   //Clock Frame

   glPushMatrix();

   glColor3f(0, 0, 0);

   glTranslatef(0.0, 0.0, 0.0);

   glScaled(1, 1, 0.01);

   glutSolidTorus(0.5, 7.5, 22, 55);

   glPopMatrix();

   //Clock Background

   glPushMatrix();

   glTranslatef(0, 0, 1.0);

   glColor3f(1.0, 1.0, 1.0);

   glScaled(7, 7, 0.01);

   glutSolidSphere(1, 22, 22);

   glPopMatrix();

   glPushMatrix();// Draw hour hand

   glColor3f(1.0, 0.5, 0.5);

   glRotatef((360 / 12) * newtime->tm_hour + (360 / 60) * (60 /
(newtime->tm_min + 1)), 0.0, 0.0, 1.0);

   glScaled(0.6, 4, 0.2);

   glTranslated(0, -0.45, 0);

   glutSolidCube(1);

   glPopMatrix();

   glPushMatrix();// Draw minute hand

   glColor3f(1.0, 0.5, 1.0);

   glRotatef((360 / 60) * newtime->tm_min, 0.0, 0.0, 1.0);

   glScaled(0.4, 5, 0.2);

   glTranslated(0, -0.45, 0);
```

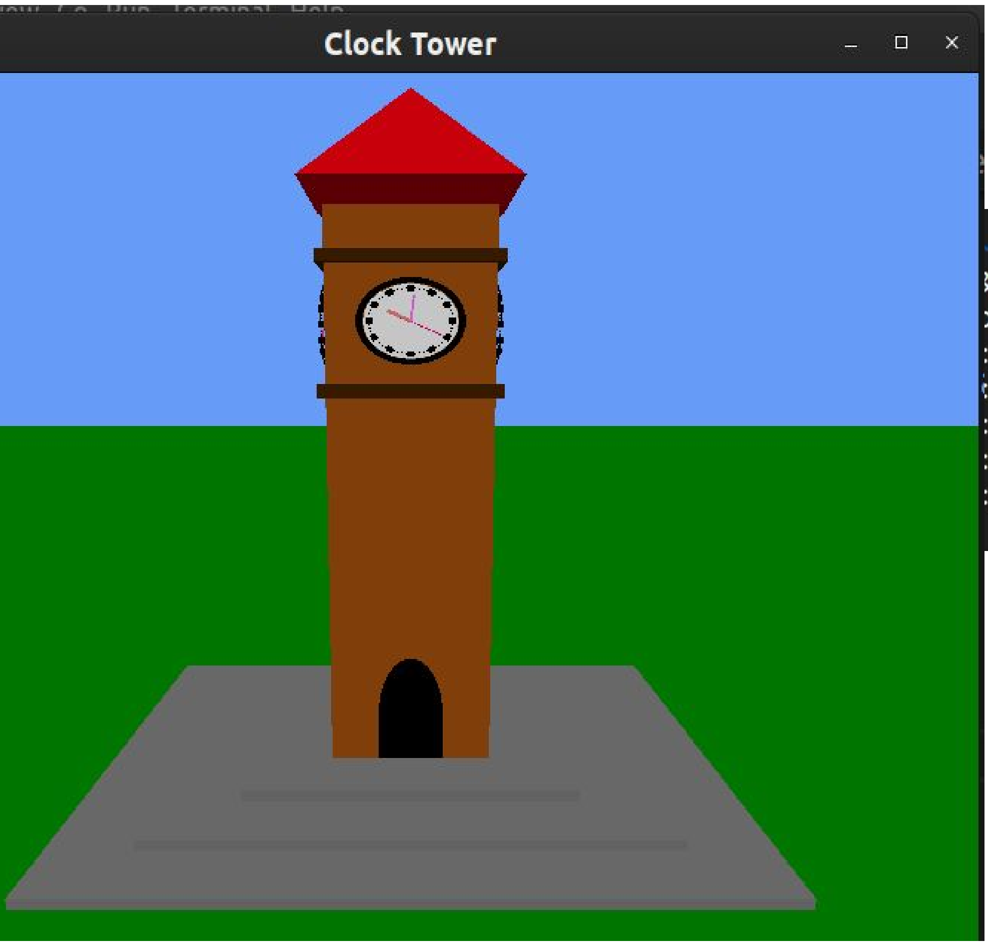
```
glutSolidCube(1);

glPopMatrix();

glPushMatrix();// Draw second hand

glColor3f(1.0, 0.0, 0.5);

glRotatef((360 / 60) *newtime->tm_sec, 0.0, 0.0, 1.0);

glScaled(6, 0.2, 0.2);

glTranslated(-0.45, 0, 0);

glutSolidCube(1);

glPopMatrix();

for (int hour_ticks = 0; hour_ticks < 12; hour_ticks++)

{

    glPushMatrix();// Draw next arm axis.

    glColor3f(0.0, 0, 0); // give it a color

    glTranslatef(0.0, 0.0, 0.0);

    glRotatef((360 / 12) * hour_ticks, 0.0, 0.0, 1.0);

    glTranslatef(6.0, 0.0, 0.0);

    glutSolidCube(1.0);

    glPopMatrix();

}

for (int sec_ticks = 0; sec_ticks < 60; sec_ticks++)

{

    glPushMatrix();

    glTranslatef(0.0, 0.0, 0.0);

    glRotatef((360 / 60) * sec_ticks, 0.0, 0.0, 1.0);

    glTranslatef(6.0, 0.0, 0.0);

    glutSolidCube(0.25);

    glPopMatrix();

}

glPopMatrix();
```

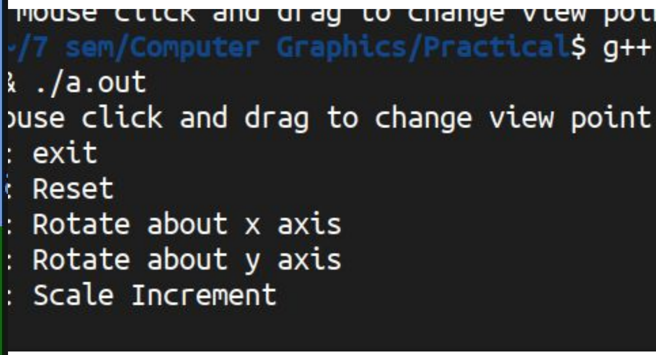
```
}
void display(){
   time(&ltime); // Get time
   newtime = localtime(&ltime); // Convert to local time
   glClearColor(0.4, 0.61, 0.97, 1);
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(60, 1, 1, 900);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   gluLookAt(0.0, 15.0, 32.0, 0.0, 10.0, 0.0, 0.0, 30.0, 0.0);
   glRotatef(x_angle, 0, 1, 0);
   glRotatef(y_angle, 1, 0, 0);
   glScalef(scale_size, scale_size, scale_size);
   Tower()  ;
   Clock1() ;
   Clock2() ;
   Clock3() ;
   Clock4() ;
   Stairs() ;
   Land()   ;
   glutSwapBuffers();
}
void mymouse(int button, int state, int x, int y)
{
   if (state == GLUT_DOWN) {
       press_x = x; press_y = y;
       if (button == GLUT_LEFT_BUTTON)
```

```
            xform_mode = XFORM_ROTATE;
        else if (button == GLUT_RIGHT_BUTTON)
            xform_mode = XFORM_SCALE;
    }
    else if (state == GLUT_UP) {
        xform_mode = XFORM_NONE;
    }
}
void myidle(int val) {
    int angle = 1;
    if (ani == 0) glutTimerFunc(timer, myidle, 0);
    angle = (angle + 10) % 360;
    glutPostRedisplay();
}
void mymotion(int x, int y){
    if (xform_mode == XFORM_ROTATE) {
        x_angle += (x - press_x) / 10.0;
        if (x_angle > 180) x_angle -= 360;
        else if (x_angle < -180) x_angle += 360;
        press_x = x;
        y_angle += (y - press_y) / 10.0;
        if (y_angle > 180) y_angle -= 360;
        else if (y_angle < -180) y_angle += 360;
        press_y = y;
    }
    else if (xform_mode == XFORM_SCALE) {
        float old_size = scale_size;
        scale_size *= (1 + (y - press_y) / 600.0);
        if (scale_size < 0) scale_size = old_size;
        press_y = y;
    }
    glutPostRedisplay();
}
void mykey(unsigned char key, int x, int y)
{
    if (key == '-') {
        scale_size--;
    }
    if (key == '+') {
        scale_size++;
    }
    if (key == 'd') {
        x_angle--;
    }
    if (key == 'a') {
        x_angle++;
    }
    if (key == 's') {
        y_angle--;
    }
    if (key == 'w') {
        y_angle++;
    }
    if (key == 27)
    {
        exit(0);
    }
    if (key == ' ')
```

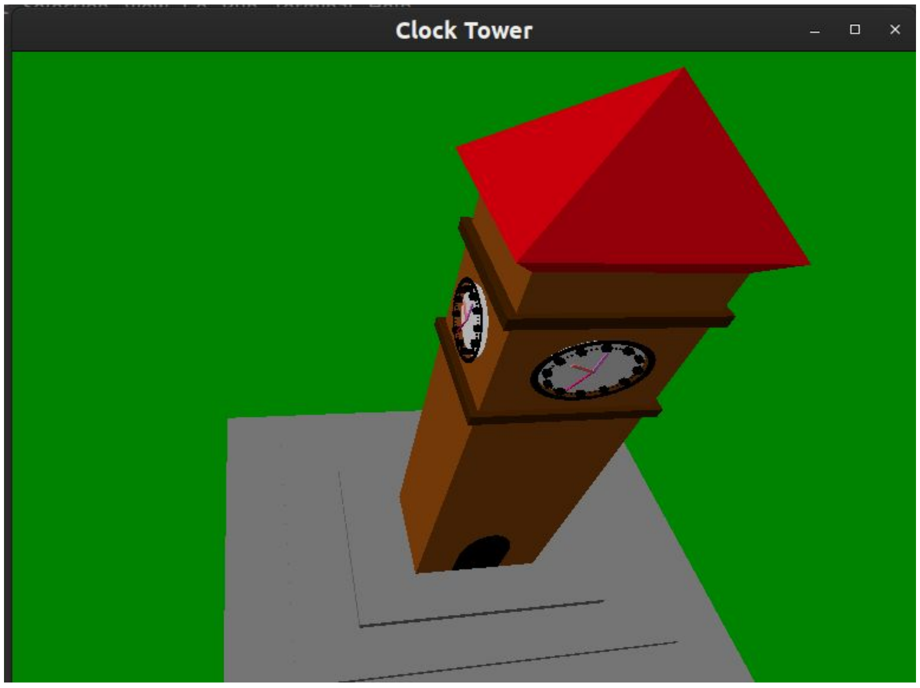
```
       {
           x_angle = 0.0, y_angle = 0.0, scale_size = 1 ;
       }
}
void PrintFun()
{
  cout<<"\nUse mouse click and drag to change view point\nEse  : exit\n\"
\"  : Reset\na/d  : Rotate about x axis\nw/s  : Rotate about y axis\n  +
: Scale Increment\n  -  : Scale Decrement"  ;
}
int main(int argc, char** argv)
{
   PrintFun();
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
   glutInitWindowSize(800, 600);
   glutCreateWindow("Clock Tower");
   glutDisplayFunc(display);
   glutMouseFunc(mymouse);
   glutMotionFunc(mymotion);
   initLighting();
   glutKeyboardFunc(mykey);
   glutTimerFunc(timer, myidle, 0);
   glutMainLoop();
}
```

OUTPUT





```
 Mouse click and drag to change view poi
~/7 sem/Computer Graphics/Practical$ g++
& ./a.out
ouse click and drag to change view point
: exit
: Reset
: Rotate about x axis
: Rotate about y axis
: Scale Increment
```


Clock Tower