Chapter - 1

Introduction and Application

Questions

1) Differentiate Random and Raster display technology. [4]

2) Write short notes on:

a) Raster scan display [5] b) Application of computer graphics [3] c) Flat panel display [5]

3) Consider a raster scan system having 12 by 10 inch screen with resolution of 100 pixels per inch in each direction. If the display controller of this system refreshes the screen at rate of 50 frames per second, how many pixels could be accessed per second and what is the access time per pixel of the system? [4]

4) How much time is spent scanning across each pixels during screen refresh on a raster system with resolution 1024 x 768 and refresh rate of 60 frames per second? [4]

5) What is the size of frame buffer required to store SVGA with 24 bit true color video of 10 min without compression? [4]

Purpose of Computer Graphics

→ Computer graphics is a field related to generation of graphics using computers comprising the process of creation, storage, processing and manipulation of images of objects

→ The sole purpose of computer graphics is to provide handy access to the user to control contents, structure and appearance of images by using input devices.

→ It also allows easy interaction between user and computer system.

Application of Computer Graphics

1) Computer Aided Design (CAD)

→ Interactive designs of components and systems of mechanical, electrical, electronic and structural fields.

2) User Interface

→ Implementation of user interfaces for managing multiple simultaneous activities and various on click and on point facilities.

→ Eg: word processing, desktop publishing, application development tools, etc.

3) Plotting and Visualization

→ Plotting of graphs to represent complex data and visualize them easily.

→ Eg: bar graph, pie chart, etc.

4) Image Processing

→ Controlling the image quality

5) Simulation

→ Used to displays the imitation of conditions

6) Art and Commerce
   → Produces pictures delivering attentive messages.
   → Slide productions.

7) Education and Cartography
   → Models the physical system for easy and interactive learning.
   → Production of maps based on data.

8) Entertainment
   → Movies and animations production
   → Computer and video games.

Input Devices
→ Those hardwares responsible for providing information to a computer system
→ Eg: keyboard, mouse, trackball, touchpad, joystick, scanner, data gloves, light pen, digitizer / tablet

Hard Copy Output Devices
→ Those hardwares that provides output in tangible form.
→ Eg: printer and plotter.

Cathode Ray Tube (CRT)

→ A beam of electrons emitted by an electron gun, passes through focusing and deflection systems that direct the beam toward specified positions on phosphor coated screen.

→ Electron gun contains heated metal cathode and control grid. Heat is supplied to the cathode by directing a current through a coil of wire (filament).

→ Control grid is used to control intensity of electron beam by setting voltage levels.

→ Focusing system is used to force electron beam to converge into a small spot as it strikes the phosphor. It is done either with electric or magnetic field.

→ Deflection system controls the deflection of the electron beam to specified position on the screen.

→ Light emitted by phosphor fades very quickly, so repeatedly the electron beams are directed back over the same points to maintain screen picture. Such display is called refresh buffer.

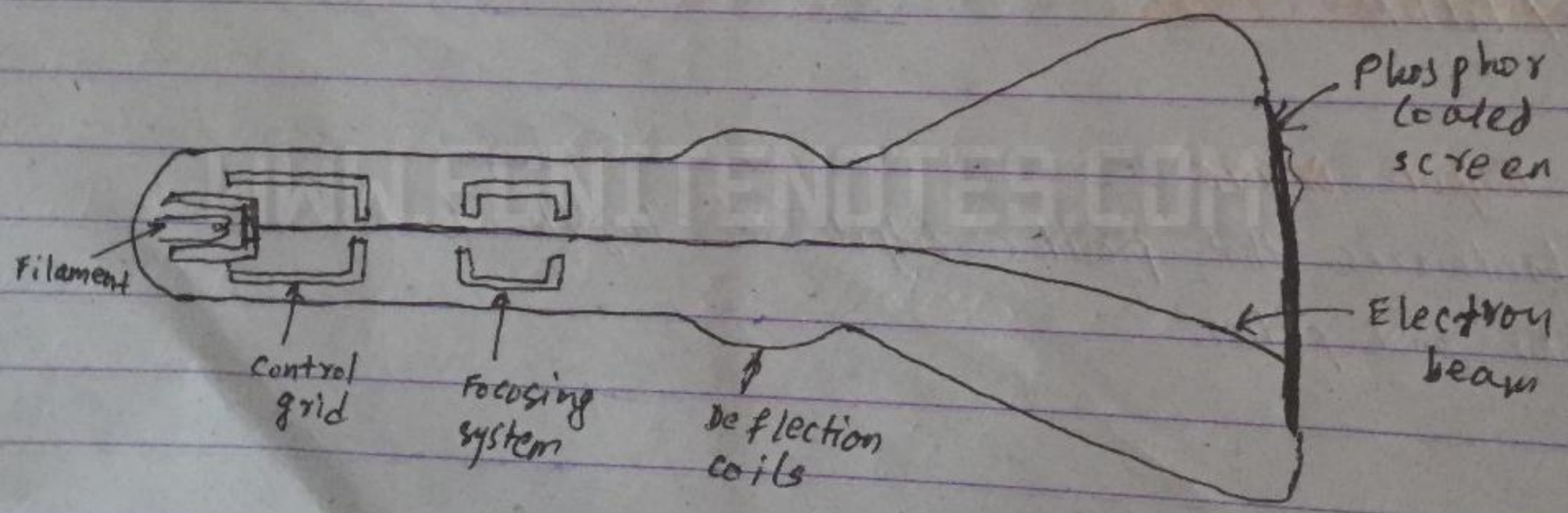


Fig. Basic design of CRT

Flat Panel Display

→ Reduced volume, weight and power requirements.
→ Two types: emissive and non-emissive displays.
→ Emissive displays converts electrical energy into light
  # Plasma panel, thin-film electroluminescent display, LED
→ Non-emissive displays uses optical effects to convert light from some other sources into graphics pattern. Eg: Liquid crystal display.

Resolution

→ The maximum number of points that can be displayed without overlap on a CRT is called resolution.

Aspect Ratio

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

Raster-Scan Display

Raster Scan Display

→ The electron beam is swept across the screen, one row at a time from top to bottom.
→ The picture definition is stored in a memory area i.e. refresh/frame buffer, which holds set of intensity values for all screen points.
→ Each screen point is called pixel.
→ It is useful for display of scenes with shading and color patterns.
→ In B/W system, each screen point is either on or off i.e. one bit per pixel.
→ For color systems, additional bit is required.
→ The return to left of the screen after refreshing each scan line is called horizontal retrace.
→ The return to top left corner of screen to begin next frame at the end of each frame is called vertical retrace.
→ It is used for interlacing technique.

Random Scan Display

→ The electron beam directs only to the part of screen where a picture is to be drawn.
→ Also called vector display as it draws picture one line at a time.
→ Refresh rate depends on no. of lines to be displayed.
→ Picture definition is stored in refresh buffer as a set of line drawing commands.

Differences: Raster and Random scan display

Raster Scan display	Random Scan display.
a) It is suitable for displaying shading and color area.	a) It is restricted to line drawing applications.
b) Interlacing technique is used.	b) No interlacing technique.
c) Low resolution.	c) High resolution.
d) Pattern fill is easy	d) Pattern fill is difficult.
e) Editing is difficult.	e) Editing is easy.
f) Refresh area is independent of picture complexity.	f) Refresh area depends on picture complexity.
g) Produces jagged line.	g) Produces smooth line.
h) Eg: CRT, printer	h) Eg: Plotter

→ If n is no. of bits used to store color information for a pixel.

∴ Total no. of possible colors = 2^n

∴ Memory to store image N*M pixels with color depth of n bits = $\frac{N*M*n}{8}$ (bytes)

→ For n bit plane, intensity level varies from 0 to 2^n-1.

8.3 Ans ⟹

Screen size = 12 by 10 inch

Resolution = 100 pixel per inch in each direction

= (12 x 100) * (10 x 100)

= 1200 x 1000 pixels

Refresh rate = 50 frames per second

Pixel accessed per second = ?

Access time per pixel = ?

Now,

Here, 1 frame consists of 1200 x 1000 pixels.

∴ 50 frames consists of 1200 x 1000 x 50 pixels.

= 60000000 pixels

∴ Pixel accessed per second = 60000000 pixels // Ans.

Also,

Access time per pixel = $\frac{1}{60000000}$ seconds

= 1/60 ms // Ans.

8.4 Ans ⟹ Resolution = 1024 × 768

Refresh rate = 60 frames per second

= 47185920 pixels per second

∴ Time spent to scan across each pixel = $\frac{1}{47185920}$ second

= 21.19 ns ⟹ Ans.

8.5 Ans ⟹

no. of bits to store color = 24 bits per pixel

time taken = 10 min = 10 × 60 sec = 600 sec

Chapter - 2

Scan Conversion

2.1 Scan - Converting a point

2.2 Scan - Converting a straight line

→ DDA Line Algorithm

→ Bresenham's Line Algorithm

2.3 Scan - Converting a circle and an ellipse

→ Mid point circle algorithm

→ Mid point ellipse algorithm

[10 - marks]

Questions

1) Compare DDA and Bresenham's algorithm. Derive and write mid point algorithm to draw ellipse. [10]

~~2) Write down co~~

2) Derive Bresenham's decision parameter to draw a line with negative slope and $|m| > 1$. [8]

3) What is scan conversion? [2]

4) Derive Bresenham's decision parameter to draw a line moving from left to right and having negative slope. [8]

5) State the condition to identify you are in second region of ellipse using mid point algorithm. [2]

6) Mention disadvantages of DDA. Write Bresenham's line drawing algorithm. [2+4]

) Derive decision parameters for midpoint circle algorithm assuming start position as $(r, 0)$ and points are to be generated along curve path in counter clock wise order. What is symmetry property? [8+2]

What are advantages of Bresenham's over DDA algorithm? [4]

2.1 Scan Conversion

Scan conversion is the process of representing the graphical objects of continuous form as a collection of numerous discrete pixels pattern. For eg: a graphical object i.e. line can be defined by its two end points and a line equation.

2.2 DDA Line Algorithm

The DDA line drawing algorithm is given as follows:-

1) Start

2) Get input of two end points i.e. (x_0, y_0) and (x_1, y_1).

3) Calculate difference between end points.

$dx = x_1 - x_0$

$dy = y_1 - y_0$

4) If $dx > dy$

steps = absolute (dx)

else

steps = absolute (dy)

5) Calculate increment in x and y coordinates

x_increment = dx / (float) steps

y_increment = dy / (float) steps

6) Initialize v=0

7) Put pixel at (x_0, y_0) ~~and initialize v=0~~

8) $x_0 = x_0 + x\_increment$

$y_0 = y_0 + y\_increment$

$v = v+1$

9) If $v <$ steps

Goto step 7

else

Goto step 10

10) Stop

2.2.2 Advantages of DDA

1) It is simple and easy to implement.
2) It is faster than direct use of line equation.

2.2.3 Disadvantages of DDA

1) It involves floating point arithmetic, which is time consuming.
2) It is orientation dependent. So, end point accuracy is poor.

2.3 Bresenham Line Drawing Algorithm

Bresenham line algorithm is an incremental scan conversion algorithm that uses only integer calculations.

2.3.1 Decision Parameters

→ ~~Negative~~ Positive slope and $|m| < 1$

Consider a line to be drawn with positive slope such that $|m| < 1$. In this case, both x and y increases as we move from left to right. Here, the pixel positions along a line path are determined by sampling at unit x intervals starting from left end point (x_0, y_0). We step to each successing x position and plot pixel whose scan line y value is closest to line path.

Assume we determine pixel at (x_k, y_k) is displayed, now we need to decide which pixel to plot in column x_{k+1} i.e either (x_k+1, y_k) or (x_k+1, y_k+1).

At sampling position x_k+1, we label vertical pixel separations from mathematical line path as d_1 and d_2 as shown in fig (2). The y coordinate on mathematical line at pixel position x_k+1 is given by:

$$y = m(x_k+1) + b \quad \text{---} \quad (1)$$

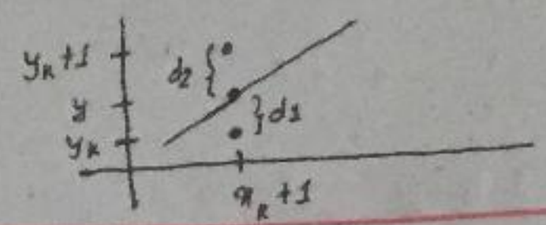


Fig. (2)

Now,

$$d_1 = y - y_k = m(x_k+1) + b - y_k$$

$$d_2 = (y_k+1) - y = y_k + 1 - m(x_k+1) - b$$

The difference of these two separations is:

$$d_1 - d_2 = 2m(x_k+1) - 2y_k + 2b - 1 \quad \text{——} (2)$$

A decision parameter p_k for k^{th} step can be obtained by rearranging eq^n (2) so that it involves only integer calculation.

$$\text{i.e } p_k = (d_1 - d_2)\Delta x = 2\Delta y\, x_k - 2\Delta x\, y_k + c \quad \text{——} (3)$$

where, $c = 2\Delta y + \Delta x(2b-1)$ i.e. independent of pixel position.

The sign of p_k is same as sign of $d_1 - d_2$ as $\Delta x > 0$.

If pixel at y_k is closer to line path (i.e. $d_1 < d_2$), then p_k is negative and we plot lower pixel; otherwise we plot upper pixel.

At step $k+1$, the decision parameter is:

$$p_{k+1} = 2\Delta y\, x_{k+1} - 2\Delta x\, y_{k+1} + c \quad \text{——} (4)$$

Subtracting (4) from (3);

$$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

But $x_{k+1} = x_k + 1$, so

$$\boxed{p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)} \quad \text{——} (5)$$

Here, $(y_{k+1} - y_k)$ is either 0 or 1, depending on sign of p_k. For p_k to be negative, $y_{k+1} - y_k = 0$; otherwise $y_{k+1} - y_k = 1$.

The first parameter p_0 can be evaluated as:

$$\boxed{p_0 = 2\Delta y - \Delta x}$$

→ Positive slope and $|m| > 1$

$$p_{k+1} = p_k + 2\Delta x - 2\Delta y (x_{k+1} - x_k)$$

→ Negative slope and $|m| < 1$

Here, one coordinate decreases as the other increases.

2.3.2 Algorithm

1) Start

2) Input two end points of line i.e. (x_1, y_1) and (x_2, y_2)

3) Compute $\Delta x = |x_2 - x_1|$ and $\Delta y = |y_2 - y_1|$

4) if $(x_2 > x_1)$, $lx = 1$

otherwise, $lx = -1$

5) if $(y_2 > y_1)$, $ly = 1$

otherwise, $ly = -1$

6) plot (x_1, y_1)

7) If $\Delta x > \Delta y$

a) Calculate $p_0 = 2\Delta y - \Delta x$

b) Starting at $k=0$ to Δx times, repeat:

i) If $p_k < 0$

$$x_1 = x_1 + lx, \quad y_1 = y_1$$

$$p_k = p_k + 2\Delta y$$

otherwise,

$$x_1 = x_1 + lx, \quad y_1 = y_1 + ly$$

$$p_k = p_k + 2\Delta y - 2\Delta x$$

ii) plot (x_1, y_1)

8) If $\Delta x < \Delta y$

a) Calculate $p_0 = 2\Delta x - \Delta y$

b) Starting at $k=0$ to Δy times, repeat:

i) If $p_k < 0$

$x_s = x_s$, $y_s = y_s + 1y$

$p_k = p_k + 2\Delta x$

otherwise,

$x_s = x_s + 1x$, $y_s = y_s + 1x$

$p_k = p_k + 2\Delta x - 2\Delta y$

ii) plot (x_s, y_s)

9) Stop

2.4 Mid point circle algorithm

2.4.1 Decision Parameter

Consider start position to be $(0, r)$ and points to be generated along curve path in clock wise order to draw a circle with radius 'r' and center $(0,0)$. Along circle section from $x=0$ to $x=y$ in first quadrant, the slope varies from 0 to -1. Therefore, we can take unit steps in positive x-direction over this octant and use decision parameter to determine which of two possible y positions is closer to circle path at each step.

To apply mid point method, we define circle function as:

$$f_{circle}(x,y) = x^2 + y^2 - r^2 \quad \text{—①}$$

The relative position of any point (x,y) can be determined by:

$$f_{circle}(x,y) \begin{cases} <0 & ; (x,y) \text{ is inside circle boundary} \\ =0 & ; (x,y) \text{ is on circle boundary} \\ >0 & ; (x,y) \text{ is outside circle boundary} \end{cases} \quad \text{—②}$$

This test is performed for midpoints between two candidate pixels at sampling position x_k+1. Assume, we plot pixel at (x_k, y_k). We need to determine whether pixel at position (x_k+1, y_k) or (x_k+1, y_k-1) is closer to circle. The function of eqn ① for midpoint between these points gives the decision parameter.

$$\therefore p_k = f_{circle}(x_k+1, y_k - 1/2) = (x_k+1)^2 + (y_k - 1/2)^2 - r^2 \quad \text{—③}$$

If $p_k < 0$, midpoint is inside circle and pixel on scan line y_k is closer to circle boundary. Otherwise, we select pixel on scanline $y_k - 1$.

For next pixel at sampling position $x_{k+1} + 1 = x_k + 2$;

$$p_{k+1} = f_{circle}(x_{k+1} + 1, y_{k+1} - 1/2)$$
$$= [(x_k + 1) + 1]^2 + (y_{k+1} - 1/2)^2 - r^2 \quad \text{—— (4)}$$

Now,

$$p_{k+1} - p_k = (x_k^2 + 4x_k + 4 + y_{k+1}^2 - y_{k+1} + 1/4 - r^2) - (x_k^2 + 2x_k + 1 + y_k^2 - y_k + 1/4 - r^2)$$

or, $\boxed{p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1}$

where, y_{k+1} is either y_k or $y_k - 1$, depending on sign of p_k.

So,

If p_k is negative;

$$\boxed{p_{k+1} = p_k + 2(x_k + 1) + 1}$$

If p_k is positive;

$$\boxed{p_{k+1} = p_k + 2(x_k + 1) - 2(y_k - 1) + 1}$$

At start position $(0, r)$;

$$p_0 = f_{circle}(1, r - 1/2)$$
or, $p_0 = 1 + (r - 1/2)^2 - r^2$
or, $p_0 = 1 + r^2 - r + 1/4 - r^2$
$$\therefore \boxed{p_0 = 5/4 - r}$$

If radius r specified as an integer, we can simply round p_0 to:

$$\boxed{p_0 = 1 - r}$$

2.4.2 Algorithm

1) Start

2) Input radius r and center (x_c, y_c)

3) Obtain first point on circumference of a circle centered on origin.

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

4) Calculate initial decision parameter.

$p_0 = 5/4 - r$

5) At each x_k position, starting at $k = 0$:

a) If $p_k < 0$,

i) next point along circle centered on origin is (x_{k+1}, y_k)

ii) $p_{k+1} = p_k + 2x_{k+1} + 1$ such that $x_{k+1} = x_k + 1$

b) otherwise,

i) next point along circle is $(x_k + 1, y_k - 1)$

ii) $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$ such that $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k - 1$.

6) Determine symmetry points in the other seven octants.

7) Move each pixel position (x,y) onto circular path centered on (x_c, y_c)

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

and plot (x,y)

8) Repeat step 5 through 7 until $x \geq y$

9) Stop

2.5 Mid point ellipse algorithm

2.5.1 Decision Parameter

Let an ellipse with major axis 'r_x', minor axis 'r_y' and center (x_c, y_c) is to be drawn. In the first quadrant, ellipse can be divided into two regions according to slope of an ellipse. It is processed by taking unit step in x direction where $|m| < 1$ and by taking unit step in y direction where $|m| > 1$.

Consider $(0, r_y)$ be the start position and processing is done along clockwise order. The ellipse function can be defined as:

$$f_{ellipse}(x,y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2 \quad \text{——} (1)$$

with the properties defined as:

$$f_{ellipse}(x,y) \begin{cases} <0 & ; (x,y) \text{ is inside ellipse boundary} \\ =0 & ; (x,y) \text{ is on ellipse boundary} \\ >0 & ; (x,y) \text{ is outside ellipse boundary} \end{cases} \quad \text{——} (2)$$

[redacted] region

At the boundary between region 1 and regim 2, the slope is given as:

$$\frac{dy}{dx} = -\frac{2r_y^2 x}{2r_x^2 y} = -1$$

$$\therefore 2r_y^2 x = 2r_x^2 y$$

Assume (x_k, y_k) has been plotted in region 1 and the next position is to be determined. So, the decision parameter is given by the function of ellipse at midpoint of (x_k+1, y_k) and (x_k+1, y_k-1).

$$\therefore p1_k = f_{ellipse}(x_k+1, y_k - 1/2)$$

$$\therefore p1_k = r_y^2 (x_k+1)^2 + r_x^2 (y_k - 1/2)^2 - r_x^2 r_y^2$$

If $p1_k < 0$, midpoint is inside ellipse and pixel on scan line y_k is closer, otherwise we select pixel on scan line $y_k - 1$.

At next sampling position $x_{k+1}+1 = x_k + 2$,

$$p1_{k+1} = f_{ellipse}(x_{k+1}+1, y_{k+1} - 1/2)$$
$$= r_y^2[(x_k+1)+1]^2 + r_x^2(y_{k+1} - 1/2)^2 - r_x^2 r_y^2$$

Now,

$$p1_{k+1} - p1_k = 2(x_k+1)r_y^2 + r_y^2 + r_x^2(y_{k+1}^2 - y_{k+1} + 1/4 - y_k^2 + y_k - 1/4)$$
$$= 2(x_k+1)r_y^2 + r_x^2(y_{k+1}^2 - y_k^2) - r_x^2(y_{k+1} - y_k) + r_y^2$$

$$\therefore p1_{k+1} = p1_k + 2r_y^2(x_k+1) + r_x^2(y_{k+1}^2 - y_k^2) - r_x^2(y_{k+1} - y_k) + r_y^2$$

where, y_{k+1} is either y_k or $y_k - 1$, depending on sign of $p1_k$.

If $p1_k < 0$, $y_{k+1} = y_k$

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

If $p1_k > 0$, $y_{k+1} = y_k - 1$

$$\therefore p1_{k+1} = p1_k + 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}$$

The initial value of decision parameter is:

$$p1_0 = f_{ellipse}(1, r_y - 1/2)$$

$$\therefore p1_0 = r_y^2 - r_x^2 r_y + 1/4\, r_x^2$$

For region 2, we can write:

$$p2_{k+1} = p2_k - 2r_x^2(y_k - 1) + r_x^2 + r_y^2(x_{k+1}^2 - x_k^2) + r_y^2(x_{k+1}^2 - x_k)$$

$$\therefore p2_0 = f_{ellipse}(x_0 + 1/2, y_0 - 1)$$

2.5.2 Algorithm

1) Start

2) Input r_x, r_y and center (x_c, y_c).

3) Obtain starting point of ellipse centered on origin

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

4) Calculate initial value of decision parameter in region 1.

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

5) At each x_k in region 1, starting at $k = 0$;

a) If $p1_k < 0$

i) next point on ellipse is (x_{k+1}, y_k)

ii) $p1_{k+1} = p1_k + 2r_y^2 x_{k+1} + r_y^2$

b) Otherwise,

i) next point on ellipse is $(x_k + 1, y_k - 1)$

ii) $p1_{k+1} = p1_k + 2r_y^2 x_{k+1} + r_y^2 - 2r_x^2 y_{k+1}$

6) Repeat step 5 until $2r_y^2 x \geq 2r_x^2 y$.

7) Calculate initial value of decision parameter in region 2.

$$p2_0 = r_y^2 (x_0 + \frac{1}{2})^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

8) At each y_k in region 2, starting at $k = 0$;

a) If $p2_k > 0$,

i) next point is $(x_k, y_k - 1)$

ii) $p2_{k+1} = p2_k - 2r_x^2 y_{k+1} + r_x^2$

b) Otherwise,

i) next point is $(x_k + 1, y_k - 1)$

ii) $p2_{k+1} = p2_k - 2r_x^2 y_{k+1} + r_x^2 + 2r_y^2 x_{k+1}$

9) Determine symmetry points in other quadrants

10) Shift pixel position (x, y) on ellipse centered (x_c, y_c) as:

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

and plot (x, y).

11) Stop.

Problems

1) Draw line from (0,0) to (6,6) using DDA algorithm.

Sol^n,

$(x_1, y_1) = (0, 0)$

$(x_2, y_2) = (6, 6)$

Now,

$|x_2 - x_1| = 6$

$|y_2 - y_1| = 6$

$\therefore$ steps = 6

Also,

$$x\text{-inc} = \frac{|x_2 - x_1|}{\text{steps}} = 1$$

$$y\text{-inc} = \frac{|y_2 - y_1|}{\text{steps}} = 1$$

$\therefore x = x_1 + 0.5 * sign(\Delta x) = 0.5$

$\therefore y = y_1 + 0.5 * sign(\Delta y) = 0.5$

In tabular form;

i	Plot	x	y
1	(0, 0)	0.5	0.5
2	(1, 1)	1.5	1.5
3	(2, 2)	2.5	2.5
4	(3, 3)	3.5	3.5
5	(4, 4)	4.5	4.5
6	(5, 5)	5.5	5.5
7	(6, 6)	6.5	6.5

=> Digitize line with end points (20,10) and (30,18) with slope of 0.8, $\Delta x = 10$ and $\Delta y = 8$.

Sol^n,

Here, $(x_1, y_1) = (20, 10)$

$(x_2, y_2) = (30, 18)$

$\therefore \Delta x = x_2 - x_1 = 10$

$\Delta y = y_2 - y_1 = 8$

Since, $x_2 > x_1$; $lx = 1$

$y_2 > y_1$; $ly = 1$

Also,

$\Delta x > \Delta y$, so the initial decision parameter is:

$\therefore p_0 = 2\Delta y - \Delta x = 2 \times 8 - 10 = 6$

And,

The increments are,

$2\Delta y = 16$; $p_k < 0$

$2\Delta y - 2\Delta x = -4$; $p_k > 0$

Plot the initial point (20,10) and other pixel positions are tabulated below:-

k	p_k	(x_{k+1}, y_{k+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)

3) Digitize a circle with radius 6 and centre (20, 20) in first quadrant.

Sol^n,

Here, $r = 6$

$(x_c, y_c) = (20, 20)$

Let $(x_0, y_0) = (0, 6)$ be the first point on circumference of circle centered origin.

The initial value of decision parameter is:

$p_0 = 1 - r = 1 - 6 = -5$

The increments for decision parameters are:

$2(x_k + 1) + 1 = 3$; $p_k < 0$

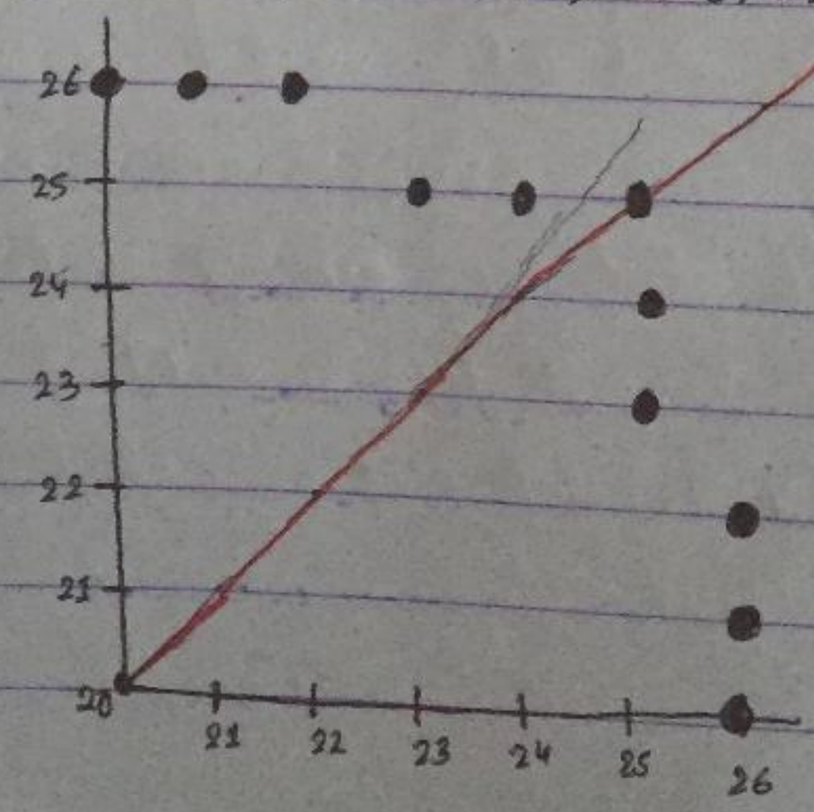
$2(x_k + 1) + 1 - 2(y_k - 1) = -7$; $p_k > 0$

The successive pixels are tabulated below for first octant.

k	p_k	(x_{k+1}, y_{k+1}) For circle centered (0,0)	(x, y) For circle centered (x_c, y_c)
0	-5	(1, 6)	(21, 26)
1	-2	(2, 6)	(22, 26)
2	1	(3, 5)	(23, 25)
3	-6	(4, 5)	(24, 25)
4	-3	(5, 5)	(25, 25)

Using symmetry property, the pixels on second octant of first quadrant are:-

(25, 24), (25, 23), (26, 22), (26, 21).



Chapter - 3

2 - D Transformations

3.1 2D Transforms

→ Translation

→ Rotation

→ Scaling

→ Reflection

→ Shear

3.2 2D Composite Transforms

3.3 2D Viewing

→ Viewing pipeline

→ World to screen viewing transformation

→ Clipping

* Cohen - Sutherland

* Liang - Barsky

[10 - marks]

Questions

1) The reflection along line $y=x$ is equivalent to reflection along x-axis followed by counter clockwise rotation by α degree. Find α. [10]

2) Write condition for point clipping. [2]

3) Explain 2D viewing pipeline. Derive 2D transformation matrix for scaling with respect to an arbitary fixed point. [4+6]

4) Justify with matrix operations that two successive 2D rotations is additive. [10]

5) Find composite transformation matrix for reflection about line $y=mx+c$. [8]

6) Which transformation converts square to rhombus? Obtain reflection matrix to reflect a point about line $y=x$. [3+7]

3.1.1 Translation

A translation is applied to an object by repositioning it along a straight line path from one coordinate location to another. A 2D point is translated by adding translation distances t_x and t_y to original position (x, y) to move to new position (x', y') such that:

$$x' = x + t_x \quad \text{and} \quad y' = y + t_y$$

Let P be a point translated to P' with translation matrix T. Then,

$$P' = P + T$$

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

In terms of column vector,

$$P' = R \cdot P$$

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

Q. Translate square A (0,0), B(5,0), C(5,5), D(0,5) by 2 units in x-direction and 3 units in y-direction.

Sol^n,

The object coordinate matrix : $P = \begin{matrix} A & B & C & D \end{matrix}$

$$\begin{bmatrix} 0 & 5 & 5 & 0 \\ 0 & 0 & 5 & 5 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

The transformation matrix : $R = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}$

Now,

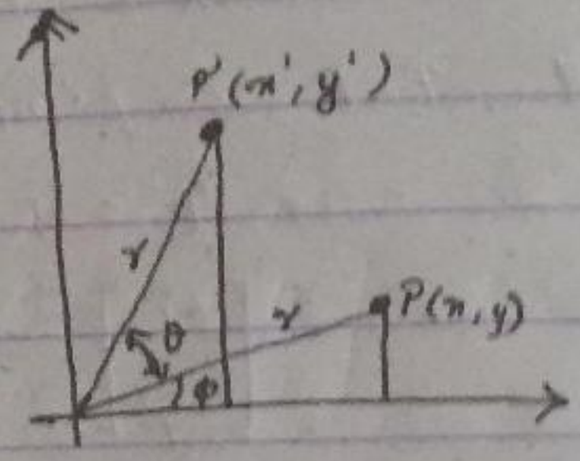
$$P' = R \cdot P = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 5 & 5 & 0 \\ 0 & 0 & 5 & 5 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{matrix} A' & B' & C' & D' \end{matrix} \begin{bmatrix} 2 & 7 & 7 & 2 \\ 3 & 3 & 8 & 8 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Hence, the obtained square after translation is A'(2,3), B'(7,3), C'(7,8) and D'(2,8). // Ans.

3.1.2 Rotation

A rotation is applied to an object by repositioning it along a circular path in the xy plane. For this, a rotation angle 'θ' and rotation point (x_r, y_r) are needed. If θ is positive, rotation is counter clockwise and if θ is negative, rotation is clockwise.

Consider a point P(x,y) is rotated through an angle θ with origin as pivot. to give P'(x';y').



Using trigonometric identities;

$$\left.\begin{aligned} x' &= r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta \\ y' &= r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta \end{aligned}\right\} 1$$

Also,

$$\left.\begin{aligned} x &= r\cos\phi \\ y &= r\sin\phi \end{aligned}\right\} 2$$

From 1 and 2;

$$\left.\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}\right\} 3$$

In matrix form:

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

$$\therefore P' = RP$$

where, $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ = Rotation matrix

For rotating a point about an arbitary pivot (x_r, y_r);

$$\left.\begin{aligned} x' &= x_r + (x-x_r)\cos\theta - (y-y_r)\sin\theta \\ y' &= y_r + (x-x_r)\sin\theta + (y-y_r)\cos\theta \end{aligned}\right\} \text{--- (1)}$$

B) Rotate A(0,0), B(1,0), C(1,1), D(0,1) by 45° about origin.

Sol^n,

$$P = \begin{matrix} A & B & C & D \end{matrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now,

$$P' = R \cdot P = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{matrix} A' & B' & C' & D' \end{matrix} \begin{bmatrix} 0 & 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 0 & 1/\sqrt{2} & \sqrt{2} & 1/\sqrt{2} \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

3.1.3 Scaling

Scaling is the transformation that alters the size of an object by multiplying (x,y) of each point by scaling factors s_x and s_y (>0) to produce (x', y').

i.e. $x' = x \cdot s_x$, $y' = y \cdot s_y$

In matrix form:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\therefore P' = S \cdot P$

The scaling factor less than 1 reduces object size while scaling factor greater than 1 enlarges the object. If both s_x and s_y are assigned same value, it is called uniform scaling. If s_x and s_y are assigned unequal values, it is called differential scaling.

The location of scaled object can be controlled by choosing a fixed point (x_f, y_f) such that:

$$\left.\begin{aligned} x' &= x_f + (x - x_f) s_x = x \cdot s_x + x_f (1 - s_x) \\ y' &= y_f + (y - y_f) s_y = y \cdot s_y + y_f (1 - s_y) \end{aligned}\right\} \text{—①}$$

8) Square ABCD A(0,0), B(3,0), C(3,3), D(0,3) is scaled in 3 units in x-direction and 3 units in y-direction with respect to origin. Find scaled object.

Sol^n,

Object matrix: $P = \begin{matrix} A & B & C & D \end{matrix}$

$$\begin{bmatrix} 0 & 3 & 3 & 0 \\ 0 & 0 & 3 & 3 \end{bmatrix}$$

Transformation matrix: $S = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$

Now,

$$P' = SP = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 0 & 3 & 3 & 0 \\ 0 & 0 & 3 & 3 \end{bmatrix} = \begin{matrix} A' & B' & C' & D' \end{matrix} \begin{bmatrix} 0 & 9 & 9 & 0 \\ 0 & 0 & 9 & 9 \end{bmatrix}$$

3.1.4 Reflection

Reflection is a transformation that produces mirror image of an object relative to an axis of reflection by rotating the object 180° about the axis.

→ Reflection in X-axis ($y = 0$)

$$x' = x$$

$$y' = -y$$

So, $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

→ Reflection in Y-axis ($x = 0$)

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

→ Reflection relative to origin

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

→ Reflection along $y = x$

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

* Clockwise rotation of 45°
* Reflection about X-axis
* Counter clockwise rotation of 45°

→ Reflection along $y = -x$

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

* Clockwise rotation of 45°
* Reflection about Y-axis
* Counter clockwise rotation of 45°

3.1.5 Shear

Shear is the transformation that distorts the shape of an object such that the transformed shape appears as if it has slide over each other.

→ X-direction shear relative to X-axis.

$$\begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix} \qquad \text{i.e. } x' = x + sh_x \cdot y \quad , y' = y$$

→ Y-direction shear relative to Y-axis.

$$\begin{bmatrix} 1 & 0 \\ sh_y & 1 \end{bmatrix}$$

→ x-direction shear relative to other reference line:

$$\begin{bmatrix} 1 & sh_x & -sh_x \cdot y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3.2 Composite Transformation

3.2.1 Translation followed by Translation

Let P be the position in 2D plane given as (x, y) and, (tx_1, ty_1) and (tx_2, ty_2) be two successive translation vectors applied to P to get final transformed position $P'(x', y')$.

Now, P' can be calculated as:

$$P' = T(tx_1, ty_2) \cdot \{T(tx_2, ty_2) \cdot P\}$$

$$= \{T(tx_1, ty_2) \cdot T(tx_2, ty_2)\} \cdot P$$

So, the composite transformation matrix be:

$$T(tx_1, ty_2) \cdot T(tx_2, ty_2) = \begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$= T(tx_1 + tx_2, ty_1 + ty_2)$$

This shows that successive translations are additive.

3.2.2 Successive Rotations

$$P' = R(\theta_2) \cdot \{R(\theta_1) \cdot P\}$$
$$= \{R(\theta_2) \cdot R(\theta_1)\} P$$
$$= R \cdot P$$

The composite transformation matrix be:

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

This shows that successive rotations are additive.

3.2.3 Successive Scaling

$$P' = S(S_{x_2}, S_{y_2}) \cdot \{S(S_{x_1}, S_{y_1}) \cdot P\}$$
$$= \{S(S_{x_2}, S_{y_2}) \cdot S(S_{x_1}, S_{y_1})\} P$$
$$= SP$$

The composite transformation matrix be:

$$S = \begin{bmatrix} S_{x_2} & 0 & 0 \\ 0 & S_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_{x_1} & 0 & 0 \\ 0 & S_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} S_{x_1} \cdot S_{x_2} & 0 & 0 \\ 0 & S_{y_1} \cdot S_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This shows that successive scaling is multiplicative.

3.2.4 General pivot point rotation

1) Translate object so that pivot coincides with origin.
2) Rotate object about origin.
3) Translate object back so that pivat is returned to its original position.

$$P' = T(x_r, y_r) \cdot [R(\theta) \cdot \{T(-x_r, -y_r) \cdot P\}$$
$$= [T(-x_r, -y_r) \cdot \{R(\theta) \cdot T(x_r, y_r)\}] \cdot P$$

The composite transformation matrix is:

$$M = \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r \sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - y_r \sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

3.2.5 General Fixed Point Scaling

1) Translate object so that fixed point coincides with origin.
2) Scale object w.r. to origin.
3) Translate back to original fixed point.

$$P' = [T(-x_f, -y_f) \cdot [S(s_x, s_y) \cdot \{T(x_f, y_f) \cdot P\}]$$
$$= [T(-x_f, -y_f) \cdot \{S(s_x, s_y) \cdot T(x_f, y_f)\}] P$$

The composite transformation matrix is:

$$M = \begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x & 0 & x_f \\ 0 & s_y & y_f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

3.2.6 Reflection about $y = mx + c$

1) Translate line so that it passes through origin.
2) Rotate line onto one of coordinate axis.
3) Reflect object
4) Finally restore line to original position with inverse rotation and translation.

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

$$\begin{bmatrix} 1 & 0 & -0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix}$$

We know,

$m = \tan\theta$

$\therefore \sin\theta = \dfrac{m}{\sqrt{1+m^2}}$, $\cos\theta = \dfrac{1}{\sqrt{1+m^2}}$

So, $M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{1+m^2}} & \frac{m}{\sqrt{1+m^2}} & 0 \\ -\frac{m}{\sqrt{1+m^2}} & \frac{1}{\sqrt{1+m^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{1+m^2}} & -\frac{m}{\sqrt{1+m^2}} & 0 \\ \frac{m}{\sqrt{1+m^2}} & \frac{1}{\sqrt{1+m^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix}$

$$= \begin{bmatrix} \frac{1-m^2}{1+m^2} & \frac{2m}{1+m^2} & -\frac{2cm}{1+m^2} \\ \frac{2m}{1+m^2} & \frac{m^2-1}{1+m^2} & \frac{2c}{1+m^2} \\ 0 & 0 & 1 \end{bmatrix}$$

3.2.7 Reflection along $y=x$ is equivalent to reflection along x-axis followed by counter clockwise rotation by α degree. Find α.

Sol^n,

For reflection along $y=x$;

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

For reflection along x-axis;

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For counter clockwise rotation by α degree.

$$M_2 = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now,

M_1 followed by M_2 is given by;

$M' = M_1 \cdot M_2$

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

$M' = M_2 \cdot M_1$

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

Since M and M' are equivalent;

$\cos\alpha = 0$

$-\sin\alpha = 1$

So,

$-\sin\alpha = 1$

or, $\sin(-\alpha) = 1$

or, $\sin(-\alpha) = \sin 90^\circ$

$\therefore \alpha = -90^\circ$

3.3 2-Dimensional Viewing

3.3.1 Viewing pipeline

→ A world coordinate area selected for display is called window.

→ An area on display device to which window is mapped is called viewport.

→ Window defines what is to be viewed; Viewport defines where it is to be displayed.

→ They are rectangles with edges parallel to coordinate system.

→ The mapping of part of world coordinate to device coordinate is viewing transformation.

Viewing transformation is carried out as follows:-

1) We construct scene in world coordinate using o/p primitives and attributes.

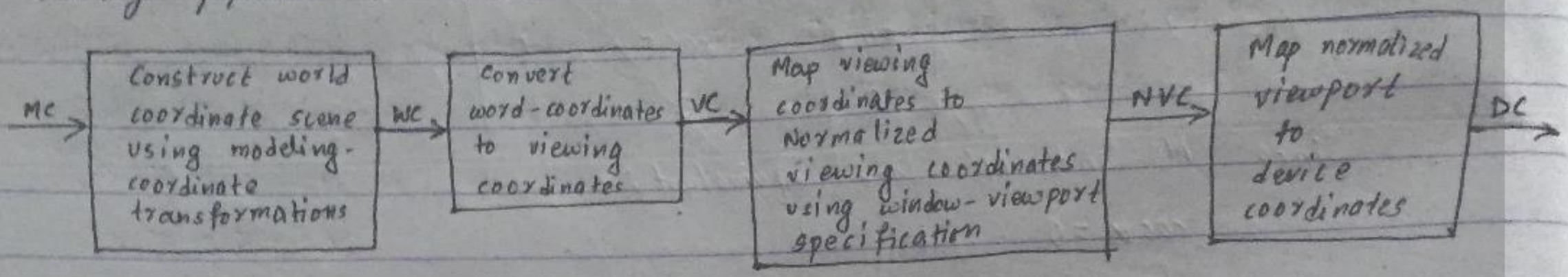
2) To obtain particular orientation for window, we set up 2-D viewing coordinate system in world coordinate and define a window.

3) When viewing reference frame is established, we can transform world coordinates to viewing coordinates.

4) We define viewport in normalized coordinate and map viewing coordinate to normalized coordinate.

5) All parts of picture outside the viewport are clipped and contents of viewport are transferred to device coordinates.

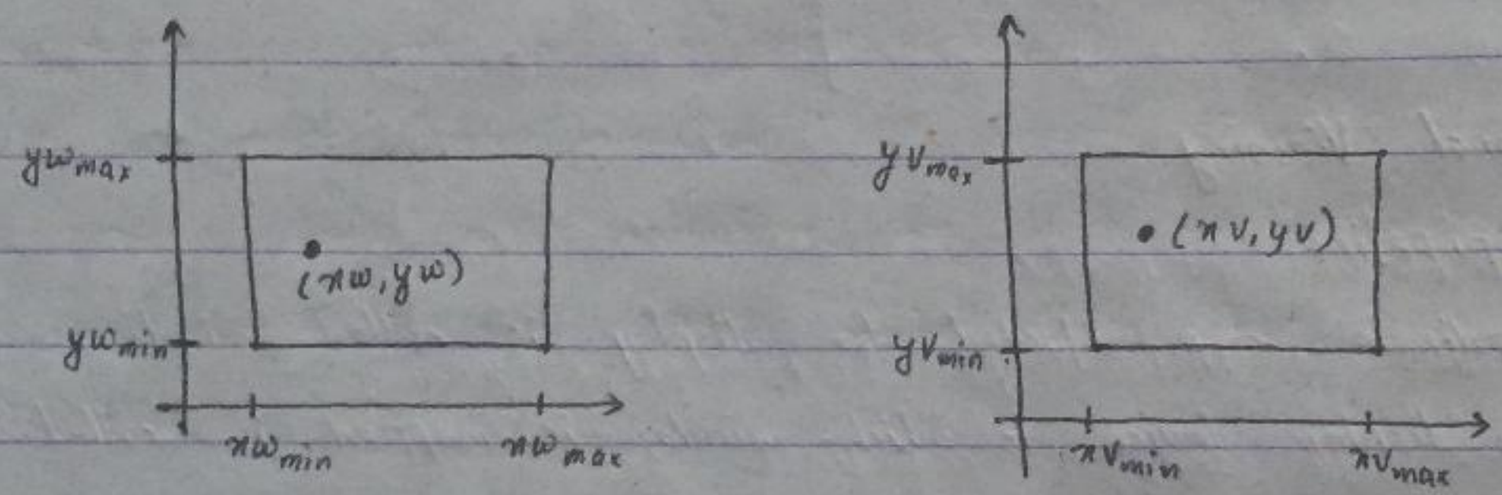
The viewing pipeline is shown below:-



3.3.2 Window to viewport coordinate transformation

After object descriptions have transferred to viewing reference frame, we choose window extents in viewing coordinates and select viewport limits in normalized coordinates.

The window to viewport mapping is shown as:



A point at (xw, yw) in window is mapped into (xv, yv) in the associated viewport such that:

$$\left.\begin{aligned} \frac{xv - xv_{min}}{xv_{max} - xv_{min}} &= \frac{xw - xw_{min}}{xw_{max} - xw_{min}} \\ \frac{yv - yv_{min}}{yv_{max} - yv_{min}} &= \frac{yw - yw_{min}}{yw_{max} - yw_{min}} \end{aligned}\right\} \quad \text{——— (1)}$$

On solving:

$$xv = xv_{min} + (xw - xw_{min})\, sx$$

$$yv = yv_{min} + (yw - yw_{min})\, sy$$

Such that;

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}, \quad s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

This transformation is performed as:

1) Scale using fixed point position of (x_{wmin}, y_{wmin}) that scales window area to size of viewport.

2) Translate scaled window area to position of viewport.

If scaling factors $s_x = s_y$, relative proportion of objects are maintained; otherwise, world objects are stretched or contracted in either x or y direction when displayed on output device.

3.3.3 Clipping

Clipping is the process of identifying those portions of a picture that are either inside or outside of a specified region of space. The region against which an object is to be clipped is called clip window.

3.3.4 Point Clipping

The condition for point clipping is:

1) Assume clip window as a rectangle in standard position, we save point $P(x,y)$ for display if it satisfies:

$$x_{wmin} \leq x \leq x_{wmax}$$

$$y_{wmin} \leq y \leq y_{wmax}$$

where, $(x_{wmin}, x_{wmax}, y_{wmin}, y_{wmax})$ represents edges of clip window.

3.3.5 Line Clipping

A line clipping involves following procedures:-

1) Test line segment to determine whether it lies completely inside clipping window.

2) Test to determine whether it lies completely outside.

3) Perform intersection calculations with one or more clipping boundaries.

3.3.6 Cohen-Sutherland Line Clipping

→ Every line end point is assigned a four digit binary code, called region code that gives location of point relative to clipping rectangle such that:

bit 1 : left

bit 2 : right

bit 3 : below

bit 4 : above

→ If both end points have same bit as 1, it is completely outside.

→ If both end points have region code 0000, it is completely inside.

Algorithm

1) Assign region code for each end point.

2) If both end points have a region 0000, accept those lines.

3) Else, perform logical AND operations for both region codes.

3.1) If result is not 0000, reject those lines.

3.2) Else,

3.2.1) Choose an endpoint that is outside window.

3.2.2) Find intersection point at window boundary.

3.2.3) Replace end point with intersection point and update region code.

3.2.4) Repeat step 2 and 3 until we find clipped line either accepted or reject

4) Repeat steps 1 to 3 for other lines.

5) Stop

Advantages

1) Easy to implement

2) Erly accept or reject tests

Disadvantages

1) Slow to many clipped lines

3.3.7 Liang-Barsky Line Clipping

→ It is based on analysis of parametric equation of line.

i.e. $x = x_1 + u\Delta x$

$y = y_1 + u\Delta y \quad ; \quad 0 \leq u \leq 1$

where,

$\Delta x = x_2 - x_1$

$\Delta y = y_2 - y_1$

→ The clipping window is represented by:

$x_{min} \leq x_1 + u\Delta x \leq x_{max}$

$y_{min} \leq y_1 + u\Delta y \leq y_{max}$

These inequalities can be expressed as:

$u p_k \leq q_k \quad ; \quad k = 1, 2, 3, 4$

where,

p and q are defined as:

$p_1 = -\Delta x$ $\qquad$ $q_1 = x_1 - x_{min}$

$p_2 = \Delta x$ $\qquad$ $q_2 = x_{max} - x_1$

$p_3 = -\Delta y$ $\qquad$ $q_3 = y_1 - y_{min}$

$p_4 = \Delta y$ $\qquad$ $q_4 = y_{max} - y_1$

→ The clipped line will be:

$$\left.\begin{aligned} x_1' &= x_1 + u_1\Delta x \\ y_1' &= y_1 + u_1\Delta y \end{aligned}\right\} \quad u_1 \geq 0$$

$$\left.\begin{aligned} x_2' &= x_1 + u_2\Delta x \\ y_2' &= y_1 + u_2\Delta y \end{aligned}\right\} \quad u_2 \leq 1$$

→ u_1 = max. value between 0 and u for $p_k < 0$; starting value is 0.

→ u_2 = min. value between u and 1 for $p_k > 0$; starting value is 1.

Algorithm

1) Initialize $u_1 = 0$ and $u_2 = 1$.

2) For $k = 1, 2, 3, 4$

2.1) Calculate p_k and q_k

2.2) Calculate r_k

2.3) If $p_k < 0$, find u_1

2.4) If $p_k > 0$, find u_2

2.5) If $p_k = 0$ and $q_k < 0$, reject line and goto 5.

3) If $(u_1 > u_2)$, reject line and goto 5.

4) Find clipped line:

$x_1' = x_1 + u_1 \Delta x$ $\quad$ $x_2' = x_1 + u_2 \Delta x$

$y_1' = y_1 + u_1 \Delta y$ $\quad$ $y_2' = y_1 + u_2 \Delta y$

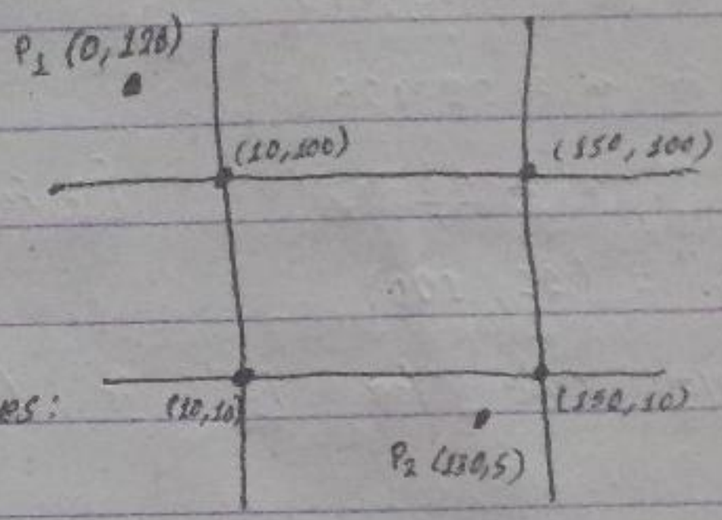
5) Repeat 1 to 4 for other lines

6) Stop

Numericals

1) Obtain end points of a line that connects $P_1(0, 120)$ and $P_2(130, 5)$ after cohen-sutherland clipping. [$xw_{min} = 10$, $yw_{min} = 10$, $xw_{max} = 150$, $yw_{max} = 100$]

Sol^n,



The region codes for end points are:-

$P_1 = 1001$

$P_2 = 0100$

The logical AND operations of region codes for P_1 and P_2 gives:

$= 0000$

We determine that the line needs clipping.

Now,

Choose P_1 with region code (1001)

As P_1 lies on left side of window, we should calculate intersection of line with left boundary as:

- $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5 - 120}{130 - 0} = -0.8846$

$\therefore y = y_1 + m(x - x_1)$

$= 120 + (-0.8846) \times (10 - 0)$ [$\because$ For point in left boundary, $x = 10$]

$= 111.154$

≈ 111

So, $P_1' = (10, 111)$

The region code for P_1' is (1000)

Again,

The logical AND operations of region codes for P_1' and P_2 gives:

$= 0000$

So, Choose P_1' with region code (1000)

As P_1' lies on above side of window, we should calculate intersection of line with upper boundary as:

$m = -0.8846$

$\therefore y = y_1 + m(x - x_1)$

or, $100 = 111 + (-0.8846)(x - 10)$

or, $0.8846(x - 10) = 11$

or, $x - 10 = 12.435$

or, $x = 22.435$

$\therefore x \approx 22$ [$\because$ For upper boundary, $y = 100$]

So, $P_1'' = (22, 100)$

The region code for P_1'' is (0000).

Again,

The logical AND operations for region code for P_1'' and P_2 is:

$= 0000$

Choose P_2 with region code (0100)

Calculate intersection with bottom boundary (i.e $y = 10$) as:

$m = -0.8846$

$\therefore x = x_1 + \frac{y - y_1}{m}$

$= 130 + \frac{10 - 5}{(-0.8846)}$

$= 129.35$

≈ 124

So, $P_2' = (124, 10)$

The region code for P_2' is (0000).

Since, the region codes for both end points are 0000, they are accepted.

Hence, the end points after clipping are:

$P_1'' = (22, 100)$

$P_2' = (124, 10)$

Ans.

2) Perform Liang Barskey line clipping:

Clipping window: $(x_{min}, y_{min}) = (0, 0)$ $(x_{max}, y_{max}) = (100, 50)$

Line: $(x_1, y_1) = (10, 10)$ $(x_2, y_2) = (110, 40)$

Sol^n,

Initialize: $u_1 = 0$

$u_2 = 1$

Now,

Construction of table,

k	p_k	q_k	q_k/p_k	Remark
1	$-\Delta x$ $= -(110-10)$ $= -100 < 0$	$x_1 - x_{min}$ $= 10 - 0$ $= 10$	$= -1/10$	$u_1 = -1/10$
2	Δx $= 100 > 0$	$x_{max} - x_1$ $= 100 - 10$ $= 90$	$= 9/10$	$u_2 = 9/10$
3	$-\Delta y$ $= -(40-10)$ $= -30 < 0$	$y_1 - y_{min}$ $= 10 - 0$ $= 10$	$= -1/3$	$u_1 = -1/3$
4	Δy $= 30 > 0$	$y_{max} - y_1$ $= 50 - 10$ $= 40$	$= 4/3$	$u_2 = 4/3$

Here,

$u_1 = 0$ or $-1/10$ or $-1/3$

The maximum value is 0. for $u_1 \geq 0$.

$\therefore u_1 = 0$ —— (1)

$u_2 = 1$ or $9/10$ or $4/3$

The minimum value is $9/10$ for $u_2 \leq 1$.

$\therefore u_2 = 9/10$ —— (2)

The clipped lines can be calculated as:

$x_1' = x_1 + u_1 \Delta x = 10 + 0 \times 100 = 10$

$y_1' = y_1 + u_1 \Delta y = 10 + 0 \times 30 = 10$

$x_2' = x_1 + u_2 \Delta x = 10 + 9/10 \times 100 = 100$

$y_2' = y_1 + u_2 \Delta y = 10 + 9/10 \times 30 = 37$

Hence, the clipped line is (10, 10) and (100, 37).

Ans.

Chapter - 4

3-D Transformations

4.1 3-D Transformation

→ Translation

→ Scaling

→ Reflection

→ Rotation

→ Shear

4.2 3-D composite transformation

4.3 3-D Viewing

→ Viewing pipeline

→ World to screen viewing transformation

→ Projection Concepts

* Orthographic

* Parallel

* Perspective

[10 - marks]

Questions

1) Write rotation matrix in clockwise direction w.r. to x-axis, y-axis and z-axis. [3]

2) Describe 3D window to view port transformation with matrix representation for each step. [4]

3) Derive oblique projection matrix with necessary assumptions. [5]

4) How can you perform 3-D rotation of an object about some arbitary axis? [8]

5) Why 3D graphics is more complex than 2D? Explain with the help of viewing pipeline. [

6) What is projection? Explain parallel projection and derive the conditions for the oblique projections. [16]

4.1.1 Translation

A point $P(x,y,z)$ is translated to position $P'(x',y',z')$ under translation vector $T(t_x, t_y, t_z)$ such that:

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

4.1.2 Scaling

A point $P(x,y,z)$ is scaled by $S(S_x, S_y, S_z)$ to map to position $P'(x',y',z')$ such that:

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

→ Scaling about a fixed point

1) Translate object so that it coincides to origin.
2) Scale object about origin.
3) Inverse translation to restore original position.

So,

$$P' = [\{T(x_f, y_f, z_f) \cdot S(S_x, S_y, S_z)\} \cdot T(-x_f, -y_f, -z_f)] \cdot P$$

$$= \begin{bmatrix} S_x & 0 & 0 & (1-S_x)\,x_f \\ 0 & S_y & 0 & (1-S_y)\,y_f \\ 0 & 0 & S_z & (1-S_z)\,z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

4.1.3 Rotation

→ About z-axis (XY plane)

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

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

$$z' = z$$

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

→ About x-axis (YZ plane)

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

→ About y-axis (ZX plane)

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

→ Rotation about an axis parallel to one of coordinate axis

1) Translate rotation axis so it coincides to one of coordinate axis
2) Rotate about coordinate axis.
3) Apply inverse translation.

→ Rotation about arbitary axis

1) Translate so that P_1 coincide to origin.

2) Rotate line so it coincide to z-axis.

3) Rotate about z-axis

4) Apply inverse rotation to restore original orientation.

5) Apply inverse translation to restore original orientation.

Let us choose z-axis to map the rotation axis onto. The rotation axis is defined by 2 points: $P1(x_1, y_1, z_1)$ and $P2(x_2, y_2, z_2)$; which defines a vector

$$V = (x_2 - x_1, y_2 - y_1, z_2 - z_1) = (dx, dy, dz)$$

with unit vectors,

$$u = \frac{V}{|V|} = \frac{(dx, dy, dz)}{\sqrt{dx^2 + dy^2 + dz^2}} = \left(\frac{dx}{\sqrt{dx^2+dy^2+dz^2}}, \frac{dy}{\sqrt{dx^2+dy^2+dz^2}}, \frac{dz}{\sqrt{dx^2+dy^2+dz^2}} \right) = (a, b, c)$$

The translation matrix is:

$$T = \begin{bmatrix} 1 & 0 & 0 & x_1 \\ 0 & 1 & 0 & y_1 \\ 0 & 0 & 1 & z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation of rotation axis about x-axis is:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation of rotation axis about y-axis is:

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about z-axis is:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Also,

$$\cos\alpha = \frac{u' \cdot u_z}{|u'|\,|u_z|} = \frac{c}{d}$$

Projection of u in yz plane as vector $u' = (0, b, c)$

where, $d = \sqrt{b^2 + c^2}$

Also,

$$\sin\alpha = \frac{b}{d}$$

Similarly,

$$\cos\beta = d$$

$$\sin\beta = -a$$

The required rotation about arbitary axis represented by P_1 and P_2 is :

$$R(\theta) = T^{-1} R_x^{-1}(\alpha) R_y^{-1}(\beta) R_z(\theta) R_y(\beta) R_x(\alpha) T \quad \text{——— (1)}$$

$\therefore R(\theta) =$ [

4.1.4 Reflection

→ About z-axis (xy plane)

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

4.1.5 Shear

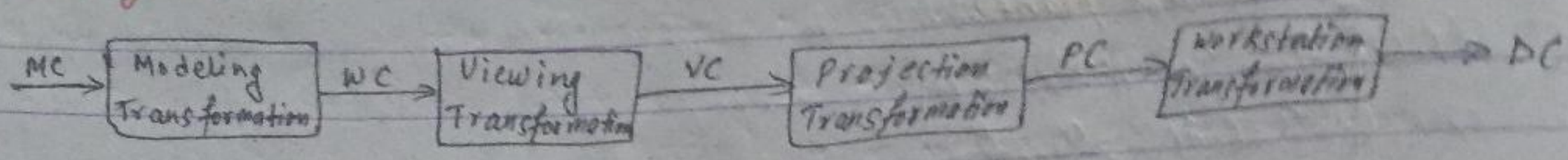
→ Z-axis shear

$$\begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.2 Composite Transformation

Similar to 2D

04.3 Viewing Pipeline



4.3.1 Transformation from world to viewing coordinates

→ Converting object descriptions from world to viewing coordinates is equivalent to transformation that superimposes viewing reference frame onto world frame using translate-rotate operation

→ Steps

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

→ Let view reference point is specified at world position (x_0, y_0, z_0)

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

→ Rotate around world x_w axis to bring z_v into $x_w z_w$ plane.

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

→ Rotate around world y_w axis to align z_w and z_v axes.

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

→ Rotate around world Z_w axis to align y_w and y_v axes.

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

∴ The complete transformation matrix is:

$$M_{wc,vc} = R_z \cdot R_y \cdot R_x \cdot T$$

4.4 Projection

→ Projection is defined as the process of projecting viewing coordinates of 3D objects onto two D view plane

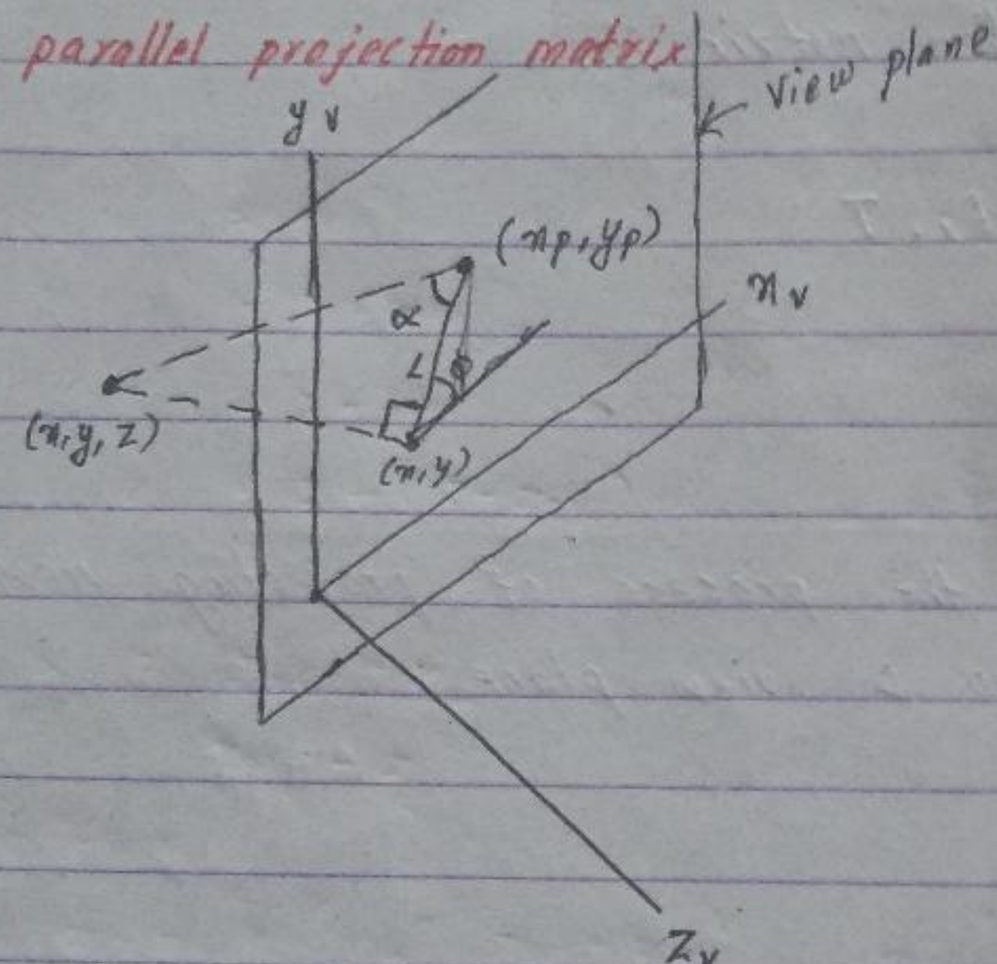
→ Parallel projection

→ Perspective projection

4.4.1 Parallel Projection

→ It preserves the relative proportions of objects.
→ Coordinate positions are transformed to the view plane along parallel lines.
→ When projection is ⊥ to view plane, it is orthographic projection.
→ When projection is not ⊥ to view plane, it is oblique projection.

4.4.2 Oblique parallel projection matrix



Consider point (x, y, z) is projected to (x_p, y_p) on the view plane. Let (x, y) be the orthographic projection coordinates on the plane. The oblique projection line from (x, y, z) to (x_p, y_p) makes an angle α with line joining (x_p, y_p) and (x, y). Let, this line of length L is at angle ϕ with horizontal direction in projection plane.

So, we can write;

$$\left.\begin{aligned} x_p &= x + L\cos\phi \\ y_p &= y + L\sin\phi \end{aligned}\right\} \quad \text{—— (1)}$$

Length L depends on angle α and z-coordinate of point to be projected

$$\therefore \tan\alpha = \frac{z}{L}$$

$$\Rightarrow L = z/\tan\alpha = z L_1 \quad \text{where } L_1 = 1/\tan\alpha.$$

So,
$$\left.\begin{aligned} x_p &= x + z(L_1\cos\phi) \\ y_p &= y + z(L_1\sin\phi) \end{aligned}\right\} \text{---} (2)$$

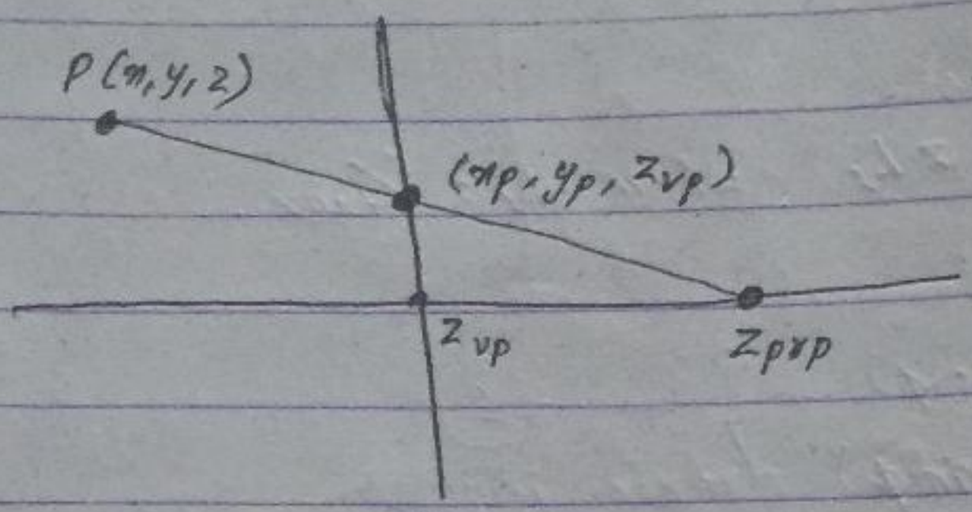
Hence, we can derive transformation matrix as:

$$M_p = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\Rightarrow \tan\alpha = 1$, cavalier projection

$\Rightarrow \tan\alpha = 2$, cabinet projection

4.4.3 Perspective Projection



In parametric form,

$$x' = x - xu$$

$$y' = y - yu$$

$$z' = z - (z - z_{prp})u \quad , \text{where } 0 \le u \le 1$$

Now,

$$u = \frac{z_{vp} - z}{z_{prp} - z}$$

On substitution,

$$x_p = x\left(1 - \frac{z_{vp} - z}{z_{prp} - z}\right) = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) = x\left(\frac{d_p}{z_{prp} - z}\right)$$

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

$\therefore$ $M_{perspective}$ =

Chapter - 5

Curve Modelling

5.1 Parametric Cubic Curve

5.2 Splines

5.3 Bezier Curves

[8 - marks]

Questions

1) Write properties of Bezier curve. Find eqⁿ of Bezier curve whose control points are P0(2,6), P1(6,8) and P2(9,12). Also find coordinate of point at u=0.3. [10]

2) List properties of Bezier curve. What is order of continuity? Explain. [8]

3) Derive eqⁿ for cubic Bezier curve. Also write its properties. [8]

4) Define Hermite interpolation in defining a curve. Use it to find blending function of parametric cubic curve in 2D graphics. [2+6]

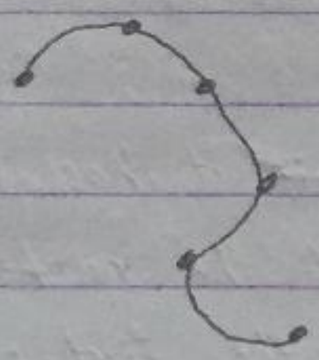
5.1 Parametric Cubic Curves

5.1.1 Interpolated and Approximation Splines

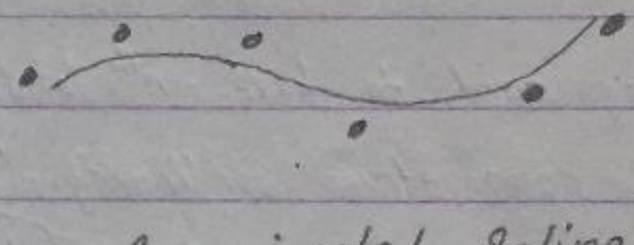
A spline curve is specified by a set of coordinates (i.e. control points) which indicates shape of curve. These points are then fitted with piecewise continuous parametric polynomial functions

A curve is called interpolated spline if polynomial sections are fitted so that it passes through each control points. It is used to digitize drawings or to specify animation paths.

A curve is called approximated spline if polynomial sections are fitted to general control-point path without necessary passing through any control point. It is used as design tools to structure object surfaces.



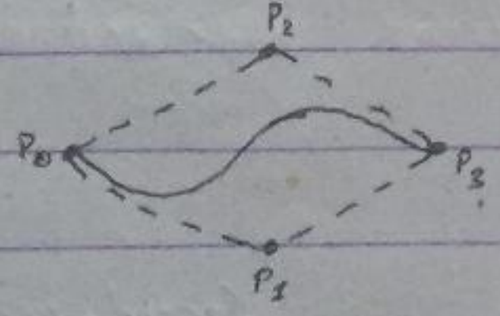
Interpolated spline



Approximated Spline

5.1.2 Convex Hull

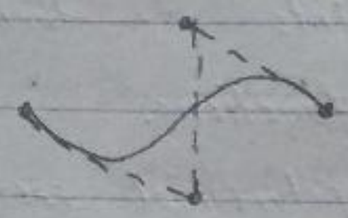
The convex polygon boundary that encloses a set of control points is called convex hull. It provides a measure for deviation of a curve from region bounding control points.



Convex hull shape

5.1.3 Control Graph

The set of connected line segments connecting sequence of control points is called control graph of a curve.



Control graph shape

5.1.4 Parametric Continuity Conditions

→ Parametric continuity is a concept applied to parametric curves describing the smoothness of the parameter's value with distance along the curve.

→ A curve can be said to have C^n continuity if $\frac{d^n s}{dt^n}$ is continuous throughout the curve.

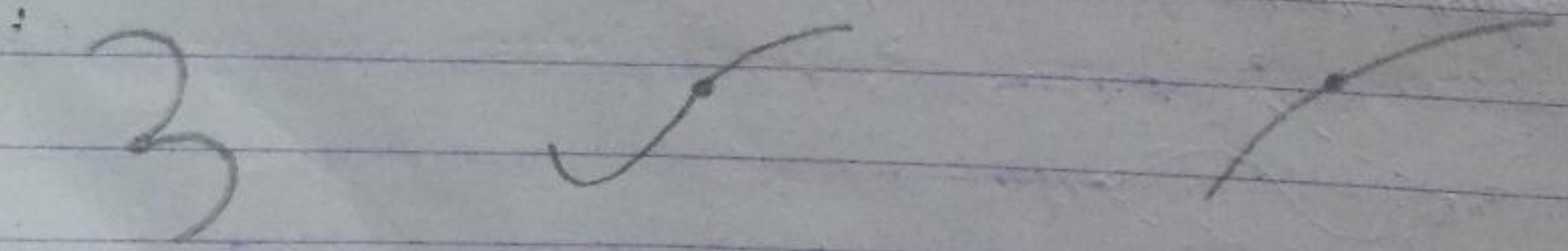
→ A curve is said to have n^{th} order of continuity if it has C^n continuity.

→ Zero order parametric continuity (C^0) means that the curves are joined.

→ First order parametric continuity (C^1) means that the first parametric derivatives of coordinate function for two successive curve sections are equal at their connection point.

→ Second order parametric continuity (C^2) means that both first and second parametric derivatives of two curve sections are same at their connection point.

Eg:



Zero-order First-order Second-order

5.1.5 Geometric Continuity Conditions

→ In case of geometric condinuity, only the parametric derivatives of two sections of curves should be proportional to each other at their common boundary.

→ First order geometric continuity (G^1) means the parametric first derivatives are proportional at the intersection of two successive sections. If $P(u)$ be parametric position on curve, the direction of $P'(u)$, but not necessarily its magnitude, will be same for two successive curve sections.

→ Second order geometric continuity (G^2) means the parametric first and second derivatives are proportional at their boundary.

5.1.6 Spline Specifications

1) We can state set of boundary conditions imposed on the spline.
2) We can state matrix that characterizes the spline.
3) We can state set of blending functions that determine how specified geometric constraints on curve are combined to calculate positions along curved path.

Eg: Consider parametric cubic polynomial representations:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x, \quad 0 \le u \le 1$$

1) Boundary conditions can be set i.e. $x(0)$ and $x(1)$, $x'(0)$ and $x'(1)$.
2) From boundary condition, we can obtain matrix:

$$x(u) = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix} = U \cdot C$$

i.e. Coefficient matrix (C) = $M_{spline} \cdot M_{geom}$

where, M_{geom} = 4 element column matrix containing boundary conditions on spline.
M_{spline} = 4×4 matrix that transforms geometric constraints to polynomial coefficient

3) $x(u) = U \cdot M_{spline} \cdot M_{geom}$

On expanding,

$$x(u) = \sum_{k=0}^{3} g_k \cdot BF_k$$

where, g_k = Constraint parameter (Control point coordinate)

BF_k = Polynomial Blending Function

5.1.7 Natural Cubic Splines

→ It is formulated by requiring that two adjacent curve have same first and second parametric derivatives at their common boundary i.e. C^2 continuity.

→ If we have (n+1) control points to fit, then we have n curve sections with a total of 4n polynomial coefficients to be determined.

→ The two curve sections on either sides of control point must have same first and second parametric derivatives at that point, and each curve must pass through that control point giving 4n-4 equations.

→ First control point (p_0) and last point (p_n) gives 2 equations.

→ The 2 equations can be obtained by setting second derivatives at p_0 and p_n to zero.

→ If position of any one control point is altered, the entire curve is affected.

3) $x(u) = U \cdot M_{spline} \cdot M_{geom}$

On expanding,

$$x(u) = \sum_{k=0}^{3} g_k \cdot BF_k$$

where, g_k = Constraint parameter (Control point coordinate)

BF_k = Polynomial Blending Function

5.1.7 Natural Cubic Splines

→ It is formulated by requiring that two adjacent curve have same first and second parametric derivatives at their common boundary i.e. C^2 continuity.

→ If we have (n+1) control points to fit, then we have n curve sections with a total of 4n polynomial coefficients to be determined.

→ The two curve sections on either sides of control point must have same first and second parametric derivatives at that point, and each curve must pass through that control point giving 4n-4 equations.

→ First control point (p_0) and last point (p_n) gives 2 equations.

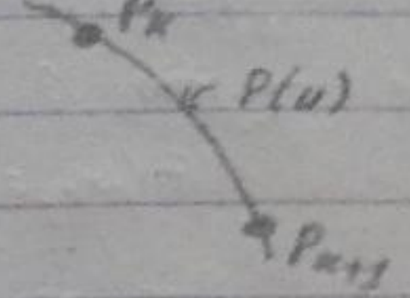
→ The 2 equations can be obtained by setting second derivatives at p_0 and p_n to zero.

→ If position of any one control point is altered, the entire curve is affected.

5.1.8 Hermite Interpolation

Hermite interpolation is the process of interpolating piecewise cubic polynomial with a specified tangent at each control point. Each curve section is only dependent on its endpoint constraints.

Consider a parametric cubic point function for curve section between control points p_k and p_{k+1} be $P(u)$ as shown in figure.



So, the boundary conditions defining Hermite curve sections are:-

$$\left.\begin{aligned} P(0) &= p_k \\ P(1) &= p_{k+1} \\ P'(0) &= Dp_k \\ P'(1) &= Dp_{k+1} \end{aligned}\right\} \quad \text{—— (1)}$$

where, Dp_k and Dp_{k+1} are parametric derivatives at p_k and p_{k+1} respectively.

The vector equivalent for Hermine curve section is:

$$P(u) = au^3 + bu^2 + cu + d \quad , \quad 0 \le u \le 1 \quad \text{—— (2)}$$

where, x component of P is $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$ and similarly for y and z-component.

The matrix equivalent is:

$$P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \quad \text{—— (3)}$$

The derivative of point function is:

$$P'(u) = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \quad \text{—— (4)}$$

Substituting endpoint values 0 and 1 for parameter u in eq^n (3) and (4), we get:

$$\begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \quad \text{—— (5)}$$

Solving for polynomial coefficients;

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} P_k \\ P_{k+1} \\ D_{P_k} \\ D_{P_{k+1}} \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_k \\ P_{k+1} \\ D_{P_k} \\ D_{P_{k+1}} \end{bmatrix} = M_H \cdot \begin{bmatrix} P_k \\ P_{k+1} \\ D_{P_k} \\ D_{P_{k+1}} \end{bmatrix} \quad \text{---(6)}$$

where, M_H is Hermite matrix.

So, equation 3 can be written as:

$$P(u) = [u^3 \; u^2 \; u \; 1] \cdot M_H \cdot \begin{bmatrix} P_k \\ P_{k+1} \\ D_{P_k} \\ D_{P_{k+1}} \end{bmatrix} \quad \text{---(7)}$$

The Hermite blending functions can be obtained as:

$$P(u) = P_k(2u^3 - 3u^2 + 1) + P_{k+1}(-2u^3 + 3u^2) + DP_k(u^3 - 2u^2 + u) + DP_{k+1}(u^3 - u^2)$$

$$= P_k H_0(u) + P_{k+1} H_1(u) + DP_k H_2(u) + DP_{k+1} H_3(u)$$

The polynomials $H_k(u)$ for $k = 0, 1, 2, 3$ are Hermite blending functions.

5.2 Bezier Curves

→ A Bezier curve can be fitted to any number of control points.

→ The number of control points to be approximated and their relative position determine degree of Bezier polynomial.

→ It is a polynomial of degree one less than the number of control points used.

5.2.1 Equation of Bezier curve

Consider n+1 control points i.e. $p_k = (x_k, y_k, z_k)$ with k varying from 0 to n. These points can be blended to produce following position vector P(u), which describes the path of an approximating Bezier polynomial function between p_0 and p_n.

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

where, $BEZ_{k,n}(u)$ is the Bezier blending functions given by:

$$\therefore BEZ_{k,n}(u) = C(n,k)\, u^k \,(1-u)^{n-k} \qquad \text{— (2)}$$

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

Also, Bezier blending functions can be defined with recursive calculation.

$$\therefore BEZ_{k,n}(u) = (1-u)\, BEZ_{k,n-1}(u) + u\, BEZ_{k-1,n-1}(u) \quad , \quad n > k \ge 1 \qquad \text{— (3)}$$

where, $BEZ_{k,k} = u^k$ and $BEZ_{0,k} = (1-u)^k$

The vector equations for eqn (1) are:

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

$$y(u) = \sum_{k=0}^{n} y_k \, BEZ_{k,n}(u) \qquad \text{—— (4)}$$

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

5.2.2 Properties

a) It passes through first and last control points such that boundary conditions at the two end points of the curve are:

$$p(0) = p_0 \quad \text{and} \quad p(1) = p_n$$

b) The values of parametric first derivatives of Bezier curve at the end points can be calculated from control point coordinates.

i.e. $P'(0) = -np_0 + np_1$

$P'(1) = -np_{n-1} + np_n$

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

c) It lies within the convex hull of the control points, ensuring that the polynomial follows the control points smoothly.

d) The parametric second derivatives of a Bezior curve at end points can be calculated as:

$$P''(0) = n(n-1)[(p_2 - p_1) - (p_1 - p_0)]$$

$$P''(1) = n(n-1)[(p_{n-2} - p_{n-1}) - (p_{n-1} - p_n)]$$

B) Find equation of Bezier curve with control point P0(2,6), P2(6,8) and P2(9,12). Also find coordinate of point at $u = 0.8$.

Sol^n,

Here, number of control points $(n) = 3$ varying from 0 to 2.

We know that,

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

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

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

$BEZ_{2,2}(u) = u^2$

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

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

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

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

At the end points of Bezier curve, the parametric first derivatives are:-

$$P'(0) = 3(P_1 - P_0)$$

$$P'(1) = 3(P_2 - P_0)$$

So,

$$(x'(0), y'(0)) = (12, 6)$$

$$(x'(1), y'(1)) = (9, 12)$$

The equation of Bezier curve is:

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

$$\therefore P(u) = P_0(1-u)^2 + P_1\{2u(1-u)\} + P_2 u^2 \quad \text{—— (1)}$$

The vector equations are:-

$$\left.\begin{aligned} \therefore x(u) &= x_0(1-u)^2 + x_1\{2u(1-u)\} + x_2 u^2 = 2(1-u)^2 + 12u(1-u) + 9u^2 \\ \therefore y(u) &= y_0(1-u)^2 + y_1\{2u(1-u)\} + y_2 u^2 = 6(1-u)^2 + 16u(1-u) + 12u^2 \end{aligned}\right\} \text{(2)}$$

Now, at $u = 0.8$;

$$x(u) = 2(1-0.8)^2 + 12 \times 0.8 \times (1-0.8) + 9 \times 0.8^2 = {}^{194}/_{25}$$

$$y(u) = 6(1-0.8)^2 + 16 \times 0.8 \times (1-0.8) + 12 \times 0.8^2 = {}^{262}/_{25}$$

$\therefore P(0.8) = ({}^{194}/_{25}, {}^{262}/_{25})$ Ans.

Chapter - 6

Surface Modelling

6.1 Polygon Surface

6.2 Vertex table, Edge table and Polygon table

6.3 Spatial Orientation of polygon surface

[8-marks]

Questions

1) Explain boundary representation technique to represent 3D object with example. How can you find spatial orientation of a surface? [8+2]

2) Explain significance of spatial orientation and polygon tables with example. [8]

3) Explain technique to calculate spatial orientation of individual surface component of 3D object. [4]

4) Describe polygon, vertex and edge table. How are they important in graphics? [8]

5) Explain how geometric & attribute information of a 3D object are stored for object representation? What are conditions for error free generation of polygon table. [5+3]

6.1 Boundary Representation

→ It describe a 3-D object as a set of surfaces that separate the object interior from the environment.

→ It is used in solid modeling.

→ It speeds up surface rendering and display of objects.

6.1.1 Polygon Surfaces

→ Commonly used boundary representation technique.

→ A 3-D object is a set of surface polygons that enclose object interior.

→ It describes surface features of the object.

→ Polygon mesh is a set of connected polygonally bounded planar surfaces.

6.1.2 Polygon Tables

→ Polygon surface is specified with a set of vertex coordinates and associated attribute parameters.

→ For each polygon, information is input whose data are placed into tables that can be used in subsequent processing, display and manipulation of objects in a scene.

→ Such table is called polygon table.

→ Polygon tables grouped into geometric tables (contain vertex coordinate to identify spatial orientation of polygon surface) and attribute tables (contain parameters specifying degree of transparency of object and its surface reflectivity).

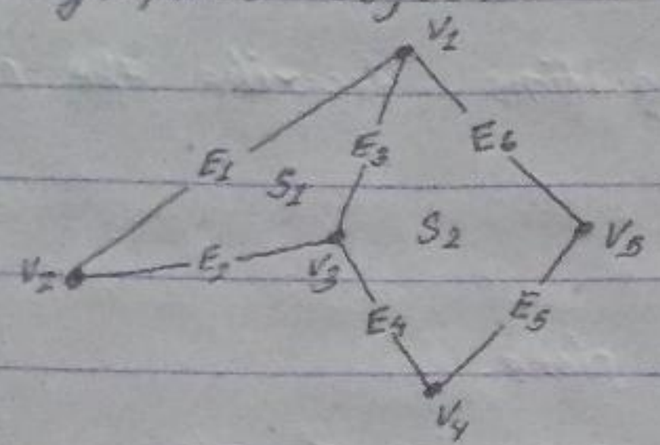
→ Geometric datas are stored in vertex table, edge table and polygon surface table.

→ Vertex table stores coordinate values for each vertex in the object.

→ Edge table stores pointers back into vertex table to identify the vertices of each polygon edge.

→ Polygon surface table stores pointers back into edge table to identify the edges for each polygon surface.

Eg: Consider the graphics object



The geometric data of this object can be stored as:

Vertex Table	Edge Table	Polygon Surface Table
$V_1 : x_1, y_1, z_1$	$E_1 : V_1, V_2$	$S_1 : E_1, E_2, E_3$
$V_2 : x_2, y_2, z_2$	$E_2 : V_2, V_3$	$S_2 : E_3, E_4, E_5, E_6$
$V_3 : x_3, y_3, z_3$	$E_3 : V_1, V_3$	
$V_4 : x_4, y_4, z_4$	$E_4 : V_3, V_4$	
$V_5 : x_5, y_5, z_5$	$E_5 : V_4, V_5$	
	$E_6 : V_1, V_5$	

6.1.3 Guidelines to generate error free tables

a) Every vertices listed must act as end point for at least two edges.

b) Every edge must be part of at least one polygon.

c) Every polygon is closed.

d) Each polygon has at least one shared edge.

e) If edge table contains pointer to polygons, every edge referenced by a polygon pointer has a reciprocal pointer back to the polygon.

6.2 Spatial Orientation of Polygon Surface

→ The information from vertex table and equations describe the polygon planes.

→ The equation for a plane surface is:

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

where, (x, y, z) is any point on a plane.

A, B, C, D describes spatial properties of plane.

6.2.1 Method of Determinants

Consider 3 successive polygon vertices be $V_1(x_1, y_1, z_1)$, $V_2(x_2, y_2, z_2)$, $V_3(x_3, y_3, z_3)$.

So, $Ax_1 + By_1 + Cz_1 + D = 0$

$Ax_2 + By_2 + Cz_2 + D = 0$

$Ax_3 + By_3 + Cz_3 + D = 0$

i.e. $\frac{A}{D} x_k + \frac{B}{D} y_k + \frac{C}{D} z_k = -1$ where, $k = 1, 2, 3$.

Applying Cramer's Rule,

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

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

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

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

On expanding determinants:

$$A = y_1 (z_2 - z_3) + y_2 (z_3 - z_1) + y_3 (z_1 - z_2)$$

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

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

$$D = -x_1 (y_2 z_3 - y_3 z_2) - x_2 (y_3 z_1 - z_3 y_1) - x_3 (y_1 z_2 - y_2 z_1)$$

→ Orientation of plane surface in space can be described with normal vector to the plane having cartesian components (A, B, C).

6.2.2 Using Normal Vector

Consider 3 vertices forming two vectors $\vec{V_1}$ to $\vec{V_2}$ and $\vec{V_1}$ to $\vec{V_3}$, the normal vector $\vec{N}$ can be calculated as:

$$\vec{N} = (\vec{V_2} - \vec{V_1}) \times (\vec{V_3} - \vec{V_1})$$

This generates values for plane parameters A, B and C.

We can obtain value of D by substituting these values and coordinate of one vertex.

The plane equation in vector form can be expressed as:

$$\vec{N} \cdot \vec{P} = -D$$

6.2.3 Significance

For any point (x, y, z) not on plane with parameters A, B, C, D;

$$Ax + By + Cz + D \neq 0$$

→ If $Ax + By + Cz + D < 0$, (x, y, z) is inside surface.

→ If $Ax + By + Cz + D > 0$, (x, y, z) is outside surface.

Chapter - 7

Visible Surface Determination

7.1 Image space and Object space technique

7.2 Back face Detection, Z-Buffer, A-Buffer, Scan-Line Method

[10 - marks]

Questions

1) Explain z-buffer algorithm along with necessary steps to calculate depth. What is its drawback [10]

2) Compare z-buffer and a-buffer algorithm. Also write algorithm for scan line method. [10]

3) Describe scan line method to find visible lines with example. [10]

4) What is difference between object space and image space method. [3]

5) Write short note on: Back face Detection Algorithm [5]

6) Describe A-Buffer method of visible surface detection. [6]

7.1 Image Space and Object Space Method

7.1.1 Object Space Method

→ Visible surface detection is done by dealing with object definitions directly.
→ Compares objects and parts of objects to each other within the scene definition to determine which surfaces should label as visible.
→ Eg: Line display algorithm, Back face detection method, etc.

7.1.2 Image Space Method

→ Deals with projected images of an object.
→ Visibility is decided point by point at each pixel position on the projection plane.
→ Eg: Z-Buffer method, Scan line method, etc.

7.2 Backface Detection / Plane Equation Method

Backface detection is an object space method for identifying back faces of a polyhedron which is based on inside-outside tests.

A point (x, y, z) is inside a polygon surface with plane parameters A, B, C and D if $Ax + By + Cz + D < 0$. When an inside point is along the line of sight to the surface, the polygon must be a back face.

Consider a normal vector $\vec{N}$ to a polygon surface having cartesian components (A, B, C). If $\vec{V}$ is in the viewing direction from the eye position, then this polygon is a back face if:

$$\vec{V} \cdot \vec{N} > 0 \quad \text{—— (1)}$$

If viewing direction is parallel to viewing z-axis, then $V = (0, 0, V_z)$ and

$$\vec{V} \cdot \vec{N} = V_z C \quad \text{—— (2)}$$

where, C is z-component of normal vector.

If viewing direction is along negative z-axis, polygon is a back face if $C < 0$.
Also, we can not see any face whose normal has $C \leq 0$.

In general, we can label any polygon as back face if its normal vector has z-component $C \leq 0$ when viewing direction is along negative z axis and $C \geq 0$ when viewing direction is along positive z axis.

7.2.1 Drawbacks

a) It does not deal with overlapping point faces due to multiple objects and concave objects

b) It does not deal with non-polygonal and non-closed objects

7.3 Z-Buffer / Depth-Buffer Method

Z-Buffer method is an image space method of visible surface detection which compares surface depths at each pixel position on the projection plane. It is called so because object depth is measured from the viewplane along Z-axis of a viewing system.

7.3.1 Depth Buffer and Refresh Buffer

Depth buffer is used to store depth values for each (x, y) position as surfaces are processed. Initially, it is set to zero.

Refresh buffer is used to store intensity values for each position. Initially, it is set to background intensity.

7.3.2 Algorithms

1) Initialize depth buffer and refresh buffer.

i.e for all positions (x, y) : depth $(x, y) = 0$, refresh $(x, y) = I_{background}$

2) For each position on each polygon surface, compare depth values to previously stored depth buffer to determine visibility.

a) Calculate depth z for each (x, y) position on polygon.

b) If $z >$ depth (x, y), then set:

depth $(x, y) = z$, refresh $(x, y) = I_{surf}(x, y)$

3) After processing of all surfaces, depth buffer contains depth values for visible surfaces and refresh buffer contains corresponding intensity for these surfaces.

7.3.3 Depth Computations

Depth value for a surface position (x,y) are calculated from plane equation for each surface as

$$z = \frac{-Ax - By - D}{C} \quad \text{---} (1)$$

For one position $(x+1, y)$ along $y-1$ scan line, depth is:

$$z' = \frac{-A(x+1) - By - D}{C} = z - \frac{A}{C} \quad \text{---} (2)$$

The ratio $-A/C$ is constant.

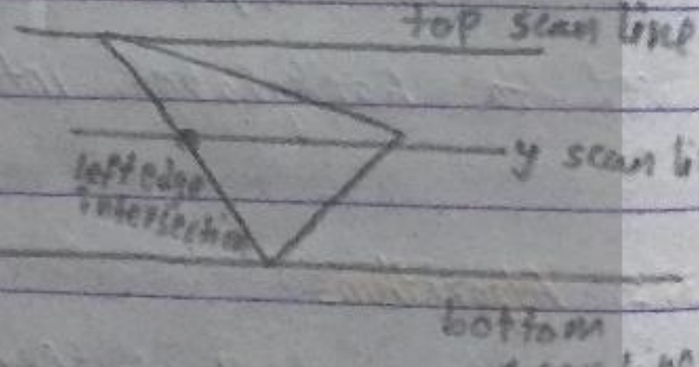
So, succeeding depth values across a scan line are obtained from preceeding values with addition of a single constant value.

Steps

1) First determine the y-coordinate extents of each polygon and process the surface from topmost scanline to the bottom scan line.
2) Starting at top vertex, we can recursively calculate x positions down a left edge of polygon as $x' = x - 1/m$, where m is slope of edge.
3) Depth values down the edge is given as:

$$z' = z + \frac{A/m + B}{C}$$

If edge is vertical, slope is infinite. i.e. $z' = z + B/C$.



7.3.4 Drawbacks

1) It can only find one visible surface at each pixel position.
2) It deals with only opaque surfaces and can not accumulate intensity values for more than 1 surface.

7.4 A-Buffer Method

A-Buffer method is the extension of Z-buffer method such that each position (x,y) in the buffer can reference a linked list of surfaces so that more than one surface intensity can be taken into consideration at each pixel position.

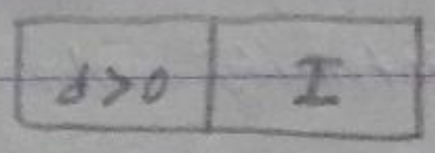
Each position in A-buffer has two fields:

a) depth field

b) intensity field

The surfaces can be dived as:

a) Single surface overlap of corresponding pixel area

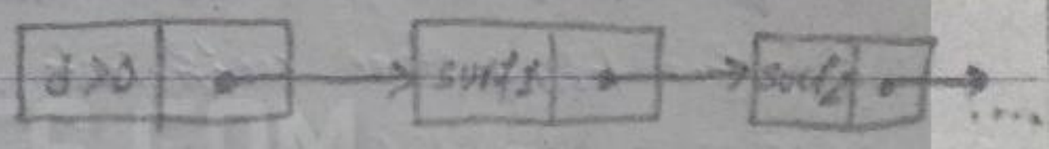


→ Here, $d > 0$

→ The number stored gives the depth

→ intensity field stores RGB components of the surface color at that point and the % of pixel coverage.

b) Multiple surface overlap

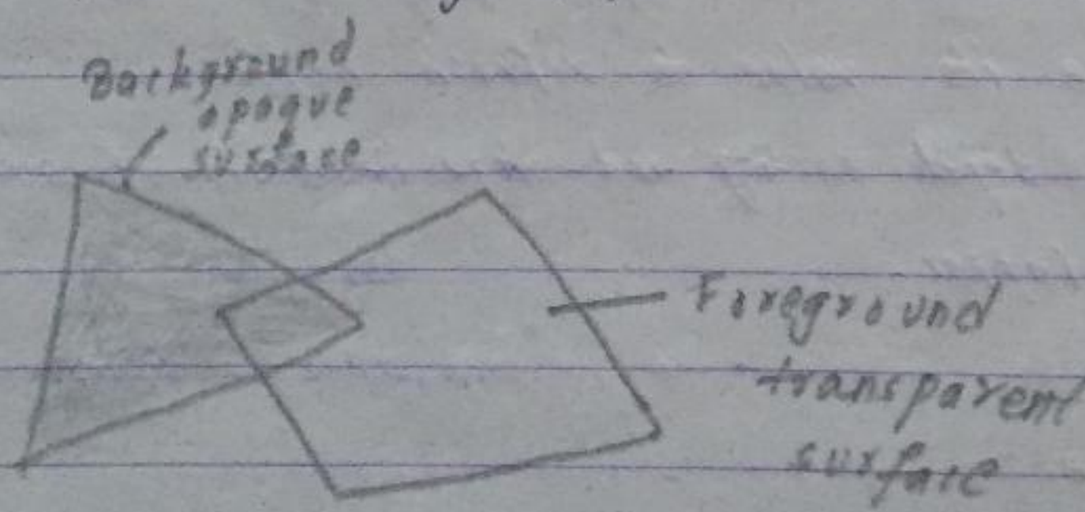


→ Here, $d < 0$

→ Indicates multiple surface contributing to the pixel intensity.

→ Intensity field stores a pointer to a linked list of surface data.

→ Each linked list includes RGB components, opacity parameter, depth, surface identifier, % of area coverage, pointer to next surface, etc.



7.5 Scan Line Method

We assume edge table and polygon table for each surfaces are set up. We can set up active list of edges from edge table such that it keeps track of edges intersected by the current scan line sorted in order of increasing x. Also, we define a flag for each surface that is turned on or off to indicate whether a position along a scan line is inside or outside of the surface. For each scan line, it is processed from left to right. At leftmost boundary, flag is turned on and at rightmost boundary, flag is turned off. For each scan line processing, all polygon surface intersecting that line are examined to determine which are visible. In case of overlapping surface, depth is calculated to determine which is nearest to view plane. When visible surface has been determined, intensity value for that position is entered into refresh buffer.

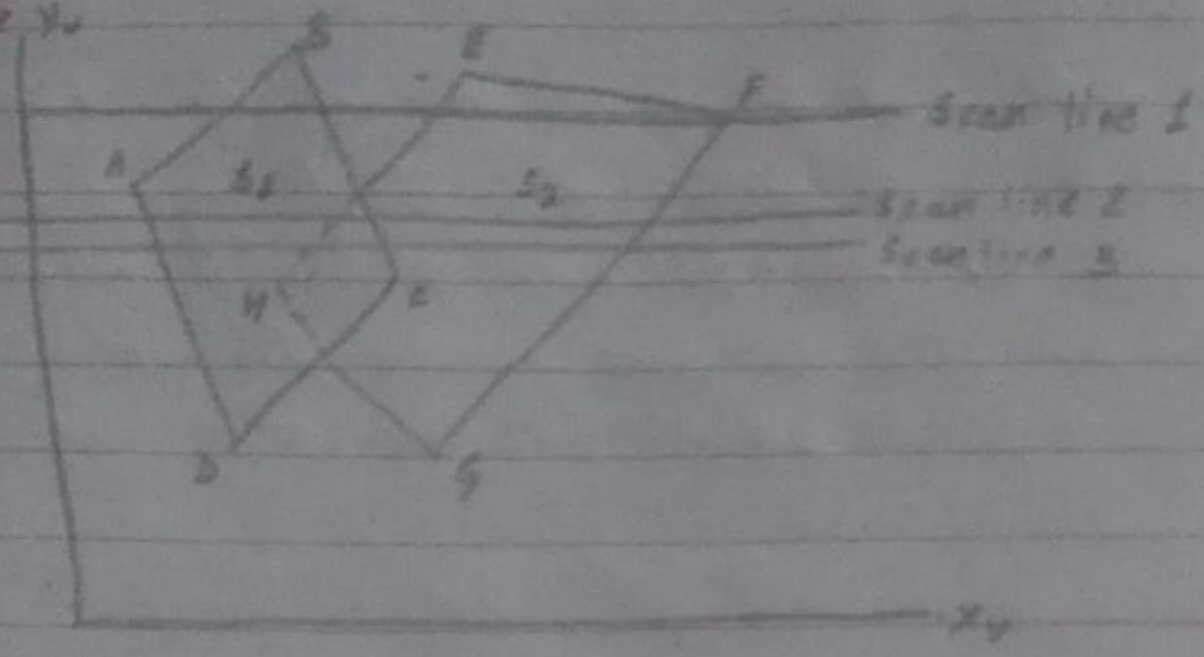
7.5.1 Algorithm

1) Initialize all necessary data structures.
   a) Edge table : end point coordinates, inverse slope and polygon pointer.
   b) Surface table : plane coeffficients and surface intensity.
   c) Active edge list
   d) Flag for each surface.
2) For each scan line, repeat:
   a) Update active edge list
   b) Determine intersection point and set surface flag on or off.
   c) If one flag is on, store its value to refresh buffer.
   d) If more flags are on, do depth sorting and store intensity of surface nearest to view plane in refresh buffer.

7.5.2 Drawbacks of using coherence

1) It do not work if surfaces cut through or cyclically overlap each other

7.5.3 Example



For scan line 1:

→ Active edge list = AB, BC, EH, FG

→ For positions along scan line 1 between AB and BC, only flag for S_1 is on.

→ For positions between EH and FG, only flag for S_2 is on.

→

For scan line 2 and 3:

→ Active edge list - AD, EH, BC, FG

→ Between AD and EH, flag for S_1 is only on.

→ Between EH and BC, flag for both S_1 and S_2 are on. So, depth is calculated. In this example, depth of S_1 is assumed to be less than of S_2, so intensities for S_1 are loaded to refresh buffer until BC is encountered.

→ Between BC and FG, flag for S_2 is only on.

Chapter - 8

Illumination and Surface Rendering Method

8.1 Algorithms to simulate ambient, diffuse and specular reflections

8.2 Constant, Gourand and phong shading models

[14 - marks]

Questions

1) Define: ambient light, lambert cosine law, diffuse reflection and specular reflection. Also, find equation for intensity of point by using Phong illumination model. [10]

2) Explain general illumination model. How it is used for rendering by using gouround shading? [7+7]

3) Under what conditions flat shading gives accurate rending? Mention disadvantage of intensity interpolation technique. Explain Phong shading with mathematical calculation. Explain diffuse reflection. [3+1+6+4]

4) Calculate total intensity using phong specular reflection model considering all light sources. [8]

5) Compare and contrast Gourand and Phong shading model. [8]

8.1 Illumination model / Lighting model / Sharing model

→ Illumination model is the model used to calculate the intensity of light that we should see at a given point at the surface of an object

→ Surface rendering algorithm uses the calculated intensity from illumination model to determine light intensity for projected pixel position for various surface in the scene.

8.2 Light Sources

→ When we see an opaque non-luminous object, we see reflected light from the surfaces of the object.

→ The total reflected light is the sum of contributions from light sources and other reflecting surfaces in the scene.

→ Point source is a model in which rays from the source radially diverged from source position. Eg: bulb, sun, etc.

→ Distributed source is a model in which the rays from the source are parallel. Eg: flourescent lamp.

8.3 Basic Illumination Model

8.3.1 Ambient Light

→ Ambient or background light is the light due to the multiple reflection of nearby light reflecting objects which is uniform

→ It is independent of the viewing direction and the spatial orientation of a surface

→ ~~It is constant~~

→ The amount of ambient light incident on each object is constant for all surfaces and over all directions.

→ The intensity of reflected light depends on optical properties of the surface

→ Ambient illumination is calculated as;

$$I = I_a k_a$$

where, I_a = ambient light intensity = constant
k_a = ambient reflection coefficient

8.3.2 Diffuse Reflection

The surfaces that are rough tend to scatter the reflected light in all directions so that the surface appears equally bright from all viewing directions. Such scattered light is called diffuse reflection. Diffuse reflections are constant over each surface in a scene independent of viewing direction. The incident light is scattered with equal intensity in all directions independent of angle of ~~incidence~~ reflection but depends on angle of incidence.

Suppose a surface is exposed only to ambient light, the intensity of diffuse reflection at any point on the surface is:

$$I_{amb,diff} = I_a k_d$$

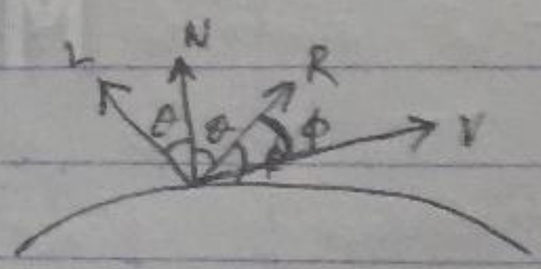
where, I_a = intensity of ambient light

k_d = diffuse reflection coefficient or diffuse reflectionvity $(0 \leq k_d \leq 1)$

→ If θ be angle of incidence, $I_{amb,diff} = I_a k_d \cos\theta$

8.3.3 Specular Reflection

The surfaces that are smooth or shiny results in total reflection of incident light in a concentrated region. Such phenomenon is called specular reflection.



Let $\vec{N}$ be unit normal surface vector, $\vec{R}$ be unit vector in direction of ideal specular reflection and $\vec{L}$ be unit vector directed toward point light source. Let $\vec{V}$ be unit vector pointing to viewer from surface position. Here, the specular reflection angle equals the angle of incident light (θ).

For ideal reflector, reflected light can be seen only when $\vec{V}$ and $\vec{R}$ coincides i.e ($\phi = 0$).

8.3.4 Phong Illumination Model

→ Sets the intensity of specular reflection proportional to $\cos^{n_s}\phi$

i.e. $I_{spec} \propto \cos^{n_s}\phi$ —— (1)

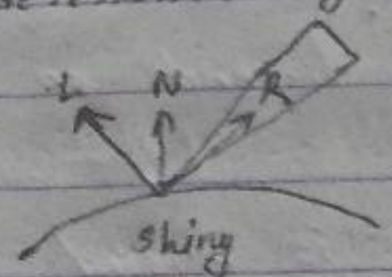
where, ϕ = ~~angle~~ viewing angle relative to specular reflection direction = 0° to 90°

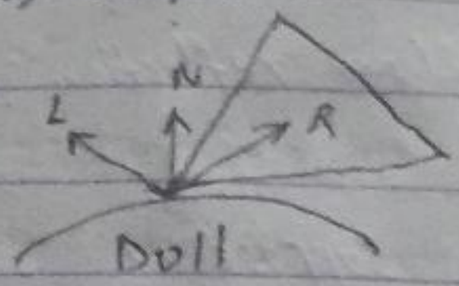
n_s is determined by type of surface.

→ $n_s = \infty$, for perfect reflector

→ For shiny surface, n_s is large

→ For dull surface, n_s is small



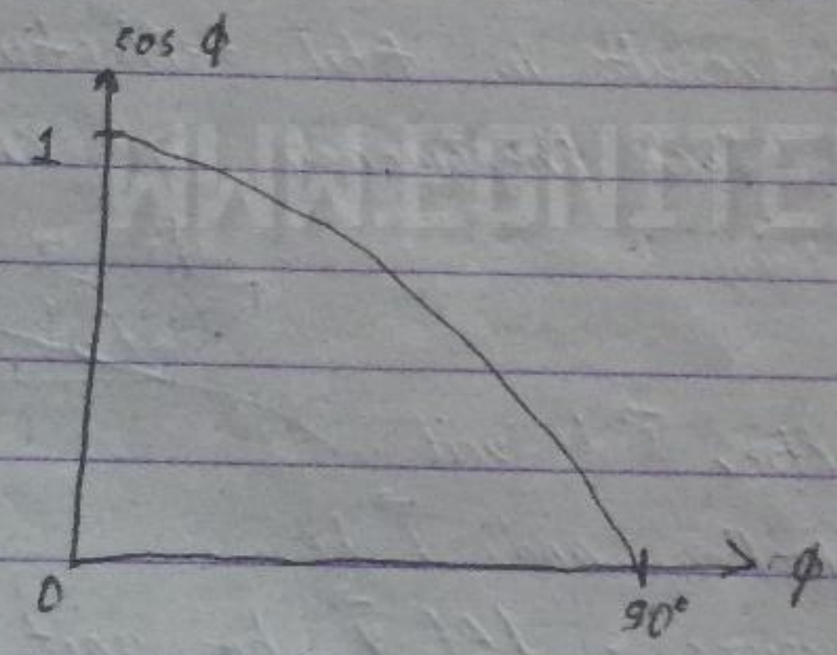


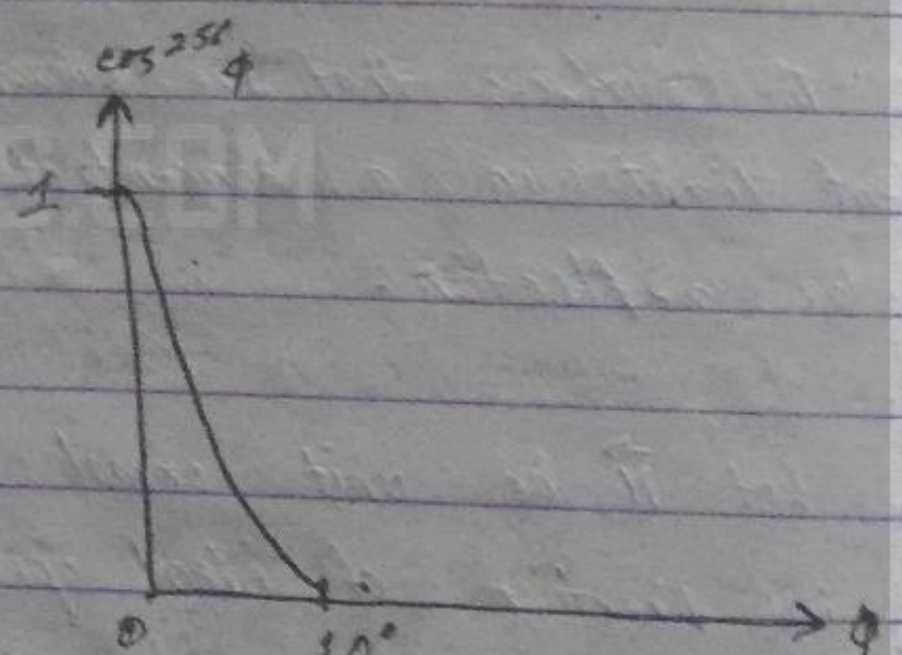
→ If $W(\theta)$ be specular reflection coefficient, the Phong model is:

$$I_{spec} = W(\theta)\, I_l \cos^n\theta \quad \text{—— (2)}$$

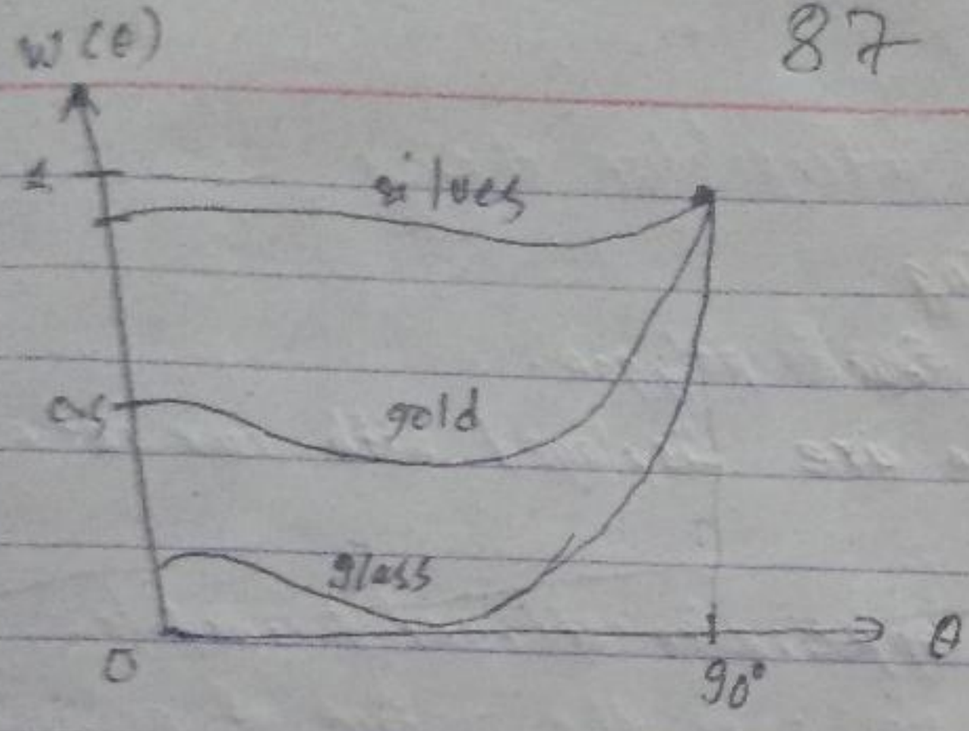
where, I_l = intensity of light source

→ $W(\theta)$ depends on angle of incidence. If angle of incidence increases, $W(\theta)$ tends to increase.





Plots of $\cos^{n_s}\phi$ for several values of n_s



→ For most of the opaque objects, specular reflection is nearly constant for all incidence angles.

i.e. $W(\theta) = k_s$; $0 \le k_s \le 1$.

→ Since $\vec{V}$ and $\vec{R}$ are unit vectors in ~~specular~~ viewing and specular reflection direction, we get:

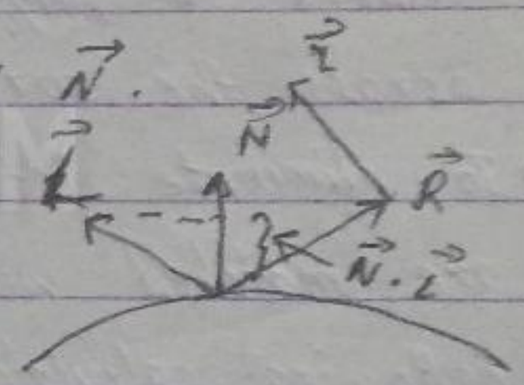
$$\cos\theta = \vec{V} \cdot \vec{R}$$

So, $I_{spec} = k_s I_l (\vec{V} \cdot \vec{R})^{n_s}$ ——— (3)

→ Vector $\vec{R}$ can be expressed in terms of $\vec{L}$ and $\vec{N}$.

From fig, $\vec{R} + \vec{L} = 2(\vec{N} \cdot \vec{L})\vec{N}$

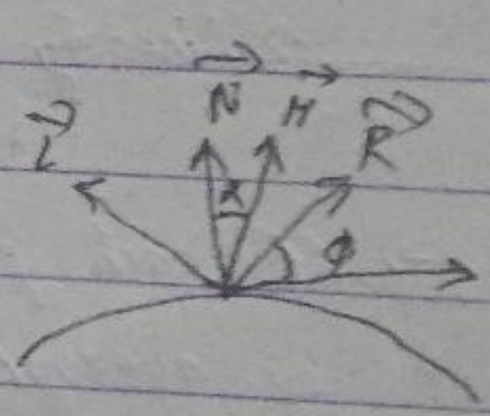
$\Rightarrow \vec{R} = 2(\vec{N} \cdot \vec{L})\vec{N} - \vec{L}$



→ Simplified model is obtained by using half way vector $\vec{H}$ between $\vec{L}$ and $\vec{V}$.

We can replace $\vec{V} \cdot \vec{R}$ with $\vec{N} \cdot \vec{H}$, it replaces $\cos\phi$ with $\cos\alpha$.

$\vec{H} = \dfrac{\vec{L} + \vec{V}}{|\vec{L} + \vec{V}|}$



→ If viewer and source are far from surface, $\vec{V}$ and $\vec{L}$ is constant over the surface

8.4 Rendering Methods

8.4.1 Constant intensity shading / Flat shading

→ A single intensity is calculated for each polygon.
→ All points over the surface of polygon are displayed with same intensity values.
→ Useful for quick display of general appearance of curved surface.

→ It provides accurate rendering if following assumptions are valid:-
a) The object is polyhedron and is not an approximation of an object with a ~~surface surf~~ curved surface.
b) All light sources illuminating the object are far from the surface so that $\vec{N} \cdot \vec{L}$ is constant over the surface.
c) The viewing position is far from the surface so that $\vec{V} \cdot \vec{R}$ is constant.

8.4.2 Gouraud shading / Intensity interpolation shading

→ Renders a polygon surface by linear interpolation of intensity values across the surface.
→ To eliminate intensity discontinuity, intensity values for each polygon are matched with values of adjacent polygons along common edges.
→ Following calculations are performed:
a) Determine average unit normal vector at each polygon vertex
b) Apply illumination model to each vertex to calculate vertex intensity.
c) Linearly interpolate vertex intensities over surface of the polygon.

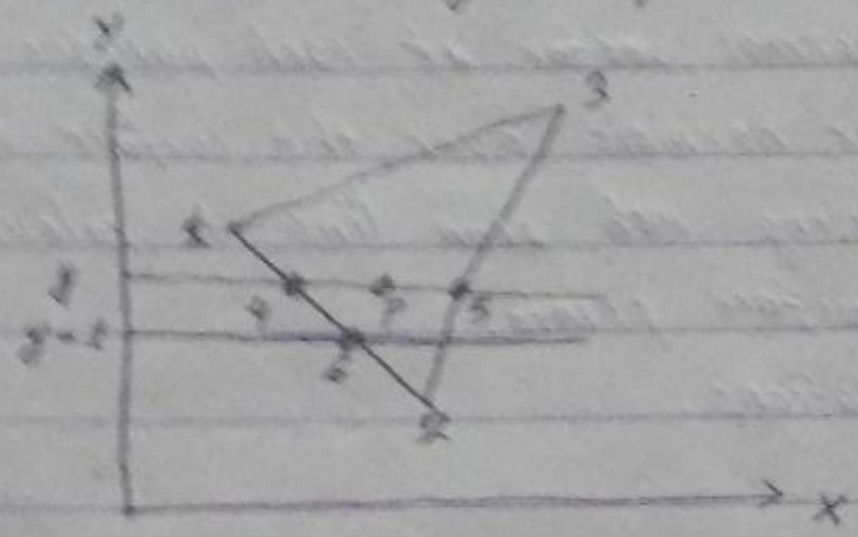
→ At each polygon vertex, average unit normal vector is obtained by averaging the surface normals of all polygons sharing that vertex.
For any vertex V,

$$N_V = \frac{\sum_{k=1}^{n} N_k}{\left| \sum_{k=1}^{n} N_k \right|} \quad \text{---------- (1)}$$

→ The intensity of each vertex V can be obtained from Phong illumination model as:

$$I_v = k_s I_s \cos^n \phi \quad \text{——} (2)$$

→ The intensities are linearly interpolated as shown in fig:



→ Let I_1, I_2 and I_3 be intensities at points 1, 2 and 3 respectively. So, intensity at point 4 (intersection of scan line and line 12) is obtained by linear interpolation from I_1 and I_2.

$$\therefore I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2 \quad \text{——} (3)$$

Similarly, for point 5:

$$\therefore I_5 = \frac{y_5 - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_5}{y_3 - y_2} I_2 \quad \text{——} (4)$$

Now, an interior point p is then assigned intensity value that is linearly interpolated from I_4 and I_5.

$$\therefore I_p = \frac{x_p - x_4}{x_5 - x_4} I_5 + \frac{x_5 - x_p}{x_5 - x_4} I_4 \quad \text{——} (5)$$

To find successive edge intensity, incremental calculations can be used.

$$\text{i.e. } I_6 = I_4 + \frac{I_2 - I_1}{y_2 - y_1} \quad \text{——} (6)$$

→ Highlights on surface are displayed with anamalous shapes.
→ Linear intensity interpolation can cause dark or bright intensity streaks (Mach bands) to appear.

8.4.3 Phong Shading Model / Normal vector interpolation shading

→ More accurate method

→ Displays realistic highlights on a surface

→ Reduces Mach-band effect

→ Following steps are performed.

a> Determine average unit normal vector at each polygon vertex.

b> Linearly interpolate vertex normals over the surface of polygon.

c> Apply illumination model along each scan line to calculate projected pixel intensities for surface points.

→ It requires more calculations.

Chapter - 9
OpenGL

9.1 Introduction to OpenGL

→ OpenGL is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics.
→ It is designed to be implemented mostly or entirely in hardware.
→ It is possible for the API to be implemented entirely in software.
→ The API is defined as a set of functions which may be called by the client program, alongside a set of named integer constants.
→ Such function definitions are language independent.
→ It is concerned with rendering.
→ It does not provide APIs related to i/p, audio or windowing.

9.2 Callback functions

→ Callback function is a function which the library i.e. GLUT calls when it needs to know how to process something.
→ Eg: When GLUT gets a key down event it uses glutKeyboardFunc callback routine to find out what to do with a key press.
→ It includes mouse callback, keyboard callback, reshape callback.

→ Eg: Let us consider that; when key 'q' is pressed, we need to exit. For this operation, the function can be defined as:

```
void mykey()
{ if (key == 'Q' | key == 'q')
      exit(0);
}
```

The callback function can be used as:

```
glutKeyboardFunc (mykey)
void mykey(unsigned char key, int x, int y)
```

// returns ASCII code of key depressed and mouse location.

3.3 GLUT (Graphics Library Utility Toolkit)

→ It is OpenGL Utility Toolkit
→ Window system independent toolkit for writing OpenGL programs.
→ Implements simple windowing API for OpenGL.
→ It makes you to write a single OpenGL program that works across all PC and workstation OS plotform.
→ Simple, easy and small.

→ GLUT supports :-
a> Multiple windows for rendering
b> Callback driven event processing
c> Sophisticated i/p devices
d> idle routine & timers
e> pop-up menu facility
f> Routine to generate solid & wire frame objects
g> bitmap and stroke fonts

9.4 Drawing pixel, line and polygons

→ To draw stand alone vertex defined shape

→ use glBegin() function with argument: GL_TRIANGLES, GL_QUADS, GL_POLYGON

→ To define vertices,

glVertex2f(x, y);

→ The vertices should be defined in counter-clockwise order. If so, the front face will be facing towards you.

→ To end shap.

glEnd()

→ Eg: To draw parallelogram with vertices (0.0, 0.0), (1.0, 0.0), (1.5, 1.2) and (0.5, 1.2)

```
glColor3f(0.0f, 0.0f, 0.0f); // Sets color to black
glBegin(GL_QUADS);
    glVertex2f(0.0f, 0.0f);
    glVertex2f(1.0f, 0.0f);
    glVertex2f(1.5f, 1.2f);
    glVertex2f(0.5f, 1.2f);
glEnd();
```

→ Lines

GL_LINES

→ Pixel

GL_POINTS