MODEL TEST PAPER

MODEL TEST PAPER

B.Tech. V Sem. Examination Paper
Computer Science and Engineering
5CS4-04 Computer Graphics and Multimedia

Time : 3 Hours Maximum Marks : 120

Instructions to Candidates :
Attempt all ten questions from Part A, any five questions out of eight from Part B and any four questions out five from Part C. (Schematic diagrams must be shown wherever necessary). Any data you feel missing suitable be assumed and stated clearly. Units of quantities used/calculated must be stated clearly.

Part - A

(Answer should be given up to 25 words only). All questions are Compulsory.

(10 × 2 = 20)

1. If a TV screen has 525 scan lines and an aspect ratio of 3 : 4 and if each pixel contains 12 bits of intensity information, how many bits are required for refresh rate 30 frames per second?
2. What is the refreshing rate for raster system.
3. What do you mean by geometric transformations.
4. Define scaling transformation.
5. A cubic Bezier curve segment is described by control points $P_0(2,2)$, $P_1(4,8)$, $P_2(8,8)$ and $P_3(9,5)$. Another curve segment is described by $q_0(a,b)$, $q_1(c,2)$, $q_2(15,2)$ and $q_3(18,2)$. Determine the values of a,b and c so that the two curve segments Join smoothly.
6. Which curve is better B-spline or Bezier?
7. What do you mean by illumination? Write its types.
8. Why RGB model is used.
9. What is turtle graphics program?
10. What is Koch Curve?

Part - B

(Analytical/Problem solving questions). Attempt any five questions.

(5 × 8 = 40)

1. Write short note on anti aliasing technique.
2. Differentiate between boundary fill and flood fill techniques.
3. What do you mean by homogeneous co-ordinates? How these co-ordinates are useful in transformation?
4. Show rotation of a 2D Box represented by (5,5) to (10,15) with respect to (5,5) by 90° in anticlockwise direction.
5. Write a procedure a display 2D, cubic Bezier curves given a set of 4 control points in XY plane.
6. Write the two methods of curve generation.
7. Explain how to simulate reflections from surfaces of different roughness using a reflection map.
8. Write short note on raster animations.

Part - C

(Descriptive/Analytical/Problem Solving/Design question). Attempt any four questions.

(4 × 15 = 60)

Explain scan conversion, write Bresenham's algorithm for line m > 1. What are the major adverse side effects of scan conversion?

Describe Cohen-Sutherland line clipping algorithm.

What is perspective representation? Explain various types of perspective projection?

Write a routine to convert RGB color model to HSV color model.

Write the strategy for design of animation sequences. Explain animation function also.

❑❑❑

❑ **Filled Area Primitives :** A common method for providing polygon fill on raster systems is the scan–line fill algorithm, which determines interior pixel spans across scan lines that intersect the polygon. The scan line algorithm can also be used to fill the interior of objects with curved boundaries. Two other methods for filling the interior regions of objects are **boundary-fill algorithm** and the **flood-fill algorithm**. These two fill procedures point the interior, one pixel at a time, outward from a specified interior point.

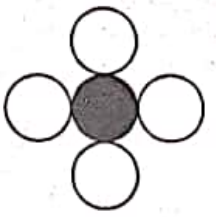
❑ **Boundary Fill Algorithm**

In this, we start at a point inside a region and point the interior outward toward the boundary. If boundary is specified in a single color, fill algorithm proceeds outward pixel by pixel until the boundary color is encountered. This method is useful in interactive painting packages, where interior points are easily selected.

The procedure, accepts as input the coordinates of an interior point (x, y), a fill color and a boundary color. Starting from (x, y), procedure tests neighbouring positions to determine whether they are of boundary color. If not, they are painted with fill color and their neighbours are tested. This process continues until all pixels upto the boundary color for the area have been tested.

Two methods for proceeding to neighbouring pixels from the current test position :

1. 4-connected approach :



2. 8-connected approach : Used to fill complex figures. It correctly fills the interior of area.

```
Procedure boundary fill (x,y, fill, boundary : integer);
   var
        current : integer;
        begin
        current = get pixel (x, y);
If (current <> boundary) and (current <> fill) then
   begin
        setpixel (x, y, fill);
        boundaryfill 4 (x + 1, y, fill boundary);
        boundaryfill 4 (x – 1, y, fill, boundary);
        boundaryfill 4 (x, y + 1, fill, boundary);
        boundaryfill 4(x, y – 1, fill, boundary);
   end
   end; {boundaryfill}
```

❑ **Flood-Fill Algorithm**

Sometimes we want to fill in an area that is not defined within a single color boundary. We point such areas by replacing a specified interior color instead of searching for a boundary color value.

```
Procedure floodfill 4(x, y, fillcolor, oldcolor
integer);
   begin
   If getpixel (x, y) = oldcolor then
   begin
        setpixel (x, y, fillcolor);
        floodfill 4 (x + 1, y, fillcolor, oldcolor);
        floodfill 4(x – 1, y, fillcolor, oldcolor);
        floodfill 4 (x, y + 1, fillcolor, oldcolor);
        floodfill 4 (x, y – 1, fillcolor, oldcolor);
   end
   end; {floodfill}
```

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 ***If a TV screen has 525 scan lines and an aspect ratio of 3:4 and if each pixel contains 12 bits of intensity information, how many bits are required for refresh rate 30 frames per second?*** *[R.T.U. 2017]*

Ans. 525 scan lines means vertical resolution is 525.

Aspect ratio is 3 : 4

This,

Vertical resolution = 4x

Horizontal resolution = 3x

4x = 525

x = 525/4

Vertical resolution = 525

Horizontal resolution = (3 × 525)/4 = 393.75

So,
Total bits = 525 × 393.75 × 12 bits × 30 frames
= 74418750 ≈ 74 Mbits/sec.

Q.2 ***In a Raster system with resolution 2560 × 2048. How many pixels could be accessed per second by a display controller that refresh that screen at a rate of 60 frames per second. Also calculate access time per pixel in the system.*** *[R.T.U. 2012]*

Ans. Since 60 frames are refreshed per second and each frame consists of 2560 × 2048 pixels, the access rate of such a system is:

$(2560 \times 2048)*60 = 314572800$ pixels/second
$= 3.14572800 \times 10^8$ pixels/second

According to the definition of access rate, we know that the access time per pixel should be:
1/ (access rate).

Therefore, the access time is around $3.17891439 \times 10^{-9}$ seconds/pixel or around 3.17 nanoseconds/pixel.

Q.3 ***What do you mean by vector displays?***

Ans. Vector Display : Random scan monitors draw one picture in one line at a time so they are also referred as Vector displays.

Q.4 ***What is the refreshing rate for raster system.***

Ans. Refreshing rate for raster system is generally 60 to 80 frames per second, it can be higher for some systems.

Q.5 ***Define graphics controller.***

Ans. Graphics Controller : One way to set up the organization of a raster system containing a separate display processor, also referred as graphics controller or display co-propressor.

Q.6 ***Define scan conversion.***

Ans. Scan Conversion : A major task of display processor is digitizing a picture definition given in an application program into a set of pixel-intensity values for storage in the frame buffer. This digitization process is called *scan conversion.*

Q.7 ***Why digital processor and random scan displays are designed.***

Ans. Random scan displays are designed to draw all component lines of a picture 30 to 60 times per second. Random scan system are designed for line drawing applications and cannot display realistic shaded scenes.

Digital processors are also designed to interface with interactive input devices such as mouse.

PART-B

Q.8 ***Write short note on anti aliasing technique.*** *[R.T.U. 2018, 2015, 2013, R.U. 2009, 2006]*

Ans. The distortion of information due to low frequency sampling (under sampling) is called aliasing. The appearance of displayed raster lines can be improved by applying antialising methods that compensate for the under sampling process.

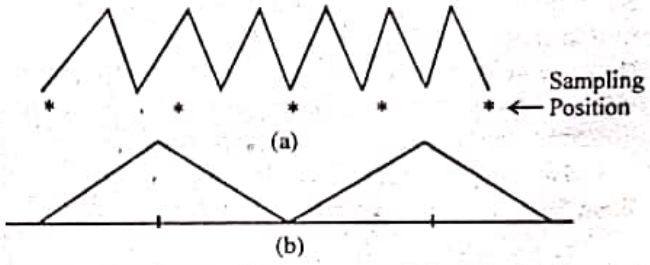


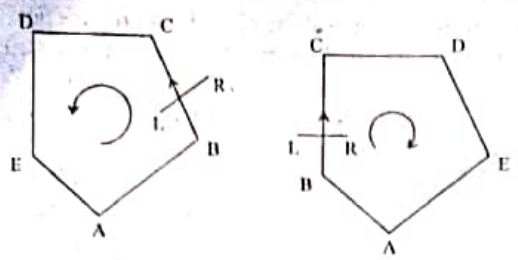
Fig. : Sampling the periodic shape in (a) at the marked positions produces the alised lower frequency representation in (b)

Various Antialising Techniques are :

(1) A straight forward antialiasing method is to increase sampling rate by treating the screen as if it were covered with a finer grid than is actually available. Then multiple sample points can be used across this finer grid to determine an appropriate intensity level for each screen pixel. This technique of sampling object characteristics at a high resolution and displaying the results at a lower resolution is called super sampling. By super sampling intensity information can be obtained from multiple points that contribute to the overall intensity of a pixel.

(2) Area sampling method involves computing the overlap checks *i.e.* pixel intensity is determined by calculating the area of overlap of each pixel with the objects to be displayed. Pixel overlap across are obtained by determining where object boundaries intersect individual pixel boundaries.

(3) Pixel phasing technique used for antialiasing raster objects, shift the display location of pixel area, pixel phasing s applied by micro positioning the electron beam in relation to object geometry.



(a) Positive orientation *(b) Negative orientation*

Fig.

Let A (x_1, y_1) and B(x_2, y_2) be the endpoints of a directed line segment. A point P(x, y) will be to the left of the line segment if the expression

$$C = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$

is positive. We say that the point is to the right of the line segment if this quantity is negative. If a point P is to the right of any one edge of a positively oriented, convex polygon, it is outside the polygon. If it is to the left of every edge of the polygon, it is inside the polygon.

This observation forms the basis for clipping any polygon, convex or concave, against a convex polygonal clipping window.

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 ***Show that the composition of two rotations is additive by concatenating the matrix representation for $R(\theta_1)$ and $R(\theta_2)$ to obtain : $R(\theta_1).R(\theta_2) = R(\theta_1+\theta_2)$.*** *[R.T.U. 2017]*

Ans. We know that the rotation matrix is

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

so, $R(\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix}$

and $R(\theta_2) = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{bmatrix}$

$$\therefore \quad R(\theta_1).R(\theta_2) = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) \end{bmatrix}$$

$$= R(\theta_1 + \theta_2)$$

$$\therefore \quad R(\theta_1).R(\theta_2) = R(\theta_1 + \theta_2)$$ **Hence Proved**

Q.2 ***Discuss issues related to polygon clipping.*** *[R.T.U. 2016]*

Ans. There are various issues related to the polygon clipping.

1. Edges of polygon need to be tested against clipping rectangle.
2. May need to add new edges.
3. Edges discarded or divided.
4. Multiple polygon can result from a single polygon.

Q.3 ***What do you mean by geometric transformations.***

Ans. Geometric Transformations : Changes in orientation, size and shape are accomplished with geometric transformations that alter the coordinate description of objects. The basic geometric transformations are translation, rotation and scaling.

Q.4 ***Define translation transformation.***

Ans. Translation Transformation : Translation is a rigid-body transformation that moves objects without deformation, that is every point on the object is translated by the same amount.

Q.5 ***Why rotation transformation is applied? Explain.***

Ans. A true dimensional rotation is applied to an object by repositioning it along a circular path in the xy plane. To generate a rotation, we specify a rotation angle θ and the position (x, y) of the rotation point about which the object is to be rotated.

Q.6 ***Define scaling transformation.***

Ans. Scaling Transformation : A scaling transformation alters the size of an object. This operation can be carried out for polygons by multiplying the coordinate value (x, y) of each vertex by scaling factor δx and δy to produce the transformed coordinates (x', y').

$$x' = x \cdot \delta x, \quad y' = y \cdot \delta y$$

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 *A cubic Bezier curve segment is described by control points $P_0(2,2)$, $P_1(4,8)$, $P_2(8,8)$ and $P_3(9,5)$. Another curve segment is described by $q_0(a,b)$, $q_1(c,2)$, $q_2(15,2)$ and $q_3(18,2)$. Determine the values of a,b and c so that the two curve segments Join smoothly.* [R.T.U. Dec. 2013]

Ans. For C^0 continuity at the point of contain $P_3(9, 5)$ & $Q_0(a, b)$ must be the same

$$P_3(9, 5) = Q_0(a, b)$$

Hence, $a = 9, b = 5$

as the curve join smoothly at the point of coincident C^1, continuity at this point should be the same.

Thus $P'_3 = Q'_0$ that is the tangent vector at the end point of the just curved should be identical to the tangent vector at the starting point of the second curve.

From equation :

$$3(P_3 - P_2) = 3(Q_1 - Q_2)$$

$$P_3 - P_2 = Q_1 - Q_0$$

$$\{(9, 5) - (8 - 8) = \{(c, 2) - (a, - b)\}$$

$$(c, 2) = 2(9, 5) - (8, 8) = (10, 2)$$

$$\boxed{\text{Thus, } a = 9, b = 5, c = 2} = c = 2$$

Q.2 *Why B-spline curves are used.*

Ans. B-spline curves are used for drawing and evaluating smooth curves especially in computer graphics and animation.

Q.3 *Which curve is better B-spline or Bezier?*

Ans. B-spline offer more control and flexibility compare to Bezier curves.

PART-B

Q.4 *What are Bezier cubic curves? Derive their properties. Also show that the sum of the blending functions is identical to 1 for all values of K why is it important?* [R.T.U. 2018, R.U. 2006]

OR

Discuss properties of Bezier curves. [R.T.U. 2016]

OR

Write short note on Bezier Curve. [R.T.U. 2015]

OR

Prove that "The Sum of blending functions is unity for every value of parameter in Bezier curves". [R.T.U. Dec. 2013]

OR

What is Bezier curve? Define blending functions? [R.T.U. 2013]

Ans. Bezier Curve : A Bezier curve section can be fitted to any number of control points. The number of control points to be approximated and their relative position determine the degree of the Bezier polynomial.

Suppose (n + 1) control point positions are given

$p_k = (x_k, y_k, z_k)$ with k = 0 to n.

These coordinate points can be blended to produce the following position vector P(u), which describes the path of an approximating Bezier polynomial function between p_0 and p_n.

$$P(u) = \sum_{k=0}^{n} p_k \, BEZ_{k,n}(u), \quad 0 \le u \le 1$$

The Bezier blending function $BEZ_{k,n}(u)$ are the Bernstein polynomials :

$$BEZ_{k,n}(u) = C(n, k)\, u^k (1 - u)^{n-k}$$

Where C(n, k) are binomial coefficients :

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

The four blending functions for cubic Bezier curve is obtained by putting (n = 3) since number of control points are 4.

$$\therefore BEZ_{0,3}(u) = (1-u)^3, BEZ_{1,3}(u) = 3u(1-u)^2$$

$$\therefore BEZ_{2,3}(u) = 3u^2(1-u), BEZ_{3,3}(u) = u^3$$

ILLUMINATION AND COLOUR MODELS 4

CHAPTER IN A NUTSHELL

❑ **Illumination :** Illumination is the exposure of an object to the light. Illumination and shading is the last important step in 3D graphics pipeline. An illumination which is also called lighting model or shading model, is used to calculate the intensity of the light that is reflected at a given point on the surface of the object.

Illumination consists of three parts

(1) Ambient reflection
(2) Diffuse reflection
(3) Specular reflection

1. **Ambient Reflection :** Few object surfaces in a scene are not illuminated by direct light sources but they are still available. These light rays are categorized as ambient light. This results from the effect of multiple reflections of light from many surfaces present in the environment.

2. **Diffuse Reflection :** Diffuse reflection is the characteristics of light reflected from a dull, non-shiny surface. Diffuse reflections are constant over-each surface in a scene, independent of the viewing direction.

3. **Specular Refection :** Specular reflection is observed on a shiny surface. Specular reflection is when the reflection is stronger in one viewing direction, in a concentrated region around specular reflection angle.

❑ **RGB Color Model :** The RGB color model is composed of the primary colors red, green and blue. Thus, RGB model is used in most color CRT monitors and color raster graphics.

The RGB colors are considered the "additive primaries" since the color are added together to produce the desired color.

YIQ Color Model : This is the color model used by U.S. commercial color television broadcasting. It is a recording of RGB for transmission efficiency and for downward compatibility for black and white television.

CMY Color Model : The CMY color model stands for cyan, magenta and Yellow which are the complements of red, green and blue respectively. Thus system is used primarily for printing. The CMY color are called the "subtractive primaries" since the color is obtained from, by what is removed from white not added.

HSV Color Model : HSV color model stands for HUV, saturation and value based on the artists. Use selects a spectral color and the amount of white and black that are to be added to obtain different shaded tints and tones.

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 ***What do you mean by illumination? Write its types.***

Ans. An illumination which is also called lighting model or shading model, is used to calculate the intensity of the light that is reflected at a given point on the surface of the object.

Illumination consists of three parts

(1) Ambient reflection
(2) Diffuse reflection
(3) Specular reflection

Q.2 ***Write two differences between CMY and RGB color model.***

Ans.

S.No.	RGB	CMY
1	More color space	Low color space
2	Use less numbers to encode a color	Use more number to encode a color

Q.3 ***What do you mean by diffuse reflection.***

Ans. Diffuse reflection is the characteristics of light reflected from a dull, non-shiny surface.

Q.4 ***Why RGB model is used.***

Ans. The RGB color model is composed of the primary colors red, green and blue. Thus, RGB model is used in most color CRT monitors and color raster graphics.

PART-B

Animation and Computer Graphics Realism 5

Chapter in a Nutshell

❑ **Ray Tracing Methods :** Ray tracing is an extension of this basic idea. Instead of merely looking for the visible surface for each pixel, We continue to bounce the ray around the scene collecting intensity contributions. This provides a simple and powerful rendering technique for obtaining global reflection and transmission effects.

Previous Years Questions

Part-A

Q.1 ***What is computer graphics realism?***

Ans. Computer Graphics Realism : The creation of realistic picture in computer graphics is known as realism. It is important in fields such as simulation, design, entertainments, advertising, research, education, command, and control.

Q.2 ***How realistic pictures are created in computer graphics?***

Ans. To create a realistic picture, it must be process the scene or picture through viewing coordinate transformations and projection that transform 3D viewing coordinates onto 2D device coordinates.

Q.3 ***What is Fractals?***

Ans. Fractal : A fractal is an image or a geometric object with self-similar, infinite properties produced by recursive or iterative algorithms.

Q.4 ***What is Koch Curve?***

Ans. Koch Curve : The Koch curve is a curve that is generated by a simple geometric procedure which can be iterated an infinite number of times by dividing a straight line segment into three equal parts and substituting the intermediate part with two segment of the same length.

Q.5 ***What is turtle graphics program?***

Ans. Turtle Graphics Program : The turtle program is a Robert that can move in two dimensions and it has a pencil for drawing. The turtle is defined by the following parameters :

Position of the turtle(x, y)

Heading of the turtle( ) the angle from the x-axis.

Part-B

Q.6 ***Write short note on simple recursive ray tracing without antialiasing.*** ***[R.T.U. Dec. 2013]***

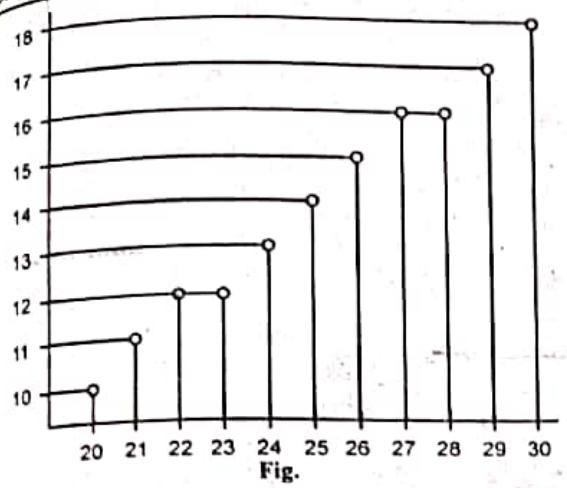


Fig.

Q.11 ***Explain flood fill algorithm. Differentiate it with Boundary fill algorithm.*** ***[R.T.U. 2017]***

OR

Differentiate between boundary fill and flood fill techniques. ***[R.T.U. 2016]***

OR

Write down flood fill Algorithm for Area filling? ***[R.T.U. 2015, 13, R.U. 2005]***

Ans. Flood-Fill Algorithm : Sometimes we want to fill in (or recolor) an area that is not defined within a single color boundary. Fig. shows an area bordered by several different color regions.

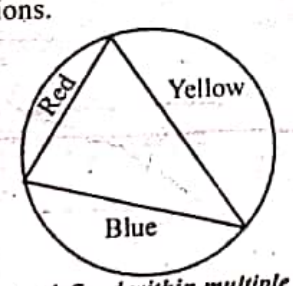


Fig. : An area defined within multiple boundaries

We can paint such areas by replacing a specified interior color instead of searching for a boundary color value. This approach is called a flood-fill algorithm. We start from a specified interior point (x, y) and reassign all pixel values that are currently set to a given interior color with the desired fill color. If the area we want to paint has more than one interior color, we can first reassign pixel values so that all interior points have the same color. Using either a 4-connected or 8-connected approach, we then step through pixel positions until all interior points have been repainted. The following procedure flood fills a 4-connected region recursively, starting from the input position.

```
void floodFill4 (int x, int y, int fill color, int old color)
    {
if (get pixel (x, y) = = oldcolor)
    {
        SetColor (fillcolor);
        setPixel (x, y) ;
        floodFill 4 (x + 1, y, fill color, oldcolor) ;
        floodFill 4 (x – 1, y, fill color, oldcolor) ;
        floodFill 4 (x, y + 1, fill color, oldcolor) ;
        floodFill 4 (x, y – 1, fill color, oldcolor) ;
    }
    }
```

We can modify procedure floodFill4 to reduce the storage requirements of the stack by filling horizontal pixel spans. In this approach, we stack only the beginning positions for those pixel spans having the value oldcolor. The steps in this modified flood-fill algorithm are similar to a boundary fill. Starting at the first position of each span, the pixel values are replaced until a value other than oldcolor is encountered.

Difference between boundary fill and flood fill techniques

S. No.	Flood Fill	Boundary Fill
1.	The flood fill algorithm looks for interior color which is to be replaced as a boundary color.	Another approach to area filling is to start at a point inside a region and paint the interior outward toward the boundary.
2.	Area can be bordered by several different color regions. We can paint such areas by replacing a specified interior color instead of searching for a boundary color value. This approach is called flood fill algorithm.	If the boundary is specified in a single color, the fill algorithm proceeds outward pixel by pixel until the boundary color is encountered. This method, called the boundary-fill algorithm.
3.	In this algorithm, we take a point P(x, y) which is interior to the region. This point is known as seed point. Then using a 4-connected or 8-connected approach the entire area can be filled by a specified color.	In this algorithm, the adjacent pixels are checked for boundary color has reached or not. If pixel colors are not that of boundary color, the pixel is colored.

4.	For large polygons, this algorithm may fail because it requires a frame buffer of free pixels with the polygon interior style.	This approach is more accurate than flood fill algorithm.
5.	It is costlier than boundary fill algorithm.	Less costlier than flood fill algorithm.

Q.12 ***Explain the scan line method for displaying the visible surface of a given polyhedron.***

[R.T.U. 2017]

Ans. This image-space method for removing hidden surfaces is an extension of the scan-line algorithm for filling polygon interiors. Instead of filling just one surface, we now deal with multiple surfaces. As each scan line is processed, all polygon surfaces intersecting that line are examined to determine which are visible. Across each scan line, depth calculations are made for each overlapping surface to determine which is nearest to the view plane. When the visible surface has been determined, the intensity value for that position is entered into the refresh buffer.

We assume that tables are set up for the various surfaces, which include both an edge table and a polygon table. The edge table contains coordinate endpoints for each line in the scene, the inverse slope ot each line, and pointers into the polygon table to identify the surfaces bounded by each line. The polygon table contains coefficients of the plane equation for each surface, intensity information for the surfaces, and possibly pointers into the edge table. To facilitate the search for surfaces crossing a given scan line, we can set up an active list of edges from information in the edge table. This active list will contain only edges that cross the current scan line, sorted in order of increasing x. In addition, we define a flag for each surface that is set on or off to indicate whether a position along a scan line is inside or outside of the surface. Scan lines are processed from left to right. At the leftmost boundary of a surface, the surface flag is turned on; and at the rightmost boundary, it is turned off.

Figure illustrates the scan-line method for locating visible portions of surfaces for pixel positions along the line. The active list for scan line 1 contains information from the edge table for edges AB, BC, EH. and FG. For positions along this scan line between edges AB and BC, only the flag for surface S1 is on. Therefore, no depth calculations are necessary, and intensity information for surface S1 is entered from the polygon table into the refresh buffer. Similarly, between edges EH and FG, only the flag for surface S2 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.

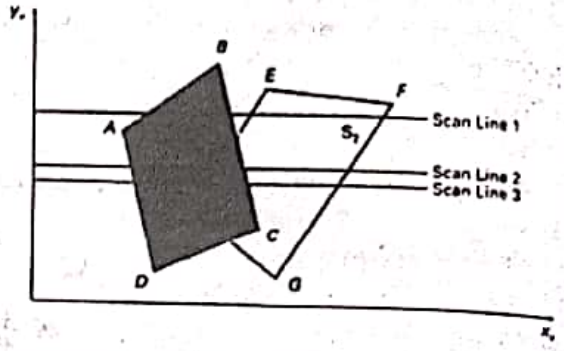


Fig. : Scan lines crossing the projectionof two surfaces, S_1 and S_2 in the view plane. Dashed lines indicate the boundaries of hidden surfaces.

For scan lines 2 and 3 in Fig., 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 S1 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 S1 is assumed to be less than that of S2. so intensities for surface S are loaded into the refresh buffer until boundary BC is encountered. Then the flag for surtace S1 goes off, and intensities for surface S2 are stored until edge FG is passed.

We can take advantage of coherence along the scan lines as we SS from one scan line to the next. In Fig., scan line 3 has the same active list of edges as scan line 2. Since no changes have occurred in line intersections, it is unnecessary again to make depth calculations between edges EH and BC. The two surfaces must be in the same orientation as determined on scan line 2, so the intensities for surface S, can be entered without further calculations.

Any number of overlapping polygon surfaces can be processed with this scan-line method. Flags for the surfaces are set to indicate whether a position is inside or outside, and depth calculations are performed when surfaces overlap.

Q.13 ***Explain the following terms in context of display devices :***

(i) resolution
(ii) flickering
(iii) interlacing
(iv) refreshing

[R.T.U. 2016]

Q.7 ***What do you mean by clipping algorithm.***

Ans. Clipping Algorithm : 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 clipping algorithm.

PART-B

Q.8 ***What do you mean by homogeneous co-ordinates? How these co-ordinates are useful in transformation?***

[R.T.U. 2018, 2015, 2013, R.U. 2004]

OR

What is homogeneous co-ordinate? Discuss the composite transformation matrices for two successive translations and scaling.

[R.T.U. Dec. 2013, R.U. 2007]

Ans. Homogeneous Coordinates : The homogeneous coordinates representations of points, lines and planes are useful for describing and manipulating graphical objects. The homogeneous representation of an object in n-space is an object in (n+1) space.

The coordinates in n-space are called ordinary coordinates and those in (n+1) space are called homogeneous coordinates. Example, the homogeneous representation of a two dimensional point [x, y] is (x_h, y_h, h) where h is any non-zero scalar which is called the scale factor. The mapping from a homogeneous point [a, b, c] back to its two dimensional image is [a/c, b/c]. An alternate view of the two dimensional image of (x_h, y_h, h) is that it is a projection onto the plane h = 1.

Thus when a Cartesian point (x, y) is converted to a homogeneous representation (x_h, y_h, h) equations containing x and y such as f(x, y)=0 become homogeneous equations in the three parameters x_h, y_h and h. This just means that if each of the three parameters is replaced by any value V times that parameter, the value V can be factored out of that equation.

Homogeneous coordinates are useful because expressing position in homogeneous coordinates allows us represent all geometric transformation equations as matrix multiplications.

As graphic applications require sequence of transformation, (for example in describing and picture construction applications, a sequence of transformations such as translations, rotations and scaling are performed to fit the picture components into their proper positions) in such cases, expressing positions in homogeneous coordinates allows us to represent such transformations as matrix multiplications.

Coordinates are expressed with three element column vectors and transformation operations are written by 3 by 3 matrix. For example, for translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Which can be written as

$$P' = T(t_x, t_y) \cdot P$$

Here h = 1 is the scaling factor.

Scaling transformation relative to the coordinate origin is expressed as θ the matrix multiplication

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$p' = S(S_x, S_y).p$

where $S(S_x, S_y)$ is the 3 by 3 matrix with parameters S_x and S_y.

Q.9 ***Write the followings :***

(i) Composite Transformations *[R.T.U. 2018, 2013]*

(ii) Inverse Transformations *[R.T.U. 2018, 2013]*

OR

Provide an example of inverse transformation in homogeneous coordinate system. *[R.T.U. 2016]*

Ans.(i) Composite Transformations

To deal with complex transformations like two successive translations, two successive rotations, rotation about an arbitrary point, reflection about line not passing through origin. All these include more than one transformations. Thus they are composite transformations.

Say, we want to apply a series of transformations T_1, T_2, T_3 to a set of points. There are two ways to calculate the new transformed coordinates.

(1) Calculate $P' = T_1.P$, $P'' = T_2.P'$. $P''' = T_3.P''$

P'''is the final transformed coordinate.

(2) Calculate $T = T_1 * T_2 * T_3$ then $P''' = T.P$.

Second method saves large number of additions and multiplications (computational time) and needs approximately one third of as many operations. Thus we compose matrices one into one final transformation matrix and then apply it to the points.

To Successive Translations

Assume a given point P(x, y) to be translated by a factor (t_{x1}, t_{y1}) and retranslation by (t_{x2}, t_{y2}).

Corresponding translation matrices will be :

$$T_1 = \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix},$$

$$T_2 = \begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

So, composite translation matrix would be :

$$T = T_1.T_2 = \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

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

New coordinates after two successive translations

$$P' = T_2(t_{x2}, t_{x2}).T_1(t_{x1}, t_{y1}).P$$
$$= T.P.$$
$$= T(t_{x1}+t_{x2}, t_{y1}+t_{y2}).P$$

From the composite matrix, we can deduce that two successive translations are additive.

Two Successive Rotations

Assume a point to be rotated by θ_1 and then by θ_2, we can calculate resultant points in two ways :

(1) Stick $(\theta_1 + \theta_2)$ in for θ_1, or

(2) Calculate T_1 for θ_1, then T_2 for θ_2 and then multiply them.

Both of the above gives same result.

Rotation matrices for rotation of θ_1 and θ_2 are :

$$T_1 \text{ or } R(\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_2 \text{ or } R(\theta_2) = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R(\theta_1).R(\theta_2)$$
$$= \begin{bmatrix} \cos\theta_1.\cos\theta_2 - \sin\theta_1.\sin\theta_2 & -\cos\theta_1.\sin\theta_2 - \sin\theta_1.\cos\theta_2 & 0 \\ \sin\theta_1.\cos\theta_2 + \cos\theta_1.\sin\theta_2 & -\sin\theta_1.\sin\theta_2 + \cos\theta_1.\cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= R(\theta_1 + \theta_2)$$

So, $P' = R(\theta_1 + \theta_2)P$

Ans. (ii) Inverse Transformations

In graphics, for many reasons, the user wants to undo the transformation once it has been performed. Obviously, to undo any given transformation, one more transformation is required to be performed. This second transformation must be exactly opposite to the first one. The transformation which is opposite to any given transformation is called *inverse transformation* for the given transformation. In this sense, the original transformation and its inverse transformation are always inverse of each other. Here the question is how to find out inverse transformation of any given transformation? For this, we can exploit an important property of matrix. That is

$$TT^{-1} = I$$

Here T^{-1} is an inverse matrix of matrix T and I is a unit matrix.

Thus, by multiplying any transformation matrix by its inverse matrix, we can undo the process of transformation. For example, to undo translation of

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

We need an inverse matrix $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -t_x & -t_y & 1 \end{bmatrix}$

Q.10 ***Show rotation of a 2D Box represented by (5,5) to (10,15) with respect to (5,5) by 90° in anticlockwise direction.*** *[R.T.U. 2017]*

Ans. The composite matrix for anti-clockwise rotation about on arbitrory point is given as

$$T_1.R.T_2 = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ -x_p\cos\theta + y_p\sin\theta + x_p & -x_p\sin\theta - y_p\cos\theta + y_p & 1 \end{bmatrix}$$

Here, $\theta = 90°$, $x_p = 5$ and $y_p = 5$ substituting these valeus in the composite matrix, we get

$$T_1.R.T_2 = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 10 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} 5 & 5 & 1 \\ 5 & 15 & 1 \\ 10 & 15 & 1 \\ 10 & 5 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 10 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 5 & 1 \\ -5 & 5 & 1 \\ -5 & 10 & 1 \\ 5 & 10 & 1 \end{bmatrix}$$

Q.11 What is the difference between scaling and rotation? *[R.T.U. 2015]*

Ans. Scaling : A scaling transformation is used to change the size of an object. Scaling about the origin (0, 0) is achieved by multiplying the coordinates of a point by x- and y-scale factors:

$x^2 = x \cdot s_x$

$y^2 = y \cdot s_y$

If |sx| and |sy| are both >1, the effect is of increasing the size of an object. In order to reduce the size, |sx| and |sy| must be <1.

Suppose we want to double the size of a 2-D object. What do we mean by double? Double in size, width only, height only, along some line only? When we talk about scaling we usually mean some amount of scaling along each dimension. That is, we must specify how much to change the size along each dimension. Below we see a triangle and a house that have been doubled in both width and height (note, the area is more than doubled).

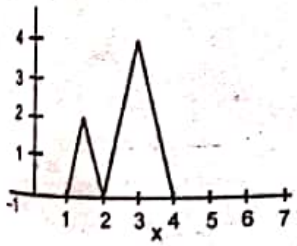


Fig.

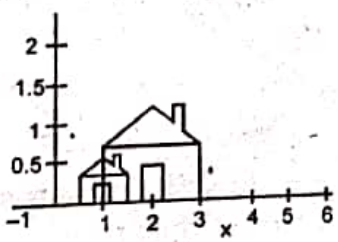


Fig.

The scaling for the x dimension does not have to be the same as the y dimension. If these are different, then the object is distorted. What is the scaling in each dimension of the pictures below?

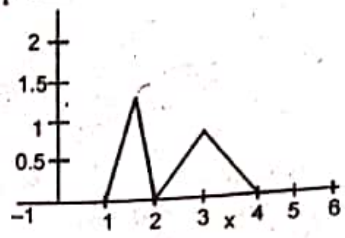


Fig.

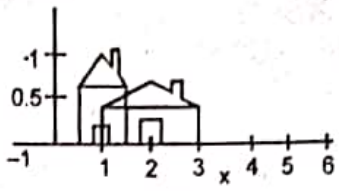


Fig.

And if we double the size, where is the resulting object? In the pictures above, the scaled object is always shifted to the right. This is because it is scaled with respect to the origin. That is, the point at the origin is left fixed. Thus scaling by more than 1 moves the object away from the origin and scaling of less than 1 moves the object toward the origin.

Rotation

Another common type of transformation is rotation. This is used to orientate objects. Below, we see objects that have been rotate by 25 degrees.

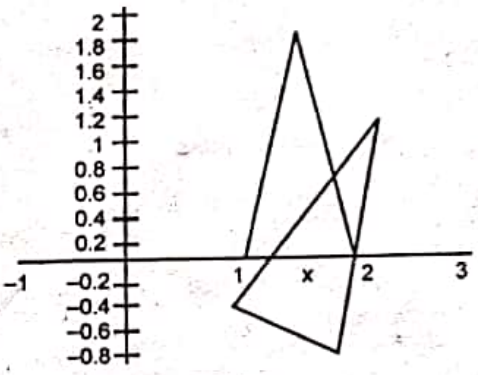


Fig.

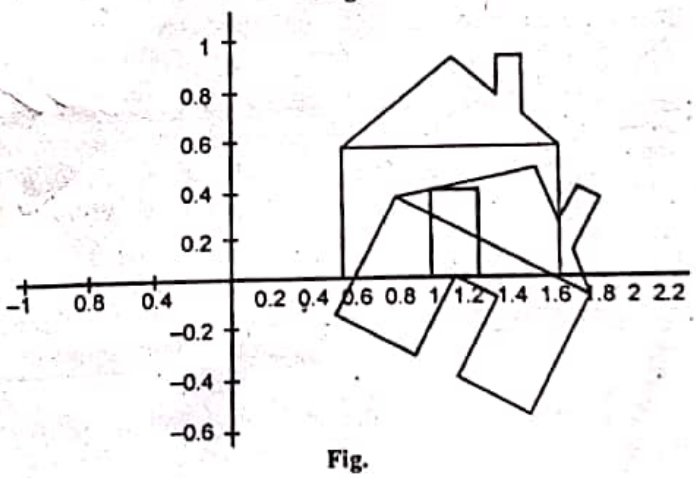


Fig.

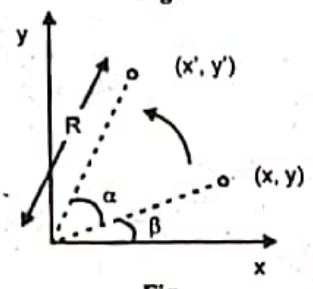


Fig.

∴ Therefore

$$P(u) = p_0(1-u)^3 + p_1 \cdot 3u(1-u)^2 + p_2 \cdot 3u^2(1-u) + p_3u^3$$

Where p_0, p_1, p_2, p_3 are control points

$P(0) = p_0 \Rightarrow$ By putting $u = 0$

$P(1) = p_3 \Rightarrow$ By putting $u = 1$

i.e. a Bezier curve always passes through the first and last control points.

Properties of Bezier Curves

Property 1

The parametric first derivatives at end positions are :

$$P'(0) = 3(p_1 - p_0)$$
$$P'(1) = 3(p_3 - p_2)$$

Thus the slope of the beginning of the curve is along the line joining the first two control points and the slope at the end of the curve is along the line joining the last two end points.

Property 2

Parametric second derivatives are :

$P''(0) = 6(p_0 - 2p_1 + p_2)$; $P''(1) = 6(p_1 - 2p_2 + p_3)$

Also sum of all blending functions is 1.

$$\sum_{k=0}^{n} BEZ_{k,n}(u) = 1$$

i.e. (for cubic Bezier curves)

$$(1-u)^3 + 3u(1-u)^2 + 3u^2(1-u) + u^3$$
$$= 1 + 3u^2 - 3u - u^3 + 3u - 6u^2 + 3u^3 + 3u^2 - 3u^3 + u^3 = 1$$

Bezier curves are widely available in various CAD system in graphics packages in painting packages since they are reasonably powerful in curve design.

Many graphic packages provide only cubic spline functions. This give reasonable design flexibility while avoiding the increased calculation needed with higher order polynomials.

Q.5 ***Write a procedure of display 2D, cubic Bezier curves given a set of 4 central points in XY plane.*** ***[Note : Read central as control.]*** ***[R.T.U. 2018]***

OR

Write a procedure a display 2D, cubic Bezier curves given a set of 4 control points in XY plane. ***[R.T.U. 2012]***

Ans. Cubic Bezier curves are generated with four control points. The four blending functions for cubic Bezier curves, obtained by substituting

$n = 3$ into Equation:

$$BEZ_{k,n}(u) = C(n, k)u^k(1-u)^{n-k}$$

are:

$$BEZ_{0,3}(u) = (1-u)^3$$
$$BEZ_{1,3}(u) = 3u(1-u)^2$$
$$BEZ_{2,3}(u) = 3u^2(1-u)$$
$$BEZ_{3,3}(u) = u^3$$

Plots of the four cubic Bezier blending functions are given in following figure. The form of the blending functions determine how the control points influence the shape of the curve for values of parameter u over the range from 0 to 1. At $u = 0$, the only nonzero blending function is $BEZ_{0,3}$ which has the value 1. At $u = 1$, the only nonzero function is $BEZ_{3,3}$ with a value of 1 at that point. Thus, the cubic Bezier curve will always pass through control points p_0 and p_3. The other functions, $BEZ_{1,3}$ and $BEZ_{2,3}$, influence the shape of the curve, at intermediate values of parameter u, so that the resulting curve tends toward points p_1 and p_2. Blending function $BEZ_{1,3}$ is maximum at $u = 1/3$, and $BEZ_{2,3}$ is maximum at $u = 2/3$.

We note in following figure that each of the four blending functions is nonzero over the entire range of parameter u. Thus, Bezier curves do not allow for local control of the curve shape. If we decide to reposition, any one of the control points, the entire curve will be affected. At the end positions of the cubic Bezier curve, the parametric first derivatives (slopes) are

$$P'(0) = 3(p_1 - p_0),$$
$$P'(1) = 3(p_3 - p_2)$$

And the parametric second derivatives are:

$$P''(0) = 6(p_0 - 2p_1 + p_2),$$
$$P''(1) = 6(p_1 - 2p_2 + p_3)$$

We can use these expressions for the parametric derivatives to construct piece-wise curves with C^1 or C^2 continuity between sections.

By expanding the polynomial expressions for the blending functions, we can write the cubic Bezier point function in the matrix form :

$$P(u) = [u^3u^2u].M_{Bez}.\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Where the Bezier matrix is:

$$M_{Bez} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

We could also introduce additional parameters to allow adjustment of curve "Tension" and "bias".

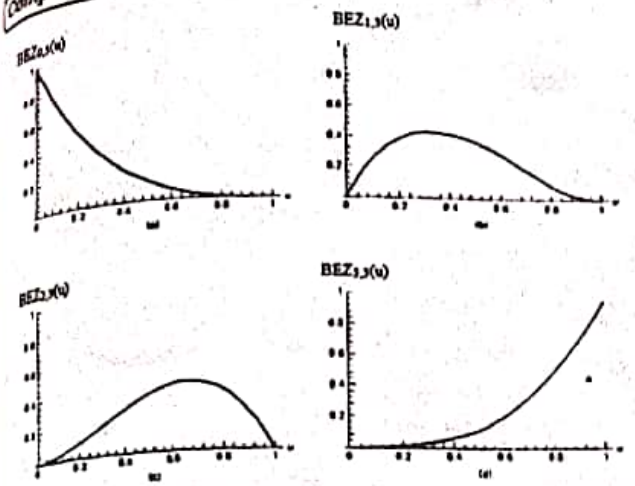


Fig. The four Bezier blending functions for cubic curves (n = 3)

Q.6 Write the two methods of curve generation. *[R.T.U. 2018]*

Ans. B-Spline Curves and Surfaces : We can write a general expression for the calculation of coordinate position along a B-spline curve in a blending function formulation as

$$P(u)=\sum_{k=0}^{n} p_k B_{k,d}(u),\ u_{min} \le u_{max}\ \ 2 \le d \le n+1 \quad ...(1)$$

where the p_k are an input set of n + 1 control points. There are several differences between this B-spline formulation and that for Bezier splines. The range of parameter u now depends on how we choose the B-spline parameters. And the B-spline blending functions $B_{k,d}$ are polynomials of degree d – 1 where parameter d can be chosen to be any integer value in the range from 2 up to the number of control points, n + 1. (Actually, we can also set the value of d at 1 but then our "curve" is just a point plot of the control points). Local control for B-splines is achieved by defining the blending function over subintervals of the total range of u.

Blending function for B-spline curves are defined by the Cox-deBoor recursion formulas:

$$B_{k,1}(u)=\begin{cases}1, & \text{if } u_k \le u \le u_{k+1}\\ 2, & \text{otherwise}\end{cases} \quad ...(2)$$

$$B_{k,d}(u)=\frac{u-u_k}{u_{k+d-1}-u_k}B_{k,d-1}(u)+\frac{u_{k+d}-u}{u_{k+d}-u_{k+1}}B_{k+1,d-1}(u)$$

Where each blending function is defined over d subintervals of the total range of u. The selected set of subinterval endpoints u_j is referred to as a knot vector. We can choose any values for the subinterval endpoints satisfying the relation $u_j \le u_{j+1}$. Values for u_{min} and u_{max} then depend on the number of control points we select, the value we choose for parameter d, and how we set up the subintervals (knot vector). Since it is possible to choose the elements of the knot vector so that the denominators in the previous calculations can have a value of 0, this formulation assumes that any terms evaluated as 0/0 are to be assigned the value 0.

Fig. demonstrates the local-control characteristics of B-splines. In addition to local control, B-splines allow us to vary the number of control points used to design a curve without changing the degree of the polynomial. Also, any number of control points can be added or modified to manipulate curve shapes. Similarly, we can increase the number of values in the knot vector to aid in curve design.

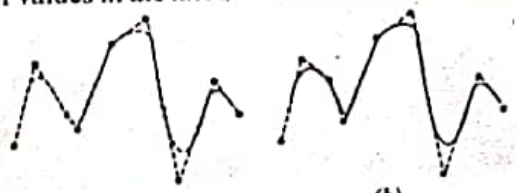


Fig.: Local modification of a B-spline curve. Changing one of the control points in (a) produces curve (b) which is modified only in the neighborhood of the altered control point

When we do this, however, we also need to add counts since the size of the knot vector depends on parameter n.

B-spline curves have the following properties :

(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 d subintervals of the total range of u, starting at 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 labelled as $\{u_0 + u_1 \dots u_{n+d}\}$, 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 (between two successive knot values) is influenced by d control points.

(vii) Any one control point can affect the shape of at most d curve sections.

In addition, a B-spline curve lies within the convex hull of at most d + 1 control points, so that B-splines are tightly bound to the input positions. For any value of u in the interval from knot value u_{d-1} to u_{n+1}, the sum over all basis function is 1:

$$\sum_{i=0}^{n} B_{k,d}(u)=1 \quad ...(3)$$

$$I = I_{diff} + I_{spec}$$

$$= k_a I_a + k_d I_l (N.L) + k_s I_l (N.H)^{ns}$$

For a multiple point light source the above equation can be modified as

$$I = k_a I_a + \sum_{i=1}^{M} I_{l_i} \left[k_d (N.L_i) + k_s (N.H_i)^{ns} \right]$$

where

K_a = ambient-reflection coefficient
I_a = ambient light intensity
K_d = diffuse-reflection coefficient
I_Q = diffuse light intensity
K_s = specular-reflection coefficient
η_s = specular-reflection parameter

Therefore, in case of multiple point light sources the light reflected at any surface point is given by summing the contributions from the individual sources.

Q.6 ***Explain Illumination model.*** *[R.T.U. 2017]*

Ans. Illumination model : An illumination which is also called lighting model or shading model, is used to calculate the intensity of the light that is reflected at a given point on the surface of the object. This illumination model is further used by rendering process to determine the light intensity of each projected pixel position in a scene.

Creating a virtual reality of a real scene (say a class room) involves:

- Modeling and positioning of several complex objects.
- Determine the visible surface and project the view w.r.t. the viewer,
- Obtain shading using surface normal, surface properties and light sources.
- Obtain shadows from occlusions

Q.7 ***Discuss about the difference between CMY and RGB color?*** *[R.T.U. 2015]*

Ans. RGB and its subset CMY form the most basic and well-known color model. This model bears closest resemblance to how we perceive color. It also corresponds to the principles of additive and subtractive colors.

Differences between CMY and RGB Color

S.No.	RGB	CMY
1	RGB is based on projecting. Red light plus Green light plus Blue light all projected together create white. Black is encoded as the absence of any color.	CMYK is based on ink. Superimpose Cyan ink plus Magenta ink plus Yellow ink, and we get black, although this format also encodes Black (K) directly. White is encoded by the absence of any color.
2	Prism uses RGB internally. Exporting in RGB will give you results very close to what we see on screen.	Even though it uses one more number to encode a color, the CMYK scheme encodes a smaller "color space" than does RGB.
3	More color space	Low color space
4	Use less numbers to encode a color	Use more number to encode a color
5	It is additive	It is subtractive
6	Mostly used for screen purpose	Mostly used for printing purpose

Q.8 ***Explain how to simulate reflections from surfaces of different roughness using a reflection map.*** *[R.T.U. Dec. 2013]*

Ans. A basic Reflection map creates the illusion of chrome, glass, or metal by applying a map to the geometry so that the image looks like a reflection on the surface.

An automatic Reflection map does not use mapping coordinates, but instead looks outward from the center of the object and maps what it "sees" onto the surface.

Another way to generate reflections automatically is to assign a Raytrace map to be the reflection map.

A flat-mirror Reflection map is applied to a series of coplanar faces and reflects objects facing it, exactly like a real mirror. The most common use of Reflection maps in a realistic scene is to add just a touch of reflection to an otherwise non-reflective surface. By default, Reflection map strength is 100 percent, as it is for other maps. For many kinds of surfaces, however, reducing the strength gives the most realistic result. A polished table top, for example, primarily shows a wood grain; the reflections are secondary.

slow-downs, and curved motion paths. Kinematic specifications of a motion can also be given by simply describing the motion path. This is often done using spline curves.

Q.8 *What is tiling the plane? Write its types.*

Ans. Tiling the Plane

- Use one or more geometric shapes
- Tessellation (without gaps) of flat surface
- Shape repeated
- Moving infinity
- Covering entire plane
- Used arts, mosaics, wall papers, tiled floor

Types of Tiling

- Monohedral tiling
- Cairo tiling
- Drawing tiling
- Reptiles

Monohedral Tiling : This is based on single polygon

Cairo Tiling

- Four pentagon fit together to form hexagon
- Used to tile the plane
- Many street in cairo, Egypt in this pattern

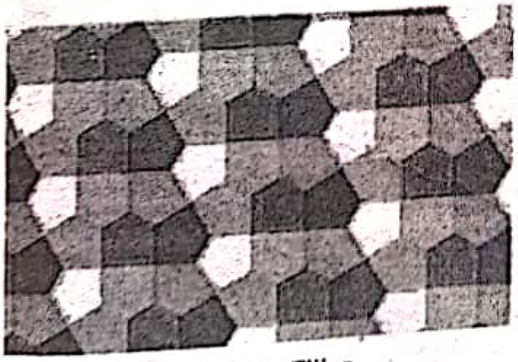


Fig. : Cairo Tiling

Drawing Tiling

- Large window setup
- Tiles grouped together into single figure
- Single figure drawn again and again
- Non periodic figure include
- Small to large and large to small

Reptiles

- Non periodic tiling
- Based on square, equilateral triangle

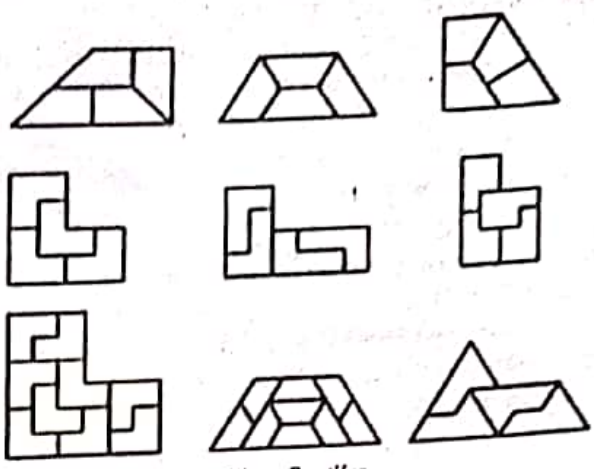


Fig. : Reptiles

Q.9 *Write short note on raster animations.*

Ans. Raster Animations : On raster systems, we can generate real-time animation in limited applications using raster operations. A simple method for translation in the xy plane is to transfer a rectangular block of pixel values from one location to another. Two-dimensional rotations in multiples of 90° are also simple to perform, although we can rotate rectangular blocks of pixels through arbitrary angles using antialiasing procedures. To rotate a block of pixels, we need to determine the percent of area coverage for those pixels that overlap the rotated block. Sequences of raster operations can be executed to produce real-time animation of either two-dimensional or three-dimensional objects, as long as we restrict the animation to motions in the projection plane. Then no viewing or visible-surface algorithms need be invoked.

We can also animate objects along two-dimensional motion paths using the color-table transformations. Here we predefine the object at successive positions along the motion path, and set the successive blocks of pixel values to color-table entries.

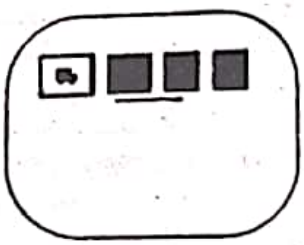


Fig. : Real time raster color table animation

We set the pixels at the first position of the objects to "on" values and we set the pixels at the other object

positions to the background color. The animation is then accomplished by changing the color-table values so that the object is "on" at successively positions along the animation path as the preceding position is set to the background intensity (Fig.).

Q.10 *Write note on Random fractals?*

Ans. Random Fractals : Fractal is the term associated with randomly generated curves and surfaces that exhibit a degree of self-similarity. These curves are used to provide naturalistic shapes for representing objects such as coastlines, rugged mountains, grass and fire.

Fractalizing a Segment : The simplest random fractal is formed by recursively roughening or fractalizing a line segment. At each step, each line segment is replaced with a random elbow.

This process applied to the line segment S having endpoints A and B. S is replaced by the two segments from A to C and from C to B. For a fractal curve, point C is randomly chosen along the perpendicular bisector L of S. the elbow lies randomly on one or the other side of the parent segment AB.

Three stages are required in the fractalization of a segment. In the first stage, the midpoint of AB is perturbed to form point C. In the second stage, each of the two segment has its midpoints perturbed to form points D and E. In the third and final stage, the new points F are added.

The color for a solid interior or for a hollow area outline is chosen with where fill color parameter fc is set to the desired color code. SetInteriorColourIndex (fc).

Pattern Fill : We select fill patterns with Set Interior Style Index (fi) where Patten index parameter (Pi) specifies a table position. For example, the following set of statements would fill the area defined in the fill area command with the second pattern type stored in the pattern table.

SetInteriorStyle (pattern)
SetInteriorStyleIndex (2)
Fill area (n, points)

Index(Pi)	Pattern(CP)
1	$\begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$
2	$\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix}$

Ans. (b) Character Attributes : The appearance of displayed a character can be controlled by attributes such as font, size, color and orientation.

Attributes can be set both for entire character strings and for individual characters defined as marker symbols.

Text Attributes : The choice of font or font type is set of characters with particulars design styles as courier, times new roman's font is set by the function

SetTextFont (tf)

font also displayed with styles i.e., (Solid, dotted, doubled) in boldface in italics.

1. Text color is set by SetTextColourIndex (tc).
2. Text size is set by SetCharacterHeight (ch).
3. Text width is set by SetcharacterExpansionFactor (cw).
4. Spacing between characters is controlled by SetCharacterSpacing (cs).
5. To arrange character Strings vertically or horizontally by SetTextPath (tp).

Marker Attributes : A marker symbol is a single character than can be displayed in different colors in different sizes.

We can select a particular character to be displayed the marker symbol with SetMarketType (mt).

Set the marker size with SetMarkersizeScaleFactor (ms)

Marker color is set with SetPolymarkerColorIndex (mc).

PART-C

Q.22 ***Explain scan conversion, write Bresenham's algorithm for line m > 1. What are the major adverse side effects of scan conversion?***
[R.T.U. 2018, Dec. 2013, 2013, 2010, Raj. Univ. 2002]

Ans. Scan Conversion : It is a process of digitizing picture definition into a set of pixel intensity.

Sampling : It defines the distance between two pixels.

Both are performed by display processor.

(1) A picture can be described in several ways. Let we have a raster display, a picture is completely specified by the set of intensities for the pixel position in the display.

(2) Shapes and colors of the object can be described internally with pixels arrays or with sets of basic geometric structures such as straight line segments and polygon color areas.

(3) The scene is then displayed either by loading the pixel arrays into the frame buffer or by scan converting the basic geometric structure specifications into pixel patterns.

(4) Graphics programming packages provide functions to describe a scene in terms of these basic geometric structures, referred to as *output primitive.*

(5) Each output primitive is specified with input coordinate data and other information about the way that object is to be displayed.

Like points and straight lines segments are the simplest geometric components of pictures.

(6) Additional output primitive are :

Circles, conic sections, quadratic surfaces, spline curves and surfaces, polygon color areas.

Points : Points plotting is accomplished by converting an application program into appropriate operations.

Procedure :

(1) With a CRT monitor, the e⁻ beam is turned on the illuminate the screen phosphor at specific location.

(2) A random scan system (vector) stores the point plotting instructions in the display list and co-ordinate values in these instructions are converted to deflection voltages that position the e⁻beam during each refresh cycle.

For a black and white raster system, a point is plotted by setting the last value corresponding to specified position within the frame refer to 1.

Line :

(1) Line drawing is accomplished by calculating intermediate positions along the line path between two specified end point positions.

(2) An output device is directed to fill these positions in between end points. Eg. a vector pen plotter.

(3) Linearly varying horizontal and vertical voltage are generated that are proportional to the required changes in the x and y directions to produce smooth line.

(4) Digital devices display a straight line segment by plotting discrete points between the two end points.

(5) Raster video display line color is loaded into the frame buffer, the video controller then "plots" the screen pixels.

Bresenham's algorithm for straight line :

Working : In this algorithm, the x or y increment is set to 1 in the direction of x axis or y axis respectively depending upon the slope of line and the increment in other direction (y or x) is calculated by selecting the closest pixel from the true line at each position of x or y based upon the value of decision variable.

Given **Case : m ≥ 1 :** When magnitude of the slope is greater than 1 (line lies between 45° to 90°).

In this algorithm we step along y direction (instead of x) in unit steps and calculate the successive value of x using the Bresenham's Principle (closest pixel), as shown in Fig.

Unit Sampling Direction

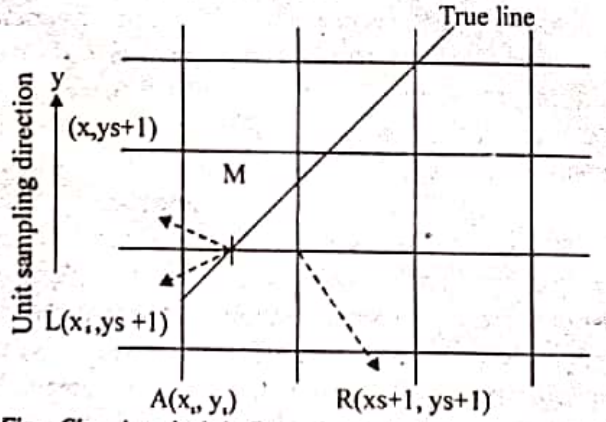


Fig. : Choosing pixels in Bresenham's line algorithm (m> 1)

Algorithm

Input xs, ys, xe, ye

$dy = ye - ys$

$dx = xe - xs$

$inc1 = 2^* dy$

$inc2 = 2 * (dy - dx)$

$d = inc1 - dx$

$x = xs$

$y = ys$

If $(dx < 0)$

$x = xe$

$y = ye$

$x\,end = xs$

else

$x = xs$

$y = ys$

$x\,end = xe$

end if

while $(x \le x\,end)$

plot pixel (x, y)

If $(d \ge 0)$ then

$x = x + 1$

$d = d + inc2$ /* Select Right Pixel

else $d = d + inc2$

end If $y = y + 1$

end while

stop

Adverse side effects of scan conversion

Scan conversion is essentially a systematic approach to mapping objects that are defined in continuous space to their discrete approximation. The various forms of distortion that result from this operation are collectively referred to as the aliasing effects of scan conversion.

Staircase : A common example of aliasing effects is the staircase or jagged appearance we see when scan-converting a primitive such as a line or a circle. We also see the stair steps or "jaggies" along the border of a filled region.

Unequal Brightness : Another artifact that is less noticeable is the unequal brightness of lines of different orientation. A slanted line appears dimmer than a horizontal or vertical line, although all are presented at the same intensity level. The reason for this problem can be explained using Fig. below.

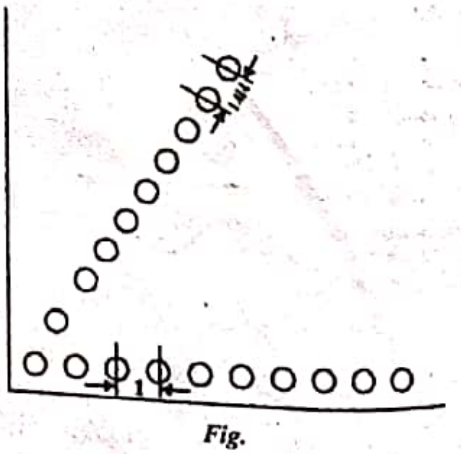


Fig.

where the pixels on the horizontal line are placed one unit apart, whereas those on the diagonal line are approximately 1.414 units apart. This difference in density produces the perceived difference in brightness.

The Picket Fence Problem : The picket fence problem occurs when an object is not aligned with, or does not fit into, the pixel grid properly. Fig.(a) below shows a picket fence where the distance between two adjacent pickets is not a multiple of the unit distance between pixels. Scan-converting it normally into the image space will result in uneven distances between pickets since the endpoints will have to be snapped to pixel coordinates [see Fig.(b)]. This is sometimes called global aliasing, as the overall length of the picket fence is approximately correct. On the other hand, an attempt to maintain equal spacing will greatly distort the overall length of the fence [see Fig.(c)]. This is sometimes called local aliasing, as the distances between pickets are kept close to their true distances.

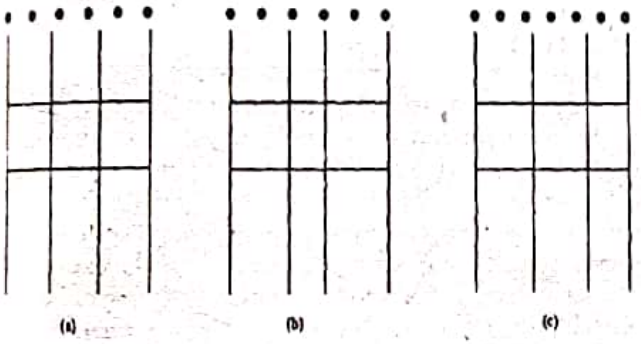


Fig. : The picket fence problem

Another example of such a problem arises with the outline font. Suppose we want to scan-convert the uppercase character "E" in Fig. below from its outline description to a bitmap consisting of pixels inside the region defined by the outline. The result in Fig. exhibits both asymmetry (the upper arm of the character is twice as thick as the other parts) and dropout (the middle arm is absent). A slight adjustment and/or realignment of the outline can lead to a reasonable outcome [see Fig.(c)].

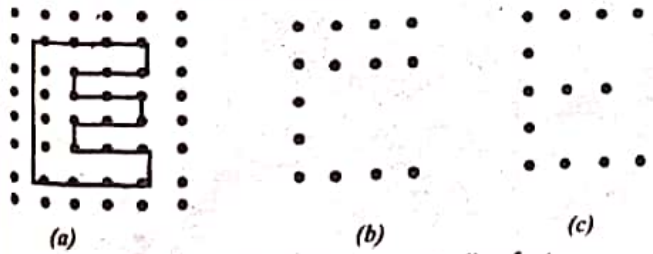


Fig. : Scan-converting an outline font

Anti-aliasing : Most aliasing artifacts, when appear in a static image at a moderate resolution, are often tolerable, and in many cases, negligible. However, they can have a significant impact on our viewing experience when left untreated in a series of images that animate moving objects. For example, a line being rotated around one of its endpoints becomes a rotating escalator with length-altering steps. A moving object with small parts or surface details may have some of those features intermittently change shape or even disappear.

Although increasing image resolution is a straightforward way to decrease the size of many aliasing artifacts and alleviate their negative, we pay a heavy price in terms of system resource (going from W × H to 2W × 2H means quadrupling the number of pixels) and the results are not always satisfactory. On the other hand, there are techniques that can greatly reduce aliasing artifacts and improve the appearance of images without increasing their resolution. These techniques are collectively referred to as anti-aliasing techniques. Some anti-aliasing techniques are designed to treat a particular type of artifact. For instance, an outline font can be associated with a set of rules or hints to guide the adjustment and realignment that is necessary for its conversion into bitmaps of relatively low resolution. An example of such approach is called the True Type font.

Pre-filtering and Post-filtering : Pre-filtering and post-filtering are two types of general-purpose anti-aliasing techniques. The concept of filtering originates from the field of signal processing, where true intensity values are continuous signals that consists of elements of various frequencies. Constant intensity values that correspond to a uniform region are at the low end of the spectrum. In order to lessen the jagged appearance of lines and other contours in the image space, we seek to smooth out sudden intensity changes, or in signal-processing terms, to filter out the high frequency components. A pre-filtering technique works on the true signal in the continuous space to derive proper values for individual pixels (filtering before sampling), whereas a post-filtering technique takes discrete samples of the continuous signal and uses the samples to compute pixel values (sampling before filtering).

Area Sampling : Area sampling is a pre-filtering technique in which we superimpose a pixel grid pattern onto the continuous object definition. For each pixel area that intersects the object, we calculate the percentage of overlap by the object. This percentage determines the proportion of the overall intensity value of the corresponding pixel that is due to the object's contribution. In other words, the higher the percentage of overlap, the greater influence the object has on the pixel's overall intensity value.

In Fig.(a) a mathematical line shown in dotted form is represented by a rectangular region that is one pixel wide. The percentage of overlap between the rectangle and each intersecting pixel is calculated analytically. Assuming that the background is black and the line is white, the percentage values can be used directly to set the intensity of the pixels [see Fig.(b)]. On the other hand, had the background been gray (0.5, 0.5, 0.5) and the line green (0, 1, 0), each blank pixel in the grid would have had the background gray value and each pixel filled with a fractional number f would have been assigned a value of [0.5 (1– f), 0.5 (1– f) + f, 0.5 (1– f)] a proportional blending of the background and object colors.

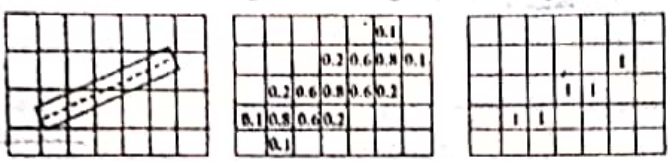


Fig. : Area sampling

Although the resultant discrete approximation of the line in Fig.(c) takes on a blurry appearance, it no longer exhibits the sudden transition from an on pixel to an off pixel and vice versa, which is what we would get with an ordinary scan-conversion method. This trade-off is characteristic of an anti-aliasing technique based on filtering.

In this approach we subdivide each pixel into subpixels and check the position of each subpixel in relation to the object to be scan-converted. The object's contribution to a pixel's overall intensity value is proportional to the number of subpixels that are inside the area occupied by the object. Fig. shows an example where we have a white object that is bounded by two slanted lines on a black background. We subdivide each pixel into nine (3 × 3) subpixels. The scene is mapped to the pixel values in Fig. The pixel at the upper right corner, for instance, is assigned 7.9 since seven of its nine subpixels are inside the object area. Had the object been red (1, 0, 0) and the background light yellow (0.5, 0.5, 0), the pixel would have been assigned $\left(1\times\frac{7}{9}+0.5\times\frac{2}{9}, 0.5\times\frac{2}{9}, 0\right)$, which is $\left(\frac{7}{9},\frac{1}{9},0\right)$.

Super sampling is often regarded as a post-filtering technique since discrete samples are first taken and then used to calculate pixel values. On the other hand, it can be viewed as an approximation to the area sampling method since we are simply using a finite number of values in each pixel area to approximate the accurate analytical result.

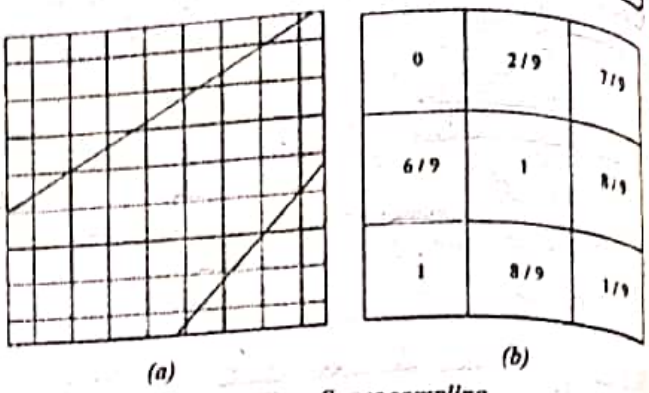


Fig. : Super sampling

Lowpass Filtering : This is a post-filtering technique in which we reassign each pixel a new value that is a weighted average of its original value and the original values of its neighbours. A lowpass filter in the form of a (2n + 1) × (2n + 1) grid, where n ≥ 1, holds the weights for the computation. All weight values in a filter should sum to one. An example of a 3 × 3 filter is given in Fig.(a). To compute a new value for a pixel, we align the filter with the pixel grid and center it at the pixel. The weighted average is simply the sum of products of each weight in the filter times the corresponding pixel's original value. The filter shown in Fig. means that half of each pixel's original value is retained in its new value, while each of the pixel's four immediate neighbors contributes one eighth of its original value to the pixel's new value. The result of applying this filter to the pixel values in Fig is shown in Fig.(b) A **lowpass filter** with equal weights, sometimes referred to as a box filter, is said to be doing neighborhood averaging. On the other hand, a filter with its weight values conforming to a two-dimensional is called a **Gaussian filter**.

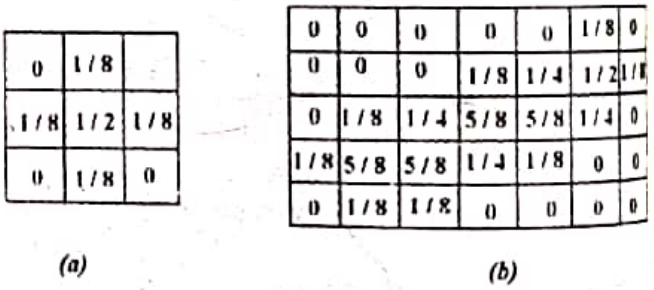


Fig. : Lowpass filtering

Pixel Phasing : Pixel phasing is a hardware-based anti-aliasing technique. The graphics system in this case is capable of shifting individual pixels from their normal positions in the pixel grid by a fraction (typically 4 and 2) of the unit distance between pixels. By moving pixels closer to the true line or other contour, this technique is very effective in smoothing out the stair steps without reducing the sharpness of the edges.

Q.16 ***Explain viewing pipeline in two-dimensional graphics.***

Ans. Two Dimensional Viewing Pipeline : The term viewing pipeline describes a series of transformations, which are passed by geometry data to end up as image data being displayed on a device. The two dimensional viewing pipeline describes this process for two dimensional data.

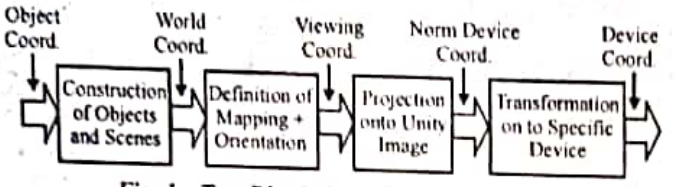


Fig. 1 : Two Dimensional Viewing pipeline

The coordinates in which individual objects created are called model (object) coordinates. When several objects are assembled into a scene they are described by world coordinates.

After transformation into the coordinate system of the viewer they became viewing coordinates.

Their projection on to a common plane yields device independent normalized coordinates. Finally after mapping those normalized coordinates to a specific device, we get device coordinates.

World coordinates : The clipping window is mapped into a viewport.

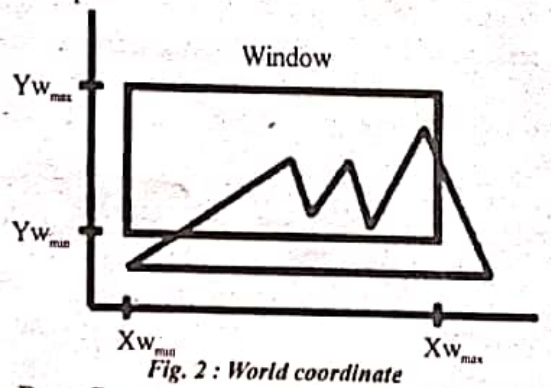


Fig. 2 : World coordinate

View Port Coordinates : Viewing world has its own coordinates, which may be a non-uniform scaling of World Coordinates.

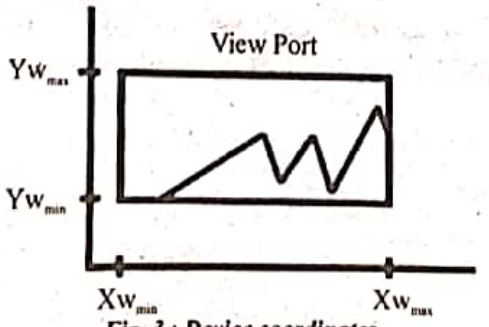


Fig. 3 : Device coordinates

PART-C

Q.17 ***Describe Cohen-Sutherland line clipping algorithm.*** *[R.T.U. 2016]*

OR

Explain Cohen Sutherland line algorithm. *[R.T.U. 2017]*

OR

Explain Cohen-Sutherland line clipping algorithm with region code details? *[R.T.U. 2015, 2012, R.U. 2006, 2003]*

Ans. Cohen - Sutherland Line Clipping :

This is one of the oldest and most popular line clipping procedures.

In this algorithm we divide the line clipping into two phases :

(1) Identify those lines which intersect the clipping window and so need to be clipped.

(2) Perform the clipping.

We can divide the lines in 3-categories :

(i) Visible : Both end points of the line lie within the window.

(ii) Not Visible : The line definitely lies outside the window. This will occur if the line from (x_1, y_1) to (x_2, y_2) satisfies any one of the following 4 inequalities

$x_1, x_2 > x_{max}$

$y_1, y_2 > y_{max}$

$x_1, x_2 < x_{min}$

$y_1, y_2 < y_{min}$

(iii) Clipping Candidate : The line is in neither category 1 nor in 2.

In Fig. 1 line AB is in category 1 (visible); line CD and EF are in category 2 (not visible); and lines GM, IJ and KL are in category 3 (Clipping Candidate).

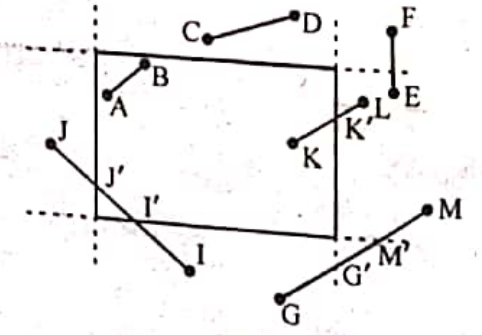


Fig. 1

The algorithm employs an efficient procedure for finding the category of a line. It proceeds in two steps:

(1) Assign a 4 bit region code to each endpoint of the line. The code is determined according to which of the following nine regions of the plane endpoint lies in.

Starting from the leftmost bit, each bit of the code is set to true (1) or false (0) according to the scheme.

1001	1000	1010
0001	0000 Window	0010
0101	0100	0110

Fig. 2

Bit 1 → endpoint is above the window
= sign $(y - y_{max})$

Bit 2 → endpoint is below the window
= sign $(y_{min} - y)$

Bit 3 → endpoint is to the right of the window
= sign $(x - x_{max})$

Bit 4 → endpoint is to the left of the window
= sign $(x_{min} - x)$

We use the convention that sign (a) = 1 if a is positive, 0 otherwise, of course a point with code 0000 is inside the window.

(2) The line is visible if both region code are 0000, and not visible if the bitwise logical AND of the codes is not 0000 and a candidate for clipping if the bitwise logical AND of the region code is 0000.

For a line in category 3 we proceed to find intersection point of the line with one of the boundaries of the clipping window, or to be exact with the infinite extension of one of the boundaries.

We choose an endpoint of the line say (x_1, y_1) that is outside the window, *i.e.* whose region code is not 0000. We then select an external line by observing that boundary lines, that are candidates for intersection are the ones for which the chosen endpoint must be "pushed across" so as to change as "1" in its code to "0".

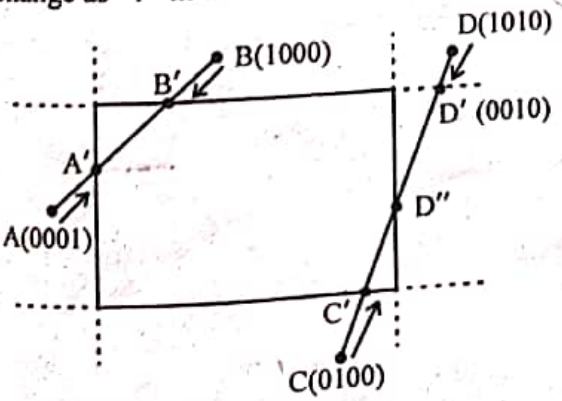


Fig. 3

This means:

if bit 1 is 1, intersect with line $y = y_{max}$
if bit 2 is 1, intersect with line $y = y_{min}$
if bit 3 is 1, intersect with line $x = x_{max}$
if bit 4 is 1, intersect with line $x = x_{min}$

Consider line CD in Fig. 3 if endpoint C is chosen, then the bottom boundary line $y = y_{min}$ is selected for computing intersection. On the other hand, if endpoint D is chosen, then either the top boundary line $y = y_{max}$ or the right boundary line $x = x_{max}$ is used.

The coordinates of the intersection point are

$x_i = x_{min}$ or x_{max}

if the line is vertical

$y_i = y_1 + m(x_i - x_1)$ or

$x_i = x_1 + m(y_i - y_1) / m$

if the boundary line is horizontal $y_i = y_{min}$ or y_{max}.

Where $m = (y_2 - y_1)/(x_2 - x_1)$ is the slope of the line.

Now we replace end point (x_1, y_1) with the intersection point (x_i, y_i) effectively eliminating the portion of the original line that is on the outside of the selected window boundary. The new endpoint is then assigned an update region code and the clipped line re-categorized and handled in the same way. This iterative process terminates when we finally reach a clipped line that belongs to either category 1 (visible) or category 2 (not visible).

Q.18 ***Derive composite transformation matrix of translation followed by reflection.*** *[R.T.U. 2016]*

Ans. Reflection is a transformation which generates the mirror image of an object. It is generated relative to an axis of reflection i.e, rotating the object 180° to the axis of reflection. If we stand in front of mirror we get reflection or the mirror image of us which is the virtual image. Mirror is the plane perpendicular to the xy-plane. Thus here reflection axis is perpendicular to the xy-plane. In other case, reflection axis can be chosen to be in xy-plane. Few cases of reflection are discussed here :

Case I : Reflection about x-axis

Transformation matrix depends on the 'a' and 'd' valves of the matrix. Which is shown in eq. (1).

The general purpose 2D transformation in homogenous coordinate representation is :

$$T = \begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix} \quad \ldots (i)$$

For x axis

$d = -1, a = 1$

Q.2 *Explain 3D projections and its types.* *[R.T.U. 2018, 2013]*

OR

What is perspective representation? Explain various types of perspective projection? *[R.T.U. 2015]*

Ans. 3D projections and its types : Viewing transformation for 3D are more complex than in 2D. One reason being the added third dimension and requirement for the 3D object to appear realistic in 2D display device. To map the 3D object on 2D display device, projection is performed. Projection is the process of transformation of point in 3D coordinate system to the points in 2D coordinate system. Thus, projection of a 3D object is defined by straight projection rays (called projectors) emanating from the centre of projection or COP passing through each point of the object and intersecting the projection plane. COP is a point in 3D space projections we are talking about here the planar geometric projections.

These are classified into perspective and parallel projections. This classification is based on whether rays coming from the object converge at the COP or not. For perspective projection, rays converge at COP and for parallel projection, rays do converge but at infinity. Centre of projection is also called perspective reference point (PRP). Parallel and perspective projections are further classified as follows :

Planar geometric projections

- Perspective
  - One point
  - two point
  - three point
- Parallel
  - Orthographic
    - Top plan
    - front elevation
    - side elevation
    - Axonometric
      - Isometric
  - Oblique parallel projection
    - cabinet
    - cavalier

Perspective Projection

Perspective projection is normal to the human eye. Distance from the center of projection to the projection plane is finite. Objects which are further appear smaller than the objects of the same size which are nearer. This is because of perspective foreshortening. It explains that the size of the perspective projection of the object varies inversely with the distance of the object from COP. Thus we don't get the real size of the object, though they appear realistic to our eyes. It follows the same concept of perspective projection.

The perspective projection of any set of parallel lines that are not parallel to the projection plane, converge to a point called vanishing point this is shown in fig.1

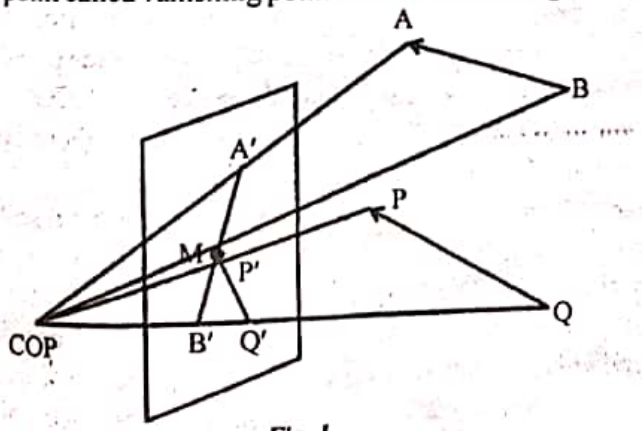


Fig. 1

Two parallel lines AB and PQ are projected as A'B' and P'Q' respectively. But A'B' and P'Q' intersect at M which is the projection of some point on AB as well as on PQ. Since AB||PQ and M, the vanishing point is the intersection of projections of AB and PQ at infinity.

Now, suppose the two parallel lines are also parallel to one of the axis, say x-axis, then the projection of the two lines would also be parallel which appear to meet somewhere at infinity called principle vanishing point. Though we say that parallel lines never meet even at infinity. Assume a railway track and road which are parallel to each other. But further away they seem to converge.

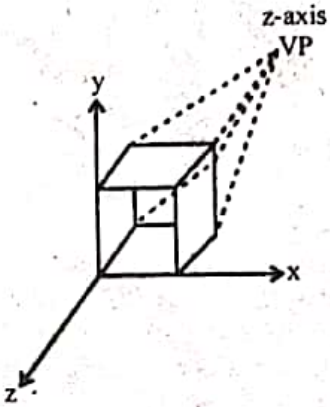


Fig. 2

In the fig. 2, a cube is shown in which lines are parallel to z-axis and appear to converge at vanishing point, so, it is z-axis vanishing point.

A perspective projection can have 3 principle vanishing points corresponding to three principal axes.

Mathematical description of a perspective projection

A perspective transformation is determined by center of projection (COP) and a viewing plane. Let P(x, y, z) be any point in 3D and centre of projection is at z_{prp}. Our purpose is to determined the projected point coordinates P'(x', y', z') on z = 0 plane, as shown in fig. 3

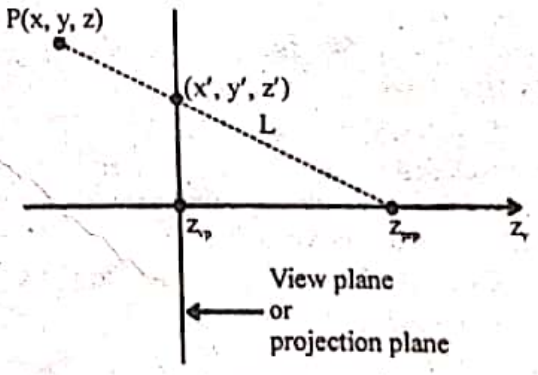


Fig. 3

z_{prp} is the centre of projection (x', y', z') are the projected points. View plane is at z_{vp}, z axis is intersecting view plane at z_{vp}. So z_{vp} is our one vanishing point along z-axis. Parametric form of equation for line L is

$$x' = x - xu$$
$$y' = y - yu$$
$$z' = z - (z - z_{prp})u, 0 \le u \le 1 \quad \ldots (1)$$

if u = 0, point is at position P(x, y, z)

if u = 1, point is at position $(0, 0, z_{prp})$ which is COP or perspective reference point (PRP).

As we can see from fig. 3 on the view plane,

$$z' = z_{vp}$$

So, from equation (1),

$$z_{vp} = z - (z - z_{prp})u$$

$$\text{or} \quad u = \frac{z_{vp} - z}{z_{prp} - z} \quad \ldots (2)$$

Substituting value of u in (1) we get,

$$x' = x - x\left(\frac{z_{vp} - z}{z_{prp} - z}\right) = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) \quad \ldots (3)$$

$$y' = y - yu$$

$$= y - y\left(\frac{z_{vp} - z}{z_{prp} - z}\right) = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right)$$

At we can see from the figure 3 that $(z_{prp} - z_{vp})$ is the distance between centre of projection and the projection plane.

Thus, the matrix representation for the new coordinates of perspective transformation is :

$$\begin{bmatrix} x'_h \\ y'_h \\ z'_h \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{vp}/(z_{prp} - z_{vp}) & z_{vp}(z_{prp})/(z_{prp} - z_{vp}) \\ 0 & 0 & -1/(z_{prp} - z_{vp}) & z_{prp}/(z_{prp} - z_{vp}) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \ldots (4)$$

The new coordinates are homogenous coordinates. The corresponding Cartesian coordinates are :

$$x' = \frac{x'_h}{h}, \; y' = \frac{y'_h}{h}$$

$$\text{where } h = \frac{z_{prp} - z}{z_{prp} - z_{vp}}$$

z-coordinates remains unchanged.

Our projection reference point or COP can be anywhere, on either side of the view plane. The case we discuss above was, when COP is somewhere along z_v axis. There are some other cases also :

Case I : When centre of projection (z_{prp}) is at the viewing coordinate origin, *i.e.*, $z_{prp} = 0$. It is shown in fig. 4.

From equation (3)

$$x' = x\left(\frac{z_{vp}}{z}\right)$$

$$y' = y\left(\frac{z_{vp}}{z}\right) \text{ for } z_{prp} = 0$$

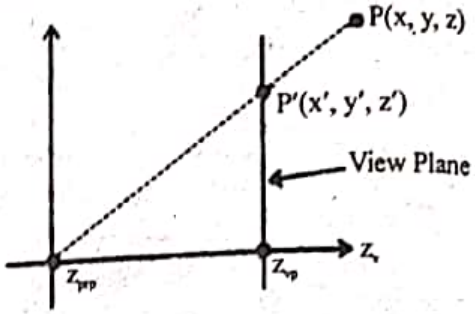


Fig. 4

Thus, matrix representation for perspective transformation would be :

$$M_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/z_{vp} & 0 \end{bmatrix} \quad \ldots (5)$$

Case II : When centre of projection is at arbitrary point. Steps that we have to follow are :

- Translate COP to the viewing coordinate origin,
- Apply general perspective transformation when $z_{prp} = 0$
- Translate to back to original COP.

Transformation matrix could be a composite matrix. Let COP be at some point (a, b, c) to translate it to the origin, matrix is

$$T(-a, -b, -c) = \begin{bmatrix} 1 & 0 & 0 & -a \\ 0 & 1 & 0 & -b \\ 0 & 0 & 1 & -c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

M_{per} is same matrix (5)

Third step is translation to the original COP

$$\text{So, } T(a, b, c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composite matrix, $M = T(a, b, c).\ M_{per}\ T(-a, -b, -c)$

We have discussed the cases when there is single principle vanishing point corresponding to one of the principle axis. This is because view plane is intersected by one principal axis. If more then one (two or three) principal axis intersect the view plane, we have two or three principal vanishing points respectively.

Parallel Projection

In parallel projection, rays coming from the object converge at infinity, *i.e.*, the distance from centre of projection to the projection plane is infinity. Therefore, projectors are parallel lines and we need to specify a direction of projection (DOP) or direction cosines. Parallel projection preserves the relative proportions of objects thus the view of object obtained is accurate but not realistic like perspective projection.

Parallel projection is classified as orthographic and oblique projection. This classification is on the basis of angle between direction of projection and projection plane.

For orthographic projection, this angle is of 90° and for other angles it is oblique parallel projection. Orthographic projection produce top plane view, front elevation and side elevation. It includes only two dimensions; length and width. Oblique projection includes three dimensions length, width and height. Fig. 5 shows the example of orthographic projection showing front view, side view and top view.

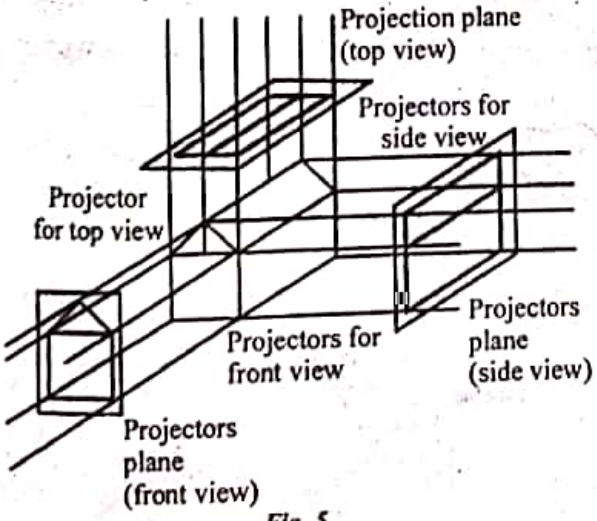


Fig. 5

Another type of orthographic projection is axonometric projection which use projection plane that are not normal to a principal axis, they therefore show multiple face of an object. Isometric projection is a special case of axonometric projection. Projection plane normal makes equal angles with each principal axis. All three principal axes are equally foreshortened allowing measurements along the axes to be made with the same scale. Though in general axonometric projection, these scaling factors may be different. In other words, for isometric projection, projection plane interests each coordinates axis, in which object is defined, at the same distance from the origin.

Transformation for orthographic parallel projection onto (say) xy-plane *i.e.*, $z = 0$ plane is :

$$x' = x,\ y' = y,\ z' = 0$$

Its matrix representation is :

$$P_{par} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In isometric projection, direction of projection $d = (d_1, d_2, d_3)$ is $(1, 1, 1)$ as it makes an equal angles with all three axes. Thus, the equation of the plane is : $x + y + z = 0$. Transformation for isometric projection specifically can be find out, where $P(x, y, z)$ point is projected to $P'(x', y', z')$ onto projection plane $x + y + z = 0$.

Parametric equation of line passing through $P(x, y, z)$ the direction of $d(1, 1, 1)$ is

$$P' = P + u.d \text{ where } -\infty < u < \infty$$

or $(x', y', z') = (x + y + z) + u(1, 1, 1)$

P' lies on $x + y + z = 0$

So, $(x + u) + (y + u) + (z + u) = 0$

or $3u = -(x + y + z)$

$$u = -\frac{1}{3}(x+y+z)$$

Thus,
$$x' = x - \frac{1}{3}(x+y+z) = (2x - y - z)/3$$

$$y' = y - \frac{1}{3}(x+y+z)$$

$$= (-x + 2y - z)/3$$

$$z' = z - \frac{1}{3}(x+y+z)$$

$$= (-x - y + 2z)/3$$

Transformation matrix in homogenous form is :

$$\begin{bmatrix} 2/3 & -1/3 & -1/3 & 0 \\ -1/3 & 2/3 & -1/3 & 0 \\ -1/3 & -1/3 & 2/3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic projections are generally used in engineering and architectural drawings.

Unlike orthographic projection, projection plane normal and direction of projection differ in oblique projection. Thus, the direction of projection is not perpendicular to the plane of projection.

Fig. 6 shows oblique and also orthographic projection.

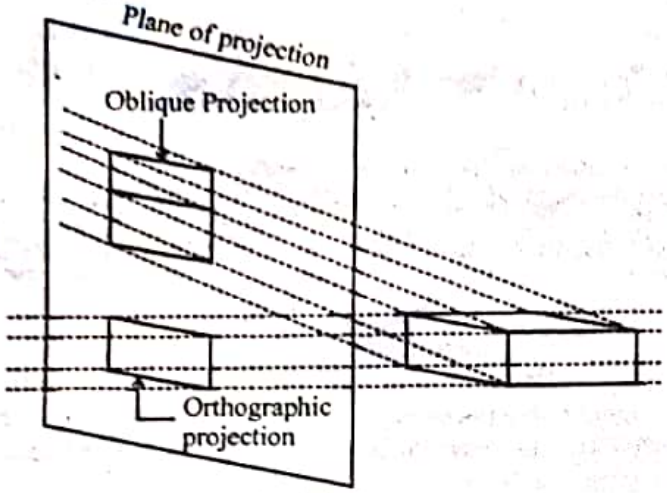


Fig. 6

Orthographic projection shows only front face in the fig. 6 that is , length and width. Oblique projection shows both the front face and top surface, that is, all the three dimensions. Thus, oblique projection shows all the dimensions is single view.

In oblique projection, projection of the face of the object which is parallel to the projection or view plane, preserves the size, angles and distances. Lines which are perpendicular to this plane are foreshortened *i.e.*, made shorter than actual lines, by the factor of DOP of rays.

General oblique projection of a point :

We are considering projection onto xy-plane, direction of projection (DOP) being $d = (d_1, d_2, d_3)$. Observe figure 1.

A′ is the projection of point A(0, 0, 1) onto xy-plane.

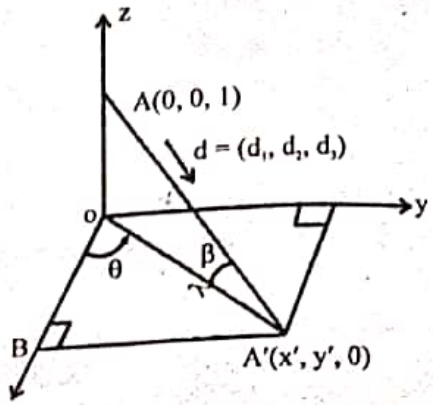


Fig. 7

Thus OA' is the projection of OA. θ is the angle between projected line OA' and positive x-axis.

β is the angle between DOP and xy-plane.

From triangle OA′B,

$$x' = l \cdot \cos\theta$$

$$y' = l \cdot \sin\theta$$

Thus, coordinates of P′: $(l \cdot \cos\theta, l \cdot \sin\theta, 0)$. Common choices for angle θ is 30° or 45°. There are two classifications for oblique projection, cavalier and cabinet. When angle (β) between top and plane of projection is 45°, it is cavalier projection. Length l depends on angle β and z-coordinate of the point to be projected.

Thus, $\tan\beta = \frac{z}{l}$

for $\beta = 45°, z = \ell$

This means, projection of line perpendicular to the projection plane has the same length as the line itself. In other case, when $\tan\beta = 2$ or $l = \frac{z}{2}$, it is cabinet projection and line projected has reduced to half the length of actual line. This makes cabinet projection more realistic than cavalier projection.

□□□

Q.15 ***Write a routine to convert RGB color model to HSV color model.*** *[R.T.U. 2012]*

Ans. Conversion Between HSV and RGB Models :
If HSV color parameters are made available to a user of a graphics package, these parameters are transformed to the RGB settings needed for the color monitor. To determine the operations needed in this transformation, we first consider how the HSV hexcone can be derived from the RGB cube. The diagonal of this cube from black (the origin) to white corresponds to the V axis of the hexcone. Also, each subcube of the RGB cube corresponds to a hexagonal cross-sectional area of the hexcone. At any cross section, all sides of the hexagon and all radial lines from the V axis to any vertex have the value V. For any set of RGB values, V is equal to the maximum value in this set. The HSV point corresponding to the set of RGB values lies on the hexagonal cross section at value V. Parameter S is then determined as the relative distance of this point from the V axis. Parameter H is determined by calculating the relative position of the point within each sextant of the hexagon. An algorithm for mapping any set of RGB values into the corresponding HSV values is given in the following procedure:

```
#include <math.h>
/* Input: h, s, v in range [0..1]
 Outputs; r, g, b in range [0..1] */
void hsvToRgb (float h, float s, float v, float * r, float
* g, float * b)
{ int i;
 float aa, bb, cc, f;
 if (s== 0) /* Grayscale */
```

```
        *r = *g = *b = v;
    else {
     if (h ==1.0) h = 0;
     h * = 6.0;
     i = ffloor (h) ;
     f = h – i;
     aa = v * (1– s);
     bb = v * (1– (s * f));
     cc = v * (1– (s * (1– f)));
     switch (i) {
     case 0: *r = v;  *g = cc; *b = aa; break;
     case 1: *r = bb; *g = v;  *b = aa; break;
     case 2: *r = aa;  *g = v;  *b = cc; break;
     case 3: *r = aa; *g = bb; *b = v; break;
     case 4: *r = cc; *g = aa; *b = v; break;
     case 5: *r = v; *g = aa; *b = bb; break;
     }
  } }
```

We obtain the transformation from HSV parameters to RGB parameters by determining the inverse of the equations in *rgbtohsv* procedure.

RGB to HSV Algorithm

```
1.  #include <math.h>
2.  #define MIN(a, b) (a < b? a : b)
3.  #define MAX(a, b) (a > b? a : b)
4.  #define No_H ∪ E – 1
5.  Void rgb to HSv (Float r, float g, float b, float *h,
    float *s, float *v)
6.  {
7.  float max = MAX (r, MAX (g, b)),
    Min = MIN(r, MIN (g, b));
8.  float delta = Max – Min;
9.  if (Max ! = 0.0)
10. *S = delta/max;
11. else
12. *S = 0.0;
13. if (*S = = 0.0) *h = No_H ∪ E;
14. else {
15. if (r = = max)
16. *h = (g – b)/delta;
17. else if (g = = max)
18. *h = 2 + (b – r)/delta;
19. else if (b = = max)
20. *h = 4  (r – g)/delta;
21. *h * = 60.0;
22. if (*h < 0) *h + = 360.0;
23. *h/ = 360.0;
24. }
25. }
```

Q.16 ***Explain dithering techniques in detail.***

Ans. Dithering Techniques : The term dithering is used in various contexts. Primarily, it refers to techniques for approximating halftones without reducing resolution, as pixel-grid patterns do. But the term is also applied to halftone-approximations methods using pixel grids, and sometimes it is used to refer to color halftone approximations only.

Random values added to pixel intensities to break up contours are often referred to as dither noise. Various algorithms have been used to generate the random distributions. The effect is to add noise over an entire picture, which tends to soften intensity boundaries.

Ordered-dither methods generate intensity variations with a one-to-one mapping of points in a scene to the display pixels. To obtain n^2 intensity levels, we set up an n by n dither matrix D_n, whose elements are distinct positive integers in the range 0 to $n^2 - 1$. For example, we can generate four intensity levels with

$$D_2 = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix} \quad \ldots (i)$$

and we can generate nine intensity levels with

$$D_3 = \begin{bmatrix} 7 & 2 & 6 \\ 4 & 0 & 1 \\ 3 & 8 & 5 \end{bmatrix} \quad \ldots (ii)$$

The matrix elements for D_2 and D_3 are in the same order as the pixel mask for setting up 2 by 2 and 3 by 3 pixel grids, respectively. For a bilevel system, we then determine display intensity values by comparing input intensities to the matrix elements. Each input intensity is first scaled to the range $0 \le I \le n^2$. If the intensity I is to be applied to screen position (x, y), we calculate row and column numbers for the dither matrix as

$$i = (x \bmod n) + 1, j = (y \bmod n) + 1 \quad \ldots (iii)$$

If $I > D_n(i,j)$, we turn on the pixel at position (x, y). Otherwise, the pixel is not turned on.

Q.12 ***Write the strategy for design of animation sequences. Explain animation function also.*** (5)

Ans. Design of Animation Sequences : In general, an animation sequence is designed with the following steps:

- Storyboard layout
- Object definitions
- Key-frame specifications
- Generation of in-between frames

This standard approach for animated cartoons is applied to other animation applications as well, although there are many special applications that do not follow this sequence. Real-time computer animations produced by flight simulators, for instance, display motion sequences in response to settings on the aircraft controls. And visualization applications are generated by the solutions of the numerical models. For frame-by-frame animation, each frame of the scene is separately generated and stored. Later, the frames can be recorded on film or they can be consecutively displayed in "real-time playback" mode.

The storyboard is an outline of the action. It defines the motion sequence as a set of basic events that are to take place. Depending on the type of animation to be produced, the storyboard could consist of a set of rough

sketches or it could be a list of the basic ideas for the motion.

An object definition is given for each participant in the action. Objects can be defined in terms of basic shapes, such as polygons or splines. In addition, the associated movements for each object are specified along with the shape.

A key frame is a detailed drawing of the scene at a certain time in the animation sequence. Within each key frame, each object is positioned according to the time for that frame. Some key frames are chosen at extreme positions in the action; others are spaced so that the time interval between key frames is not too great. More key frames are specified for intricate motions than for simple, slowly varing motions.

In betweens are the intermediate frames between the key frames. The number of in-betweens needed is determined by the media to be used to display the animation. Film requires 24 frames per second, and graphics terminals are refreshed at the rate of 30 to 60 frames per second. Typically, time intervals for the motion are set up so that there are from three to five in-betweens for each pair of key frames can be duplicated. For a 1-minute film sequence with no duplication, we would need 1440 frames. With five in-between for each pair of key frames, we would need 288 key frames. If the motion is not too complicated, we could space the key frames a little further apart.

There are several other tasks that may be required, depending on the application. They include motion verification, editing, and production and synchronization of a soundtrack. Many of the functions needed to produce general animations are now computer-generated. Fig. show example of computer-generated frames for animation sequences.

Fig. : Computer generated frame for animation sequence

General Computer Animation Functions : Some steps in the development of an animation sequence are well-suited to computer solution. These include object manipulations and rendering, camera motions, and the generations of in-betweens. Animation packages, such as Wave-front, for example, provide special functions for designing the animation and processing individual objects.

One function available in animation packages is provided to store and manage the object database. Object shapes and associated parameters are stored and updated in the database. Other object functions include those for motion generation and those for object rendering. Motions can be generated according to specified constraints using two-dimensional or three-dimensional transformations. Standard functions can then be applied to identify visible surfaces and apply the rendering algorithms.

Another typical function simulates camera movements. Standard motions are zooming, panning, and tilting. Finally, given the specifications for the key frames, the in-betweens can be automatically generated.

Q.13 ***(a) Define key-frame systems and also explain morphing process.***

(b) Explain different types of fractals and how to construct fractals and use of fractals in computer graphics?

Ans. (a) Key-Frame Systems : We generate each set of in-betweens from the specification of two (or more) key frames. Motion paths can be given with a kinematic description as a set of spline curves, or the motions can be physically based by specifying the forces acting on the objects to be animated.

For complex scenes, we can separate the frames into individual components or objects called cels (celluloid transparencies), an acronym from cartoon animation. Given the animation paths, we can interpolate the positions of individual objects between any two times.

With complex, object transformations, the shapes of objects may change over time. Examples are clothes, facial features, magnified detail, evolving shapes, exploding or disintegrating objects, and transforming one object into another object. If all surfaces are described with polygon meshes, then the number of edges per polygon can change from one frame to the next. Thus, the total number of line segments can be different in different frames.

Morphing : Transformation of object shapes from one