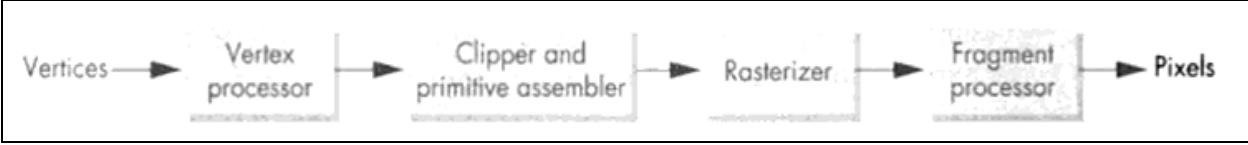


Internal Assessment Test I – March 2017

SOLUTION

Sub:	Computer Graphics & Visualization					Code:	10CS65		
Date:	30/03/2017	Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	CSE

Questions and Answers.		Marks	OBE	
			CO	RBT
1	<p>With neat diagram, explain graphics pipeline architecture with major steps involved in imaging process?</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;">  <pre> graph LR A[Vertices] --> B[Vertex processor] B --> C[Clipper and primitive assembler] C --> D[Rasterizer] D --> E[Fragment processor] E --> F[Pixels] </pre> </div> <p>Process objects one at a time in the order they are generated by the application</p> <ul style="list-style-type: none"> All steps can be implemented in hardware on the graphics card <p>Vertex Processor</p> <ul style="list-style-type: none"> Much of the work in the pipeline is in converting object representations from one coordinate system to another – Object coordinates 	10	CO1	L4

- Camera (eye) coordinates
- Screen coordinates
- Vertex processor also computes vertex colors

Primitive Assembly

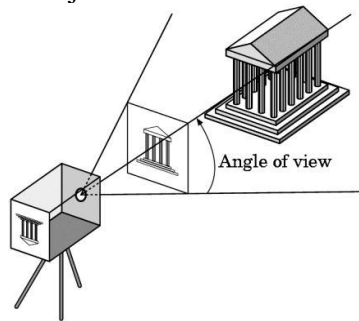
Vertices must be collected into geometric objects before clipping and rasterization can take place

- Line segments
- Polygons
- Curves and surfaces

Clipping

Just as a real camera cannot “see” the whole world, the virtual camera can only see part of the world or object space.

- Objects that are not within this volume are said to be *clipped* out of the scene



Rasterization:

- If an object is not clipped out, the appropriate pixels in the frame buffer must be assigned colors
- Rasterizer produces a set of fragments for each primitive.

	<ul style="list-style-type: none"> • Fragments are “potential pixels” <ul style="list-style-type: none"> – Have a location in frame buffer – Color and depth attributes • Vertex attributes are interpolated over objects by the rasterizer <p>Fragment Processor:</p> <ul style="list-style-type: none"> • Fragments are processed to determine the color of the corresponding pixel in the frame buffer • Colors can be determined by texture mapping or interpolation of vertex colors • Fragments may be blocked by other fragments closer to the camera 			
2	<p>Write a program to generate 3D dimensional gasket using recursive subdivision of a tetrahedron (triangle as a primitive)?</p> <pre> #include<stdio.h> #include<GL/glut.h> float vertices[4][3]={{-40,-40,-40}, {40,-40,-40}, {0,40,-40}, {0,-20,40}}; int n; void triangle(float *a, float *b, float *c) { glBegin(GL_TRIANGLES); glVertex3fv(a); glVertex3fv(b); glVertex3fv(c); glEnd(); } void d_t(float *a, float *b, float *c, int m) </pre>	10	CO2	L1

```

{
    float v1[3],v2[3],v3[3];
    if(m>0)
    {
        for(int i=0;i<3;i++)
        {
            v1[i]=(a[i]+b[i])/2;
            v2[i]=(a[i]+c[i])/2;
            v3[i]=(c[i]+b[i])/2;
        }
        d_t(c,v2,v3,m-1);
        d_t(a,v1,v2,m-1);
        d_t(b,v1,v3,m-1);
    }
    else
        triangle(a,b,c);
}
void display()
{
    glClearColor(1,1,1,1);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

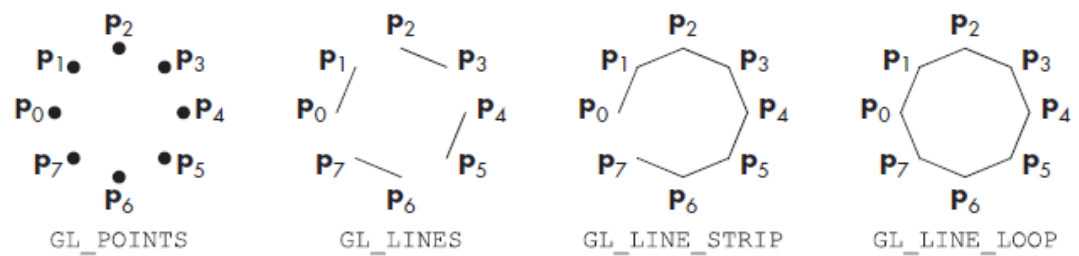
    glColor3f(1,0,0);
    d_t(vertices[0],vertices[1],vertices[3],n);
    glColor3f(0,1,0);
    d_t(vertices[0],vertices[3],vertices[2],n);
    glColor3f(0,0,1);
    d_t(vertices[1],vertices[2],vertices[3],n);
    glColor3f(1,1,0);
    d_t(vertices[0],vertices[1],vertices[2],n);

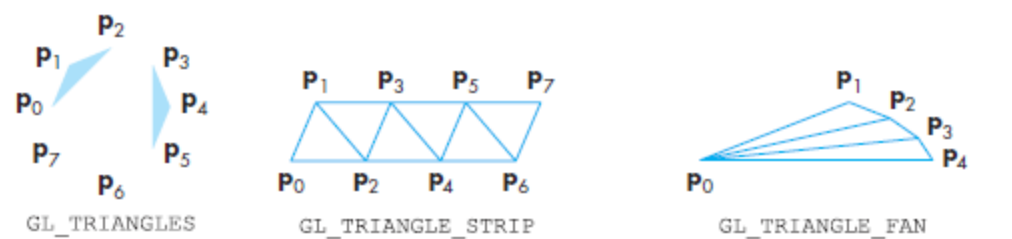
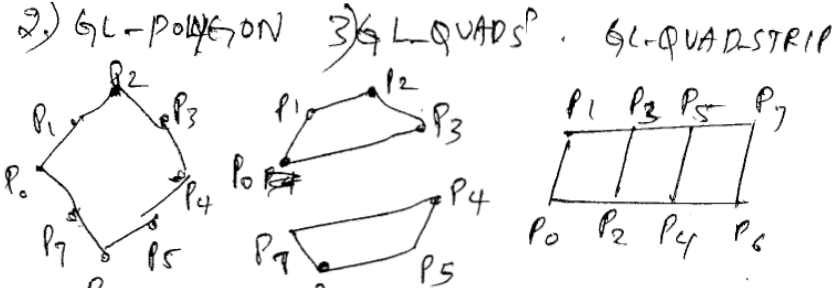
    glFlush();
}
void init()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-50,50,-50,50,-50,50);
}

```

	<pre> glMatrixMode(GL_MODELVIEW); glLoadIdentity(); } int main(int argc, char **argv) { printf("Enter the value of n: "); scanf("%d",&n); glutInit(&argc, argv); glutInitDisplayMode(GLUT_RGB GLUT_SINGLE GLUT_DEPTH); glutInitWindowPosition(0,0); glutInitWindowSize(500,500); glutCreateWindow("3D Gasket using Triangle"); glutDisplayFunc(display); init(); glEnable(GL_DEPTH_TEST); glutMainLoop(); } </pre>			
3	<p>Define view port and aspect ratio? What is display list? Give the OpenGL code that defines display list generating a red colored rectangle with vertices (50, 50), (150, 50) and (100, 50)?</p> <p>Viewport – A rectangular area of the display window, whose height and width can be adjusted to match that of the clipping window, to avoid distortion of the images. OpenGL function to set a viewport is given as below.</p> <p>void glViewport(Glint x, Glint y, GLsizei w, GLsizei h) where, x & y is the left lower corner of the viewport, w & h give the width and height of the viewport respectively.</p> <p>Aspect ratio is the ratio of width to height of a rectangle.</p> <p>The Display Processor in modern graphics systems could be considered as a graphics server. The compiled list of instructions that is sent to the display processor after processing the user program in host system were stored in a display memory called Display List.</p> <pre> # define RECT 1 glNewList(RECT, GL_COMPILE) </pre>	10	CO1	L1

	<pre>glBegin(GL_LINE_LOOP); glColor3f(1,0,0); glVertex2f(50,50); glVertex2f(150,50); glVertex2f(150,100); glVertex2f(50,100); glEnd(); glEndList();</pre> <p>GL_COMPILE_AND_EXECUTE flag can be used instead of GL_COMPILE to immediate display of the contents while the list is being constructed. GL_COMPILE tells system to send the list to server but not to display its contents.</p> <p>Whenever we wish to draw the rectangle, just a calling function is executed as below, glCallList(RECT);</p>			
4	<p>Classify the major groups of API functions of OpenGL? Explain each of them with an example?</p> <ul style="list-style-type: none"> • Primitive functions: Defines low level objects such as points, line segments, polygons etc. Ex: GL_PLOYGON, GL_LINES, GL_POINTS • Attribute functions: Attributes determine the appearance of objects. <ul style="list-style-type: none"> – Color (points, lines, polygons) Ex: glColor3f(), glPointSize() • Viewing functions: Allows us to specify various views by describing the camera's position and orientation. Ex: glOrtho(), glPerspective(), glFrustum() • Transformation functions: Provides user to carry out transformation of objects like rotation, scaling etc. Ex: glRotatef(), glTranslatef(), glScalef() • Control functions: Enables us to initialize our programs, helps in dealing with any errors during execution of the program. Ex: glutInit(), gluInitDisplayMode(), glutCreateWindow() 	10	CO1	L3

	<ul style="list-style-type: none"> • Input functions: Allows us to deal with a diverse set of input devices like keyboard, mouse etc. Ex: Mouse and Keyboard callback functions. • Query functions: Helps query information about the properties of the particular implementation. 			
5	<p>Write the different OpenGL primitives, with example for each primitive?</p> <p>Points (GL_POINTS) Each vertex is displayed at a size of at least one pixel.</p> <p>Line segments (GL_LINES) The line-segment type causes successive pairs of vertices to be interpreted as the endpoints of individual segments.</p> <p>Polylines (GL_LINE_STRIP, GL_LINE_LOOP) If successive vertices (and line segments) are to be connected, we can use the line strip, or polyline form. Many curves can be approximated via a suitable polyline. If we wish the polyline to be closed, we can locate the final vertex in the same place as the first, or we can use the GL_LINE_LOOP type, which will draw a line segment from the final vertex to the first, thus creating a closed path.</p> <div style="text-align: center;">  </div> <p>Triangles (GL_TRIANGLES) The edges are the same as they would be if we used line loops. Each successive group of three vertices specifies a new triangle.</p> <p>Strips and Fans (GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN) These objects are based on groups of triangles that share vertices and edges. In the triangle strip, for example, each additional vertex is combined with the previous two vertices to define a new triangle. A triangle fan is based on one fixed point. The next two points determine the first triangle, and subsequent triangles are formed from one new point, the previous point, and the first (fixed) point.</p>	10	CO1	L2

	 <p>GL_TRIANGLES GL_TRIANGLE_STRIP GL_TRIANGLE_FAN</p> <p>2.) GL-POLYGON 3.) GL-QUADS GL-QUADSTRIP</p> 			
6	<p>Explain RGB and Index coloring mechanisms? Write a note on viewing concept?</p> <p>RGB color:</p> <p>Each color component is stored separately in the frame buffer, usually 8 bits per component in buffer. Note in glColor3f the color values range from 0.0 (none) to 1.0 (all), whereas in glColor3ub the values range from 0 to 255. The color as set by glColor3f becomes part of the state and will be used until changed</p> <p>– Colors and other attributes are not part of the object but are assigned when the object is rendered</p> <p>RGBA color system:</p> <p>This has 4 arguments – RGB and alpha – Opacity.</p> <p><code>glClearColor(1.0,1.0,1.0,1.0)</code></p> <p>This would render the window white since all components are equal to 1.0, and is opaque as alpha is also set to 1.0</p>	10	CO2	L4

glColor3f(1.0,0.0,0.0) is used set the color state variable(red in this context) to select the rendering color of the objects that are defined after this function call.

Indexed color:

Colors are indices into tables of RGB values

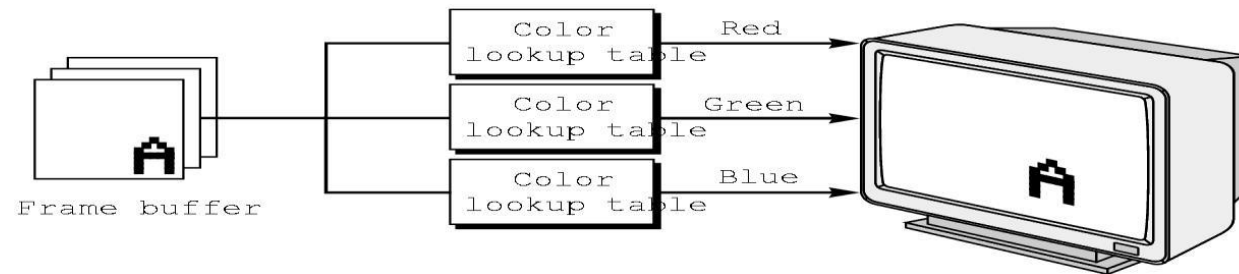
Requires less memory

o indices usually 8 bits

o not as important now

Memory inexpensive

Need more colors for shading



Classical Viewing

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

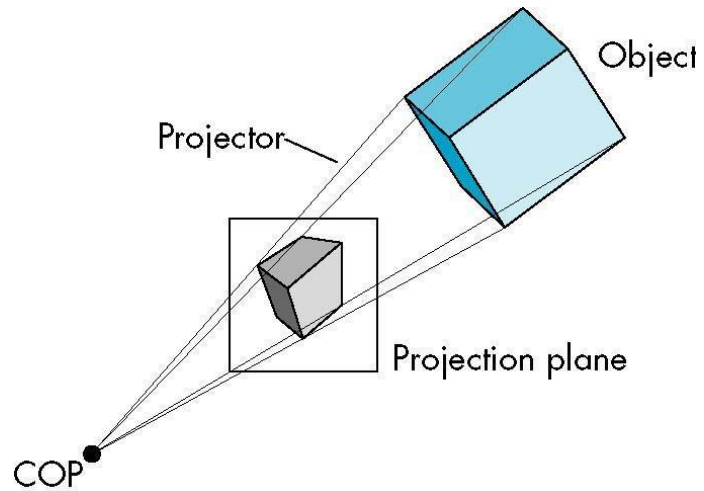
constructed from flat principal faces

- Buildings, polyhedra, manufactured objects
- Front, top and side views.

Perspective and parallel projections:

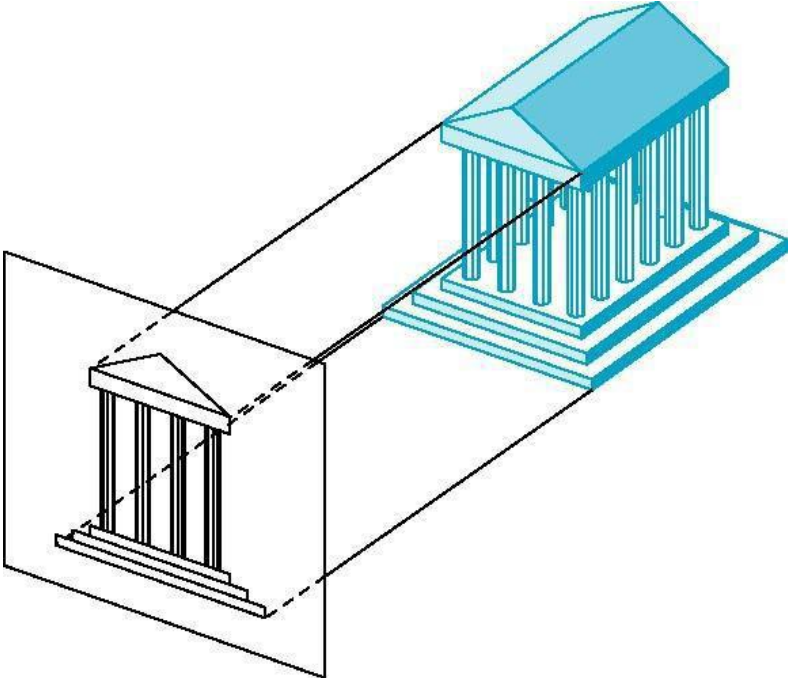
Parallel viewing is a limiting case of perspective viewing

Perspective projection has a COP where all the projector lines converge.



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

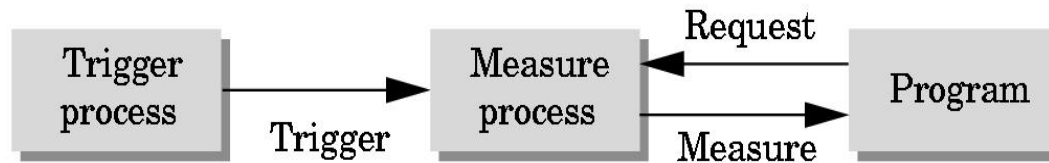
	<p>multiview orthographic projection.</p> 			
7	<p>Discuss the request mode, sample mode and event mode with figures wherever required?</p> <p>Input devices contain a <i>trigger</i> which can be used to send a signal to the operating system</p> <ul style="list-style-type: none"> o Button on mouse o Pressing or releasing a key <p>When triggered, input devices return information (their measure) to the system</p> <ul style="list-style-type: none"> o Mouse returns position information o Keyboard returns ASCII code 	10	CO3	L2

Request Mode

Measure of the device is provided to the program only when user triggers the device .

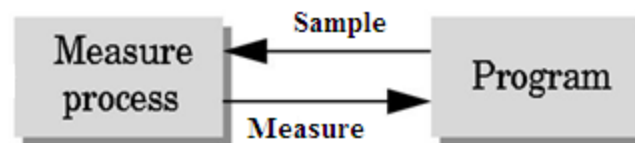
Typical of keyboard input

– Can erase (backspace), edit, correct until enter (return) key (the trigger) is depressed



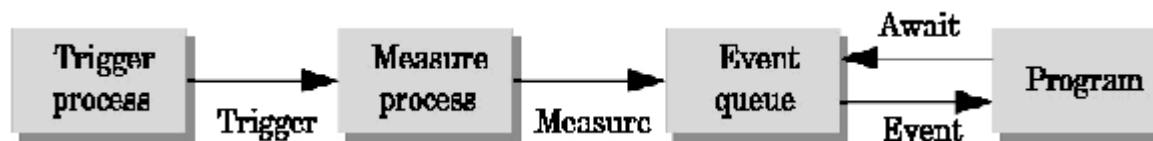
Simple Mode

In this mode the input is immediate. As soon as the function call in the user program is encountered, the measure is returned. Hence no trigger is needed.



Event Mode

Most systems have more than one input device, each of which can be triggered at an arbitrary time by a user. Each time a device is triggered, an *event* generated whose measure is put in an *event queue* which can be examined by the user program



	<p>Window: resize, expose, iconify</p> <p>Mouse: click one or more buttons Motion: move mouse</p> <p>Keyboard: press or release a key.</p>			
8	<p>Write an OpenGL program to, draw a rectangle using left click of a mouse device and the hierarchical menus to increase or decrease the size of rectangle?</p> <pre> #include<stdio.h> #include<stdlib.h> #include<GL/gl.h> #include<GL/glut.h> int wh=500,ww=500; int n=3; void display() { glClearColor(0,0,0,1); glClear(GL_COLOR_BUFFER_BIT); glFlush(); } void init() { glViewport(0,0,ww,wh); glMatrixMode(GL_PROJECTION); glLoadIdentity(); gluOrtho2D(0,ww,0,wh); glMatrixMode(GL_MODELVIEW); glColor3f(1,0,0); } void reshape(GLsizei w,GLsizei h) { glViewport(0,0,w,h); glMatrixMode(GL_PROJECTION); </pre>	10	CO3	L1

```

glLoadIdentity();
gluOrtho2D(0,w,0,h);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

ww=w;
wh=h;
}

void drawSquare(int x,int y)
{
y=wh-y;
glBegin(GL_POLYGON);
glVertex2f(x+n,y+n);
glVertex2f(x-n,y+n);
glVertex2f(x-n,y-n);
glVertex2f(x+n,y-n);
glEnd();
glFlush();

}

void size_menu(int id)
{
switch(id)
{
case 2: n=n+1;
break;
case 3: n=n-1;
break;
}
}

void main_menu(int id)
{
exit(0);
}

```

```

void mouse(int button,int state,int x,int y)
{
    if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)    drawSquare(x,y);
    if(button==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)    exit(0);
}

int main(int argc,char **argv)
{
    int sub_menu;
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Menus n Interaction");
    glutDisplayFunc(display);
    init();
    glutMouseFunc(mouse);
    sub_menu=glutCreateMenu(size_menu);
    glutAddMenuEntry("Increase Size",2);
    glutAddMenuEntry("Decrease Size",3);
    glutCreateMenu(main_menu);
    glutAddMenuEntry("Quit",2);
    glutAddSubMenu("Resize",sub_menu);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutKeyboardFunc(keyboard);
    glutReshapeFunc(reshape);
    glutMainLoop();
}

```

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