SYLLABUS

CSE-303-E

COMPUTER GRAPHICS

Sessional : 50 Marks
Theory : 100 Marks
Total : 150 Marks
Duration of Exam. : 3 Hrs.

Unit I Introduction to Computer Graphics: What is Computer Graphics, Computer Graphics Applications, Computer Graphics Hardware and software, Two dimensional Graphics Primitives: Points and Lines, Line drawing algorithms: DDA, Bresenham's; Circle drawing algorithms: Using polar coordinates, Bresenham's circle drawing, mid point circle drawing algorithm; Filled area algorithms: Scanline: Polygon filling algorithm, boundary filled algorithm.

Unit II Two/Three Dimensional Viewing: The 2-D viewing pipeline, windows, viewports, window to view port mapping; Clipping: point, clipping line (algorithms):- 4 bit code algorithm, Sutherland-cohen algorithm, parametric line clipping algorithm (Cyrus Beck). Polygon clipping algorithm: Sutherland-Hodgeman polygon clipping algorithm. Two dimensional transformations: transformations, translation, scaling, rotation, reflection, composite transformation. Three dimensional transformations: Three dimensional graphics concept, Matrix representation of 3-D Transformations, Composition of 3-D transformation.

Unit III Viewing in 3D: Projections, types of projections, the mathematics of planner geometric projections, coordinate systems.

Unit IV Hidden surface removal: Introduction to hidden surface removal. The Z- buffer algorithm, scanline algorithm, area sub-division algorithm.

Unit V Representing Curves and Surfaces: Parametric representation of curves: Bezier curves, B-Spline curves. Parametric representation of surfaces: Interpolation method.

Unit VI Illumination, shading, image manipulation: Illumination models, shading models for polygons, shadows, transparency. What is an image? Filtering, image processing, geometric transformation of images.

Note: In the semester examination, the examiner will set eight questions in all, at least one question from each unit & students will be required only 5 questions.

COMPUTER GRAPHICS

Dec - 2013

Paper Code:-CSE-303-F

Note: ***Question No. 1 is compulsory, and attempt one question from each of the four sections. All questions carry equal marks.***

Q.1(a) What are points and lines? Explain in detail. (5)

Ans. Point : *A position in a plane is known as a point.* Any point can be represented by any ordered pair of numbers (x, y) as shown below.

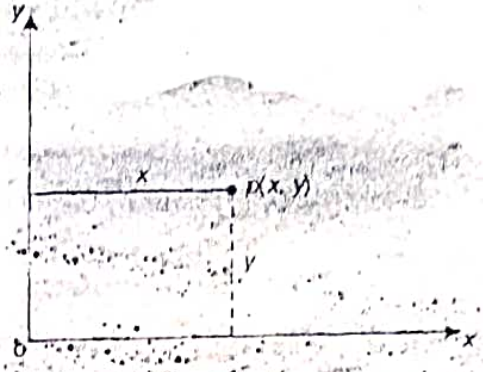


Fig. : A point

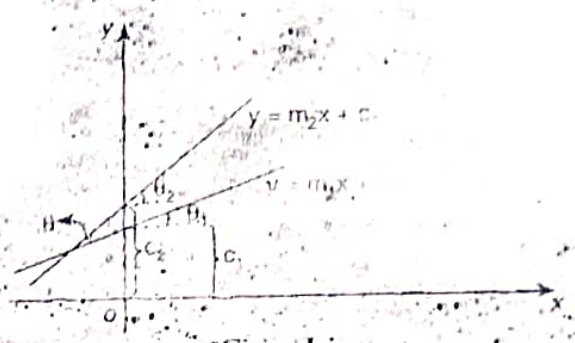


Fig. : Line

Line : *A line can be represented by two points.* Any point which satisfies the equation is on line.

Let there be two lines :

$$y = m_1 x + c_1$$

&

$$y = m_2 x + c_2$$

Let their tangents be :

$$m_1 = \tan\theta$$

and

$$m_2 = \tan\theta_2$$

$$\theta = \theta_1 - \theta_2 \qquad [\because \text{if } m_1 > m_2,\ \theta_1 - \theta_2 \text{ else } \theta_2 - \theta_1]$$

Taking tan on both sides we get :

$$\tan\theta = \tan(\theta_1 - \theta_2)$$

$$\tan\theta = \frac{\tan\theta_1 - \tan\theta_2}{1 + \tan\theta_1 . \tan\theta_2}$$

$$\tan\theta = \frac{m_1 - m_2}{1 + m_1 . m_2}$$

or

$$\theta = \tan^{-1}\left(\frac{m_1 - m_2}{1 + m_1 m_2}\right)$$

Q.1(b) Write short note on viewports and reflection.

Ans.Reflection : A reflection is a transformation that produces a mirror image of an object. In 2D reflection we consider any line in 2D plane as the mirror. The reflection is also described as the rotation by 180°

Viewport : A viewport is an area of the display device on which the window data is presented. A two-dimensional regular viewport is specified by giving the left, right, bottom and top edges of a ractangel. Viewport values may be given in actual physical device coordinates when specified in actual physical device coordinates; they are frequently given using integers. Viewport coordinates may be normalized to some arbitrary range, for example, $0 \le x \le 1.0, 0 \le y \le 1.0$; and specified by floating point numbers.

The display screen is 2-dimensional. Sometimes, it is very difficult to view all the details of an object. The most common views required to represent the object details are : isometric view, orthographic view, front view, top view, left side view, right side view and sectional views. To accommodate the display of several views simultaneously the display screen is divided into viewports. A typical screen layout with four viewports is shown in fig.(1) and (2).

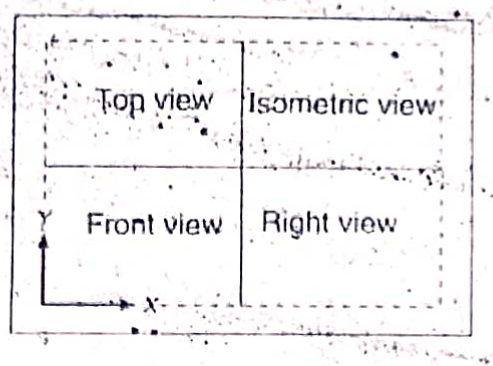


Fig.(1) : A typical screen layout

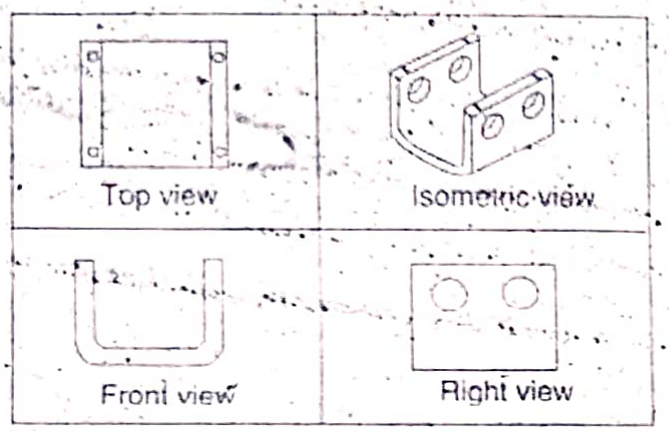


Fig.(2) : Illustration of different views of an object

Q.1(c) What is projection?

Ans. Projection can be defined as a mapping of point $P(x, y, z)$ onto its image $P'(x', y', z')$ in the projection plane or view plane, which consitutes the display surface (see fig (a)). The mapping is determined by a projection line called the projector that passes through P and intersects the view plane. The intersection points is P'.

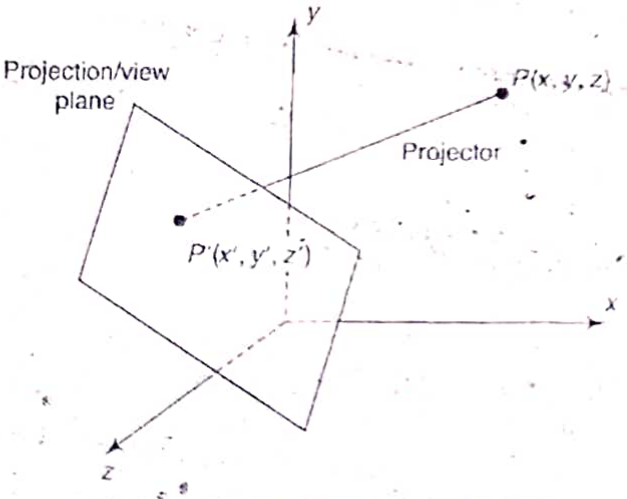


Fig.(a) : The Problem of Projection

Q.1(d) What is image? (5)

Ans. Image is composed of discrete pixels or picture elements. These pixels are arranged in a row and column fashion to from a rectangular picture area. Some times referred to as a raster. Clearly the total number of pixels in an image is a function of the size of the image and the number of pixels per unit length (e.g. inch) in the horizontal as well as the vertical direction. This number of pixels per unit length is referred to as the resolution of the image. Image size is given as the total number of pixels in the horizontal direction times the total number of pixels in the vertical direction.

Images fall into 3 district categories: *Grayscale, Indexed-colour, colour*. Pixal values are a composite number containing intensities of red, green and blue. Image depth is the number of bytes per pixel. An image with a higher image depth has a greater number of possible intensity values, which improves the quality of filtering and measurement.

Section - A

Q.2(a) What is computer graphics? Discuss the applications of computer graphics. (10)

Ans. Computer Graphics : In computer graphics, pictures or graphics objects are presented as a collection of discrete picture elements called pixels (picture element).

Applications of Computer Graphics : Computer Graphics has been widely used, such as graphics presentation, paint systems, computeraided design (CAD), image processing, simulation and virtual reality, entertainment, etc.

1. Computer-Aided Design (CAD) : It is one of the best and biggest application of computer graphics. This is particularly used in engineering applications such as building and other structural design and industrial design, design of manufacturing process, automobiles and aircraft, ships and spacecraft, very large scale-integrated (VLSI) chips, optical systems, and telephone and computer networks.

2. Computer Art : Computer graphics is used to produce picture that express a message and attract attention. In professions involving artist and in other business fields, such as fashion design, architecture, these applications of computer graphics are widely used because of effective speed of computer and easy modifiability capability.

3. Multimedia : Multimedia is the field concerned with the computer-controlled integration of text, graphics, drawings, still and moving images (video) animation, audio and any other media where every type of information can be represented, stored, transmitted and processed digitally.

4. Presentation Graphics : Presentation of data in pictorial form is always preferred since it is more understood than textual data. Presentation of graphics is widely used in posters, brochures, magazines, newspapers, training purpose, etc. This presentation may be sequential still or animated on computer monitor.

5. Internet : The word internet is derived from two words *Interconnection* and *Networks*. Also referred as the Net. Internet is a worldwide system of computer networks, that is, network of networks, which allows the participants to share information on those linked computers.

Internet is a much more broader concept than entertainment. The internet would be positively boring without graphics.

6. Graphical User Interface (GUI) : Graphical User Interface is related to learning and use of packages. It is not possible to learn packages in less times without graphic items

present on the screen. Mostly, software developed today must have a Graphical User Interface and interactivity.

7. Simulation and Virtual Reality : The computer graphics helps here in simulating the views that the operator would see under various circumstances. Complex, mechanical, chemical and industrial processes can be simulated in action so that persons can be trained in their operation before they handle the actual equipment. Such a practice avoids wastages of cost and time.

Simulation has now reached up to such a length that those user follows wearing some special devices on their body such as on eyes, ears, and fingers.

8. Desktop Publishing (DTP) : DTP is related to the production of journals: printing newsletters, book publishing in small organization. This technology has expanded to such an extent that many book publishers, newspapers would not be able to deliver material without DTP.

9. Medical Applications : Computer graphics has now becomes a very important tool of diagnosis and treatment in the hands of doctors, particularly-surgeons. Two-dimensional colorful images of cross-sections of human body or specific organs and limbs are produced. These 2D images a. transformed to special investigated tools: a surgeon can rehearse his operation procedure on the computerized 3D image of the patient's part of body.

Q.2(b) Write and explain Bresenham's line drawing algorithm. (10)

Ans. Bresenham's line algorithm uses only integer addition and subtraction and multiplication by 2, and we know that the compute can perform the operations of integer addition and subtratction very rapidly. The computer is also time-efficient when performing integer multiplication by power 2.

The basic principle of Bresenham's line algorithm is to find the optimum raster locations to represent the straight lines. To accomplish this the algorithm always increments either x or y by one unit depending on the slope of line. Once this is done, then increment in other variable is found on the basis of the distance between the actual line location and the nearest pixel. The distance is called decision variable or the error term.

Bresenham's Algorithm :

(1) Read the line end point (x_1, y_1) and (x_2, y_2) such that they are not equal.

(2) $\Delta x = |x_2 - x_1|$

$\Delta y = |y_2 - y_1|$

(3) Initialize starting point $x = x_1$

$y = y_1$

(4) $S_1 = Sign(x_2 - x_1)$

$S_2 = Sign(y_2 - y_1)$

[Sign function returns −1, 0, 1 depending on whether its agreement < 0, = 0, > 0 respectively].

(5) If $\Delta y > \Delta x$ then

exchange Δx & Δy

ex_change = 1

else

ex_change = 0

(6) $e = 2 * \Delta y - \Delta x$
(7) $i = 1$
(8) plot (x, y)
(9) while $(e \geq 0)$
{
if (ex_change = 1) then
$x = x + s_1$
else
$y = y + s_2$
$e = e - 2 * \Delta x$
}
(10) If ex_change = 1 then
$y = y + s_2$
else
$x = x + s_1$
$e = e + 2 * \Delta y$
(11) $i = i + 1$
(12) if $(i \leq \Delta x)$ then go to step (8)
(13) stop.

Q.3(a) Write and explain mid point circle drawing algorithm. **(10)**

Ans. We consider a circle in second octant from $x = 0$ to $x \leq y$. We use the same concept of eight-way symmetry to plot the entire circle.

Principle : We have 2 pixels –

Upper-pixel $(U) = (x_i + 1, y)$

Lower pixel $(L) = (x_i + 1, y_i - 1)$

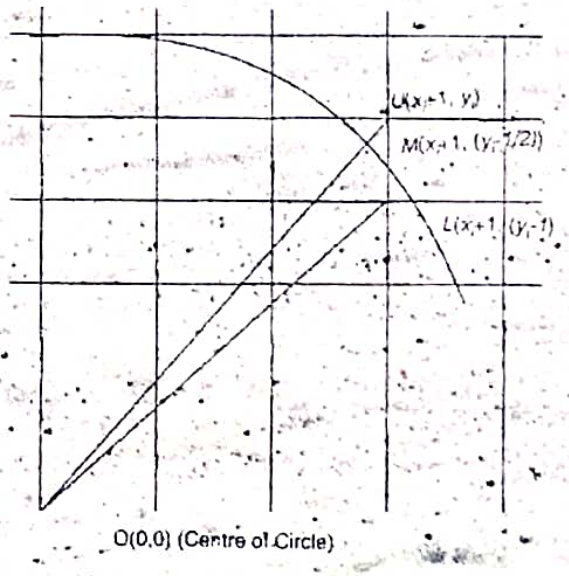


Fig. : Midpoint Circle Method

We need to select, which of these pixels is closer to the true circle and in this algorithm find the mid-point between the two pixels. This mid-point, M is given as follows :

$M(x_{i+1}, (y_i + y_{i-1})/2)$

i.e., the mid point of L and U.

Now, it depends on the Mid-point, which pixel to select. Like,

Mid-point is	Plot
On the circle	Either U or L pixel
Outside the circle	Plot U.
Inside the circle	Plot L.

where U and L are the Upper and Lower pixel respectively.

To apply the midpoint method, we define a circle funcion :

$$F(x, y) = x^2 + y^2 - r^2 = d \qquad ...(1)$$

Any point (x, y) on the boundary of the circle with radius r satisfies the equation $F(x, y) = 0$. Now,

$$d = F(x, y) = x^2 + y^2 - r^2 \begin{cases} = 0 & \text{if point } (x, y) \text{ lies on circle} \\ > 0 & \text{if point } (x, y) \text{ lies outside circle} \\ < 0 & \text{if point } (x, y) \text{ lies inside circle} \end{cases}$$

The initial decision parameter is obtained by evaluating the circle function at the starting pixel position $(x_0, y_0) = (0, r)$. The next mid point lies on $1(1, r - 1/2)$

$$\therefore d = F(1, r - 1/2) = 1^2 + (r - 1/2)^2 - r^2 \qquad ...(2)$$

$$= 5/4 - r$$

Now, for our mid point (M) at

$(x_i + 1, y_i - \frac{1}{2})$, the decision parameter is

$$d_i = f(x_i + 1, y_i - \frac{1}{2})$$

$$= (x_i + 1)^2 + (y_i - \frac{1}{2})^2 - r^2 \qquad ...(3)$$

Here, if d_i is negative, the mid point is inside the circle and we choose the pixel L.

But if d_i is positive (or = 0), the mid-point is outside the circle or on the circle and we choose pixel U.

Similarly, the decision parameter for the next step is :

$$d_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - \frac{1}{2})^2 - r^2 \qquad ...(4)$$

$\because x_{i+1} = x_i + 1$, so we have

From equations ...(3) and (4) we get :

$$d_{i+1} - d_i = [(x_{i+1}) + 1]^2 - (x_i + 1)^2 + \left| y_{i+1} - \frac{1}{2} \right|^2 - \left| y_i - \frac{1}{2} \right|^2$$

$$\therefore d_{i+1} = d_i + 2(x_i + 1) + 1 + (y_{i+1} - y_i^2) - (y_{i+1} - y_i) \qquad ...(5)$$

If pixel L is chosen ($d_i < 0$) then we have

$$y_{i+1} = y_i$$

On the other hand, if pixel U is chosen ($d_i \geq 0$) then we have

$$y_{i+1} = y_i - 1$$

So, we say.

$$d_{i+1} = \begin{cases} d_i + 2(x_i + 1) + 1 & \text{if } d_i < 0 \\ d_i + 2(x_i - 1) + 1 - 2(y_i - 1) & \text{if } d_i \geq 0 \end{cases}$$

Further simplifying these in terms of (x_i, y_i) we get :

$$d_{i+1} = \begin{cases} d_i + 2x_i + 3) & \text{if } d_i < 0 \\ d_i + 2(x_i - y_i) + 5 & \text{if } d_i \geq 0 \end{cases}$$

Or, in terms of (x_{i+1}, y_{i+1}) we get :

$$d_{i+1} = \begin{cases} d_i + 2x_{i+1} + 1 & \text{if } d_i < 0 \\ d_i + 2(x_{i+1} - y_{i+1}) + 1 & \text{if } d_i \geq 0 \end{cases}$$

If radius r is specified as an integer since all increements are integers, then we can simply round d_0 to $(1 - r)$ i.e.

$$d_0 = 1 - r$$

Let us write its algorithm now :

Step 1 : Read (h, k) as the centre of circle and r as its radius.

Step 2 : Initialize $x = 0$
$y = r$
$d = 1 - r$

Step 3 : While $(x \leq y)$

```
// Plot 8-points
plot pixel (x + h, y + k)          // Each point
plot pixel (–x + h, y + k)         // Corresponds to
plot pixel (x + h, –y + k)         // One octant.
plot pixel (–x + h, –y + k)
plot pixel (y + h, x + k)
plot pixel (–y + h, x + k)
plot pixel (y + h, –x + k)
plot pixel (–y + h, –x + k)
 if (d < 0) then
  d = d + 2* x + 3                 // upper pixel
 else
  d = d + 2 * (x – y) + 5          // lower pixel
  y = y – 1
 end if
 x = x + 1
 end while
```

Stop 4 : Stop

Q.3(b) Write and explain Circle drawing algorithm using polar co-ordinates.(10)

Ans. In this method, the point values of a circle are represented by means of trigonometric functions. Let us assume that our circle is at origin and another circle is at (h,k). So 2 cases arise-

Case I. Circle at origin (0, 0) : A circle at origin, with (x, y) as coordinates of point, θ be the current angle and radius r is shown in Fig.(1).

So, $x = r \cos \theta$

$y = r \sin \theta$

Let us develop its algorithm now-

Step 1. Read the radius of circle and angle of increment (say, ϕ).

Step 2. As initially, we are plotting at $(0, r)$ so $\theta = 90°$

Step 3. $x = r \cos \theta$

$y = r \sin \theta$

Step 4. Reflect the point (x, y) in eight ways.

Step 5. $\theta = \theta + \phi$

Step 6. If $\theta < 45°$ then go to step...(3)

Step 7. Stop.

Even this approach is inefficient as sin θ and cos θ calculations are time-consuming.

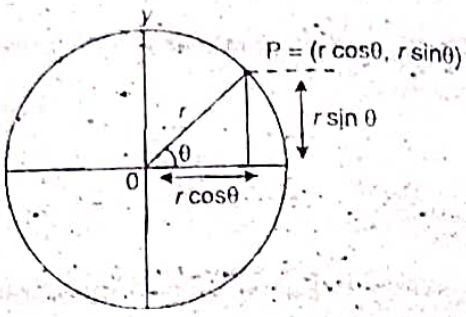


Fig.(1) : Circle at centre (0, 0)

Case II. Centre at (h, k) : As we know that the polar form of a circle is

$x = h + r \cos \theta$ ...(1)

$y = k + r \sin \theta$ ...(2)

where (h, k) is the center of circle

r is the radius

θ is measured in radian 0 to 2π.

Now, we increment the value of θ from 0 to 2π. So, the values of x and y are computed from equation (1) and (2) respectively. This method is time consuming as multiplications and trignometric functions are used. So, we try to improve the process by taking advantage of symmetry in a circle. Also, we know that the arc length are proportional to θ i.e.,

$$Arc(s) = r * \theta \quad ...(3)$$

Fixed increment of θ i.e., $\Delta\theta$ results in equal spacing, Δs between successively plotted points.

Assume now that $\Delta S = 1$ i.e.,

$$\Delta S = 1 \quad ...(4)$$

From equations (3) and (4) we get

$r * \Delta\theta = 1$

or $$\Delta\theta = \frac{1}{r} \quad ...(5)$$

So, we need to compute the first octant only to draw, a circle completely with this method, plos are generated from 0° t0 45°. This is shown in Fig.(2).

Let us develop its algorithm now :

Step 1. Read circle centre as (h, k) and r as its radius.

Step 2. Initialize temp ← 3.14/180

theta inc ← 1/r

theta ← 0

theta end ← 45

Step 3. While (theta < = theta end)
angle = temp * theta
$x = r * \cos(\text{angle})$
$y = r * \sin(\text{angle})$
plot pixel $(x + h, y + k)$ // Plot 8–poinst.
// each for one
// octant.
plot pixel $(-x + h, y + k)$
plot pixed $(x + h, -y + k)$
plot pixed $(-x + h, -y + k)$
plot pixed $(y + h, x + k)$
plot pixed $(-y + h, x + k)$
plot pixed $(y + h, -x + k)$
plot pixed $(-y + h, x + k)$
theta ← theta + thetain c
end while
stop

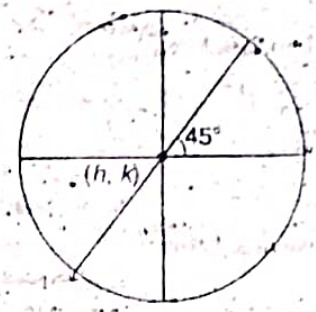


Fig.(2) : Circle at centre (h, k)

Section - B

Q.4[a] Write and explain 4-bit code algorithm for clipping line. [10]

Ans. The line clipping algorithm divides our 2D space into 9 parts (8 outside regions + 1 inside region). This is shown in figure. Each of the nine regions associated with the window is addigned a 4-bit code to identify that region. Each end-point of the line is then assigned a code of the region in which it lies.

The numbers in figure are called as outcodes or region codes. An outcode is computed for each of the two points of a line. For any endpoint (x, y) of a line, the code can be determined that identifies in which region the endpoint lies. The outcode's bit's are set according to the following conditions :

First bit, set to 1 : Point lies to left of window.

Second bit, set to 1 : Point lies to right of window.

Third bit, set to 1 : Point lies below/ bottom of window.

Fourth bit, set to 1 : Point lies above/ top of window.

b_4	b_3	b_2	b_1
T	B	R	L

where

L - Left, R - Right, B - Bottom, T - Top

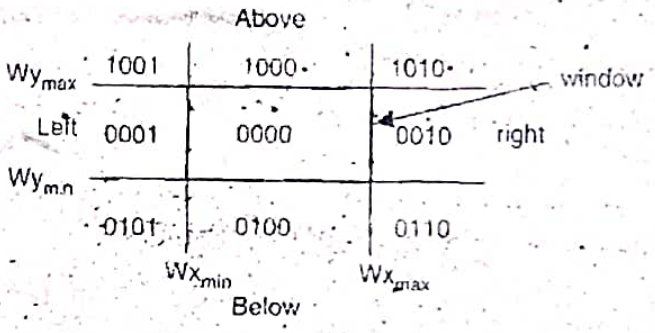


Fig. : Outcodes

But please note that the outcodes/region codes must be recalculated on each iteration after clipping occurs. A point that is in top-right of the viewport means on outcode of 1010 for instance.

Let us write its algorithm now.

Step 1 : Compute the outcodes for the two endpoints of the segment.

Step 2 : Enter a loop, wihtin the loop check to see if both the outcodes are zero. Enter the segment into the display file, exit the loop and return.

Step 3 : If both the outcodes are not zero then take Logical-And of both the outcodes and check for non-zero result. If so, then reject the line, exit the loop and return.

Step 4 : If result is zero, then subdivide the line segment from intersection point of line segment and clipping boundary.

Step 5 : Repeat steps (2), (3) and (4) for each segment.

Q.4(b) Write and explain Sutherland-Cohen algorithm. **(10)**

Ans. Sutherland-Cohen algorithm : The Cohen Sutherland line clipping algorithm quickly detects and dispenses with two common and trivial cases (visible and non visible). In other words, this algorithm performs initial tests on a line to determine whether intersection calculations can be avoided.

The Cohen Sutherland algorithm uses a divide and conquer strategy. The line segment's end-points are tested to see if the line can with be trivially accepted or rejected. If the line cannot be trivially accepted or rejected, an intersection of the line with a window edge is determined and the trivial reject/accept test is repeated. This process is continued until the line is accepted. To perform the trivial acceptance and rejection tests, we extend the edges of the window to divide the plane of the window into the nine regions-the eight "outside" regions and the one "inside" region. Each of the nine regions associated with the window is assigned a 4- bit code to identify the region. Each end point of the line segment is then assigned th code of the region in which it lies.

Algorithm of Cohen Sutherland :

Step 1. Calcuate positions of both end points of line.

Step 2. Perform OR operation on both of these end -pionts

Step 3. If OR operation gives 0000

then

line is considered to be visible

else

perform AND operation on both end points

If And ≠ 0000

then line is invsible

else

And = 0000

line is considered clipped case.

Step 4. If line is clipped case, find intersection with boundries of window

$m = (y_2 - y_1)/(x_2 - x_1)$

(a) If bit 1 is "1" line intersect with lift boundry of rectangle window

$y_3 = y_1 + m(x - X_1)$

where $X = X_{wmin}$

where x_{wmin} is min value of X co-ordinate of window

(b) If bit 2 is "1 line intersect with right boundry $y_3 = y_1 + m(X - X_1)$

where $X = X_{wmax}$

where X more is maximum value of X co-ordinate of window.

(c) If bit 3 is "1" line intersect with bottom boundry

$X_3 = X_1 + (y - y_1)/m$

where $y = y_{wmin}$

y_{wmin} is minimum value of Y co-ordinate of window.

(d) If bit 4 is "1" line intersect with top boundry

$X_3 = X_1 + (y - y_1)/m$

where $y = y_{wmax}$

y_{wmax} is maximum value of Y co-ordinate of window.

Step 5. Repeat step 1 to 3 until line is completely accepted or rejected.

Q.5(a) Explain two dimension transformation discuss about tranformation, translation, scaling and rotation. **(10)**

Ans. Two dimension transformation : This process of changing of sizes, orientations or positions of objects by a mathematical operation is called as transformation. This change is accomplished by two methods:

(a) Geometric transformation

(b) Coordinate transformation

(a) Geometric Transformation : It is a type of transformation in which an object itself is transformed (changed) relative to stationary coordinate system. It is applied to each point of the object. This is shown in fig.(1).

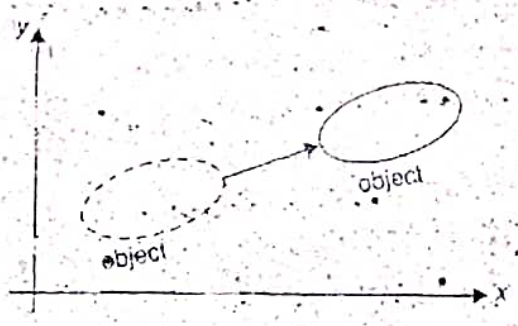


Fig.(1) : Geometric Translation

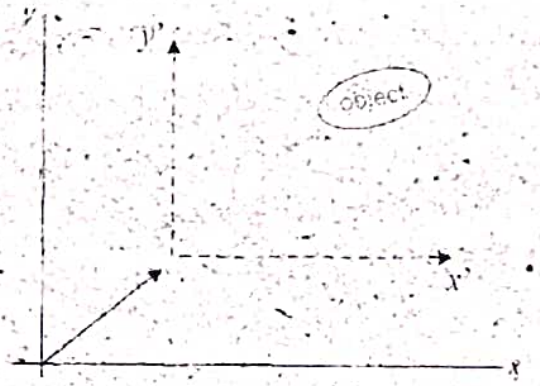


Fig.(2) : Coordinate Translation

(b) Coordinate Transformation : It is a type of transformation in which the object is held stationary while the coordinate system is transformed relative to the object. This is shown in fig.(2).

Translation : Translation consists of a shift of the object parallel to itself in any direction in the (x, y) plane. Any such shift can be accomplished by a shift in x-direction plus-a shift in y-direction. If the amount of x-shaft is called t_x and the amount of y-shaft t_y, the translation of the point (x, y) into the point (x', y') is expressed by the formulas.

$$x' = x + t_x$$

$$y' = y + t_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

To translate an object with multiple points, we just translate each point individually and connect them together.

Translation is a rigid-body transformation that moves objects without deformation. That is, every point on the object is translated by the same amount. Translatoin is the only transformation that is not in relation to a reference point. Its effect is independent of the original position of the object.

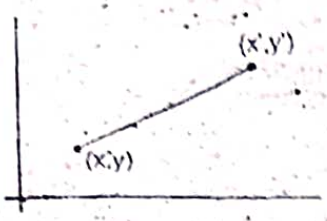


Fig. : (a)

Scaling : Scaling is a transformation that changes the size or shape of an object. Scaling with respect to origin can be carried out by multiplying the co-ordinate values (x, y) of each vertex of a polygon, or each endpoint of a line by scaling factors S_x and S_y respectively to produce the coordinates (x', y').

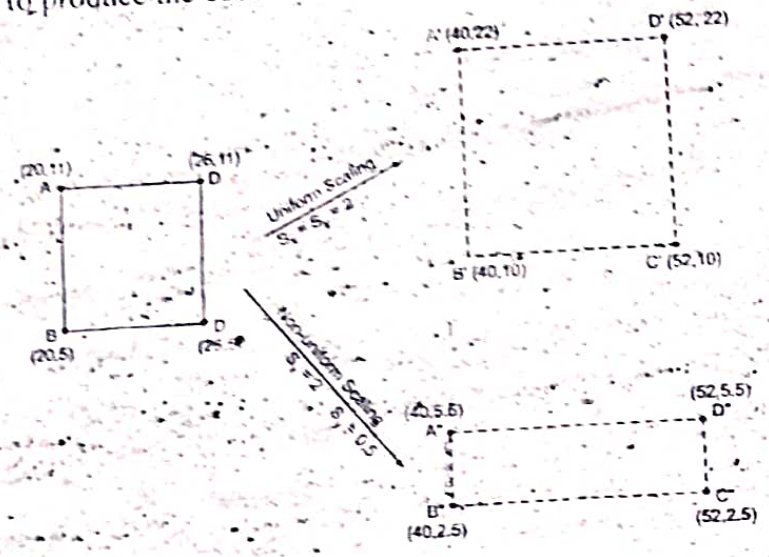


Fig. : (b)

The mathematical expression for pure scaling is

$$x' = S_x * x$$

$$y' = S_y * y$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

or symbolically $[\chi'] = [T]\,[X]$

After applying scaling factor or matrix on any object, coordinate is either increased or decreased depending on the value of scaling factors S_x and S_y. If it is greater than one, then enlargement of the object will occur and if it is less than one compression will occur.

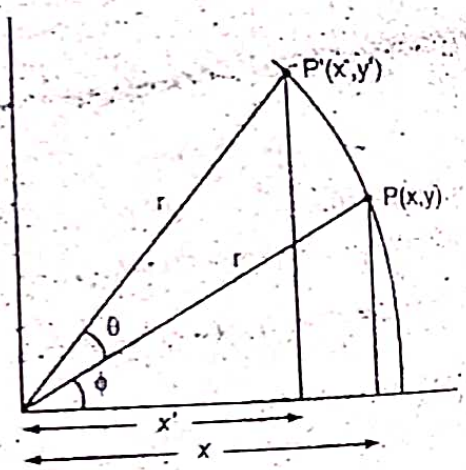


Fig. : (c)

Rotation : Refer fig.(c). A two-dimensional rotation is applied to an object by repositioning it along a circular path in the xy plane. Point can be rotated through an angle θ about the origin. The sign of angle determines the direction of rotation. Positive values for the rotation angle defines counterclockwise rotations and negative values rotate objects in clockwise direction.

Q.5(b) Explain Reflection and composite Transformation. (10)

Ans. Reflection : A reflection is a transformation that produces a mirror image of an object. In 2D reflection we consider any line in 2D plane as the mirror. The reflection is also described as the rotation by 180°

(a) *Reflection about x-axis* : The basic principles behind reflection transformation are :

(i) The image of an object is formed on the side opposite to where the object is lying w.r.t. mirror line.

(ii) The perpendicular distance of the object from the mirror line is same to the distance of the reflected image.

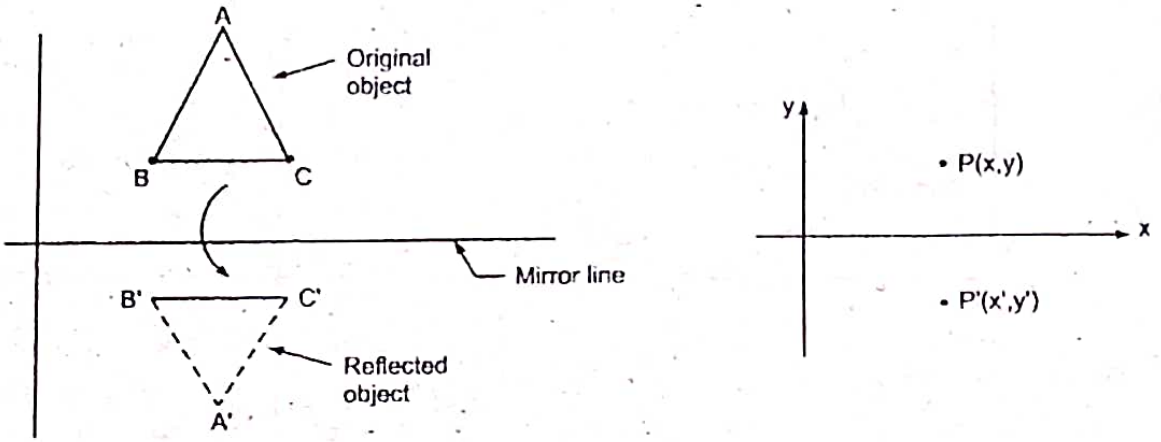


Fig. : Reflection about x-axis

For reflection about x-axis, x co-ordinate is not changed and sign of y-coordinate is changed. Thus if we reflect point (x, y) in the x-axis, we get $(x, -y)$

i.e., $x' = x$

$y' = -y$

So, the transformation matrix for reflection about x-axis or $y = 0$ axis is,

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

and the transformation is represented as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

(b) *Reflection about Y-axis* : A reflection about Y-axis ($x = 0$) flips x-coordinates while y-coordinates remains the same. For reflection of P(x, y) to $P'(x', y')$

$x' = -x$

$y' = y$

This transformation is identified by the reflection transformation matrix as

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

(c) *Reflection about the Origin* : In this case, we actually choose the mirror line as the axis perpendicular to the xy plane and passing through the origin. After reflection both the x y coordinate of the object point is flipped i.e., x' becomes $-x$ and y' becomes $-y$.

i.e., $x' = -x$

$y' = -y$

This can be represented in matrix form as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Thus the transformatin matrix for reflection about origin is

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$$

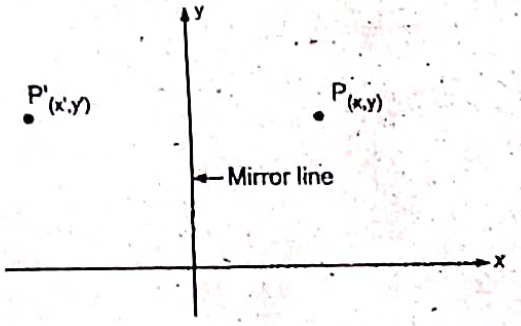


Fig. : Reflection about Y-axis

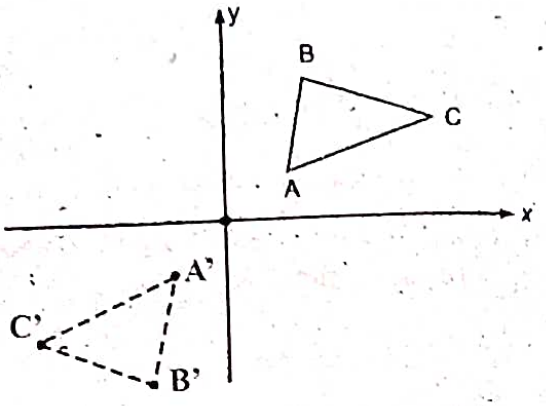


Fig. : Reflection about Origin

Composite Transformations : To control the size of an object, we need to perform matrix operations on the position vector which defines the vertices. It is not necessary that we get the required orientation by applying single tranformation, it may require more than one transformation. Since matrix multiplication is non commutative, the order of application of the transformation is important.

We can set up a matrix for any sequence of transformations as a composite transformation matrix by calculating the matrix product of the individual transforamtions.

(i) If two successive translation are applied then

$$\begin{pmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_{x1} + t_{x2} \\ 0 & 1 & t_{y1} + t_{y2} \\ 0 & 0 & 1 \end{pmatrix}$$

(ii) If two successive rotations θ_1 and θ_2 then

$$\begin{pmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & 0 \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(iii) If two successive scaling are applies then

$$\begin{pmatrix} S_{x1}.S_{x2} & 0 & 0 \\ 0 & S_{y1}.S_{y2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Section - C

Q.6(a) What is projection and type of projections ? (10)

Ans. Projection can be defined as a mapping of point $P(x, y, z)$ onto its image $P'(x', y', z')$ in the projection plane or view plane, which consitutes the display surface (see fig (a)). The mapping is determined by a projection line called the projector that passes through P and intersects the view plane. The intersection points is P'.

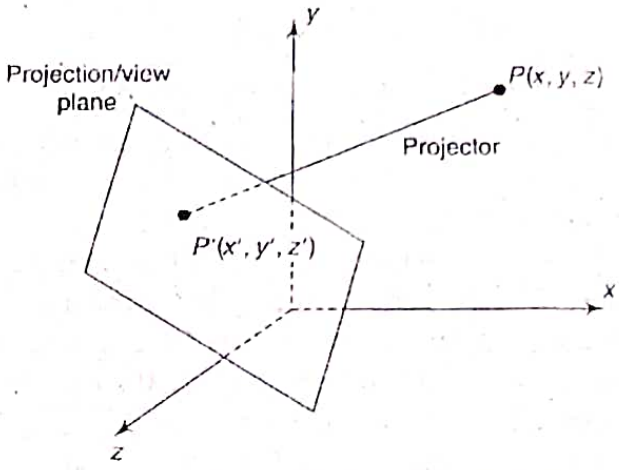


Fig.(a) : The Problem of Projection

Parallel Projection : Parallel projection is a type of projection in which the centre of projection is situated at infinite distance such that the projectors (lines) are parallel to each other. The plane on which we are taking projection is called as a view plane. And a parallel projection is formed by extending parallel lines from each vertex of the object until they intersect this viewplane. The point of intersection is the projection of the vertex. We can then connect the projected vertices by line segments which corresponds to the connections on the original object. Our objective is to represent a 3D object onto a 2D plane like our mointor and the simplest way is to discard the z-coordinates.

Oblique Parallel Projection : A projection in which the angle between the projectors and the plane of projection is not equal to 90° is known as an oblique projection.

There are two common types of oblique parallel projections :

(a) Cavalier Parallel Projection

(b) Cabinet projection.

Orthographic Parallel Projection : A type of parallel projection in which centre of projection lies at infinity and the projectors are perpendicular to the view plane is known as orthographic parallel projection.

There are two basic types of orthographic parallel projections :

(a) Multiview orthographic parallel projection

(b) Axonometric orthographic parallel projection

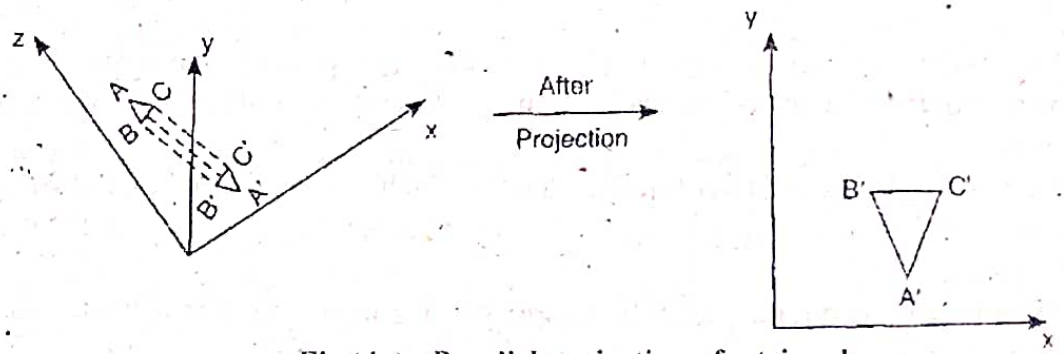


Fig.(b) : Parallel projection of a triangle

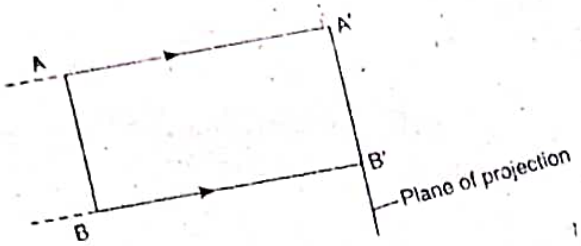


Fig.(c) : Oblique projection

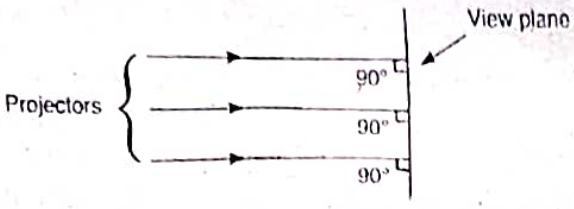


Fig.(d) : An orthographic projection

Perspective Projection : In this type of projection, that object is away (very far) from the viewer and appears to be smaller. This provides the viewer which a depth cue, an indication of which portions of the image correspond to parts of the objects which are close or far away. Please note here that the farther the object is from the viewer, the smaller it appears. To obtain a perspective projection of a 3D-object, we transform points along projection lines which are not parallel to each other and converge to meet at a finite point known as the projection reference point or the center of projection. Also note that in real world, the centre of projection is a human eye. The projected view is obtained by calculating the intersection of the projection lines with the view plane. This is shown in fig. (e).

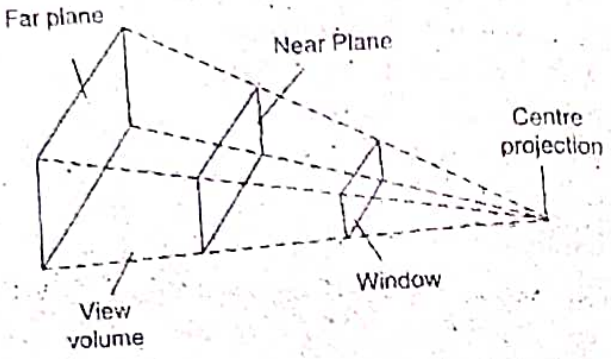


Fig.(e) : Perspective Projection

Perspective projections are of three types :

(a) 1 - point perspective projection
(b) 2 - point perspective projection
(c) 3 - point perspective projection

Q.6(b) Explain the coordinate system. (10)

Ans. Coordinate systems : The various coordinates used in the viewing transformations are as follows :

(1) World coordinate System (WCS)
(2) Viewing Coordinate System (VCS)
(3) Normalized Coordinate System (NCS)
(4) Device Coordinate System (DCS)

- The WCS is the right-handed Cartesian system in which the picture to be displayed is described.

- VCS is defined as a Cartesian coordinate system with the origin having world coordinates (u_0, v_0). The axis of the viewing coordinate system makes an angle of θ with the worls coordinate system x-y is.

- NCS is a right handed coordinate system in which the display area of the virtual display device corresponds to the unit (1 × 1) square whose lower left hand corner is at the origin of the coordinate system.

- DCS is the coordinate system that corresponds to the device or workstation where the picture is to be displayed.

Q.7(a) Write and explain scan line algorithm. (10)

Ans. A scan-line algorithm locates the intersection points of the scanline with each edge of the area to be filled. We proceed from left to right, pair all intersections and the intervening pixels are set to the specified fill intensity or color. A scanline that intersects a polygon vertex may produce an even number of intersections or an odd number of intersections. Even number of intersections can be paired to correctly identify interior pixel but odd number of intersections do not correspond to the polygon interior.

Consider the following polygon with two scanlines, y and y', in fig.

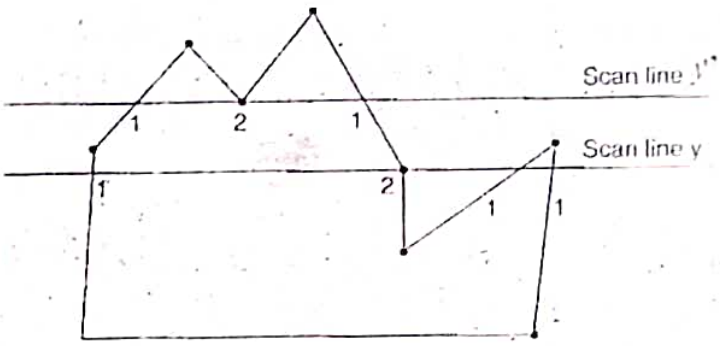


Fig. : Two scan-lines cutting polygon edges.

Here, scanline y intersects 5 polygon edges whereas scanline y' intersects 4 polygon edges. But there is a difference. For scanline y, the two intersecting edges sharing a vertex are an opposite side of the scanline but for scanline y', the two intersecting edges are both above the scanline. In scanline y, we need to do some additional processing i.e., the vertices that require additional processing are those that have connecting edges in opposite side of the scanline. If the two intersecting edges sharing a vertex are an opposite side of scanline than we store only one intersection point for that vertex else we store two intersection points. Please note that it is necessary to calculate x intersection points for scan line with every polygon side. We can simplify these calculations by using coherence property. It is defined as a property of a scene by which we can relate one part of the scene with the other parts of a scene. Here, we can use a slope of an edge as a coherence property.

By using this property, we can find out the x-intersection value on the lower scan line if the x intersection value for current scan line is known. i.e.

$$x_{i+1} = x_i - \frac{1}{m}$$ (where m is slope of edge)

As we scan from top to bottom, value of y coordinates between the two scanlie changes by 1 i.e., $y_{i-1} = y_i - 1$

Many times it is not necessary to compute the x-intersections for scanline with every polygon side. It is easier to identify which polygon sides should be tested for x-intesection if we first sort the sides in order of their maximum y value. Once the sides are sorted we can process the scanlines from the polygon top its bottom producing active edge list (or simply active edges) for each scanline crossing the polygon boundaries. This active edge list for a scanline contains all edges crossed by that scanline. Its algorithm is as follows :

Step 1 : Sort the polygon sides on the largest y-value.

Step 2 : Start with the largest y-value and scan down the polygon.

Step 3 : For each y, determine which sides can be intersected and find the x values of these intersection point.

Step 4 : Sort, pair and pass the x-values to the line drawing routine.

Let us write a complete scanline conversion algorithm for polygon filling now :

(1) Read the number of vertices of polygon, n.

(2) Read the x and y coordinates of all vertices in array $x[n]$ and $y[n]$.

(3) Find the minimum and maximum value of y and store as y_{min} and y_{max}.

(4) Store the initial x-value (x_1 and x_2). y-values y_1 and y_2 for two endpoints and x-increment by $\frac{1}{m}$ from scanline to line for each edge in the array edges. For each edge check that $y_1 > y_2$, if not the ninterchange (x_1, y_1) and (x_2, y_2) so that for each edge, y_1 represents its maximum y-coordinate and y_2 represents its minimum y-coordinate.

(5) Sort array edges in descending order of y_1.

(6) Set $y = y_{max}$.

(7) Find the active edges and update active edge list :

If ($y >= y_1$ and $y_1 < y_2$)

{edge is active}

Else

{edge is not active}

(8) Compute the x-intersects for all active edges for current y-value and x-intersects for successive y-values can be given as :

$$x_{new} = x_{old} - \frac{1}{m} \qquad \left[m = \frac{y_2 - y_1}{x_2 - x_1} \right]$$

(9) Sort the x-intersections in ascending order. If x-intersect is vertex i.e., x-intersect = x_1, and $y = y_1$ then apply vertex test to check whether to consider on intersect or two intersects.

(10) Extract pairs of intersects from the sorted x-intersection.

(11) Pass pairs of x-values to the line drawing routine to draw corresponding line segements.

(12) Set $y = y - 1$

(13) Repeat steps 7 to 12 until $y \geq y_{min}$

(14) Stop.

Q.7(b) Write and explain Mid-point sub-division algorithm ? **(10)**

Ans. Mid-point Subdivision Algorithm : The only difficult part in Cohen-Southerland algorithm is to find the intersection point of line with window boundary. An alternative to finding intersection points by equation solving is based on bisection method of numerical analysis. The line segment is divided at its midpoint into two smaller line segments. The clipping categories of the two new line segments are then determined. Each segment in category 3 is divided again into smaller segments and categorized. The bisection and categorization process continues until all segments are in category 1 (visible) or category 2 (invisible). Th mid-point co-ordinates (x_m, y_m) of a line segment joining $P_1(x_1, y_1)$ to $P_2(x_2, y_2)$ are given by

$$x_m = \frac{x_1 + x_2}{2} \; y_m = \frac{y_1 + y_2}{2}$$

We can call mid-point subdivision algorithm as a special case of Cohen-Sutherland algorithm because it used the same technique as Cohen-sutherland. It uses the line endpoint outcodes and associated tests to immediatedly identify which lines are visible and which are invisible. If both the endpoints of a line has outcodes 0000 then the lines is totally visible. If both

the outcodes are not zero then we have to take logical AND of both outcodes and then we have to decide whether that line is visible or not. If the logical AND result is nonzero, it means the line is totally invisible and if the logical AND is zero, it means a line may be partially visible.

The algorithm is formalized in the following steps:

For each endpoint :

(a) If the endpoint is visible, then it is farthest visible point. The process is complete. If not, continue.

(b) If the line is trivially invisible, no output is generated. The process is complete. If not, continue.

(c) Divdie the line P_1P_2 at its midpoint, P_m. Apply the previous tests to the two segments P_1P_m and P_mP_2. If P_mP_2 is rejected as trivally invisible, the midpoint is an overestimation of the farthest visible point. Continue with P_1P_m. otherwise the midpoint is an underestimation of the farthese visible point. Continue with P_2P_m. If the segment becomes so short that the midpoint corresponds to the accuracy of the machine or, as specified to the end points, evaluate the visibility of the point and the process is complete.

Section - D

Q.8 Discuss about parametric representation of surface also explain the interpolation method. **(20)**

Ans. Parametric representation for surfaces:

- Surfaces can be represented parametrically as

$$p(u,v)=\begin{cases} x=f_x(u,v) \\ y=f_y(u,v) \\ z=f_z(u,v) \end{cases}$$

- example: sphere of radius r centered at origin

$$\begin{cases} x=r\cos(u)\cos(v) \\ y=r\sin(u)\cos(v) \\ z=r\sin(v) \end{cases}$$

Linear interpolation, in computer graphics often called "LERP" (Linear interpolation), is a very (if not THE simplest) method of interpolation.

For a set of discrete values linear interpolation can approximate other values inbetween, assuming a linear development between these discrete values. An interpolated value, calculated with linear interpolation, is calculated only in respect to the two surrounding values, which makes it a quite inappropriate choice if the desired curve should be smooth. If a curvier interpolation is needed, Splines might be an option.

To calculate a value between two other values, using linear interpolation, a certain factor, often called "step", must be used. The value has to be between value 1 (v_0) and value 2 (v_1). If "step" is 0.0 the interpolated value to v_0 and if step is 1.0, it's equal to v_1. So the formula is as follows.

$$v_0*(1.0-step)+v_1*step$$

or (computationally a more efficient way):

$$v_0(v_1-v_0)*step$$

Alternative

Given two points $(x_0, y_0), (x_1, y_1)$ on a graph. A value (x_i, y_i) can be calculated like this.

$$y_i = y_0 * \left(1.0 - \frac{x_i - x_0}{x_1 - x_0}\right) + y_1 * \frac{x_i - x_0}{x_1 - x_0}$$

or this :

$$y_i = y_0 + (y_1 - y_0)\frac{x_i - x_0}{x_1 - x_0}$$

where $\frac{x_i - x_0}{x_1 - x_0}$ is the step-value.

Q.9(a) What is shadow and transparency ? (10)

Ans. Transparency : A transparent surface, in gereral, produces both reflected and transmitted light. It has transparency coefficient T as well as values for reflectivity and specular reflection. The coefficient of transparency depends on the thickness of the object because the transmission of light depends exponentially on the distance which the light ray must travel within the object. The expression for coefficient of transparency is given as

$$T = t\, e^{-ad}$$

Where t is the coefficient of property of material which determines how much of the light is transmitted at the surface instead of reflected, a is the coefficient of property of material which tells how quickly the material absorbs or attenuates the light, d is the distance the light must trav in the object.

When light crosses the boundary between two media it changes the direction as shown in the Fig.(1). This effect is called refraction. The effect of refraction is observed because the speed of light is different in different materials resulting different path for refracted light from that of incident light. The direction of the refracted light is specified by the angle of refraction (θ_r). It is the function of the property material called the index of refraction (η). The angle of refraction θ_r is calculated from the angle of incidence θ_i, the index ofrefraction η_i of the incident material (usually air) and the index of refraction η_r of the refracting material according to Snell's Law.

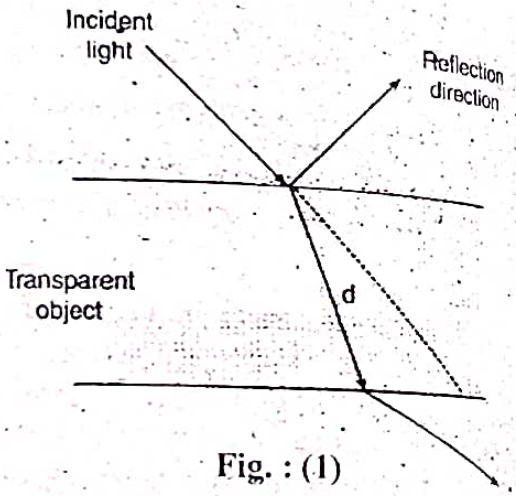


Fig. : (1)

$$\sin\theta_r = \frac{\eta_i}{\eta_r}\sin\theta_i$$

Shadow : A shadowed object is one where light is blocked from reaching i.e., the object which is hidden with respect ot the light source. However, shadows are seen only when observer's position is not coincides with the light source. The shadowed areas are not visible from the light source position but are visible from observer's position. Such areas can be identified by applying the hidden surface detection methods with viewpoint assumed at the light source position.

A shadowed object is one which is hidden from the light source. It is possible to use hidden surface algorithms to locate the areas where light sources produce shadow. In order to achieve this we have to repeat the hidden-surface calculation using light source as the viewpoint.

This calculation divides the surfaces into shadowed and unshadowed groups. The surfaces that are visible from the light source are not in shadow; those that are not visible from the light source are in shadow. Surfaces which are visible and which are also visible from the light source are shown with both the background illumination and the light-source illumination. Surfaces which are visible but which are hidden from the light source are displayed with only the background illumination, as shown in the Fig.(2).

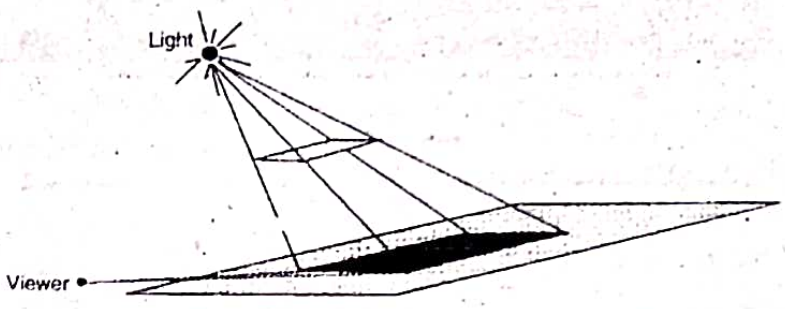


Fig. : (2)

There are two types of shadows : *shelf shadow* and *projected shadow*. Shelf-shadows are created on some of the object planes when the object itself occuldes light from illuminating those planes. A projected shadow results on a surface when an intervening object blocks light from reaching the surface.

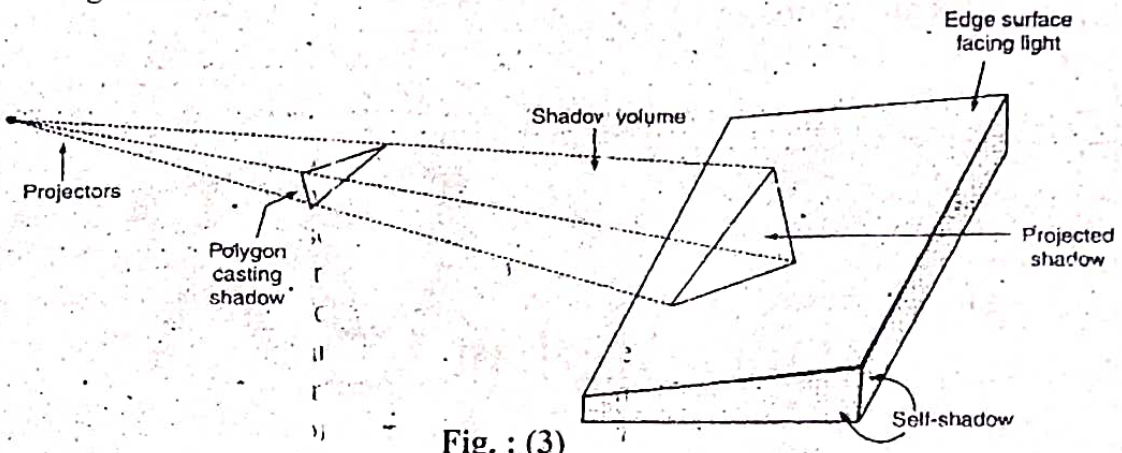


Fig. : (3)

Q.9(b) Differnetiate between image processing and computer graphics? Explain the components of an image processing system in detail. (10)

Ans. Difference between Image processing and Computer Graphics : In computer graphics and image processing overlap, two areas are conerned with fundamentally different operations. In computer graphics, a computer is used to create a picture. Image processing on the other hand applies techniques to modify or interpret existing pictures such as photographs.

Components of an image Processing System : Image Processing System consists of four essential components :

(1) An image acquisition system. In the simplest case, this could be a CCD camera, a flatbed scanner, or a video recorder.

(2) A device known as a frame grabber to convert the electrical signal (normally an analog video signal) of the image acquisition system into a digital image that can be stored.

(3) A personal computer or a workstation that provides the processing power.

(4) Image processing software that provides the tools to manipulate and analyze the images.

COMPUTER GRAPHICS

Dec - 2014

Paper Code:-CSE-303-F

Note : *Attempt five questions in all, selecting one question from each Section. Question No. 1 is compulsory. All questions carry equal marks.*

Q.1.(a) Explain the following : (20)

(a) Random scan vs raster scan.

(b) 2D transformation.

(c) Types of projections.

(d) Diffuse illumination and Diffuse of reflection :

Ans. (a)Random scan vs raster scan : Difference between Vector scan display and Raster Scan Display :

Vector(random) scan display	Raster Scan Display
(i) Vector display only draws lines and characters.	(i) Raster display has ability to display areas fillled with solid colours or patterns.
(ii) Don't use interlacing.	(ii) Uses interlacing.
(iii) In vector scan display the beam is moved between the end points of the graphics primitives.	(iii) In raster scan display the beam is moved all over the screen one scan line at a time from top to bottom and then back to top.
(iv) Higher resolution.	(iv) Lower resolution.
(v) More expensive.	(v) Less expensive.
(vi) Uses monochrome or beam-penetration type.	(vi) Uses monochrome or shadow mask type.
(vii) Vector display draws a continuous and smooth lines.	(vii) Raster display can display mathematically smooth lines polygons and boundaries of curved primitives only by approximating them with pixel on the raster grid.
(viii) Editing is easy.	(viii) Editing is difficult.
(ix) Refresh rate depends directly on picture complexity.	(ix) Refresh rate independent of picture complexity.
(x) Scan conversion is not required.	(x) Graphics primitives are specifies in term of their end points and must be scan converted into their corresponding pixel in the frame buffer.

Ans. (b)2D transformation : This process of changing of sizes, orientations or positions of objects by a mathematical operation is called as transformation. This change is accomplished by two methods:

(1) Geometric Transformation : It is a type of transformation in which an object itself is transformed (changed) relative to stationary coordinate system. It is applied to each point of the object. This is shown in fig.(1).

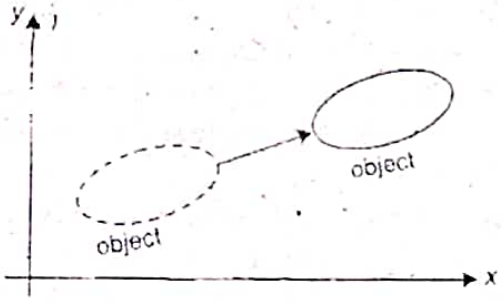


Fig.(1) : Geometric Translation

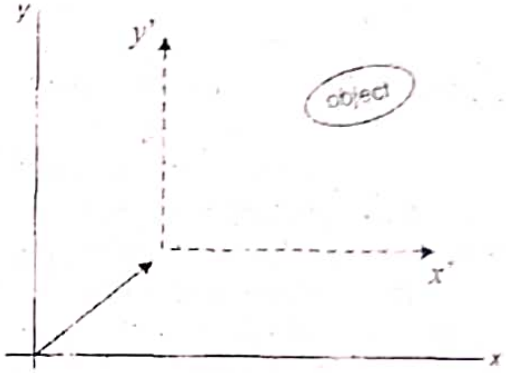


Fig.(2) : Coordinate Translation

(2) Coordinate Transformation : It is a type of transformation in which the object is held stationary while the coordinate system is transformed relative to the object. This is shown in fig.(2).

Ans. (c)Types of projections :

Parallel Projection : Parallel projection is a type of projection in which the centre of projection is situated at infinite distance such that the projectors (lines) are parallel to each other. The plane on which we are taking projection is called as a view plane. And a parallel projection is formed by extending parallel lines from each vertex of the object until they interesect this viewplane.

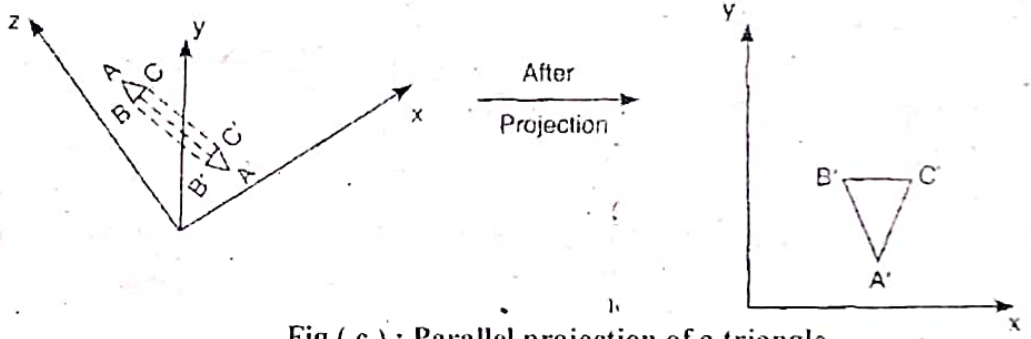


Fig.(c) : Parallel projection of a triangle

Perspective Projection : In this type of projection. that object is away (very far) from the viewer and appears to be smaller. This provides the viewer which a depth one, an indication of which portions of the image correspond to parts of the objects which are close or far away.

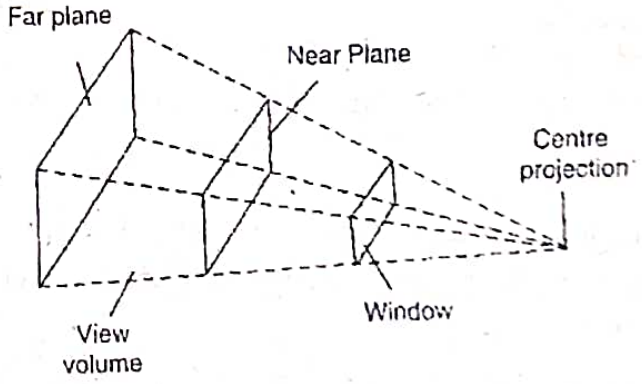


Fig.(f) : Perspective Projection

Ans. (d)Diffuse illumination and Diffuse of reflection : We will assume that our object is illuminated by light which does not come from any particular source but which comes from all directions, *i.e.*, ambient light. When such illumination is uniform from all directions, the illumination is called *diffuse illumination*. Actually, *diffuse illumination is a background light which is reflected from walls, floor and ceiling*. We will assume that there is as much light going up as there is going down and that there is the same amount going right as there is going left. When we assume that going up, down, right and left is of same amount then we can say that the reflection are constant over each surface of the object and they are independent of the viewing direction. Such a reflection is called diffuse reflection.

In practice, when object is illuminated, some part of light energy is absorbed by the surface of the object, while the rest is reflected. The ratio of the light reflected from the surface to the total incoming light to the surface is called coefficient of reflection or the reflectivity. It is donated by R. The value of R varies from 0 to 1. It is closer to 1 for white surface and closer to 0 for black surface. This is because white surface reflects nearly all incident light whereas black surface absorbs most of the incident light. Reflection coefficient for gray shades is in between 0 to 1. In case of colour object reflection coefficient are various for different colour surfaces.

Section-A

Q.2.(*a*) Write the step rquired to plot a line whose slope is between 10° and 60° using the slop intercept equation. (10)

Ans. Steps Required :

(i) Compute dx : $dx = x_2 - x_1$

(ii) Compute dy : $dy = y_2 - y_1$

(iii) Compute m : $m = dy/dx$

(iv) Compute b : $b = y_1 - m \times x_1$

(v) Set (x, y) equal to lower left-hand end point and set x_{end} equal to the largest value of x. If $dx < 0$, then $x = x_2$, $y = y_2$ and $x_{end} = x_1$. If $dx > 0$, then $x = x_1$, $y = y_1$, and $x_{end} = x_2$.

(vi) Test to determine whether the entire line has been drawn.

If $x > x_{end}$, stop.

(vii) Plot a point at current (x, y) coordinates

(viii) Increment x : $x = x_1 + 1$

(ix) Compute the next value of y from equation $y = mx + b$

(x) Go to step 6.

Q.2.(*b*) Indicate which raster location would be chosen by Bresenham's algorithm when scan converting a line from pixel coordinate (5,5) to pixel coordinate (8, 9). (10)

Ans. (5, 5) → (8, 9)

$$dx = x_2 - x_1 = 8 - 5 = 3$$
$$dy = y_2 - y_1 = 9 - 5 = 4$$

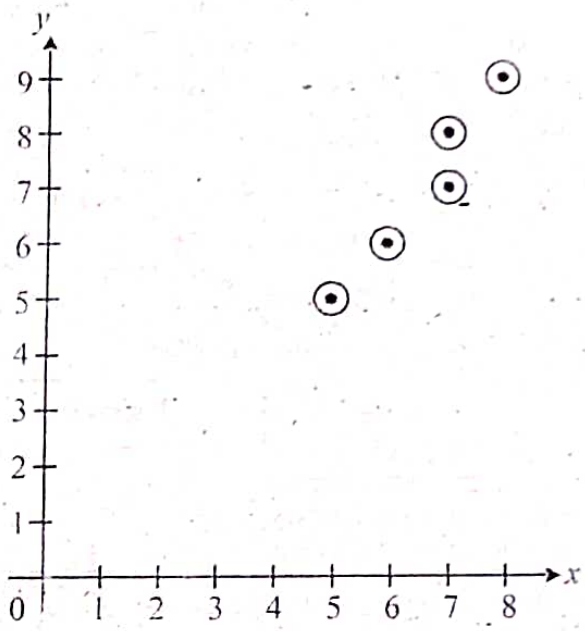
$$Inc_1 = 2dy = 2 \times 4 = 8$$
$$Inc_2 = 2\,(dy - dx) = 2\,(4 - 3) = 2$$
$$d = Inc_1 - dx = 8 - 3 = 5$$

Here m > 1

So here we swep dx by dy i.e. x with y.

Hence point will be

x	y
5	5
6	6
7	7
7	8
8	9



Q.3 Explain the architecture of Raster Scan Display. Give the logical organization of video controller and explain its importance in Raster Scan display. (20)

Ans. Raster Scan Display : A raster scan display is based on intensity control of pixels in form of a rectangular box, called raster, on the screen. A raster scan display is shown in fig.(a). In this beam will move across the screen, one row at a time. The direction of movement is top to bottom. It is used in televisions. When beam move from top to bottom beam will be *on* or *off* so pattern of spots in created. Information of on and off pixels is stored in refresh buffer or frame buffer. Television is our house are based on Raster Scan method.

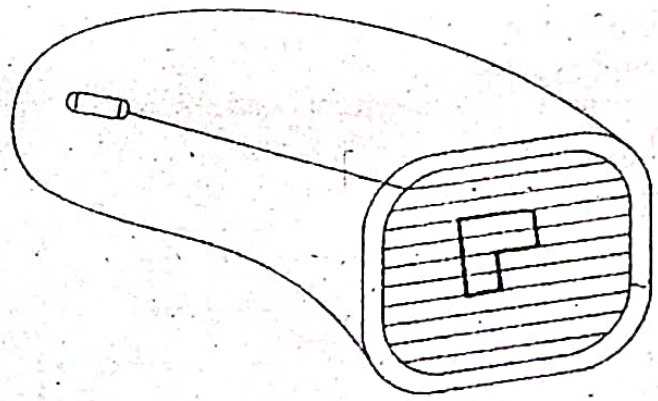


Fig.(a) : Raster Scan Display

The raster scan system can store information of each pixel position so it is suitable for realistic display of objects. Good shading effect and coloring patterns can be displayed using raster scan. For black and white computer each pixel is represented by *on* or *off*. So *zero* or *one*

is used to represent each pixel. A *one* indicates electron beam is turned on at this point. A zero indicates that the beam is turned off at this point. For color monitors more bits are needed. For highest quality systems 24 bits per pixel are used. Raster scan provide refresh rate of 60 to 80 frames per second.

Architecture of Raster Scan Display : The architecture of a raster display, it consists of display controller, CPU, video controller, refresh buffer, keyboard, mouse and the CRT.

As shown in the fig. (b) , the display image is stored in the from of 1's and 0's in the refresh buffer. The video controller reads this refresh buffer and produces the actual image on the screen.

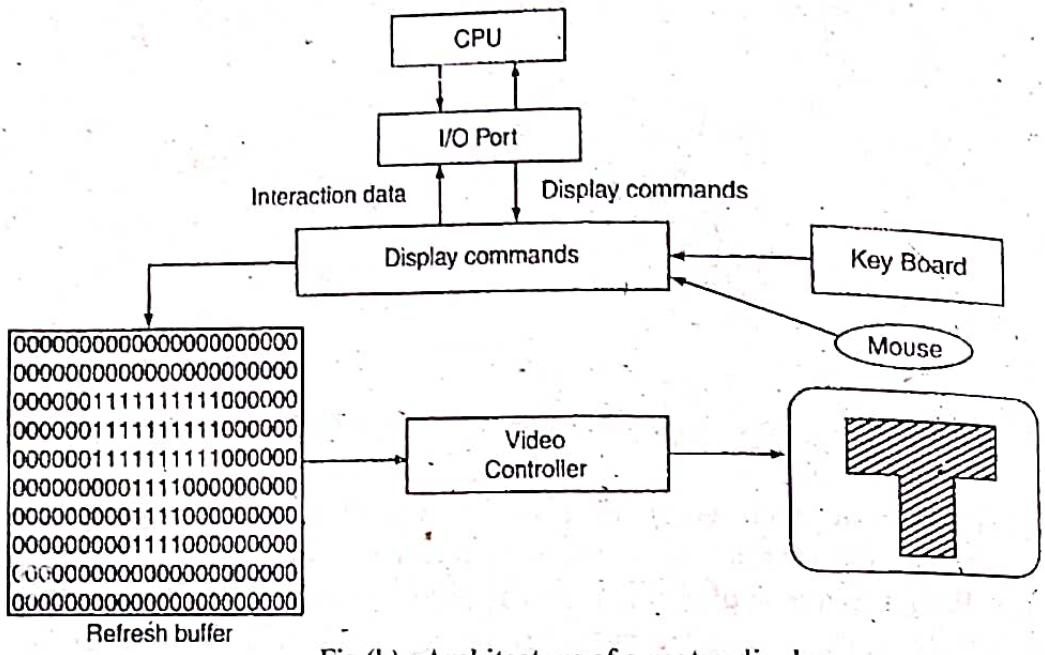


Fig.(b) : Architecture of a raster display

The raster scan proceeds as follows : Starting from the top left corner of the screen, the electron gun scans horizontally from left to right, one scan line, that is one row at a time, jumping (without tracing the line) to the left end of the next lower row until the bottom right corner is reached. Then it jumps (again without tracing) to the top left corner and starts again, finishing one complete refresh cycle.

Here, the beam is swept back and forth from the left to the right across the screen. When the beam is moved from the left to the right, it is ON. The beam is OFF, when it is moved from the right to the left as shown by dotted line in fig.(c).

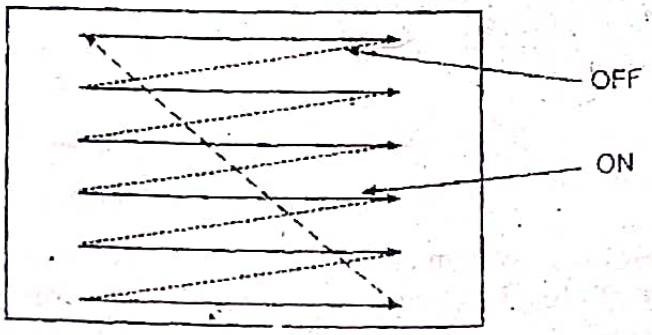


Fig.(c) : Raster scan cycle

When the beam reaches the bottom of the screen, it is made OFF and rapidly retraced back to the top left to start again. A display produced in this way is called raster scan display.

Section-B

Q.4.(a) Perform a 60° rotation of triangle A (0, 0), B(1, 1) C(5, 2) (10)

(i) About the origin

(ii) About P (–1, – 1)

Ans. Given : Triangle, ABC with coordinates as

A (0, 0), B (1, 1), C (5, 2)

∴ Triangle ABC in matrix form is

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{bmatrix}$$ (Row major)

$$\begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$ (Column major)

(a) Matrix of rotation is

$$R_{60^\circ} = \begin{bmatrix} \cos 60^\circ & -\sin 60^\circ & 0 \\ \sin 60^\circ & \cos 60^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now coordinate A', B', C' of rotated ΔABC can be found as

$$[A'\ B'\ C'] = R_{60^\circ} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \frac{1-\sqrt{3}}{2} & \frac{5-2\sqrt{3}}{2} \\ 0 & \frac{1+\sqrt{3}}{2} & \frac{5\sqrt{3}+2}{2} \\ 1 & 1 & 1 \end{bmatrix}$$

$A' = (0, 0)$

$B' = \left(\frac{1-\sqrt{3}}{2}, \frac{1+\sqrt{3}}{2}\right)$

$C'' = \left(\frac{5-2\sqrt{3}}{2}, \frac{5\sqrt{3}+2}{2}\right)$

(b) Rotation Matrix is

$$= \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} & \frac{-\left(1+\sqrt{3}\right)}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & \frac{\sqrt{3}-1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

Now, $$[A', B', C'] = \begin{bmatrix} \frac{1}{2} & \frac{-\sqrt{3}}{2} & \frac{-\left(1+\sqrt{3}\right)}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & \frac{\sqrt{3}+1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{-\left(1+\sqrt{3}\right)}{2} & \frac{-2\sqrt{3}}{2} & \frac{4-3\sqrt{3}}{2} \\ \frac{\sqrt{3}-1}{2} & \frac{2\sqrt{3}-1}{2} & \frac{6\sqrt{3}+1}{2} \\ 1 & 1 & 1 \end{bmatrix}$$

$$A' = \left[\frac{-\left(1+\sqrt{3}\right)}{2}, \frac{\sqrt{3}-1}{2}\right]$$

$$B' = \left[\frac{-2\sqrt{3}}{2}, \frac{2\sqrt{3}-1}{2}\right]$$

$$C' = \left|\frac{4-3\sqrt{3}}{2}, \frac{6\sqrt{3}+1}{2}\right|$$

Q.4.(b) Write the general from of a shearing matrix with respect to a fixed point $P(h, k)$. **(10)**

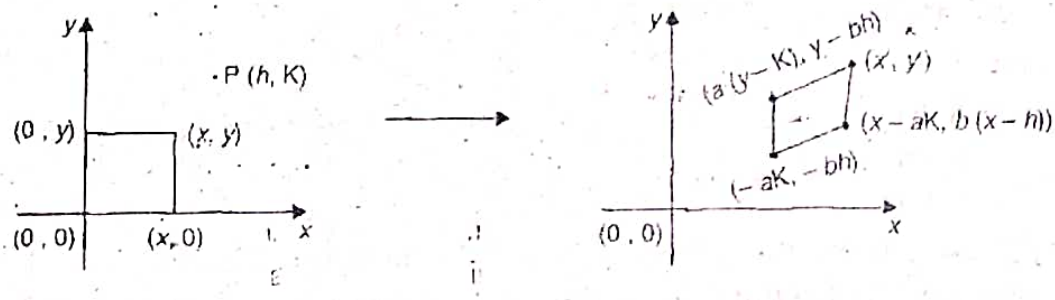
Ans. Shearing transformation w.r.t. any arbitrary point : Let us consider a case of simultaneous shearing w.r.t. an arbitrary point, P (h, K). Then, as shown in Fig., the shearing are as follows :

(*i*) In x–direction :

$$x' = x + a(y - K)$$

(*ii*) In y–direction :

$$y' = b(x - h) + y$$



The transformation matrix is as follows :

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & a & -aK \\ b & 1 & -bh \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

This transformation will shift a coordinate position vertically and horizontally by an amount proportional to its distance from the reference point (h, K).

Q.5. Contrast theefficiency of clipping between Sutherland-Cohen and Mid-point algorithm. Describe Sutherland-Hodgeman algorithm for polygon clipping. Explain why this alogrithm works for convex polygons. **(20)**

Ans. Sutherland-Cohen algorithm This line clipping algorithm divides our 2D space into 9 parts (8 outside regions + 1 inside region). This is showen in fig.(1). Each of the nine regions associated with the window is assigned a 4 bit code is idetify that region. Each end point of the line is then assigned a code a of the region in which it lies.

The number in Fig.(1) above are called as *outcodes* or *region codes*. An outcode in computer for each of the two points of a line. For any endpoint (x, y) of a line, the code can be determined that identifies in which region the endpoint lies. The outcode's bit's are set according to the following conditions :

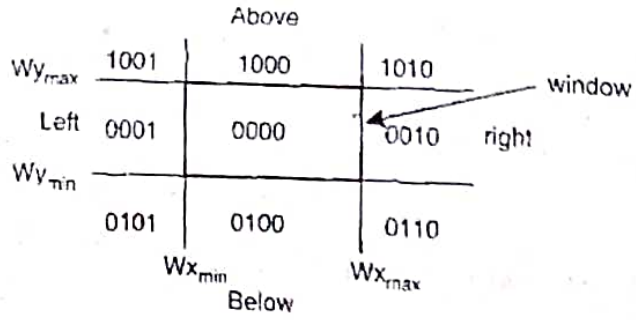


Fig.(1) : Outcodes

First bit, set to 1 : point lies to left of window.
Second bit, set to 1 : point lies to right of window
Third bit, set to 1 : point lies below/bottom of window.
Fourth bit, set to 1 : point lies above/tip of window.
The outcode bits which are of size 4 are as follows :

b_4	b_3	b_2	b_1
T	B	R	L

Where L- Left
R- Right
B-Bottom
T-Top

But please note that the out codes/region codes must be recalculated on each iteration after clipping occurs. A point that is in top-right of the viewport means on outcode of 1010 for instance.

Mid- point algorithm : It is an extension of divide-and-conquer based choen-sutherland algorithm. It is a recursive process in which we used binary searching. This is so because here we compute the outcodes for each of the line segment that has been divided at its mid point. The mid point coordinates $P_m(x_m, y_m)$ of a line segment joining $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ are given as :

$$x_m = \frac{x_1 + x_2}{2}$$

and

$$y_m = \frac{y_1 + y_2}{2}$$

For each of this sub-line segment the outcodes (or region codes) are determined and again logical AND's are taken.

(a) If both the endpoints of a line has outwards as (0000) then the line is totally visible.
(b) If both the outcodes are not zero then we have to take logical AND of both outcodes and then decide whether that line is visible or not.
(c) If the logical AND result is non zero, it means the line is totally invisible.
(d) If the logical AND result is zero, it means a line may be partially visible.

Flow chart of Sutherland Hodgman Algoritum : This algorithm reads as input the vertices of a polygon one at a time. For each input vertex, either zero, one or two output vertices are generated depending on the relationship of the input vertices to the clipping edge E.

Let $P \leftarrow$ input vertex

$S \leftarrow$ previous input vertex

$F \leftarrow$ first arriving input vertex.

Then the vertex or vertices to be outputed are determined according to the flowchart given in fig.(2). If the polygon has n edges then the edge $\overline{P_n P_1}$ is closing the polygon. In order to avoid the need to duplicate the input of P_1 as the final vertex, the closing logic is called after processing of the final input vertex P_n. This flowchart is shown in fig.(3).

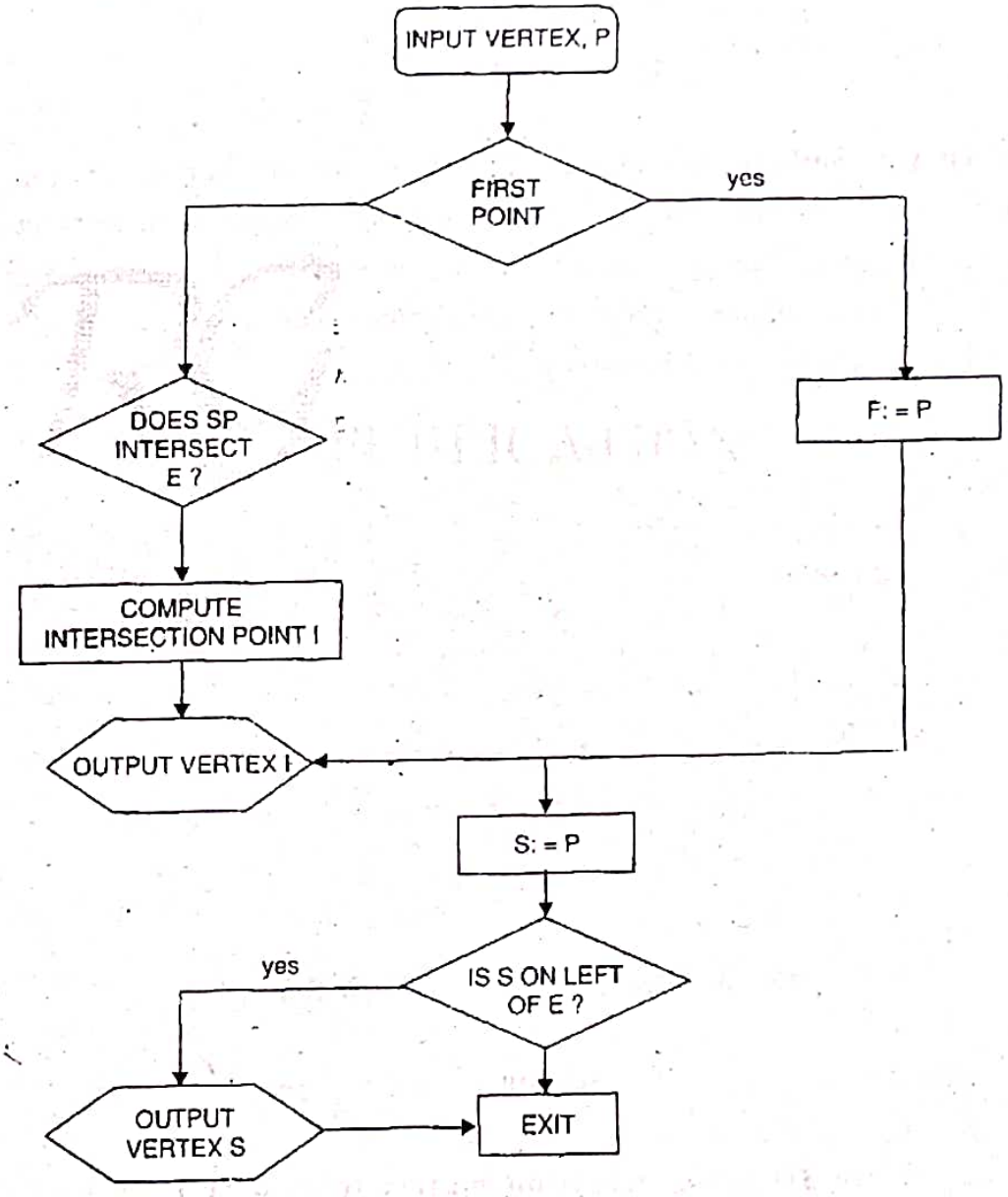


Fig.(2) : Flowchart for Sutherland-Hodgman Algorithm

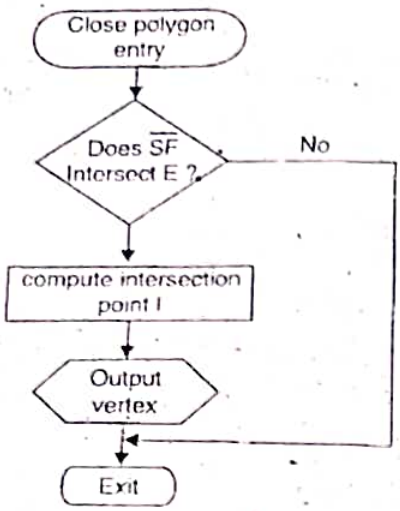


Fig.(3) : Closing logic

Problem with Sutherland-Hodgman Algorithm : All convex polygons are correctly clipped by this algorithm but concave polygons may be displayed with some extraneous lines. This occurs when the clipped polygon should have two or more separate sections. But there is only one output vertex list, the last vertex in the list is always joined to the first vertex i.e., we are formiing an edge between the last and first vertex.

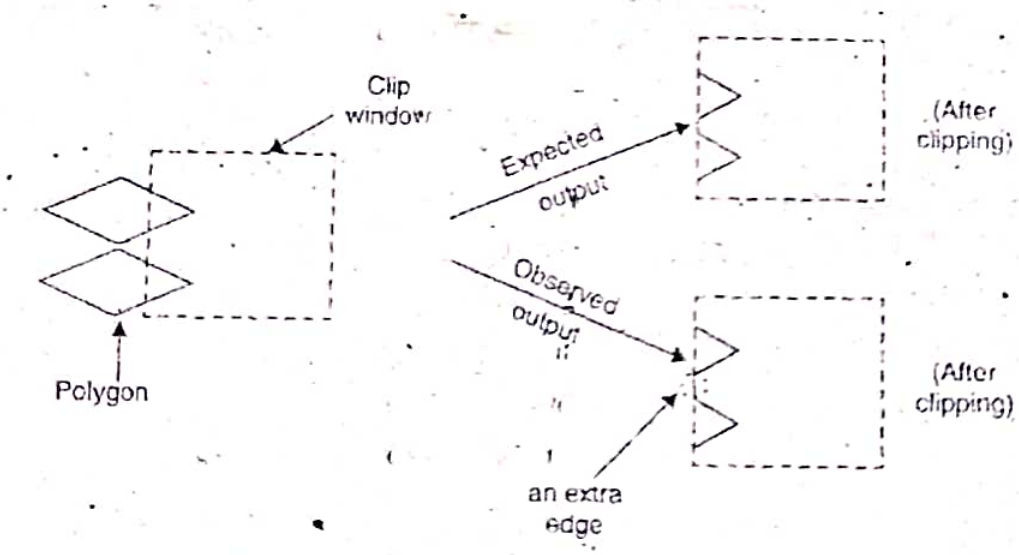


Fig.(4) : Problem with SH-Algorithm

Section-C

Q.6. (a) Write 3D transformation matrix to find reflection of a point P (15, 25, 35) about plane $z = 0$. **(10)**

Ans.

$$\begin{bmatrix}15 & 25 & 35 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 1\\ 0 & 1 & 0 & 1\\ 0 & 0 & -1 & 1\\ 1 & 1 & 1 & 1\end{bmatrix} = \begin{bmatrix}15 & 0 & 0 & 1\\ 0 & 25 & 0 & 1\\ 0 & 0 & -35 & 1\\ 1 & 1 & 1 & 1\end{bmatrix}$$

So the coordinates are (15, 25, – 35).

Q.6. (b) What is oblique projection? Provide some examples of oblique projection. **(10)**

Ans. Oblique Parallel Projection : *A projection in which the angle between the projectors and the plane of projection is not equal to 90° is known as an oblique projection.* An oblique projection is formed by parallel projections from a centre of projection at infinity that intersects the plane of projection at an oblique angle. Actually, oblique means slanting. There are three axis-vertical, horizontal and oblique. An oblique axis is also called as the receding axis and is drawn either at 30° or 45°. In oblique projection, the front face of the object is placed parallel to the plane of projection. So, we get its true size and shape shadow. This is shown in fig.(1) for an object AB. i.e., a line.

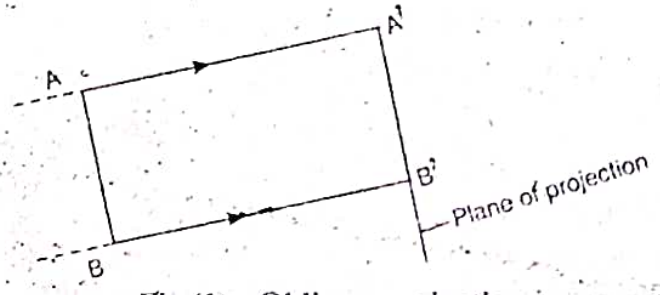


Fig.(1) : Oblique projection

An object with curves and circular features can be conveniently shown in oblique projection. We follow certain rules for selecting the position of an object of an oblique projection. They are as follows :

Rule 1 : Place the most circular outlines or the irregular contour parallel to the plane of projection.

Rule 2 : Place the lergest face of the object parallel to the plane of projection.

Rule 3 : For lengthy objects having circular outlines, place the circular face parallel to the plane of projection.

There are two common types of oblique parallel projection :

(a) Cavalier paraller Projection : The lines perpendicular to projection plane are preserved in length *i.e.* $L = 1$. The direction of projection is chose so that there is no foreshortestening of lines perpendicular to the xy-plane. In matrix from :

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

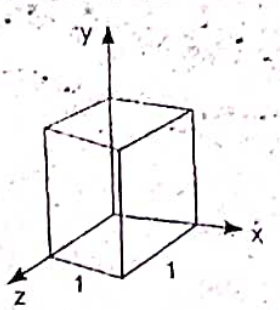


Fig. : (2)

The vector (0, 0, 1) maps to (cosθ, sinθ, 0) which has a length of 1. In this projection the angle between the oblique projectors and the plane of projection is 45° and the foreshortening factors for all three principles directions are equal. In this projection, the resulting figure is too thick.

(b) Cabinet projection : The lines perpendicular to projection plane are $\frac{1}{2}$ of their true length i.e, $L = \frac{1}{2}$ So, there is foreshortening of lines perpendicular to projection plane. In matrix form

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

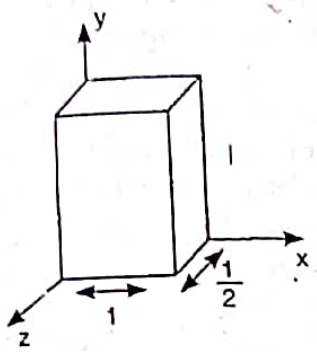


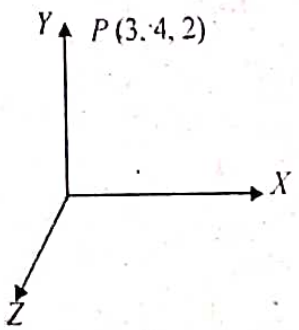
Fig. : (3)

Q.7. Write notes on :

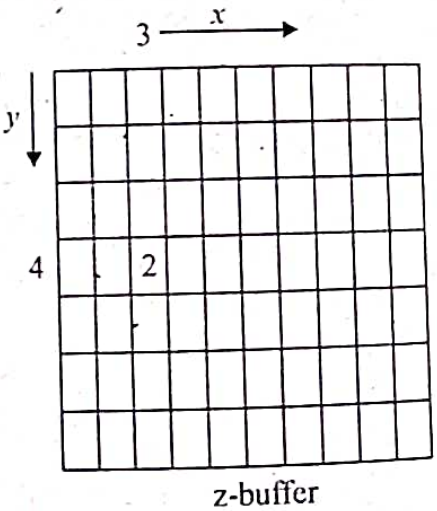
(a) Z- buffer algorithm (10)

(b) Subdivision algorithm (10)

Ans. (a) Z - Buffer Algorithm (Depth Buffer Algorithm) : The z-buffer is one of the simplest algorithms of the hidden surface removal. The technique was originally proposed by Catmull and is an image space method. The buffer is a simple extension of the frame buffer idea. A frame buffer is used to store the attributes (intensity) of each pixel in image space. The z-buffer is a separte depth buffer used to store the z-coordinate or depth of every visible pixel in image space as illustrated in fig.(1).



(a)



z-buffer

(b)

Fig.(1) : Representation of 3D point in z-buffer

The depth or z -value of a new pixel to be written to a frame buffer is compared to the depth or z -value of the pixel already stored in the z-buffer. If the comparison indicates that the z -value of the new pixel is greater than z-value already stored in the z-buffer, then z-buffer is up dated with the new z-value and intensity of new pixel is written in frame buffer, otherwise no action is taken.

Algorithm : Z-Buffer Algorithm for Hidden lines/Hidden Surface Removal

Step 1 : Set the frame buffer to the background intensity color.

$$\text{frame\_buffer}(x, y) = I_{background}$$

Step 2 : Set the z-buffer to the minimum value. $Z_{buffer}(x, y) = 0$

Step 3 : For each pixel (x, y) in the polygon, calculate the depth $z(x, y)$ at that pixel.

Step 4 : Compare the depth $z(x, y)$ with the value stored in the z-buffer at the location. $Z_{buffer}(x, y)$ to determine the visibility.

If $z(x, y) > Z_{buffer}(x,y)$, then write the polygon attributes (intensity, color, etc.) to the frame buffer at pixel location (x, y) and replace $Z_{buffer}(x,y)$ with $z(x, y)$ othwise no action is taken *i.e.*,

$$Z_{buffer}(x, y) = z(x, y)$$

$$\text{frame\_buffer}(x, y) = I_{proj}(x, y)$$

Where, $I_{proj}(x, y)$ is the projected intensity value for the surface at pixel position (x, y)

Step 5 : Repeat steps 3 to 4 for all polygons in arbitrary order.

Step 6 : Finally when all the polygons have been processed, the depth buffer contains depth value for the visible surfaces and the frame buffer contains the cooresponding intensity values for those surfaces.

Ans.(b)Subdivision algorithm : The only difficult part in Cohen-Southerland algorithm is to find the intersection point of line with window boundary. An alternative to finding intersection points by equation solving is based on bisection method of numerical analysis. The line segment is divided at its midpoint into two smaller line segments. The clipping categories of the two new line segments are then determined. Each segment in category 3 is divided again into smaller segments and categorized. The bisection and categorization process continues until all segments are in category 1 (visible) or category 2 (invisible). Th mid-point co-ordinates (x_m, y_m) of a line segment joining $P_1(x_1, y_1)$ to $P_2(x_2, y_2)$ are given by

$$x_m = \frac{x_1 + x_2}{2} \quad y_m = \frac{y_1 + y_2}{2}$$

We can call mid-point subdivision algorithm as a special case of Cohen-Sutherland algorithm because it used the same technique as Cohen-sutherland. It uses the line endpoint outcodes and associated tests to immediatedly identify which lines are visible and which are invisible. If both the endpoints of a line has outcodes 0000 then the lines is totally visible. If both the outcodes are not zero then we have to take logical AND of both outcodes and then we have to decide whether that line is visible or not. If the logical AND result is nonzero, it means the line is totally invisible and if the logical AND is zero, it means a line may be partially visible.

The algorithm is formalized in the following steps:

For each endpoint :

(a) If the endpoint is visible, then it is farthest visible point. The process is complete. If not, continue.

(b) If the line is trivially invisible, no output is generated. The process is complete. If not, continue.

(c) Divdie the line P_1P_2 at its midpoint, P_m. Apply the previous tests to the two segments P_1P_m and P_mP_2. If P_mP_2 is rejected as trivally invisible, the midpoint is an overestimation of the fartherst visible point. Continue with P_1P_m, otherwise the midpoint is an underestimation of the farthese visible point. Continue with P_mP_2. If the segment becomes so short that the midpoint corresponds to the accuracy of the machine or, as specified to the end points, evaluate the visibility of the point and the process is complete.

Section-I)

Q. [a] Explain Bezier method of curve drawing. [10]

Ans. Given a set of $(n + 1)$ control points P_0, P_1, P_2, ...P_n, a parametric Bezier curve (Bernstein-Bezier curve) segment is a weighted sum of $n + 1$ control points P_0, P_1, P_2, ...P_n that will fit to those points is mathematically defined by :

$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

where $B_{i,n}(t)$ are the Bezier blending function also known as Bernstein Basis functions (weights) defined as follows :

$$B_{i,n}(t) = C(n,i)\, t^i (1-t)^{n-i}$$

$$C(n,i) = \frac{n!}{i!(n-i)!}$$

Thus, a Bezier curve can be seen as a weighted average of all of its control points. Because all of the weights are positive, and because the weights sum to one, the Bezier curve is guaranteed to lie within the convex hull of its control points.

The Bezier curve of order $n + 1$ (degree n) has $n + 1$ control points. Following are the first three orders of Bezier curve definitions :

Linear $P(t) = (1 - t)P_0 + tP_1$

Quadratic $P(t) = (1 - t)^2P_0 + 2(1 - t)tP_1 + t^2P_2$

Cubic $P(t) = (1 - t)^3P_0 + 3(1 - t)^2tP_1 + 3(1 - t)t^2P_2 + t^3P_3$

$B(u)$ is a continuous function in 3 space defining the curve with N discrete control points P_k. $u = 0$ at the first control point ($k = 0$) and $u = 1$ at the last control point ($k = N$).

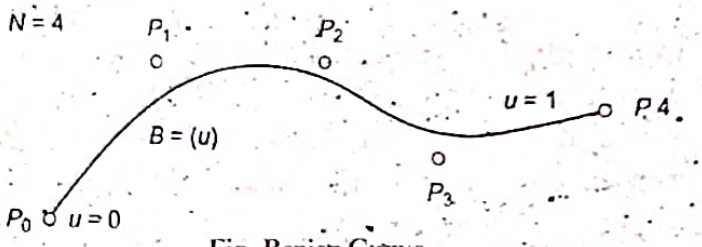


Fig. Bezier Curve

Q.8.(b) Describe methods of polygon shading. (10)

Ans. Various shading model for polygons are as follows :

1. Constant Intensity Shading : The fast and simplest method for shading polygon is constant shanding, also known as faceted shading or flat shading. In this method, illumination

model is applied onle once for each polygen to determine single intensity value. The entire polygon is then displayed with the single intensity value.

This method is valid for the following assumptions :

(i) The light source isat infinity, so N.Lis constant across the polygon face.

(ii) The viewer is at infinity, so V.R is constart over the surface

(iii) The polygon represents the actual surface b eing modeled. and is not an approximation to a curved surface.

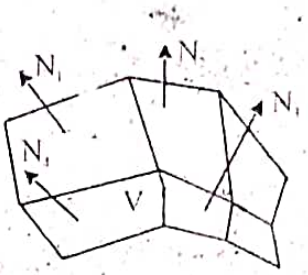


Fig. (a)

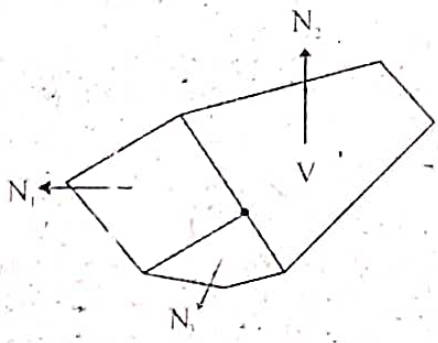


Fig.(b) : Calculation of normal vector at polygon vertex V

2. Gouraud Shading : This intensity interpolartion scheme. developed by Henri Gouraud and generally referred to as Gouraud shading. The polygon surface is displayed by linearly interpolating intensity values across the surface. Intensity values for each polygon are matched with the values of adjacent polygons along the common edges. Thus, eliminates the intensity discontinuities that can occur in flat shading.

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

(i) Determine the average unit normal vector at each polygon vertex.

(ii) Apply an illumination model to each polygon vertex to determine the vertex intensity.

(iii) Lineraly interpolate the vertex intensities over the surface of the polygon.

As shown in the fig.(b), there are three surface normals N_1, N_2 and N_3 of polygon sharing vertex V is given as

$$N_v = \frac{N_1 + N_2 + N_3}{|N_1 + N_2 + N_3|}$$

3. Phong Shading : A more accurate method for rendering a polygon surface is to interpolate normal vectors 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, interpolates the surface normal vector N, instead of the intensity.

By performing following steps, we can display polygon surface using Phong shading.

(i) Determine the average unit normal vector at each polygon vertex.

(ii) Linearly intepolate the vertex normals over the surface of the polygon.

(iii) Apply an illumination model along each scan line to determine projected pixel intensities for the surface points.

As shown in the fig.(c), the normal vector N for the scan line intersection point along the edge between vertices 1 and 2 can be obtained by vertically interpolating between edge endpoint normals.

$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

Like, Gouraud shading, here also we can use incremental methods to evaluate normals between scan lines and along each individual scan line. Once the surface normals are evaluated the surface intensity at the point is determined by applying the illumination model.

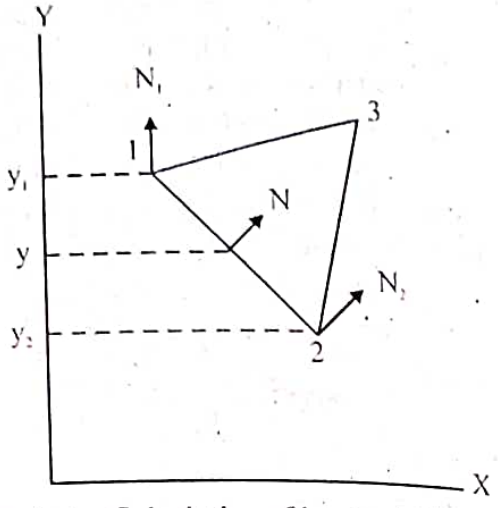


Fig.(c) : Calculation of interpolation of surface normals along a polygon edge.

4. Fast Phong shading : Surface rendering with Phong shading can be speeded up by using approximations in the illumination-model calulations of normal vectors. Fast Phong Shading approximates the intensity calculations using a Taylor-series expansion and triangular surface patches.

Since Phong shading interpolates normal vectors from vertex normals, we can express the surface normal N at any point (x, y) over a triangle as

$$N = Ax + By + C$$

where vectors A, B and C are determined from the three vertex equations :

$$N_k = Ax_k + By_k + C, \qquad k = 1, 2, 3.$$

with (x_k, y_k) denoting a vertex position.

Discarding the reflectivity and attenuation parameters, the calculations for light source diffuse reflection from a surface point (x, y) as

$$I_{diff}(x, y) = \frac{L.N}{|L||N|} = \frac{L.(Ax + By + C)}{|L||Ax + By + C|} = \frac{(L.A)x + (L.B)y + L.C}{|L||A_x + B_y + C|}$$

Now the expression can be rewritten in the form as

$$I_{diff}(x, y) = \frac{ax + by + c}{(dx^2 + exy + fy^2 + gx + hy + i)^{1/2}}$$

Where parameters a, b, c and d are used to represent the various dot products. We can express the denominator as a Tylor-series expansion and retain terms up to second degree in x and y.

$$I_{diff}(x, y) = T_5 x^2 + T_4 xy + T_3 y^2 + T_2 x + T_1 y + T_0$$

where each T_k is a function of parameter a, b, c and so forth.

Q.9. Write notes on : **(20)**

(a) Curve interpolation

(b) Fractals

Ans. (a) Curve interpolation : Some complex curves have to simple mathematical definition. We can thus draw such curves using approximations if we have an array of sample points. We can guess what the curve should look like between the sample points. If the curve is smooth, our samples are close enough together, we can guess the curve's shape. We fill in the portions of the unknown curve with pieces of known curves which pass through the nearby sample points. Since the known and unknown curves share these sample points in a local region.

We assume that in this region the two curves look alike. We fit a portion of the unknown curve with a curve that we know. Now we can fill in a gap between the sample point by finding the coordinates of points along the known approxmatiny curve and connecting these points with line segments. Our objective is to find the suitable mathematical expression for the known curve. There are polynomial, trigonometric, exponential and other classes of functions that can be used to approximate the curve, usually, polynomial functions in the parametric form are preferred. The polynomial functions in the parametric form are

$$\left.\begin{aligned} x &= f_x(u) \\ y &= f_y(u) \\ z &= f_z(u) \end{aligned}\right\}$$

Why to use parametric form ?

(i) It treats all directions equally.

(ii) It allows multiple values i.e, serveral y or z values for a given x, so that curves can double back or even cross themselves. See Fig.(1).

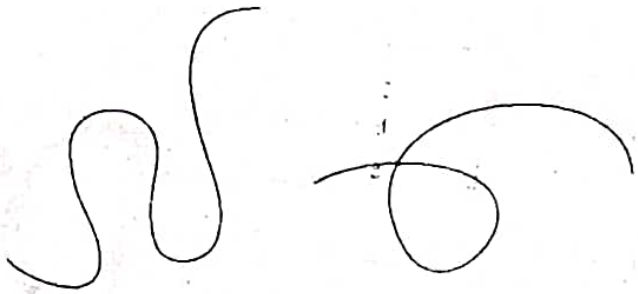


Fig.(1) : Curves with double back or crossing themselves

Please note that we have to draw the curve by determining the intermediate points between the known sample points. This can be achieved using interpolation technique.

Ans. (b) Fractals : According to Beroit Mandelbrot of IBM Research Lab, a fractal is "a rough or fragmented reduced copy of the whole". The term fractal is actually associated with randomly generated curves and surfaces that exhibit a degree of self similarity. They are used to provide naturalistic shapes for respresenting object like coast lines, ruggesd mountain's grass, five trees, rivers etc. These rough, jagged and random surfaces are called as *fractals* Mandelbroat calls various forms of self similar curves as fractals which is a short form of 'fractional dimention'. Please note that a line is 1D, a plane is 2D but a curve of infinite length that fits into a finite region of the plane must have a dimension somewhere between 1 and 2. Mandbort devised a method for computing the fractional dimension for such curves. Also note that a fractalization process can be used to generate shapes that resemble a tree.

A fractal object is generated by respeatedly applying a specified transformation function to points with in a region of space. If $P_0 = (x_0, y_0, z_0)$ is a selected initial point, each iteration of a transformation function, F, generates succesive levels of details with the calculations.

$$P_1 = F(P_0)$$
$$P_2 = F(P_2)$$

$$P_1 = F(P_0)$$

The transformation function may be defined in terms of geometric transformations (Scaling, translation, rotation) or it can be set up with non-linear coordinate transformations and decision parameters. Fractal objects contain infinite detail and so we apply the transformation function, a finite number of times. Therefore, the objects that we display actually has finite dimensions. Basically, we talk of two measures of object's dimension.

(*a*) Topological dimension

(*b*) Fractal dimension

Basic properties of Fractals : A fractal object has some basic properties :

(i) Fractals have infinite details at every region which can be seen on successive magnification.

(ii) There is Self Similarity between the object parts and the overall features of the parent object. Please note that there are many figures which are not fractals but they are self similar. Some fractals are also known as *well known fractals* like the *Koch Curve* shown below

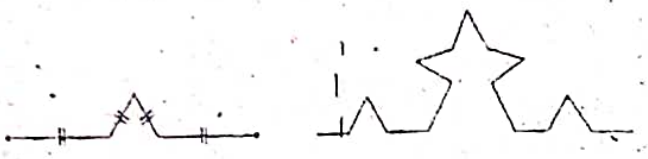


Fig. : Well known Fractals .

Types of Fractals : We can classify fractals are follows :

(*a*) Self similar fractals

(*b*) Self affine fractals

(*c*) Invariant fractals

Self similar fractals have parts that are scaled down version of the entire object. Starting with an initial shape, we construct the object subparts subparts by applying a scaling parameter, S, to the overall shape.

Self affine fractals have parts that are formed with different scaling paraments, S_x, S_y, S_z in different coodinate directions. And we can also include random variations to obtain statistically self affine fractial. Terrain, water and clouds are typically modeled with statistically self affine fractal construction methods.

Inveriant fractal sets are formed with nonlinear transformations. This class of fractals includes self squaring fractals, such as the Mandelbrot set, which are formed with squaring functions in complex space ; and self inverse fractals formed with inversion procedures.

COMPUTER GRAPHICS

Dec 2015

Paper Code: CSE-303-F

Note: *Attempt five questions in all, selecting one question from each section. Q. No. 1 is compulsory.*

Q.1.(a) What is the difference between random scan and raster scan display.

Ans. Random scan vs raster scan : Difference between Vector scan display and Raster Sc Display :

Sr.No.	Vector(random) scan display	Raster Scan Display
(1)	Vector display only draws lines and characters.	Raster display has ability to display areas fillled with solid colours or patterns.
(2)	Don't use interlacing.	Uses interlacing.
(3)	In vector scan display the beam is moved between the end points of the graphics primitives.	In raster scan display the beam is moved all over the screen one scan line at a time from top to bottom and then back to top.
(4)	Higher resolution.	Lower resolution.
(5)	More expensive.	Less expensive.
(6)	Uses monochrome or beam-penetration type.	Uses monochrome or shadow mask type.
(7)	Vector display draws a continuous and smooth lines.	Raster display can display mathematically smooth lines polygons and boundaries of curved primitives only by approximating them with pixel on the raster grid.
(8)	Editing is easy.	Editing is difficult.
(9)	Refresh rate depends directly on picture complexity.	Refresh rate independent of picture complexity.
(10)	Scan conversion is not required.	Graphics primitives are specifies in term of their end points and must be scan converted into their corresponding pixel in the frame buffer.

Q.1.(b) Explain view port and clipping.

Ans. Viewport : A viewport is an area of the display device on which the window data is presented. A two-dimensional regular viewport is specified by giving the left, right, bottom and

top edges of a ractangel. Viewport values may be given in actual physical device coordinates when specified in actual physical device coordinates; they are frequently given using integers. Viewport coordinates may be normalized to some arbitrary range, for example, $0 \le x \le 1.0, 0 \le y \le 1.0$, and specified by floating point numbers.

The display screen is 2-dimensional. Sometimes, it is very difficult to view all the details of an object. The most common views required to represent the object details are : isometric view, orthographic view, front view, top view, left side view, right side view and sectional views. To accommodate the display of several views simultaneously the display screen is divided into viewports. A typical screen layout with four viewports is shown in fig.(1) and (2).

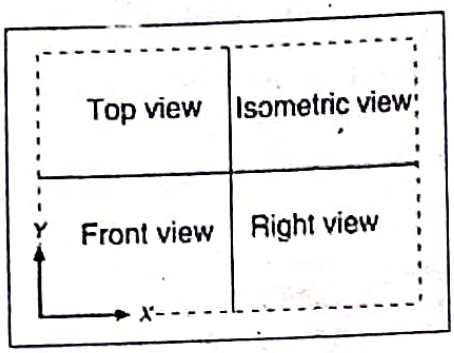


Fig.(1) : A typical screen layout

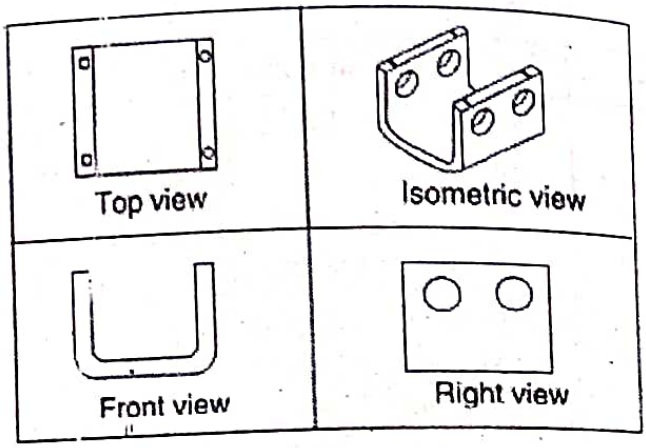


Fig.(2) : Illustration of different views of an object

Clipping : Generally, any procedure, that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as a clipping or simply Clipping. The region against which an object is to clipped is called a clip window.

Applications of clipping include extracting part of a defined scene for viewing identiflying visible surfaces in three dimensional views, antiatiasing line segements or objects boundaries; creating objects using solid modelling procedurces; displaying a multiwindow environment, and drawing and painting operations that allow parts of a picture to be selected for copying, moving erasing or duplicating. Depending on the application, the clip window can be a general polygon or it can even have curved boundaries. Chipping algorthms can be applied in wordld coordinats, so that only the contents of the window interior are mapped to device coordinates.

Q.1.(c) Explain the following with example: Translation, scaling.

Ans. Translation : Translation consists of a shift of the object parallel to itself in any direction in the (x, y) plane. Any such shift can be accomplished by a shift in x-direction plus a shift in y-direction. If the amount of x-shaft is called t_x and the amount of y-shaft t_y, the translation of the point (x, y) into the point(x', y') is expressed by the formulas.

$$x' = x + t_x$$

$$y' = y + t_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

To translate an object with multiple points, we just translate each point individually and connect them together.

Translation is a rigid-body transformation that moves objects without deformation. That is, every point on the object is translated by the same amount. Translatoin is the only transformation that is not in relation to a reference point. Its effect is independent of the original position of the object.

Scaling : Scaling is a transformation that changes the size or shape of an object. Scaling with respect to origin can be carried out by multiplying the co-ordinate values (x, y) of each vertex of a polygon, or each endpoint of a line by scaling factors S_x and S_y respectively to produce the coordinates (x', y').

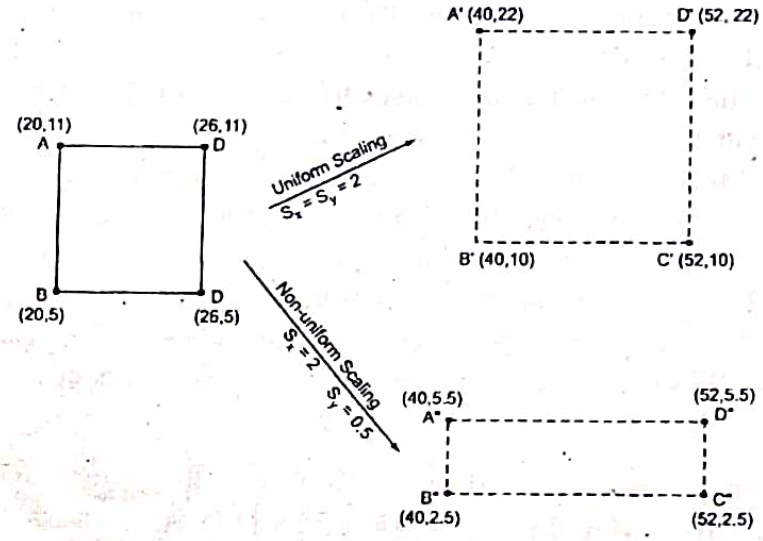


Fig. : (b)

The mathematical expression for pure scaling is

$$x' = S_x * x$$

$$y' = S_y * y$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

or symbolically $[X'] = [T]\,[X]$

Q.1.(d) What are the various operations that can be applied on image.

Ans. There are various operations that can be performed on the image are

(i) Translate or move or shift

(ii) Rotate

(iii) Scale or zoom

(iv) Reflect or flip

(v) Shear or twist or Torque

(vi) Morphing and Dithering

Translation of an object : Translation is a rigid body transformation that moves objects without distorsion. Every point on the object is translated by the same amount & there exists one to one correspondence between the transformed points & original points.

Rotation : This transformation is used to rotate objects about any point in a 2 – D. To accomplish rotation, we specify

(i) Rotation angle(θ)

(ii) Rotation point (pivet point)

+ ve value of rotation angle defines that rotation is in anticlockwise or counter-clockwise direction.

– ve values of rotation defines that rotation is in clockwise direction. Rotation may be performed in counter clockwise or clockwise.

Scaling : Scaling is a transformation that changes the size or shape of an image.

Reflection : A reflection is a transformation that produces a mirror image of an image. The mirror image for a 2D reflection is generated relative to an axis of rotation by rotating the image is about the reflection axis.

Shear : The shear transformation causes the image to slant. It is also called twist or torque or till transformation.

Morphing : Morphing an effect which is used to manipulate still images. Through, morphing, we can also create animated transformations. Morphing allows you to smoothy blind two images so that one image seems to melt into next, hence producing amusing results.

Dithering : The process whereby the color value of each pixel is changed to the closet matching color value in the target palette, using a mathematical algorithm, is called as dithering. The dithered image renders a good approximation of the original depending upon the algorithm used.

Q.1.(e) Difference between uniform *B*-Spline and non uniform *B*-spline.

Ans. Unifrom B-Spline : Those *B* - Spline in which difference between knot values is equal. If knot vector is defined as -

$$T = [u_0, u_1, \ldots u_m] \text{ then}$$

$$u_0 - u_1 = u_1 - u_2 = \ldots\ldots\ldots = u_{m-1} - u_m .$$

∴ Blending function can be calculated as –

$$B_{i,p}(u) = B_{i+1,d}[u + \Delta u]$$
$$= B_{i+2,d}[u + 2\Delta u]$$

Non Uniform B -Spline : These curves have unequal differences between their knot values and multiple internal knot values.

For example $T = [0, 2, 2, 3, 4, 4, 4]$

Section-A

Q.2.(a) What is computer graphics? Indicate the importance and application area of computer graphics.

Ans. Computer Graphics : In computer graphics, pictures or graphics objects are presented as a collection of discrete picture elements called pixels (picture element).

Importance of Computer Graphics : The importance of computer graphics are as follows :

(i) It provides tools for producing pictures not only of concrete real world objects but also of abstract, synthetic objects such as mathematical surface in 4D and of data that have no inherent geometry such as survey results.

(ii) It has to ability to show moving pictures and thus its is possible to produce animations with computer graphics.

(iii) With computer graphics user can also control the animation speed, portion of the view, the goniometric relationship the object in the scane to one another, the amount of detail shown and on.

(iv) The computer graphics provides tool called *motion dynamics*. With this tool user can move and tumble objects with respect to a stationary obsever, or he can make objects stationary and the viewer moving around them. A typical example is walk through made by builder to show flat interior and building surroundings. In many case it is also possible to move both objects and viewer.

(v) The computer graphics also provides facility called *update dynamics*. With update dynamics it is possible to change the shape, colour, or other properties of the objects being viewed.

(vi) With the recent development of *digital signal processing* (DSP) and *audio synthesis chip*, the interactive graphics can now provide audio feedback along with the graphical feedbacks to make the simulated environment even more realistic.

Applications of Computer Graphics : Computer Graphics has been widely used, such as graphics presentation, paint systems, computeraided design (CAD), image processing, simulation and virtual reality, entertainment, etc.

1. Computer-Aided Design (CAD) : It is one of the best and biggest application of computer graphics. This is particularly used in engineering applications such as building and other structural design and industrial design, design of manufacturing process, automobiles and aircraft, ships and spacecraft, very large scale-integrated (VLSI) chips, optical systems, and telephone and computer networks.

2. Computer Art : Computer graphics is used to produce picture that express a message and attract attention. In professions involving artist and in other business fields, such as fashion design, architecture, these applications of computer graphics are widely used because of effective speed of computer and easy modifiability capability.

3. Multimedia : Multimedia is the field concerned with the computer-controlled integration of text, graphics, drawings, still and moving images (video) animation, audio and any other media where every type of information can be represented, stored, transmitted and processed digitally.

4. Presentation Graphics : Presentation of data in pictorial form is always preferred since it is more understood than textual data. Presentation of graphics is widely used in posters, brochures, magazines, newspapers, training purpose, etc. This presentation may be sequential still or animated on computer monitor.

5. Internet : The word internet is derived from two words *Interconnection* and *Networks*. Also referred as the Net. Internet is a worldwide system of computer networks, that is, network of networks, which allows the participants to share information on those linked computers.

Internet is a much more broader concept than entertainment. The internet would be positively boring without graphics.

6. Graphical User Interface (GUI) : Graphical User Interface is related to learning and use of packages. It is not possible to learn packages in less times without graphic items

present on the screen. Mostly, software developed today must have a Graphical User Interface and interactivity.

7. Simulation and Virtual Reality : The computer graphics helps here in simulating the views that the operator would see under various circumstances. Complex, mechanical, chemical and industrial processes can be simulated in action so that persons can be trained in their operation before they handle the actual equipment. Such a practice avoids wastages of cost and time.

Simulation has now reached up to such a length that those user follows wearing some special devices on their body such as on eyes, ears, and fingers.

8. Desktop Publishing (DTP) : DTP is related to the production of journals, printing newsletters, book publishing in small organization. This technology has expanded to such an extent that many book publishers, newspapers would not be able to deliver material without DTP.

9. Medical Applications : Computer graphics has now becomes a very important tool of diagnosis and treatment in the hands of doctors, particularly surgeons. Two-dimensional colorful images of cross-sections of human body or specific organs and limbs are produced. These 2D images are transformed to special investigated tools; a surgeon can rehearse his operation procedure on the computerized 3D image of the patient's part of body.

Q.2.(b) Explain DDA Line drawing algorithm.

Ans. Digital Differential Analyzer (DDA) Algorithm : The vector generation algorithms (and curve generation algorithms) which step along the line (or curve) to determine to pixels which should be turned ON are sometimes called digital differential analyzers (DDA's). The name comes from the fact that we use the same technique as a numerical method of solving differential equations.

We know that the slope of a straight line is given as

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

This differential equation can be used to obtain a rasterized straight line.

For any given Δx (x interval) along a line, we can compute the Δy (y internal) as

$$\Delta y = m . \Delta x$$

i.e.,
$$\Delta y = \frac{y_2 - y_1}{x_2 - x_1} \Delta x$$

Similarly, we can obtain the x interval Δx corresponding to Δy as $\Delta x = \frac{\Delta y}{m} = \frac{x_2 - x_1}{y_2 - y_1} \Delta y$

Once the intervals are known, the next values for x and y are obtained as

$$x_{i+1} = x_i + \Delta x$$

$$x_{i+1} = x_i + \left(\frac{x_2 - x_1}{y_2 - y_1} \right) \Delta y \quad \text{...(i)}$$

and
$$y_{i+1} = y_i + \Delta x$$

$$y_{i+1} = y_i + \left(\frac{y_2 - y_1}{x_2 - x_1} \right) \Delta x \quad \text{...(ii)}$$

For simple DDA either Δx or Δy, whichever is larger, is chosen as some raster unit, i.e.,

If $|\Delta x| \geq |\Delta y|$ then

$\Delta x = 1$

else $\Delta y = 1$

if $\Delta x = 1$ then

$$y_{i+1} = y_i + \frac{y_2 - y_1}{x_2 - x_1}$$

and $x_{i+1} = x_i + 1$

If $\Delta y = 1$ then

$$y_{i+1} = y_i + 1$$

$$x_{i+1} = x_i + \frac{x_2 - x_1}{y_2 - y_1}$$

DDA algorithm :

(i) Read the line end point (x_1, y_1) and (x_2, y_2)

(ii) $\Delta x = |x_2 - x_1|$

$\Delta y = |y_2 - y_1|$

(iii) If $(\Delta x \geq \Delta y)$ then

length $= \Delta x$

else length $= \Delta y$

(iv) Select the raster unit *i.e.*,

$$\Delta x = \frac{(x_2 - x_1)}{\text{length}}$$

$$\Delta y = \frac{(y_2 - y_1)}{\text{length}}$$

(v) $x = x_1 + 0.5 * \text{Sign}(\Delta x)$

$y = y_1 + 0.5 * \text{Sign}(\Delta y)$

Sign function makes the algorithm work in all quadrant. It returns –1, 0, 1 depending on whether is agreement is $< 0, = 0, > 0$ respectively. The factor 0.5 makes it possible to round the values in the integer function rather than truncating them.

(vi) Now plot the point

$i = 1$

```
while (i ≤ length)
{
    Plot (integer (x), interger (y))
    x = x + Δx
    y = y + Δx
    i = i + 1
}
```

(vii) Stop.

Q.3.(a) How Bresenhman's algorithms can be used for generating circle ? Explain.

Ans. The Bresenham's circle drawing algorithm considers the eight way symmetry of the circle. It plots 1/8th part of the circle from 90° to 45°. As circle is drawn from 90° to 45°, the x-moves in + ve direction and y moves in the – ve direction.

To achieve best approximation to the true circle we have to select those pixels in the raster that falls the least distance from the true circle. Let us observe the 90° to 45° portion of the circle, each new point closest to the true circle can be found by applying either of two options:

(a) Increment in positive x direction by one unit

or

(b) Increment in positive x direction and negative y direction both by one unit.

If P_n is a current point with co-ordinates (x_n, y_n) then the next point could be either A B.

We have to select A or B, depending on which is close to the circle, and for that we have to perform some test.

Test closer pixel amongst these two can be determined as follows.

The distance of pixels A and B from theorigin (0, 0) are given by

$$d_A = \sqrt{(x_{n+1} - 0)^2 + (y_n - 0)^2}$$

$$d_A = \sqrt{x_{n+1}^2 + y_n^2}$$

and

$$d_B = \sqrt{x_{n+1}^2 + y_{n-1}^2}$$

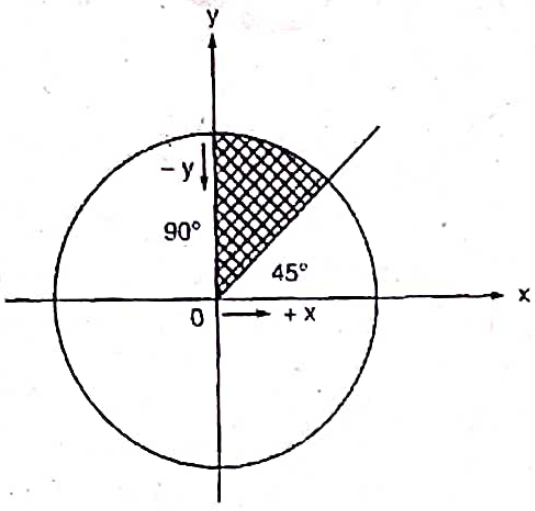


Fig. : (a)

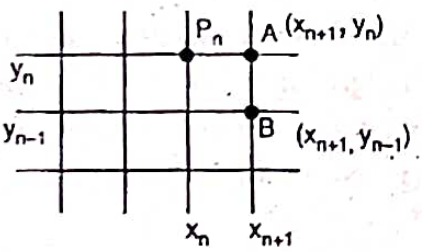


Fig. : (b)

Now, the distances of pixels A and B from the true circle whose radius r are given as

$$\delta_A = d_A - r \text{ and } \delta_B = d_B - r$$

To avoid square root in derivation of decision variable, we use

$$\delta_A = d_A^2 - r^2 \text{and } \delta_B = d_B^2 - r^2$$

But δ_A is always positive and δ_B is always negative. Therefore we can define decisio variabled_i as

$$d_i = \delta_A + \delta_B$$

If $d_i < 0$ i.e.. $\delta_A < \delta_B$ = then only x is incremented.

Otherwise x is incremented in positive direction and y is incremented in negative direction i.e.

For $d_i < 0$

$$x_{i-1} = x_i + 1$$

else

$$x_{i-1} = x_i + 1$$
$$y_{i-1} = y_i - 1$$

The equation for d_i at starting point i.e., $x = 0$ and $y = r$ is as follows

$$d_i = \delta_A + \delta_B$$
$$= x_{n+1}^2 + y_n^2 - r^2 + x_{n+1}^2 + y_{n-1}^2 - r^2$$
$$= (0 + 1)^2 + r^2 - r^2 + (0 + 1)^2 + (r - 1)^2 - r^2$$
$$= 1 + 1 + r^2 + 1 - 2r - r^2$$
$$= 3 - 2r$$

As stated earlier, if $d_i < 0$ then it implies A is closer to the true circle, hence increment only x value.

i.e., If $d_i < 0$ then $x_{i-1} = x_i + 1$

$$y_{i-1} = y_i - 1$$

So, recompute the new decision value, by substituting new value of x_i we get

$d_{i-1} = d_i + 4x_i + 6$ If $d_i < 0$

$d_{i-1} = d_i - 4(x_i - y_i) + 10$ If $d_i \geq 0$

Algorithm for Bresenham's circle drawing :

(1) Read the radius r of the circle.

(2) Initialize the decision variable

$$d = 3 - 2 * r$$

(3) $x = 0$

$$y = r$$

(4) If we are using octant symmetry to plot the pixels then until ($x < y$) we have to perform following steps.

if ($d < 0$) then

$$d = d + 4x + 6$$
$$x = x + 1$$

else

$$d = d + 4(x - y) + 10$$
$$y = y - 1$$
$$x = x + 1$$

(5) Plot (x, y)

(6) Determine the symmetry points in other octants also.

(7) Stop.

Q.3.(b) Write and explain boundary filled algorithm.

Ans. Boundary-fill algorithm : In this method, area filling starts from a random point, called a seed, inside a region and continues painting towards the boundary. If boundary is monocolour, then the fill algorithm proceeds outward pixel by pixel until the boundary colour is encountered. Such a method is called the boundary -fill algorithm.

There may be methods for proceeding to neihbouring pixels from the seed point :

(i) Four adjacentt point may be tested

(ii) Eight adjacent point may be tested

Ir first case, left, right, above and below the seed point area tested and known as 4 connected is and is used for partical filling of areas.

In second case, also includes for diagonal pixels and used to fill more complex figures. This method is called 8-connected.

Algorithm :

1. A Random point is considered *i.e.*, seed.
2. Each pixel to the left, right above and below are tested (in case of 4 connected). (If 8- connected, diagonal pixels are also tested).
3. Repeat process until all pixels up to the boundary color for the area have been tested (*i.e.* when boundary colour is hit.)

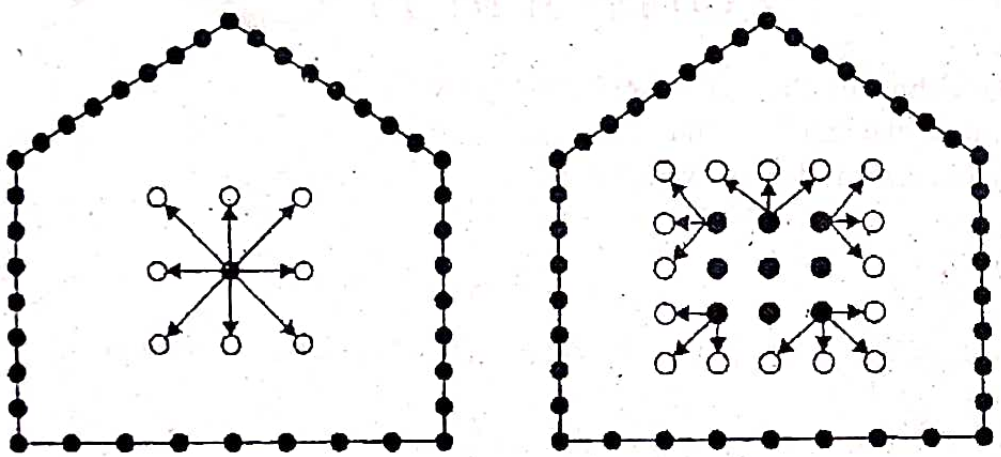


Fig : Boundary filling for polygon for 8 - connected.

This method can be employed more efficiently using stack for adjacent pixels. In this method, h:rizontal pixels are connsiderd across scan line, instead of proceeding 4-connected or 8-connected adjacent pixels. Then we need only stack a beginning position for each horizontal pixel in scanline, instead of stacking all adjacent points around the seed. In this way all scanline, present in the polygon are considered respectively for filling the bounded area by the single colour boundary.

Section-B

Q.4. Describe the transformation used in magnification and reduction with respect to origin. Find the new coordinates of the triangle A(0, 0), B (1, 1), C (5, 2) after it has been.

(a) Magnified to twice its size

(b) reduce to half its size.

Ans. Scaling transformation used in magnification and reduction with respect to origin. Scaling is defind as the process of changing the size or shape of an object (elongation or shrinking) in the x and y direction. Scaling with respect to origin can be done by multiplying the coordinate values (x, y) of each vertex of a polygon or each end point of a line by scaling factors S_x and S_y respectively to produce the coordinates (x', y'). The expression for pure scaling is

$$x' = S_x * x$$
$$y' = S_y * y$$

In matrix form, we get :

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
$$= \begin{bmatrix} xS_x & yS_y \end{bmatrix}$$

or $$[X'] = [T] \cdot [X]$$

After appling scaling factor or matrix on any object, the coordinate is either increased or decreased depending on S_x and S_y various values of scaling factors can be -

(a) $S_x = S_y > 1$ means *uniform scaling* or extension or elongation of object occurs.

(b) $S_x = S_y < 1$ means *uniform reduction* or compression or shrinking of object occurs.

(c) $S_x = S_y = 1$ means *no change* in object.

(d) $S_x \neq S_y$ means *shape distortion*.

(e) $S_x = S_y = 0$ means *object collapses*.

(f) $S_x = S_y < 0$ has no practical meaning.

Non-uniform expansion or compressions also occur. It depends on whether S_x and S_y are indvidaully > 1 or < 1 but unequal. Such a scating is also called as *differential scaling*. In uniform scaling, the basic object shape remains unaltered. On the other hand, in differential scaling both shape and size changes. In *pure uniform scalling* with factors < 1, objects move closer to origin while factors > 1 moves object farther from origin and at the same time decreasing or increasing the object size.

For example : Scaling of a triangle means

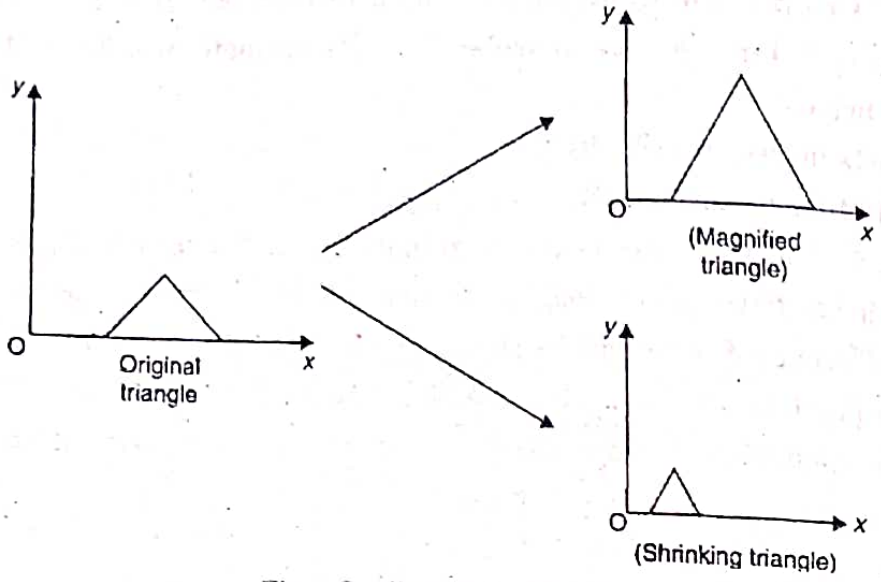


Fig. : Scaling of a triangle.

(a) Magnified to twice its size : We need to do three things here -

(i)Translate (move) this triangle to origin first.

(ii) Scale it.

(iii)Translation it back to the required position.

So, for steps *a*, *b* and *c*, the matrices are :-

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Given that : $$ABC = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{bmatrix}$$

$$\therefore \quad [S]^*[ABC] = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 2 & 2 & 1 \\ 10 & 4 & 1 \end{bmatrix}$$

∴ Now point are $A' = (0, 0)$

$B' = (2, 2)$

$C' = (10, 4)$

So, A is fixed also at (0, 0).

(b) reduce to half its size :

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Given that : $$ABC = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{bmatrix}$$

∴ $$[S]*[ABC] = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 1/2 & 1/2 & 1 \\ 5/2 & 1 & 1 \end{bmatrix}$$

∴ New point after reduce of half its size are

$A' = (0, 0)$

$B' = \left(\frac{1}{2}, \frac{1}{2}\right)$

$C' = \left(\frac{5}{2}, 1\right)$

So, A' is fixed also at (0, 0)

Q.5.(a) Write and explain sutherland-Cohan algorithms for polygon clipping.

Ans. Sutherland-Cohen algorithm : The Cohen Sutherland line clipping algorithm quickly detects and dispenses with two common and trivial cases (visible and non visible). In

other words, this algorithm performs initial tests on a line to determine whether intersection calculations can be avoided.

The Cohen Sutherland algorithm uses a divide and conquer strategy. The line segment's end-points are tested to see if the line can with be trivially accepted or rejected. If the line cannot be trivially accepted or rejected, an intersection of the line with a window edge is determined and the trivial reject/accept test is repeated. This process is continued until the line is accepted. To perform the trivial acceptance and rejection tests, we extend the edges of the window to divide the plane of the window into the nine regions-the eight "outside" regions and the one "inside" region. Each of the nine regions associated with the window is assigned a 4- bit code to identify the region. Each end point of the line segment is then assigned th code of the region in which it lies.

Algorithm of Cohen Sutherland :

Step 1. Calcuate positions of both end points of line.

Step 2. Perform OR operation on both of these end -pionts

Step 3. If OR operation gives 0000

then

line is considered to be visible

else

perform AND operation on both end points

If And ≠ 0000

then line is invsible

else

And = 0000

line is considered clipped case.

Step 4. If line is clipped case, find intersection with boundries of window

$m = (y_2 - y_1)\ (x_2 - x_1)$

(a) If bit 1 is "1" line intersect with lift boundry of rectangle window

$y_3 = y_1 + m\ (x - X_1)$

where $X = X_{wmin}$

where x_{wmin} is min value of X co-ordinate of window

(b) If bit 2 is "1 line intersect with right boundry $y_3 = y_1 + m\ (X - X_1)$

where $X = X_{wmax}$

where X more is maximum value of X co-ordinate of window.

(c) If bit 3 is "1" line intersect with bottom boundry

$X_3 = X_1 + (y - y_1)/m$

where $y = y_{wmin}$

y_{wmin} is minimum value of Y co-ordinate of window.

(d) If bit 4 is "1" line intersect with top boundry

$X_3 = X_1 + (y - y_1)/m$

where $y = y_{wmax}$

y_{wmax} is maximum value of Y co-ordinate of window.

Step 5. Repeat step 1 to 3 until line is completely accepted or rejected.

Q.5.(b) Perform a 45 degree rotation of a triangle A (0, 0), B (1, 1), C(5, 2). about origin.

Ans. Triangle ABC in matrix form is given as

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 5 & 2 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Matrix of rotation is

$$R_{45^\circ} = \begin{pmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Now find the coordinates $A'B'C'$ of the rotated triangle ABC can be found as

$$\left[A'\ B'\ C'\right] = R_{45^\circ} \cdot \begin{pmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 5 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & \frac{3\sqrt{2}}{2} \\ 0 & \sqrt{2} & \frac{7\sqrt{2}}{2} \\ 1 & 1 & 1 \end{pmatrix}$$

Thus $A' = (0, 0)$

$B' = \left(0, \sqrt{2}\right)$

$C' = \left(\frac{3\sqrt{2}}{2}, \frac{7\sqrt{2}}{2}\right)$

Section-C

Q.6. What do you mean by projection ? Desccribe different types of projection with examples. What are the various projection anomalies.

Ans. Projection : Projection can be defined as a mapping of point $P(x, y, z)$ onto its image $P'(x', y', z')$ in the projection plane or view plane, which consitutes the display surface (see fig (a)). The mapping is determined by a projection line called the projector that passes through P and intersects the view plane. The intersection points is P'.

Parallel Projection : Parallel projection is a type of projection in which the centre of projection is situated at infinite distance such that the projectors (lines) are parallel to each other. The plane on which we are taking projection is called as a view plane. And a parallel projection is formed by extending parallel lines from each vertex of the object until they interesect this viewplane. The point of intersection is the projection of the vertex. We can then connect the

projected vertices by line segments which corresponds to the connections on the original object. Our objective is to represent a 3D object onto a 2D plane like our mointor and the simplest way is to discard the z -coordinates.

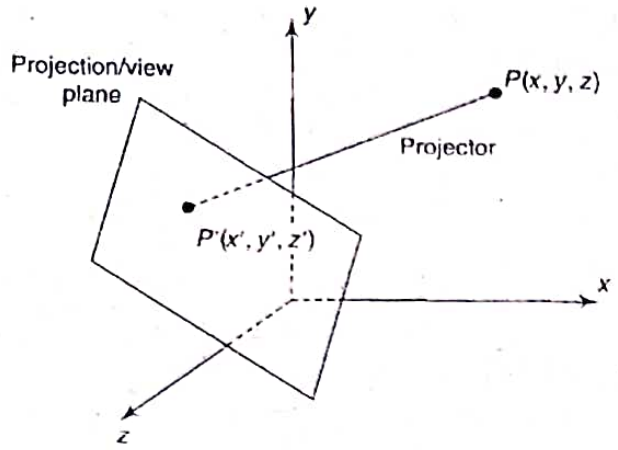


Fig.(a) : The Problem of Projection

Oblique Parallel Projection : A projection in which the angle between the projectors and the plane of projection is not equal to 90° is known as an oblique projection.

There are two common types of oblique parallel projections :

(a) Cavalier Parallel Projection

(b) Cabinet projection.

Orthographic Parallel Projection : A type of parallel projection in which centre of projection lies at infinity and the projectors are perpendicular to the view plane is known as orthographic parallel projection.

There are two basic types of orthographic parallel projections :

(a) Multiview orthographic parallel projection

(b) Axonometric orthographic parallel projection

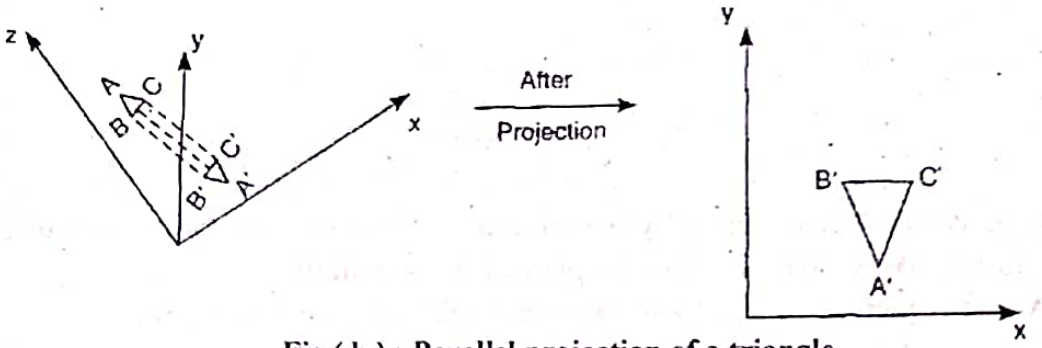


Fig.(b) : Parallel projection of a triangle

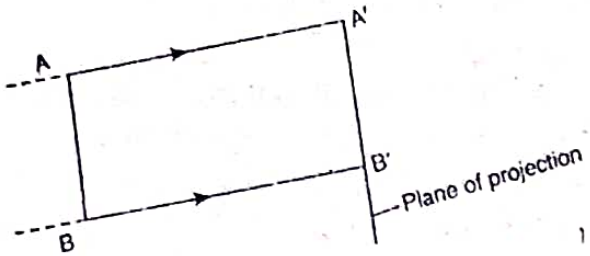


Fig.(c) : Oblique projection

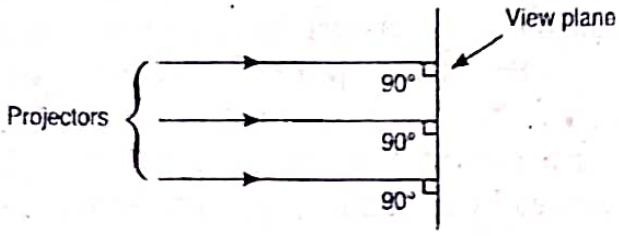


Fig.(d) : An orthographic projection

Perspective Projection : In this type of projection, that object is away (very far) from the viewer and appears to be smaller. This provides the viewer which a depth one, an indication of which portions of the image correspond to parts of the objects which are close or far away. Please note here that the farther the object is from the viewer, the smaller it appears. To obtain a perspective projection of a 3D-object, we transform points along projection lines which are not parallel to each other and converge to meet at a finite point known as the projection reference point or the center of projection. Also note that in real world, the centre of projection is a human eye. The projected view is obtained by calculating the intersection of the projection lines with the view plan This is shown in fig. (e).

Perspective projections are of three types:

(a) 1 - point perspective projection

(b) 2 - point perspective projection

(c) 3 - point perspective projection

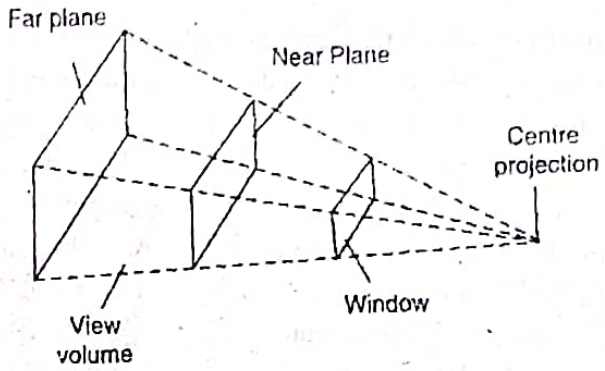


Fig.(e) : Perspective Projection

Problems with perspective projection : The process of constructing a perspective view introduces certain anomalies, which enhances realsim in terms of depth cues but also distorts actual sizes and shapes. There are four problems with perspective projections -

(1) *Perspective Foreshortening* : The farther an object is from the center of projection, the smaller it appears *i.e.*, its projected size becomes smaller.

(2) The perspective projection of objects behind the center of projection appear upside down and backward onto the view plane. This is view confusion.

(3) The relative dimensions of the objects are not preserved and hence the information is destroyed. This si topological distortion.

(4) Lines parallel to the view plane i.e., perpendicular to viewplane normal $\left(\overline{N}\right)$ are projected as pareallel lines. However, lines that are not parallel to view plane or receding parallel lines appear to meet at some point on the view plane called as vanishing points. The perspective projection of the edges will meet at a vanishing point, V_p. An example, is an illusion that the railway tracks appear to meet at a point horizon. Similarly, parallel lines appear to be bent as they converage to a vanishig point on the view plane.

However, lines parallel to principal axis (i.e. world coordinate axis) converage to a principal vanishing point. We name different types of perspective projections according to the finite number of vanishing points. A perspective projection can have 1 (minimum), 2 or 3 (maximum) principal vanishig points. For a 3D realistic display, number or $V_p = 3$, along 3 orthogonal axis.

Q.7.(a) What do you mean by hidden surface removal? Define Z -buffer algorithm for same.

Ans. Hidden Surface : When we look at the front side of a cube(say) of 6 sides, what we see is only the front side as shown in fig. This is so because we are looking from the front. Naturally, it depends on viewing position.

This is our hidden surface problem. A procedure that removes any such surfaces or lines which are not to be displayed in a 3D scene is called as hidden surface/line elimination or visible surface detection. A large number of algorithms exist for this problem.

Z - Buffer Algorithm (Depth Buffer Algorithm) : The z-buffer is one of the simplest algorithms of the hidden surface removal. The technique was originally proposed by Catmull and is an image space method. The buffer is a simple extension of the frame buffer idea. A frame buffer is used to store the attributes (intensity) of each pixel in image space. The z - buffer is a separte depth buffer used to store the z-coordinate or depth of every visible pixel in image space as illustrated in fig.(1).

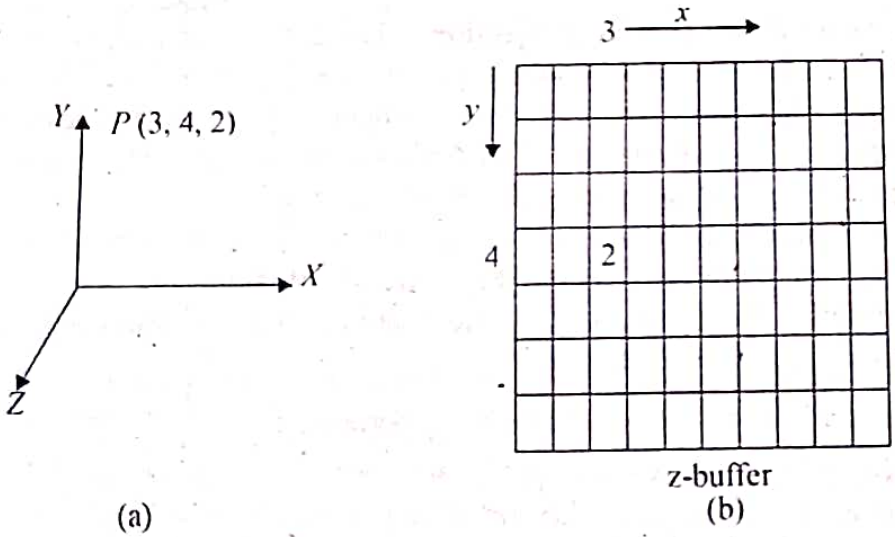


Fig.(1) : Representation of 3D point in z-buffer

The depth or z -value of a new pixel to be written to a frame buffer is compared to the depth or z -value of the pixel already stored in the z-buffer. If the comparison indicates that the z -value of the new pixel is greater than z-value already stored in the z-buffer, then z-buffer is up dated with the new z-value and intensity of new pixel is written in frame buffer, otherwise no action is taken.

Algorithm : Z-Buffer Algorithm for Hidden lines/Hidden Surface Removal

Step 1 : Set the frame buffer to the background intensity color.

frame _ buffer $(x, y) = I_{background}$

Step 2 : Set the z-buffer to the minimum value. $Z_{buffer}(x, y) = 0$

Step 3 : For each pixel (x, y) in the polygon, calculate the depth $z(x, y)$ at that pixel.

Step 4 : Compare the depth $z(x, y)$ with the value stored in the z-buffer at the location, $Z_{buffer}(x, y)$ to determine the visibility.

If $z(x, y) > Z_{buffer}(x,y)$, then write the polygon attributes (intensity, color, etc.) to the frame buffer at pixel location (x, y) and replace $Z_{buffer}(x,y)$ with $z(x, y)$ othwise no action is taken *i.e.*,

$$Z_{buffer}(x, y) = z(x, y)$$

frame _ buffer $(x, y) = I_{proj}(x, y)$

Where, $I_{proj}(x, y)$ is the projected intensity value for the surface at pixel position (x, y)

Step 5 : Repeat steps 3 to 4 for all polygons in arbitrary order.

Step 6 : Finally when all the polygons have been processed, the depth buffer contains depth value for the visible surfaces and the frame buffer contains the cooresponding intensity values for those surfaces.

Q.7.(b) Describe area sub division algorithms for hidden surface removal.

Ans. Algorithm :

(i) Initialize the area to be the whole screen.

(ii) Create the list of polygons by sorting them with their z-values of verticses. Do not include disjoint polygons in the first becasue they are not visible.

(iii) Find the relationship of each polygon.

(iv) Perform the visibility decision test

(a)If all the polygons are disjoint from the area, then fill area with background colour.

(b)If there is only one intersecting or only one contained polygon then first fill entire area with background colour and then fill the part of the polygon contained in the area with the colour of polygon.

(c)If there is a single surrounding polygon, but no intersecting or contained polygons, then fill the area with the colour of the surrounding polygon.

(d)If surrounding polygon is closer to the viewpoint than all other polygons, so that all other polygons are hidden by it, fill the area with the colour of the surrounding polygon.

(e)If the area is the pixel (x, y) and neither *a, b, c*, nor *d* applies computer the z coordinate at pixel (x, y) of all polygons in the list. The pixel is then set to colour of the polygon which is closer to the viewpoint.

(v) If none of the above tests are true then subdivide the area and go to step 2.

Unit -D

Q.8. Where has the term "spline" originated from ? Compare B-spline curve and Bezier curve. Obtain expression for Bezier curve and B-spline curve.

Ans. The term spline goes back to the long flexible strips of metal used by draftspersons to layout the surfaces of airplane, cars & ships. "Ducks" weights attached to the splines, were

used to pull the spline in various directions. The metal splines, unless severaly stressed, had second-order continuity. The mathematical equivalent of these strips, the natural cubic spline, is a C^0,C^1, & C^2 continuous cubic polynomial that interpolates the control points.

Bezier Spline Curve Vs B - Spline Curve :

(i) Bezier curves are special cases of B - spline curves.

(ii) In Bezier curve change in a portion of a curve causing changes to whole curve. In contrast B-spline has local control over curve. This property allows a designer or artist to edit one portion of a curve without causing changes to the remanining portion of the curve.

(iii) Unlike Bezier curve, B -Spline curves do not necessarily interpolate their first and last control points.

(iv) B -spline curves are more flexible and intuitive than Bezier curves with large number of control points.

(v) Curve designer has much flexibility in adjusting the curvature of B spline curve than Bezier curves.

(vi) B-splines can be designed with sharp bend even "corners" than Bezier Curves.

Bezier Curve : Given a set of $(n + 1)$ control points P_0, P_1, P_2, ...P_n, a parametric Bezier curve (Bernstein-Bezier curve) segment is a weighted sum of $n + 1$ control points P_0, P_1, P_2, ...P_n, that will fit to those points is mathematically defined by :

$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

where $B_{i,n}(t)$ are the Bezier blending function also known as Bernstein Basis functions (weights) defined as follows :

$$B_{i,n}(t) = C(n,i)\, t^i (1-t)^{n-i}$$

$$C(n, i) = \frac{n!}{i!(n-i)!}$$

Thus, a Bezier curve can be seen as a weighted average of all of its control points. Because all of the weights are positive, and because the weights sum to one, the Bezier curve is guaranteed to lie within the convex hull of its control points.

The Bezier curve of order $n + 1$ (degree n) has $n + 1$ control points. Following are the first three orders of Bezier curve definitions :

$$\text{Linear } P(t) = (1-t)P_0 + tP_1$$

$$\text{Quadratic } P(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2$$

$$\text{Cubic } P(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + t^3 P_3$$

$B(u)$ is a continuous function in 3 space defining the curve with N discrete control points P_k.$u = 0$ at the first control point ($k = 0$) and $u = 1$ at the last control point ($k = N$).

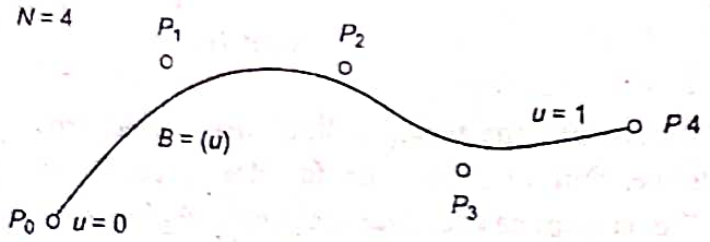


Fig. Bezier Curve

B – Spline Curves : Let a vector be known as the knot vector be defined $T = \{t_0, t_1, \ldots, t_m\}$.

where, T is a non-decreasing sequence with $t_i \in [0, 1]$, and define control points $P_0, \ldots P_n$. Define the degree as

$$p \equiv m - n - 1.$$

The "knots" $'P^{-1}, \ldots, 'm - p - 1$ are called inter knots.

Considering $P(t)$ as the parametric position vector of any point along the curve, then the general expression for a B-Spline curve of degree $(d - 1)$ is given by.

$$P(t) = \sum_{i=0}^{n} P_i B_{i,d}(t) \qquad t_{\min} \le t \le t_{\max} \text{ and } 2 \le d \le n+1$$

where, Pi (i = 0 to n) are the set of $n + 1$ control points and Bi, $d(t)$ are the $n + 1$ B-Spline blending functions (basis function) defined by the Cox-de Boor recursion formula as,

$$B_{i,d}(t) = \left\{\frac{t - t_i}{t_{i+d-1} - t_i}\right\} B_{i,d-1}(t) + \left\{\frac{t_{i+d} - t}{t_{i+d} - t_{i+1}}\right\} B_{i+1,d-1}(t)$$

where, $B_{i,1}(t) = 1$, if $t_i \le t \le t_{i-1}$

$B_{i,1} = 0$ otherwise

When there are some repeated knots, some of the denominators above may be zero; we adopt the convention that 0/0 = 0. Since, $N_{i,k}(u)$ will be identically zero when $u_{i,k} = u_i$, this means that any term with denominator equal to zero may be ignored.

Local control is an important feature of B-spline curves; it allows a designer or an artist to edit one portion of a curve without causing changes to distant parts of the curve.

Q.9. Explain Gouraud shading model and phong shading model.

Ans. Gouraud Shading : This technique was first presented by Henri Gouraud 1971. It is based upon principle that "Firstly illumination model is applied to find intensity of verices and then to apply linear interpolation to found the intensity values across the surface".

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

Step 1 : Compute the unit normal vector at each vertex N_V [refer to fig.(1)(a)(b)(c)]

Output N_A, N_B, N_C, N_D, N_E, N_F,

If plane equations are given then each polygon vertex normal is the average of the surface normal of the all polygons sharing that vertex i.e.

$$N_V = \frac{\sum_{i=1}^{m} N_i}{\left|\sum_{i=1}^{m} N_i\right|}$$

Alternatively, if equation of plane is not given then normal at vertex can be approximated by averageing the cross product of all edges that terminated at the vertex. Using the vertex v fig.(1)(b), the normal vector is

$$Nv = v \cdot v_1 \times v \cdot v_2 + v \cdot v_2 \times v \cdot v_3 + v \cdot v_3 \times v \cdot v_1$$

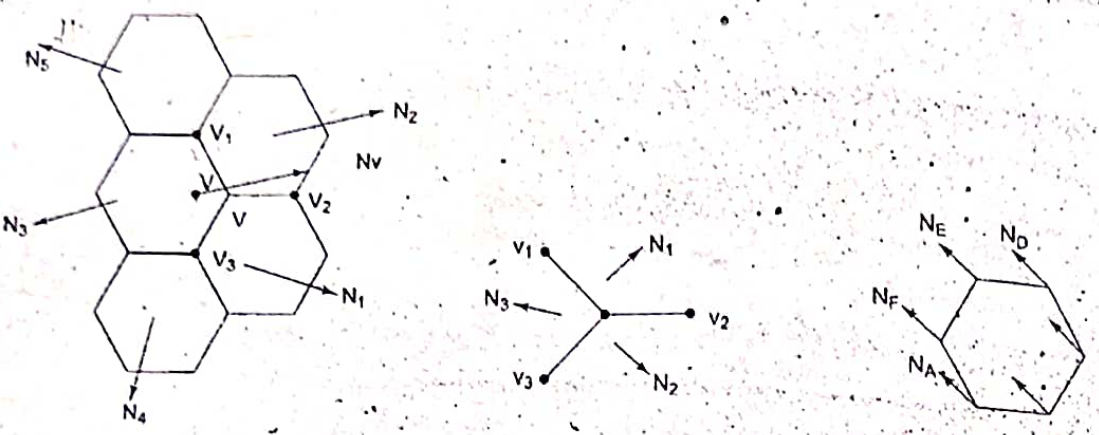


Fig. : (1)

Step 2 : Compute the intensity values at each polygon vertex using illumination model [refer to fig. (2)].

Output : I_A, I_B, I_C, I_D, I_E, I_F;

Step 3 : Compute the intensity of surface of the polygon using linear interpolation.

(a) Compute the intensities at points along polygon edges using linear interpolation, as depict in fig. (3), Q and R be any two points on edges AB and DC respectively of polygon $ABCDEF$.

Output : I_Q, I_R,

$$I_Q = u\, I_A + (1 - u)\, I_B$$

$$I_R = v\, I_C + (1 - v)\, I_D$$

$$u = \frac{AQ}{AB} = \frac{y_a - y_q}{y_a - y_b} \qquad v = \frac{CR}{CD} = \frac{y_r - y_c}{y_d - y_c}$$

(b) Compute the intensity at all interior points along the scan line using linear interpolation; let us assume an interior point P between Q and R.

Output : I_p,

$$I_p = w\, I_Q + (1 - w)\, I_R$$

$$w = \frac{PR}{QR} = \frac{x_r - x_p}{x_r - x_q}$$

Increment calculation are used on steps 3(a) and 3(b) to obtain successive intensities along each edge and to obtain successive intensities along a scan line.

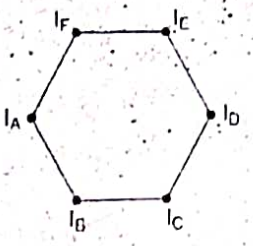


Fig. : (2)

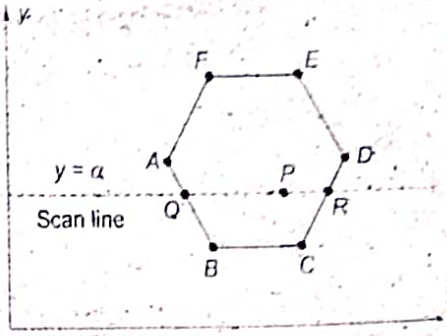


Fig. : (3)

Gouraud shading can be used with scan line algorithm. Like-wise flat shading linear interpolation may also produce bright or dark intensity streaks, called mach bands to be appear on non uniform shapes.

Phong Shading : A more accurate method for rendering a polygon surface is to interpolate normal vectors 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, interpolates the surface normal vector N, instead of the intensity.

By performing following steps, we can display polygon surface using Phong shading.

1. Determine the average unit normal over the surface of the polygon.
2. Linearly interpolate the vertex normals over the surface of the polygon.
3. Apply an illumination model along each scan line to determine projected pixel intensities for the surface points.

The first steps in the Phong shading is same as first step in the Gouraud shading. In the second step the vertex normals are linearly interpolated over the surface of the surface of the polygon.

As shown in fig., the normal vector N for the scan line intersection point along the edge between vertices 1 and 2 can be obtained by vertically interpolating between edge endpoint normals.

$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

Like, Gouraud shading, here also we can use incremental methods to evaluate normals between scan lines and along each individual scan line. Once the surface normals are evaluated the surface intensity at the point is determined by applying the illumination model.

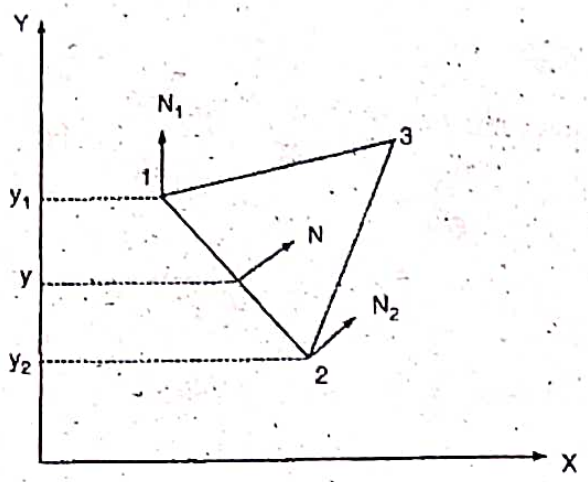


Fig : Calculation of interpolation of surface normals along a polygon edge.

Advantages : (i) It displays more realistic highlights on a surface.

(ii) It greatly reduces the Mach-band effect.

(iii) It gives more accurate results.

Disadvantages : (i) It requires more calculations and greatly increases the cost of shading steeply.

COMPUTER GRAPHICS

Dec - 2016
Paper Code:-CSE-303-F

Note : *Attempt five questions in all, selecting one question from each Section. Question No. 1 is compulsory. All questions carry equal marks.*

Q.1.(a) Define ambient, diffuse and specular reflection. (4)

Ans. Ambient reflection : The type of surface reflection that reacts to the intersity and color of the ambient light source only is called ambient reflection.

Diffuse Reflection : Diffuse reflection is characteristic of light reflected from a dull, matte surface.

Specular Reflection : Specular reflection is characteristics of light reflacted from a shiny surface, where a bright highlight appears from certain viewing directions. Mirror reflect incident light in one direction only and are considered parfect specular refelectors.

Q.1.(b) What is the difference between random scan an raster scan display? (4)

Ans. Random scan vs raster scan : Difference between Vector scan display and Raster Scan Display :

Sr.No.	Vector(random) scan display	Raster Scan Display
(1)	Vector display only draws lines and characters.	Raster display has ability to display areas fillled with solid colours or patterns.
(2)	Don't use interlacing.	Uses interlacing.
(3)	In vector scan display the beam is moved between the end points of the graphics primitives.	In raster scan display the beam is moved all over the screen one scan line at a time from top to bottom and then back to top.
(4)	Higher resolution.	Lower resolution.
(5)	More expensive.	Less expensive.
(6)	Uses monochrome or beam-penetration type.	Uses monochrome or shadow mask type.
(7)	Vector display draws a continuous and smooth lines.	Raster display can display mathematically smooth lines polygons and boundaries of curved primitives only by approximating them with pixel on the raster grid.
(8)	Editing is easy.	Editing is difficult.

(9)	Refresh rate depends directly on picture complexity.	Refresh rate independent of picture complexity.
(10)	Scan conversion is not required.	Graphics primitives are specifies in term of their end points and must be scan converted into their corresponding pixel in the frame buffer.

Q.1.(c) Explain the following with example : Scaling, Reflection. **(4)**

Ans. Scaling : Scaling is a transformation that changes the size or shape of an object. Scaling with respect to origin can be carried out by multiplying the co-ordinate values (x, y) of each vertex of a polygon, or each endpoint of a line by scaling factors S_x and S_y respectively to produce the coordinates (x', y').

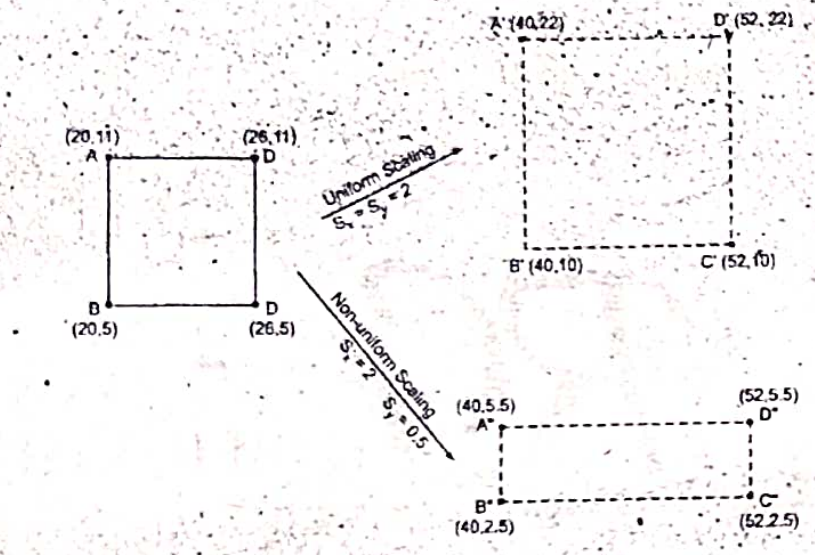


Fig. : (a)

The mathematical expression for pure scaling is

$$x' = S_x * x$$

$$y' = S_y * y$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

or symbolically $[X'] = [T][X]$

Reflection : A reflection is a transformation that produces a mirror image of an object. In 2D reflection we consider any line in 2D plane as the mirror. The reflection is also described as the rotation by 180°

The basic principles behind reflection transformation are :

(i) The image of an object is formed on the side opposite to where the object is lying w.r.t. mirror line.

(ii) The perpendicular distance of the object from the mirror line is same to the distance of the reflected image.

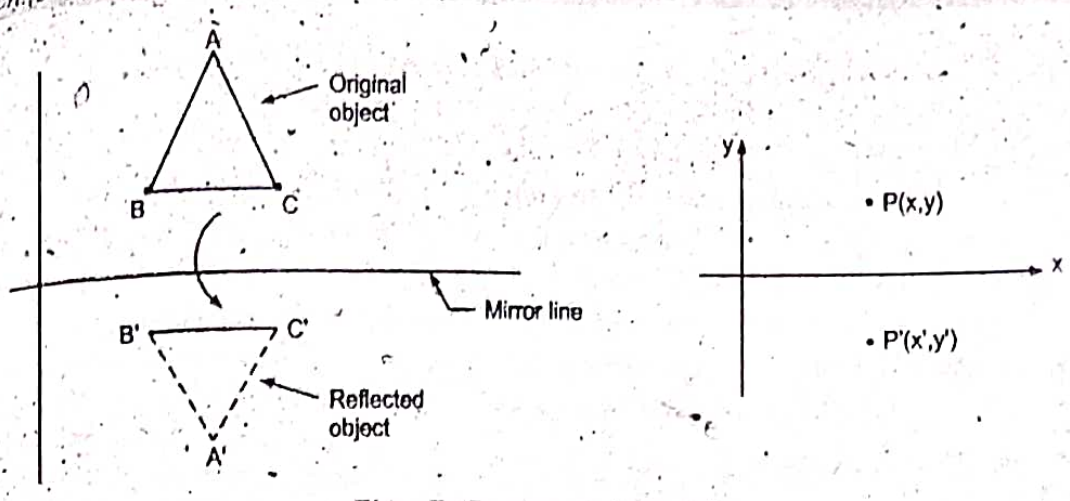


Fig. : Reflection about x-axis

For reflection about x-axis, x co-ordinate is not changed and sign of y-coordinate is changed. Thus if we reflect point (x, y) in the x-axis, we get $(x, -y)$

i.e., $x' = x$

$y' = -y$

So, the transformation matrix for reflection about x-axis or $y = 0$ axis is,

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

and the transformation is represented as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Q.1.(d) What is the importance of removal of Hidden Surface ?

Ans. When we look at the front side of a cube (say) of six sides, what we see is only the front side as shown in Fig. This is so because we are looking from the front. Naturally, it depends on viewing position.

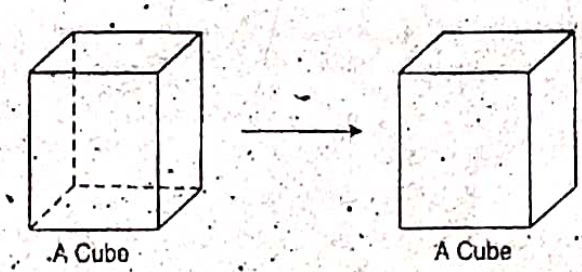


Fig. : Hidden surface removed

This is our hidden surface problem. A procedure that removes any such surface or lines which are not to be displayed in a 3D scene is called as hidden surface/line elimination or visible surface detection. The surface that are blocked or hidden from view must be "Removed" in order to construct a realistic view the 3D scene.

Q.1.(e) Explain view port and clipping.

Ans. Viewport : A viewport is an area of the display device on which the window data is presented. A two-dimensional regular viewport is specified by giving the left, right, bottom and

top edges of a ractangel. Viewport values may be given in actual physical device coordinates when specified in actual physical device coordinates; they are frequently given using integers. Viewport coordinates may be normalized to some arbitrary range, for example, $0 \le x \le 1.0, 0 \le y \le 1.0$, and specified by floating point numbers.

The display screen is 2-dimensional. Sometimes, it is very difficult to view all the details of an object. The most common views required to represent the object details are : isometric view, orthographic view, front view, top view, left side view, right side view and sectional views. To accommodate the display of several views simultaneously the display screen is divided into viewports. A typical screen layout with four viewports is shown in fig.(1) and (2).

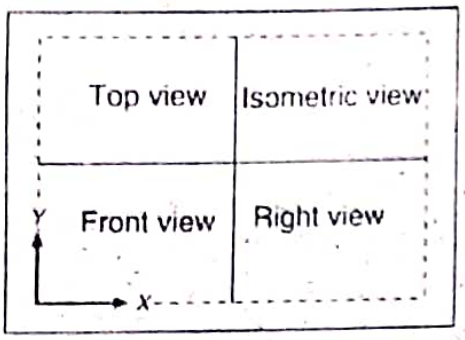


Fig.(1) : A typical screen layout

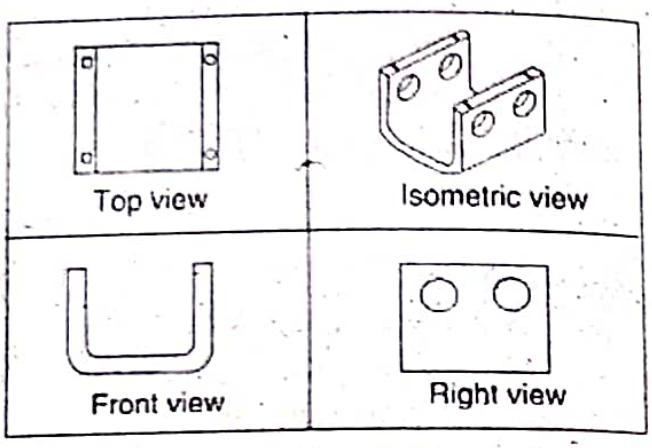


Fig.(2) : Illustration of different views of an object

Clipping : Generally, any procedure, that identifies those portions of a picture that are either inside or outside of a specified region of space is referred to as a clipping or simply Clipping. The region against which an object is to clipped is called a clip window.

Applications of clipping include extracting part of a defined scene for viewing identiflying visible surfaces in three dimensional views, antiatiasing line segements or objects boundaries ; creating objects using solid modelling procedurces, displaying a multiwindow environment, and drawing and painting operations that allow parts of a picture to be selected for copying, moving erasing or duplicating. Depending on the application, the clip window can be a general polygon or it can even have curved boundaries. Chipping algorthms can be applied in wordld coordinats, so that only the contents of the window interior are mapped to device coordinates.

Section - A

Q.2.(a) In what way interactive graphics differ from passive graphic ? Enumerate some application area of interactive graphics system. **(10)**

Ans. Non-interactive/Passive/Offline computer graphics : In this type of computer graphics, the user has no control over picture.

For example, a static website, a tv monitor etc. In both the cases the user has no control over the contents of monitor. In this the development takes place independently in off line mode. By sying that the user has no control we mean to say that the user can't move picture, translate picture, remote it or scale it as per his needs.

Interactive/Active/Online computer graphics : In this type of computer graphic,the user has control over picture. It is also called as online graphics. Dispalys are controlled by mouse, trackball, joystick etc. This is termed as interactive computer graphics because the user can interact with the machine as per his requirements. Videogames, dynamic websites, special effects in movies, cartoons are all making use of interactive computer graphics.

Q.2.(b) Explain midpoint circle drawing algorithms. **(10)**

Ans. We consider a circle in second octant from $x = 0$ to $x \leq y$. We use the same concept of eight-way symmetry to plot the entire circle.

Principle : We have 2 pixels –

Upper-pixel $(U) = (x_i + 1, y)$

Lower pixel $(L) = (x_i + 1, y_i - 1)$

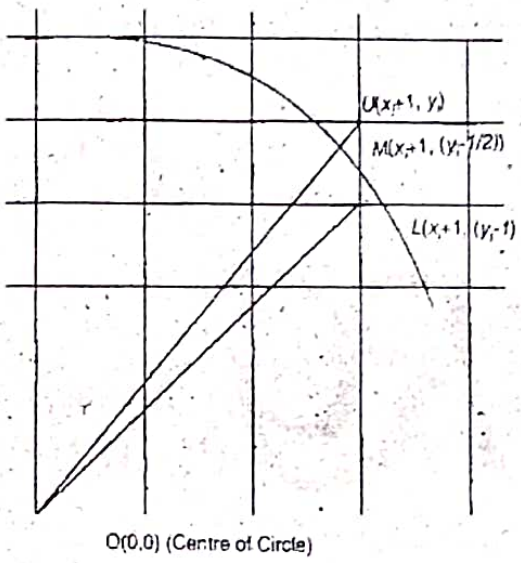


Fig. : Midpoint Circle Method

We need to select, which of these pixels is closer to the true circle and in this algorithm find the mid-point between the two pixels. This mid-point, M is given as follows :

$$M(x_{i-1}, (y_i + y_{i-1})/2)$$

i.e., the mid point of L and U.

Now, it depends on the Mid-point, which pixel to select. Like,

Mid-point is	Plot
On the circle	Either U or L pixel
Outside the circle	Plot U.
Inside the circle	Plot L.

where U and L are the Upper and Lower pixel respectively.

To apply the midpoint method, we define a circle funcion :

$$F(x, y) = x^2 + y^2 - r^2 = d \quad \ldots(1)$$

Any point (x, y) on the boundary of the circle with radius r satisfies the equation $F(x, y) = 0$. Now,

$$d = F(x, y) = x^2 + y^2 - r^2 \begin{cases} = 0 & \text{if point } (x,y) \text{ lies on circle} \\ > 0 & \text{if point } (x,y) \text{ lies outside circle} \\ < 0 & \text{if point } (x,y) \text{ lies inside circle} \end{cases}$$

The initial decision parameter is obtained by evaluating the circle function at the starting pixel position $(x_0, y_0) = (0, r)$. The next mid point lies on $1(1, r - 1/2)$

$$\therefore d = F(1, r - 1/2) = 1^2 + (r - 1/2)^2 - r^2 \quad ...(2)$$

$$= 5/4 - r$$

Now, for our mid point (M) at

$(x_i + 1, y_i - \frac{1}{2})$, the decision parameter is

$$d_i = f(x_i + 1, y_i - \frac{1}{2})$$

$$= (x_i + 1)^2 + (y_i - \frac{1}{2})^2 - r^2 \quad ...(3)$$

Here, if d_i is negative, the mid point is inside the circle and we choose the pixel L.

But if d_i is positive (or = 0), the mid-point is outside the circle or on the circle and we choose pixel U.

Similarly, the decision parameter for the next step is :

$$d_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - \frac{1}{2})^2 - r^2 \quad ...(4)$$

$x_{i+1} = x_i + 1$, so we have

From equations ...(3) and (4) we get :

$$d_{i+1} - d_i = [(x_{+1}) + 1]^2 - (x_i + 1)^2 + \left[y_{i+1} - \frac{1}{2}\right]^2 - \left[y_i - \frac{1}{2}\right]^2$$

$$\therefore \quad d_{i+1} = d_i + 2(x_i + 1) + 1 + (y_{+1}^2 - y_i^2) - (y_{i+1} - y_i) \quad ...(5)$$

If pixel L is chosen ($d_i < 0$) then we have

$$y_{i+1} = y_i$$

On the other hand, if pixel U is chosen ($d_i \geq 0$) then we have

$$y_{i+1} = y_i - 1$$

So, we say,

$$d_{i+1} = \begin{cases} d_i + 2(x_i + 1) + 1 & \text{if } d_i < 0 \\ d_i + 2(x_i - 1) + 1 - 2(y_i - 1) & \text{if } d_i \geq 0 \end{cases}$$

Further simplifying these in terms of (x_i, y_i) we get :

$$d_{i+1} = \begin{cases} d_i + 2x_i + 3) & \text{if } d_i < 0 \\ d_i + 2(x_i - y_i) + 5 & \text{if } d_i \geq 0 \end{cases}$$

Or, in terms of (x_{i+1}, y_{i+1}) we get :

$$d_{i+1} = \begin{cases} d_i + 2x_{i+1} + 1 & \text{if } d_i < 0 \\ d_i + 2(x_{i+1} - y_{i+1}) + 1 & \text{if } d_i \geq 0 \end{cases}$$

If radius r is specified as an integer since all increments are integers, then we can simply round d_0 to $(1 - r)$ i.e.

$$d_0 = 1 - r$$

Let us write its algorithm now :

Step 1 : Read (h, k) as the centre of circle and r as its radius.

Step 2 : Initialize $x = 0$

$y = r$

$d = 1 - r$

Step 3 : While $(x \leq y)$

// Plot 8-points

plot pixel $(x + h, y + k)$ // Each point

plot pixel $(-x + h, y + k)$ // Corresponds to

plot pixel $(x + h, -y + k)$ // One octant.

plot pixel $(-x + h, -y + k)$

plot pixel $(y + h, x + k)$

plot pixel $(-y + h, x + k)$

plot pixel $(y + h, -x + k)$

plot pixel $(-y + h, -x + k)$

if $(d < 0)$ then

$d = d + 2 * x + 3$ // upper pixel

else

$d = d + 2 * (x - y) + 5$ // lower pixel

$y = y - 1$

end if

$x = x + 1$

end while

Stop 4 : Stop

Q.3.(a) Write and explain the boundary fill algorithms. **(10)**

Ans. Refer Q3(b) of paper Dec. 2015.

Q.3.(b) Explain Bresenham's line drawing algorithm **(10)**

Ans. Bresenham's line algorithm uses only integer addition and subtraction and multiplication by 2, and we know that the compute can perform the operations of integer addition and subtratction very rapidly. The computer is also time-efficient when performing integer multiplication by power 2.

The basic principle of Bresenham's line algorithm is to find the optimum raster locations to represent the straight lines. To accomplish this the algorithm always increments either x y by one unit depending on the slope of line. Once this is done, then increment in other variable is found on the basis of the distance between the actual line location and the nearest pixel. The distance is called decision variable or the error term.

Bresenham's Algorithm :

(1) Read the line end point (x_1, y_1) and (x_2, y_2) such that they are not equal.

(2) $\Delta x = | x_2 - x_1)$

$\Delta y = | y_2 - y_1)$

(3) Initialize starting point $x = x_1$.

$y = y_1$

(4) $S_1 = Sign(x_2 - x_1)$

$S_2 = Sign(y_2 - y_1)$

[Sign function returns –1, 0, 1 depending on whether its agreement $< 0, = 0, > 0$ respectively].

(5) If $\Delta y > \Delta x$ then

exchange Δx & Δy

ex_change = 1

else

ex_change = 0

(6) $e = 2 * \Delta y - \Delta x$

(7) $i = 1$

(8) plot (x, y)

(9) while $(e \geq 0)$

{

if (ex_change = 1) then

$x = x + s_1$

else

$y = y + s_2$

$e = e - 2 * \Delta x$

}

(10) If ex_change = 1 then

$y = y + s_2$

else

$x = x + s_1$

$e = e + 2 * \Delta y$

(11) $i = i + 1$

(12) if $(i \leq \Delta x)$ then go to step (8)

(13) stop.

Section - B

Q.4. Describe the transformation used in magnification and reduction with respect to origin. Find the new coordinates of the triangle A (0, 0), B(1, 1), C (5, 2) after it has been (a) magnified to twice its size and (b) reduce to half its size. (20)

Ans. Refer Q.4 of paper Dec. 2015.

Q.5.(a) Explain Cyrus Back line clippling algorithm. (10)

Ans. Liang-Barsky Line Clipping algorithm (or Cyrus beck algorithm) : Liang and Barakey devised a more faster parametric line clipping algorithm called Liang Barsky line

clipping algorithm. They established a geometrical relationship between the window and the parametric line. As already discussed, the parametric equations of a line segment is in the form :

$$x = x_1 + u\,\Delta_x$$
$$y = y_1 + u\,\Delta_y \qquad \text{(where } 0 \le u \le 1\text{)}$$

and

For a point (x, y) inside the clipping window, we have

$xW_{min} \le x_1 + \Delta x., u \le xW_{max}$

$yW_{min} \le y_1 = \Delta y. u \le yW_{max}$

Rewriting these four inequalities as

$$p_k u \le q_k \quad \text{Where } k = 1,2,3,4$$

Here parameters p and q are defined as follows :

$p_1 = -\Delta x$, $q_1 = x_1 - xW_{min}$ (for left (1st) boundary)

$p_2 = \Delta x$, $q_2 = xW_{max} - x_1$ (for right (2nd) boundary)

$p_3 = \Delta y$, $q_3 = y_1 - yW_{min}$ (for bottom (3rd) boundary)

$p_4 = \Delta y$, $q_4 = yW_{max} - y_1$ (for top (4th) boundary)

It is algorithm is :

Step 1. The line intersection parameters are initialized to the values as $u_1 = 0$ and $u_2 = 1$.

Step 2. For each clipping boundary, the appropriate vlaues for p and q are calculated and used to find out whether the line can be rejected or the interstions parameters are to be adjusted.

Step 3. A value of r_K is calculated for each of these boundaries and the value of u_2 the minimum of the set consisting of 1 and the calculated r - values.

Step 4. If $u_1 > u_2$, the line is completely outside the clip window and it can be rejected. Else.

The end-points of the clipped line are calculated from the two values of parameter, u.

Step 5. When $p < 0$, the parameter r is used to update u_1.

When $p > 0$, the paremeter r is used to update u_2.

Step 6. If updation of u_1 or u_2 results in $u_1 > u_2$, we reject the line. Else.

We update the appropriate u parameter only if the new value results in shortening of the line.

Step 7. When $p = 0$ and $q < 0$, we discard the line since it is parallel to and outside of this boundary.

Step 8. If the line has not rejected after all four values of p and q have been tested, then the end points of the clipped line are determined from values of u_1 and u_2.

Q.5.(b) Write and explain Sutherland-Cohan alogrithms for polygon clipping ? **(10)**

Ans. Refer Q5(a) of paper Dec. 2015.

Section - C

Q.6.(a) Explain Z-buffer algorithm. **(10)**

Ans. Z - Buffer Algorithm (Depth Buffer Algorithm) : The z-buffer is one of the simplest algorithms of the hidden surface removal. The technique was originally proposed by Catmull and is an image-space method. The buffer is a simple extension of the frame buffer idea. A frame buffer is used to store the attributes (intensity) of each pixel in image space. The z-buffer is a separte depth buffer used to store the z-coordinate or depth of every visible pixel in image space as illustrated in fig.(1).

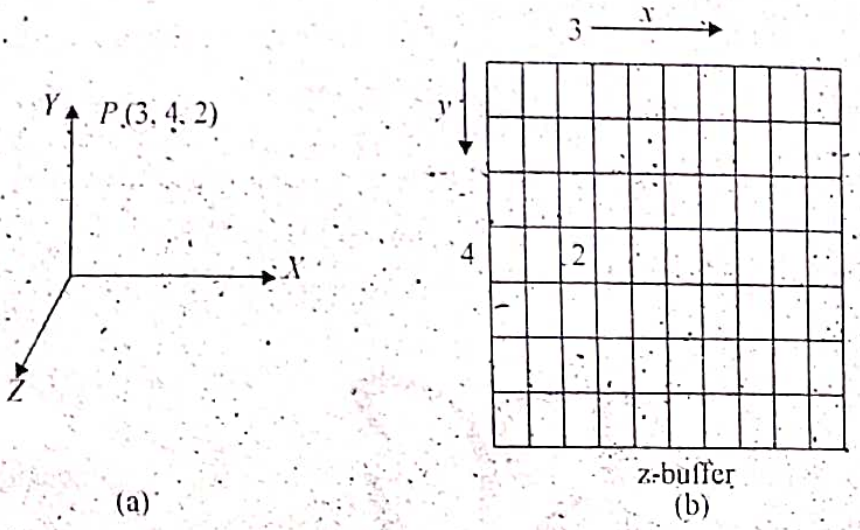


Fig.(1) : Representation of 3D point in z-buffer

The depth or z -value of a new pixel to be written to a frame buffer is compared to the depth or z -value of the pixel already stored in the z-buffer. If the comparison indicates that the z -value of the new pixel is greater than z-value already stored in the z-buffer, then z-buffer is updated with the new z-value and intensity of new pixel is written in frame buffer, otherwise no action is taken.

Algorithm : Z-Buffer Algorithm for Hidden lines/Hidden Surface Removal

Step 1 : Set the frame buffer to the background intensity color.

$$\text{frame\_buffer}\ (x, y) = I_{background}$$

Step 2 : Set the z-buffer to the minimum value. $Z_{buffer}\ (x, y) = 0$

Step 3 : For each pixel (x, y) in the polygon, calculate the depth $z(x, y)$ at that pixel.

Step 4 : Compare the depth $z(x, y)$ with the value stored in the z-buffer at the location, $Z_{buffer}\ (x, y)$ to determine the visibility.

If $z(x, y) > Z_{buffer}\ (x,y)$, then write the polygon attributes (intensity, color, etc.) to the frame buffer at pixel location (x, y) and replace $Z_{buffer}\ (x,y)$ with $z(x, y)$ othwise no action is taken *i.e.*,

$$Z_{buffer}\ (x, y) = z\ (x, y)$$

$$\text{frame\_buffer}\ (x, y) = I_{proj}\ (x, y)$$

Where, $I_{proj}\ (x, y)$ is the projected intensity value for the surface at pixel position (x, y)

Step 5 : Repeat steps 3 to 4 for all polygons in arbitrary order.

Step 6 : Finally when all the polygons have been processed, the depth buffer contains depth value for the visible surfaces and the frame buffer contains the cooresponding intensity values for those surfaces.

Q.6.(b) Describe scale line algorithms for hidden surface removal. (10)

Ans. A Scan-line Algorithm : This image space method for removing hidden surface is an extension of scan-line algorithm for filling polygon interiors.

Fig.(a) illustrates the scan-line method for locating visible portions of surfaces for pixel position along the line. The Active list for scan line 1 contains information from the edge table for edge *AB, BC, EH,* and *FG*. For positions along this scan line between edges *AB* and *BC*, only the flag for surface S_1 is on. Therefore, no depth calculations are necessary, and intensity information for surface S_1 is entered from the polygon table into the refresh buffer. Similarly, between edges *EH* and *FG*, only the flag for surface S_2 is on. No other positions along scan line 1 intersect surfaces, so the intensity values in the other areas are set to the background intensity. The background intensity can be loaded throughout the buffer in an initialization routine.

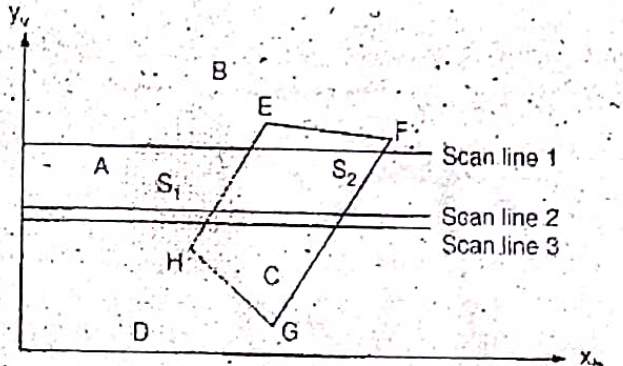


Fig : (a)

For scan lines 2 and 3 in Fig.(a) the active edge list contains edges *AD, EH, BC* and *FG*. Along scan line 2 from edge *AD* to edge *EH*, only the flag for surface S_2 is on. But between edges *EH* and *BC*, the flags for both surfaces are on. In this interval, depth calculations must be made using the plane coefficients for the two surfaces. For this example the depth of surface S_1 is assumed to be less than that of S_2, so intensities for surface S_1 are loaded into the refresh buffer until boundary *BC* is encountered. Then the flag for surface S_1 goes off, and intensities for surface S_2 are stored until edge *FG* is passed.

A scan line algorithm consists of two nested loop an x-scan loop nested with in a y-scan loop.

***y*-scan :** For each y value, say $y = \alpha$, intersect the polygons to be rendered with the scan plane $y = \alpha$. This scan plane is parallel to the xz plane and the resulting intersections are line segments in this plane.

***x*-scan :** (i) For each value of x, say $x = \beta$, intersect the line segments found above with the x-scan line $x = \beta$ lying on the y-scan plane. This intersection results in a set of points that lies on the x-scan line.

(ii) Sort these points with respect to their z-coordinates.

The point (x, y, z) with the smallest z value is visible, and the color of the polygon containing this point is the color set at the pixel corresponding to this point.

Section - C

Q.6.(a) Explain Z-buffer algorithm. (10)

Ans. Z - Buffer Algorithm (Depth Buffer Algorithm) : The z-buffer is one of the simplest algorithms of the hidden surface removal. The technique was originally proposed by Catmull and is an image-space method. The buffer is a simple extension of the frame buffer idea. A frame buffer is used to store the attributes (intensity) of each pixel in image space. The z-buffer is a separte depth buffer used to store the z-coordinate or depth of every visible pixel in image space as illustrated in fig.(1).

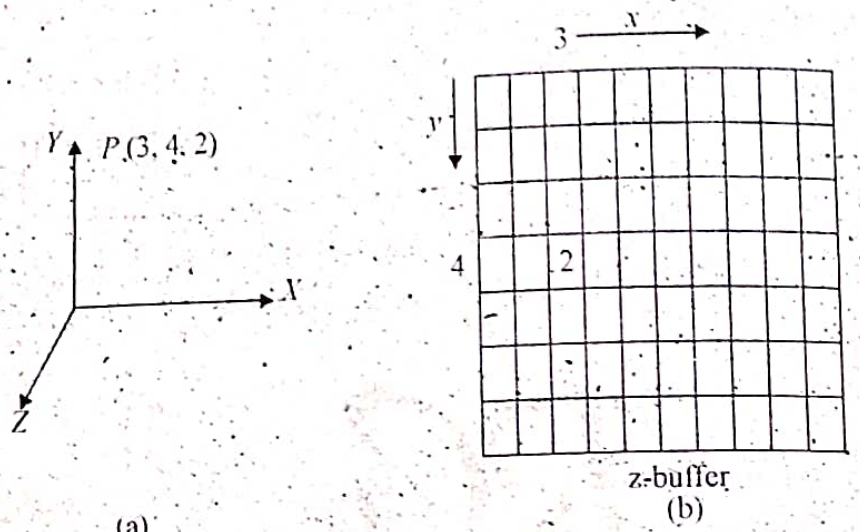


Fig.(1) : Representation of 3D point in z-buffer

The depth or z -value of a new pixel to be written to a frame buffer is compared to the depth or z -value of the pixel already stored in the z-buffer. If the comparison indicates that the z -value of the new pixel is greater than z-value already stored in the z-buffer, then z-buffer is updated with the new z-value and intensity of new pixel is written in frame buffer, otherwise no action is taken.

Algorithm : Z-Buffer Algorithm for Hidden lines/Hidden Surface Removal

Step 1 : Set the frame buffer to the background intensity color.

$$\text{frame\_buffer}(x, y) = I_{background}$$

Step 2 : Set the z-buffer to the minimum value. $Z_{buffer}(x, y) = 0$

Step 3 : For each pixel (x, y) in the polygon, calculate the depth $z(x, y)$ at that pixel.

Step 4 : Compare the depth $z(x, y)$ with the value stored in the z-buffer at the location. $Z_{buffer}(x, y)$ to determine the visibility.

If $z(x, y) > Z_{buffer}(x,y)$, then write the polygon attributes (intensity, color, etc.) to the frame buffer at pixel location (x, y) and replace $Z_{buffer}(x,y)$ with $z(x, y)$ othwise no action is taken *i.e.*,

$$Z_{buffer}(x, y) = z(x, y)$$

$$\text{frame\_buffer}(x, y) = I_{proj}(x, y)$$

Where, $I_{proj}(x, y)$ is the projected intensity value for the surface at pixel position (x, y)

Step 5 : Repeat steps 3 to 4 for all polygons in arbitrary order.

Step 6 : Finally when all the polygons have been processed, the depth buffer contains depth value for the visible surfaces and the frame buffer contains the cooresponding intensity values for those surfaces.

Q.6.(b) Describe scale line algorithms for hidden surface removal. (10)

Ans. A Scan-line Algorithm : This image space method for removing hidden surface is an extension of scan-line algorithm for filling polygon interiors.

Fig.(a) illustrates the scan-line method for locating visible portions of surfaces for pixel position along the line. The Active list for scan line 1 contains information from the edge table for edge *AB*, *BC*, *EH*, and *FG*. For positions along this scan line between edges *AB* and *BC*, only the flag for surface, S_1 is on. Therefore, no depth calculations are necessary, and intensity information for surface S_1 is entered from the polygon table into the refresh buffer. Similarly, between edges *EH* and *FG*, only the flag for surface S_2 is on. No other positions along scan line 1 intersect surfaces, so the intensity values in the other areas are set to the background intensity. The background intensity can be loaded throughout the buffer in an initialization routine.

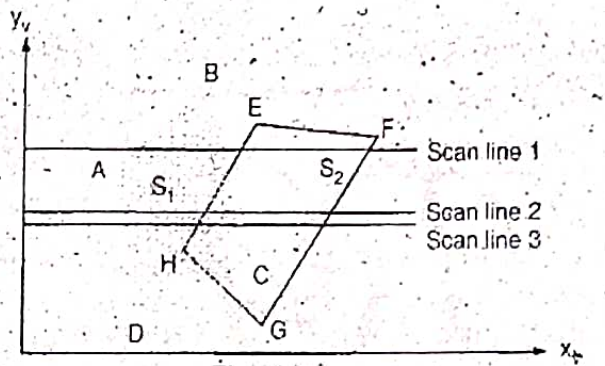


Fig : (a)

For scan lines 2 and 3 in Fig.(a) the active edge list contains edges *AD*, *EH*, *BC* and *FG*. Along scan line 2 from edge *AD* to edge *EH*, only the flag for surface S_2 is on. But between edges *EH* and *BC*, the flags for both surfaces are on. In this interval, depth calculations must be made using the plane coefficients for the two surfaces. For this example the depth of surface S_1 is assumed to be less than that of S_2, so intensities for surface S_1 are loaded into the refresh buffer until boundary *BC* is encountered. Then the flag for surface S_1 goes off, and intensities for surface S_2 are stored until edge *FG* is passed.

A scan line algorithm consists of two nested loop an x-scan loop nested with in a y-scan loop.

***y*-scan :** For each y value, say $y = \alpha$, intersect the polygons to be rendered with the scan plane $y = \alpha$. This scan plane is parallel to the xz plane and the resulting intersections are line segments in this plane.

***x*-scan :** (i) For each value of x, say $x = \beta$, intersect the line segments found above with the x-scan line $x = \beta$ lying on the y-scan plane. This intersection results in a set of points that lies on the x-scan line.

(ii) Sort these points with respect to their z-coordinates.

The point (x, y, z) with the smallest z value is visible, and the color of the polygon containing this point is the color set at the pixel corresponding to this point.

In order to reduce the amount of calculation in each scan-line loop, we try to take advantage of relationships and dependencies called coherences, between different elements that comprise a scene.

Scan line algorithm proceeds as follows :

(1) *Intialization* : (a) Initialize each screen pixel to a background colour.

(b, Set y to the smallest y_{min} value in the edge list.

(2) Repet steps (3) and (4) until no further processing can be performed.

(3) y-*scan Loop* : Active edges whose y_{min} is equal to y sort active edges in order of increasing x.

(4) x-*scan Loop* : Process from left-to-right, each active edge as follows :

(a) Invert the IN/OUT flag of the polygon list which contains the edge count the number of active polygons if this number is 1, only one polygon is visible. All the pixel values from this edge and up to the next edge are set to the colour of the polygon. If this number is greater than 1, determine the visible polygon by the smallest z value of each polygon. The pixels from this edge and up to the next edge are set to the colour of this polygon. If this number is 0, pixels from this edge and up to the next one are left unchanged.

(b) When the last active edge is processed, we then proceed as follows :

(i) Remove those edges for which the value of y_{max} equals the present scan-line value y. If no edges remain, the algorithm has finished.

(ii) For each remaining active edge in order replace x by $x + 1/m$. This is the edge intersection with the next scan line $y + 1$.

(iii) Increment y to $y + 1$, the next scan line and repeat step (3).

Any number of overlapping polygon surfaces can be processed with this scan-line method. Flags for the surface are set to indicate whether a position is inside or outside, and depthe calculations are performed when surfaces overlap. When these coherence methods are used, we need to be careful to keep track of which surface section is visible on each scan line. This works only if surfaces do not cut through or otherwise cyclically overlap each other.

If any kind of cyclic overlap is present in a scene, we can divide the surfaces to eliminate the overlaps. The dashed lines in the fig.(b) indicate where planes could be subdivided to form two distinct surfaces, so that the cyclic overlaps are eliminated.

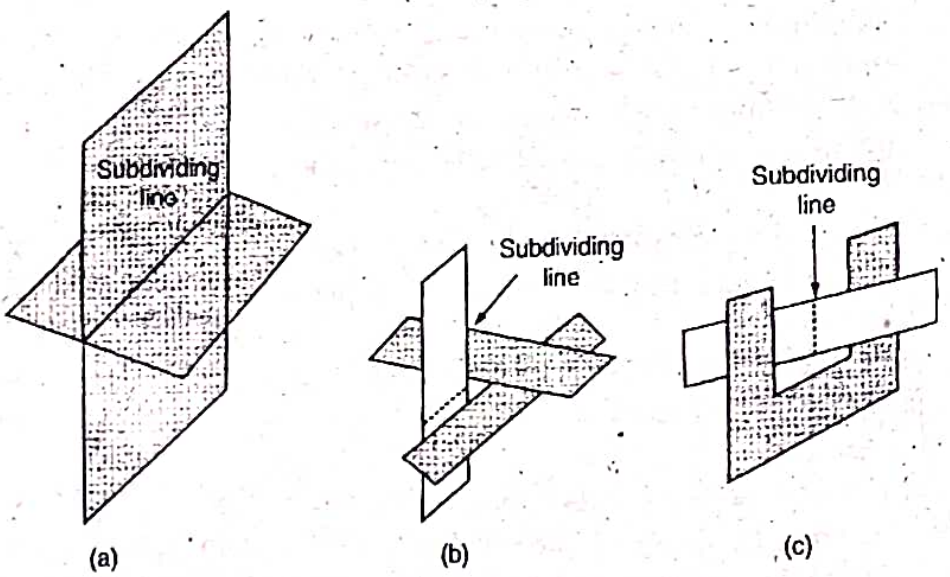


Fig : (b)

Q.7.(a) Explain the following with example : Translation, Scaling, Rotation and Composite transformation. (10)

Ans. Translation : Translation consists of a shift of the object parallel to itself in any direction in the (x, y) plane. Any such shift can be accomplished by a shift in x-direction plus a shift in y-direction. If the amount of x-shaft is called t_x and the amount of y-shaft t_y, the translation of the point (x, y) into the point(x', y') is expressed by the formulas.

$$x' = x + t_x$$

$$y' = y + t_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

To translate an object with multiple points, we just translate each point individually and connect them together.

Translation is a rigid-body transformation that moves objects without deformation. That is, every point on the object is translated by the same amount. Translatoin is the only transformation that is not in relation to a reference point. Its effect is independent of the original position of the object.

Scaling : Scaling is a transformation that changes the size or shape of an object. Scaling with respect to origin can be carried out by multiplying the co-ordinate values (x, y) of each vertex of a polygon, or each endpoint of a line by scaling factors S_x and S_y respectively to produce the coordinates (x', y').

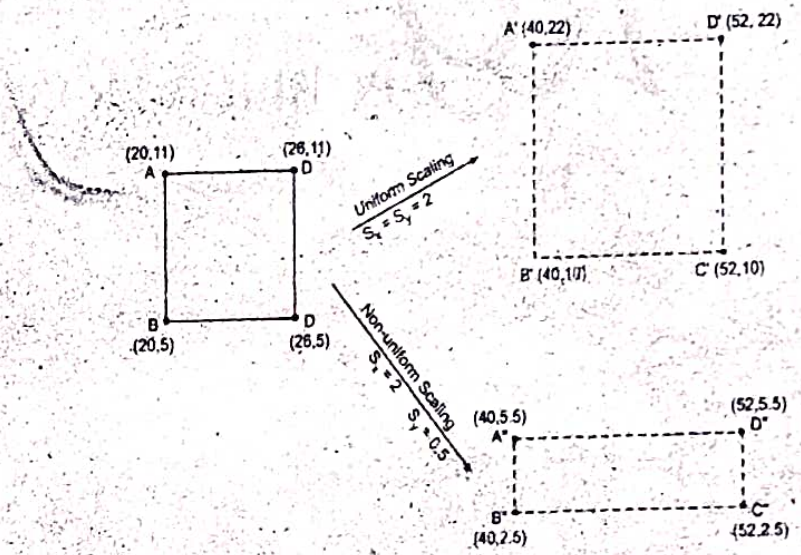


Fig. : (b)

The mathematical expression for pure scaling is

$$x' = S_x * x$$

$$y' = S_y * y$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

or symbolically $[X'] = [T][X]$

Rotation : A two-dimensional rotation is applied to an object by repositioning it along a circular path in the xy plane. Point can be rotated throught an angle about the origin. The sign of angle determines the direction of rotation. Positive values for the rotation angle defines counterclockwise rotation and negative values rotate object in clockwise directions.

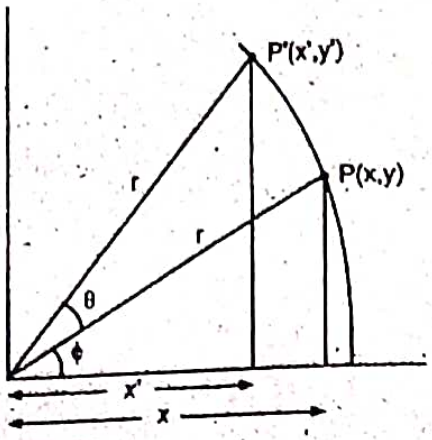


Fig.

Suppose rotation by θ transforms the point P(x, y) into P(x', y'). Because the rotations is about the origin, the distances from the origin to P and to P' is r are equal.

$$x = r \cos \phi$$
$$y = r \sin \phi$$

and $x' = r \cos (\theta + o) = r \cos \theta \cos \phi - r \sin \phi \sin \theta$

$$y' = r \sin (\theta + o) = r \sin \theta \cos \phi + r \sin \theta \cos \theta$$

Put the velue of x and y, we get

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

In matric from (row-major order)

$$[X'] = [X].[T] \text{ or } [x' \; y'] = [x \; y] [T]$$

Put the velue of x' and y' we get.

$$[x \cos \theta - y \sin \theta \quad \sin \theta + y \cos \theta] = [x \; y] \; [T]$$

Hence $$[T] = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Composite Transformations : To control the size of an object, we need to perform matrix operations on the position vector which defines the vertices. It is not necessary that we get the required orientation by applying single tranformation, it may require more than one transformation. Since matrix multiplication is non commutative, the order of application of the transformation is important.

We can set up a matrix for any sequence of transformations as a composite transformation matrix by calculating the matrix product of the individual transforamtions.

(i) If two successive translation are applied then

$$\begin{pmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{pmatrix}=\begin{pmatrix} 1 & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \\ 0 & 0 & 1 \end{pmatrix}$$

(ii) If two successive rotations θ_1 and θ_2 then

$$\begin{pmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(iii) If two successive scaling are applies then

$$\begin{pmatrix} S_{x1}.S_{x2} & 0 & 0 \\ 0 & S_{y1}.S_{y2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Q.7.(b) What do you mean by projection ? Explain different types of projection. **(10)**

Ans. Projection : Projection can be defined as a mapping of point $P(x, y, z)$ onto its image $P'(x', y', z')$ in the projection plane or view plane, which consitutes the display surface (see fig (a)). The mapping is determined by a projection line called the projector that passes through P and intersects the view plane. The intersection points is P'.

Parallel Projection : Parallel projection is a type of projection in which the centre of projection is situated at infinite distance such that the projectors (lines) are parallel to each other. The plane on which we are taking projection is called as a view plane. And a parallel projection is formed by extending parallel lines from each vertex of the object until they interesect this viewplane. The point of intersection is the projection of the vertex. We can then connect the projected vertices by line segments which corresponds to the connections on the original object. Our objective is to represent a 3D object onto a 2D plane like our mointor and the simplest way is to discard the z -coordinates.

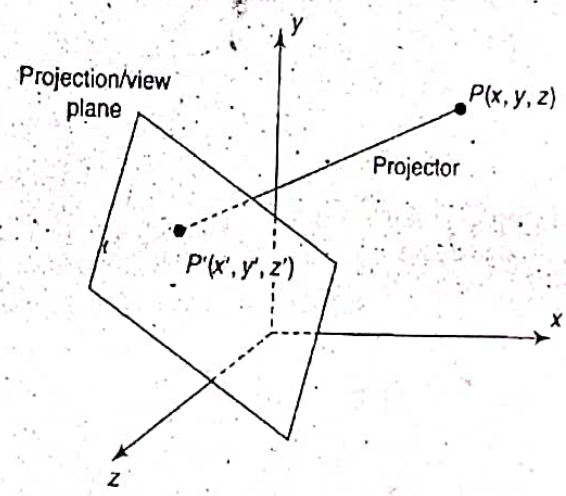


Fig.(a) : The Problem of Projection

Oblique Parallel Projection : A projection in which the angle between the projectors and the plane of projection is not equal to 90° is known as an oblique projection.

There are two common types of oblique parallel projections :

(a) Cavalier Parallel Projection

(b) Cabinet projection.

Orthographic Parallel Projection : A type of parallel projection in which centre of projection lies at infinity and the projectors are perpendicular to the view plane is known as orthographic parallel projection.

There are two basic types of orthographic parallel projections :

(a) Multiview orthographic parallel projection

(b) Axonometric orthographic parallel projection

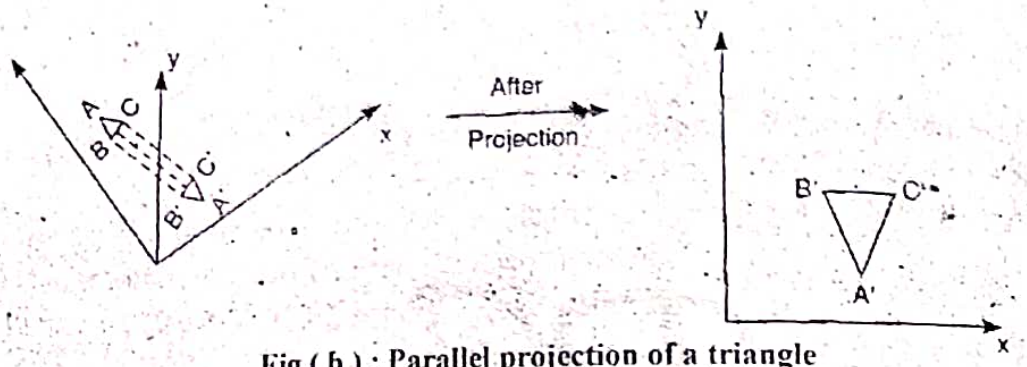


Fig.(b) : Parallel projection of a triangle

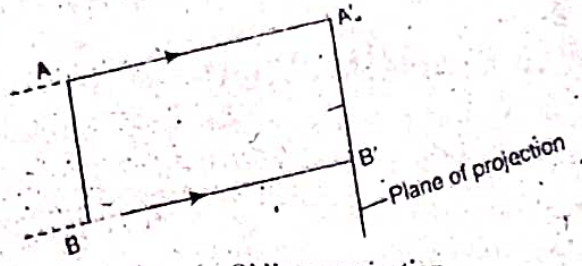


Fig.(c) : Oblique projection

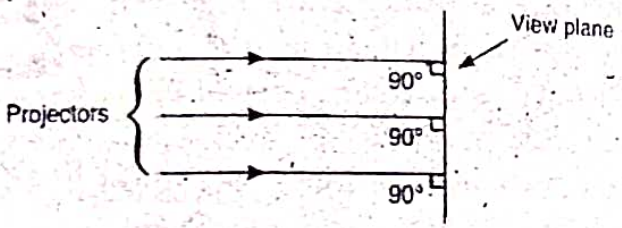


Fig.(d) : An orthographic projection

Perspective Projection : In this type of projection, that object is away (very far) from the viewer and appears to be smaller. This provides the viewer which a depth one, an indication of which portions of the image correspond to parts of the objects which are close or far away. Please note here that the farther the object is from the viewer, the smaller it appears. To obtain a perspective projection of a 3D-object, we transform points along projection lines which are not parallel to each other and converge to meet at a finite point known as the projection reference point or the center of projection. Also note that in real world, the centre of projection is a human eye. The projected view is obtained by calculating the intersection of the projection lines with the view plane. This is shown in fig. (e).

Perspective projections are of three types :

(a) 1 - point perspective projection

(b) 2 - point perspective projection

(c) 3 - point perspective projection

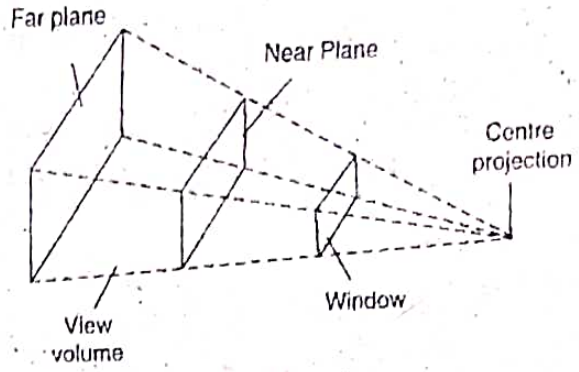


Fig.(e) : Perspective Projection

Section - D

Q.3.(a) Differentiate between unifrom B-Spline and non-unifrom B-Spline with suitable examples. (10)

Ans. Unifrom B-Spline : Those B - Spline in which difference between knot values is equal. If knot vector is defined as -

$$T = [u_0, u_1, \ldots u_m] \text{ then}$$

$$u_0 - u_1 = u_1 - u_2 = \ldots\ldots\ldots = u_{m-1} - u_m$$

$\therefore$ Blending function can be calculated as -

$$B_{i,p}(u) = B_{i+1,d}[u + \Delta u]$$
$$= B_{i+2,d}[u + 2\Delta u]$$

Non Uniform B -Spline : These curves have unequal differences between their knot values and multiple internal knot values.

For example $T = [0, 2, 2, 3, 4, 4, 4]$

Q.8.(b) What is Bezier curve ? Describe various property of Bezier Curve. (10)

Ans. Bezier Curve : Given a set of $(n + 1)$ control points P_0, P_1, P_2, ...P_n, a parametric Bezier curve (Bernstein-Bezier curve) segment is a weighted sum of $n + 1$ control points P_0, P_1, P_2, ...P_n, that will fit to those points is mathematically defined by :

$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

where $B_{i,n}(t)$ are the Bezier blending function also known as Bernstein Basis functions (weights) defined as follows :

$$B_{i,n}(t) = C(n,i)\, t^i (1-t)^{n-i}$$

$$C(n, i) = \frac{n!}{i!(n-i)!}$$

Thus, a Bezier curve can be seen as a weighted average of all of its control points. Because all of the weights are positive, and because the weights sum to one, the Bezier curve is guaranteed to lie within the convex hull of its control points.

The Bezier curve of order $n + 1$ (degree n) has $n + 1$ control points. Following are the first three orders of Bezier curve definitions :

$$\text{Linear } P(t) = (1 - t)P_0 + tP_1$$

Quadratic $P(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2$

Cubic $P(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + t^3 P_3$

$B(u)$ is a continuous function in 3 space defining the curve with N discrete control points P_k. $u = 0$ at the first control point ($k = 0$) and $u = 1$ at the last control point ($k = N$).

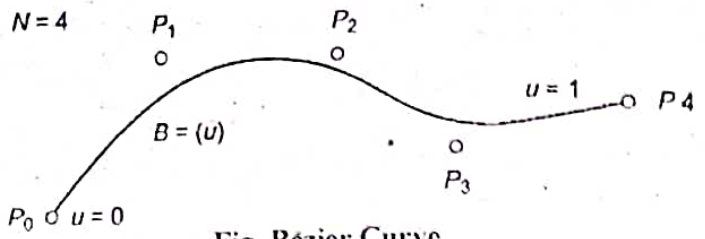


Fig. Bezier Curve

Properties of Bezier Curve :

(1) The basis functions are real.

(2) Bezier curve always passes thrugh the first and last control points i.e., curve has same and point as the guiding polygon.

(3) The degree of the polynomial defining the curve segment is one less then the number of defining polygon point. Therefore, for 4 control point, the degree of the polynomial is three, i.e., cubic polynomial.

(4) The curve generally follows the shape of the defining polygon.

(5) The direction of the tangent vector at the end points is the same as that of the vector determined by first and last segment.

(6) The curve lies entirely within the convex hull formed by four control points.

(7) The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control point.

(8) The curve exhibits the veriation diminishing property. This mean that the curve does not oscillate about any straight line more often than the defining polygon.

(9) The curve is invariant under an affine transformation.

Q.9.(a) What is an image ? Explain different types of image. (10)

Ans. Image : Image is composed of discrete pixels or picture elements. These pixels are arranged in a row and column fashion to from a rectangular picture area. Some times referred to as a raster. Clearly the total number of pixels in an image is a function of the size of the image and the number of pixels per unit length (e.g. inch) in the horizontal as well as the vertical direction. This number of pixels per unit length is referred to as the resolution of the image. Image size is given as the total number of pixels in the horizontal direction times the total number of pixels in the vertical direction.

Different types of image :

(1) Grayscale : Pixel values directly correspont to intensity. There may in addition be a table mapping each intensity to an optical density value. Grayscale images are usually 8, 12 or 16 bits/pixal, for 256, 4096, or 65536 gray levels, respectively. Since the monitor display 256 gray levels at a time, these gray levels are reduced to 256 in image by a grayscale intensity map, which can oe adjusted to highlight any specific intensity range. Flase-colour images are a subset

of grayscale images in which the computer displays an arbitrary colour instead of a shade of gray.

(2) **Indexed-colour :** Pixel value have no interinsic meaning and are merely indices into a colour map containing 256 colours in a randome sequence, which is totally different for each image. Indexed colour images are almost always 8 bits/pixal.

(3) **Colour :** Pixel values are a composite intensities of red, green and blue. The number of bits of each colour is diffrant for each standard image depth (i.e,.15, 16,24,32 and 48 bits/ pixel).

Q.9.(b) Define the term shading ? Differentiate between Gouraud Shading model and phong shading model. **(10)**

Ans. Shading : In computer graphics, shading refers to the process of altering the color of an object/surface/polygon in the 3D scene, based on its angle to lights and its distance from lights to create a photorealistic effect. Shading is performed during the rendering process by a program called a shader.

Difference between Gouraud Shading model and phong shading model :

Gouraud Shading : This intensity interpolartion scheme, developed by Henri Gouraud and generally referred to as Gouraud shading. The polygon surface is displayed by linearly interpolating intensity values across the surface. Intensity values for each polygon are matched with the values of adjacent polygons along the common edges. Thus, eliminates the intensity discontinuities that can occur in flat shading.

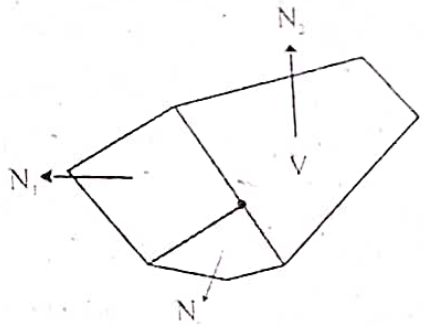


Fig.(a) : Calculation of normal vector at polygon vertex V

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

(i) Determine the average unit normal vector at each polygon vertex.

(ii) Apply an illumination model to each polygon vertex to determine the vertex intensity.

(iii) Lineraly interpolate the vertex intensities over the surface of the polygon.

As shown in the fig.(a), there are three surface normals N_1, N_2 and N_3 of polygon sharing vertex V is given as

$$N_v = \frac{N_1 + N_2 + N_3}{|N_1 + N_2 + N_3|}$$

Phong Shading : A more accurate method for rendering a polygon surface is to interpolate normal vectors 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, interpolates the surface normal vector N, instead of the intensity..

By performing following steps, we can display polygon surface using Phong shading.

(i) Determine the average unit normal vector at each polygon vertex.

(ii) Linearly intepolate the vertex normals over the surface of the polygon.

(iii) Apply an illumination model along each scan line to determine projected pixel intensities for the surface points.

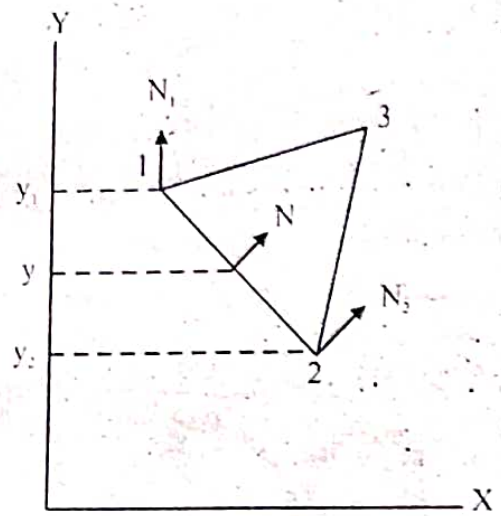


Fig.(b) : Calculation of interpolation of surface normals along a polygon edge.

As shown in the fig.(b), the normal vector N for the scan line intersection point along the edge between vertices 1 and 2 can be obtained by vertically interpolating between edge endpoint normals.

$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

Like. Gouraud shading, here also we can use incremental methods to evaluate normals between scan lines and along each individual scan line. Once the surface normals are evaluated the surface intensity at the point is determined by applying the illumination model.

COMPUTER GRAPHICS

Dec - 2017

Paper Code:-CSE-303-F

Note : *Attempt five questions in all, selecting one question from each Section. Question No. 1 is compulsory. All questions carry equal marks.*

Q. .. Explain the following : (4×5=20)

(a) Applications of computer graphics.

(b) Window to viewport mapping.

(c) Types of projections.

(d) Coefficient of reflection and halfway vector.

Ans.(a) Applications of Computer Graphics : Computer Graphics has been widely used, such as graphics presentation, paint systems, computeraided design (CAD), image-processing, simulation and virtual reality, entertainment, etc.

1. Computer-Aided Design (CAD) : It is one of the best and biggest application of computer graphics. This is particularly used in engineering applications such as building and other structural design and industrial design, design of manufacturing process, automobiles and aircraft, ships and spacecraft, very large scale-integrated (VLSI) chips, optical systems, and telephone and computer networks.

2. Computer Art : Computer graphics is used to produce picture that express a message and attrac. .ttention. In professions involving artist and in other business fields, such as fashion design, architecture, these applications of computer graphics are widely used because of effective speed of computer and easy modifiability capability.

3. Multimedia : Multimedia is the field concerned with the computer-controlled integration of text, graphics, drawings, still and moving images (video) animation, audio and any other media where every type of information can be represented, stored, transmitted and processed digitally.

4. Presentation Graphics : Presentation of data in pictorial form is always preferred since it is more understood than textual data. Presentation of graphics is widely used in posters, brochures, magazines, newspapers, training purpose, etc. This presentation may be sequential still or animated on computer monitor.

5. Internet : The word internet is derived from two words *Interconnection* and *Networks*. Also referred as the Net. Internet is a worldwide system of computer networks, that is, network of networks, which allows the participants to share information on those linked computers.

Internet is a much more broader concept than entertainment. The internet would be positively boring without graphics.

6. Graphical User Interface (GUI) : Graphical User Interface is related to learning and use of packages. It is not possible to learn packages in less times without graphic items present on the screen. Mostly, software developed today must have a Graphical User Interface and interactivity.

7. Simulation and Virtual Reality : The computer graphics helps here in simulating the views that the operator would see under various circumstances. Complex, mechanical, chemical and industrial processes can be simulated in action so that persons can be trained in their operation before they handle the actual equipment. Such a practice avoids wastages of cost and time.

Simulation has now reached up to such a length that those user follows wearing some special devices on their body such as on eyes, ears, and fingers.

8. Desktop Publishing (DTP) : DTP is related to the production of journals, printing newsletters, book publishing in small organization. This technology has expanded to such an extent that many book publishers, newspapers would not be able to deliver material without DTP.

9. Medical Applications : Computer graphics has now becomes a very important tool of diagnosis and treatment in the hands of doctors, particularly surgeons. Two-dimensional colorful images of cross-sections of human body or specific organs and limbs are produced. These 2D images are transformed to special investigated tools; a surgeon can rehearse his operation procedure on the computerized 3D image of the patient's part of body.

Ans.(b)Window-to view port mapping : In general the mapping of a part of world co-ordinate scene to device co-ordinates is referred as viewing transformation. Sometimes it is also called Window-viewport transformation or windowing transformation.

The viewing transformation is a process of 3 steps.

Step 1: The object together with its window is translated until the lower-left corner of the window is at the origin.

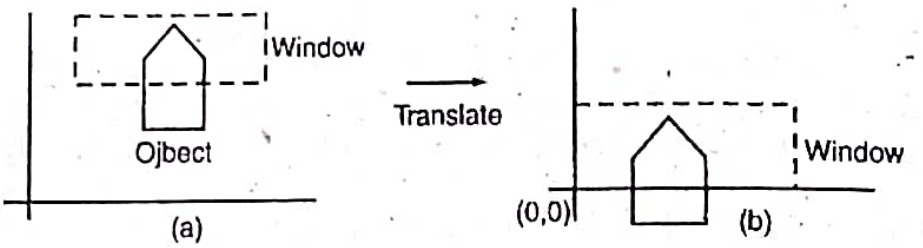


Fig. : (1)

Step 2: The object and the window are now scaled until the window has the dimension same as view port. In other words we are converting the object into image and window in view port.

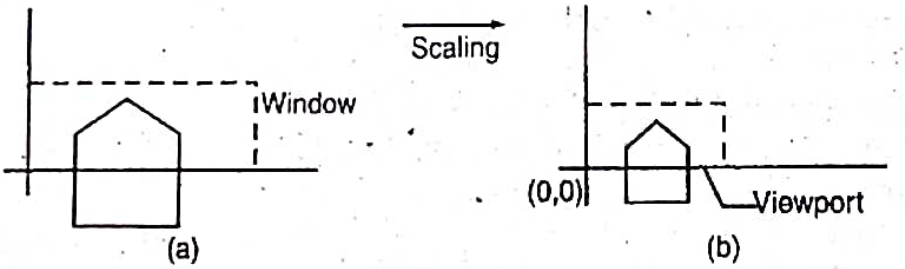


Fig. : (2)

Step 3: The final transformation step is another translation to move the viewport to its correct position on the screen:

So, the viewing transformation performs three steps :

(1) Translation

(2) Scaling

(3) Translation.

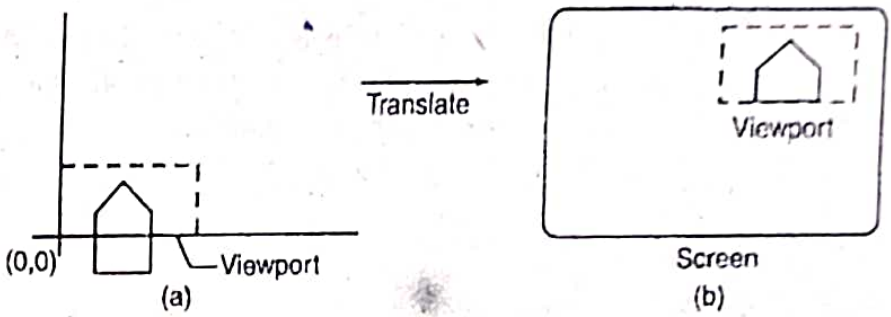


Fig. : (3)

Ans.(c)Types of projections :

Parallel Projection : Parallel projection is a type of projection in which the centre of projection is situated at infinite distance such that the projectors (lines) are parallel to each other. The plane on which we are taking projection is called as a view plane. And a parallel projection is formed by extending parallel lines from each vertex of the object until they interesect this viewplane.

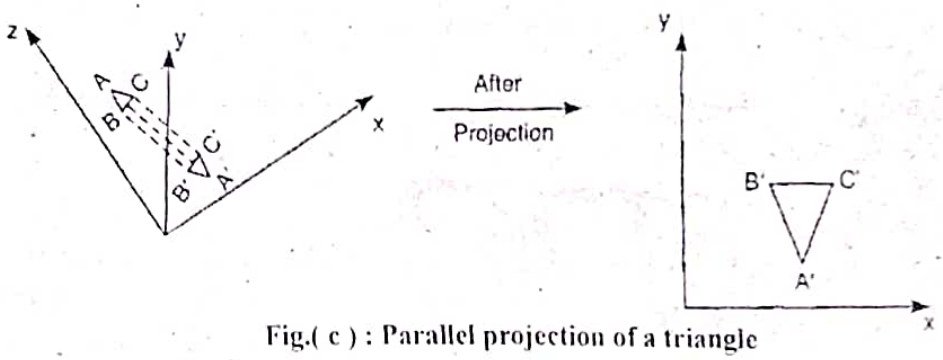


Fig.(c) : Parallel projection of a triangle

Perspective Projection : In this type of projection, that object is away (very far) from the viewer and appears to be smaller. This provides the viewer which a depth one, an indication of which portions of the image correspond to parts of the objects which are close or far away.

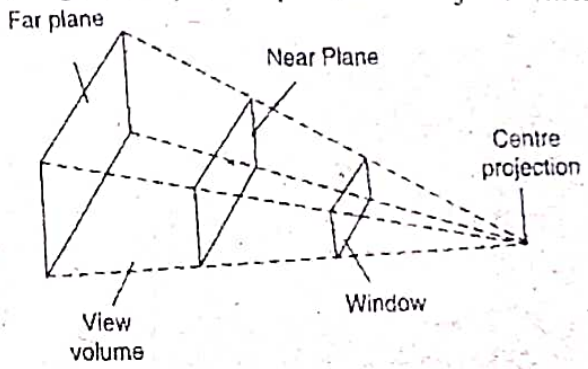


Fig.(f) : Perspective Projection

Ans.(d)Coefficient of reflection : The ratio of the light reflected from the surface to the total incoming light to the surface is called coefficient of reflection or the reflectivity. It is denoted by R. The value of R varies from 0 to 1. It is closer to 1 for white surface and closer to 0 for black surface. This is because white surface reflects nearly all incident light whereas black surface absorbs most of the incident light. Reflection coefficient for gray shades is in between 0 to 1. In case of colour object reflection coefficient are various for different colour surfaces.

Halfway Vector : More simplified way of formulation of Phong's illumination model is the use of halfway vector H. It is called halfway vector because its direction is halfway between the directions of the light source and the viewer as shown in the Fig.

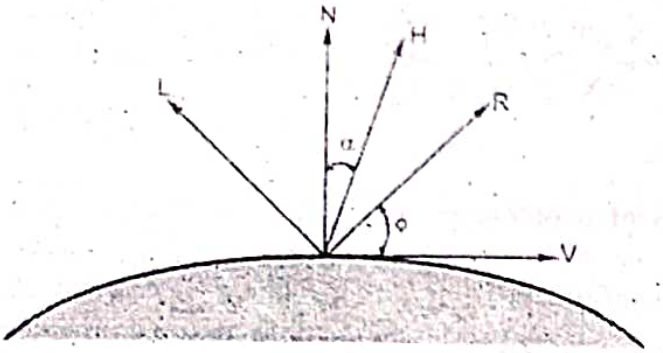


Fig. : Halfway Vector

Section – A

Q.2.(a) Write the step required to plot a line whose slope is between 0° and 45° using the slope-intercept equation. **(10)**

Ans. Refer Q.2.(a) of paper Dec. 2014.

Q.2.(b) Indicate which raster location would be chosen by Bresenham's algorithm when scan-converting a line from pixel coordinate (1, 1) to pixel coordinate (8, 5). **(10)**

Ans. We find that the starting values must be found.

$$dx = x_2 - x_1 = 8 - 1 = 7$$
$$dy = y_2 - y_1 = 5 - 1 = 4$$
$$\text{in } c_1 = 2dy = 2 \times 4 = 8$$
$$\text{in } c_2 = 2\,(dy - dx) = 2\,(4 - 7) = -6$$
$$D = \text{in } c_1 - dx$$
$$= 8 - 7 = 1$$

The followintg table shows the values :

d	x	y
1	1	1
$1 + \text{inc}_2 = -5$	2	2
$-5 + \text{inc}_2 = 3$	3	2
$3 + \text{inc}_2 = -3$	4	3
$-3 + \text{inc}_2 = 5$	5	3
$5 + \text{inc}_2 = -1$	6	4
$-1 + \text{inc}_1 = 7$	7	4
$7 + \text{inc}_2 = 1$	8	5

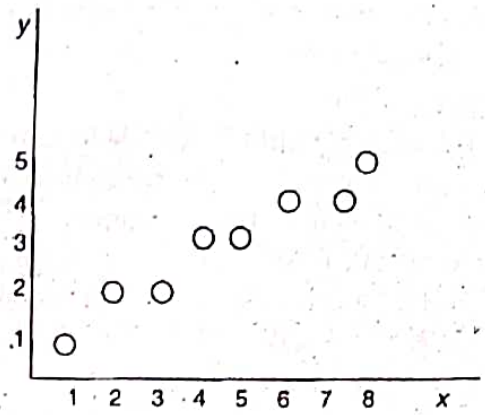


Fig. : Point Plotting

Q.3. Explain the architecture of Raster Scan Display. Give the logical organization of a Video Controller and explain its importance in Raster Scan display. **(20)**

Ans. Refer Q.3(a) of paper Dec. 2014.

Section – B

Q.4.(a) Perform a 60° rotation of triangle A(0,0), B(1, 1), C(5,2)

(i) About the origin and

(ii) About P (–1, –1). **(10)**

Ans. Refer Q.4(a) of paper Dec. 2014.

Q.4.(b) Write the general form of a shearing matrix with respect to a fixed point P(h, k). **(10)**

Ans. Refer Q.4(b) of paper Dec. 2014.

Q.5. Contrast the efficiency of clipping between Surtherland-Cohen and Mid-point algorithm. Describe Sutherland-Hodgeman algorithm for polygon clipping. Explain why this algorithm works for convex polygons. **(20)**

Ans. Refer Q.5 of paper Dec. 2014.

Section – C

Q.6.(a) Write 3D transformation matrix to find reflection of a point P (15, 25, 35) about plane z = 0. **(10)**

Ans. Refer Q.6(a) of paper Dec. 2014.

Q.6.(b) What is oblique projection ? Provide some examples of oblique projection. **(10)**

Ans. Refer Q.6(b) of paper Dec. 2014.

Q.7. Write notes on : **(20)**

(a) Z-buffer algorithm

(b) Geometric projections.

Ans. (a) Z - Buffer Algorithm (Depth Buffer Algorithm) : The z-buffer is one of the simplest algorithms of the hidden surface removal. The technique was originally proposed by Catmull and is an image space method. The buffer is a simple extension of the frame buffer idea. A frame buffer is used to store the attributes (intensity) of each pixel in image space. The z - buffer is a separte depth buffer used to store the z-coordinate or depth of every visible pixel in image space as illustrated in fig.(1).

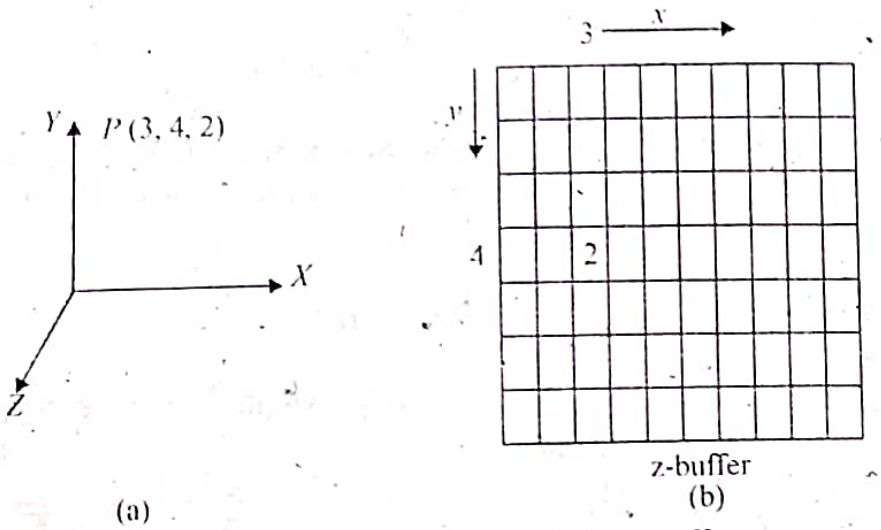


Fig.(1) : Representation of 3D point in z-buffer

The depth or z -value of a new pixel to be written to a frame buffer is compared to the depth or z -value of the pixel already stored in the z-buffer. If the comparison indicates that the z -value of the new pixel is greater than z-value already stored in the z-buffer, then z-buffer is up dated with the new z-value and intensity of new pixel is written in frame buffer, otherwise no action is taken.

Algorithm : Z-Buffer Algorithm for Hidden lines/Hidden Surface Removal

Step 1 : Set the frame buffer to the background intensity color.

$$\text{frame\_buffer}(x, y) = I_{background}$$

Step 2 : Set the z-buffer to the minimum value. $Z_{buffer}(x, y) = 0$

Step 3 : For each pixel (x, y) in the polygon, calculate the depth $z(x, y)$ at that pixel.

Step 4 : Compare the depth $z(x, y)$ with the value stored in the z-buffer at the location, $Z_{buffer}(x, y)$ to determine the visibility.

If $z(x, y) > Z_{buffer}(x,y)$, then write the polygon attributes (intensity, color, etc.) to the frame buffer at pixel location (x, y) and replace $Z_{buffer}(x,y)$ with $z(x, y)$ othwise no action is taken *i.e.*,

$$Z_{buffer}(x, y) = z(x, y)$$

$$\text{frame\_buffer}(x, y) = I_{proj}(x, y)$$

Where, $I_{proj}(x, y)$ is the projected intensity value for the surface at pixel position (x, y)

Step 5 : Repeat steps 3 to 4 for all polygons in arbitrary order.

Step 6 : Finally when all the polygons have been processed, the depth buffer contains depth value for the visible surfaces and the frame buffer contains the cooresponding intensity values for those surfaces.

Ans.(b) Geometric projection or simply projection of the objects are formed by the intersection of lines called projectors, on a plane called the projection plane. Projectors are lines from an arbitrary point called the center of projection, through each point in an object. If the center of projection is at a finite distance in three dimensional space, the result is a *perspective projection*. If the center of projection is at infinity, all the projectors are parallel and the result is a *parallel projection*. Thus, projections can be divided into two basic classes : perspective and parallel.

(*i*) Parallel Projection : In a parallel projection, coordinate positions are transformed to the view plane along parallel lines. These are linear transforms that are useful in blueprints to produce s ale drawings of three dimensional objects.

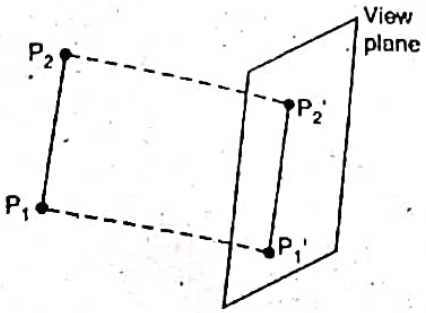


Fig.(1) : Parallel projection of an object to the view plane

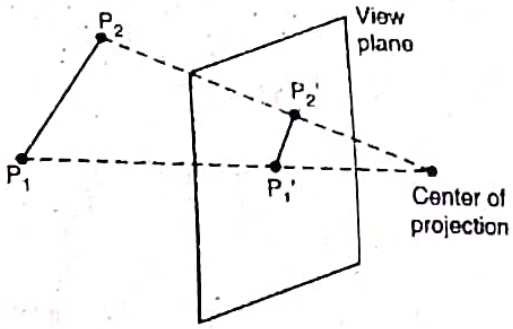


Fig.(2) : Perspective projection of an object to the view plane

(*ii*) Perspective Projection : For a perspective projection, object positions are transformed to the view plane along lines that converge to a point called to center of projection. The projected view of an object is determined by calculating the intersection of the projection lines with the view plane.

Section – D

Q.8.(a) Explain Bezier method of curve drawing. (10)

Ans. Refer Q.8(a) of paper Dec. 2014.

Q.8.(b) Describe methods of polygon shading. (10)

Ans. Refer Q.8(b) of paper Dec. 2014.

Q.9. Write notes on : *(20)*

(a) Bezier curve

(b) B-spline curve

(c) Fractals

Ans.(a) Bezier Curve : Given a set of $(n + 1)$ control points P_0, P_1, P_2, ...P_n parametric Bezier curve (Bernstein-Bezier curve) segment is a weighted sum of $n + 1$ control points P_0, P_1, P_2, ...P_n, that will fit to those points is mathematically defined by :

$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

where $B_{i,n}(t)$ are the Bezier blending function also known as Bernstein Basis functions (weights) defined as follows :

$$B_{i,n}(t) = C(n,i)\, t^i (1-t)^{n-i}$$

$$C(n, i) = \frac{n!}{i!(n-i)!}$$

Thus, a Bezier curve can be seen as a weighted average of all of its control points. Because all of the weights are positive, and because the weights sum to one, the Bezier curve is guaranteed to lie within the convex hull of its control points.

The Bezier curve of order $n + 1$ (degree n) has $n + 1$ control points. Following are the first three orders of Bezier curve definitions :

Linear $P(t) = (1 - t)P_0 + tP_1$

Quadratic $P(t) = (1 - t)^2 P_0 + 2(1 - t)tP_1 + t^2 P_2$

Cubic $P(t) = (1 - t)^3 P_0 + 3(1 - t)^2 tP_1 + 3(1 - t)t^2 P_2 + t^3 P_3$

$B(u)$ is a continuous function in 3 space defining the curve with N discrete control points P_k. $u = 0$ at the first control point $(k = 0)$ and $u = 1$ at the last control point $(k = N)$.

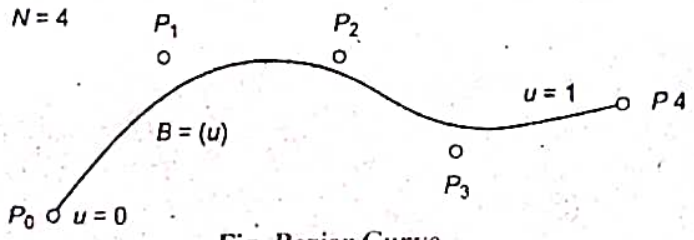


Fig. Bezier Curve

Properties of Bezier Curve :

(1) The basis functions are real.

(2) Bezier curve always passes thrugh the first and last control points i.e., curve has same and point as the guiding polygon.

(3) The degree of the polynomial defining the curve segment is one less then the number of defining polygon point. Therefore, for 4 control point, the degree of the polynomial is three, i.e., cubic polynomial.

(4) The curve generally follows the shape of the defining polygon.

(5) The direction of the tangent vector at the end points is the same as that of the vector determined by first and last segment.

(6) The curve lies entirely within the convex hull formed by four control points.

(7) The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control point.

(8) The curve exhibits the veriation diminishing property. This mean that the curve does not oscillate about any straight line more often than the defining polygon.

(9) The curve is invariant under an affine transformation.

Ans.(b) B–Spline curves : The name "B-spline" with the "B" standing for basis was defined by Sheonberg, in 1967. B-splines curves are widely used in computer aided design and manufacturing and are supported by Open GL. B-spline are powerfull tools for generating curves with many control points and provide advantages over Bizer curves. Furthemore, curve designer has much flexibility in adjusting the curvature of B-spline curve, and B-splines can be designed with sharp bend and even "corners".

B-spline curves are pulled towards the control point in much the same way that a Beizer curve is pulled towards its interior control points.

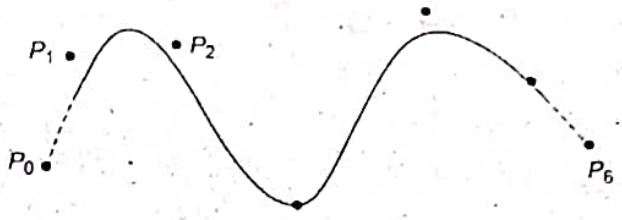


Fig. (a)

Let a vector be known as the knot vector be defined $T = \{t_0, t_1, \ldots, t_m\}$.

where, T is a non-decreasing sequence with $t_i \in [0, 1]$, and define control points $P_0, \ldots P_n$. Define the degree as

$$p \equiv m - n - 1.$$

The "knots" $'P^{-1}, \ldots, 'm - p - 1$ are called inter knots.

Considering $P(t)$ as the parametric position vector of any point along the curve, then the general expression for a B-Spline curve of degree $(d - 1)$ is given by,

$$P(t) = \sum_{i=0}^{n} P_i B_{i,d}(t) \qquad t_{min} \le t \le t_{max} \text{ and } 2 \le d \le n+1$$

where, Pi ($i = 0$ to n) are the set of $n + 1$ control points and $Bi, d(t)$ are the $n + 1$ B-Spline blending

functions (basis function) defined by the Cox-de Boor recursion formula as,

$$B_{i,d}(t) = \left\{\frac{t - t_i}{t_{i+d-1} - t_i}\right\} B_{i,d-1}(t) + \left\{\frac{t_{i+d} - t_i}{t_{i+d} - t_{i+1}}\right\} B_{i+1,d-1}(t)$$

where, $B_{i,1}(t) = 1$, if $t_i \le t \le t_{i+1}$

$B_{i,1} = 0$ otherwise

When there are some repeated knots, some of the denominators above may be zero; we adopt the convention that $0/0 = 0$. Since, $N_{i,k}(u)$ will be identically zero when $u_{i+k} = u_i$, this means that any term with denominator equal to zero may be ignored.

Local control is an important feature of B-spline curves; it allows a designer or an artist to edit one portion of a curve without causing changes to distant parts of the curve.

Properties of B-Splines Curves : (i) The polynomial curve has degree $(d - 1)$ and C^{d-2} continuity over the range of u.

(ii) For $(n + 1)$ control points, the curve is described with $n + 1$ blending functions.

(iii) Each blending function $B_{k,d}$ is defined over the d subintervals of the total range of u, starting of knot value u_k.

(iv) The range of parameter u is divided into $n + d$ subintervals by the $n + d + 1$ values specified in the knot vector.

(v) With knot values labeled as $[u_0, u_1, u_2, \ldots\ldots u_n]$ the resulting B-spline curve is defined only in the interval from knot value u_{d-1} up to knot value u_{n+1}.

(vi) Each section of the spline curve is influenced by control points.

(vii) Any one control point can affect the shape of at most d curve sections.

(viii) A B-spline with no internal knot is a Bezier curve.

Ans.(c) Fractals : According to Beroit Mandelbrot of IBM Research Lab, a fractal is "a rough or fragmented reduced copy of the whole". The term fractal is actually associated with randomly generated curves and surfaces that exhibit a degree of self similarity. They are used to provide naturalistic shapes for respresenting object like coast lines, ruggesd mountain's grass, five trees, rivers etc. These rough, jagged and random surfaces are called as *fractals* Mandelbroat calls various forms of self similar curves as fractals which is a short form of 'fractional dimention'. Please note that a line is 1D, a plane is 2D but a curve of infinite length that fits into a finite region of the plane must have a dimension somewhere between 1 and 2. Mandbort devised a method for computing the fractional dimension for such curves. Also note that a fractalization process can be used to generate shapes that resemble a tree.

A fractal object is generated by respeatedly applying a specified transformation function to points with in a region of space. If $P_0 = (x_0, y_0, z_0)$ is a selected initial point, each iteration of a transformation function, F, generates succesive levels of details with the calculations.

$$P_1 = F(P_0)$$
$$P_2 = F(P_1)$$
$$P_3 = F(P_2)$$

The transformation function may be defined in terms of geometric transformations (Scaling, translation, rotation) or it can be set up with non-linear coordinate transformations and decision parameters. Fractal objects contain infinite detail and so we apply the transformation function, a finite number of times. Therefore, the objects that we display actually has finite dimensions. Basically, we talk of two measures of object's dimension.

(*a*) Topological dimension

(*b*) Fractal dimension

Basic properties of Fractals : A fractal object has some basic properties :

(i) Fractals have infinite details at every region which can be seen on successive magnification.

(ii) There is Self Similarity between the object parts and the overall features of the parent object. Please note that there are many figures which are not fractals but they are self similar. Some fractals are also known as *well known fractals* like the *Koch Curve* shown below

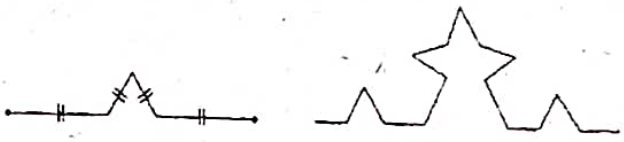


Fig. : Well known Fractals

Types of Fractals : We can classify fractals are follows :

(*a*) Self similar fractals

(*b*) Self affine fractals

(*c*) Invariant fractals

Self similar fractals have parts that are scaled down version of the entire object. Starting with an initial shape, we construct the object subparts subparts by applying a scaling parameter, S, to the overall shape.

Self affine fractals have parts that are formed with different scaling paraments, S_1, S_2, S_3 in different coodinate directions. And we can also include random variations to obtain statistically

self affine fractial. Terrain, water and clouds are typically modeled with statistically self affine fractal construction methods.

Inveriant fractal sets are formed with nonlinear transformations. This class of fractals includes self squaring fractals, such as the Mandelbrot set, which are formed with squaring functions in complex space ; and self inverse fractals formed with inversion procedures.