

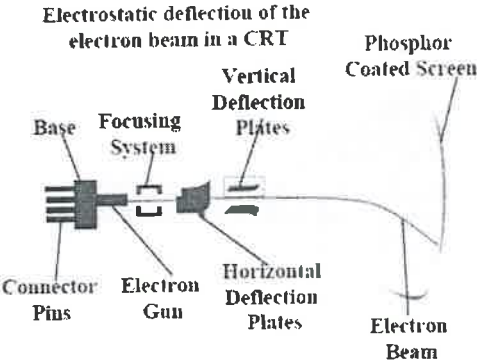
①

(Time: 2½ hours)

Q. P. Code:

65507
Total Marks: 75

- N. B.: (1) All questions are compulsory.
(2) Make suitable assumptions wherever necessary and state the assumptions made.
(3) Answers to the same question must be written together.
(4) Numbers to the right indicate marks.
(5) Draw neat labeled diagrams wherever necessary.
(6) Use of Non-programmable calculators is allowed.

1. Attempt <u>any three</u> of the following:	15
a. What is computer graphics? Explain computer graphics applications and software.	
<p>Solution: Computer graphics is a field of computer science that is concerned with digitally synthesizing and manipulating visual contents. It is also some time known as science and technology of creating, storing, displaying and manipulating images and objects.</p> <p>The terms computer graphics deals with</p> <ul style="list-style-type: none"> • Representation and manipulation of pictorial data by computer. • Various technologies used to create and manipulate such pictorial data. • Images so produced. • Study of methods for digitally synthesizing and manipulating visual contents. <p>Applications: (Write any five points and explain it).</p> <ul style="list-style-type: none"> ➤ Graphical user interface (GUIs) ➤ Entertainment ➤ Computer-Aided design (CAD) ➤ Computer Arts ➤ Education and Training ➤ Medical imaging ➤ Cartography ➤ Stimulations and modelling <p>Software's: (Write any five points with its uses).</p> <ul style="list-style-type: none"> ➤ Photoshop ➤ CorelDRAW ➤ Maya ➤ Flash ➤ 3D studio ➤ Paint brush ➤ Animator pro ➤ Picasa ➤ Canvas 	
b. Explain the operation of CRT, with a neat labelled diagram in brief.	
<p>Solution:</p> <p>Electrostatic deflection of the electron beam in a CRT</p> 	

[TURN OVER]

2

	<p>Working of CRT</p> <p>The working of CRT depends on the movement of electrons beams. The electron guns generate sharply focused electrons which are accelerated at high voltage. This high-velocity electron beam when strikes on the fluorescent screen creates luminous spot</p> <p>After exiting from the electron gun, the beam passes through the pairs of electrostatic deflection plate. These plates deflected the beams when the voltage applied across it. The one pair of plate moves the beam upward and the second pair of plate moves the beam from one side to another. The horizontal and vertical movement of the electron are independent of each other, and hence the electron beam positioned anywhere on the screen.</p> <p>The working parts of a CRT are enclosed in a vacuum glass envelope so that the emitted electron can easily move freely from one end of the tube to the other.</p>																											
c.	<p>Distinguish between Raster scan display device and random scan display device.</p> <p>Solution:</p> <table><tr><th></th><th>Raster Scan System</th><th>Random Scan System</th></tr><tr><td>Resolution</td><td>It has poor or less Resolution because picture definition is stored as a intensity value.</td><td>It has High Resolution because it stores picture definition as a set of line commands.</td></tr><tr><td>Electron-Beam</td><td>Electron Beam is directed from top to bottom and one row at a time on screen, but electron beam is directed to whole screen.</td><td>Electron Beam is directed to only that part of screen where picture is required to be drawn, one line at a time so also called Vector Display</td></tr><tr><td>Cost</td><td>It is less expensive than Random Scan System.</td><td>It is Costlier than Raster Scan System.</td></tr><tr><td>Refresh Rate</td><td>Refresh rate is 60 to 80 frame per second</td><td>Refresh Rate depends on the number of lines to be displayed i.e 30 to 60 times per second</td></tr><tr><td>Picture Definition</td><td>It Stores picture definition in Refresh Buffer also called Frame Buffer.</td><td>It Stores picture definition as a set of line commands called Refresh Display File.</td></tr><tr><td>Line Drawing</td><td>Zig - Zag line is produced because plotted values are discrete.</td><td>Smooth line is produced because directly the line path is followed by electron beam.</td></tr><tr><td>Realism in display</td><td>It contains shadow, advance shading and hidden surface technique so gives the realistic display of scenes.</td><td>It does not contain shadow and hidden surface technique so it can not give realistic display of scenes.</td></tr><tr><td>Image Drawing</td><td>It uses Pixels along scan lines for drawing an image.</td><td>It is designed for line drawing applications and uses various mathematical function to draw.</td></tr></table>		Raster Scan System	Random Scan System	Resolution	It has poor or less Resolution because picture definition is stored as a intensity value.	It has High Resolution because it stores picture definition as a set of line commands.	Electron-Beam	Electron Beam is directed from top to bottom and one row at a time on screen, but electron beam is directed to whole screen.	Electron Beam is directed to only that part of screen where picture is required to be drawn, one line at a time so also called Vector Display	Cost	It is less expensive than Random Scan System.	It is Costlier than Raster Scan System.	Refresh Rate	Refresh rate is 60 to 80 frame per second	Refresh Rate depends on the number of lines to be displayed i.e 30 to 60 times per second	Picture Definition	It Stores picture definition in Refresh Buffer also called Frame Buffer .	It Stores picture definition as a set of line commands called Refresh Display File .	Line Drawing	Zig - Zag line is produced because plotted values are discrete.	Smooth line is produced because directly the line path is followed by electron beam.	Realism in display	It contains shadow, advance shading and hidden surface technique so gives the realistic display of scenes.	It does not contain shadow and hidden surface technique so it can not give realistic display of scenes.	Image Drawing	It uses Pixels along scan lines for drawing an image.	It is designed for line drawing applications and uses various mathematical function to draw.
	Raster Scan System	Random Scan System																										
Resolution	It has poor or less Resolution because picture definition is stored as a intensity value.	It has High Resolution because it stores picture definition as a set of line commands.																										
Electron-Beam	Electron Beam is directed from top to bottom and one row at a time on screen, but electron beam is directed to whole screen.	Electron Beam is directed to only that part of screen where picture is required to be drawn, one line at a time so also called Vector Display																										
Cost	It is less expensive than Random Scan System.	It is Costlier than Raster Scan System.																										
Refresh Rate	Refresh rate is 60 to 80 frame per second	Refresh Rate depends on the number of lines to be displayed i.e 30 to 60 times per second																										
Picture Definition	It Stores picture definition in Refresh Buffer also called Frame Buffer .	It Stores picture definition as a set of line commands called Refresh Display File .																										
Line Drawing	Zig - Zag line is produced because plotted values are discrete.	Smooth line is produced because directly the line path is followed by electron beam.																										
Realism in display	It contains shadow, advance shading and hidden surface technique so gives the realistic display of scenes.	It does not contain shadow and hidden surface technique so it can not give realistic display of scenes.																										
Image Drawing	It uses Pixels along scan lines for drawing an image.	It is designed for line drawing applications and uses various mathematical function to draw.																										
d.	<p>Consider a line AB with A= (0, 0) and B= (-5,-5). Apply a simple DDA algorithm and calculate the pixels on the line.</p> <p>Solutions:</p> <p>Trace for the algorithm for the line AB.</p> <p>Step 1: [Initialize the inputs]</p> <p>$X_1 = 0$ and $Y_1 = 0$</p> <p>$X_2 = -5$ and $Y_2 = -5$</p>																											

Step 2: [Calculate Δx and Δy]

$$\Delta x = x_2 - x_1 = 5 - 0 = 5$$

and

$$\Delta y = y_2 - y_1 = 5 - 0 = 5$$

Step 3: [Calculate length estimate]

$$L = \sqrt{\Delta x^2 + \Delta y^2} = 5 \text{ as } \Delta x = \Delta y$$

Step 4: [Calculate increment factors in x and y directions]

$$\Delta x = \Delta x / L = \Delta x / L = 5/5 = 1$$

and

$$\Delta y = \Delta y / L = \Delta y / L = 5/5 = 1$$

Step 5: [Initialize the points]

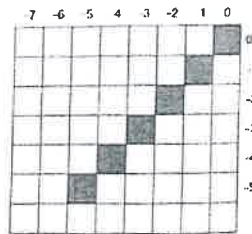
$$X_{\text{new}} = x_1 + 0.5 = 0 + 0.5 = 0.5$$

$$Y_{\text{new}} = y_1 + 0.5 = 0 + 0.5 = 0.5$$

Plot $(-1, -1)$

Figure 2.3 shows the screen plot of the line AB.

t	Plot	X_{new}	Y_{new}
	$(0, 0)$	0.5	0.5
1	$(-1, -1)$	-0.5	-0.5
2	$(-2, -2)$	-1.5	-1.5
3	$(-3, -3)$	-2.5	-2.5
4	$(-4, -4)$	-3.5	-3.5
5	$(-5, -5)$	-4.5	-4.5



- e. Explain the acceptance and rejection test using bit codes in Cohen-Sutherland line clipping algorithm. List the steps of the algorithm and give suitable example to explain the concept.

Solutions:

This is one of the oldest and most popular line clipping algorithm developed by Dan Cohen and Ivan Sutherland. To speed up the processing this algorithm performs initial tests that reduce the number of intersections that must be calculated. This algorithm uses a four digit (bit) code to indicate which of nine regions contain the end point of line. The four bit codes are called **region codes** or **outcodes**. These codes identify the location of the point relative to the boundaries of the clipping rectangle as shown in the Fig. 5.9.

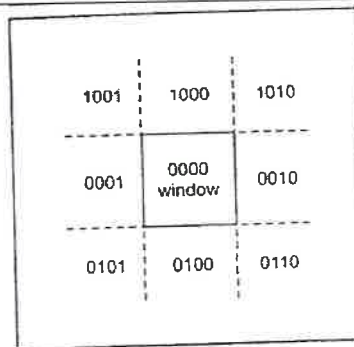


Fig. 5.9 Four-bit codes for nine regions

Otherwise, the bit is set to zero

Once we have established region codes for all the line endpoints, we can determine which lines are completely inside the clipping window and which are clearly outside. Any lines that are completely inside the window boundaries have a region code of 0000 for both endpoints and we trivially accept these lines. Any lines that have a 1 in the same bit position in the region codes for each endpoint are completely outside the clipping rectangle, and we trivially reject these lines. A method used to test lines for total clipping is equivalent to the logical AND operator. If the result of the logical AND operation with two end point codes is not 0000, the line is completely outside the clipping region. The lines that cannot be identified as completely inside or completely outside a clipping window by these tests are checked for intersection with the window boundaries.

Algorithm:

1. Read two end points of the line say $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$.
2. Read two corners (left-top and right-bottom) of the window, say (Wx_1, Wy_1) and (Wx_2, Wy_2) .
3. Assign the region codes for two endpoints P_1 and P_2 using following steps :
Initialize code with bits 0000
Set Bit 1 - if $(x < Wx_1)$
Set Bit 2 - if $(x > Wx_2)$
Set Bit 3 - if $(y < Wy_2)$
Set Bit 4 - if $(y > Wy_1)$
4. Check for visibility of line $P_1 P_2$
 - a) If region codes for both endpoints P_1 and P_2 are zero then the line is completely visible. Hence draw the line and go to step 9.
 - b) If region codes for endpoints are not zero and the logical ANDing of them is also nonzero then the line is completely invisible, so reject the line and go to step 9.
 - c) If region codes for two endpoints do not satisfy the conditions in 4a) and 4b) the line is partially visible.
5. Determine the intersecting edge of the clipping window by inspecting the region codes of two endpoints.
 - a) If region codes for both the end points are non-zero, find intersection points P_1' and P_2' with boundary edges of clipping window with respect to point P_1 and point P_2 , respectively
 - b) If region code for any one end point is non zero then find intersection point P_1' or P_2' with the boundary edge of the clipping window with respect to it.
6. Divide the line segments considering intersection points.
7. Reject the line segment if any one end point of it appears outside the clipping window.
8. Draw the remaining line segments.
9. Stop.

- f. Explain Liang-Barsky algorithm for clipping a line and also find the clipping coordinates for a line PQ where $P=(10,10)$ and $Q=(60,30)$, against window with $(x_{wmin}, y_{wmin})=(15,15)$ and $(x_{wmax}, y_{wmax})=(25,25)$.

Solution: The Liang-Barsky has developed the efficient algorithm for line clipping. The Parametric equations are given below :

Consider $P=P_1$ and $Q=P_2$ for below solution.

5

Q. P. Code:

$$x = x_1 + t\Delta x$$

$$y = y_1 + t\Delta y, \quad 0 \leq t \leq 1$$

where $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$

The point clipping conditions (Refer section 5.3.1) for Liang-Barsky approach in the parametric form can be given as

$$x_{wmin} \leq x_1 + t\Delta x \leq x_{wmax} \text{ and}$$

$$y_{wmin} \leq y_1 + t\Delta y \leq y_{wmax}$$

Liang-Barsky express these four inequalities with two parameters p and q as follows :

$$tp_i \leq q_i, \quad i = 1, 2, 3, 4$$

where parameters p and q are defined as

$$p_1 = -\Delta x, \quad q_1 = x_1 - x_{wmin}$$

$$p_2 = \Delta x, \quad q_2 = x_{wmax} - x_1$$

$$p_3 = -\Delta y, \quad q_3 = y_1 - y_{wmin}$$

$$p_4 = \Delta y, \quad q_4 = y_{wmax} - y_1$$

Following observations can be easily made from above definitions of parameters p and q .

- If $p_i = 0$: Line is parallel to left clipping boundary.
- If $p_2 = 0$: Line is parallel to right clipping boundary.
- If $p_3 = 0$: Line is parallel to bottom clipping boundary.
- If $p_4 = 0$: Line is parallel to top clipping boundary.
- If $p_i = 0$, and for that value of i ,
 - If $q_i < 0$: Line is completely outside the boundary and can be eliminated.
 - If $q_i \geq 0$: Line is inside the clipping boundary.

If $p_i < 0$: Line proceeds from outside to inside of the clipping boundary.

If $p_i > 0$: Line proceeds from inside to outside of the clipping boundary.

Therefore, for nonzero value of p_i , the line crosses the clipping boundary and we have to find parameter t . The parameter t for any clipping boundary i can be given as

$$t = \frac{q_i}{p_i} \quad i = 1, 2, 3, 4$$

Liang-Barsky algorithm calculates two values of parameter t : t_1 and t_2 that define that part of the line that lies within the clip rectangle. The value of t_1 is determined by checking

the rectangle edges for which the line proceeds from the outside to the inside ($p < 0$). The value of t_1 is taken as a largest value amongst various values of intersections with all edges. On the other hand, the value of t_2 is determined by checking the rectangle edges for which the line proceeds from the inside to the outside ($p > 0$). The minimum of the calculated value is taken as a value for t_2 .

Now, if $t_1 > t_2$, the line is completely outside the clipping window and it can be rejected. Otherwise the values of t_1 and t_2 are substituted in the parametric equations to get the end points of the clipped line.

Sol.: Here,

$$x_1 = 10 \quad x_{wmin} = 15$$

$$y_1 = 10 \quad y_{wmin} = 15$$

[TURN OVER]

6

$$\begin{aligned}
 x_2 &= 60 & x_{wmax} &= 25 \\
 y_2 &= 30 & y_{wmax} &= 25 \\
 p_1 &= -50 & q_1 &= -5 & p_1/q_1 &= 0.1 \\
 p_2 &= 50 & q_2 &= 15 & p_2/q_2 &= 0.3 \\
 p_3 &= -20 & q_3 &= -5 & p_3/q_3 &= 0.25 \\
 p_4 &= 20 & q_4 &= 15 & p_4/q_4 &= 0.75
 \end{aligned}$$

$$t_1 = \max(0.25, 0.1) = 0.25$$

since for these values $p < 0$

$$t_2 = \min(0.3, 0.75) = 0.3$$

since for these values $p > 0$

Here, $t_1 < t_2$ and the endpoints of clipped line are :

$$\begin{aligned}
 xx_1 &= x_1 + t_1 \Delta x \\
 &= 10 + 0.25 \times 50 \\
 &= 22.5
 \end{aligned}$$

$$\begin{aligned}
 yy_2 &= y_1 + t_1 \Delta y \\
 &= 10 + 0.25 \times 20 \\
 &= 15
 \end{aligned}$$

$$\begin{aligned}
 xx_2 &= x_1 + t_2 \Delta x \\
 &= 10 + 0.3 \times 50 \\
 &= 25
 \end{aligned}$$

$$\begin{aligned}
 yy_2 &= y_1 + t_2 \Delta y \\
 &= 10 + 0.3 \times 20 \\
 &= 16
 \end{aligned}$$

2. Attempt any three of the following:

15

a. Describe transformations and matrices in detail.

Solutions: The process of changing size, orientation or positions of objects by a mathematical operation is known as transformation. Two methods of transformations are

- Geometric Transformations
- Coordinate Transformations

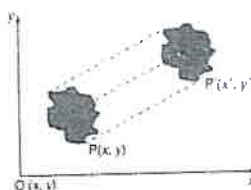


Figure 3.1 An image in a 2D coordinate system.

This image can be considered as a set of points, where every point in the image has coordinates (x, y) . If the image is transformed and moved to a new position then the point $P(x, y)$ transforms into $P'(x', y')$. The coordinates of P' can be obtained from the original point P of original image using geometrical definition of the transformation.

We need to represent the above transformation. This transformation is represented by a point-to-point transformation. The transformation is applied to x and y to get transform point x' and y' . While representing transformation, we need to have relation between (x, y) and (x', y') . This relation may be function, matrix, or system of equation.

The most general transformation is obtained using the relations

$$x = f(x, y), \quad y = g(x, y) \quad (3.1)$$

where f and g are function of x and y . We assume that the f and g are linear function of x and y for which we get a system of equation for the transformation as

$$x' = ax + by, \quad y' = cx + dy \quad (3.2)$$

When written as function of transformation, it is called transformation function given by

$$T(x', y') = (ax + by, cx + dy) \quad (3.3)$$

This transformation function may be used for phenomenon like scaling, reflection required for performing operation in 2D and 3D coordinate systems.

The clear use of transformations is to help simplify the task of geometric modeling, rendering, and animation. The detailed discussions on it can be found in the later part of the book.

3.2 Transformation matrix

Any object or image can be described as a set of points. For digital processing, matrices are used for representation of such objects. General rules of matrix algebra allow us to perform various operations on these objects. The same matrix could be used for numerous operations just by changing the individual elements of the matrix. Thus we would prefer to perform all the transformations using matrix called matrix of transformation or transformation matrix, say T . Suppose the matrix definition of original image and transformed images be A and B respectively. The transformed image B is obtained by operating a matrix T represented by $B = A \cdot T$, where the matrix T is called the geometrical operator matrix or transformation matrix. This matrix T

satisfies all the rules of matrix algebra. For 2D systems (image planes), T has dimension of 2×2 and for 3D (object space) has dimension 3×3 , as it requires (x, y) and (x, y, z) descriptions of the objects respectively.

We can also write a system of equation in a matrix form, where (x, y) and (x', y') are the coordinates of the original point P and transformed point P' , respectively. The general matrix transformation is given by

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad (3.4)$$

where a, b, c , and d are individual elements of the matrix for transformation capable of producing effects like scaling, reflection, rotation, and shearing.

Consider a point $P(x, y)$ with matrix definition given by $[x \ y]$. A transformation on this point is nothing but multiplication of this point and 2×2 transformation matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

and

$$[P][T] = [x \ y] \begin{bmatrix} a & b \\ c & d \end{bmatrix} = [ax + cy \ bx + dy] = P[x' \ y'] \quad (3.5)$$

- b Using homogeneous coordinate transformation matrix, rotate the triangle ABC with $A = (2, 3)$, $B = (5, 5)$, and $C = (4, 3)$ by an angle 45° about the point $(1, 1)$.

Solution To rotate an object about an arbitrary point (x_c, y_c) , we carry out sequence of three transformations:

- Translate point (x_c, y_c) to the origin.
- Rotate the point about origin.
- Translate back the origin to the point (x_c, y_c) .

The translation matrix to move (x_c, y_c) to origin is given by

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_c & -y_c & 1 \end{bmatrix}$$

The rotation matrix for anticlockwise rotation about origin is given by

$$T_R = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The translation matrix to move back the center to its original position is given by

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_c & y_c & 1 \end{bmatrix}$$

Therefore, the overall transformation can be obtained as

$$T = T_1 \cdot T_R \cdot T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_c & -y_c & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_c & y_c & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ -x_c \cos \theta + y_c \sin \theta + x_c & -x_c \sin \theta - y_c \cos \theta + y_c & 1 \end{bmatrix}$$

5

Therefore, the transformed triangle obtained by substituting the respective values is

$$T = \begin{bmatrix} A' \\ B' \\ C' \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 \\ 5 & 5 & 1 \\ 4 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1 & -\sqrt{2}+1 & 1 \end{bmatrix} = \begin{bmatrix} -1/\sqrt{2}+1 & 3/\sqrt{2}+1 & 1 \\ 1 & 8/\sqrt{2}+1 & 1 \\ 1/\sqrt{2}+1 & 5/\sqrt{2}+1 & 1 \end{bmatrix}$$

c. Write a short note on reflection through an arbitrary Plane in brief.

Solutions:

6.6.1 Reflection with Respect to xy Plane

Consider point $P(x, y, z)$. The reflection of this point with respect to xy plane is given by point $P'(x, y, -z)$, as shown in Fig. 6.8. Corresponding to this reflection the transformation matrix can be given as

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

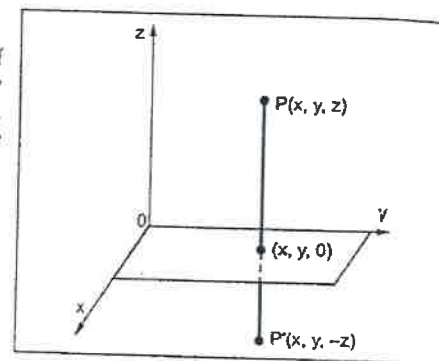


Fig. 6.8

6.6.2 Reflection with Respect to Any Plane

Often it is necessary to reflect an object through a plane other than $x = 0$ (yz plane), $y = 0$ (xz plane) or $z = 0$ (xy plane). Procedure to achieve such a reflection (reflection with respect to any plane) can be given as follows:

1. Translate a known point P_0 , that lies in the reflection plane to the origin of the co-ordinate system.
2. Rotate the normal vector to the reflection plane at the origin until it is coincident with +ve z axis, this makes the reflection plane $z = 0$ co-ordinate plane i.e. xy plane.
3. Reflect the object through $z = 0$ (xy plane) co-ordinate plane.
4. Perform the inverse transformation to those given above to achieve the result.

Let $P_0(x_0, y_0, z_0)$ be the given known point. Translate this point to the origin by using corresponding translation matrix

6.6.2 Reflection with Respect to Any Plane

Often it is necessary to reflect an object through a plane other than $x = 0$ (yz plane), $y = 0$ (xz plane) or $z = 0$ (xy plane). Procedure to achieve such a reflection (reflection with respect to any plane) can be given as follows:

1. Translate a known point P_o , that lies in the reflection plane to the origin of the co-ordinate system.
2. Rotate the normal vector to the reflection plane at the origin until it is coincident with +ve z axis, this makes the reflection plane $z = 0$ co-ordinate plane i.e. xy plane.
3. Reflect the object through $z = 0$ (xy plane) co-ordinate plane.
4. Perform the inverse transformation to those given above to achieve the result.

Let $P_o (x_o, y_o, z_o)$ be the given known point. Translate this point to the origin by using corresponding translation matrix

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_o & -y_o & -z_o & 1 \end{bmatrix}$$

Let the normal vector

$$N = n_1 I + n_2 J + n_3 K$$

$$|N| = \sqrt{n_1^2 + n_2^2 + n_3^2}$$

and

$$\lambda = \sqrt{n_2^2 + n_3^2}$$

10

As we want to match this vector with z axis, (so that the plane of reflection will be parallel to xy plane), we will use the same procedure as used in rotation.

$$R_{xy} = \begin{bmatrix} \frac{\lambda}{|N|} & 0 & \frac{n_1}{|N|} & 0 \\ \frac{-n_1 n_2}{\lambda |N|} & \frac{n_3}{\lambda} & \frac{n_2}{|N|} & 0 \\ \frac{-n_1 n_3}{\lambda |N|} & \frac{-n_2}{\lambda} & \frac{n_3}{|N|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As seen earlier for reflection about xy plane we have

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now for inverse transformation we have,

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_0 & y_0 & z_0 & 1 \end{bmatrix}$$

$$R_{xy}^{-1} = \begin{bmatrix} \frac{\lambda}{|N|} & \frac{-n_1 n_2}{\lambda |N|} & \frac{-n_1 n_3}{\lambda |N|} & 0 \\ 0 & \frac{n_3}{\lambda} & \frac{-n_2}{\lambda} & 0 \\ \frac{n_1}{|N|} & \frac{n_2}{|N|} & \frac{n_3}{|N|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Resultant transformation matrix can be given as

$$R_T = T \cdot R_{xy} \cdot M \cdot R_{xy}^{-1} \cdot T^{-1}$$

d Shear a unit cube situated at origin with a shear transformation matrix:

$$T_{shear} = \begin{bmatrix} 1 & -0.85 & 0.25 & 0 \\ -0.75 & 1 & 0.7 & 0 \\ 0.5 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solutions:

A volume matrix representing the origin situated cube vertices in homogeneous coordinates is given as:

Q. P. Code:

$$V = \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

The transformed coordinates of the vertices with coordinates $[x \ y \ z \ 1]$ are obtained by product operation $[x \ y \ z \ 1] \cdot T_{\text{data}}$

$$V' = V \cdot T_{\text{data}}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -0.85 & 0.25 & 0 \\ -0.75 & 1 & 0.7 & 0 \\ 0.5 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5 & 1 & 1 & 1 \\ 1.5 & 0.15 & 1.25 & 1 \\ 0.75 & 1.15 & 1.95 & 1 \\ -0.25 & 2 & 1.7 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & -0.85 & 0.25 & 1 \\ 0.25 & 0.15 & 0.95 & 1 \\ -0.75 & 1 & 0.7 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \\ E' \\ F' \\ G' \\ H' \end{bmatrix}$$

e. Define Vanishing point and explain vanishing point in different perspective projection in detail.
Solutions:--

A vanishing point is a point on the image plane of a perspective drawing where the two-dimensional perspective projections (or drawings) of mutually parallel lines in three-dimensional space appear to converge. When the set of parallel lines is perpendicular to a picture plane, the construction is known as one-point perspective, and their vanishing point corresponds to the oculus, or "eye point", from which the image should be viewed for correct perspective geometry.

Vanishing point in different perspective projections are: (explain in brief)

- One Point Perspective
- Two Point perspective
- Three Point Perspective

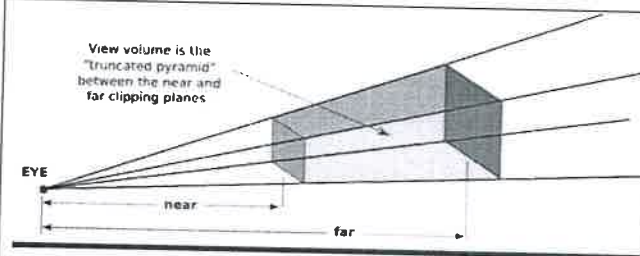
f. What is meant by view volume? Explain it with different kind of projection.

Solutions:

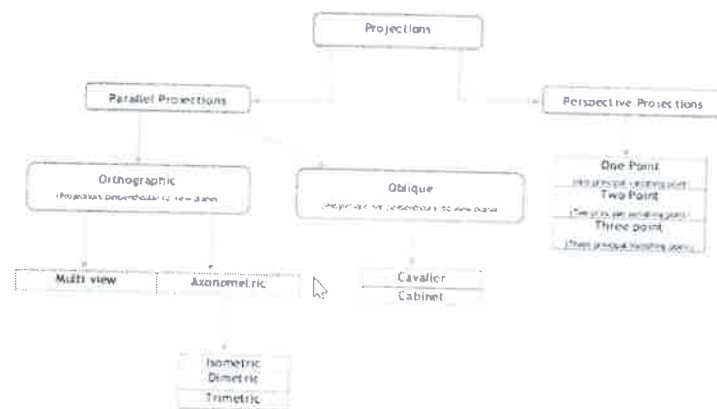
The volume of space that is actually rendered into the image is called the view volume. Things inside the view volume make it into the image; things that are not in the view volume are clipped and cannot be seen. For purposes of drawing, OpenGL applies a coordinate transform that maps the view volume onto a cube.

[TURN OVER]

12



Different types of projections are: (Define each of them with proper diagram and example)

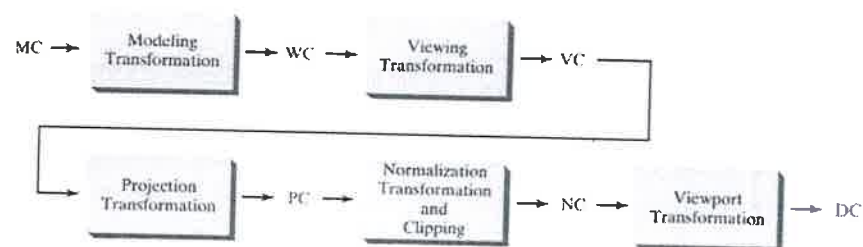


3. Attempt any three of the following:

a. Explain with neat labelled diagram stages in 3D Viewing pipeline.

15

Solutions : Explain each component in details.



- Modeling coordinates (MC)
- World coordinates (WC)
- Viewing coordinates (VC)
- Projection coordinates (PC)
- Normalized coordinates (NC)
- Device coordinates (DC)

b. Explain different coordinates systems and matrices in detail.

Solutions:

A coordinates system is use to unambiguously represents a point it contains a reference point(The origins and three linearly independent vectors (the basics)).

$$P = a_1 v_1 + a_2 v_2 + a_3 v_3 + 0$$

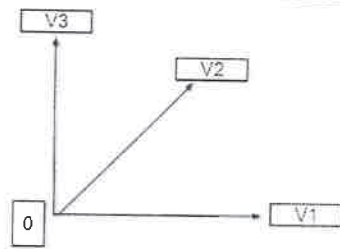


Fig: basic vectors

Different stages in 3D viewing generate and work on different coordinates system as given below. (Explain below points in details with matrix formation).

- ✓ Screen Coordinates Systems
- ✓ Local Coordinates system or 3d modelling coordinates
- ✓ World coordinates systems
- ✓ View reference coordinates
- ✓ Normalized projection coordinates or Normalized device coordinates.
- ✓ 2D Device coordinates

c. What is light? Explain Radiometry in brief.

Solutions: --

Light is electromagnetic radiation. What we see as visible light is only a tiny fraction of the electromagnetic spectrum, which extends from very-low-frequency radio waves through microwaves, infrared, visible and ultraviolet light to x-rays and ultra-energetic gamma rays. Our eyes respond to visible light; detecting the rest of the spectrum requires an arsenal of scientific instrument ranging from radio receivers to scintillation counters.

Radiometry is science of measuring light in any portion of the electromagnetic spectrum. There are two aspects of radiometry:

1. Theory
2. Practice

Practice: the practice involves the scientific instrument and materials used in measuring lights, including radiations thermocouples, bolometers, photodiodes, photosensitive dyes and emulsions, vacuum phototubes, charge-coupled device and a plethora of others.

Theory: Radiometry theory is mainly based on mathematics and physical discussions which is described below.

- > Radiant Energy
- > Spectral Radiant Energy
- > Radiant Flux (Radiant Power):
- > Spectral Radiant Flux (Spectral Radiant Power)
- > Radiant Flux Density (Irradiance and Radiant Exitance)
- > Spectral Radiant Flux Density Spectral radiant flux density is radiant flux per unit wavelength interval at wavelength λ .
- > Radiance:

$$L = d^2\Phi / [dA(d\omega \cos \theta)]$$

- > Radiant Intensity

d. Explain different properties of Bidirectional Reflectance Distribution Function (BRDF).

Solutions:--

1. **Domain:**-- For a particular surface point x, the BRDF is a four dimensional function:

Two dimensions to specify the incoming direction, and two dimensions to specify the outgoing direction. Furthermore, if the BRDF is allowed to vary spatially over an object's surface, this leads to an additional two dimensions.

2. **Range:** -- The BRDF can take on any positive value.

3. **Reciprocity:** -- The value of the BRDF remains unchanged if the incident and outgoing directions are swapped. Reciprocity means that surface reflection is invariant to the direction of the light flow, i.e., the reflected radiance remains unchanged if the light and camera positions are swapped. This property is essential for many global illumination algorithms and allows light to be traced either in the forward or backward direction.

4. **Energy Conservation:** - Due to energy conservation, a surface cannot reflect more light than it receives.

e. Explain any two-color spaces in detail.

[TURN OVER]

12

Solutions:

A range of colors can be created by the primary colors of pigment and this colors then defines a specifics color space. Color space are also known as color model . is an abstract mathematical model which simply describes the range of colors as tuples of numbers, typically as 3 or 4 values or color components.(RGB).

Different types of colors space are: Explain each one of them with neat label diagram



1. RGB color space
2. LMS color space
3. CIELAB color space
4. CIE's XYZ color space

f. Write a short note on chromatic adaptation.

Solutions:

Chromatic adaptation. Chromatic adaptation is the human visual system's ability to adjust to changes in illumination in order to preserve the appearance of object colors. ... On the other hand, a camera with no adjustment for light may register the apple as having varying color.

There are three types of adaptations:

1. Light adaptations : It refers to the changes occurs when we move from vary dark to very light environment.
i.e Dark  Light
2. Dark adaptations: It refers to the changes occurs when we move from vary light to very dark environment.
i.e light  dark
3. Cromatic adaptation :

Chromatic adaptation is the human visual system's ability to adjust to changes in illumination in order to preserve the appearance of object colors. ... This feature of the visual system is called chromatic adaptation, or color constancy; when the correction occurs in a camera it is referred to as white balance.

4. Attempt any three of the following:

a. Write a short note on back face removal technique.

15

Solutions:

Back face removal, a simple and fast object space algorithm, is also called *back face culling* where polygons (usually triangles) that are not facing the camera are removed from the rendering pipeline. No faces on the back side of the object will be displayed. Since in general about half of the faces of objects are back faces, this algorithm will remove about half of the total polygons in the image. Primitives or batches of primitives can be rejected in their entirety that usually reduces the load on a well-designed system. This is done by comparing the polygon's surface normal with the position of the camera. (The very basic rendering pipeline for the back face removal includes the following factors:

- Polygon data is defined in the object space.
- Polygons in an object space are transformed into the world space.
- Polygons in the world space are transformed into a camera space.
- Back face culling is performed.
- Front-facing polygons are rendered.

The first step in back face culling is the loading of the polygon data. The way the polygon is mathematically described will impact how back face culling is done. Most of the time, the data structure of a polygon is a mesh. Consider a sphere in a 3D coordinate system, as shown in Figure 9.2. From the figure, we can see that some of the triangles that make up the sphere can be seen, and some of them cannot. The triangles on the front of the sphere can be seen, while the triangles on the back of the sphere cannot be seen. The triangles that can be seen are considered to be facing toward the camera. These triangles are defined as front-facing. The triangles that cannot be seen are considered to be facing away from the camera. These triangles are called back-facing.

15

Q. P. Code:

We can determine which direction the face is facing by finding a vector that is perpendicular to the plane on which the face lies.

Rendering requires fixing up the coordinate system we are working with. You can choose any coordinate system and any orientation—clockwise or counter-clockwise—you just have to be consistent and adjust your definition of “front” and “back” appropriately. For a left-handed coordinate system, the x -axis values increase from left to right, the y -axis values increase from down to up, and the z -axis values increase from backward to forward. That is, the z -axis value increases as objects move away from the viewer. In order to move an object closer to yourself, decrease its z -axis location value. Moving an object in a negative z -direction brings it closer to the viewer.

Typically, the back face test involves calculating the dot product between the polygon's normal and the vector formed from the viewing point to any point on the polygon. If the result is negative then the polygon is facing towards the viewer, if the result is positive then the polygon is facing away from the viewer and is considered to be back facing.

The culling algorithm uses a rule that solid objects are made of polygons, closed, and surface normal points into open space (outward direction). If a polygon on the object is drawn in a counter-clockwise direction, the normal calculated by cross products (Newell's method) will point out from the visible side.

Let N be a surface normal of the polygon and let $E = (p_x - e_x, p_y - e_y, p_z - e_z)$ be the direction vector from the eye to any point P on the polygon. Since we are only interested in the direction, neither N nor E needs to be a unit vector. The polygon's visible face cannot be seen by the viewer if $N \cdot E \geq 0$. For a right-handed system, the polygon face is a back face if $N \cdot E < 0$.

If the scene is in eye coordinates, so that $E = (0, 0, 1)$, then the polygon's visible face cannot be seen by the viewer if the z component (N_z) of the surface normal is greater than or equal to zero.

In general let $N = (A, B, C)$ be the normal vector of the a planar polygonal face with N pointing in the direction of the polygon is facing. If the direction of the viewing is the direction of the $-z$ -axis, the polygon face is facing away from the viewer when $C \geq 0$. The angle between z -axis and N is acute ($< 90^\circ$). The polygonal face is back face when θ is acute and front face when θ is obtuse.

The reason for including $C = 0$ is that the face is parallel to the line of sight, and the projection is either hidden or overlaps with the edges.



b What is meant by BSP trees? Explain algorithm for construction of it with example.

Solutions:

A Binary Space Partitioning Tree (or BSP Tree) is a data structure that is used to organize objects within a space. Within the field of computer graphics, it has applications in hidden surface removal and ray tracing.

9.11 Binary space partitioning trees

In computer graphics, it is desirable that the drawing of a scene be both correct and quick. A simple way to draw a scene correctly is the painter's algorithm: draw it from back to front painting the background over with each closer object. However, the approach is quite limited since time is wasted in drawing objects that will be overdrawn later, and not all objects will be drawn correctly.

Z-buffering can ensure that scenes are drawn correctly and eliminate the ordering step of the painter's algorithm, but it is expensive in terms of memory use. Binary space partitioning (BSP) trees will split up objects so that the painter's algorithm will draw them correctly without need of a z -buffer and eliminate the need to sort the objects; as a simple tree traversal will yield them in the correct order. It also serves as a base for other algorithms, such as visibility lists, which seek to reduce overdraw.



Scanned with
CamScanner

[TURN OVER]

This approach was originally proposed in 3D computer graphics to increase the rendering efficiency of the algorithms. Some other well-known applications include performing geometrical operations with shapes (constructive solid geometry) in CAD, collision detection in robotics and 3D computer games, and other computer applications that involve handling of complex spatial scenes.

BSP is a method for recursively subdividing a space into convex sets by hyperplanes. This subdivision gives rise to a representation of a scene by means of a tree data structure known as a BSP tree. In other words, a BSP tree represents a recursive, hierarchical partitioning, or subdivision, of n -dimensional space into convex sub-spaces. The BSP tree construction is a process that takes a subspace and partitions it by any hyperplane that intersects the interior of that subspace. It is a method of breaking up intricately shaped polygons into convex sets, or smaller polygons consisting entirely of non-reflex angles (angles smaller than 180°). The result is two new sub-spaces that can be further partitioned by recursive application of the method.

A hyperplane in an n -dimensional space is an $n - 1$ dimensional object that can be used to divide the space into two half-spaces. For example, in a 3D space, a hyperplane is a plane. In a 2D space, a line is used. BSP trees are extremely versatile, because they are powerful sorting and classification structures. They have uses ranging from hidden surface removal and ray tracing hierarchies to solid modeling and robot motion planning.

BSP trees are closely related to quadrees and octrees. Quadrees and octrees are space partitioning trees that recursively divide sub-spaces into four and eight new sub-spaces, respectively. A BSP tree can be used to simulate both of these structures.

Let us closely look at the building of a BSP tree. A binary space partitioning is a generic process of recursively dividing a scene into two until the partitioning satisfies one or more requirements. The precise method of division varies depending on its final purpose. The final number of objects will inevitably increase, since lines or faces that cross the partitioning plane must be split into two, and it is also desirable that the final tree remains reasonably balanced (see Figure 9.16). Therefore, the algorithm for correctly and efficiently creating a good BSP tree is the most difficult part of an implementation. In a 3D space, planes are used to partition and split an object's faces; in a 2D space, lines split an object's segments.

Given a set of polygons in a 3D space, we want to build a BSP tree that contains all of the polygons. The following is the algorithm to build a BSP tree:

- Select a partition plane.
- Partition the set of polygons with the plane.
- Recurse with each of the two new sets.

Consider another example, as in Figure 9.17, of partitioning an irregular polygon into a series of convex ones.

Notice how each step produces polygons with fewer segments until arriving at G and F, which are convex and require no further partitioning. In this particular case, the partitioning line was picked between existing vertices of the polygon and intersected none of its segments. If the partitioning line intersects a segment, or face in a 3D model, the offending segments or



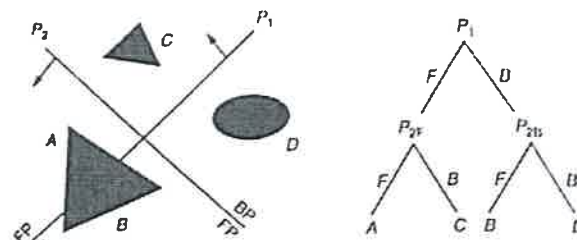


Figure 9.16 A BSP partition for a region of space.

faces have to be split into two at the line/plane, because each resulting partition must be a full, independent object (see Figure 9.17).

Note that in Figure 9.17(a), A is the root of the tree. In Figure 9.17(b), A is split into B and C, and in Figure 9.17(c), B splits into D and E. Finally in figure 9.17(d), D splits into F and G, that are convex and hence become leaves on the tree.

The choice of a partition plane depends on how the tree will be used, and what sort of efficiency criteria you have for the construction. Partitioning a set of polygons with a plane is done by classifying each member of the set with respect to the plane. If a polygon lies entirely to one side or the other of the plane, then it is not modified, and is added to the partition set for the side that it is on. If a polygon spans the plane, it is split into two or more pieces, and the resulting parts are added to the sets associated with either side as appropriate.

The decision to terminate the tree construction is, again, a matter of the specific application. Some methods terminate when the number of polygons in a leaf node is below a maximum value. Other methods continue until every polygon is placed in an internal node. Another criterion is a maximum tree depth.

Partitioning a polygon with a plane is a matter of determining which side of the plane the polygon is on. This is referred to as a front/back test and is performed by testing each point in the polygon against the plane. If all of the points lie to one side of the plane, then the entire

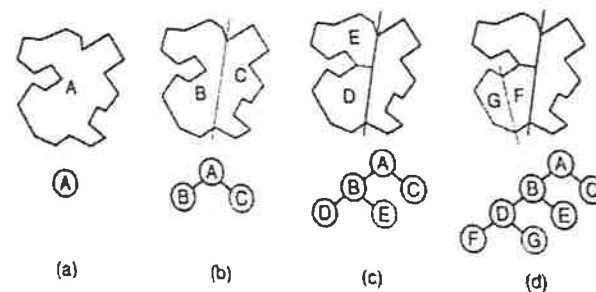


Figure 9.17 A binary space partition for an irregular polygon.

18

polygons on that side and does not need to be split. If some points lie on both sides of a plane, then the polygon is split into two or more pieces. The basic algorithm is to loop across all the edges of the polygon and find those for which one vertex is on each side of the partition plane. The intersection points of these edges and the plane are computed, and those points are used as new vertices for the resulting pieces.

Classifying a point with respect to a plane is done by passing the $[x \ y \ z]$ values of the point into the plane equation, $ax + by + cz + d = 0$. The result of this operation is the distance from a plane to a point along the plane's normal vector. It will be positive, if the point is on the side of the plane pointed to by the normal vector, negative otherwise. If the result is 0, the point is on the plane. For those not familiar with the plane equation, the values a , b , and c are the coordinate values of the normal vector. The value of d can be calculated by substituting a point known to be on the plane for x , y , and z into the plane equation.

Convex polygons are generally easier to deal with in a BSP tree construction than concave ones, because splitting them with a plane always results in exactly two convex pieces. Furthermore, the algorithm for splitting convex polygons is straightforward and robust.

When building a BSP tree, specifically for a hidden surface removal, the partition planes are usually chosen from the input polygon set. However, any arbitrary plane can be used, if there are no intersecting or concave polygons.

If the eye point is classified as being on the partition plane, the drawing order is unclear. This is not a problem if the rendering routine is smart enough to not draw polygons that are not within the viewing frustum. It is possible to substantially improve the algorithm by including the viewing direction vector in the computation. You can determine that entire subtrees are behind the viewer by comparing the view vector to the partition plane normal vector.

Since the usefulness of a BSP tree depends upon how well it was generated, a good algorithm is essential. Most algorithms will test many possibilities for each partition until finding a good compromise and might also keep backtracking information in memory so that if a branch of the tree is found to be unsatisfactory other alternative partitions may be tried. Therefore, producing a tree usually requires long computations.

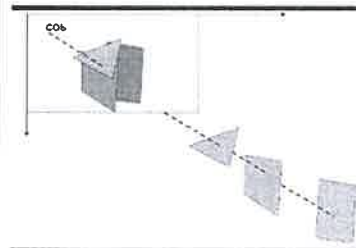
BSP trees are also used to represent natural images. Construction methods of BSP trees of images were first introduced as efficient representations, where only few hundred nodes can represent an image that normally require hundreds of thousands of pixels. Fast algorithms have also been developed to construct BST trees of images using computer vision and signal processing algorithms. These algorithms in conjunction with advanced entropy coding and signal approximation approaches were used to develop image compression methods.

c. Explain Visible surface ray tracing in brief with neat labelled diagram.

Solution:

Ray tracing is a technique used in computer graphics to create very realistic looking image. We just need to specify the position of cameras, the lights and the objects in the scene, and a ray tracer will perform the necessary computations to cast shadows and light upon objects.

The intensity of a pixel in an image is due to a ray of light, having been reflected from some objects in the scene, pierced through the center of the pixel.



A ray is fired from the centre of projection through each pixel to which the window maps, to determine the closest object intersected. So, visibility of surfaces can be determined by tracing a ray of light from the centre of projection (viewer's eye) to objects in the scene, (backward ray tracing). It performs the following steps:--

- Find out ray of light intersects on which objects.
- Determine which one of these objects is closest to the viewer.
- Set the pixel color to that object which is closest.

Mathematically, the intersection is of finding the roots of the equation:

$$\text{Surface} - \text{RAY} = 0$$

With,

$$\text{Ray Equation : } r(t) = t(P-C)$$

There are three Cases:--

Case 1:- No real roots

Case 2:- One real root

Case 3:- Two real roots

Surface and ray do not intersect

Ray tangentially grazes the surfaces

From both intersections, get the one with the smallest value of t .

Disadvantages: --

1. Suitable for complex curved surfaces.
2. Computationally expensive.
3. Need of an efficient ray-surface intersection technique.
4. Almost all visible effects can be generated.
5. Can be parallelized.
6. Has aliasing problems.

d. Explain Parametric representation of hyperbola in brief.

Solutions:

Hyperbolas in Parametric Form

There are two ways to write the parametric form for a hyperbola. You can choose the form that best fits your needs.

In the first form, a horizontal hyperbola can be described by the equation:

$$F(t)x(t)y(t)=(x(t),y(t))=asec(t)=btan(t)$$

A vertical hyperbola can be described by the equation:

$$F(t)x(t)y(t)=(x(t),y(t))=btan(t)=asec(t)$$

The second form of the parametric equation for a hyperbola uses two functions known as hyperbolic sine (sinh) and hyperbolic cosine (cosh). These functions are defined as:

$$\sinh(x)\cosh(x)=\frac{1}{2}(e^x-e^{-x})=\frac{1}{2}(e^x+e^{-x})$$

You can put a horizontal hyperbola into parametric form using these functions:

$$F(t)x(t)y(t)=(x(t),y(t))=acosh(t)=bsinh(t)$$

And

$$x(t)y(t)=-acosh(t)=bsinh(t)$$

The first set of parameters traces the right half of the hyperbola, while the second set traces the left half. This form of the parametric equation is especially useful for describing the motion of objects that only trace one half of a hyperbola.

e. Explain implicit and explicit curve representation in detail.

Solutions:--

1. Implicit Representation:-- In two dimensions, an implicit curves can be represented by the equation

$f(x,y)=0$. The implicit form is less coordinate-system dependent than is the explicit form. In three dimensions, the implicit form

$$f(x,y,z)=0$$

Curves in three dimensions are not as easily represented in implicit form.

We can represent a curve as the intersection, if it exists of the two surfaces:--

$$F(x,y,z)=0, g(x,y,z)=0$$

2. Explicit Representation :-- The explicit form of a curve in two dimensions gives the value of one variable i.e. the dependent variable, in terms of other independent variable.

In x,y space, it is written as, $y=f(x)$.

f. Explain Bezier Surfaces in detail and state it's any five properties.

Solution:

8.10 Bezier surfaces

Bezier surfaces are a species of a mathematical spline used in computer graphics, computer-aided design (CAD), and finite element modeling. As with the Bezier curve, a Bezier surface is defined by a set of control points. Similar to interpolation in many respects, the key difference is that the surface does not, in general, pass through the central controls points. Rather it is stretched towards those points as though each is an attractive force. They are visually intuitive and for many applications, mathematically convenient.

Bezier surfaces were first described in 1972 by the French engineer Pierre Bezier who used them to design automobile bodies. Bezier surfaces can be of any degree, but bicubic Bezier surfaces generally provide enough degrees of freedom for most applications.

A Bezier patch is a 3D surface generated from the Cartesian product of two curves. A given Bezier surface of an order (m, n) is defined by a set of $(m+1)(n+1)$ control points, $P_{i,j}$. It maps the unit square into a smooth and continuous surface embedded within a space of the same dimensionality as $\{P_{i,j}\}$.

A two-dimensional (2D) Bezier surface can be defined as a parametric surface where the position of a point P as a function of the parametric coordinates u, v is given by

$$P(u, v) = \sum_{i=0}^m \sum_{j=0}^n B_i^m(u) B_j^n(v) P_{i,j} \quad (8.19)$$

It is evaluated over the unit square, where the two one-dimensional basis functions (Bernstein polynomials) are defined as follows:

$$B_i^m(u) = \binom{m}{i} u^i (1-u)^{m-i} \quad \text{and} \quad B_j^n(v) = \binom{n}{j} v^j (1-v)^{n-j} \quad (8.20)$$

where

$$\binom{m}{i} = \frac{m!}{i!(m-i)!} \quad \text{and} \quad \binom{n}{j} = \frac{n!}{j!(n-j)!}$$

are binomial coefficients.

Generally, the most common use of Bezier surfaces is as nets of bicubic patches (where $m=n=3$). The geometry of a single bicubic patch is thus completely defined by a 4×4 grid of 16 control points, giving a compact mathematical definition of such a surface without the need to store each of the interpolated points. It is a generalization of the Bezier curve, in which each of the four rows of the control points can be thought of as a Bezier curve in 2D. These are typically linked up to form a B-spline surface in a similar way to the way Bezier curves are linked up to form a B-spline curve. A Bezier patch representation is powerful for its analytical description of the surface that may then be easily manipulated.

These evenly spaced control points form a surface made up of nine rectangular subpatches. These control points can be thought of as specifying the desired shape of the patch; it will attain this shape within the limits imposed by smoothness and continuity. A Bezier patch is generated above the control point grid and interprets the shape of the grid to create a surface that is smooth and continuous. A Bezier patch does not necessarily pass through all of its control points—only the four corner points of the control grid are guaranteed to lie on the surface of the patch.

A Bezier patch is defined by a 4×4 matrix, P , that contains the heights of the 16 control points. A patch is generated by the function, $P(u, v)$, for values of u and v that are between

2.1.1. A Bézier surface is a smooth surface defined by a set of control points. It is a generalization of the Bézier curve to two dimensions. The surface is defined by a set of control points P_{ij} where i and j range from 0 to n . The surface is defined by the equation:

$$P(u, v) = \sum_{i=0}^n \sum_{j=0}^n B_i^n(u) B_j^n(v) P_{ij} \quad (8.1)$$

where

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i} \quad \text{and} \quad B_j^n(v) = \binom{n}{j} v^j (1-v)^{n-j} \quad (8.2)$$

The Bézier surface is a smooth surface. One of the advantages of Bézier patches is that they allow a much more concise representation than vertex or polygon lists. Other virtues of parametric surfaces are that they provide exact analytical descriptions of surfaces and permit easy deformations of those surfaces. Finally, if a texture mapping is being performed, parametric surfaces are convenient, because the (u, v) parameters used to generate the surface can easily be reused as (u, v) texture parameters.

Some properties of Bézier surfaces include:

- **Boundary:** $P(u, v)$ passes through the control points at the four corners of the control net. The $P(u, v)$ in the patch, corresponding to the corners of the deformed unit square, coincide with four of the control points: $P_{0,0}$, $P_{m,0}$, $P_{0,n}$, and $P_{m,n}$. In fact, we have $P(0, 0) = P_{0,0}$, $P(1, 0) = P_{m,0}$, $P(0, 1) = P_{0,n}$, and $P(1, 1) = P_{m,n}$. However, a Bézier surface does not generally pass through its other control points.
- **Non-negativity:** $B_i^n(u) B_j^n(v)$ is non-negative for all m, n, i, j, u , and v in the range of 0 and 1.
- **Partition of unity:** The sum of all $B_i^n(u) B_j^n(v)$ is 1 for all u and v in the range of 0 and 1. More precisely, this means for any pair of u and v in the range of 0 and 1, the following holds:

$$\sum_{i=0}^m \sum_{j=0}^n B_i^n(u) B_j^n(v) = 1 \quad (8.3)$$

- **Convex hull property:** A Bézier surface $P(u, v)$ lies in the convex hull defined by its control net. Since $P(u, v)$ is a linear combination of all its control points with positive coefficients whose sum is 1 (partition of unity), the surface lies completely within the convex hull of its control points and, therefore, also completely within the bounding box of its control points in any given Cartesian coordinate system.



Figure 8.16: Bezier surface

- **Affine invariance:** This means that to apply an affine transformation to a Bezier surface, one can apply the transformation to all control points and the surface defined by the transformed control points is identical to the one obtained by applying the same transformation to the surface's equation. A Bezier surface will transform in the same way as its control points under all linear transformations and translations.
- All $u = \text{constant}$ and $v = \text{constant}$ lines in the (u, v) space, and, in particular, all the four edges of the deformed (u, v) unit square are Bezier curves (see Figure 8.16).

Bezier patch meshes are superior to meshes of triangles as a representation of smooth surfaces, since they are much more compact, easier to manipulate, and have much better continuity properties. In addition, other common parametric surfaces, such as spheres and cylinders, can be well approximated by relatively small numbers of cubic Bezier patches.

However, Bezier patch meshes are difficult to render directly. One problem with Bezier patches is that calculating their intersections with lines is difficult, making them awkward for pure ray tracing or other direct geometric techniques that do not use subdivision or successive approximation techniques. They are also difficult to combine directly with perspective projection algorithms.

5. Attempt any three of the following:

a. What is an animation? Explain any two principles of animation in detail.

15

Solution:

Computer animation is the process used for generating animated images. The more general term computer-generated imagery (CGI) encompasses both static scenes and dynamic images, while computer animation only refers to the moving images. Modern computer animation usually uses 3D computer graphics, although 2D computer graphics are still used for stylistic, low bandwidth, and faster real-time renderings. Sometimes, the target of the animation is the computer itself, but sometimes film as well.

Principles of animations are : **Explain any two of them in detail**

- ✓ Timing
- ✓ Ease In and Out
- ✓ Arcs
- ✓ Anticipation
- ✓ Exaggerations
- ✓ Squash and Stretch
- ✓ Secondary action
- ✓ Follow through and overlapping action
- ✓ Straight ahead action and Pose to Pose action.
- ✓ Staging
- ✓ Appeal
- ✓ Personality

b. Explain procedural techniques in brief.

Solutions:--

In this technique, the characters are animated by a set of rules (i.e. a procedure) and not by key framing. It is also known as **algorithmic animation**.

In this technique Animation is generated by writing a program that results the position/shape/whatever of the view of object the animator wants to display.

Generally :--

- The animator defines the object and their transformations.
- He also defines set of rules for how the system will behave (i.e. sequence of animation)
- And choose initial conditions for the world.
- Then Run the program to simulate rules, to guide what happens.

Rules are often based on physical rules of the real world expressed by mathematical equations.

The basic idea behind this technique is that the moments of an objects can be made procedurally.

The most common technique among number of animation techniques to create the animations procedurally is based on simulating physics referred as physically-based animation.

With Procedural techniques, it is possible to simultaneously do the computation and generation of animation, the results of procedural animator hasn't created individual frames.

Advantages :--

Once the program is created, lots of motion can be generated using it.

Disadvantages:--

It is hard to control the animation in this technique.

Categories of procedural animation:---(Define each term)

1. Physics-based animation:-- Its deals with objects which are not alive.

Its includes:--

a) particle system

	<p>b)flexible dynamics c)rigid body dynamics d)fluid dynamics e)hair/fur dynamics</p> <p>2.alife(Artificial life):--Artificial life deals with objects which are virtually alive. Its includes:-- a)Behavioural animation b)Artificial evaluation c)Branching object generation</p>	
c.	<p>Explain different types of deformation in detail.</p> <p>Solutions:</p> <p>Deformation is changes in an objects shapes or forms due to the applications of a force or forces. Types of deformer are: Explain each of them in brief with diagram.</p> <ol style="list-style-type: none"> 1. Bone Deformer (Skeleton deformations) 2. Curve Deformer (Skinning deformations) 	
d.	<p>What is an Image? Give any five formats of it.</p> <p>Solutions:</p> <p>An image is a visual representation of something. In information technology, the term has several usages: An image is a picture that has been created or copied and stored in electronic form. An image can be described in terms of vector graphics or raster graphics. An image stored in raster form is sometimes called a bitmap. An image map is a file containing information that associates different locations on a specified image with hypertext links.</p> <p>Different file formats of images are given below:(explain any five in details).</p> <ul style="list-style-type: none"> ✓ JPEG ✓ GIF ✓ PNG ✓ SVG ✓ TIFF ✓ BMP ✓ BPG ✓ WebP ✓ CGM 	
e.	<p>Distinguish between lossy and lossless compression.</p> <p>Solutions:</p>	

29

BASIS FOR COMPARISON	LOSSY COMPRESSION	LOSSLESS COMPRESSION
Basic	Lossy compression is the family of data encoding method that utilizes imprecise estimates to represent the content.	Lossless compression is a group of data compression algorithms that permits the original data to be accurately rebuilt from the compressed data.
Algorithm	Transform coding, DCT, DWT, fractal compression, RSSMS.	RLW, LZW, Arithmetic encoding, Huffman encoding, Shannon Fano coding.
Used in	Images, audio and video.	Text or program, images and sound.
Application	JPEG, GUI, MP3, MP4, OGG, H-264, MKV, etc.	RAW, BMP, PNG, WAV, FLAC, ALAC etc.
Data-holding capacity of the channel	More	Less as compared to lossy method

f. Explain the concept of histogram equalization. Equalize the following histogram for $L=8$.

Gray Level	0	1	2	3	4	5	6	7
No. of pixel	790	1023	850	656	329	245	122	81

Solutions:

Histogram equalization is a technique for adjusting image intensities to enhance contrast. Let f be a given image represented as a $m \times n$ matrix of integer pixel intensities ranging from 0 to $L - 1$. L is the number of possible intensity values, often 256.

25

Q. P. Code:

Gray Level (r_k)	No. of pixels n_k	PDF $P(r_k) = (n_k/n)$	CDF (S_k)	$(L-1) \times S_k$	Round off
0	790	0.19	0.19	1.33	1
1	1023	0.25	0.44	3.08	3
2	850	0.21	0.65	4.55	5
3	656	0.16	0.81	5.67	6
4	329	0.08	0.89	6.23	6
5	245	0.06	0.95	6.65	7
6	132	0.03	0.98	6.86	7
7	81	0.02	1.00	7	7
$\sum n_k = 4096$					

Note to calculate :-

$$PDF = \frac{n_k}{\sum n_k}$$

\therefore eg:- $\frac{790}{4096} = 0.19$ (Similarly for other values pdf has to be calculated).

CDF :- To calculate CDF \rightarrow Formula \rightarrow refer algorithm steps.

\therefore 1st value of pdf \rightarrow consider

then next, i.e. $0.19 + 0.25 = 0.44$

Next value $\rightarrow 0.44 + 0.21 = 0.65$

\therefore So, on, other values has to be calculated.

[TURN OVER]

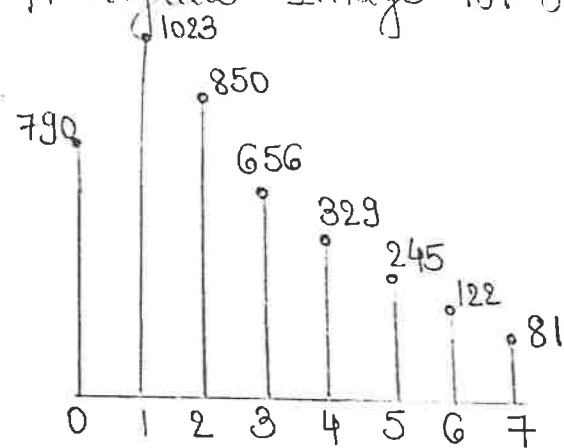
26

To find New gray level values.

Gray Levels	Old gray level	No. pixel	New gray level
0	1023	790	1
1		1023	3
2		850	5
3		656	6
4		329	6
5		245	7
6		122	7
7		81	7

\rightarrow Common: $1023 \rightarrow 255$
 \rightarrow Common: $245 \rightarrow 22 \sim 2$
 $= 448$

Histogram Image for old gray level.



Histogram Image for New gray level.

