CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | |

Internal Assesment Test – II – May 2017

Sub:	COMPUTER GRAPHICS & VISUALIZATION							Code:	10CS65
Date:	10 / 05 / 2017	Duration:	90 mins	Max Marks:	50	Sem:	6 A,B,C	Branch:	CSE

Answer Any FIVE FULL Questions

		Marks	OBE	
			CO	RBT
1.	Explain various types of views with neat diagrams.	[10]	CO6	L2
2.	Derive the projection matrices for oblique projection matrices.	[10]	CO6	L3
3.	Write a note on hidden surface removal concept. Explain Projection Normalization.	[10]	CO2	L3
4.	Derive the simple perspective and orthographic projections.	[10]	CO6	L3
5.	What are the different types of light sources and light-material interactions used in OpenGL?	[10]	CO5	L2
6.	Explain Phong lighting model with neat diagram? How to represent (functions) material and light interaction in OpenGL?	[10]	CO5	L2
7.	Write a program to approximate a sphere by recursive sub division of tetrahedron.	[10]	CO2	L3
8.	Write a note on different polygon shading used in OpenGL.	[10]	CO5	L3

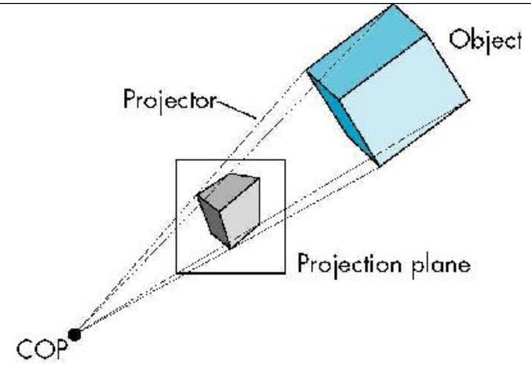
CMR
INSTITUTE OF
TECHNOLOGY

Internal Assessment Test II – April 2017

SOLUTION

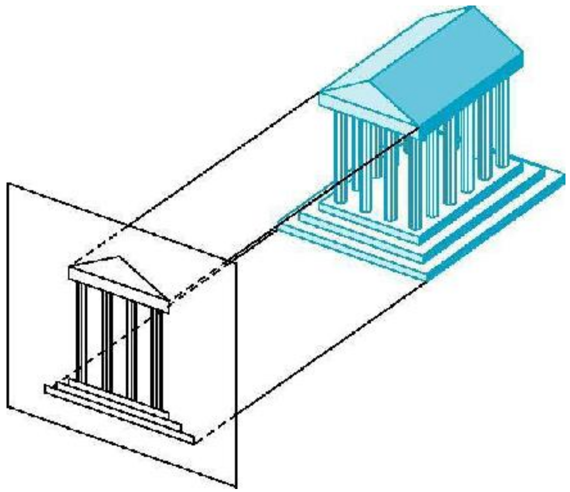
Sub:	Computer Graphics & Visualization							Code:	10CS65
Date:	10/05/2017	Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	CSE

	Questions and Answers.	Marks	OBE CO	OBE RBT
1	Explain various types of views with neat diagrams. 3 basic elements for viewing: – One or more objects – A viewer with a projection surface – Projectors that go from the object(s) to the projection surface Classical views are based on the relationship among these elements. Each object is assumed to constructed from flat *principal faces* – Buildings, polyhedra, manufactured objects – Front, top and side views. **Perspective and parallel projections:** Parallel viewing is a limiting case of perspective viewing. Perspective projection has a COP where all the projector lines converge.	10	CO6	L2



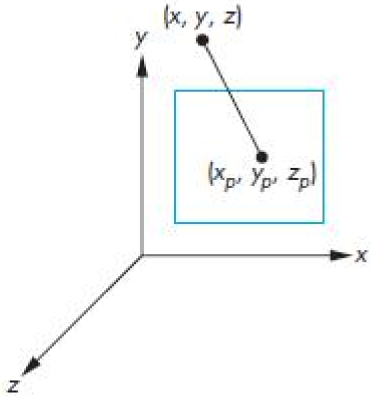
Orthographic Projections:

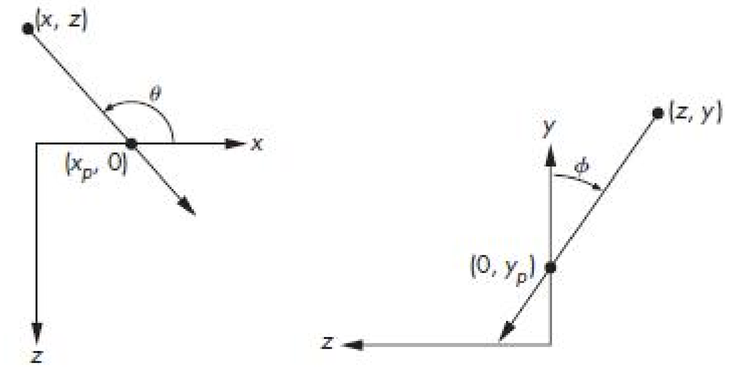
Projectors are perpendicular to the projection plane. Projection plane is kept parallel to one of the principal faces. A viewer needs more than 2 views to visualize what an object looks like from its multiview orthographic projection.



Axonometric Projections

Projectors are orthogonal to the projection plane , but projection plane can move relative to

	object. Classification by how many angles of a corner of a projected cube are the same none. **Perspective Viewing** Characterized by diminution of size i.e. when the objects move farther from the viewer it appears smaller. Major use is in architecture and animation. **Oblique Viewing** The oblique views are the most general parallel views. We obtain an oblique projection by allowing the projectors to make an arbitrary angle with the projection plane.			
2	Derive the projection matrices for oblique projection matrices.  An oblique projection can be characterized by the angle that the projectors make with the projection plane, as shown in the above figure. We can derive the equations for oblique projections by considering the top and side views in the below figure.	10	CO6	L3



If we consider the top view, we can find xp by noting that

tan θ = z/xp-x

and thus,

xp=x+z cot θ

Likewise,

yp= y + z cot φ.

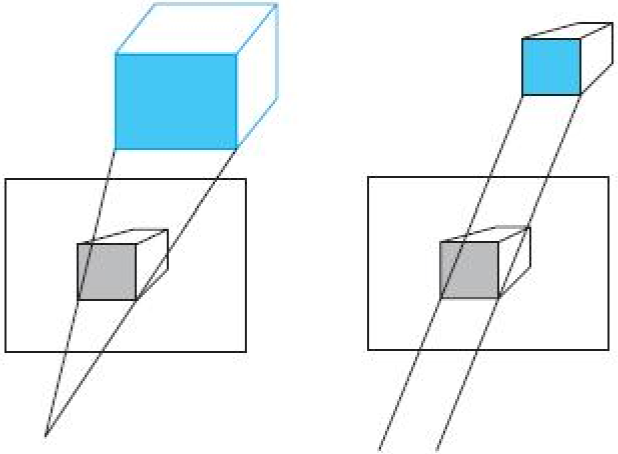
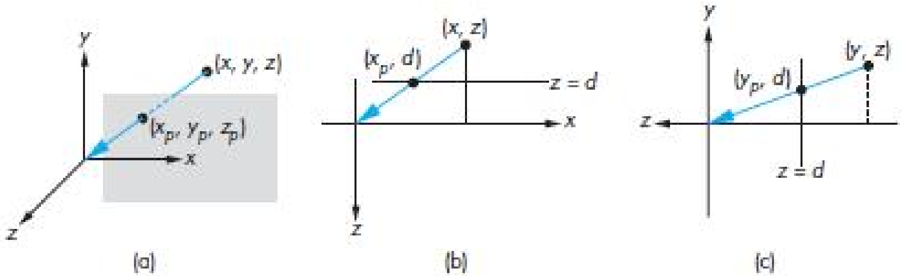
And zp=0;

$$\mathbf{P} = \mathbf{M}_{\text{orth}}\mathbf{H}(\theta, \phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \cot\theta & 0 \\ 0 & 1 & \cot\phi & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where H(θ , φ) is a shearing matrix. Thus, we can implement an oblique projection by first doing a shear of the objects by H(θ , φ) and then doing an orthographic projection.

And

	$\mathbf{ST} = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & -\frac{2}{far-near} & -\frac{far+near}{near-far} \\ 0 & 0 & 0 & 1 \end{bmatrix}$ Final Matrix for oblique projection is $N = M_{orth}STH$			
3	Write a note on hidden surface removal concept. Explain Projection Normalization. A graphics system passes all the faces of a 3d object down the graphics pipeline to generate the image. But the view might not be able to view all these phases. For e.g. all the 6 faces of a cube might not be visible to a viewer. Thus the graphics system must be careful as to which surfaces it has to display. Hidden surface – removal algorithms are those that remove the surfaces of the image that should not be visible to the viewer. Projection normalization is used to convert all projections into orthogonal projections by first distorting the objects such that the orthogonal projection of the distorted objects is the same as the desired projection of the original objects. The concatenation of the normalization matrix, which carries out the distortion and the simple orthogonal projection matrix yields a homogeneous coordinate matrix that produces the desired projection.	10	CO2	L3

	 One advantage of this approach is that we can design the normalization matrix so that view volume is distorted into the canonical view volume, which is the cube defined by the planes. $x = \pm 1,\ y = \pm 1,\ z = \pm 1.$ Besides the advantage of having both perspective and parallel views supported by the same pipeline by loading in the proper normalization matrix, the canonical view volume simplifies the clipping process because the sides are aligned with the coordinate axes.			
4	Derive the simple perspective and orthographic projections. Simple Perspective projection:  FIGURE 4.32 Three views of perspective projection. (a) Three-dimensional view. (b) Top view. (c) Side view.	10	CO6	L3

A point in space (x, y, z) is projected along a projector into the point (xp, yp, zp). All projectors pass through the origin, and, because the projection plane is perpendicular to the z-axis.

zp= d

Because the camera is pointing in the negative z-direction, the projection plane is in the negative half-space z < 0, and the value of d is negative.

From the top view shown in Figure 4.32(b), we see two similar triangles whose tangents must be the same. Hence,

x/z = xp/d,

therefore, xp=x/z/d

similarly, using figure 4.32(c) we get,

yp=y/z/d

Although this perspective transformation preserves lines, it is not affine. It is also irreversible. Because all points along a projector project into the same point, we cannot recover a point from its projection.

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

If p={x,y,z} is the point, then

Mp = q

Where $\mathbf{q} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix}$

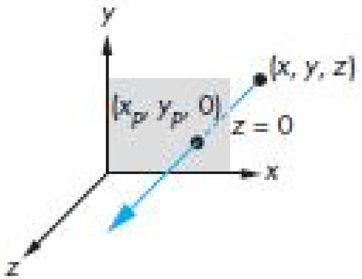
we have to divide the first three components by the fourth to return to our original three dimensional space, we obtain the results

$$x_p = \frac{x}{z/d},$$

$$y_p = \frac{y}{z/d},$$

$$z_p = \frac{z}{z/d} = d,$$

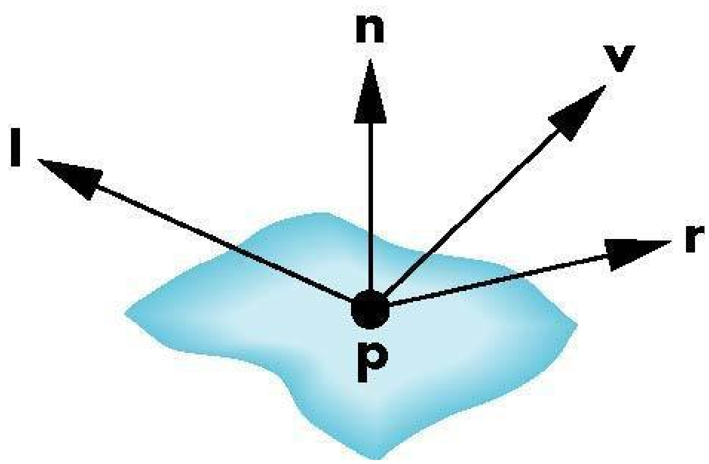
Simple Orthogonal projection:



Let plane be at z=0, and if a point P={x,y,z} is projected orthogonally then,

	xp= x, yp= y, zp= 0. We can write this result using our original homogeneous coordinates. $$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$			
5	What are the different types of light sources and light-material interactions used in OpenGL? These are the different types of light sources: Point source: This kind of source can be said as a distant source or present infinite distance away (parallel) Spotlight: This source can be considered as a restrict light from ideal point source. A Spotlight origins at a particular point and covers only a specific area in a cone shape. Ambient light – Same amount of light everywhere in scene – Can model contribution of many sources and reflecting surfaces Any kind of light source will have 3 component colors namely R, G and B **Types of Materials**	10	CO5	L2

	Specular surfaces – These surfaces exhibit high reflectivity. In these surfaces, the angle of incidence is almost equal to the angle of reflection. Diffuse surfaces – These are the surfaces which have a matt finish. These types of surfaces scatter light Translucent surfaces – These surfaces allow the light falling on them to partially pass through them. The smoother a surface, the more reflected light is concentrated in the direction a perfect mirror would reflect light. A very rough surface scatters light in all directions.			
6	Explain Phong lighting model with neat diagram? How to represent (functions) material and light interaction in OpenGL? Phong developed a simple model that can be computed rapidly It considers three components o Diffuse o Specular o Ambient And Uses four vectors – To source represented by the vector l – To viewer represented by the vector v – Normal represented by the vector n – Perfect reflector represented by the vector r	10	CO5	L2



Ambient Reflection

The amount of light reflected from an ambient source I_a is given by the ambient reflection coefficient: $R_a = k_a$ Since the ambient reflection co efficient is some positive factor,

$0 <= k_a <= 1$

Therefore $I_a = k_a L_a$

Diffuse Reflection

A Lambertian Surface has: Perfectly diffuse reflector, Light scattered equally in all directions. Here the light reflected is proportional to the vertical component of incoming light

– reflected light ~cos q_i

– cos q_i = **l** · **n** if vectors normalized

– There are also three coefficients, k_r, k_b, k_g that show how much of each color component is reflected

Specular Surfaces

Most surfaces are neither ideal diffusers nor perfectly specular (ideal reflectors). Smooth surfaces show specular highlights due to incoming light being reflected in directions concentrated close to the direction of a perfect reflection . This kind of specular reflection could be observed in mirrors.

$$I = \frac{1}{a + bd + cd^2}(k_d L_d \max(\mathbf{l} \cdot \mathbf{n}, 0) + k_s L_s \max((\mathbf{r} \cdot \mathbf{v})^{\alpha}, 0)) + k_a L_a.$$

Shading calculations are enabled by

- glEnable(GL_LIGHTING)
- Once lighting is enabled, glColor() ignored

Must enable each light source individually

- glEnable(GL_LIGHTi) i=0,1…..

For each light source, we can set an RGB for the diffuse, specular, and ambient parts, and the position

```
GLfloat diffuse0[]={1.0, 0.0, 0.0, 1.0};
 GLfloat ambient0[]={1.0, 0.0, 0.0, 1.0};
GLfloat specular0[]={1.0, 0.0, 0.0, 1.0};
 Glfloat light0_pos[]={1.0, 2.0, 3,0, 1.0};
```

glEnable(GL_LIGHTING);

glEnable(GL_LIGHT0);

glLightv(GL_LIGHT0, GL_POSITION, light0_pos);

glLightv(GL_LIGHT0, GL_AMBIENT, ambient0);

glLightv(GL_LIGHT0, GL_DIFFUSE, diffuse0);

glLightv(GL_LIGHT0, GL_SPECULAR, specular0);

Material Properties

All material properties are specified by :

glMaterialfv(GLenum face, GLenum type, GLfloat *pointer_to_array)

We have seen that each material has a different ambient, diffuse and specular properties.

GLfloat ambient[] = {1.0,0.0,0.0,1.0}

GLfloat diffuse[] = {1.0,0.8,0.0,1.0}

GLfloat specular[] = {1.0, 1.0, 1.0,1.0}

Defining shininess and emissive properties

glMaterialf(GL_FRONT_AND_BACK,GL_SHINENESS,100.0)

GLfloat emission [] = {0.0,0.3,0.3,1.0};

glMaterialfv(GL_FRONT_AND_BACK,GL_EMISSION, emission)

Defining Material Structures

```
typedef struct material
{
        GLfloat ambient[4];
         GLfloat diffuse[4];
        GLfloat specular[4];
        GLfloat shineness;
}
materialStruct;
```

7	Write a program to approximate a sphere by recursive sub division of tetrahedron.	10	CO2	L3

```
#include <GL/glut.h>
#include<stdio.h>
#include<math.h>
float v[4][3]={{0.0,0.0,1.0},
             {0.0,0.94,-0.33},
             {-0.82,-0.47,-0.33},
             {0.82,-0.47,-0.33}};
int n;
void triangle (GLfloat *va,GLfloat *vb,GLfloat *vc)
{
   glBegin(GL_TRIANGLES);
                    glVertex3fv(va);
                    glVertex3fv(vb);
                    glVertex3fv(vc);
   glEnd();
}
void normalize(float *p)
{
         double d=0.0;
```

```
            int i;
            for(i=0;i<3;i++)
        d+=p[i]*p[i];

            d=sqrt(d);
            if(d>0.0)
        for(i=0;i<3;i++)
           p[i]/=d;
}
void divide_tetra(GLfloat *a,GLfloat *b,GLfloat *c,int m)
{
            float m1[3],m2[3],m3[3];
            int j;
            if(m>0)
            {           /*compute six midpoints*/
                        for(j=0;j<3;j++) m1[j]=(a[j]+b[j])/2;
                        normalize(m1);
                        for(j=0;j<3;j++) m2[j]=(a[j]+c[j])/2;
                        normalize(m2);
                        for(j=0;j<3;j++) m3[j]=(c[j]+b[j])/2;
                        normalize(m3);

        divide_tetra(a,m2,m1,m-1);
        divide_tetra(c,m3,m2,m-1);
        divide_tetra(b,m1,m3,m-1);
        divide_tetra(m1,m2,m3,m-1);
            }
            else
        triangle( a, b, c);      //draw triangle at end of recursion//
}
void display(void)
{
    glClearColor(1.0,1.0,1.0,1.0);
            glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glColor3f(1,0,0);
```

```
        divide_tetra(v[0],v[1],v[2],n);
        divide_tetra(v[3],v[2],v[1],n);
        divide_tetra(v[0],v[3],v[1],n);
        divide_tetra(v[0],v[2],v[3],n);

        glFlush();
}
void myReshape(int w,int h)
{
        glViewport(0,0,w,h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-2.0,2.0,-2.0,2.0 ,-10.0,10.0);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
}
int main(int argc,char **argv)
{
printf("enter the no of division ");
scanf("%d",&n);
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
glutInitWindowSize(500,500);
glutCreateWindow("3d gasket");
glutReshapeFunc(myReshape);
glutDisplayFunc(display);
glEnable(GL_DEPTH_TEST);
glutMainLoop();
return 0;
}
```

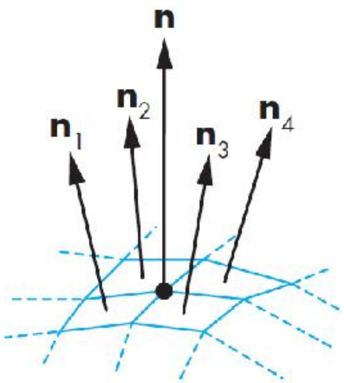
8	Write a note on different polygon shading used in OpenGL. OpenGL exploit the efficiencies possible for rendering flat polygons by decomposing curved surfaces into many small, flat polygons. Consider a polygonal mesh, where each polygon is flat and thus has a well-defined normal vector. There are three ways to shade the	10	CO5	L3

polygons: flat shading, smooth or Gouraud shading, and Phong shading.

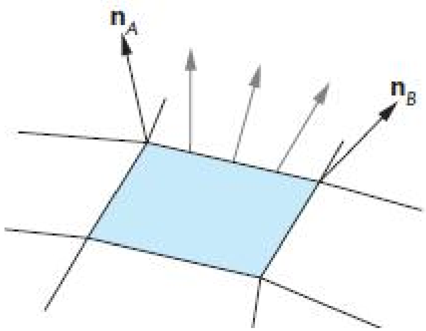
Flat Shading: The three vectors—l, n, and v—can vary as wemove frompoint to point on a surface. For a flat polygon, however, n is constant. If we assume a distant viewer, v is constant over the polygon. Finally, if the light source is distant, l is constant. Here distant could be interpreted in the strict sense of meaning that the source is at infinity. The necessary adjustments, such as changing the location of the source to the direction of the source, could then be made to the shading equations and to their implementation. Distant could also be interpreted in terms of the size of the polygon relative to how far the polygon is from the source or viewer, If the three vectors are constant, then the shading calculation needs to be carried out only once for each polygon, and each point on the polygon is assigned the same shade. This technique is known as flat, or constant, shading. Flat shading will show differences in shading among the polygons in our mesh. If the light sources and viewer are near the polygon, the vectors l and v will be different for each polygon. However, if our polygonal mesh has been designed to model a smooth surface, flat shading will almost always be disappointing because we can see even small differences in shading between adjacent polygons.

Smooth and Gouraud Shading: Suppose that the lighting calculation is made at each vertex using the material properties and the vectors n, v, and l computed for each vertex. Thus, each vertex will have its own color that the rasterizer can use to interpolate a shade for each fragment. Note that if the light source is distant, and either the viewer is distant or there are no specular reflections, then smooth (or interpolative) shading shades a polygon in a constant color. If we consider our mesh, the idea of a normal existing at a vertex should cause concern to anyone worried about mathematical correctness. Because multiple polygons meet at interior vertices of the mesh, each of which has its own normal, the normal at the vertex is discontinuous. Although this situation might complicate the mathematics, Gouraud realized that the normal at the vertex could be defined in such a way as to achieve smoother shading through interpolation. Consider an interior vertex, as shown in below figure where four polygons meet. Each has its own normal. In Gouraud shading, we define the normal at a vertex to be the normalized average of the normals of the polygons that share the vertex. For our example, the vertex normal is given by

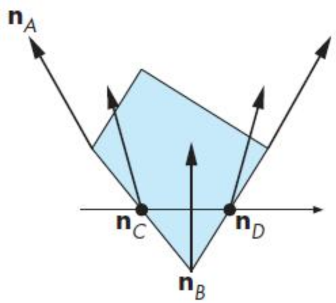
n=n1+n2+n3+n4/|n1+n2+n3+n4|



Phong Shading: Phong proposed that instead of interpolating vertex intensities, as we do in Gouraud shading, we interpolate normals across each polygon. Consider a polygon that shares edges and vertices with other polygons in the mesh, as shown below



We can compute vertex normals by interpolating over the normals of the polygons that

	share the vertex. Next, we can use interpolation to interpolate the normals over the polygon. Consider below figure. We can use the interpolated normals at vertices A and B to interpolate normals along the edge between them: $nC(\alpha) = (1-\alpha)nA + \alpha nB$. We can do a similar interpolation on all the edges. The normal at any interior point can be obtained from points on the edges by $n(\alpha, \beta) = (1-\beta)nC + \beta nD$.  Once we have the normal at each point, we can make an independent shading calculations.				

Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1:	Describe pipeline architecture w.r.t two dimensional applications.	2	1	2	1	-	-	-	-	1	-	2	-
CO2:	Explain pipeline Hidden surface removal, implicit functions, color mechanism and demonstrate approximation of sphere	2	2	1	-	3	-	-	-	-	-	-	-
CO3:	Design and Develop CAD program using picking, Display List, Menu, Input and Output devices	3	-	3	2	3	-	-	-	-	-	1	-
CO4:	Experiment affine transformation activities w.r.t to Translation, Rotation and Scaling operations.	1	-	1	1	-	-	-	-	-	-	-	-
CO5:	List and summarize details of light sources and material properties	2	-	-	1	2	2	-	-	-	-	-	-
CO6:	Analyze implementation strategies w.r.t clipping and display consideration concepts	1	1	1	2	2	2	3	-	-	-	1	-

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PO1 - *Engineering knowledge*; PO2 - *Problem analysis*; PO3 - *Design/development of solutions*; PO4 - *Conduct investigations of complex problems*; PO5 - *Modern tool usage*; PO6 - *The Engineer and society*; PO7- *Environment and sustainability*; PO8 – *Ethics*; PO9 - *Individual and team work*; PO10 - *Communication*; PO11 - *Project management and finance*; PO12 - *Life-long learning*