CS6504 COMPUTER GRAPHICS

Course Syllabus

L T P C

3 0 0 3

UNIT I INTRODUCTION 9

Survey of computer graphics, Overview of graphics systems – Video display devices, Raster scan systems, Random scan systems, Graphics monitors and Workstations, Input devices, Hard copy Devices, Graphics Software; Output primitives – points and lines, line drawing algorithms, loading the frame buffer, line function; circle and ellipse generating algorithms; Pixel addressing and object geometry, filled area primitives.

UNIT II TWO DIMENSIONAL GRAPHICS 9

Two dimensional geometric transformations – Matrix representations and homogeneous coordinates, composite transformations; Two dimensional viewing –viewing pipeline, viewing coordinate referenceframe; window-to-viewport coordinate transformation, Two dimensional viewing functions; clippingoperations – point, line, and polygon clipping algorithms.

UNIT III THREE DIMENSIONAL GRAPHICS 10

Three dimensional concepts; Three dimensional object representations – Polygon surfaces- Polygontables- Plane equations - Polygon meshes; Curved Lines and surfaces, Quadratic surfaces; Blobbyobjects; Spline representations – Bezier curves and surfaces - B-Spline curves and surfaces.

TRANSFORMATION AND VIEWING: Three dimensional geometric and modeling transformations –Translation, Rotation, Scaling, composite transformations; Three

dimensional viewing – viewing pipeline, viewing coordinates, Projections, Clipping; Visible surface detection methods.

UNIT IV ILLUMINATION AND COLOUR MODELS 7

Light sources - basic illumination models – halftone patterns and dithering techniques; Properties of light - Standard primaries and chromaticity diagram; Intuitive colour concepts - RGB colour model -YIQ colour model - CMY colour model - HSV colour model - HLS colour model; Colour selection.

UNIT V ANIMATIONS & REALISM 10

ANIMATION GRAPHICS: Design of Animation sequences – animation function – raster animation – key frame systems – motion specification –morphing – tweening.

REALISM: Tiling the plane – Recursively defined curves – Koch curves – C curves – dragons –space filling curves – fractals – Grammar based models – fractals – turtle graphics – ray tracing.

TOTAL: 45 PERIODS

TEXT BOOKS:

John F. Hughes, Andries Van Dam, Morgan Mc Guire ,David F. Sklar , James D. Foley, Steven

K. Feiner and Kurt Akeley ,"Computer Graphics: Principles and Practice", , 3rd Edition, Addison-

Wesley Professional,2013. (UNIT I, II, III, IV).

Donald Hearn and Pauline Baker M, "Computer Graphics", Prentice Hall, New Delhi, 2007 (UNIT V).

REFERENCES:

1. Donald Hearn and M. Pauline Baker, Warren Carithers,"Computer Graphics With Open GL", 4th Edition, Pearson Education, 2010.

2. Jeffrey McConnell, “Computer Graphics: Theory into Practice”, Jones and Bartlett Publishers, 2006.

3. Hill F S Jr., "Computer Graphics", Maxwell Macmillan” , 1990.

4. Peter Shirley, Michael Ashikhmin, Michael Gleicher, Stephen R Marschner, Erik Reinhard, Kelvin Sung, and AK Peters, Fundamental of Computer Graphics, CRC Press, 2010.

5. William M. Newman and Robert F.Sproull, “Principles of Interactive Computer Graphics”, Mc GrawHill 1978.

6. http://nptel.ac.in/

TABLE OF CONTENTS

Sl.no	Topic	Page No
A	Aim and Objective of the Subject	7
B	Detailed Lesson Plan	8
	Unit I Introduction	
C	Part A	11
D	Part B	14
1	DDA Line Drawing Algorithm	14
2	Video Display Devices	17
3	Bresenham‘S Line Drawing Algorithm	22
4	Fill Primitives	26
5	Midpoint Circle Drawing Algorithm	28
6	Bresenham‘S Ellipse Generating Algorithm	32
	Unit II Two Dimensional Graphics	
E	Part A	35
F	Part B	38
7	Line Clipping Algorithm	38
8	Cohen-Sutherland Line Clipping Algorithm	40
9	Point And Polygon Clipping Algorithm	42
10	Curve Clipping Algorithm	44
11	Sutherland-Hodgeman Polygon Clipping	45
12	Two Dimensional Geometric Transformations	47
	Unit III Three Dimensional Graphics	
G	Part A	53
H	Part B	56
13	Quadric And Polygon Surfaces	56
14	Three Dimensional Rotations	61
15	Visible Surface Detection Methods	63
16	Beizer And B-Spline Curves	70
17	3D Viewing	76
18	3D Transformation	79
	Unit IV Illumination And Colour Models	
I	Part A	82
J	Part B	86
19	RGB And HSV Color Model	86
20	Different Color Models	89
21	Halftone Patterns And Dithering Techniques	91
22	RGB And XYZ Chromaticity Diagrams.	93
23	Illumination Models	95
24	Phong,Warn Model	97
	Unit V Animations & Realism Animation Graphics	
K	Part A	99
L	Part B	102

AIM AND OBJECTIVE OF THE SUBJECT

- Gain knowledge about graphics hardware devices and software used.
- Understand the two dimensional graphics and their transformations.
- Understand the three dimensional graphics and their transformations.
- Appreciate illumination and color models.
- Be familiar with understand clipping techniques
- Apply two dimensional transformations.
- Design three dimensional graphics.
- Apply three dimensional transformations.
- Apply Illumination and color models.
- Apply clipping techniques to graphics.
- Design animation sequences.

Need and Importance for Study of the Subject

- This course is designed to provide a comprehensive introduction to computer graphics leading to the ability to understand contemporary terminology, progress, issues, and trends.
- A thorough introduction to computer graphics techniques, focusing on 3D modeling, image synthesis, and rendering.
- Geometric transformations, geometric algorithms, software systems ,3D object models (surface, volume and implicit), visible surface algorithms, image synthesis.
- The interdisciplinary nature of computer graphics is emphasized in the wide variety of examples and applications.

DETAILED LESSON PLAN

Sl. No.	UNIT	Topics/Portion to be covered	Hours Reqd	Cumulative	Book No.
UNIT-I : INTRODUCTION (9)					
1	Unit-1	Survey of computer graphics, Overview of graphics systems	1	1	T2
2		Video display devices, Raster scan systems, Random scan systems			T2
3		Graphics monitors and Workstations	1	2	T2
4		Input devices, Hard copy Devices, Graphics Software;	1	3	T2
5		Output primitives – points and lines,line drawing algorithms(DDA Algorithms)	1	4	T2
6		line drawing algorithms(Bresenham's Algorithm)	1	5	T2
7		loading the frame buffer, line function; circle generating algorithms	1	6	T2
8		Ellipse generating Algorithms	1	7	T2

9		Problems in circle and ellipse generating Algorithms	1	8	T2
10		Pixel addressing and object geometry, filled area primitives	1	9	T2
UNIT – II : TWO DIMENSIONAL GRAPHICS(9)					
11	Unit-2	Two dimensional geometric transformations	1	10	T2
12		Matrix representations and homogeneous coordinates, composite transformations;	1	11	T2
13,14		Two dimensional viewing – viewing pipeline, viewing coordinate reference frame	2	13	T2
15,16		widow-to-viewport coordinate transformation, Two dimensional viewing functions	1	14	T2
17		Clipping operations – point, line Clipping Algorithms	2	16	T2
18		Cohen Sutherland Line clipping Algorithm	1	17	T2
19		Clipping operations –polygon clipping algorithms.	1	18	T2
UNIT – III : THREE DIMENSIONAL GRAPHICS(10)					
20	Unit-3	Three dimensional concepts; Three dimensional object representations –Blobby objects	1	19	T2
21		Polygon surfaces- Polygon tables- Plane equations - Polygon meshes;	1	20	T2
22		Curved Lines and surfaces, Quadratic surfaces;	1	21	T2
23		Spline representations – Bezier curves and surfaces -B-Spline curves and surfaces	1	22	T2
24,25		Three dimensional geometric and modeling transformations –Translation, Rotation, Scaling, composite transformations;	2	24	T2
26		Three dimensional viewing – viewing pipeline, viewing coordinates	1	25	T2

27		Projections, Clipping	1	26	T2
28,29		Visible surface detection methods	2	28	T2
UNIT – IV : ILLUMINATION AND COLOUR MODELS (7)					
32,33	Unit-4	Light sources - basic illumination models –	1	29	T2
34		halftone patterns and dithering techniques	1	30	T2
35,36		Properties of light - Standard primaries and chromaticity diagram	1	31	T2
37		Intuitive colour concepts	1	32	T2
38,39		RGB colour model -YIQ colour model	1	33	T2
40		CMY colour model - HSV colour model	1	34	T2
41		HLS colour model			T2
42		Colour selection	1	35	T2
UNIT-V : ANIMATIONS & REALISM ANIMATION GRAPHICS(10)					
43	Unit-5	Design of Animation sequences, animation function – raster animation	1	36	T2
44,45		key frame systems – motion specification , morphing – tweening	2	38	T2
46,47		Tiling the plane – Recursively defined curves– Koch curves	2	40	T2
48		C curves – dragons –space filling curves	1	41	T2
49		Fractals – Grammar based models	1	42	T2
50		Turtle graphics	1	43	T2
51,52		Ray tracing	2	45	T2

Total Hours: 45

UNIT I

INTRODUCTION

Survey of computer graphics, Overview of graphics systems – Video display devices, Raster scan systems, Random scan systems, Graphics monitors and Workstations, Input devices, Hard copy Devices, Graphics Software; Output primitives – points and lines, line drawing algorithms, loading the frame buffer, line function; circle and ellipse generating algorithms; Pixel addressing and object geometry, filled area primitives.

1. **What are the merits and demerits of direct view storage tubes?(MAY/JUNE 2016)**

   Advantages:

   - Refreshing is not essential.
   - Without flicker, very complex pictures can be exhibit at very high resolution.

- Refreshing of screen is not required

Disadvantages:

- They normally never display color.
- Selected part of picture never removed.
- It can take quite a few seconds for composite pictures while redrawing and erasing process.
- It has poor contrast

2. **Give the contents of the display file.(NOV/DEC 2015)**

- **Picture** definition is now stored as a set of line drawing commands in an area of memory referred to as the refresh display file or simply the **refresh buffer**
- To display a specified picture, the system cycles through the **set of commands** in the display file, drawing each component line in turn.
- It contains a series of graphics commands that define an output image

3. **Compute the resolution of a 2 x 2 inch image that has 512 x 512 pixels**

(NOV/DEC 2015)

Resolution is calculated in terms of ppi.

Pixels per inch is ppi. Unfortunately this is usually called dpi

To calculate dpi:

dpi = pixels(height or width) divided by inches (height or width)

In this problem,

dpi = 512 / 2

= 256

4. **Define refresh/frame buffer. (MAY/JUNE 2016)**

Picture definition is stored in a memory area called the **refresh buffer** or **frame buffer**. This memory area holds the set of intensity values for all the screen points.

Frame buffer is where the image / picture generation information is stored in case of Video Display Monitors like CRT, Raster Scan, Random Scan, LCD, LED etc

5. **Define resolution (MAY/JUNE 2016)**

The maximum number of points that can be displayed without overlap on a CRT is referred to as the **resolution**.

Resolution is the number of points per centimeter that can be plotted horizontally and vertically, although it is often simply stated as the total number of points in each direction.

6. **Define aspect ratio. (MAY/JUNE 2013)**

Aspect ratio is the ratio of the vertical points to horizontal points necessary to produce equal length lines in both directions on the screen.

An aspect ratio of 3/4 means that a vertical line plotted with three points has the same length as a horizontal line plotted with four points.

7. **Identify the contrast between Raster and Vector Graphics APR/MAY 2015**

Raster Graphics	Vector Graphics
raster graphics are composed of pixels	vector graphics are composed of paths
raster image pixels do not retain their appearance as size increases	Vector images do retain appearance regardless of size

raster graphics are not scalable	Vector images are scalable

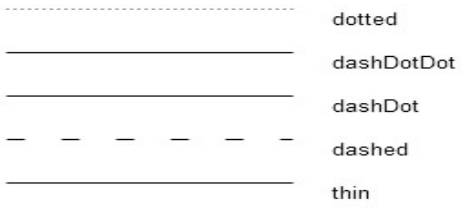
8. **List out few attributes of output primitives .(MAY/JUNE 2014)**

Attributes are the properties of the output primitives; that is, an attribute describes how a particular primitive is to be displayed. They include intensity and color specifications, line styles, text styles, and area-filling patterns. Functions within this category can be used to set attributes for an individual primitive class or for groups of output primitives.

9. **Write down any two line attributes. (AU NOV/DEC 2011)**

The basic attributes of a straight line segment are its:

- Type: solid, dashed and dotted lines.
- Width: the thickness of the line is specified.
- Color: a color index is included to provide color or intensity properties.



10. **Write down the attributes of characters.(AU MAY/JUNE 2012 IT)**

The appearance of displayed characters is controlled by attributes such as font, size, color and orientation. Attributes can be set both for entire character strings (text) and for individual characters defined as marker symbols. The choice of font gives a

particular design style. Characters can also be displayed as underlined, in boldface, in italics and in outline or shadow styles.

PART B

1.Explain in detail about the Line drawing DDA scan conversion algorithm with example . (MAY/JUNE 2016)

Algorithm

```
#define ROUND(a) ((int)(a+0.5))
voidlineDDA (int xa, int ya, int xb, intyb)
 {
    int dx = xb - xa, dy = yb - ya, steps, k;
    float xIncrement, yIncrement, x = xa, y = ya;
    if (abs (dx) > abs (dy) )
          steps = abs (dx) ;
    else steps = abs (dy);
    xIncrement = dx / (float) steps;
    yIncrement = dy / (float) steps ;
    setpixel (ROUND(x), ROUND(y) );
for (k=0; k<steps; k++)
   {
   x += xIncrement;
   y += yIncrement;
   setpixel (ROUND(x), ROUND(y));
   }
}
```

Algorithm Description:

Step 1: Accept Input as two endpoint pixel positions

Step2: Horizontal and vertical differences between the endpoint positions are assigned to parameters dx and dy (Calculate $dx=x_b-x_a$ and $dy=y_b-y_a$).

Step3: The difference with the greater magnitude determines the value of parameter steps.

Step4: Starting with pixel position (x_a, y_a), determine the offset needed at each step to generate the next pixel position along the line path.

Step 5: loop the following process for steps number of times a unit of increment or decrement in the x and y direction if x_a is less than x_b the values of increment in the x and y directions are 1 and m if x_a is greater than x_b then the decrements -1 and – m are used.

Example: Consider the line from (0,0) to (4,6)

$x_a=0$, $y_a=0$ and $x_b=4$ $y_b=6$

- $dx=x_b-x_a = 4-0 = 4$ and $dy=y_b-y_a=6-0= 6$
- x=0 and y=0
- 4 > 6 (false) so, steps=6
- Calculate xIncrement = dx/steps = 4 / 6 = 0.66 and yIncrement = dy/steps =6/6=1
- Setpixel(x,y) = Setpixel(0,0) (Starting Pixel Position)
- Iterate the calculation for xIncrement and yIncrement for steps(6) number of times
- Tabulation of the each iteration

RESULT:

k	x	Y	Plotting points (Rounded to Integer)
0	0+0.66=0.66	0+1=1	(1,1)

Fig1.1 Line plotting

Advantages of DDA Algorithm

- It is the simplest algorithm
- It is a is a **faster method** for calculating pixel positions

Disadvantages of DDA Algorithm

- Floating point arithmetic in DDA algorithm is still time-consuming
- End point accuracy is poor

2. Explain the following Video Display Devices (MAY/JUNE 2016)

(i) Refresh cathode ray tube
(ii) Raster scan systems
(iii) Random scan Displays
(iv) Color CRT Monitors

i) **Refresh cathode ray tube**

- Figure 1.2 illustrates the basic operation of a CRT. A beam of electrons (*cathode rays*), emitted by an electron gun, passes through focusing and deflection systems that direct the beam toward specified positions on the phosphor-coated screen.
- The phosphor then emits a small spot of light at each position contacted by the electron beam. Because the light emitted by the phosphor fades very rapidly.
- Some method is needed for maintaining the screen picture. One way to do this is to store the picture information as a charge distribution within the CRT.

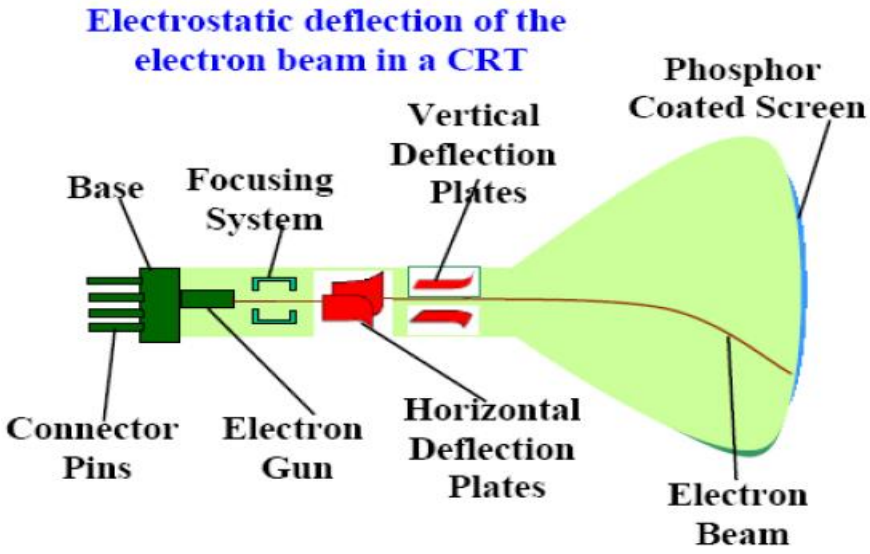


Fig 1.2 Refresh cathode ray tube

- This charge distribution can then be used to keep the phosphors activated. However, the most common method now employed for maintaining phosphor glow is to **redraw** the picture repeatedly by quickly directing the electron beam back over the same screen points.
- This type of display is called a **refreshCRT,** and the frequency at which a picture is redrawn on the screen is referred to as the **refresh rate.** The primary components of an electron gun in a CRT are the heated metal cathode and a control grid .

- Heat is supplied to the cathode by directing a current through a coil of wire, called the filament, inside the cylindrical cathode structure. This causes electrons to be ―boiled off the hot cathode surface.
- In the vacuum inside the CRT envelope, the free, negatively charged electrons are then accelerated toward the phosphor coating by a high positive voltage.
- The accelerating voltage can be generated with a positively charged metal coating on the inside of the CRT envelope near the phosphor screen, or an accelerating anode, as in Fig, can be used to provide the positive voltage

(ii) Raster Scan Systems

- The most common type of graphics monitor employing a CRT is the **raster-scan display,** based on television technology. In a raster-scan system, the electron beam is swept across the screen, one row at a time, from top to bottom.

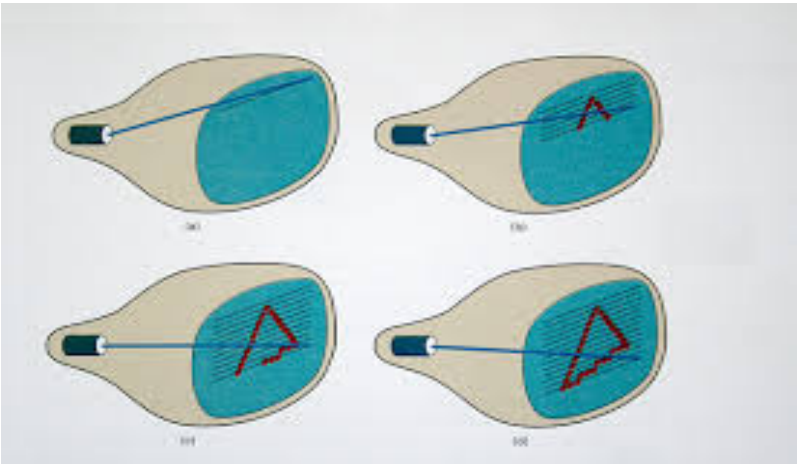


Fig 1.3 Raster Scan Systems

- Each row is referred to as a **scan line.** As the electron beam moves across a scan line, the beam intensity is turned on and off (or set to some intermediate value) to create a pattern of illuminated spots.

- Picture definition is stored in a memory area called the **refresh buffer** or **frame buffer,** where the term **frame** refers to the total screen area. This memory area holds the set of color values for the screen points.

- These stored color values are then retrieved from the refresh buffer and used to control the intensity of the electron beam as it moves from spot to spot across the screen. In this way the picture is painted on the screen one scan line at a time, as demonstrated in the above figure
- Each screen spot that can be illuminated by the electron beam is referred to as a **pixel** or **pel**(shortened forms of **picture element**). Since the refresh buffer is used to store the set of screen color values, it is also sometimes called a **color buffer.**
- **Home television** sets and printers are examples of other systems using raster-scan methods. Raster systems are commonly characterized by their resolution, which is the number of pixel positions that can be plotted.

(iii) Random Scan Displays

- When operated as a **random-scan display** unit, a CRT has the electron beam directed only to those parts of the screen where a picture is to be displayed.
- Pictures are generated as line drawings, with the electron beam tracing out the component lines one after the other. For this reason, random-scan monitors are also referred to as **vector displays** (or **stroke-writing displays** or **calligraphic displays**).
- The component lines of a picture can be drawn and refreshed by a random-scan system in any specified order (Fig.).A pen plotter operates in a similar way and is an example of a random-scan, hard-copy device.

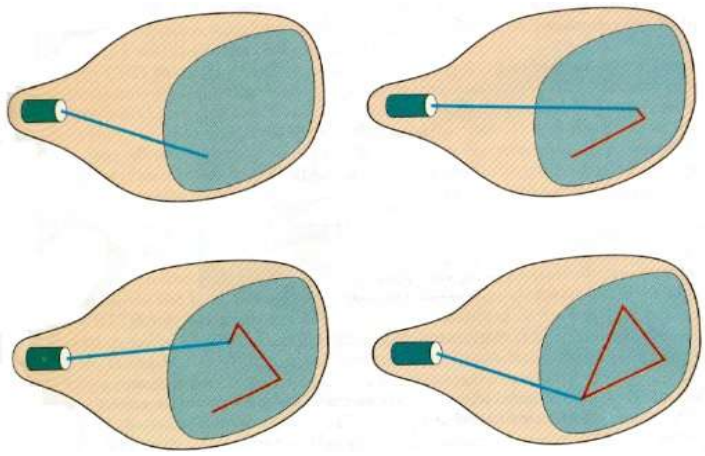


Fig 1.4 Random scan Displays

- Refresh rate on a random-scan system depends on the number of lines to be displayed on that system. Picture definition is now stored as a set of line-drawing commands in an area of memory referred to as the **display list, refresh display file, vector file,** or **display program.**
- To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn. After all line-drawing commands have been processed, the system cycles back to the first line command in the list.
- Random-scan systems were designed for line-drawing applications, such as architectural and engineering layouts, and they cannot display realistic shaded scenes.
- Vector displays generally have higher resolutions than raster systems.

(iv) Color CRT Monitors

- A CRT monitor displays color pictures by using a combination of phosphors that emit different-colored light. By combining the emitted light from the different phosphors, a range of colors can be generated.

- The two basic techniques for producing color displays with a CRT are the beam-penetration method and the shadow-mask method.
- The beam-penetration method for displaying color pictures has been used with random-scan monitors.
- Two layers of phosphor, usually red and green, are coated onto the inside of the CRT screen, and the displayed color depends on how far the electron beam penetrates into the phosphor layers.
- A beam of slow electrons excites only the outer red layer.
- A beam of very fast electrons penetrates through the red layer and excites the inner green layer. At intermediate beam speeds, combinations of red and green light are emitted to show two additional colors, orange and yellow.

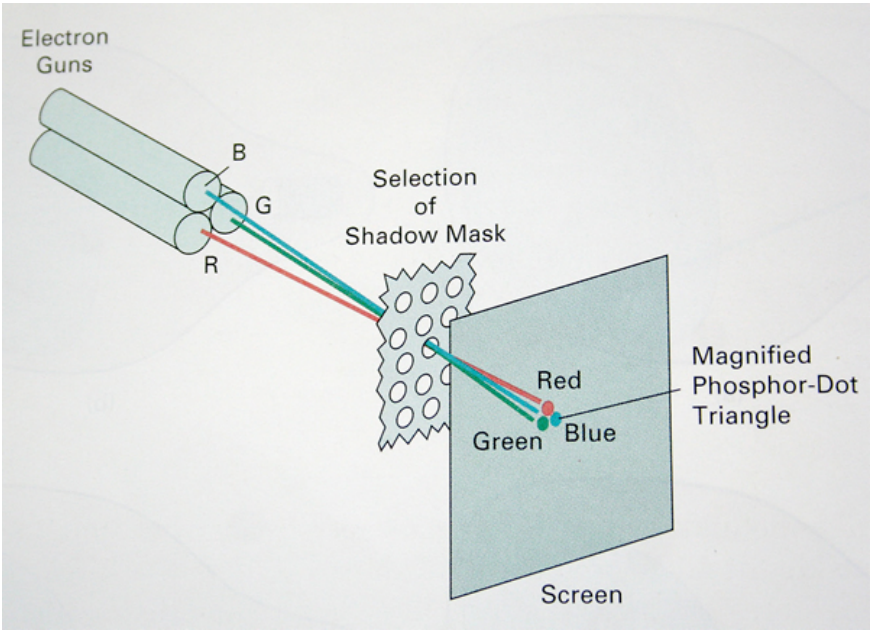


Fig 1.5 Color CRT Monitors

- The speed of the electrons, and hence the screen color at any point, is controlled by the beam-acceleration voltage. Beam penetration has been an inexpensive way to produce color in random-scan monitors, but only four colors are possible, and the quality of pictures is not as good as with other methods.

- A shadow-mask CRT has three phosphor color dots at each pixel position. One phosphor dot emits a red light, another emits a green light, and the third emits a blue light. This type of CRT has three electron guns, one for each color dot, and a shadow-mask grid just behind the phosphor-coated screen

3. Explain the Bresenham's line drawing algorithm with e.g.(MAY/JUNE 2012)

- Input the two line endpoints and store the left end point in (x0,y0)
  load (x0,y0) into frame buffer, ie. Plot the first point.
- Calculate the constants Δx, Δy, 2Δy and obtain the starting value for the decision parameter as P0 = 2Δy-Δx
- At each xkalong the line, starting at k=0 perform the following test
- If Pk< 0, the next point to plot is(xk+1,yk) and
  Pk+1 = Pk+ 2Δy
- Otherwise, the next point to plot is (xk+1,yk+1) and
  Pk+1 = Pk+ 2Δy - 2Δx 5. Perform step4 Δx times.

Implementation of Bresenham Line drawing Algorithm

```
voidlineBres (int xa,intya,intxb, int yb)
{
    int dx = abs( xa – xb) , dy = abs (ya - yb);
    int p = 2 * dy – dx;
    int twoDy = 2 * dy, twoDyDx = 2 *(dy - dx);
    int x , y, xEnd; /* Determine which point to use as
    start, which as end * / if (xa> x b )
    {
      x =xb; y= yb;
      xEnd = xa;
```

```
        }
           else
            {
             x = xa; y = ya;
             xEnd = xb;
             }
          setPixel(x,y);
          while(x<xEnd)
            {
             x++;
             if
            (p<0)
            p+=twoDy;
                 else
                 {
                  y++;
                  p+=twoDyDx;
                  }
             setPixel(x,y);
             }
}
```

Example : Consider the line with endpoints (20,10) to (30,18)

The line has the slope

M = (18-10)/(30-20)

=8/10

=0.8

Δx = 10

$\Delta y = 8$

The initial decision parameter has the value

$$p0 = 2\Delta y - \Delta x = 6$$

and the increments for calculating successive decision parameters are

$$2\Delta y = 16$$

$$2\Delta y - 2\Delta x = -4$$

We plot the initial point (x0,y0) = (20,10) and determine successive pixel positions along the line path from the decision parameter as

TABULATION:

k	Pk	(xk+1, yK+1)
0	6	(21,11)
1	2	(22,12)
2	-2	(23,12)
3	14	(24,13)
4	10	(25,14)
5	6	(26,15)
6	2	(27,16)
7	-2	(28,16)
8	14	(29,17)
9	10	(30,18)

RESULT:

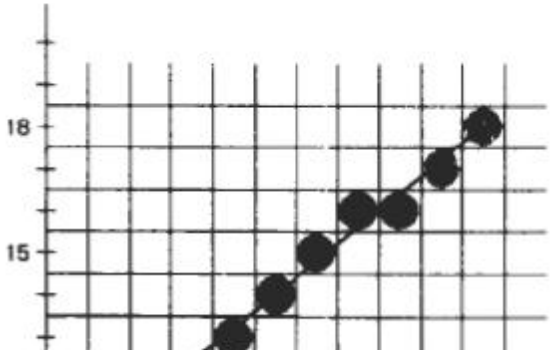


Fig 1.6 Line plotting points

Advantages

- Algorithm is Fast
- Uses only integer calculations

Disadvantages

- It is meant only for basic line drawing

4. Discuss the primitives used for filling. (NOV-DEC-2015)

AREA FILL ATTRIBUTES

- Options for filling a defined region include a choice between a solid color or a patterned fill and choices for the particular colors and patterns. these fill options can be applied to polygon regions or to areas defined with curved boundaries, depending on the capabilities of the available package. In addition, areas can be painted using various brush styles, colors, and transparency parameters.

FILL STYLES

- Areas are displayed with three basic fill styles:
- hollow with a color border, filled with a solid color, or **Wed** with a specified pattern or design. A basic fill style is selected in a **PHIGS** program with the function

Set InteriorStyle (fs)

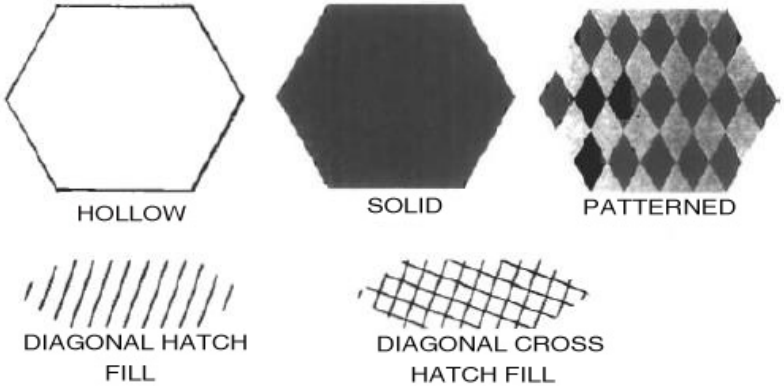


Fig 1.7 Area fill attributes

- Values for the fill-style parameter fs include hollow, ***solid,*** and pattern
- Another value for fill style is hatch, which is used to fill an area with selected hatching patterns-parallel **lines** or **crossed** lines.
- **As** with line attributes, a selected fillstyle value is recorded in the list of system attributes and applied to fill the interiors of subsequently specified areas. Fill selections for parameter f s are normally applied to polygon areas, but they can also be implemented to fill **regions** with curved boundaries.
- Hollow **areas** are displayed using only the boundary outline, with the interior color the same as the background color. A solid fill is displayed in a single color

up to and including the borders of the region. The color for a solid interior or for a hollow area outline is chosen with

setInteriorColourIndex (fc).

- where fillcolor parameter fc is set to the desired color code. **A** polygon hollow fill is generated with a linedrawing routine as a closed polyline. Solid fill of a region can be accomplished with the scan-line procedures
- Other fill options include specifications for the edge type, edge width, and edge color of a region.
- **These** attributes are set independently of the fill style or fill color, and they provide for the same options as the line-attribute parameters (line type, line width, and line color). That is, we can display area edges dotted or dashed, fat or thin, and in any available color regardless of how we have filled the interior.

Soft Fill:

- Modified boundary-fill and flood-fill procedures that are applied to repaint areas so that the fill color is combined with the background colors are referred to as soft-till or tint fill algorithms.
- One use for these fill methods is to soften the fill colors at object borders that have been blurred to antialias the edges.
- Another is to allow repainting of a color area that was originally filled with a semitransparent brush, where the current color is then a mixture of the brush color and the background colors "behind" the area.
- In either case, we want the new fill color to have the same variations over the area as the current fill color.
- The linear soft-fill algorithm repaints an area that was originally painted by merging a foreground color F with a single background color **B,** where $F \neq B$.

5. Explain the midpoint circle drawing algorithm. Assume 10 cm as the radius and co-ordinate origin as the center of the circle. (AU NOV/DEC 2011)

Midpoint Circle Algorithm

1. Input radius r and circle center (x, y,), and obtain the first point on the circumference of a circle centered on the origin as

$$(x_0,y_0) = (0,r)$$

2. calculate the initial value of the decision parameter as

$$P_0 = 5/4 - r$$

3. At each xk position, starting at k = *0,* perform the following test:

If pk *C* 0, the next point along the circle centered on (0,O) is (xk, yk) and

$$P_{k+1} = p_k + 2\ x_{k+1} + 1$$

Otherwise, the next point along the circle is (xk + 1, yk - 1) and

$$P_{k+1} = p_k + 2\ x_{k+1} + 1 - 2\ y_{k+1}$$

where 2xk+, = kt + 2 and 2yt+, = 2yt - 2.

4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x, y) onto the circular path centered on (x, yc) and plot the coordinate values:

x=x+xc,

y=y+yc

Repeat steps 3 through 5 until *x* >= y.

Example

Given a circle radius r=10

The circle octant in the first quadrant from x=0 to x=y. The initial value of the decision parameter is

$$P_0=1-r$$

$= -9$

For the circle centered on the coordinate origin, the initial point is

$(X_0, y0)=(0, 10)$

and initial increment terms for calculating the decision parameters are

$2x_0=0,$

$2y_0=20$

Successive midpoint decision parameter values and the corresponding coordinate positions along the circle path are listed in the following table.

TABULATION :

k	**p_k**	**(xk+1, yk-1)**	**2xk+1**	**2yk+1**
0	-9	(1,10)	2	20
1	-6	(2,10)	4	20
2	-1	(3,10)	6	20
3	6	(4,9)	8	18
4	-3	(5,9)	10	18
5	8	(6,8)	12	16
6	5	(7,7)	14	14

RESULT :

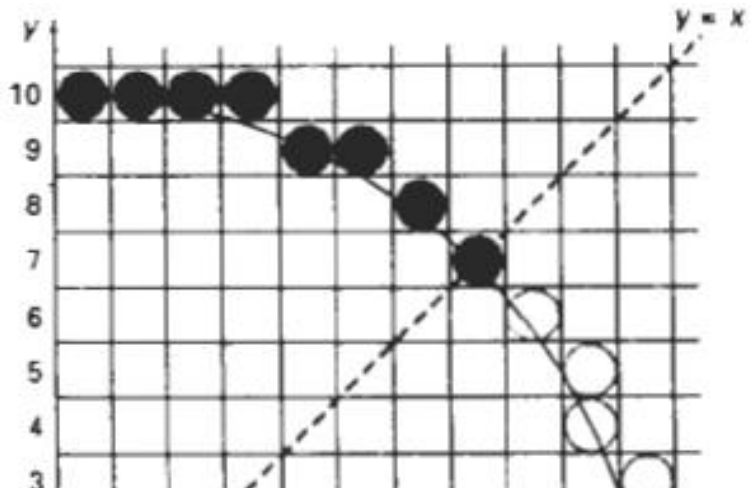


Fig 1.8 Circle plotting Points

Implementation of Midpoint Circle Algorithm

```
Void circleMidpoint (int xCenter, int yCenter, int radius)
{
        int x = 0; int y = radius;
        int p = 1 - radius;
          voidcirclePlotPoints (int, int, int, int); /* Plot first set of points */
          CirclePlotPoints (xCenter, yCenter, x, y);
          while (x < y)
          {
           x++ ;
           if (p < 0)
               p +=2*x +1;
               else
                   {
                   y--; p +=2* (x - Y) + 1;
                   }
             circlePlotPoints(xCenter, yCenter, x, y)
```

```
        }
}
Void circlePlotPolnts (int xCenter, int yCenter, int x, int y)
        {
        setpixel (xCenter + x, yCenter + y ) ;
        setpixel (xCenter - x, yCenter - y ) ;
        setpixel (xCenter + y, yCenter + x);
         setpixel (xCenter - y , yCenter + x);
         setpixel (xCenter t y , yCenter - x);
         setpixel (xCenter - x. yCenter + y);
        (xCenter + x, yCenter - y); y , yCenter - x);    }
```

6. Explain the basic concept of midpoint ellipse drawing algorithm. Give example.

(MAY/JUNE-2014)

Midpoint Ellipse Algorithm

- Input r_x,r_y and ellipse center (x_c,y_c) and obtain the first point on an ellipse centered on the origin as (x_0,y_0) = (0,r_y)
- Calculate the initial value of the decision parameter in region 1 as

$$\mathbf{P1_0 = ry^2 - r_x^2 r_y + (1/4) r_x 2}$$

At each xk position in region1 starting at k=0 perform the following test.

If P1k<0, the next point along the ellipse centered on (0,0) is (xk+1, yk)

and

$$\mathbf{p1_{k+1} = p1_k + 2\ r_y^2 x_{k+1} + r_y^2}$$

- Otherwise the next point along the ellipse is (x_{k+1}, y_{k-1}) and

$$\mathbf{p1_{k+1} = p1k + 2\ r_y^2 x_{k+1} - 2 r_x^2\ y_{k+1} + r_y^2}$$

with $2\ r_y^2 x_{k+1} = 2\ ry^2 x_k + 2r_y^2$ $2\ r_x^2 y_{k+1} = 2\ r_x^2 y_k + 2r_x^2$ And continue

until $2r_y^2\ x >= 2r_x^2\ y$

- Calculate the initial value of the decision parameter in region 2 using the last point (x0,y0) is the last position calculated in region 1.

$$p2_0 = r_y^2(x_0+1/2)2 + r_x^2(y_o-1)2 - r_x^2r_y^2$$

- At each position y_k in region 2, starting at k=0 perform the following test, If $p2_k>0$ the next point along the ellipse centered on (0,0) is (xk,yk-1) and

$p2_{k+1} = p_{2k} - 2r_x^2y_{k+1}+r_x^2$ Otherwise the next point along the ellipse is (x_{k+1},y_{k-1}) and

$p2_{k+1} = p2_k + 2r_y^2x_{k+1} - 2r_x^2y_{k+1}$ + rx2 Using the same incremental calculations for x any y as in region 1.

- Determine symmetry points in the other three quadrants.
- Move each calculate pixel position (x,y) onto the elliptical path centered on (xc,yc) and plot the coordinate values

$$x=x+x_c,\ y=y+y_c$$

Repeat the steps for region1 unit $2r_y^2x>=2r_x^2y$

EXAMPLE

Given input ellipse parameters r_x = 8 and r_y = 6, we illustrate the steps in the midpoint ellipse algorithm by determining raster positions along the ellipse path in the first quadrant. lnitial values and increments for the decision parameter calculationsare

$$2r_y^2x = 0 \quad \textbf{(with increment } 2r_y^2 = 72\textbf{)}$$

$$2r_x^2y = 2r_x^2r_y \quad \textbf{(withincrement-}2r_x^2\textbf{=-128)}$$

For region 1:

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

$$P1_0 = r_x^2 - r_x^2r_y + \tfrac{1}{4} r_x^2 = -332$$

Successive decision parameter values and positions along the ellipse path are calculated using the midpoint method as

k	$P1_k$	(x_{k+1},y_{k+1})	$2r_y^2x_{k+1}$	$2r_x^2y_{k+1}$
0	-332	(1,6)	72	768
1	-224	(2,6)	144	768
2	-44	(3,6)	216	768
3	208	(4,5)	288	640
4	-108	(5,5)	360	640
5	288	(6,4)	432	512
6	244	(7,3)	504	384

We now move out of region 1, since $\mathbf{2r_y^2x > 2r_x^2y}$

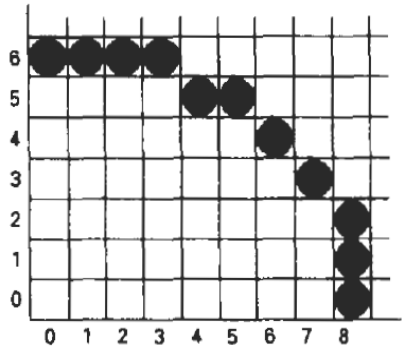


Fig:1.9 Positions along an elliptical path centered on the origin with r_x= 8 and r_y= 6 using the midpoint algorithm to calculate pixel addresses in the first quadrant.

For region 2:

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

$$P2_0 = f_{ellipse}(7+1/2,2)$$

= -151

Remaining positions along the ellipse path in the first quadrant are then calculated as

k	$P2_k$	(x_{k+1},y_{k+1})	$2r_y^2x_{k+1}$	$2r_x^2y_{k+1}$
0	-151	(8,2)	576	256
1	233	(8,1)	576	128
2	745	(8,0)	--	--

UNIT II

TWO DIMENTIONAL GRAPHICS

Two dimensional geometric transformations – Matrix representations and homogeneous coordinates, composite transformations; Two dimensional viewing – viewing pipeline, viewing coordinate reference frame; widow-to-viewport coordinate transformation, Two dimensional viewing functions; clipping operations – point, line, and polygon clipping algorithms.

PART-A

1. **Define Translation. (APR/MAY 2011)**

A translation is applied to an object by repositioning it along a straight line path from one co-ordinate location to another. We translate a two-dimensional point by adding translation distances, tx and ty, to original coordinate position (x,y) to move the point to a new position (x', y').

x' = x + tx,

y' = y + ty.

2. **Define Reflection.(APR/MAY 2012)**

A Reflection is a transformation that produces a mirror image of an object. The mirror image for a 2D reflection is generated relative to an axis of reflection by rotating the object 180 degree about the reflection axis.

3. **Reflect all the given triangle about X-axis whose coordinates are A(4,1) B(5,2) C(4,3) and find out the new coordinates. (MAY/JUNE 2016)**

Reflection about x-axis means

X= X

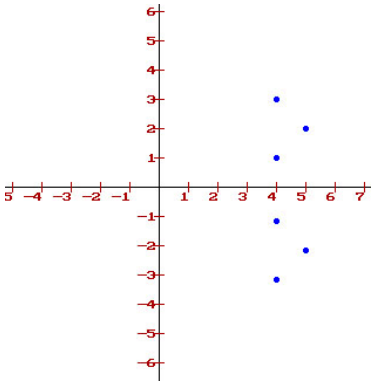
Y= -Y

The points are converted as

(4,1) = (4,-1)

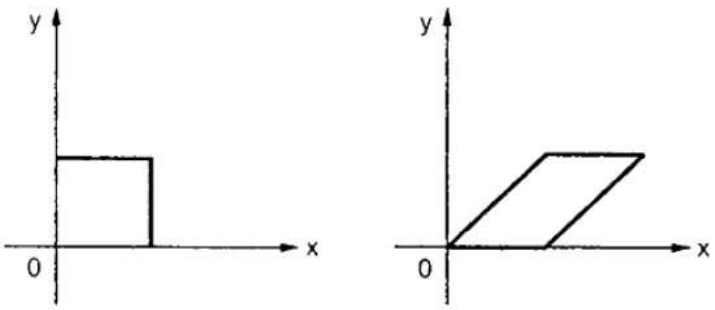
(4,3) = (4,-3)

(5,2) = (5,-2)



4. **Define Shear.(AU MAY 2013,DEC 2013)**

A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called a shear



5. **Define Window and view port.(AU DEC 2011)**

A world-coordinate area selected for display is called a window. An area on a display device to which a window is mapped is called a view port.

6. **Distinguish between window port and view port? (MAY/JUNE 2012)**

A portion of a picture that is to be displayed by a window is known as window port. The display area of the part selected or the form in which the selected part is viewed is known as view port.

7. **What is viewing transformation? (AU DEC 2011)**

The mapping of a part of a world-coordinate scene to device coordinates is referred to as viewing transformation.

8. **Define Clipping and Clip window. (AU MAY/JUNE 2014)**

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

9. **Write. down the condition for Point clipping in window.(AU NOV./DEC 2015)**

Assuming that the clip window is a rectangle in standard position, we save a point P = ***(x,* y)** for display if the following inequalities are satisfied:

$$xw_{min} \leq x \leq xw_{mxx}$$

$$yw_{min} \leq y \leq yw_{max}$$

Where the *edges of* the clip window ($xw_{min,}$ $xw_{mxx,}$ $yw_{min,}$ $yw_{max)}$ can be either the world-coordinate window boundaries or viewport boundaries. If any one of these four inequalities is not satisfied, the point is clipped (not saved for display)

10. **What is 'Shear' Transformation?(AU MAY/JUNE 2015)**

A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called a shear.

PART B

1. Explain Line clipping algorithm. (NOV/DEC 2012)

A line clipping procedure involves several parts. First, we can test a given line segment to determine whether it lies completely inside the clipping window. If it does not, we try to determine whether it lies completely outside the window. Finally, if we cannot identify a line as completely inside or completely outside, we must perform intersection calculations with one or more clipping boundaries.

We process lines through the "inside-outside" tests by checking the line endpoints. A line with both endpoints inside all clipping boundaries, such as the line from P_1 to P_2 saved.

A line with both endpoints outside any one of the clip boundaries is outside the window.All other lines cross one or more clipping boundaries, and may require calculation of multiple intersection points.

TCI minimize calculations, we try to devise Clipping algorithms that can efficiently identify outside lines and reduce intersection calculations.

For a line segment with endpoints (x_1,y_1) and (x_2,y_2) and one or both endpoints outside the clipping rectangle, the parametric representation.

$$\mathbf{x_1=x_1+u(x_2-x_1)}$$
$$\mathbf{y_1=y_1+u\ (y_2-y_1)} \qquad \mathbf{0\leq u\leq 1}$$

Could be used to determine values of parameter u for intersections with the clipping boundary coordinates. If the value of u for an intersection with a rectangle boundary edge is outside the range 0 to 1, the line does not enter the interior of through window at that boundary.

If the value of u is within the range from 0 to 1, the line segment does indeed cross into the clipping area. This method can be applied to each clipping boundary edge in turn to determine whether any part of the line segment is to be displayed. Line segments that are parallel to window edges can be handled as specia1 cases.

Clipping line segments with these parametric tests require a good deal of computation, and faster approaches to clipping are possible. Some algorithms are designed explicitly

for two-dimensional pictures and some are each adapted to three dimensional applications.

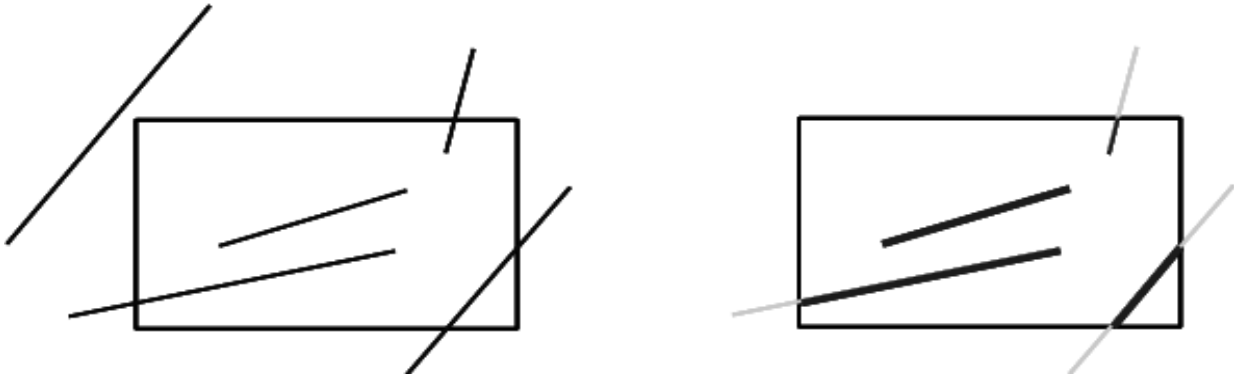


Fig.2.1 Before Clipping **After Clipping**

2. Explain in detail the Cohen-Sutherland line clipping algorithm with an example.
[NOV/DEC 2011], [MAY/JUNE 2012]

This is one of the oldest and most popular line-clipping procedures. Generally, the method speeds up the processing of line segments by performing initial tests that reduce the number of intersections that must he calculated.

Every line end point in a picture is assigned a four-digit binary code, called a region code that identifies the location of the point relative to the boundaries of the clipping rectangle. Regions are set up in reference to the boundaries as shown in figure below.

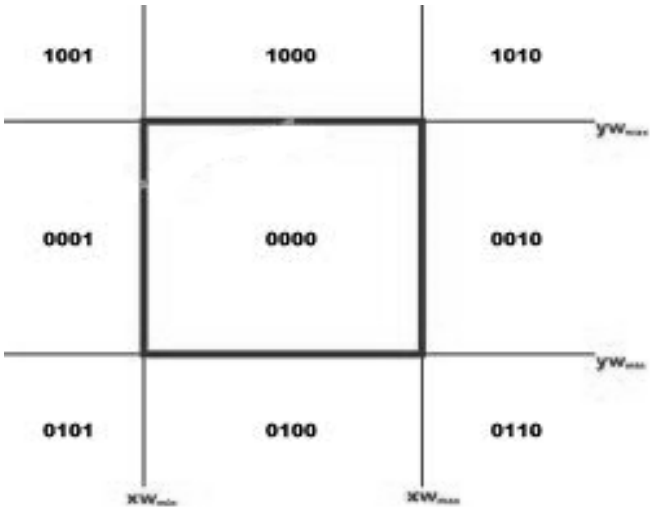


Fig 2.2 Region code

Each bit position in the region code is used to indicate one of the four relative coordinate positions of the point with respect to the clip window: to the left, right, top, or bottom.

By numbering the bit positions in the region code as **1** through 4 from right to left, the coordinate regions can be correlated with the bit positions as

bit **1:** left

bit 2: right

bit 3: below

bit 4: above

A value of 1 in any bit position indicates that the point is in that relative position; otherwise, the bit position is set to 0. If a point is within the clipping rectangle, the region code is 0000. A point that is below and to the left of the rectangle has a region code of 0101.

Bit values in the region code are determined by comparing endpoint coordinate values (x, y) to the clip boundaries.

Region-code bit values can be determined with the following two steps:

Calculate differences between endpoint coordinates and clipping boundaries.

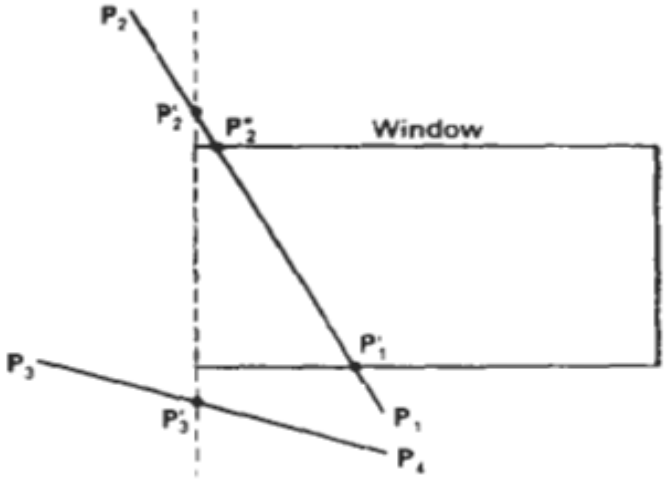


Fig 2.3 Line Clipping

1.To illustrate the specific steps in clipping lines against rectangular boundaries using the Cohen-Sutherland algorithm, we show how the lines in Figure could be processed.

2.Starting with the bottom endpoint of the line from P_1 to P_2 we check P_1 against the left, right, and bottom boundaries in turn and find that this point is below the clipping rectangle.

3.Then find the intersection point P_1^1 with the bottom boundary and discard the line section from P_1^1 to P_2.

4.The line now has been reduced to the section from P_1^1 to P_2 Since P_2 is outside the clip window, we check this endpoint against the boundaries and find that it is to the left of the window.

5.Intersection point P_2^1 is calculated, but this point is above the window. So the final intersection calculation yields P_2^{11} and the line from P_1^1 to P_2^{11} is saved.

6.This completes processing for this line, so we save this part and go on to the next line. Point P_3 in the next line is to the left of the clipping rectangle, so we determine the intersection P_3^1 and eliminate the line section from P_3 to P_3^1

7.By checking region codes for the line section from P_3^1 to P_4 we find that the remainder of the line is below the clip window and can be discarded also.

8.Intersection points with a clipping boundary can be calculated using the slope-intercept form of the line equation.

9.For a line with the point coordinates (x_1, y_1) and (x_2, y_2), they coordinate of the intersection point with a vertical boundary can be obtained with the calculation

$$\mathbf{y= y_1+m(x-x_1)}$$

10.If we are looking for the intersection with a horizontal boundary, the ***x*** coordinate can be calculated as $x= x_1+ \frac{y-y1}{m}$ with y set either to yw_{min} or to yw_{max}

3. Explain in detail Clipping Algorithms. [NOV/DEC 2012]

a.Point clipping algorithm.

Assuming that the clip window is a rectangle in standard position, we save a point P = ***(x,* y)** for display if the following inequalities are satisfied:

$$\mathbf{xw_{min} \leq x \leq xw_{max}}$$

$$\mathbf{yw_{min} \leq y \leq yw_{max}}$$

Where the *edges of* the clip window ($xw_{min,}$ $xw_{mxx,}$ $yw_{min,}$ $yw_{max)}$ can be either the world-coordinate window boundaries or viewport boundaries.

If any one of these four inequalities is not satisfied, the point is clipped (not saved for display). Although point clipping is applied less often than line or polygon clipping, some .applications may require a point clipping procedure.

For example, point clipping can be applied to scenes involving explosions or sea foam that are modelled with particles (points) distributed in some region of the scene.

b.Polygon clipping algorithm.

To clip polygons, we need to modify the line-clipping procedures discussed in the previous section.

A polygon boundary processed with a line clipper may be displayed as a series of unconnected line segments, depending on the orientation of the polygon to the clipping window. What we really want to display is a bounded area after clipping.

For polygon clipping, we require an algorithm that will generate one or more closed areas that are then scan converted for the appropriate area fill.

The output of a polygon clipper should be a sequence of vertices that defines the clipped polygon boundaries.

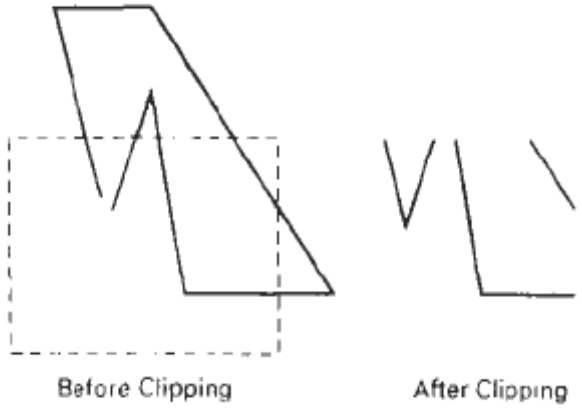


Fig.2.4 Display of a polygon processed by a line clipping algorithm

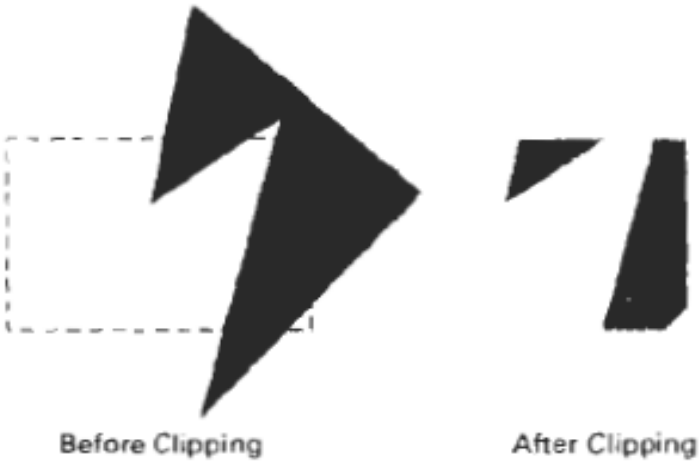


Fig.2.5 Display of a correctly clipped polygon

4. With suitable examples explain the Curve clipping algorithm.[NOV/DEC 2012]

a. Curve clipping algorithm

Areas with curved boundaries can be clipped with methods similar to those discussed in the previous .sections.

Curve-clipping procedures will involve nonlinear equations, however, and this requires more processing than for objects with linear boundaries.

The bounding rectangle for a circle or other curved object can be used first to test for overlap with a rectangular clip window. If the bounding rectangle for the object is completely inside the window, we save the object.

If the rectangle is determined to be completely outside the window, we discard the object. In either case, there is no further computation necessary. But if the bounding rectangle test fails, we can look for other computation-saving approaches.

For a circle, we can use the coordinate extents of individual quadrants and then octants for preliminary testing before calculating curve-window intersections.

For an ellipse, we can test the coordinate extents of individual quadrants. Figure (a) illustrates circle clipping against a rectangular window.

Similar procedures can be applied when clipping a curved object against a general polygon clip region.

On the first pass, we can clip the bounding rectangle of the object against the bounding rectangle of the clip region. If the two regions overlap, we will need to solve the simultaneous line-curve equations to obtain the clipping intersection points.

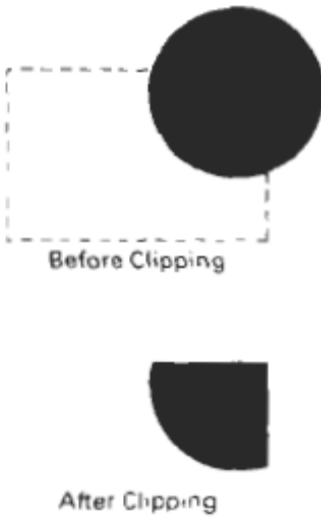


Fig.2.6 clipping a filled circle

5. a)Explain Sutherland-Hodgeman algorithm for polygon clipping.

We can correctly clip a polygon by processing the polygon boundary as a whole against each window edge.

This could be accomplished by processing all polygon vertices against each clip rectangle boundary in turn. Beginning with the initial set of polygon vertices, we could first clip the polygon against the left rectangle boundary to produce a new sequence of vertices.

The new set of vertices could then k successively passed to a right boundary clipper, a bottom boundary clipper, and a top boundary clipper, as in Figure. At each step, a new sequence of output vertices is generated and passed to the next window boundary clipper.

There are four possible cases when processing vertices in sequence around the perimeter of a polygon. As each pair of adjacent polygon vertices is passed to a window boundary clipper, we make the following tests:

(1) If the first vertex is outside the window boundary and the second vertex is inside, both the intersection point of the polygon edge with the window boundary and the second vertex are added to the output vertex list.

(2) If both input vertices are inside the window boundary, only the second vertex is added to the output vertex list.

(3) If the first vertex is inside the window boundary and the second vertex is outside, only the edge intersection with the window boundary is added to the output vertex list.

(4) If both input vertices are outside the window boundary, nothing is added to the output list. These four cases are illustrated in Figure for successive pairs of polygon vertices. Once all vertices have been processed for one clip window boundary, the output 11st of vertices is clipped against the next window boundary.

Fig.2.7 clipping a polygon against successive window boundaries

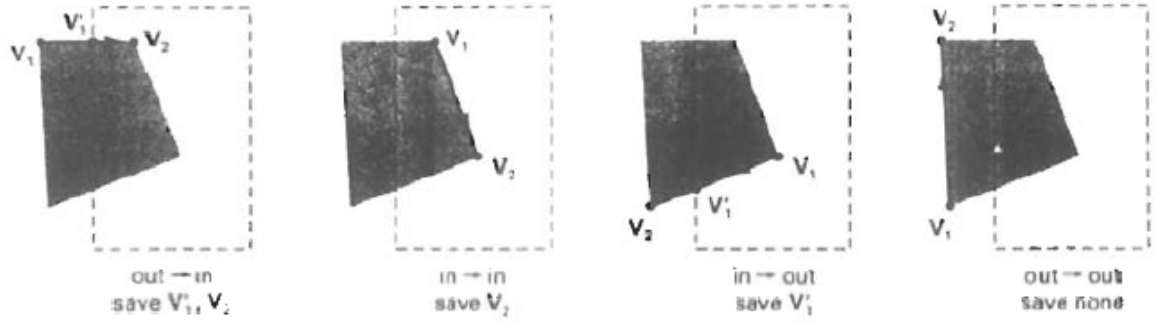


Fig.2.8 successive processing of pairs of polygon vertices against the left window boundary

Convex polygons are correctly clipped by the Sutherland-Hodgeman algorithm,

But concave polygons may be displayed with extraneous lines. This occurs when the clipped polygon should have two or more separate sections. But since there is only one output vertex list, the last vertex in the list is always joined to the first vertex. There are several things we could do to correctly display concave polygons. For one, we could split the concave polygon into two or more convex polygons and process each convex polygon

separately. Another possibility is to modify the Sutherland-Hodgeman approach to check the final vertex list for multiple vertex points along any clip window boundary and correctly join pairs of vertices.

5.(b) Explain about Two dimensional Geometric Transformations.

[NOV/DEC 2012]

Translation:

A translation is applied to an object by repositioning it along a straight-line path from one coordinate location to another. We translate a two-dimensional point by adding translation distances, *f,* and t,, to the original coordinate position (x, y) to move the point to a new position *(x ′ ,* y')

$$x' = x + t,,\ y' = y + t,$$

The translation distance pair (tx, ty) is called a translation vector or shift vector.

Translation matrices can be expressed as

$$P = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$P' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}$$

$$T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

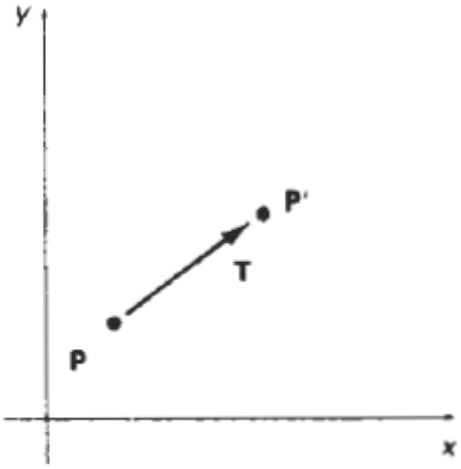
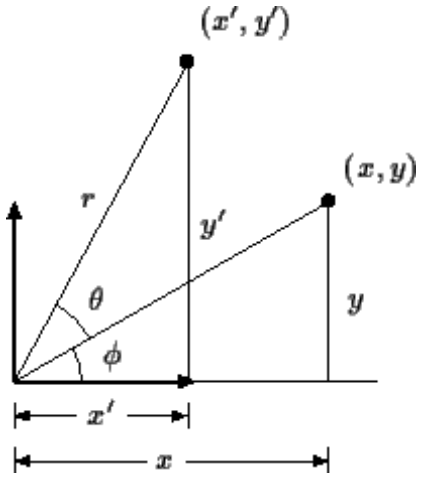


Fig.2.9 Translation

Rotation:

- **A** two-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 ***(x1,*** yl) of the rotation point (or pivot point) about which the object is to be rotated.



z

Fig 2.9.1 Rotation

The matrix representation is,

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

Scaling:

A scaling transformation alters the shape of an object. This operation can be carried out by multiplying the coordinate axes.

The transformed coordinates of x and y are formed by scaling factors sx and sy.

The matrix representation is,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Matrix representations & Homogeneous coordinates

Many graphics applications involve sequences of geometric transformations. An animation, for example, might require an object to be translated and rotated at each increment of the motion. In design and picture construction applications. The basic transformations can be expressed in the general matrix form

$$\mathbf{P^1 = M * P_1 + M_2}$$

With coordinate positions P and P' represented as column vectors. Matrix M_1 is a 2 by 2 array containing multiplicative factors, and M_2 is a two-element column matrix containing translational terms. For translation, M_1 is the identity matrix.

For rotation or scaling, M_2 contains the translational terms associated with the pivot point or scaling fixed point. To produce a sequence of transformations with these equations, such as scaling followed by rotation then translation, we must calculate the transformed coordinate's one step at i1 time. First, coordinate positions are scaled,

then these scaled coordinates are rotated, and finally the rotated coordinates are translated.*A* more efficient approach would be to combine the transformations so that the final coordinate positions are obtained directly from the initial coordinates, thereby eliminating the calculation of intermediate coordinate values. To be able to do this, we need to reformulate above equation to eliminate the matrix addition associated with the translation terms in M_2.We can combine the multiplicative and translational terms for two-dimensional geometric transformations into a single matrix representation by expanding the 2 by 2 matrix representations to 3 by 3 matrices.

This allows us to express all transformation equations as matrix multiplications, providing that we also expand the matrix representations for coordinate positions. To express any two-dimensional transformation as a matrix multiplication, we represent each Cartesian coordinate position (x,y) with the homogeneous coordinate triple (x_h , y_h ,h) where

$$\mathbf{x}=\frac{x_h}{h}$$

$$\mathbf{y}=\frac{y_h}{h}$$

Thus, a general homogeneous coordinate representation can also be written as (h*x, h*y. h). For two-dimensional geometric transformations, we can choose the homogeneous parameter h to be any nonzero value. Thus, there is an infinite number of equivalent homogeneous representations for each coordinate point (x, y).

(b)Composite transformations

With the matrix representations of the previous section, we can set up a matrix for any sequence of transformations as a composite transformation matrix by calculating the matrix product of the individual transformations.

Forming products of transformation matrices is often referred to as a concatenation, or composition, of matrices. For column-matrix representation of coordinate positions, we form composite transformations by multiplying matrices in order from right to left. That

is, each successive transformation matrix premultiplies the product of the preceding transformation matrices.

Translation:

If two successive translation vectors (t_{x1}, t_{y1}) and (t_{x2}, t_{y2}) are applied to a coordinate position P, the final transformed location P' is calculated as

$$P^1 = T(t_{x2}, t_{y2}) * [T(t_{x1}, t_{y1}) * P]$$
$$= [T(t_{x2}, t_{y2}) * T(t_{x1}, t_{y1})] * P$$

Where P and P^1 are represented as homogeneous-coordinate column vectors. We can verify this result by calculating the matrix product for the two associative groupings Also, the composite transformation matrix for this sequence of translations is

$$\begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx_1 + tx_2 \\ 0 & 1 & ty_1 + ty_2 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotations:

Two successive rotations applied to point **p** produce the transformed position.$P^1 = R(\theta_2) * (R(\theta_1)) * P = (R(\theta_2) * R(\theta_1)) * P$

By multiplying the two rotation matrices, we can verify that two successive rotations are additive: $R(\theta_2) * R(\theta_1) = R(\theta_1 + \theta_2)$,So that the final rotated coordinates can be calculated with the composite rotation matrix as $P^1 = R(\theta_1 + \theta_2) * P$

General Pivot point Rotation:

-Translate object so that the fixed point coincides with the coordinate origin.

-Rotate the object with respect to the coordinate origin.

-Use the inverse translation of step 1 to return the object to its original position

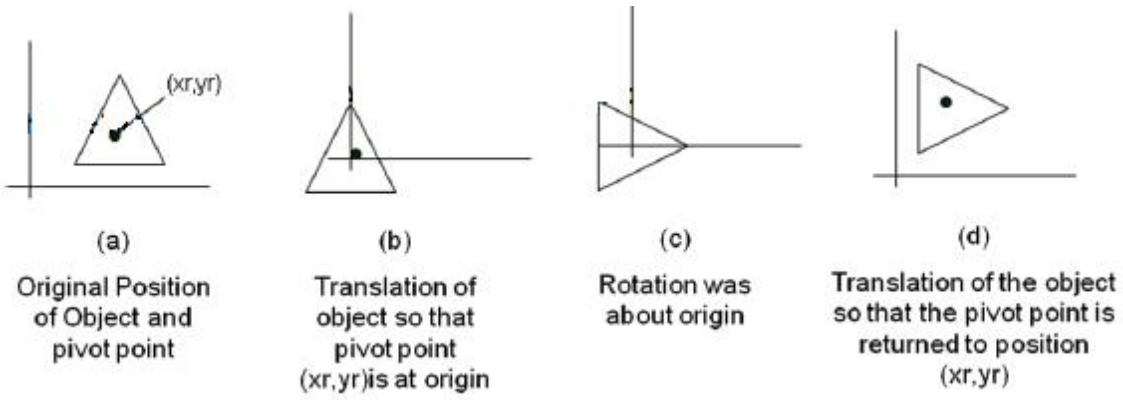


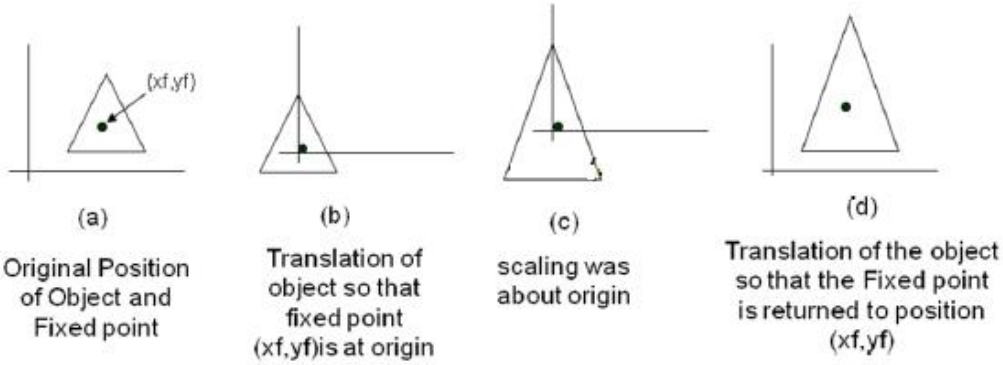
Fig 2.9.2 General Pivot point Rotation

Scalings:

Translate object so that the fixed point coincides with the coordinate origin.

Rotate the object with respect to the coordinate origin.

Use the inverse translation of step 1 to return the object to its original position



Concatenating transformation matrices for two successive scaling operations produces the following composite scaling matrix:

$$\begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x1} * s_{x2} & 0 & 0 \\ s_{y1} * s_{y2} & s_{x1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

UNIT III

THREE DIMENSIONAL GRAPHICS

Three dimensional concepts; Three dimensional object representations – Polygon surfaces- Polygon tables- Plane equations – Polygon meshes; Curved Lines and surfaces, Quadratic surfaces; Blobby objects; Spline representations – Bezier curves and surfaces -B-Spline curves and surfaces. TRANSFORMATION AND VIEWING: Three dimensional geometric and modeling transformations – Translation, Rotation, Scaling, composite transformations; Three dimensional viewing – viewing pipeline, viewing coordinates, Projections, Clipping; Visible surface detection methods.

PART A

1. **What are blobby objects? ?(MAY/JUNE 2011)**

Some objects do not maintain a fixed shape, but change their surface characteristics in certain motions or when in proximity with other objects. These objects are referred to as blobby objects, since their shapes show a certain degree of fluidness.

2. **Define Spline curves.(AU NOV/DEC 2011 & NOV/DEC 2012, MAY JUNE 2014)**

The term spline is a flexible strip used to produce a smooth curve through a designated set of points. In computer graphics, the term spline curve refers to any composite curve formed with polynomial sections satisfying specified continuity conditions at the boundary of the pieces.

3. **What are the advantages of B spline over Bezier curve? (AU MAY/JUNE 2013)**

- The degree of B-spline polynomial can be set independently of the number of control points
- B-Spline allow local control over the shape of a spline curve or surface
- A Bezier curve is a particular polynomial function, usually either cubic or quadratic, that defines a curve that goes from point A to point B given some control points in between. A Bezier spline is an aggregation of *n* of these.

4. **Differentiate parallel projection from perspective projection.(MAY/JUNE 2012,2016)**

Parallel Projection	**Perspective Projection**
In parallel projection, coordinate positions are transformed to the view plane along parallel lines.	In perspective projection, object positions are transformed to the view plane along lines that converge to a point called projection reference point or center of projection
Preserves the relative proportions of objects.	Produce realistic views but does not preserve relative proportions.
Used in drafting to produce scale drawings of 3Dobjects.	Projections of distant objects are smaller than the projections of objects of the same size that are closer to the projection plane.

5. **What is the need for space partitioning representations? (MAY/JUNE 2011)**

Space partitioning representations are used to describe interior properties, by partitioning the spatial region containing an object into a set of small non overlapping, contiguous solids. A common space partitioning description for a three object is an octree representation.

6. **What are the categories of visible surface detection algorithms? Give examples. (AU NOV/DEC 2015)**

- Back-Face detection
- Depth buffer Method
- A Buffer Method
- Scan line method
- Depth sorting methods
- BSP tree method

7. **How will you represent a curve in graphics? (AU NOV/DEC 2015)**

Curve can be generated from an input set of mathematical functions defining the objects or from a set of user specified points. When functions are specified, a package can project the defining equations for a curve to the display plane and plot pixel positions along the path of the projected plane.

8. **Define quadric surfaces. (AU NOV/DEC 2011)**

Quadric surfaces are described with second degree equations (quadrics). They include sphere, ellipsoids, tori, paraboloids and hyperboloids. Spheres and ellipsoids are common elements of graphic scenes; they are often available in graphics packages from which more complex objects can be constructed.

9. **What is critical fusion frequency (AU MAY/JUNE 2013)**

Frequency of light simulation at which it becomes perceived as a stable, continuous sensation. The frequency depends upon various factors like luminance, color, contrast etc.

PART B

1. **Write shot notes on**

(i) **Quadric surfaces**

(ii) **Polygon surfaces** **(MAY/JUNE 2016)**

Quadric Surfaces:

A frequently used class of objects is the quadric surfaces, which are described with second degree equations.

Sphere:

- A spherical surface with radius r centered on the coordinate origin is defined as a set of points(x, y, z) that satisfy the equation

$$x^2 + y^2 + z^2 = r^2$$

- In parametric form, using longitude and latitude angles

$$x = r\cos\emptyset\cos\theta,\ -\pi/2 \le \emptyset \le \pi/2$$

$$y = r\cos\emptyset\sin\theta,\ -\pi \le \theta \le \pi$$

$$z = r\sin\emptyset$$

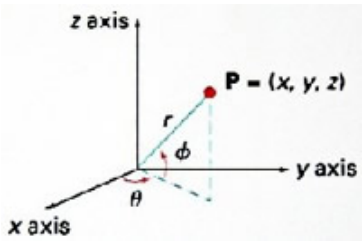


Figure 3.1 Parametric coordinate on the surface of a sphere

Ellipsoid:

- An ellipsoidal surface can be described as an extension of spherical surface, where radii in three mutually perpendicular directions can have different values.
- The Cartesian representation for points over the surface of an ellipsoid centered on the origin is

$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

And parametric representation,

$$x = r_x \cos\emptyset \cos\theta, \ -\pi/2 \le \emptyset \le \pi/2$$

$$y = r_y \cos\emptyset \sin\theta, \ -\pi \le \theta \le \pi$$

$$z = r_z \sin\emptyset$$

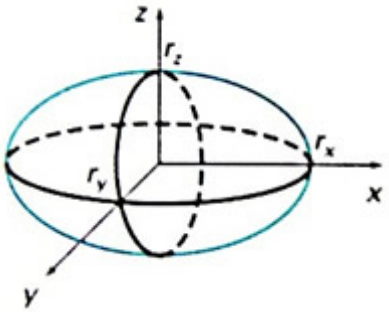


Figure 3.2: A ellipsoid with radii rx,ry,rz centered with coordinate origin

Torus:

- A torus is a doughnut-shaped object. It can be generated by rotating a circle or other conic about a specified axis.

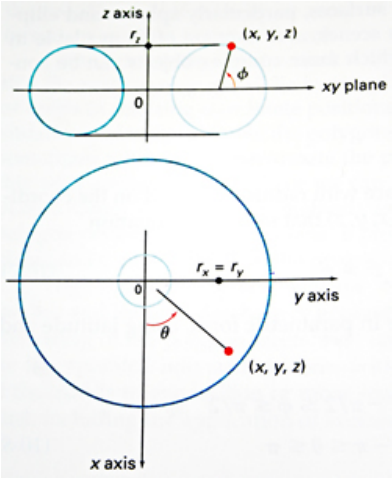


Figure 3.3 A torus with circular cross section

- The Cartesian representation for points over the surface of torus is

$$\left[r - \sqrt{\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2}\right]^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

Where r is any given offset value.

- In parametric form,

$$x = r_x(r + \cos\emptyset)\cos\theta\,, -\pi \le \emptyset \le \pi$$

$$y = r_y(r + \cos\emptyset)\sin\theta\,, -\pi \le \emptyset \le \pi,$$

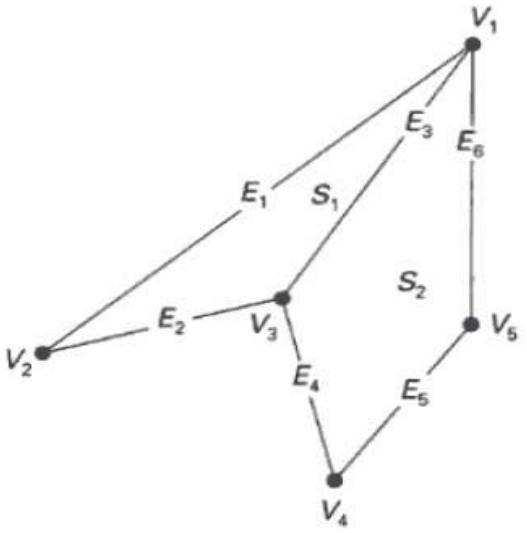
$$z = r_z \sin\emptyset$$

Polygon surfaces:

A polygon representation of a polyhedron defines the surface features of the object.

Polygon tables:

- Polygon data tables can be organized into two groups: Geometric tables and attribute tables.
- Geometric data tables contain vertex coordinates and parameters to identify the spatial orientation of the polygon surfaces.
- Attribute information includes parameters specifying the degree of transparency and its surface reflectivity and texture characteristics.
- Geometric data is to create three lists, a vertex table, an edge table and a polygon table.



VERTEX TABLE	
V_1:	x_1, y_1, z_1
V_2:	x_2, y_2, z_2
V_3:	x_3, y_3, z_3
V_4:	x_4, y_4, z_4
V_5:	x_5, y_5, z_5

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

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

Figure 3.4: Geometric data table reprsentation

- Vertex table stores the entire vertex in the object.
- The edge table contains pointers back to vertex table to identify the edges for each polygon
- The polygon table contains pointer back to the edge table to identify the edges for each polygon.
- Additional geometric information that is usually stored in data tables includes slope for each edge and coordinate extend for each polygon.

Plane Equations:

The equation of plane surface can be expressed in the form

$$Ax + By + Cz + D = 0$$

Where (x, y, z) is a point in the plane and coefficients A, B, C, D are constants.

By solving a set of three plane equations

$$\left(\frac{A}{D}\right)x_k + \left(\frac{B}{D}\right)y_k + \left(\frac{C}{D}\right)z_k = -1, k = 1,2,3$$

The solution is obtained by the determinant form,

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix}$$

$$B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

$$D = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

Expanding the determinants

$$A = y_1(z_1 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_3)$$

$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_3)$$

$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$

$$D = -x_1(y_2y_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)$$

$$If\ Ax + By + Cz + D < 0\ then\ point\ (x, y, z)\ is\ inside\ the\ surface$$

$$If\ Ax + By + Cz + D > 0\ then\ point\ (x, y, z) is\ outside\ the\ surface$$

Polygon Meshes:

- A single plane can be specified with a function such as fillArea.
- One type of polygon mesh is triangular strip.
- This function produces n-2 connected triangles.

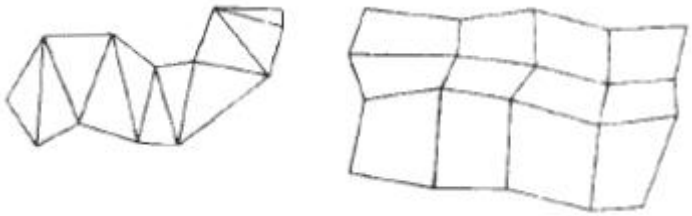


Figure 3.4: Triangular strips

2.How will you perform three dimensional rotations about any arbitrary axis in space? (MAY/JUNE 2016, NOV/DEC 2015)

To generate a rotation transformation of an object, the axis of rotation and the amount of angular rotation are needed.

Coordinate axes rotations

The two dimensional z axis rotation equations are easily extended to three dimensions

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

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

$z' = z$

Parameter Θ specifies the rotation angle

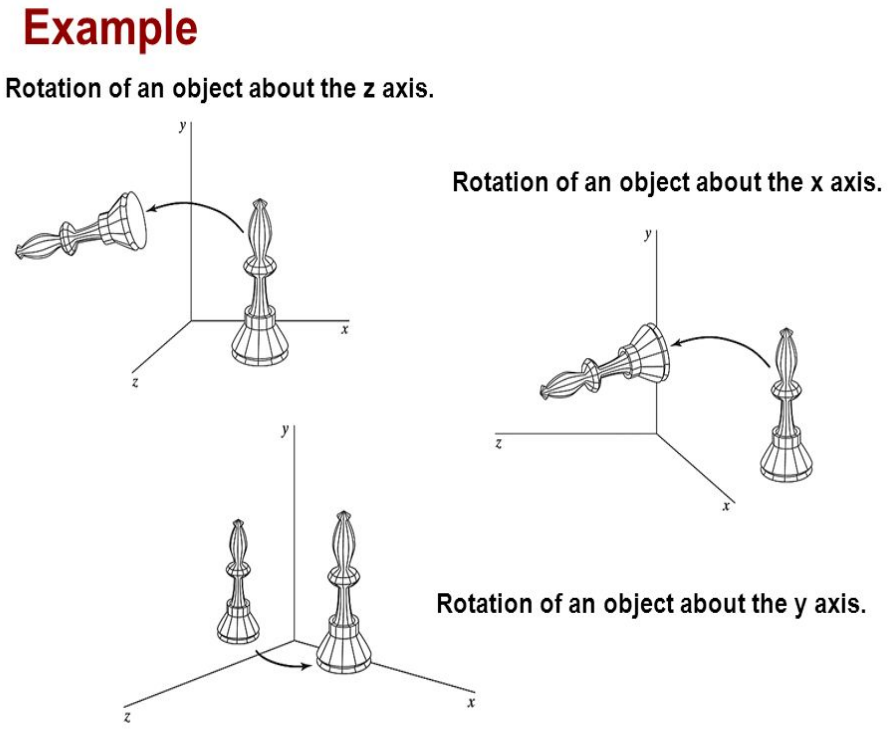


Figure 3.5 Three dimensional rotation

In homogenous coordinate form three dimensional z axis rotation equations are expressed as

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

It can be written more compactly as

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

Transformation coordinates for other two axes can be obtained with circular permutation of coordinate parameters x,y,z

Replacing $x \rightarrow y \rightarrow z \rightarrow x$

Equations for x axis as

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

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

$$x' = x$$

General Three dimensional Rotations:

When the object is rotated about an axis that is parallel to one of the coordinate axis, then the desired rotation can be obtained in following sequence

- Translate the object so that the rotation axis coincides with the parallel coordinate axis
- Perform the specified rotation about that axis
- Translate the object so that the rotation axis is moved to the original position.

When an object is rotated about an axis that is not parallel to one of the coordinate axis, then some additional transformations need to be performed.

- Translate the object so that rotation axis passes through the coordinate origin
- Rotate the object so that axis of rotation coincides with one of the coordinate axes.
- Perform the specified rotation about the coordinate axis.
- Apply inverse rotations to bring the rotation axis back to its original orientation
- Apply the inverse translation to bring the rotation axis back to its original position.

3. Discuss the visible surface detection methods in detail.

(NOV/DEC 2015,MAY/JUNE 2014, MAY/JUNE 2013)

Back Face detection:

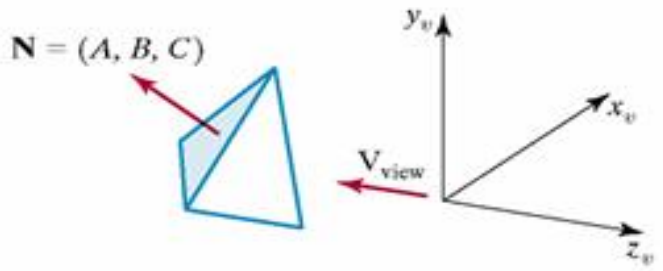
- A fast and simple object-space method for identifying the back faces of a polyhedron is based on the "inside-outside" tests.

- A point (x, y, z) is "inside" a polygon surface with plane parameters A, B, C, and D if When an inside point is along the line of sight to the surface, the polygon must be a back face
- This test can be done by considering the normal vector **N** to a polygon surface, which has Cartesian components (A, B, C).
- In general, if V is a vector in the viewing direction from the eye (or "camera") position, then this polygon is a back face if

  V.N > 0
- Furthermore, if object descriptions are converted to projection coordinates and your viewing direction is parallel to the viewing z-axis, then

  $V = (0, 0, V_z)$ and $V.N = V_zC$
- In a right-handed viewing system with viewing direction along the negative Z_v axis, the polygon is a back face if $C < 0$.

  Also, any face cannot be seen whose normal has z component C = 0, since your viewing direction is towards that polygon.



A polygon surface with plane parameter $C < 0$ in a right-handed viewing coordinate system is identified as a back face when the viewing direction is along the negative z_v axis.

Figure 3.6 polygon surface with plane parameters

Depth Buffer Method:

- It is an image-space approach. The basic idea is to test the Z-depth of each surface to determine the closest (visible) surface.
- In this method each surface is processed separately one pixel position at a time across the surface. The depth values for a pixel are compared and the closest (smallest z) surface determines the color to be displayed in the frame buffer.
- It is applied very efficiently on surfaces of polygon. Surfaces can be processed in any order. To override the closer polygons from the far ones, two buffers named frame buffer and depth buffer are used.
- **Depth buffer** is used to store depth values for (x, y) position, as surfaces are processed ($0 \leq$ depth ≤ 1).
- The **frame buffer** is used to store the intensity value of color value at each position (x, y).
- The z-coordinates are usually normalized to the range [0, 1]. The 0 value for z-coordinate indicates back clipping pane and 1 value for z-coordinates indicates front clipping pane.

Algorithm

1. Initialize the depth and refresh buffer
   Depthbuffer (x, y) = 0
   Framebuffer (x, y) = background color
2. For each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility.
   For each projected (x, y) pixel position of a polygon, calculate depth z.
   If Z > depthbuffer (x, y)
   set depthbuffer (x, y) = z,
   framebuffer (x, y) = surfacecolor (x, y)

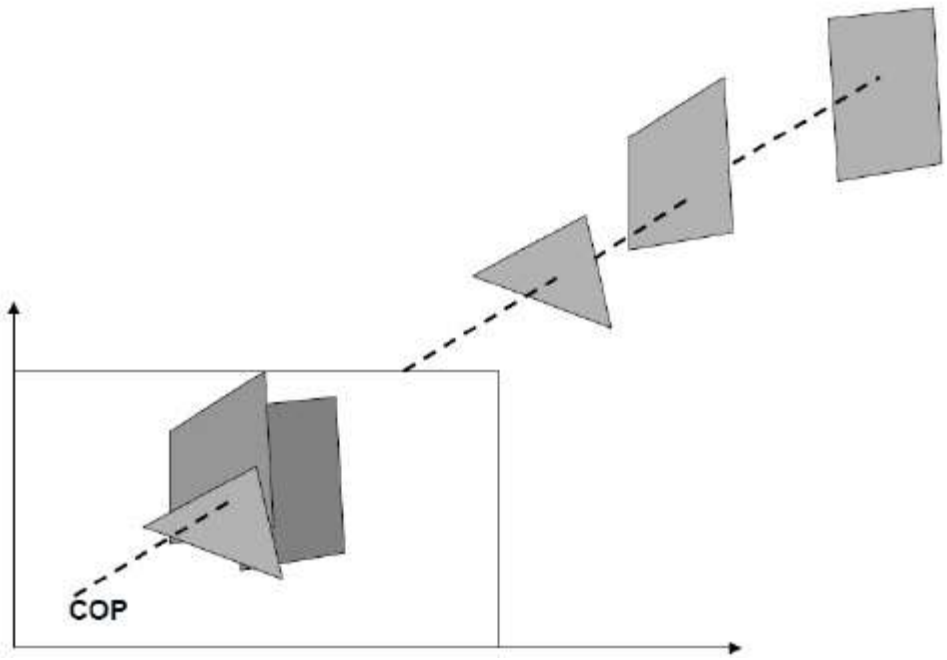


Figure 3.7 A view plane position

A buffer Method:

- The A-buffer method is an extension of the depth-buffer method.
- The A-buffer method is a visibility detection method developed at Lucas film Studios for the rendering system Renders Everything You Ever Saw (REYES).
- The A-buffer expands on the depth buffer method to allow transparencies. The key data structure in the A-buffer is the accumulation buffer.
- Each position in the A-buffer has two fields −
- **Depth field** − It stores a positive or negative real number
- **Intensity field** − It stores surface-intensity information or a pointer value

If depth >= 0, the number stored at that position is the depth of a single surface overlapping the corresponding pixel area. The intensity field then stores the RGB components of the surface color at that point and the percent of pixel coverage.

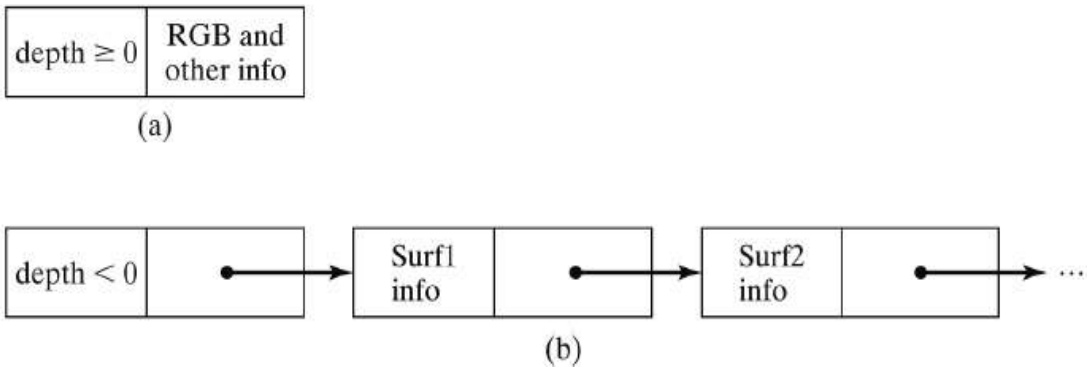


Fig.3.8 : Organization of an A buffer pixel position (a) Single-surface overlap (b) Multiple-surface overlap

- If depth < 0, it indicates multiple-surface contributions to the pixel intensity. The intensity field then stores a pointer to a linked list of surface data. The surface buffer in the A-buffer includes
  - RGB intensity components
  - Opacity Parameter
  - Depth
  - Percent of area coverage
  - Surface identifier

Scan line Method:

- It is an image-space method to identify visible surface.
- This method has depth information for only single scan-line. In order to require one scan-line of depth values, it is necessary to group and process all polygons intersecting a given scan-line at the same time before processing the next scan-line.
- Two important tables -edge table and polygon table**,** are maintained for this.
- **The Edge Table** − It contains coordinate endpoints of each line in the scene, the inverse slope of each line, and pointers into the polygon table to connect edges to surfaces.

- **The Polygon Table** − It contains the plane coefficients, surface material properties, other surface data, and may be pointers to the edge table.

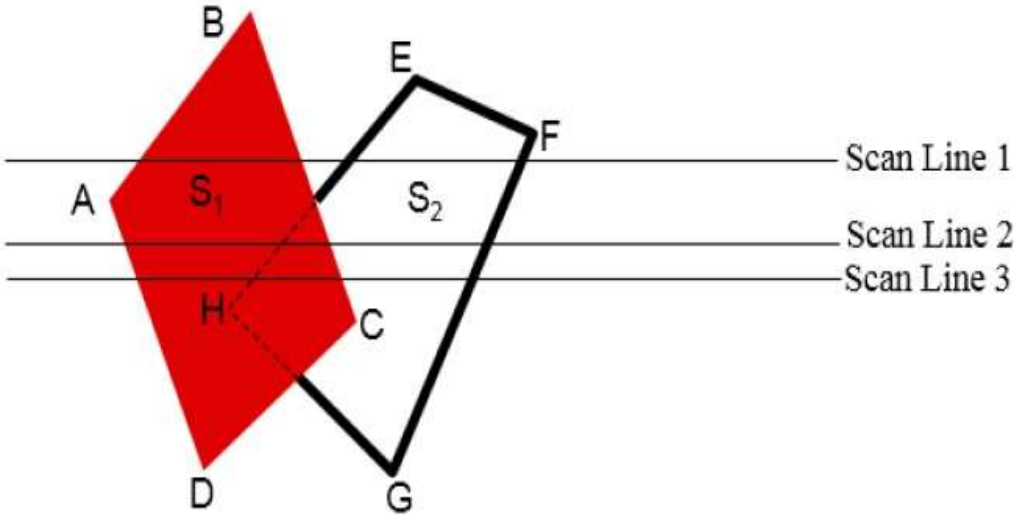


Fig.3.9 scan line intersecting a polygon surface

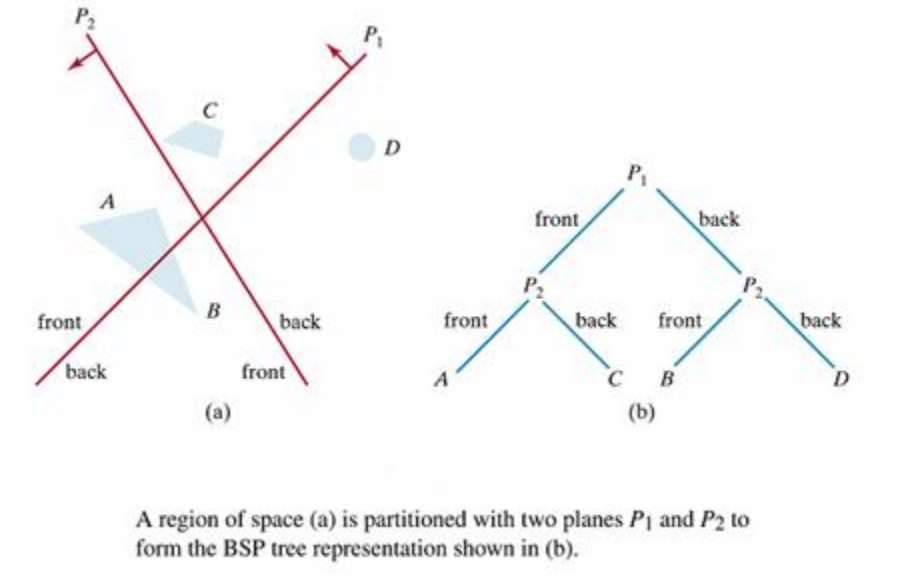
- To facilitate the search for surfaces crossing a given scan-line, an active list of edges is formed. The active list stores only those edges that cross the scan-line in order of increasing x.
- Also a flag is set for each surface to indicate whether a position along a scan-line is either inside or outside the surface.

Depth Sorting Method:

- Depth sorting method uses both image space and object-space operations. The depth-sorting method performs two basic functions
- First, the surfaces are sorted in order of decreasing depth.
- Second, the surfaces are scan-converted in order, starting with the surface of greatest depth.
- The scan conversion of the polygon surfaces is performed in image space. This method for solving the hidden-surface problem is often referred to as the painter's algorithm.

BSP Tree Method:

- Binary space partitioning is used to calculate visibility.
- To build the BSP trees, one should start with polygons and label all the edges. Dealing with only one edge at a time, extend each edge so that it splits the plane in two. Place the first edge in the tree as root.
- Add subsequent edges based on whether they are inside or outside. Edges that span the extension of an edge that is already in the tree are split into two and both are added to the tree.



A region of space (a) is partitioned with two planes P_1 and P_2 to form the BSP tree representation shown in (b).

Figure 3.10 A region partitioned into two planes

Area Subdivision Method:

- The area-subdivision method takes advantage by locating those view areas that represent part of a single surface.
- Divide the total viewing area into smaller and smaller rectangles until each small area is the projection of part of a single visible surface or no surface at all.
- Continue this process until the subdivisions are easily analyzed as belonging to a single surface or until they are reduced to the size of a single pixel.

- An easy way to do this is to successively divide the area into four equal parts at each step. There are four possible relationships that a surface can have with a specified area boundary.
  - **Surrounding surface** − One that completely encloses the area.
  - **Overlapping surface** − One that is partly inside and partly outside the area.
  - **Inside surface** − One that is completely inside the area.
  - **Outside surface** − One that is completely outside the area.

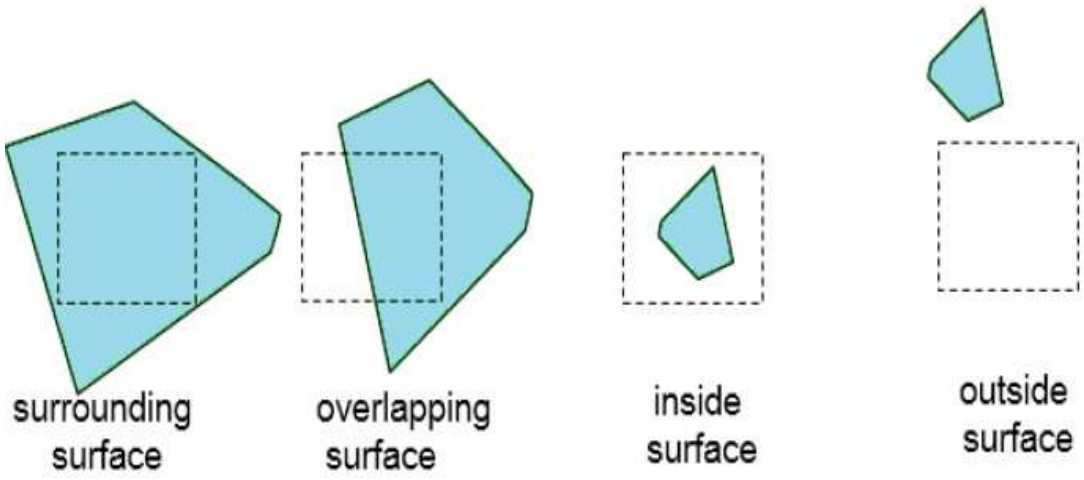


Fig.3.11 Possible relationship between polygon surfaces and rectangular area

4. Explain in detail

(i) **Parametric Equation for a cubic Beizer Curves**

(ii) **B-Spline Curves**

(iii) **Spline Representation** **(NOV/DEC 2015)**

i)Parametric equation for a Beizer curve:

Beizer curve can be fitted to any number of control points. The number of control points to be approximated and their relative positions determine the degree of the Beizer polynomial.

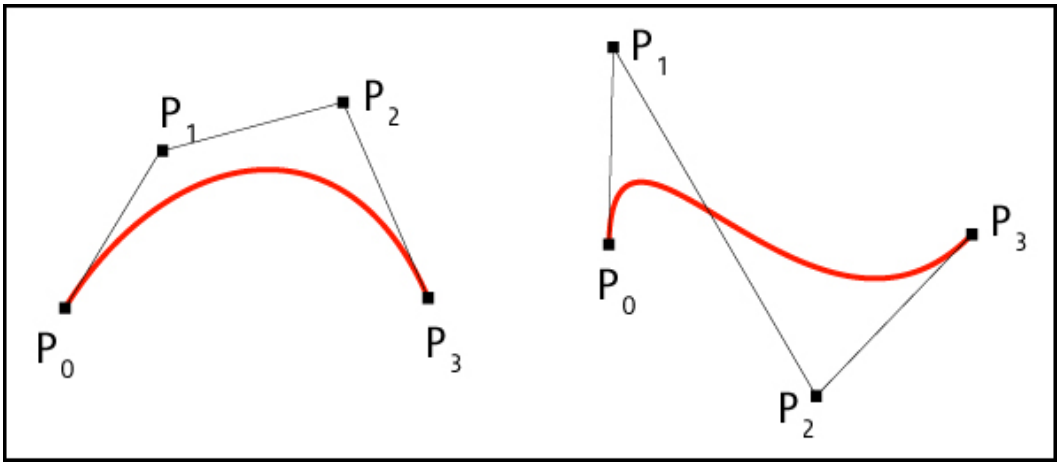


Figure 3.12 Beizer curve

Cubic Bezier curves:

Cubic Beizer curves are generated with four control points.

The four blending functions for cubic curves are

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

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

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

$$BEZ_{3,3}(u) = u^3$$

At the end positions of the cubic Bezier curve, the parametric first derivatives (slopes) are

$$P'(0) = 3(P_1 - P_0), \qquad P'(1) = 3(P_3 - P_2)$$

And the parametric second derivatives are

$$P''(0) = 6(P_0 - 2P_1 + P_2), \qquad P''(1) = 6(P_1 - 2P_2 + P_3)$$

By expanding the polynomial expression for blending functions

$$P(u) = [u^3\ u^2\ u\ 1].M_{Bez}.\begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

Where the **Bezier matrix** is

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

ii) B- Spline Curves:

A general expression for the calculation of coordinate positions along a B-spline curve in a blending function formulation as

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

Where p_k are an input set of n+ 1 control points and spline blending functions $B_{k,d}$ are polynomials of degree d-1, d can be chosen any integer from 2 up to n+1 control points. Blending functions for B-spline curves are define by Cox-deBoor recursion formulas:

$$B_{k,1}\ (u) = \begin{cases} 1, if\ u_{k \leq u \leq u_{K+1}} \\ 0, otherwise \end{cases}$$

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

Properties:

- Each basis function has precisely one maximum value, except for k=1.
- The maximum order of the curve is equal to the number of vertices of defining polygon.

- The degree of B-spline polynomial is independent on the number of vertices of defining polygon.
- B-spline allows the local control over the curve surface because each vertex affects the shape of a curve only over a range of parameter values where its associated basis function is nonzero.
- The curve exhibits the variation diminishing property.
- The curve generally follows the shape of defining polygon.
- Any affine transformation can be applied to the curve by applying it to the vertices of defining polygon.
- The curve line within the convex hull of its defining polygon.

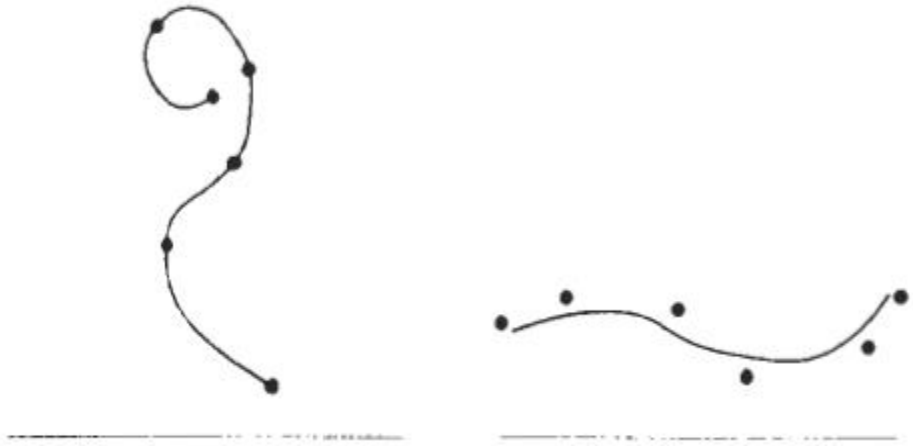
iii) Spline representation:

- Spline curve refers to any composite curve formed with polynomial sections satisfying specified continuity conditions at the boundary of the pieces.
- A spline surface is generated with two sets of orthogonal spline curves.

Interpolation and Approximation curves:

Spline curve is specified by set of coordinate positions called control points, which indicates the general shape of the curve.

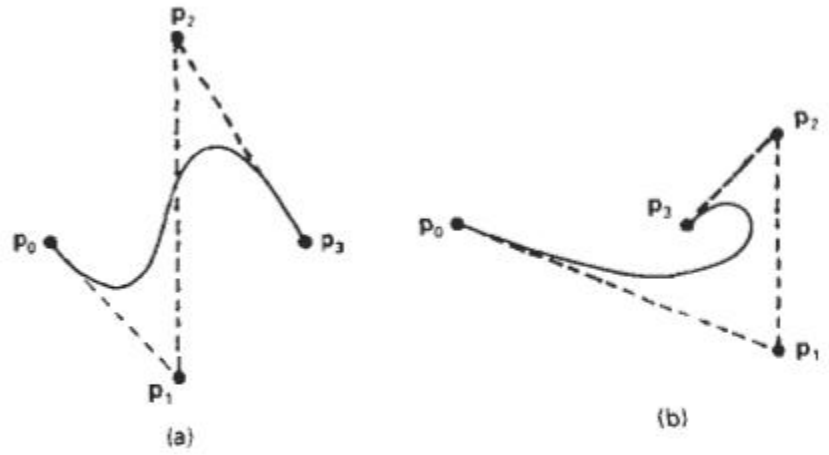
- When polynomial sections are fitted so that the curve passes through each control point, the resulting curve is said to interpolate the set of control points.
- When the polynomials are fitted to the general control path without necessarily passing through any control point, the resulting curve is said to approximate the set of control points.
- A spline curve is defined, modified and manipulated with operations on control points



A set of six control points interpolated with piecewise continuous polynomial sections.

A set of six control points approximated with piecewise continuous polynomial sections

Figure 3.13 Spline through control points



Control-graph shapes (dashed lines) for two different sets of control points.

Figure 3.14 Convex hull

- The convex polygon boundary that encloses a set of control points is called convex hull. A way to envision the shape of convex hull is to imagine rubber band stretched around the positions of the control points

Parametric continuity conditions:

To ensure smooth transition continuity conditions at connection points can be imposed.

$x = x(u), y = y(u), z = z(u), u_1 \leq u \leq u_2$

Figure 3.15 Parametric continuity

- Zero order parametric continuity means simply that the curves meet
- First order parametric continuity referred as C^1means that the first parametric derivatives are equal at the joining point.
- Second order parametric continuity referred as C^2means that both first and second order parametric derivatives are same at intersection.

Spline specifications:

There are three equivalent methods for specifying spline representation

- The set of boundary conditions that are imposed on the spline are stated.
- The matrixes that characterize the spline are stated.
- The set of blending functions are stated.

5. Explain the concepts of 3D viewing in detail. (NOV/DEC 2012)

Viewing Pipeline:

The steps for computer generation of a view of a three dimensional scene is somewhat like the process of taking a photograph. First, it is necessary to position the camera at a particular point in the space.

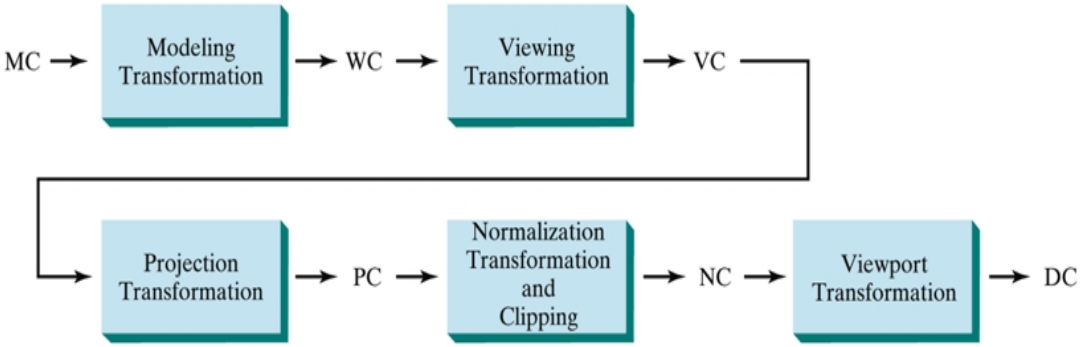


Figure 3. 16 Viewing pipeline

- Once the scene has been modeled, world coordinate positions are transformed to viewing coordinate position
- Next, projection operations are performed to convert the viewing coordinate description of the scene to coordinate position.
- Objects outside the view are clipped and remaining objects are processed using visible surface detection.

Viewing coordinates:

- A particular view of the scene is chosen by establishing the viewing coordinate of the system also called view reference coordinate system.
- A view plane or projection plane is then set up perpendicular to the viewing z_v axis.

- To establish the viewing coordinate reference frame, a world coordinate position called view reference point is chosen.

Transformation from World to viewing coordinates:

The transformation sequence is

- Translate the view reference point to the origin of the world coordinate system.
- Apply rotations to align the x_v, y_v, z_v axes with the world x_w, y_w, z_w axes respectively.

Projections:

- Once the world coordinate description of the scene is converted to viewing coordinates, then the three dimensional objects are projected on to the two dimensional view plane.
- In a parallel projection, coordinate positions are transformed to the view plane along parallel lines.
- In perspective projection object positions are transformed to view plane along the line converge to appoint called projection reference point (center of projection).

Parallel projections

- Parallel projection can be specified with a projection vector that defines the direction for the projection lines.
- When the projection is perpendicular to the view plane then the projection obtained is orthographic parallel projection. Otherwise it is oblique parallel projection.
- Orthographic projections are most often used to produce the front, side and top views of an object. Front rear and side projections are called elevations. And top orthographic projection is called plan view.

- Orthographic projections that display more than one face of an object can also be formed. Such views are called axonometric orthographic projections. The most commonly used axonometric projection is the isometric projection.

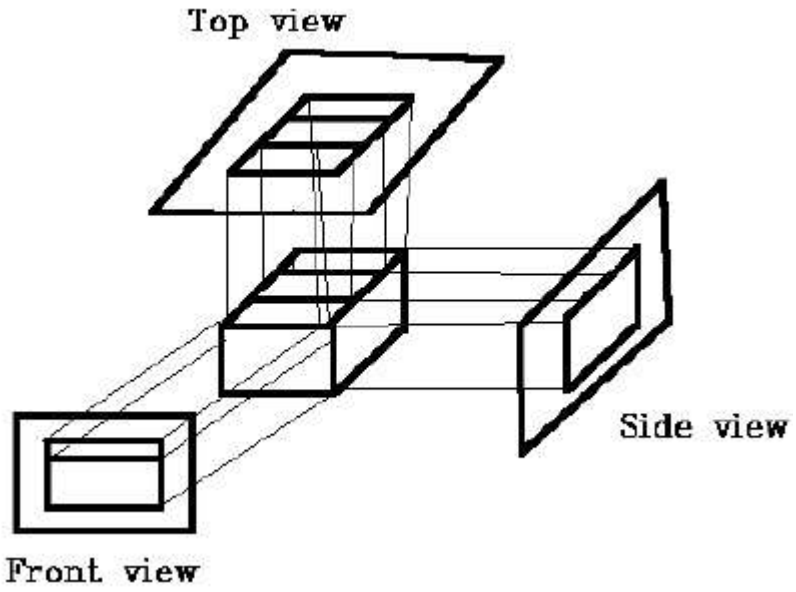


Figure 3.17 Orthographic projections

Transformation equation for orthographic parallel projections is

$x_p = x, y_p = y$

An oblique projection is obtained by projecting points along parallel lines that are not perpendicular to the projection plane.

The projection coordinates in terms of x, y, L, Φ as,

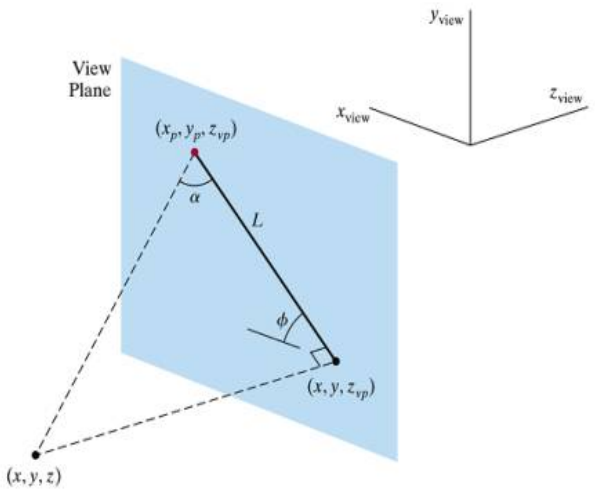
$x_p = x + L\cos\emptyset$

$y_p = x + L\sin\emptyset$

Length L depends on angle α and z coordinate

$\tan\propto = \frac{z}{L}$

Thus $L = \frac{z}{\tan\propto}$



Oblique parallel projection of coordinate position (x, y, z) to position (x_p, y_p, z_{vp}) on a projection plane at position z_{vp} along the z_{view} axis.

Fig.3.18 Oblique parallel projection

When α=45° the views obtained is cavalier projections.

When the projection angle is chosen so that tan α =2, the resulting view is cabinet projection.

Perspective Projection:

To obtain perspective projection of a three dimensional object, the points are transformed along the projection lines that meet at the projection reference point.

6.Explain 3D transformation in detail.

Translation:

In a three-dimensional homogeneous coordinate representation, a point is translated from position P = ***(x, y, z)*** to position P' = ***(x', y',*** z') with the matrix operation

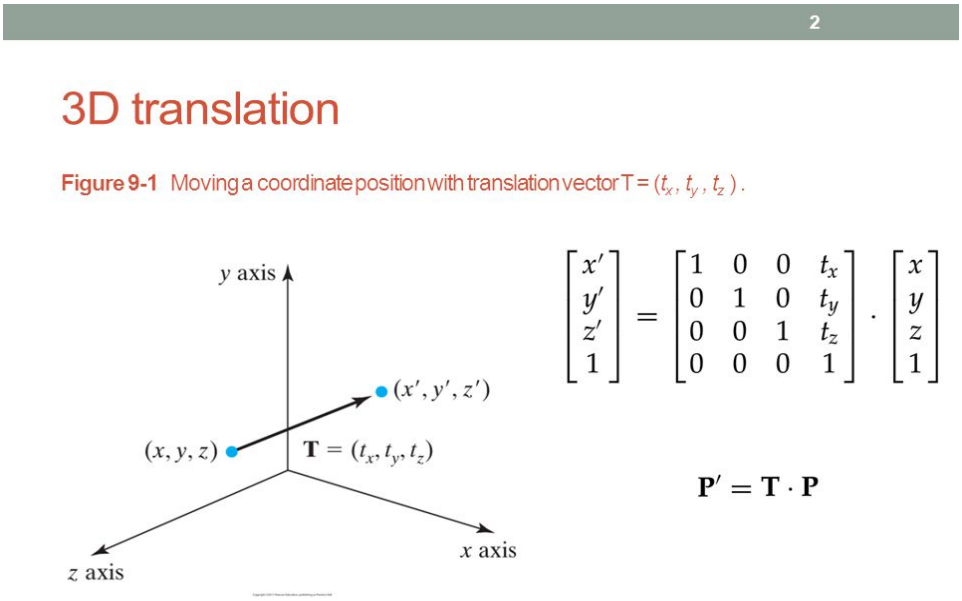


Figure 3.19 Three Dimensional Translation

Rotation:

To generate a rotation transformation of an object, the axis of rotation and the amount of angular rotation are needed.

Coordinate axes rotations

The two dimensional z axis rotation equations are easily extended to three dimensions

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

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

$$z' = z$$

Parameter Θ specifies the rotation angle.

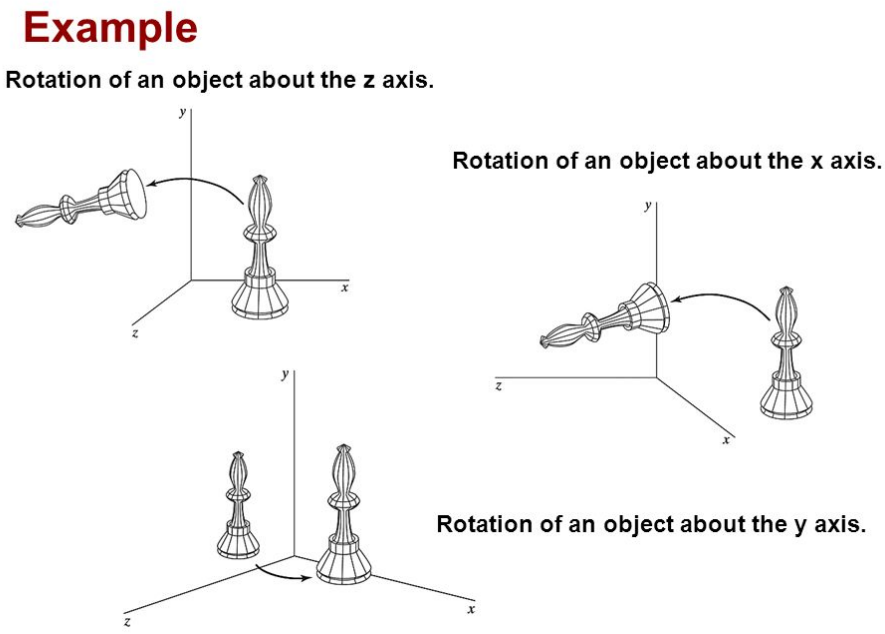


Figure 3.20 Three Dimensional Rotation

Scaling:

The matrix expression for scaling is,

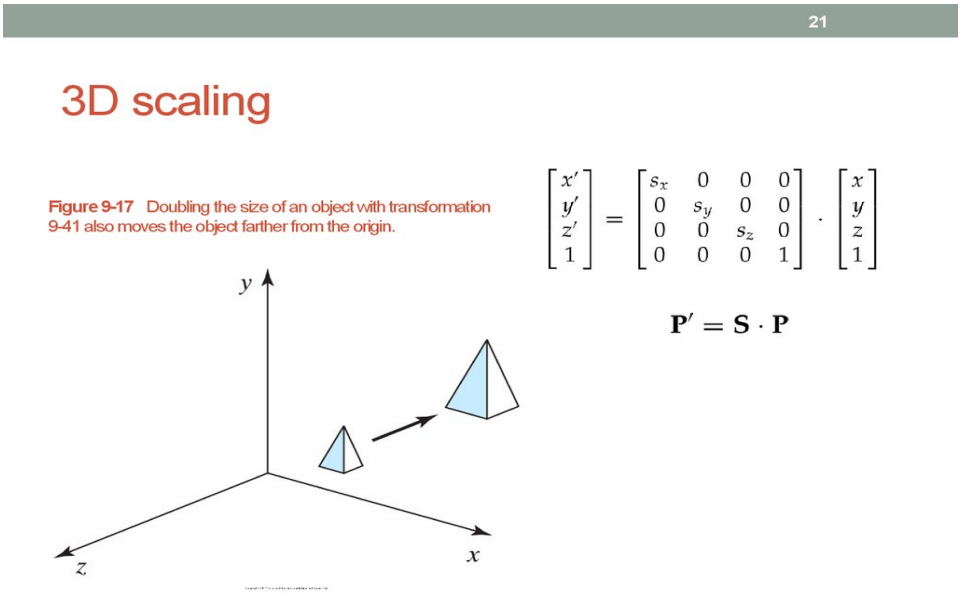


Figure 3.20 Three DimensionalScaling

UNIT – IV

ILLUMINATION AND COLOUR MODELS

Light sources – basic illumination models – halftone patterns and dithering techniques; Properties of light – Standard primaries and chromaticity diagram; Intuitive colour concepts – RGB colour model – YIQ colour model – CMY colour model – HSV colour model – HLS colour model; Colour selection.

PART-A

1. How will you convert from YIQ to RGB color model? (MAY/JUNE 2012 IT)

Conversion from YIQ space to RGB space is done with the inverse matrix transformation:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.9563 & 0.6210 \\ 1 & -0.2721 & -0.6474 \\ 1 & -1.1070 & 1.7046 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

2. Differentiate specular reflection and diffuse reflection (MAY/JUNE 2016)

specular reflection

- reflection off of shiny surfaces - you see a highlight
- shiny metal or plastic has high specular component
- chalk or carpet has very low specular component
- position of the viewer IS important in specular reflection

diffuse reflection

Using a point light:

- comes from a specific direction
- reflects off of dull surfaces
- light reflected with equal intensity in all directions
- brightness depends on theta - angle between surface normal (N) and the direction to the light source (L)
- position of viewer is not important

3.State the difference between CMY and HSV color models. (AU NOV/DEC 2012)

CMY Model	HSV Model
A color model defined with the primary colors cyan, magenta and yellow (CMY) is useful for describing color output to hard-copy devices.	The HSV model uses color descriptors that have a more natural appeal to the user. Color parameters in this model are hue (H), saturation (S) and value(V).
Hard-copy devices such as plotters produce a Color picture by coating a paper with color pigments.	To give color specification, a user selects a spectral color and the amounts of black and white that are to be added to obtain different shades, tints and tones.

4.What are subtractive colors? (AU MAY/JUNE 2012)

In CMY color model, colors are seen by reflected light a subtractive process. Cyan can be formed by adding green and blue light. Therefore, when white light is reflected from cyan-colored ink, the reflected light must have no red component. The red light is absorbed or subtracted by the ink. Similarly magenta ink subtracts the green component from incident light and yellow subtracts the blue component.

5. Mention the uses of Chromaticity Diagram.? (AU APR/MAY 2015)

- The chromaticity diagram is commonly used to evaluate a color against a gamut. The presumption is that if the chromaticity of the color lies within the gamut boundary (typically a triangle), then the color may be reproduced on that device, or may be represented by that color system.
- But color is three-dimensional and a chromaticity diagram is only two-dimensional. Because of this difference, using a chromaticity diagram to determine whether or not a color is in-gamut may lead you to a wrong conclusion.

6.What is dithering? (AU MAY/JUNE 2013)

- The term dithering is used in various contexts. Primarily, if refers to techniques for approximating halftones without reducing resolutions pixel – grid patterns do. But the term is also applied to halftone approximation methods using pixel grids and sometimes if 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.

7. List out the various properties that describe the characteristics of light. (MAY/JUNE 2016)

- Reflection
- Refraction
- Dispersion
- Interference.
- Diffraction

8. State the difference between RGB and HSV color models. [MAY/JUNE 2012]

RGB

It is an additive model, Uses Red, Green and Blue as primary colors, Represented by an unit cube defined on the R, G and B axes, The origin represents black and the vertex with coordinates(1,1,1) represents white,

Any color Cλ can be represented as RGB components as

$$C\lambda = RR + GG + BB$$

HSV

The "Whiteness" / "lightness" is a function of R, G and B when we view it in the cube while it's a separate dimension in the HSV space know as Value.

The Saturation or "colorfulness" is given by the distance of your color from the 3D diagonal of the RGB cube . In the HSV model however, the saturation is directly given as another dimension.

9.Convert the given colour value to CMY colour mode where R= 0.23, G = 0.57, B=0.11. (NOV/DEC 2015)

CMY to RGB conversion formula is

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

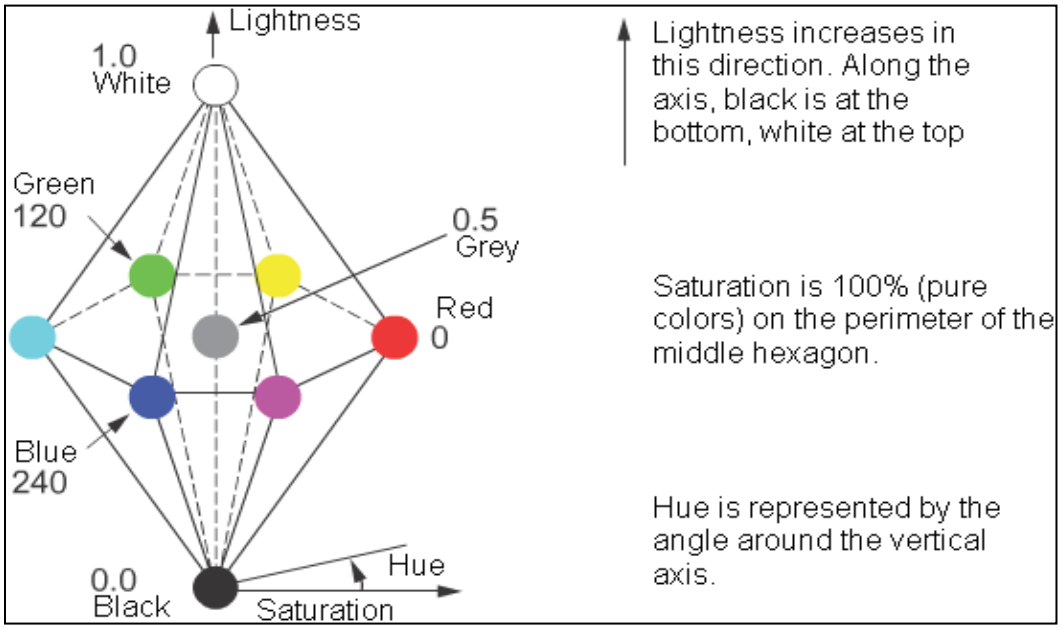
Substituting the RGB values

we got the values C = 0.77

M = 0.43

Y = 0.89

10.Draw the colour model HLS double cone.?(MAY/JUNE 2013)



PART B

1. Explain in detail on RGB and HSV color model.(MAY/JUNE 2016)(NOV/DEC 2012,2014)

i) RGB color model

- Colors are displayed based on the theory of vision,(eyes perceive colors through the stimulation of three visual pigments in the cones of the retina)
- It is an additive model, Uses Red, Green and Blue as primary colors, Represented by an unit cube defined on the R, G and B axes, The origin represents black and the vertex with coordinates(1,1,1) represents white,
- Any color Cλ can be represented as RGB components as

$$C\lambda = RR + GG + BB$$

In the RGB color model, we use red, green, and blue as the 3 primary colors. We don't actually specify what wavelengths these primary colors correspond to, so this will be different for different types of output media, e.g., different monitors, film, videotape, slides, etc.

- This is an additive model since the phosphors are emitting light. A subtractive model would be one in which the color is the reflected light. We can represent the RGB model by using a unit cube. Each point in the cube (or vector where the other point is the origin) represents a specific color. This model is the best for setting the electron guns for a CRT.

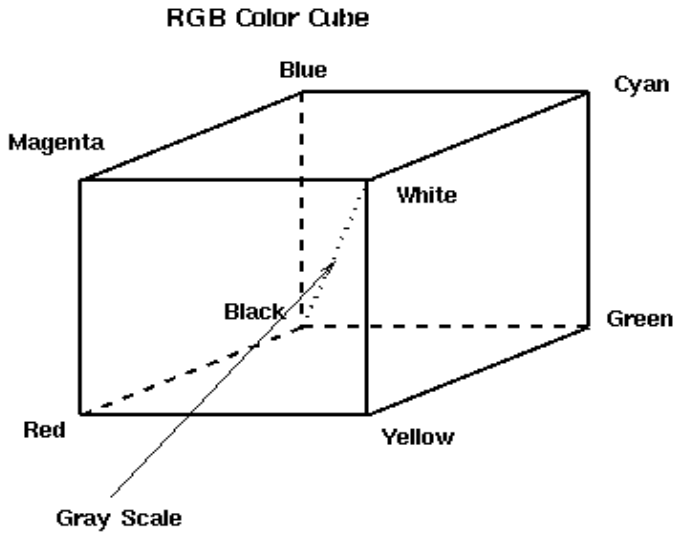


Fig 4.1 RGB Color Model

Note that for the "complementary" colors the sum of the values equals white light (1, 1, 1).

For example:

red(1, 0, 0) + cyan (0, 1, 1) = white (1, 1, 1)

green (0, 1, 0) + magenta (1, 0, 1) = white (1, 1, 1)

blue (0, 0, 1) + yellow (1, 1, 0) = white (1, 1, 1)

ii) HSV color models

Hue, Saturation, Value or **HSV** is a color model that describes colors (hue or tint) in terms of their shade (saturation or amount of gray) and their brightness (value or luminance).

The HSV color wheel may be depicted as a cone or cylinder. Instead of Value, the color model may use Brightness, making it HSB (Photoshop uses HSB).

- **H**ue is expressed as a number from 0 to 360 degrees representing hues of red (starts at 0), yellow (starts at 60), green (starts at 120), cyan (starts at 180), blue (starts at 240), and magenta (starts at 300).
- **S**aturation is the amount of gray (0% to 100%) in the color.
- **V**alue (or **B**rightness) works in conjunction with saturation and describes the brightness or intensity of the color from 0% to 100%.

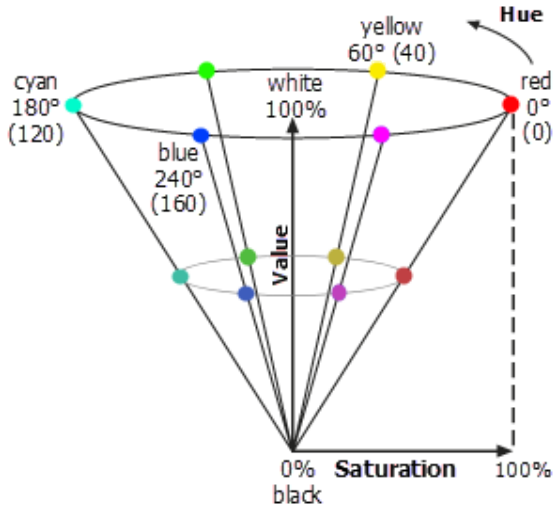


Fig 4.2 HSV Color Model

- Adding white decreases S while V remains constant
- Hue is represented as an angle about vertical axis ranging from 0 degree to 360 degrees S varies from 0 to 1

  V varies from 0 to 1

2. Briefly explain different color models in detail.

(MAY/JUNE 2013)(NOV/DEC2013)

(MAY/JUNE 2014)(MAY/JUNE2015)(NOV/DEC 2015)

i) HSL or HLS

- **Hue** indicates the color sensation of the light, in other words if the color is red, yellow, green, cyan, blue, magenta, ... This representation looks almost the same as the visible spectrum of light, except on the right is now the color magenta (the combination of red and blue), instead of violet (light with a frequency higher than blue):
- **Saturation** indicates the degree to which the hue differs from a neutral gray. The values run from 0%, which is no color, to 100%, which is the fullest saturation of a given hue at a given percentage of illumination. The more the spectrum of the light is concentrated around one wavelength, the more saturated the color will be.

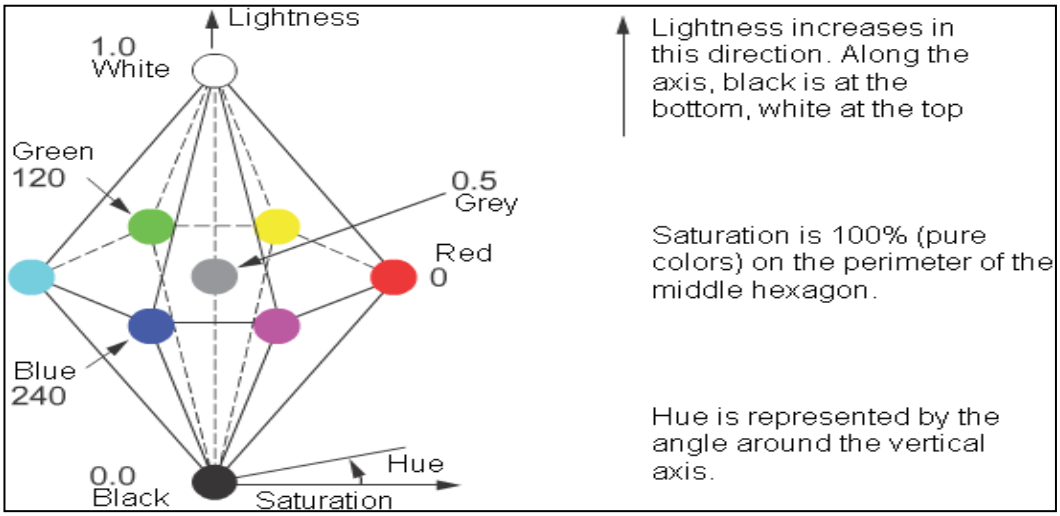


Fig 4.3 HLS Color Model

- **Lightness** indicates the illumination of the color, at 0% the color is completely black, at 50% the color is pure, and at 100% it becomes white. In HSL color, a color with maximum lightness (L=255) is always white, no matter what the hue or saturation components are. Lightness is defined as (maxColor+minColor)/2 where maxColoris the R, G or B component with the maximum value, and minColor the one with the minimum value.

iii) YIQ

The **YIQ colour** space **model** is use in U.S. commercial **colour** television broadcasting (NTSC). It is a rotation of the RGB **colour** space such that the Y axis contains the luminance information, allowing backwards-compatibility with black-and-white **colour** Tv's, which display only this axis of the **colour** space.

Y is luminance

Sometimes you have to use it

- video input/output.Makes sense in image compression:
- better compression ratio if changing class Y before compression
- High bandwidth for Y
- Small bandwidth for chromaticity
- Lab is fine for that too

Y = 0.257*R + 0.504*G + 0.098*B + 16

YIQ color space (Matlab conversion function: rgb2ntsc):

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.595716 & -0.274453 & -0.321263 \\ 0.211456 & -0.522591 & 0.311135 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

iv) CMY color model

The CMY color model defined with subtractive process inside a unit cube

- Based on subtractive process
- Primary colors are cyan , magenta , yellow
- Useful for describing color output to hard copy devices
- Color picture is produced by coating a paper with color pigments
- The printing process generates a color print with a collection of four ink dots (one each for the primary & one for black)

RGB to CMY transformation matrix-CMY to RGB transformation matrix

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

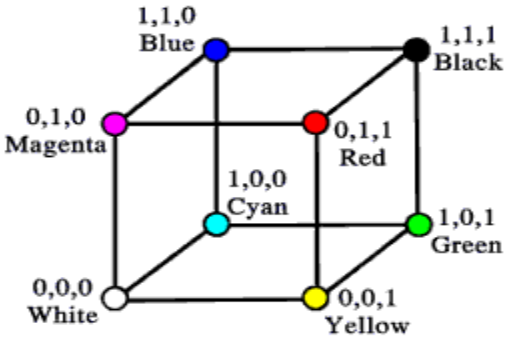


Fig 4.4 CMY Color Model

3.Explain in detail about halftone patterns and dithering techniques.

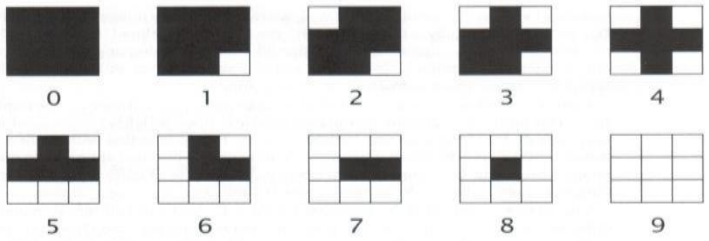
(Regulation 2013)

- The process of generating a binary pattern of black and white dots from an image is termed **halftoning.**
- In traditional newspaper and magazine production, this process is carried out photographically by projection of a transparency through a 'halftone screen' onto film.
- The screen is a glass plate with a grid etched into it.
- Different screens can be used to control the size and shape of the dots in the halftoned image.
- A fine grid, with a 'screen frequency' of 200-300 lines per inch, gives the image quality necessary for magazine production.
- A screen frequency of 85 lines per inch is deemed acceptable for newspapers.

- A simple digital halftoning technique known as **patterning** involves replacing each pixel by a pattern taken from a 'binary font'.

  Figure shows such a font, made up of ten 3 x 3 matrices of pixels.

  This font can be used to print an image consisting of ten grey levels.



  - A pixel with a grey level of 0 is replaced by a matrix containing no white pixels; a pixel with a grey level of 1 is replaced by a matrix containing a single white pixel; and so on.
  - Note that, since we are replacing each pixel by a 3 x 3 block of pixels, both the width and the height of the image increase by a factor of 3.

    Figure shows an example of halftoning using the binary font depicted in Figure

Another technique for digital halftoning is **dithering**

- Dithering can be accomplished by thresholding the image against a **dither matrix.**
- The first two dither matrices, rescaled for application to 8-bit images, are
- The elements of a dither matrix are thresholds.
- The matrix is laid like a tile over the entire image and each pixel value is compared with the corresponding threshold from the matrix.
- The pixel becomes white if its value exceeds the threshold or black otherwise.
- This approach produces an output image with the same dimensions as the input
  - image, but with less detail visible.

Algorithm to halftone an image using a dither matrix.

```
for all x & y do
    if f(x,y) > m(x,y) then
        g(x,y) =white
    else
        g(x,y)= black
    end if
End for
```

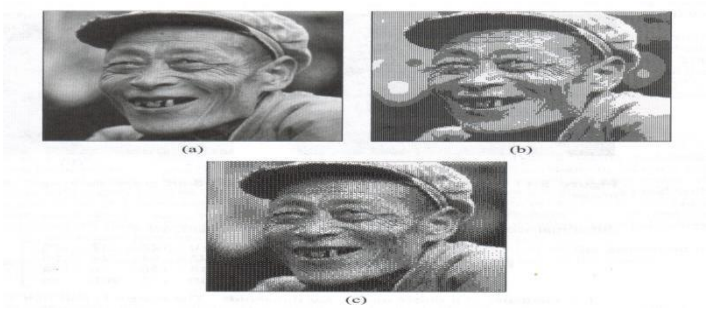


Fig 4.4 Halftone Image

4. Explain in detail the RGB chromaticity and XYZ chromaticity(regulation 2013)

Chromaticity is an objective specification of the quality of a color regardless of its luminance.

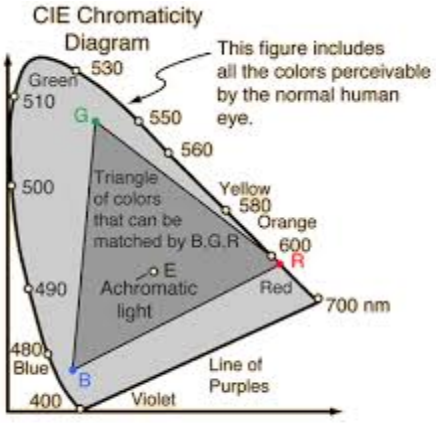


Fig 4.5 Chromaticity Diagram

RGB chromaticity

- An RGB color space is any additive color space based on the RGB color model. A particular RGB color space is defined by the three chromaticities of the red, green,

and blue additive primaries, and can produce any chromaticity that is the triangle defined by those primary colors.

- The complete specification of an RGB color space also requires a white point chromaticity and a gamma correction curve. RGB is an abbreviation for red–green–blue

XYZ chromaticity

There are two axes,

- The vertical axis represents Relative Response 0 - 2.0 (shown here) or Reflective Intensity 0 - 120% (not shown).
- The horizontal represents Wavelength in nanometres, usually from about 380 to about 720. It should be emphasised that this is a 'device-independent' colour space in which each primary colour (X,Y,Z) is always constant, unlike RGB which varies with every individual device (monitor, scanner, camera, etc.).
- XYZ is typically used to report the spectral response of a sample measured by a colorimeter or a spectrophotometer.
- A colorimeter may contain as few as three sensors, one each for red, green and blue, (or X,Y and Z), and will typically be used for display calibration and profiling.
- A spectrophotometer will report the entire spectral response at frequent intervals along the spectrum, say every 10 nanometres, and will typically be used to measure printed sheets to control a press or create an ICC profile.
- All human-visible colors have positive X, Y, and Z values. This means that we only care about one octanct of XYZ space when it comes to color conversion.
- The Y value of an XYZ color represents the relative luminance of the color as percieved by the human eye (because all eyes are a bit different, this is really an approximation based on experimental data).
  - Colors with higher Y values are perceived brighter and colors with equal Y values are perceived to have the same brightness.

- Only a subsection of the positive octant even corresponds to actual colors represented by visible light (or any light for that matter). In other words, some XYZ values have no counterpart in the real world. As long as we behave and stay in this visible subspace (i.e. don't just start defining colors as random XYZ values), we won't run into any issues.

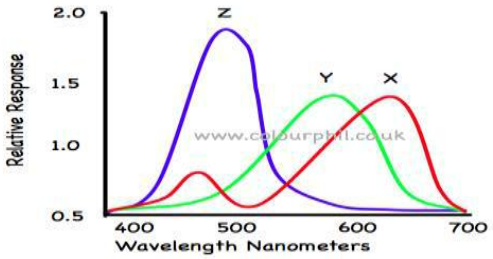


Fig 4.6 XYZ Chromaticity Diagram

5. **Explain the illumination models in detail.** **(NOV /DEC 2015)**

Ambient

- Non-directional light source
- Simulates light that has been reflected so many times from so many surfaces it appears to come equally from all directions
- intensity is constant over polygon's surface
- intensity is not affected by anything else in the world
- intensity is not affected by the position or orientation of the polygon in the world
- position of viewer is not important
- No 3-D information provided.
- Generally used as a base lighting in combination with other lights

  I = IaKa I: intensity Ia: intensity of Ambient light Ka: object's ambient reflection coefficient, 0.0 - 1.0 for each of R, G, and B

Diffuse Reflection (Lambertian Reflection)

Using a point light:

- comes from a specific direction
- reflects off of dull surfaces
- light reflected with equal intensity in all directions
- brightness depends on theta - angle between surface normal (N) and the direction to the light source (L)
- position of viewer is not important

  I = Ip Kd cos(theta) or I = Ip Kd(N' * L') I: intensity Ip: intensity of point light Kd: object's diffuse reflection reflection coefficient, 0.0 - 1.0 for each of R, G, and B ,N': normalized surface normal L': normalized direction to light source *:represents the dot product of the two vectors

Using a directional light:

- theta is constant
- L' is constant

Directional lights are faster than point lights because L' does not need to be recomputed for each polygon.

Specular Reflection

- reflection off of shiny surfaces - you see a highlight
- shiny metal or plastic has high specular component
- chalk or carpet has very low specular component
- position of the viewer IS important in specular reflection

I = Ip cos^n(a) W(theta) I: intensity Ip: intensity of point light n: specular-reflection exponent (higher is sharper falloff) W: gives specular component of non-specular materials.So if we put all the lighting models depending on light together we add up their various components to get:

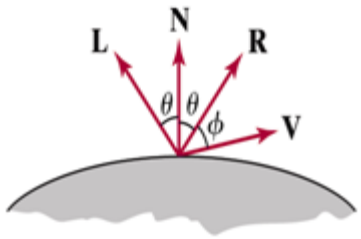
I = Ia Ka + Ip Kd(N' * L') + Ip cos^n(a) W(theta)

6.Write notes on Phong Warn model in detail.(MAY/JUNE 2016)

In 3D computer graphics, it is sometimes ambiguously referred to as "Phong shading", in particular if the model is used in combination with the interpolation method of the same name and in the context of pixel shaders or other places where a lighting calculation can be referred to as “shading”.

Phong specular-reflection model

– Note that N, L, and R are coplanar, but V may not be coplanar to the others



$$I = k_s I_l \cos^n \phi = k_s I_l (R \cdot V)^n$$

Specular reflection angle equals angle of incidence θ.

–

Fig 4.7 Phong Model

I_l : intensity of the incident light

k_s : color-independent specular coefficient

n : the gloss of the surface

Phong reflection is an empirical model of local illumination. It describes the way a surface reflects light as a combination of the diffuse reflection of rough surfaces with the specular reflection of shiny surfaces. It is based on Bui Tuong Phong's informal

observation that shiny surfaces have small intense specular highlights, while dull surfaces have large highlights that fall off more gradually. The model also includes an *ambient* term to account for the small amount of light that is scattered about the entire scene.

Warn Model

The Warn model provides a method for simulating studio lighting effects by controlling light intensity in different directions.

• Light sources are modeled as points on a reflecting surface, using the Phong model for the surface points.

• Then the intensity in different directions is controlled by selecting values for the Phong exponent

In addition, light controls and spotlighting, used by studio photographers can be simulated in the Warn model.

• Flaps are used to control the amount of light emitted by a source In various directions

UNIT V

ANIMATIONS AND REALISM

ANIMATION GRAPHICS: Design of Animation sequences – animation function – raster animation – key frame systems – motion specification –morphing – tweening. COMPUTER GRAPHICS REALISM: Tiling the plane – Recursively defined curves – Koch curves – C curves – Dragons – space filling curves – fractals – Grammar based models – fractals – turtle graphics – ray tracing.

PART A

1. What is animation? (AU NOV/DEC 2011)

Computer Animation generally refers to any time sequence of visual changes in a scene. In addition to changing positions with translations or rotations, a computer generated animations could display time variations in object size, color, transparency or surface texture. Animations often transition from one object shape to another.

2. Define Key frame systems. (AU NOV/DEC 2012)

Key-frame systems are specialized animation languages designed to generate the in-between frames from user specified key frames. Each object in the scene is defined as a set of rigid bodies connected at the joints and with a limited number of degrees of freedom. In-between frames are generated from the specification of two or more fey frames. Motion paths can be given by kinematic description as a set of spline curves or physically based by specifying the forces acting on the objects to be animated

3. Define Fractals. (MAY/JUNE 2012,13)

Fractals are those which have the property of a shape that has the same degree of roughness no matter how much it is magnified. A fractal appears the same at every scale. No matter how much one enlarges a picture of the curve, it has the same level of detail.

4. What is a turtle graphics program. (MAY/JUNE 2016)

Turtle graphics is a term in computer graphics for a method of programming vector graphics using a relative cursor upon a Cartesian plane. Turtle graphics is a key feature of the Logo Programming language.

The turtle is defined by the following parameters.

a) Position of the turtle (x, y)

b) Heading of the turtle 0 the angle from the x axis.

5. List the attributes of turtle in graphics. (AU NOV/DEC 2015)

Turtle graphics has three attributes

- Current Position location
- Current direction Orientation
- Pen

6. Differentiate Mandelbrot sets and Julia sets. (MAY/JUNE 2011)

Mandelbrot sets	Julia sets
A very famous fractal is obtained from the Mandelbrot set, which is a set of complex values z that do not diverge under squaring transformation z0=z, zk=z2k-1+z0,k=1,2,3.	For some functions, the boundary between those points that move towards those points that move towards infinity and those that tend toward a finite limit is a fractal. The boundary of the fractal is called the Julia set
It is the black inner portion, which appears to consist of a cardioid along with a number of wart like circles glued to it. Its border is complicated and this complexity can be explored by zooming in on a portion of the border.	Julia sets are extremely complicated sets of points in the complex plane. There is a different Julia set Jc for each value of c.

7. What is Koch curve? (MAY/JUNE 2012)

The Koch curve can be drawn by dividing line into 4 equal segments with scaling factor 1/3. and middle 2 segments are so adjusted that they form adjustment sides of an equilateral triangle.

8. What is Morphing and tweening?

Transformation of object shape from one form to another is called morphing.
Tweening is the process, which is applicable to animation objects defined by a sequence of points, and that change shape from frame to frame.

9. What are Peano curves? (MAY/JUNE 2012)

A fractal curve can in fact fill the plane and therefore have a dimension of two. Such curves are called Peano curves.

10. What is a Scripting system.(MAY/JUNE 2016)

Scripting systems allow object specifications and animation sequences to be defined with a user input string. From the script, a library of various objects and motions can be constructed.

PART B

1. Define animation sequence. Explain the various steps involved in animation sequence. (MAY/JUNE 2016)

Animation sequence is designed with following steps

- ❖ Story board layout
- ❖ Object definitions
- ❖ Key frame specifications
- ❖ Generation of in between frames

- ➢ Real time computer animations produced by flight simulators display motion sequences in response to settings on aircraft controls.
- ➢ For frame-by-frame animation, each frame of the scene is separately generated and stored. Then 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 nurnber 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. Depending on the speed specified for the motion, some key frames can be duplicated.
- For a 1 minute film sequence with no duplication, 1440 frames are needed. With five in-betweens for each pair of key frames, 288 key frames would be needed. If the motion is not too complicated, we could space the key frames a little farther apart.

2. Distinguish between raster animation and key frame animation in detail.

(MAY/JUNE 2016,AU NOV/DEC 2015)

Raster Animation

- On raster systems, real-time animation in limited applications can be generated 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 can be performed by rotating rectangular blocks of pixels through arbitrary angles using antialiasing procedures.
- To rotate a block of pixels, the percent of area coverage for those pixels that overlap the rotated block are determined.
- Sequences of raster operations can be executed to produce real-time animation of either two-dimensional or three-dimensional objects, as long as the animation to motions in the projection plane is restricted.
- Then no viewing or visible-surface algorithms need be invoked.
- Animation of objects along two-dimensional motion paths can also be done using the color-table transformations.
- Here, the objects at successive positions are predefined along the motion path, and the successive blocks of pixel values to color-table entries are set.
- The pixels at the first position of the object are set to "on" values, and then the pixels at the other object positions are set 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.

Key frame animation:

- In-betweens frames from the specification of two (or more) key frames have to be generated.
- Motion paths can be given with a kinematic description as a set of spline curves or the forces acting on the objects can be specified.
- For complex scenes, the frames are divided into individual components or objects called cells (celluloid transparencies), Then in-between frames of individuals objects are generated using interpolation.

- 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 form to another is called morphing, which is a shortened form of metamorphosis.
- Morphing methods can he applied to any motion or transition involving a change in shape.
- When object is described using polygon, the two key frames for which in-between frames are to be generated are compared.

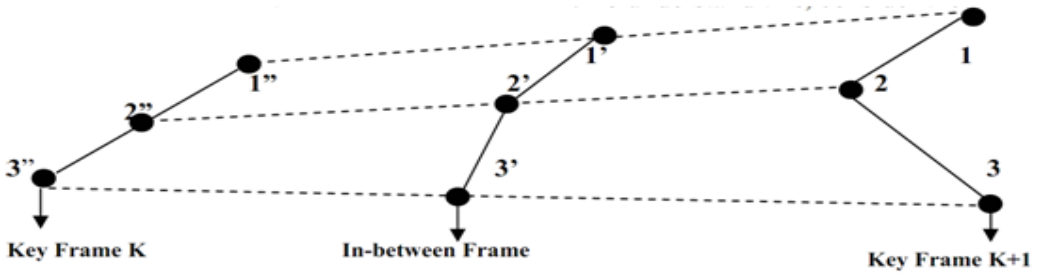


Figure 5.1: Generation of in-between frames using linear interpolation

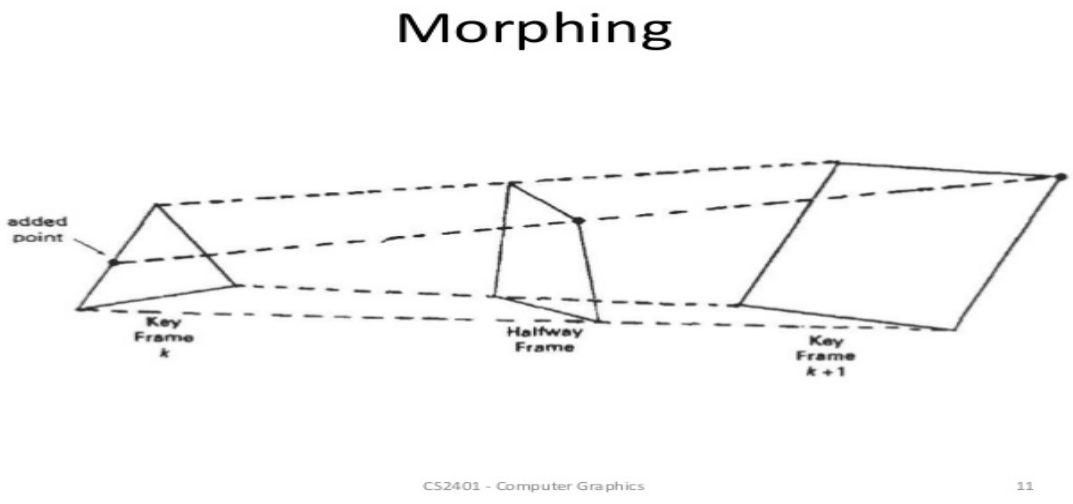


Figure 5.2: Transforming a triangle into a quadrilateral

3.Write short notes

(i) **Ray Tracing**

(ii) **Koch curves** **(MAY/JUNE 2016)**

(AU NOV/DEC 2015)

Ray tracing:

- Ray tracing is a technique for generating an image by tracing the path of light through pixels in an image plane and simulating effects of its encounters with virtual objects.

Basic Ray Tracing algorithms:

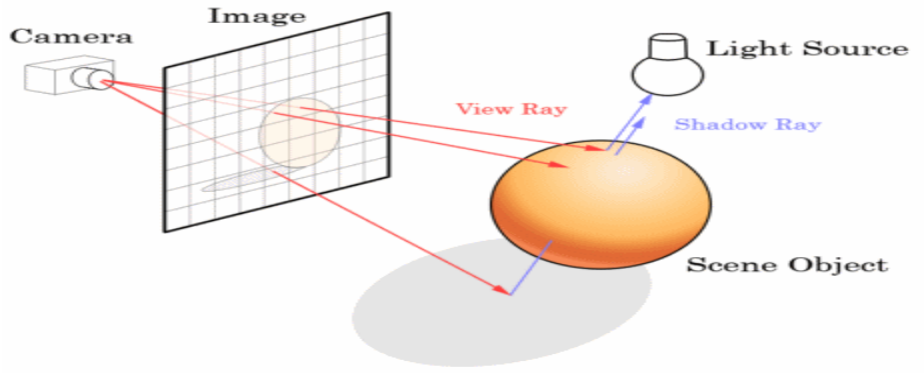


Figure 5.3 Basic ray tracing

- This technique is capable of producing a very high degree of visual realism, usually higher than that of typical scanline rendering methods, but at a greater computational cost.
- Ray tracing is capable of simulating a wide variety of optical effects, such as reflection and refraction, scattering and chromatic aberration.
- A typical scene may contain various objects. These objects are described in some fashion and stored in the object list.
- For each pixel, a ray is constructed that starts at the eye or camera and passes through the lower left corner of the pixel.
- For each pixel ray, each object is tested in the picture to determine if it is intersected by the ray.

- If the ray intersects the object in the scene, the hit time is noted. Hit time is the value of t at which the ray r(t) coincides with the object's surface.
- When all objects have been tested, the object with smallest hit time is closest to the eye or camera.
- Next the color of the light that reflects off the object, in the direction of the camera or eye, is computed and stored in the pixel.
- Reflection and refraction rays are known as secondary rays.
  successively intersected surface is added to binary ray tracing tree

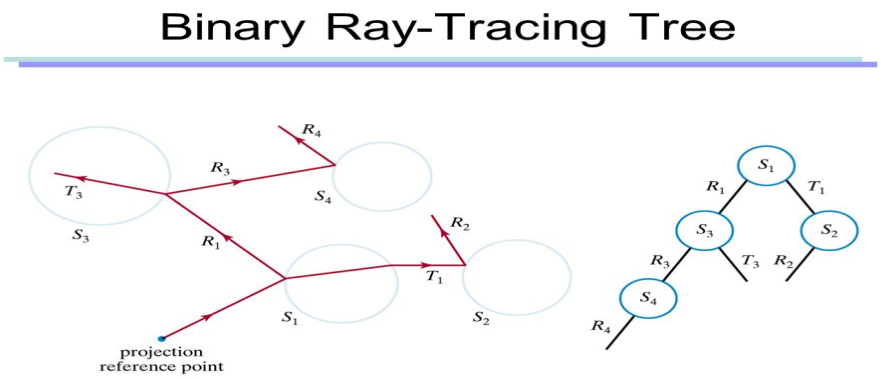


Figure 5.4 Binary ray tracing

Antialiased Ray Tracing:

- Two basic techniques are
  - Super sampling
  - Adaptive sampling
- In super sampling and adaptive sampling the pixels are treated as a finite square area instead of a single point
- Super sampling uses evenly spaced rays over each pixel area.
- Adaptive sampling uses unevenly spaced rays in some regions of pixel area.
- When multiple rays per pixel are used, the intensities of the pixel rays are averaged to produce the overall pixel intensity.

Koch curves

The Koch curve can be drawn by dividing line into 4 segments with scaling factor 1/3 and middle two segments are so adjusted that they form adjacent sides of an equilateral triangles.

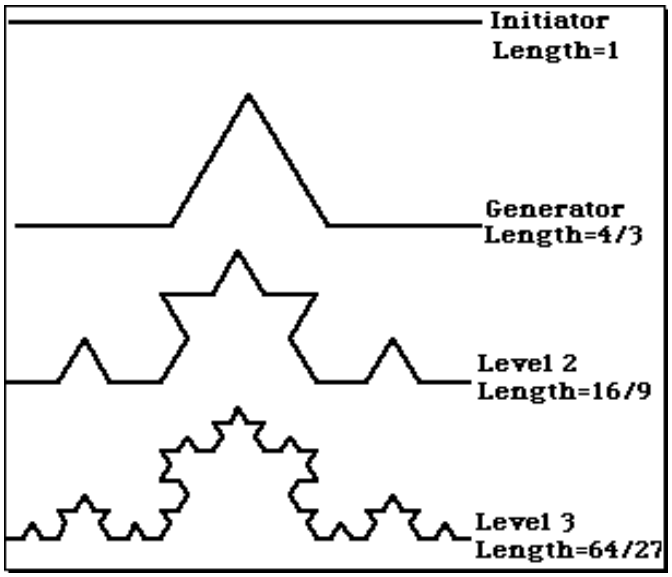


Figure 5.5: First, Second and third approximation of a Koch curve

From the above figure, the features of Koch curve are noted as

- Each repetition increases the length of the curve by factor 4/3.
- Length of the curve is infinite
- Unlike Hilbert's curve it doesn't fill an area
- It doesn't deviate much from its original shape.
- If the scale of the curve is reduced by 3 then the curve looks like original one.

  $4=3^D$
- Solving for D,

  $D=\log_3 4 = \log 4/\log 3= 1.2618$
- Therefore the topological dimension for Koch curve is 1 and fractal dimension is 1.2618

Koch curve fractal:

- A complex curve can be constructed by repeatedly refining a simple curve.
- Successive generations are named as k0,k1,k2…
- The Koch curve is a fractal starts with a simple pattern made of a line divided into 3 parts. Erase the middle segment and replace it with an upside down 'V' shape. Now it is made up of 4 line segments. The total length now becomes 4/3
- Fractals are never ending patterns made by repeating the same idea over again.
- Eventually the pattern starts look like a fractal in nature, such as coastline or part of a snowflake.

String production of Koch curve:

Curves can be generated by refining line segments. A simple approach is to generate these curves using L-System(lindenmayer systems) where curves are drawn based on simple rules.

In L-system

- Variables : F
- Constants: + -
- Starts : F
- Rules: F-> + F - - F +

  Where

  "F" means "draw forward"

  "+" means "turn clockwise 45^{o}"

  "-" means "turn clockwise 45^{o}"

4. What is Fractal? Explain in detail the various fractals.

(NOV/DEC 2013,MAY/JUNE 2016)

Fractals are those which have the property of a shape that has the same degree of roughness no matter how much it is magnified. A fractal appears the same at every scale. No matter how much one enlarges a picture of the curve, it has the same level of detail.

Classification of Fractals:

The fractals can be classified as

- ❖ Self similar fractals
- ❖ Self affine fractals
- ❖ Invariant fractal sets

Self similar fractals:

- ➢ It has parts those are scaled-down versions of the entire object. In these fractals object subparts are constructed by applying a scaling factor s to overall shape. It is the choice of user to use the scaling factor s for all parts.
- ➢ Another Subclass of fractals is called statistically self similar fractals, in which user can also apply random variations to the scaled down subparts.
- ➢ These fractals are used to model trees, shrubs and other plants.

Self affine fractals:

- ➢ These fractals have parts those are formed with different scaling parameters in different co-ordinate directions.
- ➢ In these fractals random variations can be included to obtain statistically self affine fractals.
- ➢ These fractals are commonly used to model water, clouds and terrain.

Invariant fractals:

- These fractals are formed with nonlinear transformation.
- This class of fractals includes self squaring fractals, which are formed with squaring functions in complex space and self inverse fractals formed with inverse procedures.

Fractal dimensions:

The detail variation in a fractal object can be described with a number D, called the fractal dimension, which is a measure of the roughness, or fragmentation, of the object.

More jagged-looking objects have larger fractal dimensions.

Some iterative procedures can be set to generate fractal objects using a given value for the fractal dimension D.

With other procedures, the fractal dimension from the properties of the constructed object may be determined, although, in general, the fractal dimension is difficult to calculate.

An expression for the fractal dimension of a self-similar fractal, constructed with a single scalar factor s, is obtained by analogy with the subdivision of a Euclidean object.

Figure shows the relationships between the scaling factor r; and the number of subparts n for subdivision of a unit straight-line segment, A square, and a cube.

With s = 1 /2, the unit line segment is divided into two equal-length subparts. Similarly, the square in is divided into four equal-area subparts, and the cube is divided into eight equal-volume subparts.

For each of these objects, the relationship between the number of subparts and the scaling factor $ns^D = 1$

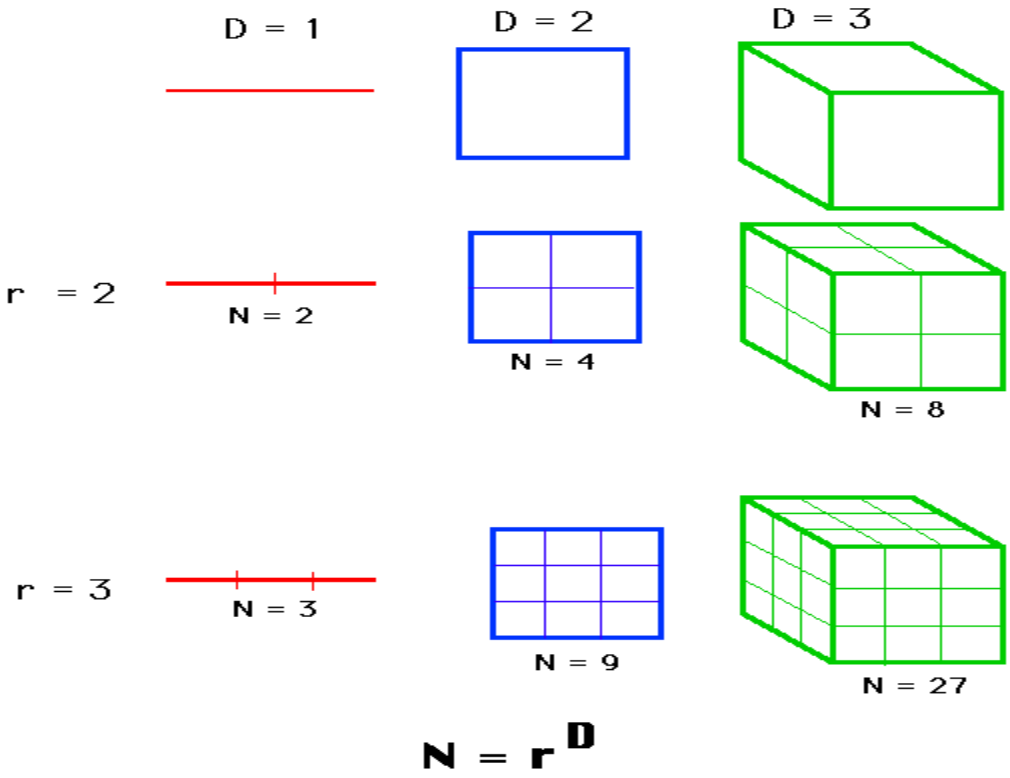
Solving this expression for D, the fractal similarity dimension,

$$D = \frac{\ln n}{\ln (1/s)}$$

For a self similar fractal ,the fractal dimension is obtained by

$$\sum_{k=1}^{n} s_k^D = 1$$

Where s_k is the scaling factor for subpart k



Geometric Construction of Deterministic self similar fractals:

To geometrically construct a deterministic (nonrandom) self-similar fractal, it is necessary to start with a given geometric shape, called the initiator.

Subparts of the initiator are then placed with a pattern, called the generator.

Example: Koch curve

Each straight-line segment in the initiator is replaced with four equal-length line segments at each step. The scaling factor is 1/3, so the fractal dimension is D = In 4/In 3 -- 1.2619. Also, the length of each line segment in the initiator increases by a factor of 4/3 at each step.

Another way to create a self similar fractal is to punch holes in a given initiator, instead of adding more surfaces.

Geometric construction of statistically self similar fractals:

One way to generate self similar fractals is to choose a generator randomly at each step from a set of predefined shapes

Another way is to generate random self similar is to compute coordinate displacements randomly.

Display of trees and other plants can be constructed with this method.

Affine fractal construction:

Using affine fractal methods highly realistic representations for terrain and other natural objects can be obtained.

5. How will you generate grammar based model? Explain. NOV/DEC 2015

(0) B (1) A[B]AA[B] (2) AA[A[B]AA[B]]AAAA[A[B]AA[B]]

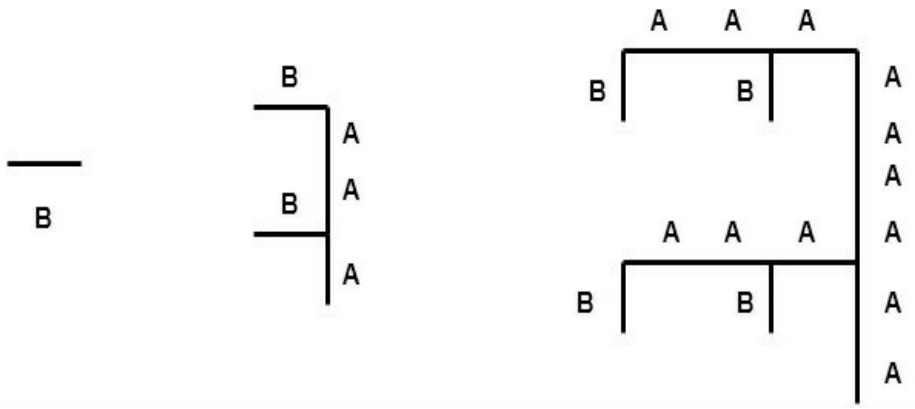


Figure5.7 : Tree representation using grammar based languages

The language described by a grammar consisting of production rules, alphabets, symbols like "[" and "]" are known as L-grammers or graftals.

These languages are used to describe structures of certain plants. These structures created using grammar based languages are called grammar based models.

From the above figure, the observations are

- Language represents a sequence of segments in a graph structure.

- Bracketed portions represent portions that branch from the symbol preceding them.
- Symbol '[' and ']' are used to branch down to the left of the current main axis and '(' and ')' are used to branch down to the right of the current main axis.
- Such graph structures can be used to generate extremely complex patterns.

Pseudo code for plant-growth algorithm:

```
while(There are buds to process)
do Bud viability test
if(bud is still alive) then
bud growth test
if (bud is growing then )
create an internode by knowing
its position and direction
create apical bud
for(each possible bud at old bud location)
if (ramification) then
create auxillary buds;
end if
end if
end if
```

INDUSTRIAL AND PRACTICAL CONNECTIVITY OF THE SUBJECT

To create Graphics Softwares in the Information Technology Field

Make Graphics Devices to support end user applications.

Develop Primitive oriented Applications.

Development of Computer Games

B.E/B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2012.
Seventh Semester
Computer Science and Engineering
CS 2401/CS 71-COMPUTER GRAPHICS
(Common to Information Technology)
(Regulation-2008)
PART-A

1. Write down the shear transformation matrix. (Pg No 38)
2. Define text clipping.
3. Differentiate oblique and orthogonal projections.

4. Define spline curves.

5. State the difference between CMY/RGB and HSV color models. . (Pg No 83)

6. What are key frame systems? (Pg No 99)

7. What is a shadow?

8. Define texture.

9. List down the properties of piano curves.

10. Write down some of the Boolean operation on objects.

PART-B

11.a) Explain the following(pg. no 33)

(i) Line drawing algorithm. (Pg No 22)

(ii) Line clipping algorithm. (Pg No 38)

(OR)

b) With suitable example, explain the following

(i) Rotational transformation.

(ii) Curve clipping algorithm. (pg. no 44)

12.a) (i) Differentiate parallel and perspective projections.

(ii) Writes notes on 3D viewing.(pg.no 76)

(OR)

b) (i) With suitable examples, explain the 3D transformations.

(ii) Writes notes on quadric surfaces.

13.a) (i) Explain RGB color model in detail.

(ii) Explain how 3D scenes are drawn.

(OR)

b) (i) Discuss the computer animation techniques.

(ii) Explain how 3D objects are drawn.

14.a) Differentiate flat and smooth shading models.

(OR)

b) Discuss the methods to draw and add shadows to objects.

15.a) Write notes on the following

(i) Mandelbrot sets.

(ii) Julia sets.

(OR)

b) (i) Describe the creation of images by iterated functions.

(ii) Explain methods for adding surface texture.

B.E/B.Tech. DEGREE EXAMINATION, MAY/JUNE 2013

Seventh Semester

Computer Science and Engineering

CS 2401/CS 71-COMPUTER GRAPHICS

(Common to Information Technology)

(Regulation-2008)

PART-A

1. Define aspect ratio. (pg. no13)
2. How will you clip a point?
3. What are the advantages of B spline over Bezier curve? (Page no 54)

4. What is the critical fusion frequency? (Pg No 56)

5. Draw the colour model HLS double cone. (pg. no 86)

6. What is dithering? (pg. no 84)

7. Define rendering.

8. Differentiate flat and smooth shading.

9. Define fractals.(pg.no 100).

10. What is surface patch?

PART-B

11. a) Explain the basic concept of midpoint ellipse drawing algorithm. Derive the decision parameter of the algorithm steps.

(OR)

b) (i) Explain two dimensional translation and scaling with an example. (Pg No 47)

(ii) Obtain a transformation matrix for rotating an object about a specified pivot point.

12.(a) (i) Determine the blending function for uniform periodic B spline curve for n=4, d=4.

(ii) Explain any one visible surface identification algorithm. (pg.no.56)

(OR)

(b) Explain a method to rotate an object about an axis that is not parallel to the coordinate axis with neat block diagram and derive the transformation matrix for the same.

13. (a) Briefly explain different color models in detail.

(OR)

(a) How will you model three dimensional objects and scenes in OPENGL? Explain with an example code.

14. (a) (i) Explain the method of adding shadows to objects.

(ii) Explain gauraud shading technique and write the deficiencies in the method and how it is rectified using phong shading technique.

(OR)

b) (i) Explain how to add texture to faces.

(ii) How will you build and fix camera position in a graphics program?.Explain.

15. (a) Briefly explain different types of fractals with neat diagram and also explain how to construct fractals and the uses of fractals in computer graphics.

(OR)

b) (i) Explain ray tracing method in detail.

(ii)Write short notes on applying Boolean operations on modeled objects to create new objects.

B.E/B.Tech DEGREE EXAMINATIONS, NOVEMBER/ DECEMBER 2013
Seventh Semester
Computer Science and Engineering
CS 2401/CS 71/ 10144 CS 702- COMPUTER GRAPHICS
(Common to Information Technology)
(Regulation 2008/2010)

Time: Three hours **Maximum: 100 marks**

Answer ALL questions

PART A-(10*2=20 marks)

1. Digitize a line from (10,12) to (15,15) on a raster screen using Bresenhams straight line algorithm.
2. List the different types of text clipping methods available.
3. Give the general expression of Beizer Bernstein polynomial.
4. Give the single point perspective projection transformation matrix when projectors are placed on the z-axis.
5. List any four real-time animation techniques
6. How are mouse data sent to an OpenGL application?

7. Which shading method is faster and easier to calculate? Why?
8. What are the types of reflection of incident light?
9. Where does the ray r(t)=(4, 1, 3) +(-3, -5, -3)t hit the generic plane?
10. How objects are modeled using constructive solid geometry technique?

PART B-(5*16=80 marks)

11. (a) (i) Calculate the pixel location approximating the first octant of a circle having Center at (4, 5) and radius 4 units using Bresenhams algorithm. (8)

(ii) Discuss in brief: Antialiasing techniques (8)

Or

(b) (i) A polygon has four vertices located at A(20 , 10) B(60, 10) C(60, 30) D(20, 30). Calculate the vertices after applying a transformation matrix to double the size of polygon with point A located on the same place. (8)

(ii) The reflection along the line y=x is equivalent to the reflection along the X axis followed by counter clockwise rotation by Θ degrees. Find the value of Θ. (8)

12. (a)(i) A cube has its vertices located at A(0,0,10), B(10,0,10), C(10,10,10), D(0,10,10), E(0,0,0), F(10,0,0), G(10,10,0), H(0,10,0). The y axis is vertical and Z axis is oriented towards the viewer. The cube is being viewed from point (0, 20.80). Calculate the perspective view of the cube on XY plane. (8)

(ii) Discuss on the various visualization techniques in detail. (8)

Or

(b) (i) Calculate the coordinates of a block rotated about x axis by an angle of = 30 degrees. The original coordinates of the block are given relative to the global xyz axis system. A(1,1,2) B(2,1,2) C(2,2,2) D(1,2,2) E(1,1,1) F(2,1,1) G(2,2,1) H(1,2,1). (8)

(ii) Discuss on Area subdivision method of hidden surface identification algorithm (8)

13. (a) Discuss on various colour models in detail. (16)

Or

(b) Discuss on the methods used in OPENGL for handling a window and also write a simple program to display a window on the screen (16)

14. (a) Discuss the process of adding textures to faces of real objects. (16)

Or

(b) Compare Flat shading and Smooth shading with respect to their characteristics and types. (16)

15. (a) (i) Discuss the Ray tracing Process with an example. (8)

(ii) Explain how refraction of light in a transparent object changes the view of the three dimensional object (8)

Or

(b) Write short notes on:

(i) Mandelbrot Sets (5)

(ii) Fractal geometry (5)

(iii) Boolean operations on objects (6)

B.E/B.Tech DEGREE EXAMINATIONS, MAY/ JUNE 2014

Seventh Semester

Computer Science and Engineering

CS 2401/CS 71/ 10144 CS 702- COMPUTER GRAPHICS

(Common to Information Technology)

(Regulation 2008/2010)

Time: Three hours **Maximum: 100 marks**

Answer ALL questions

PART A-(10*2=20 marks)

1. List out few attributes of output primitives.(pg.no 13)
2. What do you mean by clipping? (Pg No 37)
3. Define Splines. (Pg no 53)
4. What do you mean by viewing?
5. List out basic graph primitives in open GL.
6. Define Key Frame.
7. What is flat shading?
8. How do you add texture to faces?
9. What are peano curves?
10. What is the use of fractals in graphics applications?

PART B- (5*16=80 marks)

11. (a) Describe the ellipse drawing algorithm(pg .no 32) (16)

Or

(b) Explain the Cohen-Sutherland line clipping algorithm. (16)

12.(a) Write short notes on

(i) Parallel and perspective projections. (8)

(ii) Visualization of data sets for scalar fields. (8)

Or

(b) Explain the following:

(i) Basic 3 D transformations (8)

(ii) Visible surface Identification.(pg no 63) (8)

13. (a) Describe briefly about the various color models(RGB, YIQ, CMY and HSV)

Or

(b)Discuss in detail the process of drawing three dimensional objects.

14. (a) Discuss about creation, rendering textures and drawing shadows for shaded Objects.

Or

(b) Discuss about adding shadow of objects and building a camera in a program.

15.(a) (i) Explain about creation of images by iterated functions. (8)

(ii) Write about Mandelbrot and Julia Sets (8)

Or

(b) Write Short notes on:

(i) Ray Tracing (8)

(ii) Boolean operations on Objects (8)

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2014

Seventh Semester

Computer Science and Engineering

CS2401/CS71/10144 CS 702-COMPUTER GRAPHICS

(Common to Information Technology)

(Regulation 2008/2010)

Time: Three hours **Maximum: 100 marks**

Answer ALL questions

PART A-(10 x 2=20 marks)

1. List down any two attributes of lines.
2. Give an example for text clipping.
3. Differentiate parallel and perspective projections.
4. What are splines?
5. State the difference between CMY and HSV color models.
6. Write down the different types of animation.
7. What is meant by flat shading?
8. Define texture pattern.
9. What are Julia sets?
10. Define ray tracing.

PART B-(5x16=80 marks)

11. (a) Explain the ellipse drawing algorithm with an example.

Or

(b) Explain briefly the line clipping algorithm with an example. (pg. no 37)

12. (a) Discuss the three dimensional object representation in detail.

Or

(b) Discuss the visible surface detection methods in detail.

13. (a) (i) Explain briefly the RGB color model. (8)

(ii) Mention the sailent features of Raster Animation (8)

Or

(b) Discuss the following:

(i) Methods to draw 3D objects. (8)

(ii) Basic OPENGL operations. (8)

14. (a) Explain the steps involved in the following:

(i) Smooth and Flat Shading. (8)

(ii) Adding shadows of objects. (8)

Or

(b) Explain the following:

(i) Adding texture to faces. (8)

(ii) Building camera in a program. (8)

15. (b) Write notes on the following:

(i) Random fractals. (8)

(ii) Boolean operations on objects. (8)

Or

(b) Discuss the following:

(i) Adding surface texture. (8)

(ii) Peano curves. (8)

B.E./B.Tech. DEGREE EXAMINATION, APRIL /MAY 2015

Seventh Semester

Computer Science and Engineering

CS2401/CS71/10144 CS 702-COMPUTER GRAPHICS

(Common to Information Technology)

(Regulation 2008/2010)

Time: Three hours **Maximum: 100 marks**

Answer ALL questions

PART A-(10 x 2=20 marks)

1. Identify the contrast between Raster and Vector Graphics. (pg. no13)
2. What is 'Shear' Transformation? ?(Page no33)
3. What is 'Mesh Modeling'
4. Draw the 3D viewing pipeline.
5. Give the code snippet for setting up a coordinate system in OpenGL.
6. Mention the uses of chromaticity diagram. (pg. no 84)
7. What is 'Flat Shading'?
8. What are the two types of texture applied on surfaces?
9. What is 'Koch Curve'
10. What is CSG technique?

PART B-(5x16=80 marks)

11. (a) (i) Summarize Midpoint Circle drawing procedure. (8)

(ii)Use the above procedure to compute points on a circle with centre at (5,5) and radius of 8 units. (8)

Or

(b) (i) Rotate a triangle [(4,6),(2,2),(6,2)] about the vertex (4,6) by 180^0 CCW and find the new vertices. (8)

(ii) Prove that Reflection is equal to Rotation by 180^0 (Pg No 61) (8)

12. (a) (i) Write short notes on 3D transformations. (8)

(ii) Present any simple method for visible surface detection. (8)

Or

(b) Describe projection transformations in 3D. (16)

13. (a) Describe about the most commonly used color models used in Computer Graphics. (16)

Or

(b) (i) Write short notes on techniques for Computer Animation. (8)

(ii) Write code snippet for drawing basic 2D primitives on OpenGL. (8)

14. (a) (i) How are diffuse and specular components computed in a shading model. (8)

(ii) Write about Gouraud and Phong Shading techniques. (8)

Or

(i) How are shadows created using texture? Discuss. (8)

(ii) Present a brief discussion on 'Reflection mapping' (8)

15. (b) (i) How are Peano curves produced? Give examples. (8)

(ii) Write short notes on Mandelbrot sets. (8)

Or

(b) Describe the process of Ray Tracing. Explain how it is used to create Reflections and Transparency (16)

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2015
Seventh Semester
Computer Science and Engineering
CS2401/CS71/10144 CS 702-COMPUTER GRAPHICS
(Common to Information Technology)
(Regulation 2008/2010)

Time: Three hours **Maximum: 100 marks**

Answer ALL questions

PART A-(10 x 2=20 marks)

1. Write down the principle of two dimensional clipping.
2. Give the general two dimensional rotation transformation matrix about a fixed point.
3. What are the categories of visible surface detection algorithms? (Pg No 55)
4. How will you represent a curve in graphics? (Pg No 55)
5. Explain the term key frame.
6. How will you open a new project in OPENGL?
7. List the various camera movements.
8. How will you add texture to an object?
9. Write down the equation of fractal similarity dimension.
10. How is Julia set created?

PART B-(5x16=80 marks)

11. (a) (i) Plot one quadrant of a circle of radius 7 pixels with the origin at the centre using Midpoint circle drawing algorithm. (model (pg .no.25)) (8)
(ii)Explain text clipping algorithm in detail. (8)

Or

(b) (i) Clip a quadrilateral ABCD with coordinates (10,18) (22,18) (34,27) and (10,37) against a window QRST with coordinates(5,15) (30,15) (30,25) and (5,25) using Cohen Sutherland algorithm. (8)
(ii) Explain the attributes of output primitives in detail. (pg .no.26) (8)

12. (a) (i) Enumerate the difference between a window and a viewpoint. (8)

(ii) Demonstrate local scaling taking scaling vectors along the x,y,z axes as

2,3,1 respectively for a cube with homogeneous position vectors. (8)

Or

(b) (i)Explain the advantages and disadvantages of B spline surfaces over Bezier algorithm. (8)

(ii)Explain the different types of data with the techniques of visualization applied over a data. (8)

13. (a) Compare and contrast the various colour models in detail(Pg No 89) (16)

Or

(b) Explain the structure and primitives of Graphics programming using OPENGL. (16)

14. (a) (i) Explain the process of mapping texture over a cylindrical surface. (8)

(ii) Explain the vector interpolation technique used by Phong Shading model (8)

Or

(b) (i) How does environment mapping differ from surface texturing process? What is the effect of any directional light source? (8)

(ii) Explain the process of drawing shadows for modelled objects (8)

15. (b)(i) Explain random midpoint displacement method of modeling terrains. (8)

(ii) Explain the ray tracing method of rendering three dimensional objects. (8)

Or

(b) Write short notes on:

(i)Boolean operation on objects. (5)

(ii)Grammer based models. (5)

(iii) Self squaring fractals. (6)

B.E/B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2015.

FIFTH SEMESTER

Computer Science and Engineering

CS 6504-COMPUTER GRAPHICS

Time: Three hours **Maximum:100** Marks

Answer All Questions

PART-A (10 x 2 =20 Marks)

1. Compute the resolution of a 2 x 2 inch image that has 512 x 512 pixels.(pg. no12)
2. Give the contents of the display file. (pg. no12)
3. Derive the general form of scaling matrix about a fixed point (X_f, yf).
4. Write down the conditions for point clipping window.
5. Represent the parametric representation of a cubic Bezier curve.
6. Define projection plane and centre of projection.
7. Define dithering .Where does this occur?
8. Convert the given colour value to CMY colour mode where R= 0.23, G = 0.57, B=0.11. (pg. no.85)
9. Give the basic principle of animation.
10. List the attributes of turtle in graphics.

PART- (5 x 16 = 80 Marks)

11. (a) (i) Define and differentiate Random scan and Raster scan devices (8)

(ii) Using Bresenhams circle drawing algorithm plot one quadrant of a circle of radius 7 pixels with origin as centre (pg.No 28)

Or

(b) (i) How are event driven input devices handled by the hardware? Explain (8)

(ii) Discuss the primitives used for filling. (8)

12. (a) (i) Flip the given quadrilateral A(10,8),B(22,8),C(34,17),D(10,27) about the origin and then zoom it to twice its size. Find the new positions of the quadrilateral . (8)

(ii) Derive the viewing transformation matrix. (8)

Or

(b) (i) Clip the given line A(1,3),B(4,1) against a window p(2,2) Q(5,2) R(5,2) S(2,4) using Liang Barsky line clipping algorithm (8)

(ii)Explain the two dimensional viewing pipeline in detail (8)

13. (a) (i) Derive the parametric equation for a cubic Bezier curve.(pg no 70) (8)

(ii) Compare and contrast orthographic, Axonometric and Oblique projections (8)

Or

(b) (i) Write down the Back face detection algorithm. (8)

(ii) How will you perform three dimensional rotations about any arbitrary axis in space? (8)

14. (a) Discuss on colour spectrum, colour concepts and colour models in detail (16)

Or

(b) Explain the illumination models in detail.(Pg.no 97) (16)

15. (a) (i) Distinguish between raster animation and key frame animation in detail (8)

(ii) How will you generate grammar based model? Explain.(Pg.no 113)(8)

Or

(b) Write short notes on:

(i) Ray tracing (6)

(ii) Koch curves (5)

(iii) Morphing (5)

B.E/B.Tech. DEGREE EXAMINATION, MAY/JUNE 2016.

FIFTH SEMESTER

Computer Science and Engineering

CS 6504-COMPUTER GRAPHICS

(Regulation 2013)

Time: Three hours Maximum:100 Marks

Answer All Questions

PART-A (10 x 2 =20 Marks)

1. Define refresh/frame buffer. (pg. no12)
2. What are the merits and demerits of direct view storage tubes?(Pg no 11)
3. Define Shear. ?(Page no 32)
4. Define Window. ?(Page no 32)
5. Differentiate parallel projection from perspective projection. (Pg no 54)
6. What is the need for space portioning representations?
7. What is the need for shading model?
8. List out various properties that describe the characteristics of light? (pg. no 84)
9. What is a scripting system? (pg. no 102)
10. What is a turtle graphics program? (pg. no 100)

PART-B (5 x 16 = 80 Marks)

11. (a) Explain in detail about the Line drawing DDA scan conversion algorithm with example (pg .no 14) (16)

Or

(b) Explain the following Video Display Devices(pg .no 17)

(i) Refresh cathode ray tube (4)

(ii) Raster scan systems (4)

(iii) Random scan Displays (4)

(iv) Colour CRT Monitors (4)

12. (a) Explain on the following 2D transformations

(i) General Pivot Point rotation (4)

(ii) General Fixed Point Scaling (4)

(iii) Perform 45 degree rotation of a triangle A(90,0), B(1,1) and C(5,3) about P(-1,-1) (8)

Or

(b) Explain in detail the Cohen-Sutherland line clipping algorithm with example (16)

13.(a) Write notes on: (pg.no 56)

(i) Quadric surfaces (8)

(ii) Polygon surfaces (8)

Or

(b) Explain a method to rotate an object about an axis that is not parallel to the coordinate axis With a neat block diagram and derive the transformation matrix for the same (pg.no 54) (16)

14. (a) Explain in detail on RGB and HSV colour models (Pg.no 86) (16)

Or

(b) Write notes on Phong model and Warn model in detail.(Pg.no 97) (16)

15. (a) (i) Define animation sequence. Explain the various steps involved in animation sequence. (Pg.no 102) (8)

(ii) What is Koch Curve? Explain in detail. (Pg.no 106) (8)

Or

(b) (i) Explain Raster Animation. (Pg.no 103) (8)

(ii) What is Fractal? Explain in detail the various fractals.(Pg. 110) (8)

B.E/B.Tech. DEGREE EXAMINATION, MAY/JUNE 2016.

SEVENTH SEMESTER

Computer Science and Engineering

CS 2401-COMPUTER GRAPHICS

(Regulation 2008)

Time: Three hours **Maximum:100 Marks**

Answer All Questions

PART-A (10 x 2 =20 Marks)

1. Reflect all the given triangle about X-axis whose coordinates are A(4,1) B(5,2) C(4,3) and find out the new coordinates. (Pg No 36)
2. Define resolution. (pg. no12)
3. Write down the significance of Vanishing point.
4. A unit cube is located at the origin. Find the oblique view (Parallel Projection) on XY plane with light ray falling along the line (1,0,0) and (9,0,6)
5. How are intermediate frames generated in animation ?
6. When do you use init function in Opengl ?
7. Differentiate specular reflection and diffuse reflection.(pg.no 82)
8. How will you create a frame buffer using Opengl ?
9. Calculate the fractal dimensions of a sierpinski triangle.
10. Write an two applications where constructive solid geometry is applied.

PART-B(5 x 16 = 80 Marks)

11. (a) (i) Scan convert a straight line whose end points are (5,10) and (15,35) using Bresenhams line drawing algorithm f or an aspect ratio 1:1. (8)

(ii) Apply Sutherland Hodgman algorithm to the pentagon whose vertices are A (16,7) B (24, 7) C (36, 10) D (27, 19) E (6, 12). The Window edges are x=10 and 30 and Y=51 and 15 (8)

Or

(b) (i) The matrix { 1 a b 1 } Defines a shearing transformation . When b=0 the shearing is in the x direction and when a=0 the shearing is in the y

direction. Demonstrate the effect of these shearing transformation on the square A(0,0) B(2,0) C(2,2) and D(0,2) when a=3 and b=4. (8)

(ii) Scan convert an ellipse with major and minor axes as 5 and 3 units and centered at origin (8)

12. (a) (i) List the properties of Bezier curves (8)

(ii) Determine the coordinates of the rectangular block with x, y and z dimensions of 4,6 and 8 units with the block entirely in the first quadrant transformed about a focus (2,-1,3) (8)

Or

(b) (i) Discuss on information visualization techniques in detail (8)

(ii) For an unit cube whose (0,0,0,) is centered at origin, eliminate the hidden lines using Back face detection method (8)

13. (a) (i) Compare and contrast RGB and YIQ color models in detail (8)

(ii) Write down the code snippet to set the camera in Opengl for parallel projection (8)

Or

(b) (i) Discuss about various animation techniques in detail (8)

(ii) With code snippets explain the process of reading a scene description from a file Using SDL (8)

14. (a) Explain the process of adding textures to faces in detail (16)

Or

(b) Explain the technique of Bump mapping and non photo realistic photo rendering in Detail (16)

15. (a) (i) Explain the process of ray tracing a scene using pseudo code skeleton of a ray tracer. (8)

(ii) How will you build a projection extent for each node of a CSG object ?

(8)

Or

(b) Write short note on:

(i) Ray tracing Vs Ray casting (8)

(ii) Intersecting rays with cube (8)

(iii) Shadow addition (8)