TRIBHUVAN UNIVERSITY

Institution of Science and Technology

Course Title: Computer Graphics
Course No: CSC 209
Nature of the Course: Theory + Lab
Semester: III

Full Marks: 60
Pass Marks: 24
Time: 3 hrs

Computer Graphics

TU QUESTIONS-ANSWERS 2075

Section A

Attempt any two questions. **(2 ×10 =20)**

1. Define window, viewport and viewing transformation. Let ABCD be the rectangular window with A(20, 20), B(90, 20), C(90, 70), D(20, 70). Find the region codes for end points and use Cohen Sutherland algorithm to clip the lines P(10, 20), Q(80, 90). **3+7=10**

Ans: Window

A world-coordinate area selected for display is called a window. That is, window is the section of the 2D scene that is selected for viewing. The window defines what is to be viewed.

Viewport

An area on a display device to which a window is mapped is called a viewport. The viewport indicates where on an output device selected part will be displayed. Figure given below illustrates the mapping of a 2D picture that falls within a rectangular window onto a designated rectangular viewpoint.

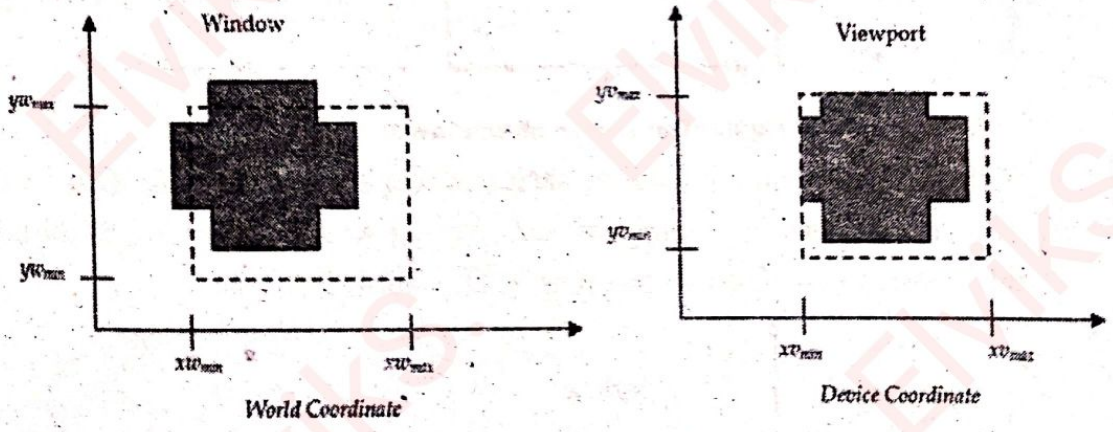


Figure: World and Viewport Coordinates, specified as rectangles aligned with the co-ordinate axes.

Window and viewport are often rectangular in standard positions, because it simplifies the transformation process and clipping process. Other shapes such as polygons, circles take longer time to process.

Viewing transformation

Viewing Transformation is the mapping of coordinates of points and lines that form the picture into appropriate coordinates on the display device. The

window-to-viewport transformation can be explained in three steps as below:

1. Translate object along with window such that the lower left corner of the window is at the origin. That is, apply T $(-xw_{min}, -yw_{min})$.

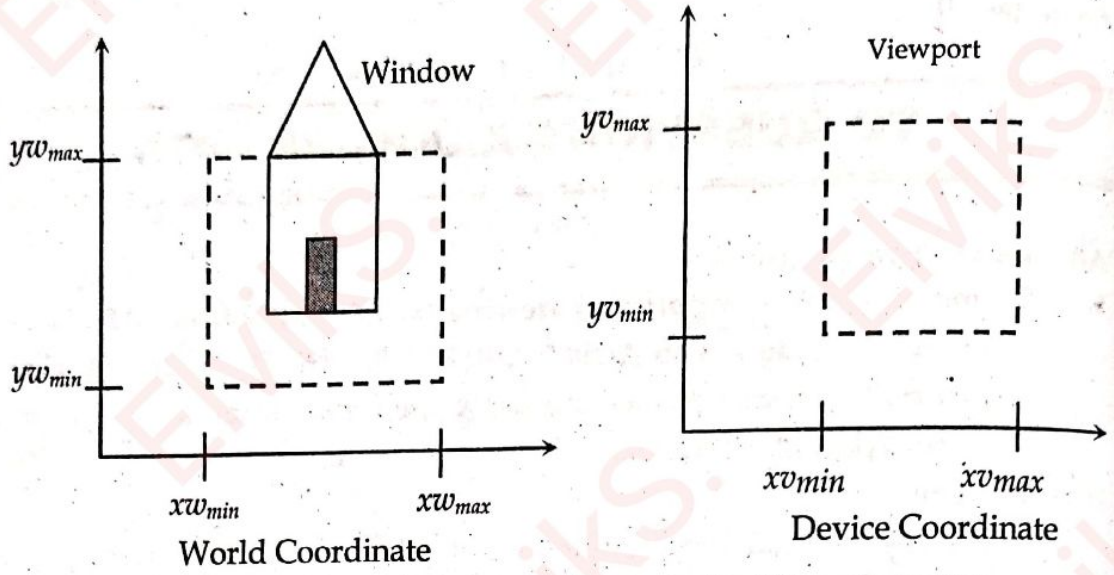


Figure: Window port and viewport before coordinate transformation

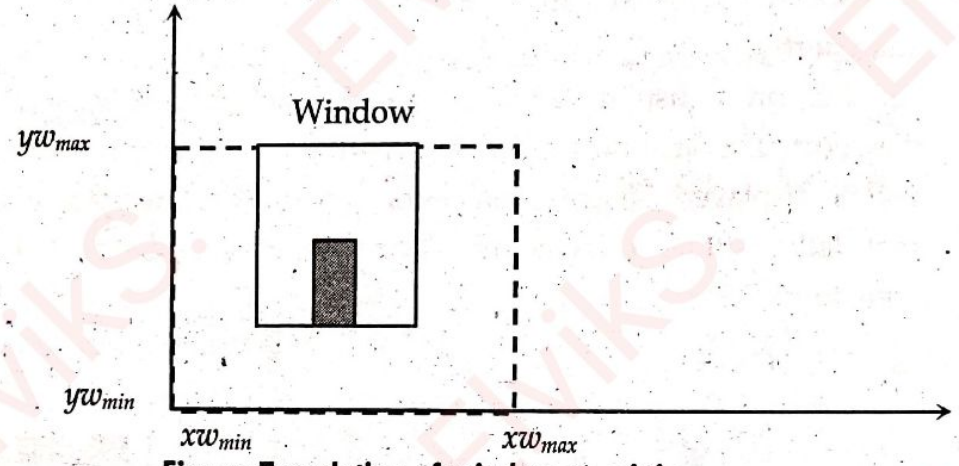


Figure: Translation of window to origin

2. Scale the object and the window such that window has the same dimension as that of a viewport. Simply we can say that we are converting the object into image and window in viewport. That is, apply S (sx, sy).

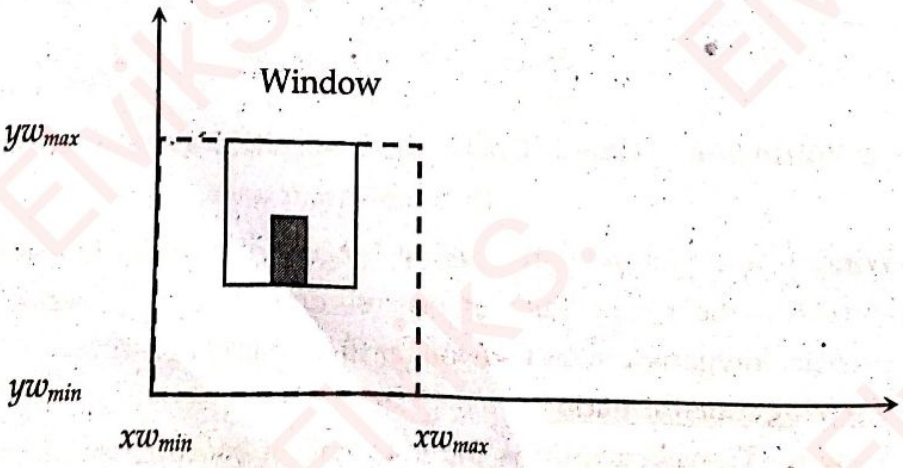


Figure: Scaling the window to the size of view port

3. Finally, apply another translation to move the viewport to its original position on the screen. That is, apply $T(xv_{min}, yv_{min})$.

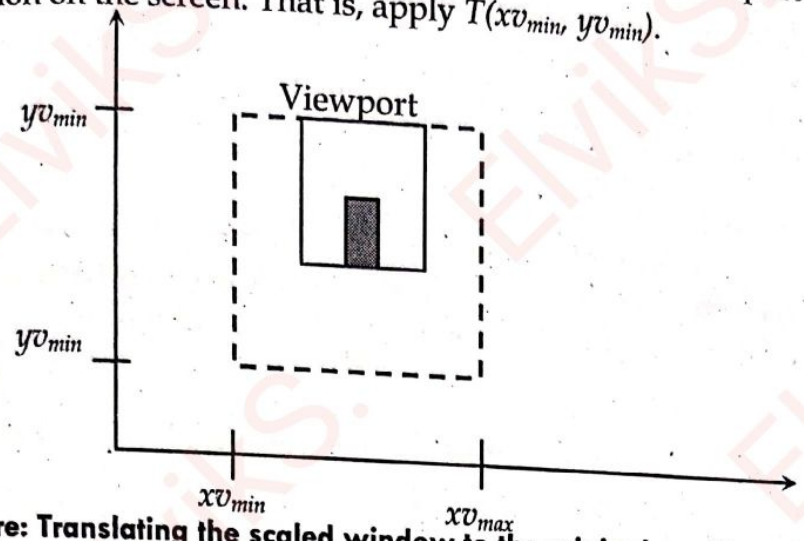


Figure: Translating the scaled window to the original position of viewport.

Therefore, net transformation (Composite Transformation),

$T_{WV} = T(xv_{min}, yv_{min}).\ S(S_x, S_y).T(-xw_{min}, -yw_{min})$ .....(i)

Here, Sx and Sy are scaling factors, where;

$$Sx = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} \qquad Sy = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

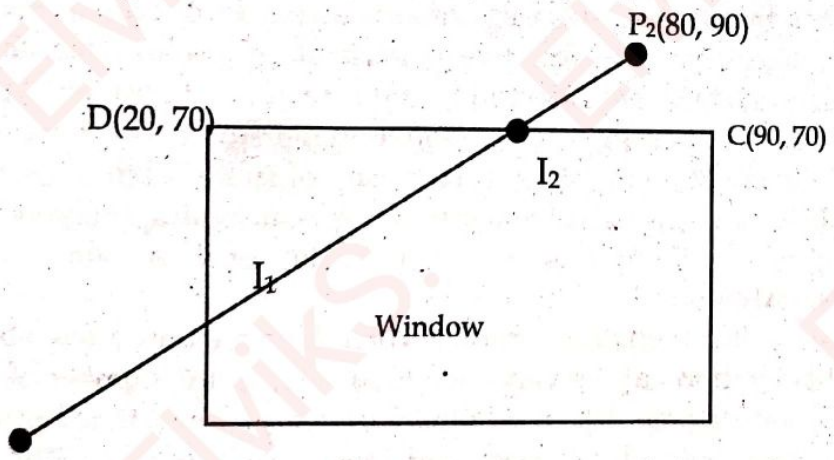
Now, writing in matrix form;

$$Twv = \begin{bmatrix} 1 & 0 & xv_{min} \\ 0 & 1 & yv_{min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -yw_{min} \\ 0 & 1 & -yw_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

This is required window-to-viewpoint transformation. Let $P(x, y)$ be the world coordinate point that is mapped onto the viewport point $P'(u, v)$, then we must have:

$P' = TWV.P$

Numerical Part



Since the point P_1 lies left outside the window hence its outcode is 0001

Similarly the outcode of P_2=1000

The outcode for I_1 is 0001

And outcode for I_2 is 1000

The Logical AND for P_1P_2 = 0001 AND 1000 ≠ 0000.

Which is zero hence line is partially visible.

Since left edge of window is having x coordinate as 20. The intersection points z-coordinate will also be 20 only. Now we have to find y-coordinate of intersection point. For this we will use slope intercept form of line.
i.e. let's find slope of line P_1P_2,

Hence, m = = = = 0.8

So, slope of line p_1p_2 is 0.8
We know that the slope between two endpoints of a line is same as that of slope between one endpoint and any other point which is lying on the line.
Hence slope between points P1 and intersection point will be,

m =

where, x1 and y1 are related with P_1, x_2 is known i.e. 20, y_2 we have to find.
Hence, $m(x_2 - x_1) = y_2 - y_1$
⇨ $y_2 = m(x_2 - x_1) + y1$
⇨ $y_2 = 0.8\ (20 - 10) + 30$
$= 0.8 * 10 + 30 = 8 + 30 = 38$
⇨ $y_2 = 38$
Therefore the intersection point is (20, 38). Let's call this point as I1. Since I1 point is exactly at the boundary. Its region code will be (0, 0, 0, 0). Now we have to discard the line segment $P_1\ I_1$ since P_1 point is outside window.
Similarly we have to find another intersection point I_2 with respect to top boundary. Now I_2 points region.
Code will be 0000. So we will discard I_2P_2 since P_2 point is above the window.
Now our line segment is I_1I_2. Both I_1 and I_2 's region codes are 0000. So the line I_1I_2 is completely visible and we have to display I_1I_2 line.

2. **List any two advantages of BSP tree method in visible surface detection. Make a compression between Painter's algorithm and A-Buffer algorithm. (2+8=10)**

Ans: A binary space partitioning (BSP) tree is an efficient method for determining object visibility by painting surfaces onto the screen from back to front as in the painter's algorithm. The BSP tree is particularly useful when the view reference point changes, but object in a scene are at fixed position. Applying a BSP tree to visibility testing involves identifying surfaces that are "inside" or "outside" the partitioning plane at each step of space subdivision relative to viewing direction. It is useful and efficient for calculating visibility among a static group of 3D polygons as seen from an arbitrary viewpoint.

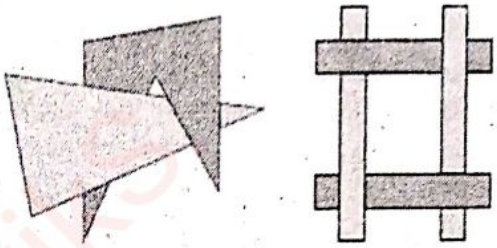
Depth Sorting Method

The basic idea of this method is simple. When there are only a few objects in the scene, this method can be very fast. However, as the number of objects increases, the sorting process can become very complex and time consuming. This method uses both object space and image space method. The depth-sorting method performs two basic functions:

- First, the surfaces are sorted in order of decreasing depth.
- Second, the surfaces are scan-converted in order, starting with the surface of greatest depth.

The intensity values for farthest surface are then entered into the refresh buffer. That is farthest polygon is displayed first, then the second farthest polygon, so on, and finally, the closest polygon surface. After all surfaces have been processed, the refresh buffer stores the final intensity values for all

visible surfaces. When there are only a few objects in the scene, this method can be very fast. However, as the number of objects increases, the sorting process can become very complex and time consuming. The scan conversion of the polygon surfaces is performed in image space. This method for solving the hidden-surface problem is often referred to as the painter's algorithm. The following figure shows the effect of depth sorting:



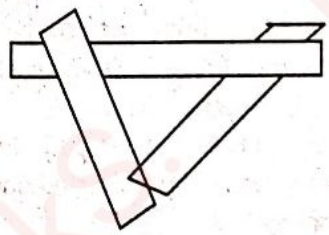
In this method, the newly displayed surface is partly or completely obscures the previously displayed surface. Essentially, we are sorting the surface into priority order such that surface with lower priority (lower z, far objects) can be obscured by those with higher priority (high z-value).

Advantages

- The sorting computation is very fast, so a quick calculation of the image display is possible.
- The finished image data remains object based (sorted polygons), and can be edited in terms of line weight, exploded views, etc.
- It will also print at the highest resolution available on devices such as postscript laser printers.

Limitations

- Certain conditions, such as intersecting polygons or cyclic overlap (A covers part of B, which covers part of C, which covers part of A), are not processed correctly, since parts of each should be visible, but the algorithm will always yield a sorted list. The problem described as shown in figure below.



- Different polygons may have same depth.
- The nearest polygon could also be farthest.
- We cannot use simple depth-sorting to remove the hidden-surfaces in the images.

A-Buffer method

A-Buffer method is an extension of depth buffer method. A drawback of the depth buffer method is that it can only find one visible surface at each pixel position. That means, it deals only with opaque surface and cannot accumulate intensity values for more than one surface, as is necessary if transparent surfaces are to be displayed.

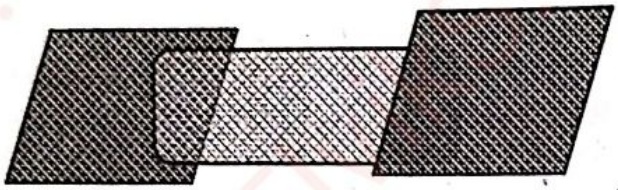


Figure: Viewing two background opaque surfaces through a foreground transparent surface

The A-buffer method expands the depth buffer so that each position in the buffer can reference a linked list of surfaces. Thus more than one surface intensity can be taken into consideration at each pixel position. Each pixel position in the A-buffer has following fields:

- **Depth field:** It stores a positive or negative real numbers
- **Intensity field:** It stores surface intensity information or a pointer value.

If the depth field is positive, the number stored at that position is the depth of a single surface overlapping the corresponding pixel area. If the depth field is negative, this indicates multiple surface contributions to the pixel intensity.

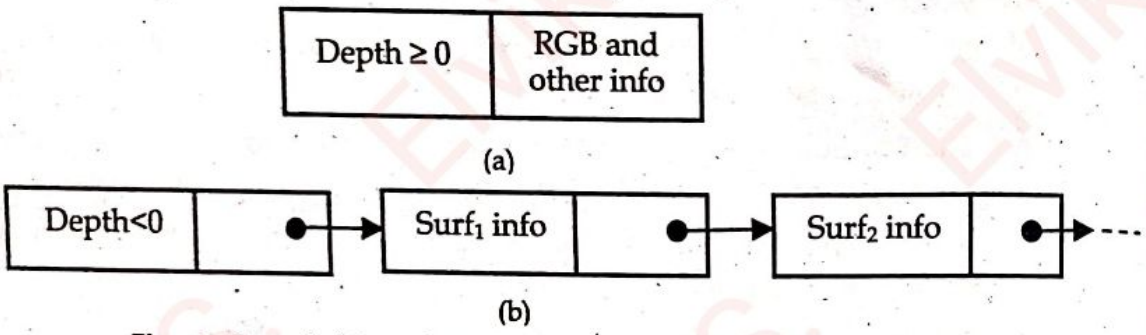


Figure: Organization of an A-buffer pixel position: (a) single-surface (b) multiple surfaces overlap

As shown in the above figure (a), if the value of depth is >= 0, the number stored at that position is the **depth of single surface** overlapping the corresponding pixel area. The 2nd field, i.e., the intensity field then stores the RGB components of the surface color at that point and the percent of pixel coverage.

As shown in the above figure (b), **multiple-surface contributions** to the pixel intensity are indicated by depth < 0. The 2nd field, i.e., the intensity field then stores a pointer to a linked list of surface data.

3. **Describe the architecture of raster scan display. Explain about sweep, octree and boundary representations for solid modeling.** **(4+6=10)**

Ans: Architecture of Raster-Scan System

The raster graphics systems typically consist of several processing units. CPU is the main processing unit of computer systems. Besides CPU, graphics system consists of a special buffer processor called video controller or display controller. It is used to control the operation of the display device. In addition, to the video controller, raster scan systems can have other processors as co-processors which are called graphic controller or display processors. A fixed area of system memory is reserved for the frame buffer. The video controller has the direct access to the frame buffer for refreshing the screen. The video controller cycles through the frame buffer, one scan line at a time, typically at 60 times per second or higher. The contents of

frame buffer are used to control the CRT beam's intensity or color. The organization of simple raster system is shown in the figure below.

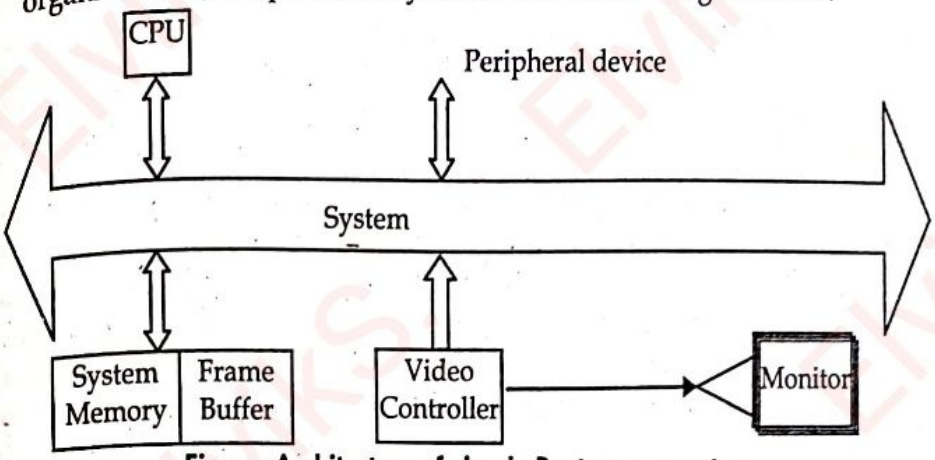
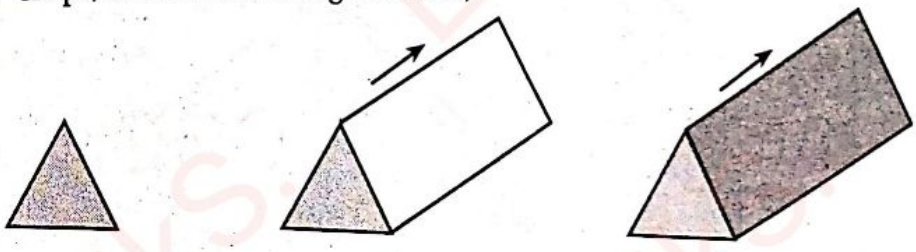


Figure: Architecture of simple Raster-scan system.

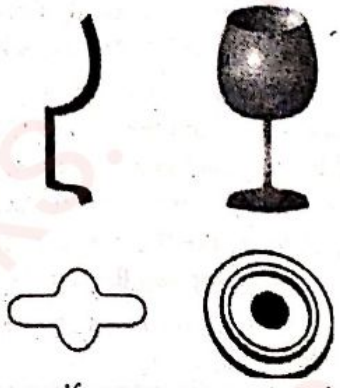
Sweep Representations

Sweep representations are used to construct three dimensional objects from two dimensional shape .There are two ways to achieve sweep: **Translational sweep and Rotational sweep.** In translational sweeps, the 2D shape is swept along a linear path normal to the plane of the area to construct three dimensional objects. To obtain the wireframe representation we have to replicate the 2D shape and draw a set of connecting lines in the direction of shape, as shown in the figure below,



(b) 3D shape (c) Wire frame representation

In rotational sweeps, the 2D shape is rotated about an axis of rotation specified in the plane of 2D shape to produce three dimensional objects. This is illustrated in figure below,



In general we can specify sweep constructions using any path. For translation we can vary the shape or size of the original 2D shape along the sweep path. For rotational sweeps, we can move along a circular path through any angular distance from 0° to 360°. These sweeps whose generating area or volume changes in size, shape or orientation as they are swept and that follow an arbitrary curved trajectory are called general

sweeps .General sweeps are difficult to model efficiently for example, the trajectory and object shape may make the swept object intersect itself, making volume calculations complicated. Furthermore, general sweeps do not always generate solids. For example, sweeping a 2D shape in its own plane generates another 2D shape.

Octree

Octree are three dimensional quadtrees, its three dimensions are recursively subdivided into octants and have quadrants. The number of nodes in a quadtree or octree is proportional to the object's perimeter or surface, respectively because subdivision occurs only from the need to represent an object's boundary. Therefore, subdivision only occurs in those quadrants where a boundary passes. Boolean set operators can also apply to both quadtrees and octrees by traversing the two trees in parallel. Individual partitions of 3D space are called Voxels (Volume Elements). Octrees are hierarchical tree structures used to represent solid objects Octrees are particularly useful in applications that require cross sectional views - for example medical applications. Octrees are typically used when the interior of objects is important. Octrees are usually applied in ray casting and shadow casting.

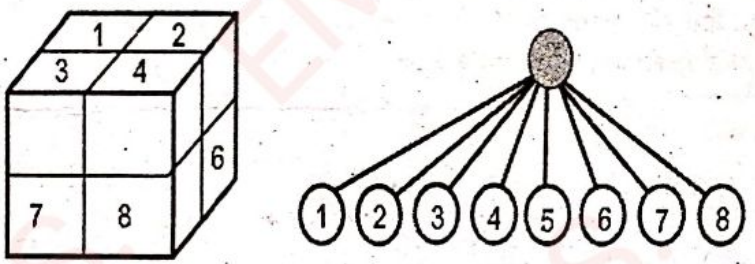


Figure: Octree

Boundary representations for solid modeling

Boundary representation is one of the two most popular and widely used schemes to create solid models of physical objects. A B-rep model or boundary model is based on the topological notion that a physical object is bounded by a set of faces. These faces are regions or subsets of closed and orientable surface. A closed surface is one that is continuous without breaks. An orientable surface is one in which it is possible to distinguish two sides by using the direction of the surface normal to point to the inside or outside of the solid model under construction. Each face is bounded by edges and each edge is bounded by vertices. Thus, topologically, a boundary model of an object is comprised of faces, edges, and vertices of the object linked together in such a way as to ensure the topological consistency of the model.

Section B

Short Answer Questions

Attempt any *EIGHT* questions **[8 × 5 = 40]**

4. Given some basic color model. Give the basic command to draw the pixel and polygon in OpenGL. **(2+3)**

Ans: Basic color model

A color space or color model is a method by which we can specify, create and visualize color. A color model is a method for explaining the properties or behavior of color within particular context. It is a mathematical way of representing a set of colors. A color model is an orderly system for creating a

whole range of colors from a small set of primary colors. There is several established color models used in computer graphics, RGB, CMY, CMYK, YIQ etc. But the two most common are the RGB model (Red-Green-Blue) for computer display and the CMYK model (Cyan-Magenta-Yellow-black) for printing. Several Color models, but fall into two broad categories:

- Additive Color Model and
- Subtractive Color Model

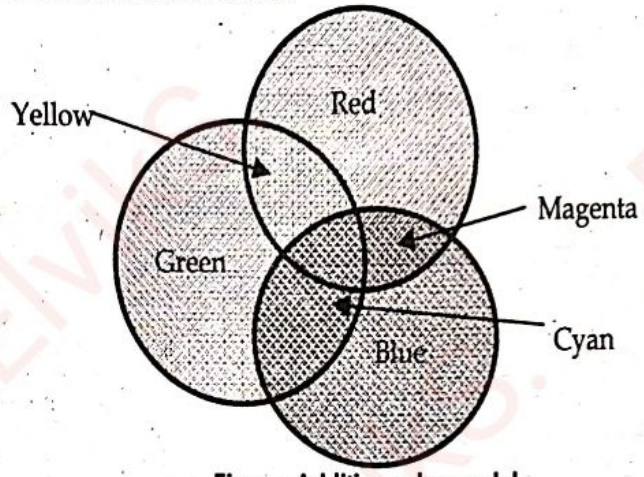


Figure: Additive color model

Points

A point is represented by a set of floating-point numbers called a vertex. All internal calculations are done as if vertices are three-dimensional. Vertices specified by the user as two-dimensional (that is, with only *x* and *y* coordinates) are assigned a *z* coordinate equal to zero by OpenGL. OpenGL works in the homogeneous coordinates of three-dimensional projective geometry, so for internal calculations, all vertices are represented with four floating-point coordinates (x, y, z, w). If w is different from zero, these coordinates correspond to the Euclidean three-dimensional point (x/w, y/w, z/w).

To control the size of a rendered point, use **glPointSize()** and supply the desired size in pixels as the argument.

void **glPointSize**(GLfloat size);

Sets the width in pixels for rendered points; size must be greater than 0.0 and by default is 1.0.

Polygon

Polygons are typically drawn by filling in all the pixels enclosed within the boundary, but you can also draw them as outlined polygons or simply as points at the vertices. A filled polygon might be solidly filled or stippled with a certain pattern. Although the exact details are omitted here, filled polygons are drawn in such a way that if adjacent polygons share an edge or vertex, the pixels making up the edge or vertex are drawn exactly once—they're included in only one of the polygons. This is done so that partially transparent polygons don't have their edges drawn twice, which would make those edges appear darker (or brighter, depending on what color you're drawing with).

A polygon has two sides—front and back—and might be rendered differently depending on which side is facing the viewer. This allows you to have cutaway views of solid objects in which there is an obvious distinction between the parts that are inside and those that are outside. By default, both

front and back faces are drawn in the same way. To change this, or to draw only outlines or vertices, use glPolygonMode().

We can draw polygon by following code of segment;

```
glBegin(GL_POLYGON); // Draw A Quad
glVertex3f(-0.5f, 1.0f, 0.0f); // Top Left
glVertex3f(-1.0f, 0.0f, 0.0f); // Left
glVertex3f(-0.5f, 1.0f, 0.0f); // Bottom Left
glVertex3f(0.5f, 1.0f, 0.0f); // Top Right
glVertex3f(1.0f, 0.0f, 0.0f); // Right
glVertex3f(0.5f, 1.0f, 0.0f); // Bottom Right
glEnd();
```

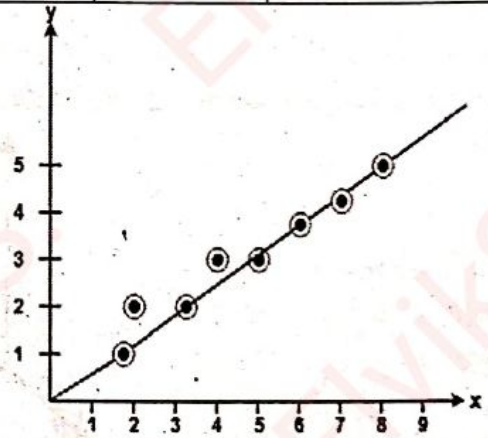
Complete example of drawing a polygon in OpenGL by using c/c++ as below:

```
#include <windows.h>
#include <stdio.h>
include <conio.h>
#include <iostream.h>
#include <stdlib.h>
#include <glut.h>
void myInit(void)
{
glClearColor(1.0,1.0,1.0,0.0);
glColor3f(1.0f,0.0f,0.0f);
glPointSize(4.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0,800.0,0.0,600.0);
}
void myDisplay(void)
{
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_POLYGON);
glVertex2i(100,100);
glVertex2i(100,300);
glVertex2i(400,300);
glVertex2i(600,150);
glVertex2i(400,100);
glEnd(); glFlush();
}
int main(int argc,char * argv[])
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(800,600);
glutInitWindowPosition(100,100);
glutCreateWindow("opengl Window");
glutDisplayFunc(myDisplay);
myInit(); glutMainLoop();
getch();
return 0;
}
```

5. **Trace the Bresenham's Line drawing algorithm for the end points (1, 1) and (8, 5).** (5)

Ans: x1=1
y1=1
x2=8
y2=5
dx= x2-x1=8-1=7
dy=y2-y1=5-1=4
I1=2* Δy=2*4=8
I2=2*(Δy-Δx)=2*(4-7)=-6
d = I1-Δx=8-7=1

x	y	d=d+I_1 or I_2
1	1	d+I_2=1+(-6)=-5
2	2	d+I_1=-5+8=3
3	2	d+I_2=3+(-6)=-3
4	3	d+I_1=-3+8=5
5	3	d+I_2=5+(-6)=-1
6	4	d+I_1=-1+8=7
7	4	d+I_2=7+(-6)=1
8	5	



6. **Derive the relation for tree-dimensional translation and rotation.** (5)

Ans: Translation

Translation is used to move a point, or a set of points, linearly in space. Since now we are talking about 3D, therefore each point has 3 coordinates i.e. x, y and z. Similarly, the translation distances can also be specified in any of the 3 dimensions. These Translation Distances are given by t_x, t_y and t_z.

For any point P(x, y, z) after translation we have P'(x', y', z') where

$x' = x + t_x$,
$y' = y + t_y$,
$z' = z + t_z$

And (t_x, t_y, t_z) is Translation vector.

Now this can be expressed as a single matrix equation.

$P' = P + T$

Where,

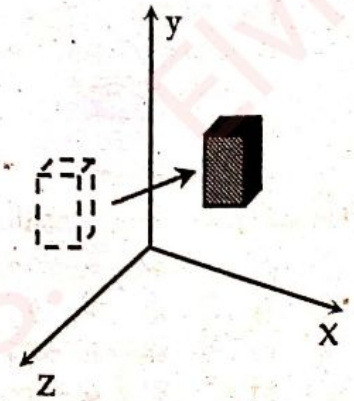


Fig: 3D Translation

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad P' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}, \text{ and } T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Rotation

Rotation is the process of moving a point in space in a non-linear manner. More particularly, it involves moving the point from one position on a sphere whose center is at the origin to another position on the sphere. 3D rotation is not same as 2D rotation. In 3D rotation, we have to specify the angle of rotation along with the axis of rotation. We can perform 3D coordinate axes rotation as

- Z-axis rotation (Roll)
- Y-axis rotation (Yaw)
- X-axis rotation (Pitch)

Z-axis rotation (Roll)

If you look closely, you should note that when we rotate around the Z axis, the Z element of the point does not change. In fact, we can just ignore the Z – we already know what it will be after the rotation. If we ignore the Z element, then we have the same case as if we were rotating the two-dimensional point <x, y> through the angle θ. Z-axis rotation is same as the origin about the 2D for which we have the derived matrices already.

$x' = x\cos\theta - y\sin\theta$

$y' = x\sin\theta - y\cos\theta$

$z' = z$.

Homogeneous representation is:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

i.e. $P' = R_z(\theta).P$

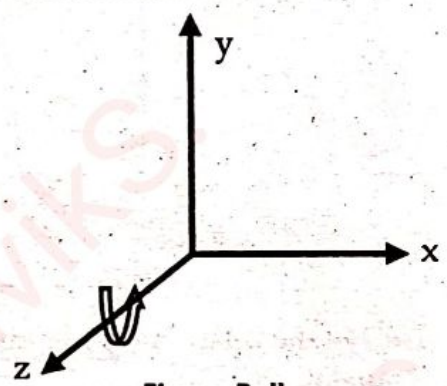


Figure: Roll

Y-axis rotation (Yaw)

The equations for Y-axis rotation

$x' = x\cos\theta + z\sin\theta$

$y' = y$

$z' = z\cos\theta - x\sin\theta$

Homogeneous representation is:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

i. e. $P' = R_y(\theta).P$

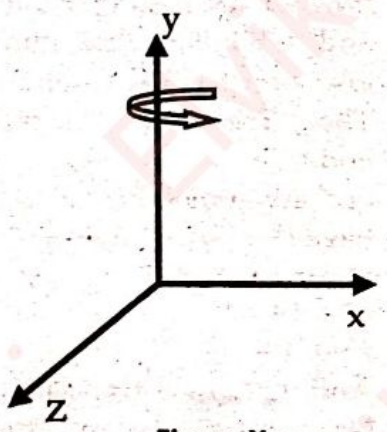


Figure: Yaw

X-axis rotation (Pitch)

The equations for X-axis rotation

$x' = x$

$y' = y\cos\theta - z\sin\theta$

$z' = y\sin\theta + z\cos\theta$

Homogeneous representation is:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & .0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

i.e. $P' = R_x(\theta).P$

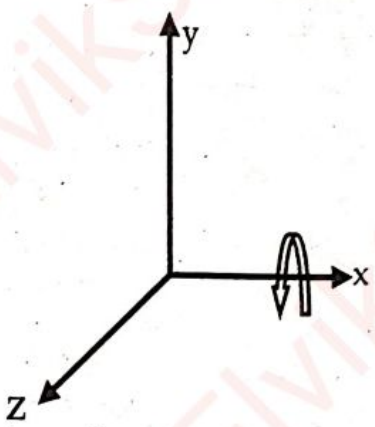


Figure: Pitch

7. What is the purpose of wireframe representation? Describe about boundary and space partitioning. (2+3)

Ans: A wireframe is a three-dimensional model that only includes vertices and lines. It does not contain surfaces, textures, or lighting like a 3D mesh. Instead, a wireframe model is a 3D image comprised of only "wires" that represent three-dimensional shapes.

Wireframes provide the most basic representation of a three-dimensional scene or object. They are often used as the starting point in 3D modeling since they create a "frame" for 3D structures. For example, a 3D graphic designer can create a model from scratch by simply defining points (vertices) and connecting them with lines (paths). Once the shape is created, surfaces or textures can be added to make the model appear more realistic.

Objects are represented as a collection of surfaces. 3D object representation is divided into two categories.

- Boundary Representations B–reps – It describes a 3D object as a set of surfaces that separates the object interior from the environment.
- Space–partitioning representations – It is used to describe interior properties, by partitioning the spatial region containing an object into a set of small, non-overlapping, contiguous solids usually cubes.

The most commonly used boundary representation for a 3D graphics object is a set of surface polygons that enclose the object interior. Many graphics system use this method. Set of polygons are stored for object description. This simplifies and speeds up the surface rendering and display of object since all surfaces can be described with linear equations.

The polygon surfaces are common in design and solid-modeling applications, since their wireframe display can be done quickly to give general indication of surface structure. Then realistic scenes are produced by interpolating shading patterns across polygon surface to illuminate.

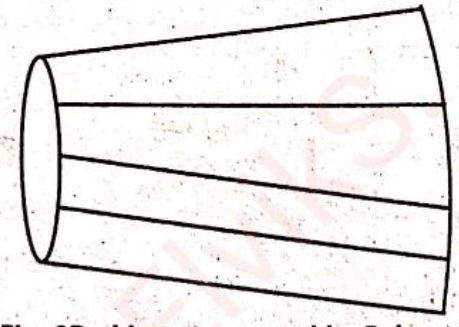


Fig: 3D object represented by Polygons

8. **Plot the ellipse centered at (0, 0) with radios $r_x = 8$ and $r_y = 6$, using midpoint ellipse drawing algorithm.** (5)

Ans: Input ellipse parameters $r_x = 8$ and $r_y = 6$ the midpoint ellipse algorithm by determining raster position along the ellipse path is the first quadrant. Initial values and increments for the decision parameter calculations are:

$2r_{y2}^2 x = 0$ (with increment $2r_y^2 = 72$)

$2r_x^2 y = 2r_x^2 r_y$ (with increment $- 2r_x^2 = -128$)

For region 1 the initial point for the ellipse centered on the origin is $(x_0, y_0) = (0, 6)$ and the initial decision parameter value is:

$p_{10} = r_y^2 - r_x^2 r_y^2 + 1/4 r_x^2 = -332$

Successive midpoint decision parameter values and the pixel positions along the ellipse are listed in the following table.

k	pl_k	x_{k+1}, y_{k+1}	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	– 332	(1, 6)	72	768
1	– 224	(2, 6)	144	768
2	– 44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	– 108	(5, 5)	360	640
5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

Move out of region 1, $2r_y^2 x > 2r_x^2 y$

For a region 2 the initial point is $(x_0, y_0) = (7, 3)$ and the initial decision parameter is,

$p_{20} = f_{ellipse}(7+1/2, 2) = -151$

The remaining positions along the ellipse path in the first quadrant are then calculated as,

k	$p2_k$	x_{k+1}, y_{k+1}	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	– 151	(8, 2)	576	256
1	233	(8, 1)	576	128
2	745	(8, 0)	–	–

9. **Define clipping. Discuss about cubic spline interpolation.** (2+3)

Ans: Clipping

The process of discarding (cutting off) those parts of a picture which are outside of a specified region (windows) is called clipping. Any procedure that identifies those parts of a picture that are either inside or outside of the specified region is called a clipping algorithm. The region against which the clipping operation is performed is called a clip window.

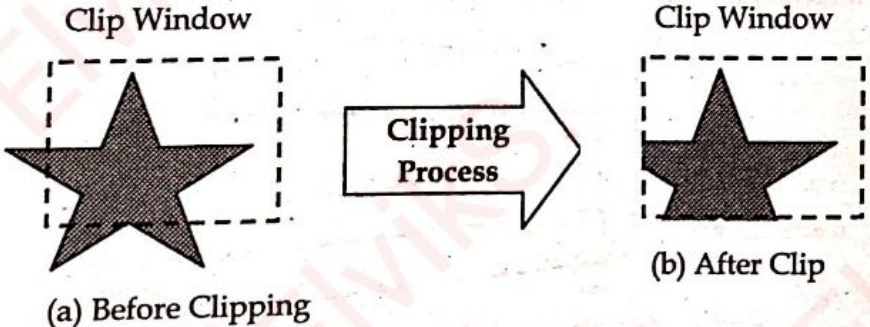


Figure: Clipping process (a) before clipping (b) after clipping

Cubic spline interpolation

The cubic spline was originally introduced by James Ferguson. Spline was a term for elastic rulers that were bent to pass through a number of predefined points ("knots"). These were used to make technical drawings for shipbuilding and construction by hand. Cubic spline interpolation is a special case for Spline interpolation that is used very often to avoid the problem of Runge's phenomenon. This method gives an interpolating polynomial that is smoother and has smaller error than some other interpolating polynomials such as Lagrange polynomial and Newton polynomial. Cubic polynomials offer a reasonable compromise between flexibility and speed of computation. Cubic spline requires less calculations compares to higher order polynomials and less memory. Compared to lower polynomials cubic spline are more flexible for modeling arbitrary curve shape.

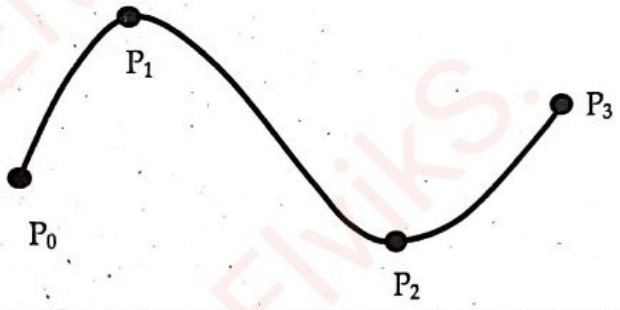


Figure: Interpolation with cubic splines between 4 control points

Given a set of control points, cubic interpolation splines are obtained by fitting the input points with a piecewise cubic polynomial curve that passes through every control points.

Suppose we have n+1 control points specified with co-ordinates

$p_k = (x_k, y_k, z_k),\ k = 0,1,2.....n$

The parametric cubic polynomial that is to be filled between each pair of control points with the following set of equations.

$$x(u) = a_x u^3 + b_x u^2 + c_x u + dx$$

$$y(u) = a_y u^3 + b_y u^2 + c_y u + dy \quad (0 \le u \le 1)$$

$$z_u(u) = a_z u^3 + b_z u^2 + c_z u + dz$$

For these three equations, we need to determine the values for the four coefficients a, b, c, and d in the polynomial representation for each of the n curve sections between the n+1 control points. We do this by setting enough boundary conditions at the control-point positions between curve sections so that we can obtain numerical values for all the coefficients.

10. How can we detect shadows in computer graphics? List the challenged in computing light and manipulating interfaces in virtual reality? Justify. (2+3)

Ans: Shadows appear in many scenes. Human can easily distinguish shadows from objects, but it is one of the challenges for shadow detection intelligent automated systems. Accurate shadow detection can be difficult due to the illumination variations of the background and similarity between appearance of the objects and the background. Color and edge information are two popular features that have been used to distinguish cast shadows

from objects. However, this become a problem when the difference of color information between object, shadow and background is poor, the edge of the shadow area is not clear and the shadow detection method is supposed to use only color or edge information method.

Challenges of virtual reality are listed below:

VR as a New Medium

We should say a word of warning here before continuing: adapting existing applications to VR is difficult if they weren't designed for this from the outset. VR is like radio or TV at their beginnings: radio was only used to broadcast opera, and TV was only used to broadcast theater plays. Slowly, people started to create content specifically tailored for these new media. Camera movement, zoom, and cuts created a new grammar for film, for instance.

Latency

The first enemy of VR is latency. If you move your head in the real world and the resulting image takes one second to appear, your brain will not accept that this image is related to the head movement. Moreover as a result, you will probably get sick.

Interactions

Interactions with a 3D/VR world are more complex than it seems. There are even several scientific conferences dedicated to this sole topic: 3D User Interfaces (3DUI), Spatial User Interfaces (SUI). The main problem is that you don't have access to a regular keyboard or mouse. The second problem is that interacting in 3D is a hard ergonomic problem!

There are multiple ways of interacting in VR:

- Navigation in an environment,
- Selecting and manipulating an object,
- Menus and graphical user interfaces (GUIs),
- Entering numbers and text,

All those tasks can be accomplished in a lot of different ways which depend on a particular application, hardware and even user! Think of the navigation: it can be done with a simple joystick, by pointing a particular destination in space with a hand-held device, by saying out loud the destination, by walking in place, by gestures, by looking at it, by picking it on a map.

Avatars

When you are using a head-mounted display (HMD), you are completely immersed in the virtual world, and you don't see your own body anymore. It is very important to display a virtual representation of yourself and others, called avatars. They can be realistic, look like yourself, or be completely different.

If your VR system offers full body tracking and if you want an avatar that has exactly the same dimensions as you, this can be a simple task. But if your VR system only has a few trackers and you want an avatar that is taller or smaller than you, it can be difficult to extrapolate the position of the limbs that are not tracked and to adapt your body posture to a different virtual body.

Collaboration

Once you start being in a virtual world, you very quickly want to share the experience and not be alone. VR is particularly suited for collaborative work with people physically in the same space or in completely different parts of

the world! When creating a collaborative application, you need to make sure each VR application is connected to one server, that all data between the applications are synchronized securely, that you can see the avatar of others.

High-end VR systems

There are specific issues when dealing with high-end VR systems such as CAVEs(tm), domes, workbenches etc:

- computing the correct perspective depending on the position of the user,
- Displaying different types of stereoscopy (active, passive, auto stereoscopic, side-by-side etc.),
- managing multi-screens and/or multiple graphics cards,
- managing multiple computers and the synchronization of the application (Frame lock), display of new images (Swap lock) and display of stereoscopic images (Gen lock)
- managing the warping and blending of several projectors projecting on a non-planar surface,
- haptics force-feedback, etc.

Running on different VR hardware

When your application is finished, you might want to run it on a different hardware: You might have created your application for the Oculus Rift and want to try with another HMD, or maybe a stereoscopic wall or a Cave.

A Coherent World, Not Necessarily a Realistic One

We have seen that perceptive presence requires you to fool your senses in the most realistic way. Cognitive presence – fooling the mind, not the senses – results from a sense that your actions have effects on the virtual environment, and that these events are credible. This means that you must believe in the "rules" of the simulation. For this, you must make sure that your world is coherent, not necessarily realistic. If a player can grab a particular glass, for example, but can't grab another one, it will break presence because the rules are not consistent. Once cognitive presence is broken, it's very difficult to "fix" it. The player is constantly reminded that the simulation is not real, and it will take some time to accept it again as reality.

11. List some applications of VR. What might be the possible navigation techniques and manipulating interfaces in virtual reality? Justify. (2+3)

Ans: Application area of virtual reality

Here is a list of the many applications of virtual reality:

- In the Military
- In Education
- in Healthcare
- in Entertainment
- Fashion
- Heritage
- in Business
- in Engineering
- in Sport
- in Media
- Scientific Visualizations
- in Telecommunications
- in Construction
- in Film
- Programming languages

The layout of the action-perception space is dependent on whether the user is left-handed or right-handed. The layout described below is meant for a right handed person. The layout consists of a linearly structured image database browser, virtual paper (7), a floating toolbar (6), EPP (9) and a 3to2D window (8) in the action perception space. The communication space is currently used to provide a surface rendering of the 3D model. This surface model is used to visualize the position of the RISP with respect to the 3D data.

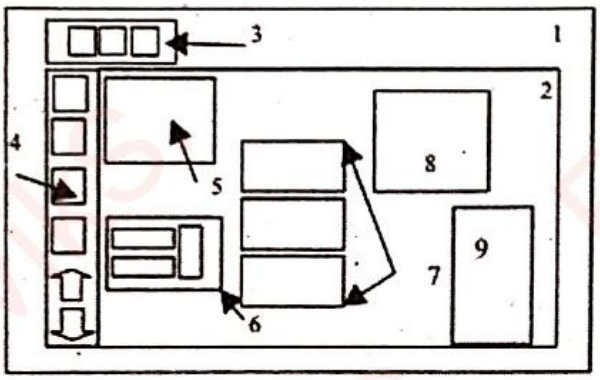
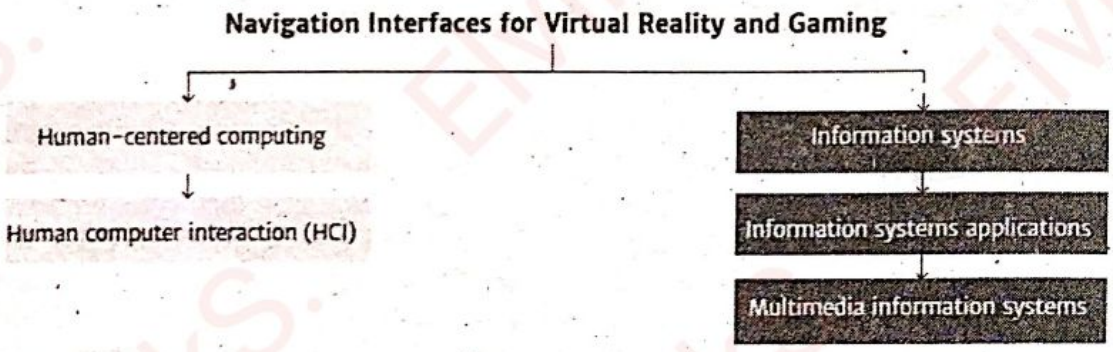


Figure: The action-perception space layout. 1-Entire action-perception space, 2-Sketchable area (UltraPad), 3, 4, 5-Image database browser, 6- Floating toolbar, 7-Virtual Papers, 8 – 3 to 2D window, 9 – Enhanced Paper Prop.



12. Mention any two color command in OpenGL. Explain about Hermite curve. (2+3)

Ans: Color command in OpenGL

OpenGL has two color modes: the RGBA mode and color index mode. In RGBA mode, a color is specified by three intensities (for the Red, Green, and Blue components of the color) and optionally a fourth value, Alpha, which controls transparency.

The function glColor4f (red, green, blue, alpha)

Maps available red, green, and blue intensities onto (0.0, 1.0) where 0.0 means that the color component is absent ((0.0, 0.0, 0.0) is black) and 1.0 is a saturated color ((1.0, 1.0, 1.0) is white.) The number of bits used to represent each color depends upon the graphics card. Current graphics cards have several Mbytes of memory, and use 24 or 32 bits for color (24-bit: 8 bits per color; 32-bit: 8 bits per color + 8 bits padding or transparency). The term bit plane refers to an image of single-bit values. Thus, a system with 24 bits of color has 24 bit planes.

For a system with 8 bit planes, 28 = 256 different colors could be displayed simultaneously. These 256 colors could be selected from a set of 2m colors. For m = 24 this gives 22 4 ≈ 16 million colors. The color table could be altered to give different palettes of colors. OpenGL supports this mode, called color index mode.

Using glColor3f

glColor3f() takes 3 arguments: the red, green and blue components of the color. After we use glColor3f, everything draw will be in that color. For example, consider this display function:

```
void display()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glColor3f(0.5f, 0.0f, 1.0f); // (0.5, 0, 1) is half red and full blue, giving purple.
glBegin(GL_QUADS);
glVertex2f(-0.75, 0.75);
glVertex2f(-0.75, -0.75);
glVertex2f(0.75, -0.75);
glVertex2f(0.75, 0.75);
glEnd();
glutSwapBuffers();
}
```

Hermite curve

Hermite curves are very easy to calculate but also very powerful. They are used to smoothly interpolate between key-points (like object movement in key frame animation or camera control). Understanding the mathematical background of hermite curves will help you to understand the entire family of splines. Maybe you have some experience with 3D programming and have already used them without knowing that (the so called kb-splines, curves with control over tension, continuity and bias are just a special form of the hermite curves).

To keep it simple we first with 2-dimensional curves here. Hermite curves work in any number of dimensions. To calculate a hermite curve you need the following vectors:

- P1: the start point of the curve
- T1: the tangent (e.g. direction and speed) to how the curve leaves the start point
- P2: the endpoint of the curve
- T2: the tangent (e.g. direction and speed) to how the curves meets the endpoint

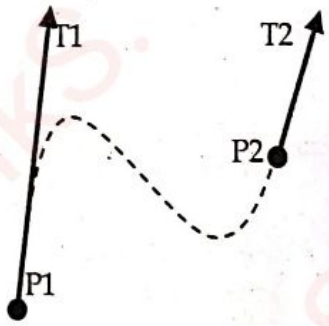


Figure: Hermite Curve

Matrix form of Hermite Curve

Vector S: The interpolation-point and its powers up to 3:

Vector C: The parameters of our hermite curve:

Matrix h: The matrix form of the 4 hermite polynomials:

$$S = \begin{bmatrix} s^3 \\ s^2 \\ s^1 \\ 1 \end{bmatrix} \qquad C = \begin{bmatrix} P1 \\ P2 \\ T1 \\ T2 \end{bmatrix} \qquad h = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

To calculate a point on the curve you build the Vector S, multiply it with the matrix h and then multiply with C.

$P = S \times h \times C$.

❑❑❑

TRIBHUVAN UNIVERSITY

Institution of Science and Technology

Course Title: Computer Graphics
Course No: CSC 209
Nature of the Course: Theory + Lab
Semester: III

Full Marks: 60
Pass Marks: 24
Time: 3 hrs.

Computer Graphics

TU QUESTIONS-ANSWERS 2077

Section A

Attempt any two questions. **(2 ×10 =20)**

1. List the major differences between DAA and Bradenham's Line drawing algorithm. Illustrate the DAA algorithm to the line width end points (2, 2) and (9, 2).

Ans: While drawing a line on computers they need to perform a set of computation, and it is not that simple as humans can do it in a simple way. So, in computer graphics, there are two algorithms used for drawing a line over the screen that is DDA (Digital Differential Analyzer) algorithm and Bresenham algorithm. The chief difference between them is that the DDA algorithm uses floating point values while Bresenham employs integer with round off functions.

The major differentiate between them are tabulated below:

DDA Algorithm	Bresenham's Line Algorithm
1. DDA Algorithm uses floating point, real arithmetic.	1. Bresenham's Algorithm uses fixed points, integer arithmetic.
2. DDA algorithm uses multiplication and division operations.	2. Bresenham's Algorithm uses addition and subtraction operations.
3. DDA algorithm is slower than Bresenham's Algorithm because it uses real arithmetic floating point operations.	3. Bresenham's Algorithm is faster than DDA algorithm because it uses integer arithmetic.
4. DDA algorithm can draw circles and curves with less accuracy.	4. Bresenham's Algorithm can draw circles and curves with much more accuracy.
5. DDA algorithm is less efficient than Bresenham's Algorithm.	5. Bresenham's Algorithm is more efficient than DDA Algorithm.
6. DDA algorithm round-off the co-ordinates to integer that is nearest to the line.	6. Bresenham's Algorithm doesn't round-off the co-ordinates.

DAA algorithm to the line width end points (2, 3) and (6, 15)

$P_1(2, 3)$ $P_{11}(6, 15)$

$x_1 = 2$

$y_1 = 3$

$x_2 = 6$

$y_2 = 15$

$dx = 6 - 2 = 4$

$dy = 15 - 3 = 12$

$\frac{dy}{dx} = \frac{12}{4}$

For calculating next value of x takes $x = x + \frac{1}{m}$

$P_1(2, 3)$ point plotted

$P_2\left(2\frac{1}{3}, 4\right)$ point plotted

$P_3\left(2\frac{2}{3}, 5\right)$ point not plotted

$P_4(3, 6)$ point plotted

$P_5\left(3\frac{1}{3}, 7\right)$ point not plotted

$P_6\left(3\frac{2}{3}, 8\right)$ point not plotted

$P_7(4, 9)$ point plotted

$P_8\left(4\frac{1}{3}, 10\right)$ point not plotted

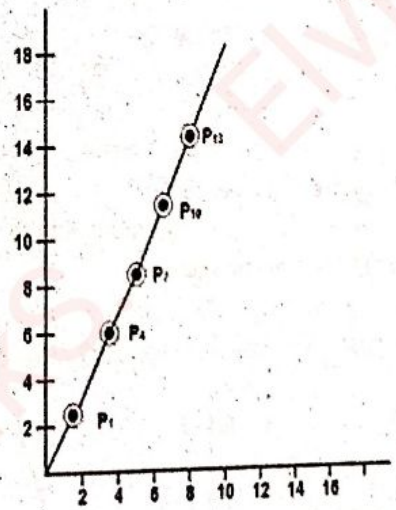
$P_9\left(4\frac{2}{3}, 11\right)$ point not plotted

$P_{10}(5, 12)$ point plotted

$P_{11}\left(5\frac{1}{3}, 13\right)$ point not plotted

$12_9\left(5\frac{2}{3}, 14\right)$ point not plotted

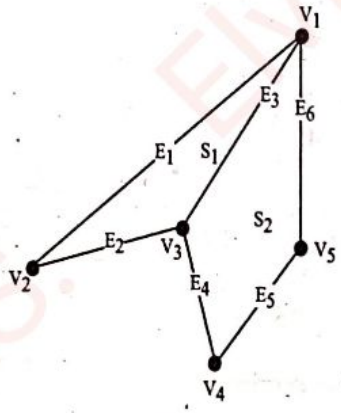
$P_{13}(6, 15)$ point plotted



2. How polygon table is used in representing polygons? Explain the representations of any three curves.

Ans: We specify objects as a set of vertices and associated attributes. This information can be stored in tables, of which there are two types: geometric tables and attribute tables. The geometry can be stored as three tables: a vertex table, an edge table, and a polygon table. Each entry in the vertex table is a list of coordinates defining that point. Each entry in the edge table

consists of a pointer to each endpoint of that edge. And the entries in the polygon table define a polygon by providing pointers to the edges that make up the polygon.



VERTEX TABLE
$V_1 : x_1, y_2, z_1$
$V_2 : x_2, y_2, z_2$
$V_3 : x_3, y_3, z_3$
$V_4 : x_4, y_4, z_4$
$V_5 : x_5, y_5, z_5$

EDGE TABLE
$E_1 : V_1, V_2$
$E_2 : V_2, V_3$
$E_3 : V_3, V_1$
$E_4 : V_3, V_4$
$E_5 : V_4, V_5$
$E_6 : V_5, V_1$

POLYGON-SURFACE TABLE
$S_1 : E_1, E_2, E_3$
$S_2 : E_3, E_4, E_5, E_6$

We can eliminate the edge table by letting the polygon table reference the vertices directly, but we can run into problems, such as drawing some edges twice, because we don't realize that we have visited the same set of points before, in a different polygon. We could go even further and eliminate the vertex table by listing all the coordinates explicitly in the polygon table, but this wastes space because the same points appear in the polygon table several times.

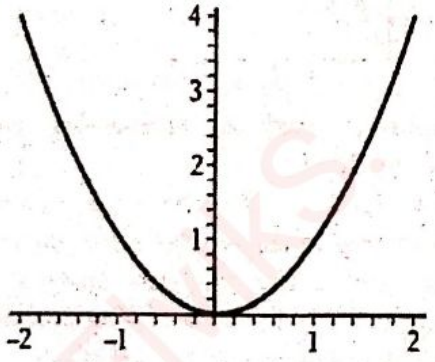
Three types of representations for curves and surfaces are common in computer graphics and geometric design: explicit, implicit, and parametric.

Explicit Representations

When you first studied analytic geometry, you used rectangular coordinates and considered equations of the form $y = f(x)$. The graphs $(x, f(x))$ of these functions are curves in the plane. For example, $y = 3x + 1$ represents a straight line, and $y = x^2$ represents a parabola (see Figure).

Similarly, you could generate surfaces by considering equations of the form $z = f(x, y)$: the equation $z = 2x + 5y - 7$ represents a plane in 3-space, and $z = x^2 - y^2$ represents a hyperbolic paraboloid.

Expressions of the form $y = f(x)$ or $z = f(x, y)$ are called explicit representations because they express one variable explicitly in terms of the other variables.



Implicit Representations

Not all curves and surfaces can be captured readily by a single explicit expression. For example, the unit circle centered at the origin is represented implicitly by all solutions to the equation $x^2 + y^2 - 1 = 0$. If we try to solve explicitly for y in terms of x, we obtain

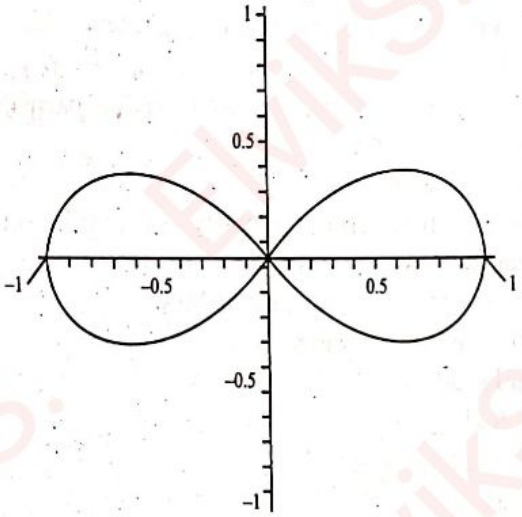
$$y = \sqrt{1 - x^2}$$

Which represents only the upper half circle. We must use two explicit formulas

$$y = \pm\sqrt{1 - x^2}$$

Implicit Representations to capture the entire circle.

Often it is easier just to stick with the original implicit equation rather than to solve explicitly for one of the variables. Thus $x^2 + y^2 - 1 = 0$ represents a circle, and $x^2 + y^2 + z^2 - 1 = 0$ represents a sphere. Equations of the form $f(x, y) = 0$ or $f(x, y, z) = 0$ are called implicit representations because they represent the curve or surface implicitly without explicitly solving for one of the variables.



Parametric Representations

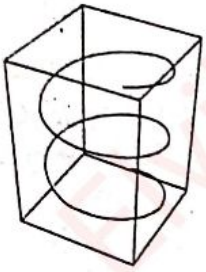
There is another standard way to represent curves and surfaces that is more general than the explicit form and yet is still easy to render. We can express curves and surfaces parametrically by representing each coordinate with an explicit equation in a new set of parameters. For planar curves we set $x = x(t)$ and $y = y(t)$; for surfaces in 3-space we set $x = x(s,t)$, $y = y(s,t)$, and $z = z(s,t)$. For example, the parametric equations

$$x(t) = \frac{2t}{1 + t^2} \quad y(t) = \frac{1 - t^2}{1 + t^2}$$

Represent the unit circle centered at the origin because by simple substitution we can readily verify that $x^2(t) + y^2(t) - 1 = 0$. Similarly, the parametric equations

$$x(s, t) = \frac{2s}{1 + s^2 + t^2} \quad y(s, t) = \frac{2t}{1 + s^2 + t^2} \quad z(s, t) = \frac{1 - s^2 - t^2}{1 + s^2 + t^2}$$

Represent a unit sphere, since $x^2(s, t) + y^2(s, t) + z^2(s, t) - 1 = 0$. Often we shall restrict the parameter domain. Thus a parametric curve is typically the image of a line segment; a parametric surface, the image of a region--usually rectangular or triangular - of the plane.



3. **Define realism in human perceptions. What is the significance difference between rendering and image synthesis in creating computer generated 3D image? Describe any two polygon rendering methods.**

Ans: Visual realism is defined as the extent to which an image appears to people as a photo rather than computer generated. Assessing visual realism is important in applications like computer graphics rendering and photo retouching. However, current realism evaluation approaches use either labor-intensive human judgments or automated algorithms largely dependent on comparing renderings to reference images.

The main difference between Raster and Vector is that Raster images are made up of colored pixels that are placed closely together in such a way that they appear to be a picture and not just dots whereas Vector images are made of lines, curves, and fills that are placed at particular coordinates for them to appear as pictures.

Polygon rendering methods

Rendering means giving proper intensity at each point in a graphical object to make it look like real world object.

Different rendering methods are listed below:

- Constant intensity shading
- Gouraud shading
- Phong shading
- Fast Phong shading etc.

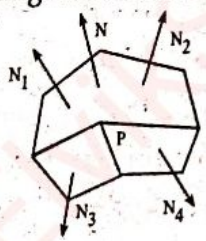
Gouraud shading

This Intensity-Interpolation scheme, developed by Gouraud and usually referred to as Gouraud Shading, renders a polygon surface by linear interpolating intensity value across the surface. Intensity values for each polygon are coordinate with the value of adjacent polygons along the common edges, thus eliminating the intensity discontinuities that can occur in flat shading.

Each polygon surface is rendered with Gouraud Shading by performing the following calculations:

- Determining the average unit normal vector at each polygon vertex.
- Apply an illumination model to each vertex to determine the vertex intensity.
- Linear interpolate the vertex intensities over the surface of the polygon.

At each polygon vertex, we obtain a normal vector by averaging the surface normal of all polygons staring that vertex as shown in fig:



Thus, for any vertex position V, we acquire the unit vertex normal with the calculation

$$N_v = \frac{\sum_{k=1}^{n} = 1 N_k}{|\sum_{k=1}^{n} N_k|}$$

Once we have the vertex normals, we can determine the intensity at the vertices from a lighting model.

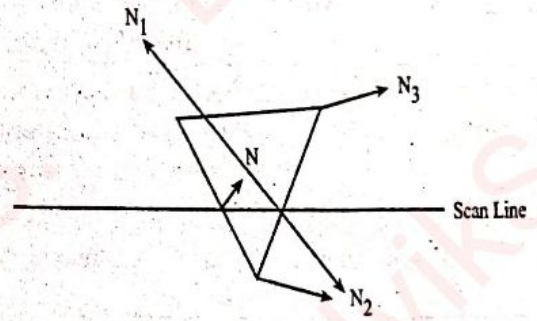
Phong Shading

A more accurate method for rendering a polygon surface is to interpolate the normal vector and then apply the illumination model to each surface point. This method developed by Phong Bui Tuong is called Phong Shading or normal vector Interpolation Shading. It displays more realistic highlights on a surface and greatly reduces the Match-band effect.

A polygon surface is rendered using Phong shading by carrying out the following steps:

- Determine the average unit normal vector at each polygon vertex.
- Linearly & interpolate the vertex normals over the surface of the polygon.
- Apply an illumination model along each scan line to calculate projected pixel intensities for the surface points.

Interpolation of the surface normal along a polynomial edge between two vertices as shown in fig:



$$N = \frac{Y - Y_2}{Y_1 - Y_2} N_1 + \frac{Y_1 - Y}{Y_1 - Y_2} N_2$$

Incremental methods are used to evaluate normals between scan lines and along each scan line. At each pixel position along a scan line, the illumination model is applied to determine the surface intensity at that point.

Intensity calculations using an approximated normal vector at each point along the scan line produce more accurate results than the direct interpolation of intensities, as in Gouraud Shading. The trade-off, however, is that phong shading requires considerably more calculations.

Section B

Short Answer Questions

Attempt any *EIGHT* questions **[8 × 5 = 40]**

4. Differentiate between vector and raster graphics.

Ans: When an image is resized or zoomed into, at times it turns out to be perfectly clear. However, sometimes the image shows aliasing. This is because of the format of the image, which can either be Raster or Vector.

The major difference between Raster and Vector Graphics are tabulated below,

Raster	Vector
1. Raster images are those which are made up of pixels.	1. Vector images are those which are made up of lines, curves, and fills.
2. They use GIF, JPEG, TIFF, XBM, PNG and PCX graphic formats.	2. They use EPS, WMF, PICT, TrueType, and PostScript graphic formats.
3. They are not as scalable as the latter.	3. They are easily scalable.
4. Painting them is like dipping a brush in paint and using it on a real canvas.	4. On painting, only the outlines of the images get colored.
5. They work best for photo editing.	5. They work best for drawings, illustrations, logos, and other technical imagery.
6. They can be used in Photoshop, paint shop and GIMP.	6. They can be used in CorelDraw, Illustrator and Inkscape.
7. Converting a Raster to a Vector image involves complexity and is time-consuming.	7. Vector images can be converted into Raster images easily.
8. It is difficult to print these images when the spot colors are limited.	8. It is easy to print these images as the number of colors can be changed at the time of printing.
9. A Raster file can be easily converted to other file formats.	9. Vector files cannot be modified or displayed in programs that do not understand the format.
10. Extensions such as .jpg, .gif, .png, and .tiff are used for these images.	10. Extensions such as .eps, .cdr, .pdf, .ai and .svg are used for these images.

5. Translate a triangle ABC with co-ordinates A(0, 0), B(5, 0) and C(5, 5) by 2 units in x-direction and 3 units in y-directions.

Ans: Given the directions and measures of translation to be done in XY-coordinate system the new coordinates can be easily drawn from them.

Let the coordinate of a point P in the XY-coordinate system be (x, y)

We have to translate the point by 'a' in the right direction (i.e., x-direction) and 'b' units upwards (i.e., y-direction). Then the translated point is given by $(x + a, y + b)$.

The coordinates of the triangle are A(0, 0), B(5, 0) and C(5, 5)

Here we have a=2 and b=3.

The points A', B' and C' of the translated triangle are,

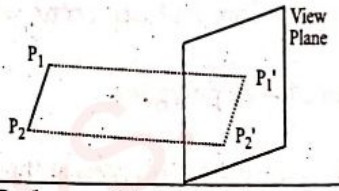
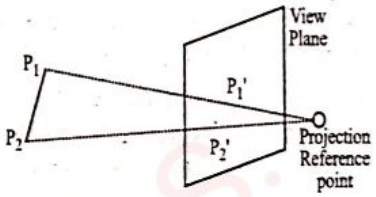
$A'(0+2, 0+3) = A'(2, 3)$

$B'(5+2, 0+3) = A'(7, 3)$

$C'(5+2, 5+3) = A'(7, 8)$

6. Differentiate between orthographies, parallel and perspective projections.

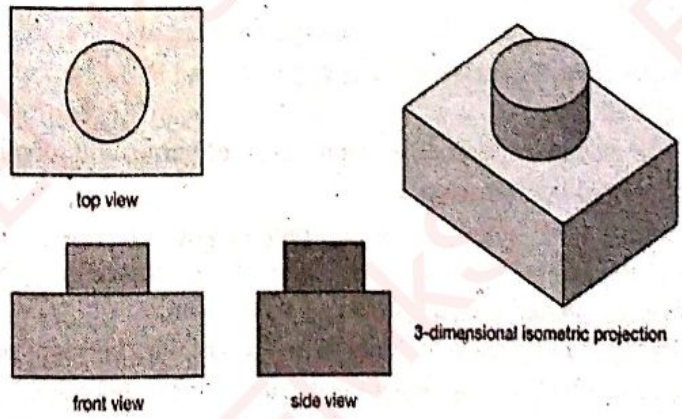
Ans: Difference between Parallel Projection and Perspective Projection:

Parallel Projection	Perspective Projection
1. Parallel projection represents the object in a different way like telescope.	1. Perspective projection represents the object in three dimensional way.
2. In parallel projection, these effects are not created.	2. In perspective projection, objects that are far away appear smaller, and objects that are near appear bigger.
3. The distance of the object from the center of projection is infinite.	3. The distance of the object from the center of projection is finite.
4. Parallel projection can give the accurate view of object.	4. Perspective projection cannot give the accurate view of object.
5. The lines of parallel projection are parallel.	5. The lines of perspective projection are not parallel.
6. Projector in parallel projection is parallel.	6. Projector in perspective projection is not parallel.
7. Two types of parallel projection: i. Orthographic ii. Oblique	7. Three types of perspective projection: i. one point perspective, ii. Two point perspective, iii. Three point perspective,
8. It does not form realistic view of object. 	8. It forms a realistic view of object. 

Orthographic Projection

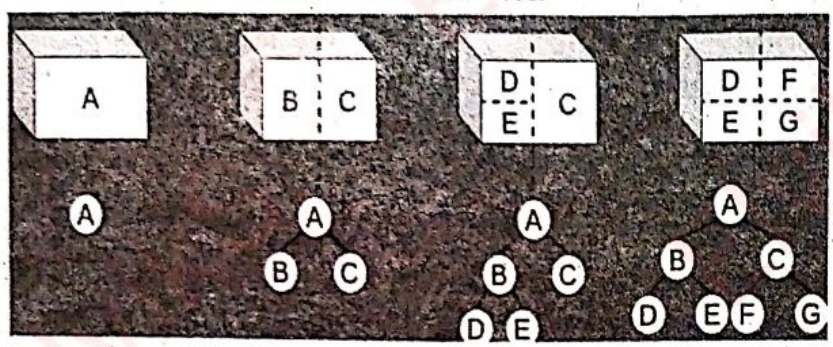
Orthographic Projections is a technical drawing in which different views of an object are projected on different reference planes observing perpendicular to respective reference plane.

Orthographic projection is a common method of representing three-dimensional objects, usually by three two-dimensional drawings in each of which the object is viewed along parallel lines that are perpendicular to the plane of the drawing. For example, an orthographic projection of a house typically consists of a top view, or plan, and a front view and one side view (front and side elevations).



7. **Describe how a polygon can be represented using BSP tree with example.**

Ans: Binary Space Partitioning is implemented for recursively subdividing a space into two convex sets by using hyper planes as partitions. This process of subdividing gives rise to the representation of objects within the space in the form of tree data structure known as BSP Tree.



Binary space partitioning arose from the computer graphics need to rapidly draw three-dimensional scenes composed of polygons. A simple way to draw such scenes is the painter's algorithm, which produces polygons in order of distance from the viewer, back to front, painting over the background, and previous polygons with each closer object. This approach has two disadvantages: the time required to sort polygons in back to front order, and the possibility of errors in overlapping polygons. Fuchs and co-authors showed that constructing a BSP tree solved both of these problems by providing a rapid method of sorting polygons with respect to a given viewpoint (linear in the number of polygons in the scene) and by subdividing overlapping polygons to avoid errors that can occur with the painter's algorithm.

Algorithm of generating a BSP Tree from a list of polygons

1. Select a polygon P from the list.
2. Make a node N in the BSP tree, and add P to the list of polygons at that node.
3. For each other polygon in the list:
   - If that polygon is wholly in front of the plane containing P, move that polygon to the list of nodes in front of P.
   - If that polygon is wholly behind the plane containing P, move that polygon to the list of nodes behind P.
   - If that polygon is intersected by the plane containing P, split it into two polygons and move them to the respective lists of polygons behind and in front of P.
   - If that polygon lies in the plane containing P, add it to the list of polygons at node N.
4. Apply this algorithm to the list of polygons in front of P.
5. Apply this algorithm to the list of polygons behind P.

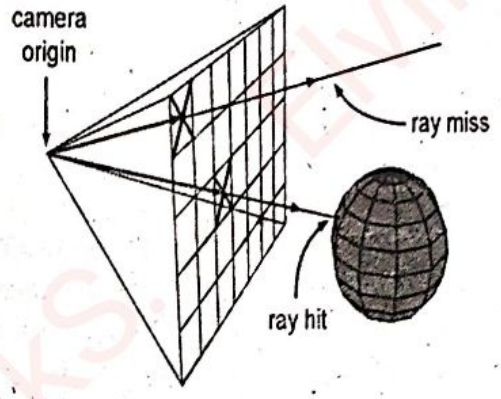
Uses of BSP

- It is used in collision detection in 3D video games and robotics.
- It is used in ray tracing
- It is involved in the handling of complex spatial scenes.

8. **What is the role of ray tracing in visible surface detection? Explain how scan line algorithm is used for back face detection.**

Ans: Ray-tracing is a technique for computing the visibility between points. Light transport algorithms are designed to simulate the way light propagates

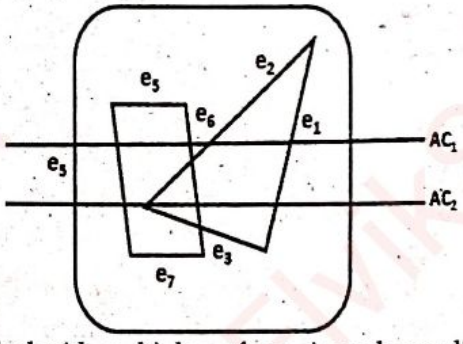
through space (while interacting with objects). They are used to compute the color of a point in the scene.



Scan Line Algorithm

It is an image space algorithm. It processes one line at a time rather than one pixel at a time. It uses the concept area of coherence. This algorithm records edge list, active edge list. So accurate bookkeeping is necessary. The edge list or edge table contains the coordinate of two endpoints. Active Edge List (AEL) contain edges a given scan line intersects during its sweep. The active edge list (AEL) should be sorted in increasing order of x. The AEL is dynamic, growing and shrinking.

Following figures shown edges and active edge list. The active edge list for scan line AC1contain e_1, e_2, e_5, e_6 edges. The active edge list for scan line AC2contain e_5, e_6, e_1.



Scan line can deal with multiple surfaces. As each scan line is processed, this line will intersect many surfaces. The intersecting line will determine which surface is visible. Depth calculation for each surface is done. The surface rear to view plane is defined. When the visibility of a surface is determined, then intensity value is entered into refresh buffer.

Algorithm

Step 1: Start

Step 2: Initialize the desired data structure

1. Create a polygon table having color, edge pointers, coefficients
2. Establish edge table contains information regarding, the endpoint of edges, pointer to polygon, inverse slope.
3. Create Active edge list. This will be sorted in increasing order of x.
4. Create a flag F. It will have two values either on or off.

Step 3: Perform the following steps for all scan lines

1. Enter values in Active edge list (AEL) in sorted order using y as value
2. Scan until the flag, i.e. F is on using a background color
3. When one polygon flag is on, and this is for surface S_1enter color intensity as I_1into refresh buffer

4. When two or image surface flag are on, sort the surfaces according to depth and use intensity value S_n for the nth surface. This surface will have least z depth value
5. Use the concept of coherence for remaining planes.

Step 4: Stop

9. How virtual realities differ with our real word? Describe some components of VR system.

Ans: The most commonly argued distinction between the real and virtual worlds is the concept of physical form and identity. It is argued by many that in the virtual world you do not have a complete physical identity and that your mind being removed from the physical body is the distinction between the real and virtual. In other words, in the virtual world you have no real identity or physical interaction, whereas in the real world you do.

Although it seems clear that the physical characteristics that identify our physical are not obviously visible in the virtual world, the physical self is not completely left behind and we do in fact experience physical manifestations within the virtual world.

The components necessary for building and experiencing VR can be divided into nine categories:

Hardware:

- Sensory Displays
- Computer
- Process Acceleration Cards
- Tracking System
- Input Devices

Software:

- 3D Modeling
- 2D Graphics
- Digital Sound Editing
- VR Simulation

A VR system can be based on a personal computer. The computer controls several different sensory display devices to immerse you in a 3-dimensional virtual environment. The most common sensory displays are head-mounted displays for 3D visual and headphones for 3D audio. Since these displays need to be updated with new sensory information more than 20 times per second it often helps to have additional processing power in the form of add-on 3D graphics cards and 3D sound cards.

A VR system needs to be able to track the position and orientation of your head in order to calculate the appropriate perspectives to display. Any other body parts, such as your hands, feet, or prehensile tails that will play an active part in the virtual environment must also be tracked. The device that does this is called (surprisingly enough) a tracking device.

Input devices make up the final category of VR hardware. In order to interact with the virtual environment you may wish to use a joystick (sometimes called a wand in VR systems), an instrumented glove, a keyboard, voice recognition, or other types of input. These devices allow you to travel through the virtual environment, manipulate objects, and perhaps even build on to the virtual world. Tracking devices are sometimes used together with input devices to add a spatial (3 dimensional) component to their operation.

In order to build virtual environments you often need auxiliary software for creating the objects that go into the virtual environment and setting their characteristics. Three-dimensional modeling software allows you to construct the geometry of the objects and specify some of their visual properties. Two-dimensional graphics software lets you manipulate textures to be applied to the objects which can often greatly enhance their visual detail. Digital sound editing software lets you mix and edit the sounds that objects make. All these software packages have other commercial uses in addition to building VR, and so there is a great variety to choose from.

The simulation software is what brings all the components together. It accepts data from the trackers and input devices, applies this information to the objects you have built, and updates the sensory displays.

10. Write a procedure to draw a line in OpenGL? Describe Painter's algorithm.

Ans: GL_LINES is used to specify two vertices and draw a line between them in OpenGL. Two vertices specify a single primitive. If you specify an odd number of vertices for GL_LINES, the last vertex is just ignored.

Example:

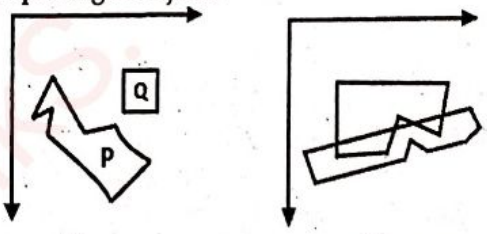
```
void renderScene(void)
{
 glClear( GL_COLOR_BUFFER_BIT);
 glBegin( GL_LINES);
 glVertex3f(0.0f, 0.0f, 0.0f);
 glVertex3f(50.0f, 50.0f, 0.0f);
 glEnd();
 glutSwapBuffers();
}
```

Painter Algorithm

It came under the category of list priority algorithm. It is also called a depth-sort algorithm. In this algorithm ordering of visibility of an object is done. If objects are reversed in a particular order, then correct picture results.

Objects are arranged in increasing order to z coordinate. Rendering is done in order of z coordinate. Further objects will obscure near one. Pixels of rear one will overwrite pixels of farther objects. If z values of two overlap, we can determine the correct order from Z value as shown in fig (a).

If z objects overlap each other as in fig (b) this correct order can be maintained by splitting of objects.



(a) (b)

Painter Algorithm

Step 1: Start

Step 2: Sort all polygons by z value keep the largest value of z first.

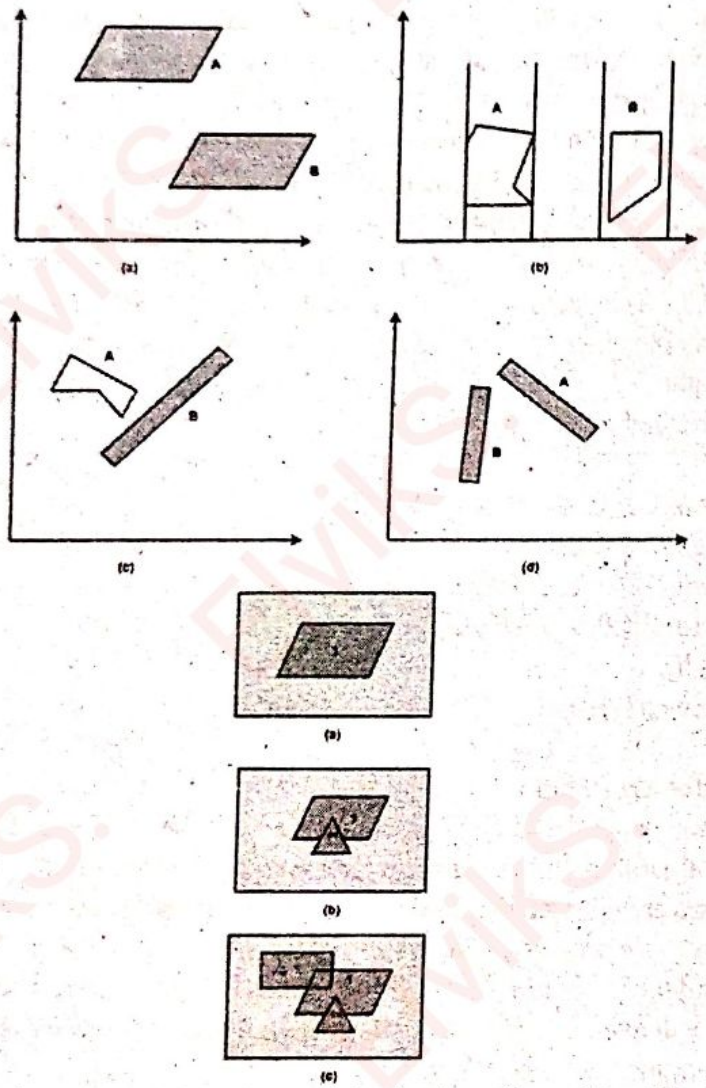
Step 3: Scan converts polygons in this order.

Test is applied

- Does A is behind and non-overlapping B in the dimension of Z as shown in fig (a)

- Does A is behind B in z and no overlapping in x or y as shown in fig (b)
- If A is behind B in Z and totally outside B with respect to view plane as shown in fig (c)
- If A is behind B in Z and B is totally inside A with respect to view plane as shown in fig (d)

The success of any test with single overlapping polygon allows F to be painted.



11. Let a rectangular window size with (5, 9). The points of the line are (4, 12) and (8, 8). Use the Liang- Barsky algorithm to clip the line and find the intersection point.

Ans: We have,

The initial point of the line (p_1) = (4, 12)

The ending point of the line (p_2) = (8, 8)

$x_1 = 4, x_2 = 8$

$y_1 = 12, y_2 = 8$

$xw_{min} = 5, xw_{max} = 9$

$yw_{min} = 5, yw_{max} = 9$

Step 1: We have to calculate the value of ?x and?y-

$?x = x_2 - x_1 = 8-4 = 4$

$?y = y_2 - y_1 = 8-12 = -4$

Step 2: Now, we will calculate-

$p_1 = -4$ $q_1 = 4-5 = -1$

$p_2 = 4$ $q_2 = 9-4 = 5$

$p_3 = 4$ $q_3 = 12-5 = 7$

$p_4 = -4$ $q_4 = 9-12 = -3$

Step 3: Now we will calculate t_1 value-

If $p_1, p_4 < 0$

Then $t_1 = \max(0, q_k / p_k)$

$= \max(0, q_1/p_1, q_4/p_4)$

$= \max(0, 1/4, 3/4)$

$t_1 = 3/4$

If $p_2, p_3 > 0$

Then $t_2 = \min(1, q_k / p_k)$

$= \min(1, q_2/p_2, q_3/p_3)$

$= \min(1, 5/4, 7/4)$

$t_2 = 1$

Step 4: Now, we have to calculate the intersection point.

$x = x_1 + t_1.?x = 4 + 3/4 * 4 = 7$

$y = y_1 + t_1.?y = 12 + 3/4 * (-4) = 9$

The coordinates intersection point = (7, 9)

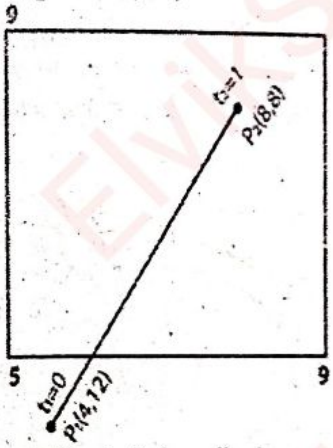


Figure: Before clipping

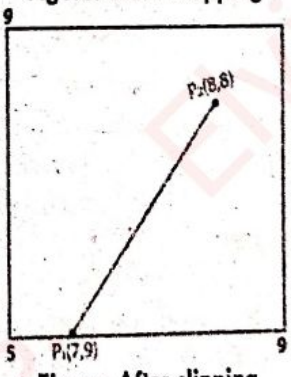


Figure: After clipping

12. Define blobby objects. Describe about basic illumination models.

Ans: Some objects do not maintain a fixed shape, but change their surface characteristics in certain motions or when in proximity to other objects. Examples in this class of objects include molecular structures, water droplets and other liquid effects, melting objects, and muscle shapes in the human body. These objects can be described as exhibiting "blobbiness" and are often simply referred to as blobby objects, since their shapes show a certain degree of fluidity.

Examples in this class of objects include

1. Water droplets
2. Melting objects

3. Muscle shapes in the human body.

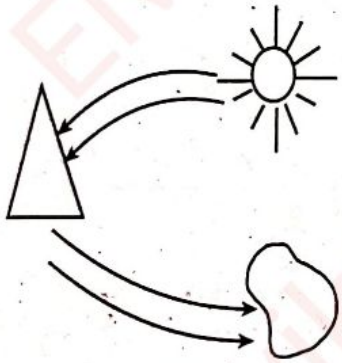
Illumination model or a lighting model is the model for calculating light intensity at a single surface point. Sometimes also referred to as a shading model. An illumination model is also called lighting model and sometimes called as a shading model which is used to calculate the intensity of light that we should see at a given point on the surface of an object. A surface-rendering algorithm uses the intensity calculations from an illumination model. Some illumination models are:

1. Ambient Light
2. Diffuse Reflection
3. Specular Reflection and phong model

Ambient light:

This is a simplest illumination model. We can think of this model, which has no external light source-self-luminous objects. A surface that is not exposed directly to light source still will be visible if nearby objects are illuminated. The combination of light reflections form various surfaces to produce a uniform illumination is called ambient light or background light.

Ambient light means the light that is already present in a scene, before any additional lighting is added. It usually refers to natural light, either outdoors or coming through windows etc. It can also mean artificial lights such as normal room lights.



Ambient light has no spatial or directional characteristics and amount on each object is a constant for all surfaces and all directions. In this model, illumination can be expressed by an illumination equation in variables associated with the point on the object being shaded. The equation expressing this simple model is;

$I=K_a$

Where I is the resulting intensity and K_a is the object's intrinsic intensity.

If we assume that ambient light impinges equally on all surface from all direction, then

$I = I_a K_a$

Where I_a is intensity of ambient light.

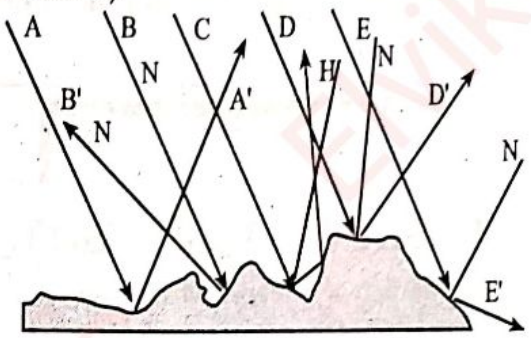
K_a is object's intrinsic intensity.

The amount of light reflected from an object's surface is determined by K_a, the ambient-reflection coefficient. K_a ranges from 0 to 1.

Diffuse Reflection

Objects illuminated by ambient light are uniformly illuminated across their surfaces even though light are more or less bright in direct proportion of ambient intensity. Surfaces are rough. Incident light is scattered with equal intensity in all directions. Surfaces appear equally bright from all direction.

Such surfaces are called ideal diffuse reflectors (also referred to as Lambertian reflectors).



Diffuse Reflection from a Surface

The diffuse-reflection coefficient, or diffuse reflectivity, kd (varying from 0 to 1) define the fractional amount of the incident light that is diffusely reflected. The parameter kd (actually function of surface color) depends on the reflecting properties of material so for highly reflective surfaces, the kd nearly equal to 1. If a surface is exposed only to ambient light, we can express the intensity of the diffuse reflection at any point on the surface as:

I_{amb}, Diff = k_d . I_a , (where I_a = intensity of ambient light)

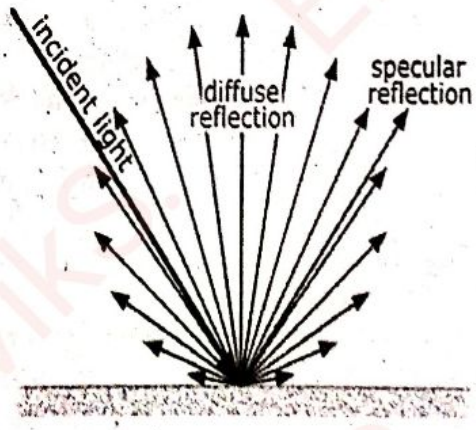
If N is the unit normal vector to a surface and L is the unit direction vector to the point light source from a position on the surface then cos θ = N.L (Lambertian Cosine Law) and the diffuse reflection equation for single point source illumination is:

I_L, diff = k_d. I_L cos θ = k_d I_L (N.L)

Specular reflection and phong model

When we look at an illuminated shiny surface, such as polished metal, a person's forehead, we see a highlight or bright spot, at certain viewing direction. Such phenomenon is called specular reflection.

It is the result of total or near total reflection of the incident light in a concentrated region around the " specular reflection angle = angle of incidence". Perfect reflector (mirror) reflects all lights to the direction where angle of reflection is identical to the angle of incidence. It accounts for the highlight.



❑❑❑