| **Sub Title : COMPUTER GRAPHICS AND VISUALIZATION LABORATORY** | | |
|---|---|---|
| **Sub Code:CSL66** | **No. of Credits:1.5=0 : 0 : 1.5 (L-T-P)** | **No. of lecture hours/week : 3** |
| **Exam Duration : 3 hours** | **CIE + SEE = 50 + 50 =100** | |

**Course objectives:**

This course will help students to achieve the following objectives:

1. Understand and explain the mathematical and theoretical principles of computer graphics eg: To draw basic objects like lines, triangles and polygons and apply animation, also model 3D objects using openGL built-in functions.
2. Use matrix algebra in computer graphics and implement fundamental algorithms and transformations involved in viewing models.
3. Write basic but complete graphics software systems projection models, illumination models and handling of hidden surfaces and clipping in computer graphics.

**Design, develop, and implement the following programs in C / C++**

| | |
|---|---|
| 1. | a. Program to implement a FLYING KITE<br>b. Create 2D Sierpinski gasket by recursive subdivision of triangle. |
| 2. | a. Write a program to rotate a square. Rotate on mouse left-button clicks. On right click, stop rotation.<br>b. Create 3D Sierpinski gasket by recursive subdivision of tetrahedron. |
| 3. | a. Write a program to handle mouse events. Plot points in random colors by pressing the mouse left button. The display window must be cleared when you press the mouse right button.<br>b. Write a program to demonstrate the usage of display lists. The list should define various geometric objects like square, rectangle, triangle. |
| 4. | a. Write a program to create a font for the letter S of user input thickness.<br>b. Write a program to perform rotation of a cube using vertex arrays. |
| 5. | a. Write a program to create a font for the letter O of user input thickness<br>b. Write a program to draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing. |
| 6. | a. Write a program to create a chessboard.<br>b. Write a program, using OpenGL functions, to demonstrate properties of the light source along with properties of the surfaces of the solid object. |
| 7. | a. Write a program to create a rotating wheel.<br>b. Write a program to draw a square. Use popup menu with options to quit or resize the square. The resize option has submenu with increase size and decrease size as options. The increase size must double the size of the square and the decrease size option must reduce the size by 2. (half size). |

| | |
|---|---|
| 8. | a. Write a program to display a hierarchical menu as below using popup menu. When an option is selected, display the option on output window.<br>View ________normal<br>\| _print<br>\|_web<br>Edit__________ cut<br>\|_ copy<br>\|_paste<br>b. Write a program to create a house like figure and rotate it about a given fixed point using OpenGL functions. |
| **Note: One program from Part A (20 marks) and One program from Part B (30marks) should be executed.** | |

**Course Outcomes:**
At the end of the course students will be able to:

**CO1:** Model 3D objects and also able to animate using openGL built-in functions.
**CO2:** Use matrix algebra in computer graphics and implement fundamental algorithms and transformations involved in viewing models.
**CO3:** Design and Develop complete graphics software systems projection models, illumination models and handling of hidden surfaces and clipping in computer graphics.

| **Cos** | **Mapping with POs** |
|---|---|
| CO1 | PO1,PO2,PO3,PO5,PO12 |
| CO2 | PO1,PO2,PO3,PO5 |
| CO3 | PO1,PO2,PO3,PO5,PO12 |

**Faculty Name:** **Prof. Asha K N**
**Prof. Vinod Kumar K P**

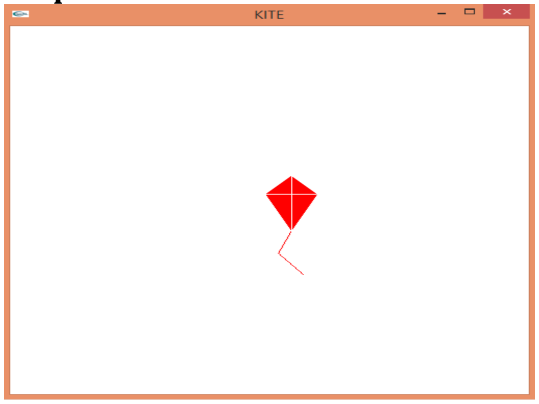
## 1. a) Program to implement a FLYING KITE

```
//KITE FLYING
#include<glut.h>

void init()
{
glViewport(0,0,500,500);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0,500,0,500);
glMatrixMode(GL_MODELVIEW);
}

void delay()
{
int i,j,r;
for(i=0;i<1000;i++)
for(j=0;j<60000;j++)
r=i*j*10;
}

void drawkite(float x,float y)
{
//to draw rhombus shape
glColor3f(1.0,1.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(x-20,y+10);
glVertex2f(x-20,y-40);
glVertex2f(x+20,y-10);
glVertex2f(x+20,y+40);
glEnd();

//to draw two cross lines
glColor3f(0.0,0.0,0.0);
glBegin(GL_LINES);
glVertex2f(x-20,y+10);
glVertex2f(x+20,y-10);

glVertex2f(x-20,y-40);
glVertex2f(x+20,y+40);
glEnd();
glFlush();
}
```

```
//to draw a tail
void drawstring(float x,float y)
{
glColor3f(1.0,0.0,1.0);
glBegin(GL_LINES);
glVertex2f(x-20,y-40);
glVertex2f(x-30,y-80);
glVertex2f(x-30,y-80);
glVertex2f(x-50,y-120);
glVertex2f(x-50,y-120);
glVertex2f(x-80,y-150);
glEnd();
glFlush();
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
float i=50,j=50;
//loop to move kite to right
while(j<=450)
{ i=50;
while(i<400)
{
drawkite(i,j);
drawstring(i,j);
glClear(GL_COLOR_BUFFER_BIT);
delay();
i=i+5;
j=j+1;
}

//loop to move kite to left

while(i>100)
{
drawkite(i,j);
drawstring(i,j);
glClear(GL_COLOR_BUFFER_BIT);
delay();
i=i-5;
j=j+1;
}
}
}
```

```
int main( )
{
glutInitWindowSize(500,500);
glutCreateWindow("Kite Flying");
glutDisplayFunc(display);
init();
glutMainLoop();
}
```

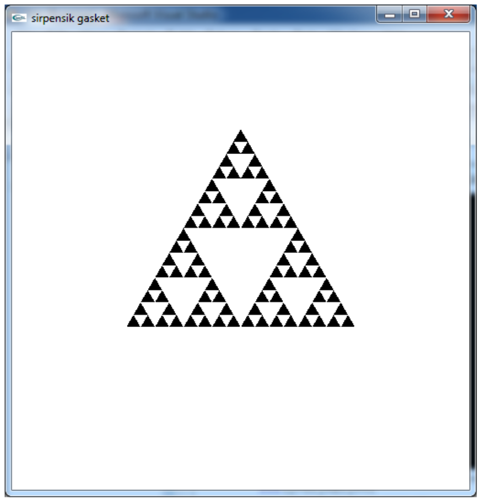
**Output:**



**1. b) Create 2D Sierpinski gasket by recursive subdivision of triangle.**

```
#include<glut.h>
float v[3][2]={{-1.0,-0.58},{1.0,-0.58},{0.0,1.15}};
int n=5;

void triangle(float *a,float *b,float *c)
{
  glVertex2fv(a);
  glVertex2fv(b);
  glVertex2fv(c);
}

void divide_triangle(float *a,float *b,float *c,int m)
{
  float v0[2],v1[2],v2[2];
  int j;
  if(m>0)
  {
            for(j=0;j<2;j++)
            v0[j]=(a[j]+b[j])/2;
            for(j=0;j<2;j++)
            v1[j]=(a[j]+c[j])/2;
            for(j=0;j<2;j++)    v2[j]=(b[j]+c[j])/2;
            divide_triangle(a,v0,v1,m-1);
            divide_triangle(c,v1,v2,m-1);
            divide_triangle(b,v0,v2,m-1);
  }
  else
            triangle(a,b,c);
}

void display()
{
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_TRIANGLES);
  divide_triangle(v[0],v[1],v[2],n);
  glEnd();
  glFlush();
}

void myinit()
{
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(-2,2,-2,2);
  glMatrixMode(GL_MODELVIEW);
  glClearColor(1,1,1,0);
  glColor3f(0,0,1);
}
void main()
{
  glutInitWindowSize(500,500);
  glutInitWindowPosition(0,0);
  glutCreateWindow("2d gasket");
  glutDisplayFunc(display);
  myinit();
  glutMainLoop();
}
```

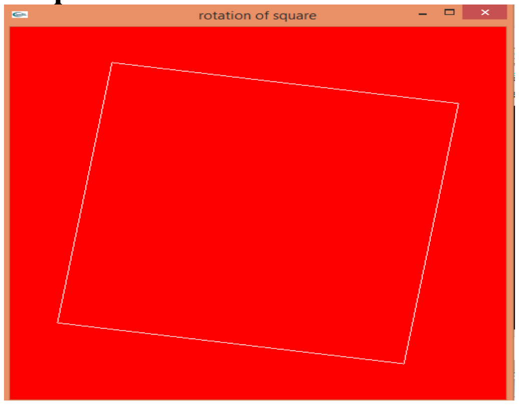
**Output:**



**2. a) Write a program to rotate a square. Rotate on mouse left-button clicks. On right click, stop rotation.**

```
#include<glut.h>
#include<math.h>
float  theta=0,r=0.5;
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_POLYGON);
 glVertex2f(r*cos(theta), r*sin(theta));
 glVertex2f(r*sin (theta), r*-cos(theta));
 glVertex2f(r*-cos(theta), r*-sin(theta));
 glVertex2f(r*-sin(theta), r*cos (theta));
glEnd();
glFlush();
}
void idle()
{
theta+=2;
glutPostRedisplay();
}
void mouse(int btn, int state, int  x, int  y)
{
if((btn==GLUT_LEFT_BUTTON) &&(state==GLUT_DOWN)    glutIdleFunc(idle);
if((btn==GLUT_LEFT_BUTTON) &&(state==GLUT_DOWN)    glutIdleFunc(NULL);
}
int main( )
{
glutCreateWindow("Rotaion of Square");
glutDisplayFunc(display);
glClearColor(0,0,0,0);
glColor3f(1,0,0);
glutMainLoop();
}
```

**Output:**

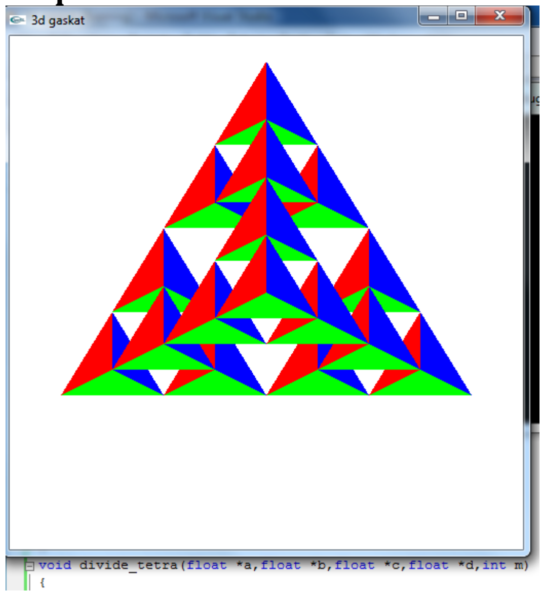


## 2. b) Create 3D Sierpinski gasket by recursive subdivision of tetrahedron.

```
#include<glut.h>
float v[4][3]={{0,0,1},{0.0,0.9,-0.3},{-0.8,-0.4,-0.3},{0.8,-0.4,-0.3}};
float colors[4][3]={{1.0,0.0,0.0},{0.0,1.0,0.0},{0.0,0.0,1.0},{0.0,0.0,0.0}};
int n=2;

void triangle(float *a,float *b,float*c)
{
  glVertex3fv(a);
  glVertex3fv(b);
  glVertex3fv(c);
}

void tetra(float *a,float *b,float *c,float *d)
{
  glColor3fv(colors[0]);
  triangle(a,b,c);
  glColor3fv(colors[1]);
  triangle(a,c,d);
  glColor3fv(colors[2]);
  triangle(a,d,b);
  glColor3fv(colors[3]);
  triangle(b,d,c);
}
void divide_tetra(float *a,float *b,float *c,float *d,int m)
{
  float mid[6][3];
  int j;
  if(m>0)
  {
            for(j=0;j<3;j++)            mid[0][j]=(a[j]+b[j])/2;
            for(j=0;j<3;j++)            mid[1][j]=(a[j]+c[j])/2;
            for(j=0;j<3;j++)            mid[2][j]=(a[j]+d[j])/2;
            for(j=0;j<3;j++)            mid[3][j]=(b[j]+c[j])/2;
            for(j=0;j<3;j++)            mid[4][j]=(c[j]+d[j])/2;
            for(j=0;j<3;j++)            mid[5][j]=(b[j]+d[j])/2;
            divide_tetra(a,mid[0],mid[1],mid[2],m-1);
            divide_tetra(mid[0],b,mid[3],mid[5],m-1);
            divide_tetra(mid[1],mid[3],c,mid[4],m-1);
            divide_tetra(mid[2],mid[5],mid[4],d,m-1);
  }
  else (tetra(a,b,c,d));
}
```

```
void display()
{
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_TRIANGLES);
  divide_tetra(v[0],v[1],v[2],v[3],n);
  glEnd();
  glFlush();
}
void myinit()
{
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-1,1,-1,1,-1,1);
}
void main()
{
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
  glutInitWindowSize(500,500);
  glutCreateWindow("3d gaskat");
  glutDisplayFunc(display);
  glEnable(GL_DEPTH_TEST);
  glClearColor(1.0,1.0,1.0,1.0);
  myinit();
  glutMainLoop();

}
```

**Output:**

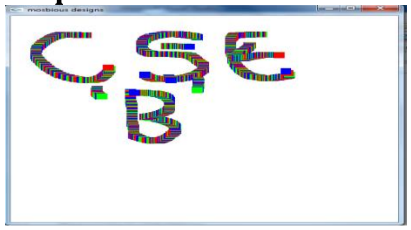


```
void divide_tetra(float *a,float *b,float *c,float *d,int m)
{
```

**3 a)Write a program to handle mouse events. Plot points in random colors by pressing the mouse left button. The display window must be cleared when you press the mouse right button.**

```
#include<stdlib.h>
#include<stdio.h>
#include<glut.h>

void disp()
{ glClear(GL_COLOR_BUFFER_BIT);
glFlush();
}
void drawPoint(int x,int y)
{
glPointSize(2);
glColor3f(rand()%3, rand()%3, rand()%3);
glBegin(GL_POINTS);
glVertex2f(x,y);
glEnd();
glFlush();
}
void myMouse(int b,int s,int x,int y)
{
        if(b==GLUT_LEFT_BUTTON && s==GLUT_DOWN)
        drawPoint(x,y);
        if(b==GLUT_RIGHT_BUTTON && s==GLUT_DOWN)
        disp();
}
void init(int w,int h)
{
glViewport(0,0,w,h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0,w,h,0);
glMatrixMode(GL_MODELVIEW);
}
int main()
{
glutCreateWindow("Square");
glutDisplayFunc(disp);
glutMouseFunc(myMouse);
glutMotionFunc(drawPoint);
glutReshapeFunc(init);
glutMainLoop();
}
```

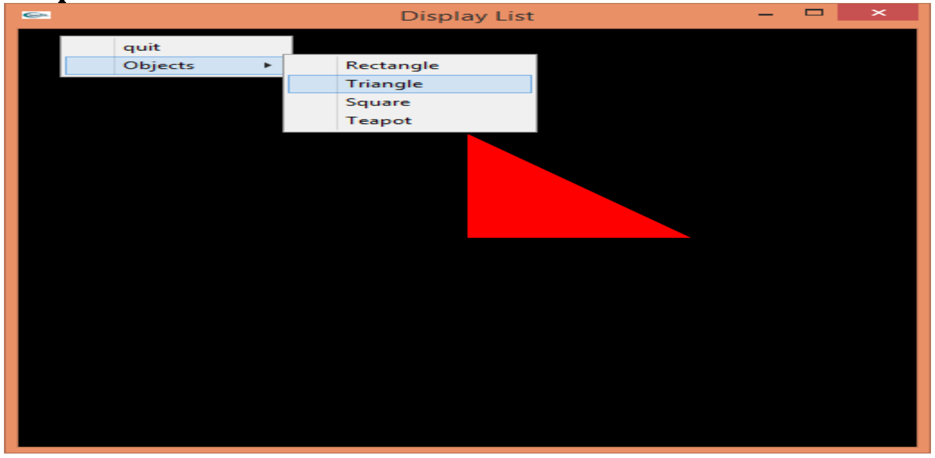
**Output:**



**3 b. Write a Program to demonstrate the usage of display lists. the list should define varios objects like square, rectangle, triangle.**

```
#include <stdlib.h>
#include <glut.h>
void init ()
{ // to draw rectangle
glNewList (1, GL_COMPILE);
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (1.0, 0.0, 1.0);
glBegin (GL_QUADS);
glVertex2f (-.5,-.5);
glVertex2f (.5,-.5);
glVertex2f (.5,0.2);
glVertex2f (-.5,0.2);
glEnd ();
glEndList ();

            // to draw triangle
            glNewList (2, GL_COMPILE);
    glClear (GL_COLOR_BUFFER_BIT);
            glColor3f (1,0,0);
     glBegin (GL_TRIANGLES);
     glVertex2f (0,0);
     glVertex2f (.5,0);
     glVertex2f (0,.5);
     glEnd ();
             glEndList ();
            // to draw square
            glNewList (3, GL_COMPILE);
    glClear (GL_COLOR_BUFFER_BIT);
            glColor3f (1.0, 1.0, 0.0);
     glBegin (GL_QUADS);
    glVertex2f (-0.5,-0.5);
     glVertex2f (0.5,-0.5);
     glVertex2f (0.5,0.5);
              glVertex2f (-0.5,0.5);
     glEnd ();
             glEndList ();
// to draw teapot
           glNewList (4, GL_COMPILE);
          glClear (GL_COLOR_BUFFER_BIT);
          glColor3f (0,1,1);
           glutSolidTeapot(.5);
           glEndList ();
 }
void display(void)
{
    glClear (GL_COLOR_BUFFER_BIT);
  glFlush ();
}
void obj(int i)
{
          switch(i)
          {
          case 2:glCallList (1);break;
          case 3:glCallList (2);break;
          case 4:glCallList (3);break;
          case 5:glCallList (4);break;
          }
          glFlush();
}
void mmenu(int i)
{
          switch(i)
          {
          case 1:exit(0);
          }
}
void main()
{
          glutCreateWindow("Display List");
          int smenu1=glutCreateMenu(obj);
          glutAddMenuEntry("Rectangle",2);
          glutAddMenuEntry("Triangle",3);
          glutAddMenuEntry("Square",4);
          glutAddMenuEntry("Teapot",5);
          glutCreateMenu(mmenu);
          glutAddMenuEntry("quit",1);
          glutAddSubMenu("Objects",smenu1);
          glutAttachMenu(GLUT_RIGHT_BUTTON);
          glutDisplayFunc(display);
          init();
          glutMainLoop();
}
```

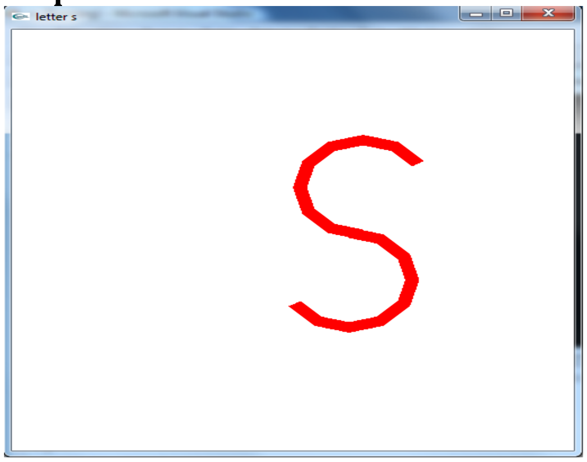
**Output:**



**4 a) Write a program to create a font for the letter S of user input thickness.**

```
#include< glut.h>
#include<math.h>

void FontS()
{
        int i;
        float angle;
        glColor3f(1,0,0);
        glBegin(GL_QUAD_STRIP);
        for(i=1; i<=9;i++)
       {
         angle = 3.14159/6.0*i;//30º in radians
         glVertex2f(0.2*cos(angle),0.2*sin(angle));
         glVertex2f(0.3*cos(angle),0.3*sin(angle));
       }
        glEnd();
        glTranslatef(0,-0.5,0);
        glBegin(GL_QUAD_STRIP);
        for(i=9; i<=18;i++)
            {
              angle = 3.14159/6.0*i;//30º in radians
              glVertex2f(0.2*cos(angle),-0.2*sin(angle));
              glVertex2f(0.3*cos(angle),-0.3*sin(angle));
             }
        glEnd();
}

void display()
{
  glClear(GL_COLOR_BUFFER_BIT );
  glLoadIdentity();
  FontS();
  glFlush();
}


void main()
{

  glutInitWindowSize(500, 500);
  glutCreateWindow("LetterS");
  glutDisplayFunc(display);
  glClearColor (1,1,1,1);
  glutMainLoop();
}
```

**Output:**

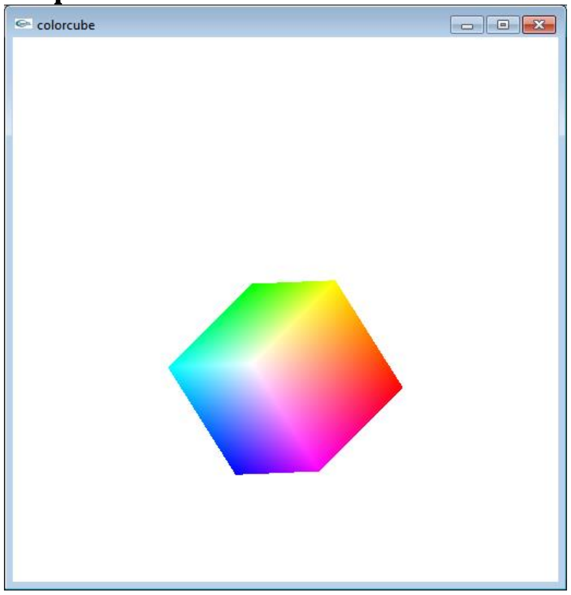


**4 b) Write a program to perform rotation of a cube using vertex arrays.**

```
#include <glut.h>
        float vertices[ ] = {-1,-1,-1,1,-1,-1,1,1,-1, -1,1,-1, -1,-1,1, 1,-1,1, 1,1,1, -1,1,1};
        float colors[ ] = {0,0,0,0,0,1,0,1,0,0,1,1,1,0,0,1,0,1,1,1,0,1,1,1};
        unsigned char cubeIndices[ ]={0,3,2,1,2,3,7,6,0,4,7,3,1,2,6,5,4,5,6,7,0,1,5,4};
float theta[ ] = {0,0,0};
int axis = 2;
void display(void)
{
/* display callback, clear frame buffer and z buffer,  rotate cube and draw, swap buffers */
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
        glLoadIdentity();

        glRotatef(theta[0], 1,0,0);
        glRotatef(theta[1], 0,1,0);
        glRotatef(theta[2],  0,0,1);

         glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, cubeIndices);
        glutSwapBuffers();
}
void spinCube()
{
/* Idle callback, spin cube .1 degrees about selected axis */
        theta[axis] += .1;
        glutPostRedisplay();
}
void mouse(int btn, int state, int x, int y)
{
/* mouse callback, selects an axis about which to rotate */
        if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
        if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
        if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN) axis = 2;
}
void myReshape(int w, int h)
{
  glViewport(0, 0, w, h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  if (w <= h)
    glOrtho(-2, 2, -2 * (float)h/(float)w,2 * (float) h /(float) w, -10,10);
  else
    glOrtho(-2 * (float) w / (float) h, 2 * (float) w / (float) h, -2, 2, -10,10);
  glMatrixMode(GL_MODELVIEW);
}
void main()
{
/* need both double buffering and z buffer */
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize(500, 500);
  glutCreateWindow("colorcube");
  glutReshapeFunc(myReshape);
  glutDisplayFunc(display);
  glutIdleFunc(spinCube);
  glutMouseFunc(mouse);
  glEnable(GL_DEPTH_TEST); /* Enable hidden--surface--removal */
        glEnableClientState(GL_COLOR_ARRAY);
        glEnableClientState(GL_VERTEX_ARRAY);
        glVertexPointer(3, GL_FLOAT, 0, vertices);
        glColorPointer(3,GL_FLOAT, 0, colors);
        glClearColor(1,1,1,1);
        glColor3f(1,1,1);
  glutMainLoop();
}
```

**Output:**

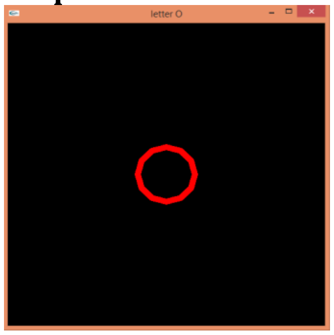


**5a) Write a program to create a font for the letter O of user input thickness.**

```
#include< glut.h>
#include<math.h>

void FontO()
{
        int i;
        float angle;
        glColor3f(1,0,0);
        glBegin(GL_QUAD_STRIP);
        for(i=1; i<=12;i++)
        {
          angle = 3.14159/6.0*i;//30o in radians
          glVertex2f(0.2*cos(angle),0.2*sin(angle));
          glVertex2f(0.3*cos(angle),0.3*sin(angle));
        }
        glEnd();

}

void display()
```

```
{
   glClear(GL_COLOR_BUFFER_BIT );
   glLoadIdentity();
   FontO();
   glFlush();
}

void main()
{

   glutInitWindowSize(500, 500);
   glutCreateWindow("LetterO");
   glutDisplayFunc(display);
   glClearColor (1,1,1,1);
   glutMainLoop();
}
```

**Output:**

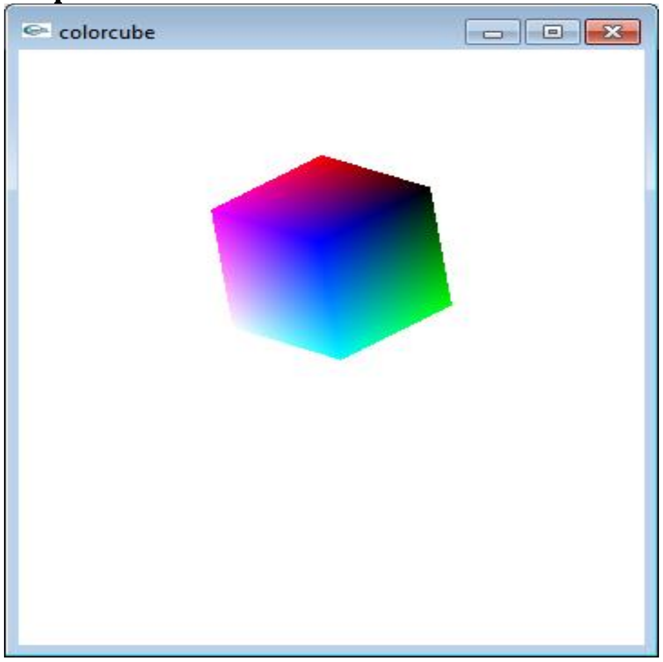


**5 b)Write a program to draw a color cube and allow the user to move the camera suitably to experiment with perspective viewing. Use OpenGL functions.**

```
#include <glut.h>
          float vertices[] = {-1,-1,-1,1,-1,-1,1,1,-1, -1,1,-1, -1,-1,1,1,-1,1, 1,1,1, -1,1,1};

float colors[] = {0,0,0,0,0,1,0,1,0,0,1,1,1,0,0,1,0,1,1,1,0,1,1,1};
unsigned char cubeIndices[]={0,3,2,1,2,3,7,6,0,4,7,3,1,2,6,5,4,5,6,7,0,1,5,4};
float theta[] = {0,0,0};
int axis = 2;
float viewer[3]={1,1,1};

void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
          glLoadIdentity();
          gluLookAt(viewer[0], viewer[1], viewer[2],0,0,0,0,1,0);
          glRotatef(theta[0], 1,0,0);
          glRotatef(theta[1], 0,1,0);
          glRotatef(theta[2],  0,0,1);
  glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, cubeIndices);
        glutSwapBuffers();
}


void spinCube()
{
/* Idle callback, spin cube .1 degrees about selected axis */
        theta[axis] += .1;
        glutPostRedisplay();
}
void mouse(int btn, int state, int x, int y)
{
/* mouse callback, selects an axis about which to rotate */
        if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
        if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
        if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN) axis = 2;
        glutPostRedisplay();
}
void keys(unsigned char k, int x, int y)
{
if (k=='x') viewer[0]-=0.1;
if(k=='X') viewer[0]+=0.1;
if (k=='y') viewer[1]-=0.1;
if(k=='Y') viewer[1]+=0.1;
if (k=='z') viewer[2]-=0.1;
if(k=='Z') viewer[2]+=0.1;
glutPostRedisplay();
}


void myReshape(int w, int h)
{
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   if (w <= h)
   glFrustum(-2, 2, -2 * (float)h/(float)w,2 * (float) h /(float) w, 2, 20);
   else
glFrustum(-2 * (float)w/ (float) h, 2 * (float) w / (float) h, -2, 2, 2, 20);
   glMatrixMode(GL_MODELVIEW);
}


void main()
{
```

```
/* need both double buffering and z buffer */
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowSize(500, 500);
  glutCreateWindow("colorcube");
  glutReshapeFunc(myReshape);
  glutDisplayFunc(display);
  glutIdleFunc(spinCube);
  glutMouseFunc(mouse);
  glutKeyboardFunc(keys);
  glEnable(GL_DEPTH_TEST); /* Enable hidden--surface--removal */
        glEnableClientState(GL_COLOR_ARRAY);
        glEnableClientState(GL_VERTEX_ARRAY);
        glVertexPointer(3, GL_FLOAT, 0, vertices);
        glColorPointer(3,GL_FLOAT, 0, colors);
        glClearColor(1,1,1,1);
        glColor3f(1,1,1);
  glutMainLoop();
}
```

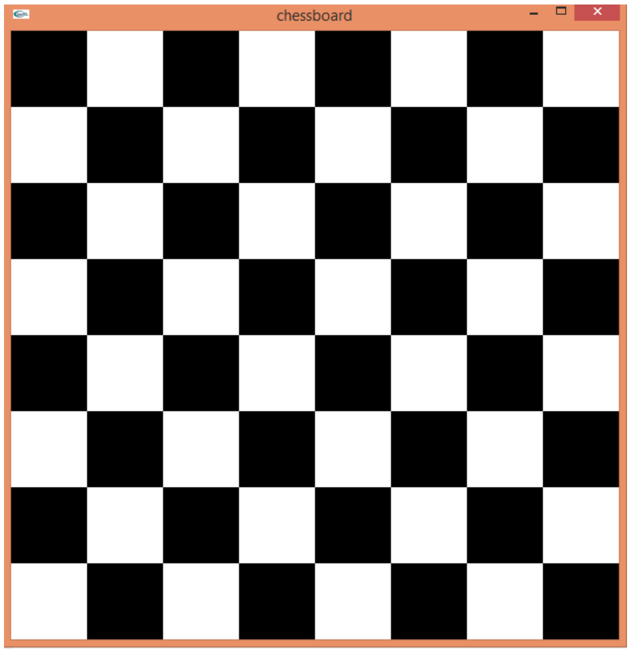
**Output:**



**6 a) Write a program to create a chessboard.**

```
//CHESS BOARD
#include<glut.h>

void display(void)
{
int i,j;
glClear(GL_COLOR_BUFFER_BIT);
glClearColor (1,1,1,1);
for(i=0;i<8;i++){
for(j=0;j<8;j++){

        if((i+j)%2==0)
                glColor3f(0.0,0.0,0.0);
        else
                glColor3f(1.0,1.0,1.0);

        glBegin(GL_QUADS);
        glVertex2f(i*5,j*5);
        glVertex2f((i+1)*5,j*5);
        glVertex2f((i+1)*5,(j+1)*5);
        glVertex2f(i*5,(j+1)*5);
        glEnd();
        }
}
glFlush();
}
void myinit()
{
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0,40,0,40);
glMatrixMode(GL_MODELVIEW);
}
int main()
{
glutInitWindowSize(500, 500);
glutCreateWindow("Chess");
glutDisplayFunc(display);
myinit();
glutMainLoop();
}
```

**Output:**

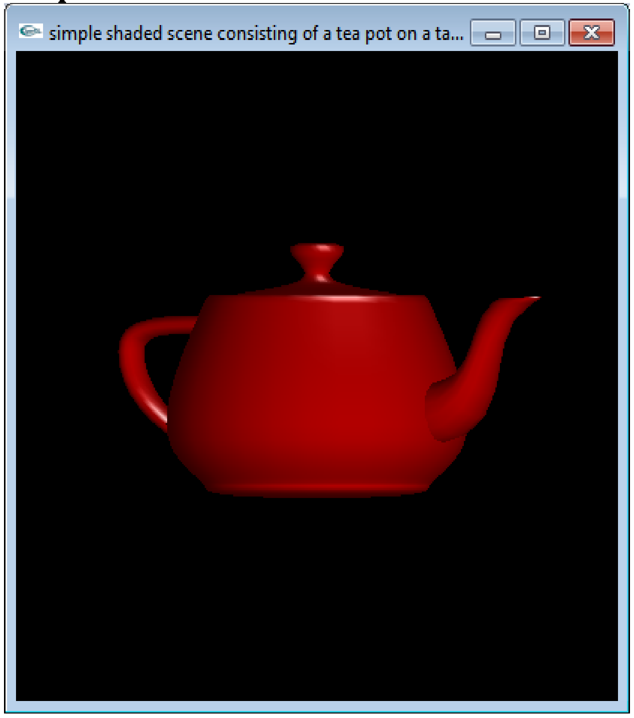


**6 b) Write a program, using OpenGL functions, to demonstrate properties of the light source along with properties of the surfaces of the solid object**

```
#include <glut.h>
float xv=0,yv=0;
void displaySolid (void)
{
//set properties of the surface material
float mat_ambient[] = {1,0,0,1}; // red
float mat_diffuse[] = {.5, .5, .5, 1};
float mat_specular[] = {1,1,1,1};
float mat_shininess[] = {50};

glMaterialfv (GL_FRONT, GL_AMBIENT, mat_ambient);
glMaterialfv (GL_FRONT, GL_DIFFUSE, mat_diffuse);
glMaterialfv (GL_FRONT, GL_SPECULAR, mat_specular);
glMaterialfv (GL_FRONT, GL_SHININESS, mat_shininess);

//set the light source properties
GLfloat lightIntensity[] = {1,0,0,1};
float light_position[]= {xv,yv,0,0};
glLightfv (GL_LIGHT0, GL_POSITION, light_position);
glLightfv (GL_LIGHT0, GL_DIFFUSE, lightIntensity);

//set the camera
glMatrixMode (GL_PROJECTION);
glLoadIdentity();
glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glPushMatrix();
glRotated (30, 0, 1, 0);
glutSolidTeapot (.5);
glPopMatrix ();
glFlush();
}
void keys(unsigned char k, int x, int y)
{
if(k=='r') xv+=1;
if(k=='l') xv-=1;
if(k=='u') yv+=1;
if(k=='d') yv-=1;
glutPostRedisplay();
}
void main ()
```

```
{
glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize (640, 480);
glutInitWindowPosition (100, 100);
glutCreateWindow ("Opengl Light");
glutDisplayFunc (displaySolid);
glutKeyboardFunc(keys);
glEnable (GL_LIGHTING);
glEnable (GL_LIGHT0);
glShadeModel (GL_SMOOTH);
glEnable (GL_DEPTH_TEST);
glEnable (GL_NORMALIZE);
glClearColor (0,0,0,0);
glViewport (0, 0, 640, 480);
glutMainLoop();
}
```

**Output:**

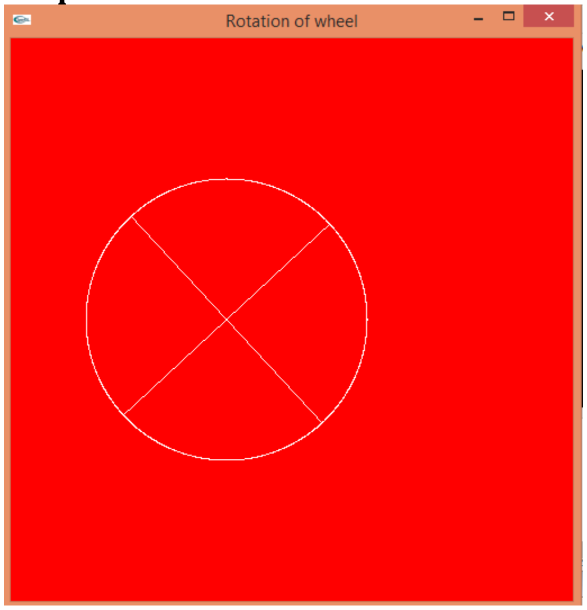


**7 a) Write a program to create a rotating wheel**

```
#include<glut.h>
#include<math.h>
float t,r=0.5,x,y,t1=360,i=-0.5;
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glClearColor(1,0,0,0);
	glColor3f(1,1,1);
	for(t=0;t<360;t=t+0.1)
	{
		x=i+r*cos(t);
		y=r*sin(t);
		glBegin(GL_POINTS);
		glVertex2f(x,y);
		glEnd();
	}
	glBegin(GL_LINES);
	glVertex2f(r*cos(t1)+i, r*sin(t1));
	glVertex2f(r*-cos(t1)+i, r*-sin(t1));
	glVertex2f(r*-sin(t1)+i, r*cos(t1));
	glVertex2f(r*sin(t1)+i, r*-cos(t1));
	glEnd();
	glFlush();
}
void idle()
{
	if(i<1)
		i=i+0.001;
	else
		i=-0.5;
	t1-=0.01;
	display();
}
void main()
{
	glutInitWindowSize(500,500);
	glutInitWindowPosition(0,0);
	glutCreateWindow("Rotation of wheel");
	glutDisplayFunc(display);
	glutIdleFunc(idle);
```

```
        glutMainLoop();
}
```

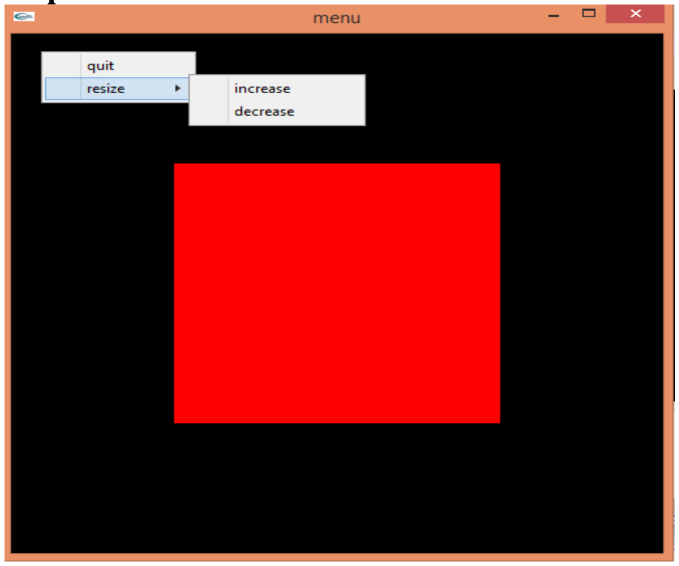
**Output:**



**7 b)Write a program to draw a square. Use popup menu with options to quit or resize the square. The resize option has submenu with increase size and decrease size as options. The increase size must double the size of the square and the decrease size option must reduce the size by 2. (half size) . Min size is 1.**

```
#include< glut.h>
#include<stdio.h>
float size=1;
void mydisplay()
{
   glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POLYGON);
      glVertex2f(-.5*size, -.5*size);
      glVertex2f(.5*size, -.5*size);
      glVertex2f(.5*size, .5*size);
      glVertex2f(-.5*size,.5*size);
    glEnd();
   glFlush();
}

void demomenu(int id)
{
switch(id)
{
case 1:exit(0);break;
}
glutPostRedisplay();
}
void sizemenu(int id)
{
switch(id)
{
case 2: size=2*size;break;
case 3: size=size/2;break;
}
glutPostRedisplay();
}
void main()
{
int submenu;
glutInitWindowSize(500,500);
glutCreateWindow("menu");
sub_menu=glutCreateMenu(sizemenu);
glutAddMenuEntry("increase size",2);
glutAddMenuEntry("decrease size",3);
glutCreateMenu(demomenu);
glutAddMenuEntry("quit",1);
glutAddSubMenu("resize",submenu);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutDisplayFunc(mydisplay);
glClearColor(1,1,1,1);
glColor3f(1,0,0);
glutMainLoop();
}
```

**Output:**

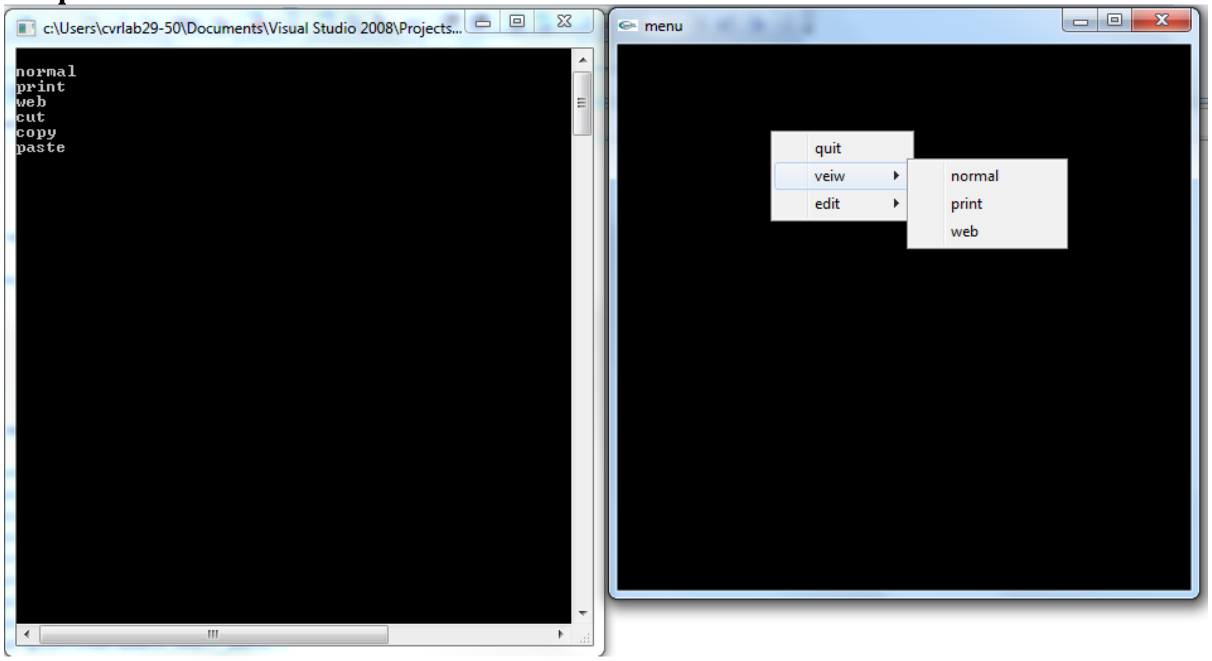


**8 a) Write a program to display a hierarchical menu as below using popup menu. When an option is selected, display the option on output window.**

```
View ______ normal
              | _print
              |_web
Edit________cut
              |_ copy
              |_paste
```

```
#include< glut.h>
#include<stdio.h>
void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT);
 glFlush();
}
void demomenu(int id)
{
        switch(id)
        {
         case 1:exit(0);break;
        }
}
void editmenu(int id)
{
        switch(id)
        {
                case 5: printf("cut");break;
                case 6: printf("copy");break;
                case 7: printf("paste");break;
        }
}
void sizemenu(int id)
{
        switch(id){
                case 2: printf("normal");break;
                case 3: printf("print");break;
                case 4:printf("web");break;
        }
}
void main()
{
        int submenu1, submenu2;
        glutInitWindowSize(500,500);
       glutCreateWindow("menu");

        sub_menu1=glutCreateMenu(sizemenu);
        glutAddMenuEntry("normal",2);
        glutAddMenuEntry("print",3);
        glutAddMenuEntry("web",4);

        sub_menu2=glutCreateMenu(editmenu);
        glutAddMenuEntry("cut",5);
        glutAddMenuEntry("copy",6);
        glutAddMenuEntry("paste",7);

        glutCreateMenu(demomenu);
        glutAddMenuEntry("quit",1);
        glutAddSubMenu("view",submenu1);
        glutAddSubMenu("edit",submenu2);
        glutAttachMenu(GLUT_RIGHT_BUTTON);

        glutDisplayFunc(mydisplay);
        init();
        glutMainLoop();
}
```

**Output:**

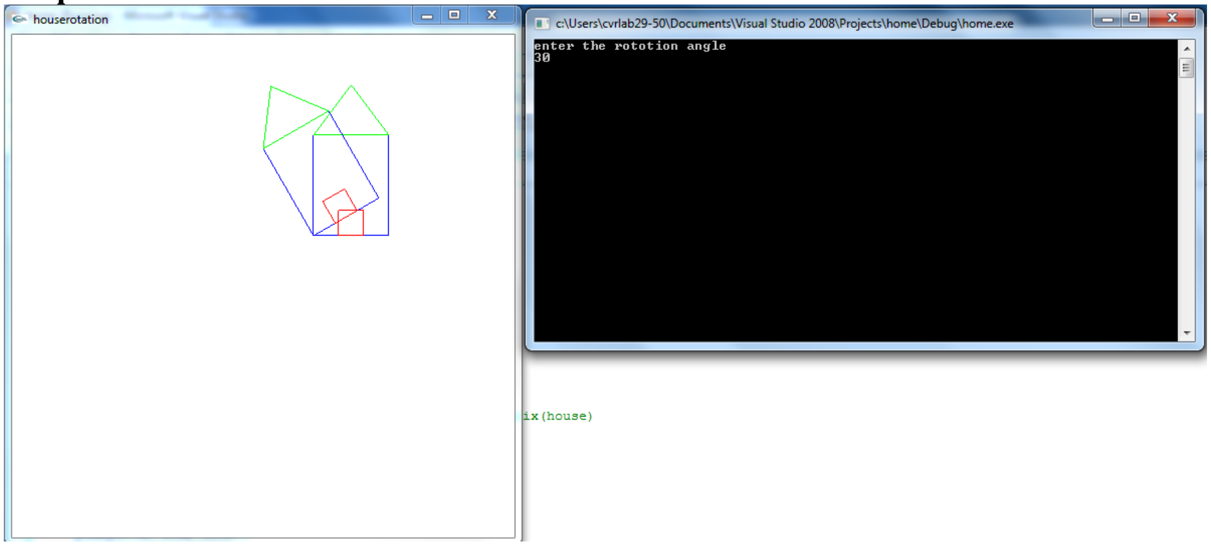

c:\Users\cvrlab29-50\Documents\Visual Studio 2008\Projects...
menu
quit
veiw
edit
normal
print
web

```
normal
print
web
cut
copy
paste
```

**8 b)Write a program to create a house like figure and rotate it about a given fixed point using OpenGL functions.**

```
#include <stdio.h>
#include <math.h>
#include < glut.h>
float house[3][9]={{100.0,100.0,175.0,250.0,250.0,150.0,150.0,200.0,200.0},
                                {100.0,300.0,400.0,300.0,100.0,100.0,150.0,150.0,100.0},
                                        {1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0}};
float h=100; // Pivot point
float k=100;
float theta;
void drawhouse( )
{       glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINE_LOOP);
 glVertex2f(house[0][0],house[1][0]);
 glVertex2f(house[0][1],house[1][1]);
 glVertex2f(house[0][3],house[1][3]);
 glVertex2f(house[0][4],house[1][4]);
 glEnd();
glColor3f(1.0,0.0,0.0);
 glBegin(GL_LINE_LOOP);
 glVertex2f(house[0][5],house[1][5]);
 glVertex2f(house[0][6],house[1][6]);
 glVertex2f(house[0][7],house[1][7]);
 glVertex2f(house[0][8],house[1][8]);
 glEnd();
glColor3f(0.0, 0.0, 1.0);
 glBegin(GL_LINE_LOOP);
 glVertex2f(house[0][1],house[1][1]);
 glVertex2f(house[0][2],house[1][2]);
 glVertex2f(house[0][3],house[1][3]);
 glEnd();
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glLoadIdentity();
drawhouse();
glTranslatef(h,k,0);
glRotatef(theta,0,0,1);
glTranslatef(-h,-k,0)
drawhouse();
glFlush();
}


void myinit()
{
        glClearColor(1,1,1,1);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(-500,500,-500,500);
        glMatrixMode(GL_MODELVIEW);
}
void main()
{
        printf("Enter the rotation angle\n");
        scanf("%f", &theta);
        glutInitWindowSize(500,500);
        glutInitWindowPosition(0,0);
        glutCreateWindow("house rotation");
        glutDisplayFunc(display);
        myinit();
        glutMainLoop();
}
```

**Output:**


houserotation
c:\Users\cvrlab29-50\Documents\Visual Studio 2008\Projects\home\Debug\home.exe

```
enter the rototion angle
30
```