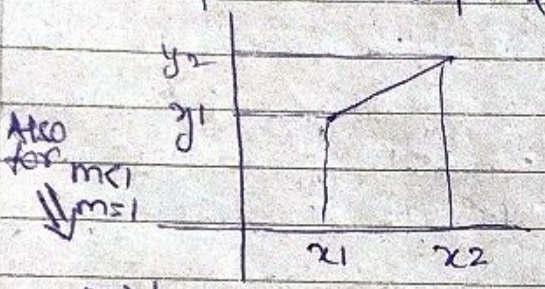
UNIT-2 Graphics Primitives

Points - Point plotting is done by converting a single coordinate pos^n furnished by an application program into appropriate operations for the output device in use.
Line drawing is done by calculating intermediate positions along the line path b/w two specified endpoint position.
→ For raster video display, the line intensity is loaded in frame buffer at corresponding pixel position.

LINE DRAWING ALGORITHMS:-

→ Cartesian slope intercept eq for straight line is $y = mx + c$
m represents slope & c intercept at y



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

$$c = y1 - m * x1$$

$\Delta y = m * \Delta x$ & $\Delta x = \Delta y / m$

m > 1, Δy can be proportional to small vertical deflection voltage & horizontal is set proportional to Δx which is calculated from above eq^n

DDA Algorithm

Digital differential Analyzer is scan conversion line drawing algorithm based on calculating either Δy or Δx using above eq^n.

• Ex:- m<1

(5,4) (12,7)

$\Delta x = 12 - 5 = 7$

$\Delta y = 7 - 4 = 3$

Steps 7

$x_{inc} = \frac{7}{7} = 1$

$y_{inc} = \frac{3}{7} = 0.4$

x	y	
5	4	4
6	4.4	4
7	4.8	5
8	5.2	5
9	5.6	6
10	6	6
11	6.4	6
12	6.8	7

Not smooth line

Procedure for DDA :-

```
Void lineDDA (int xa, int ya, int xb, int yb)
{ int dx = xb-xa, dy = yb-ya, steps, k;
  float xincrement, yincrement, x=xa, y=ya;
  if (abs (dx) > abs(dy))
     { Steps = abs (dx); }
  else { steps = abs (dy); }
  x increment = dx /(float) steps;
  yincrement = dy /(float)steps;
  set pixel (ROUND(x), ROUND(y));
  for (k=0; k<steps; k++)
     { x += xincrement;
       y += yincrement;
       set pixel (ROUND(x), ROUND(y));
     }
}
```

Advantage :-

→ It is faster Algorithm
→ It is simple Algorith

Disadvantage :-

→ floating point arithmetic is time consuming.
→ Poor end point accuracy.

BRESENHAM'S LINE ALGORITHM :-

An accurate & efficient Raster line-generating algorithm, developed by bresenham which scan convert line using only incremental integer calculation that can be modified to display circles & other curves.

Slope < 1

Derived

$y = m(x_k + 1) + c$

If $P_k < 0$ $P_{k+1} = P_k + 2\Delta y$

if $P_k >= 0$ $P_{k+1} = P_k + 2\Delta y - 2\Delta x$

$P_1 = 2\Delta y - \Delta x$

```
Algorithm Bresenham (x1,y1, x2,y2)
{ x = x1;
  y = y1;
 dx = x2-x1; dy = y2-y1;
 p = 2dx - dy
 while (x<=x2)
  { putpixel (x,y);
    x++;
    if (p<0)
     { p = p + 2dy
     }
    else
    { p = p+2dy - 2dx;
      y++;
    }
}
}
```

dx = 7

dy = 4

P = 2x4 - 7 = 1

2dy = 2x4 = 8

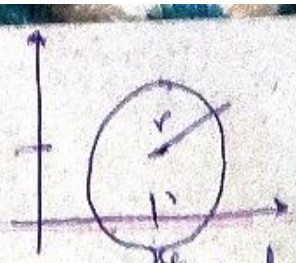
2x dy - 2x dx

2x4 - 2x7 = -6

EXAMPLE:- Point (1,1) to (8,5)

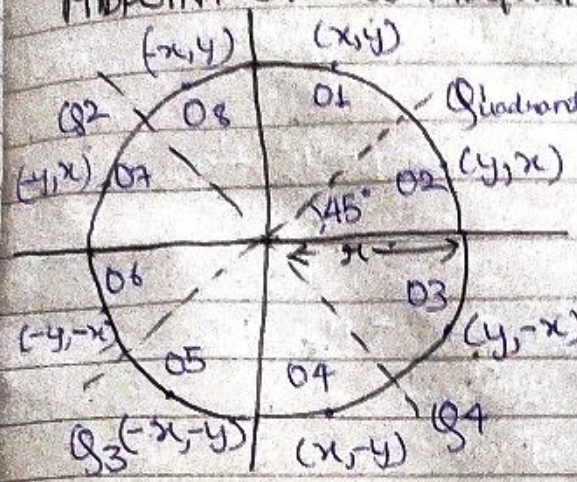
x	y	P
1	1	1
2	2	-5
3	2	3
4	3	-3
5	3	5
6	4	-1
7	4	7
8	5	

CIRCLE :-



A circle is defined as the set of points that are all at a given distance r from a center position say (x_c, y_c)

MIDPOINT CIRCLE ALGORITHM :-



8 Octant - 8 way symmetry

Circle Equation

$x^2 + y^2 = r^2$ ($C : (0,0)$)

Brute force

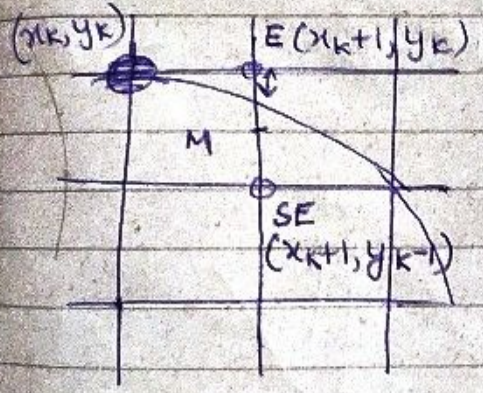
$x \to 0 - x$

$\forall x, y$?

$y = \sqrt{r^2 - x^2}$

⊗ time consuming

to overcome that Midpoint Algo is used



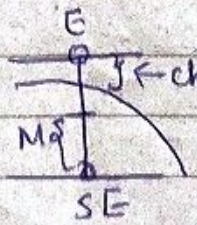
$x^2 + y^2 = r^2$ {(0,0)}

$x^2 + y^2 - r^2 = 0$

$< r^2$ < 0 inside circle

$> r^2$ > 0 Outside

$= r^2$ On Circle



If Midpoint lie inside circle then take E Point ~~other~~

If Outside then take SE

(x_k+1, y_k) (x_k+1, y_k-1) (x_M, y_M)

Mid Point $\left(\frac{x_k+1 + x_k+1}{2}, \frac{y_k + y_k-1}{2}\right) = \left(x_k+1, y_k - \frac{1}{2}\right)$

$$P_K = x_M^2 + y_M^2 - r^2 = (x_K+1)^2 + (y_K - \tfrac{1}{2})^2 - r^2$$

$$P_{K+1} = (x_{K+1}+1)^2 + (y_{K+1} - \tfrac{1}{2})^2 - r^2$$

$$P_{K+1} - P_K = (x_K)^2 + 4 + 4x_K - x_K^2 - 1 - 2x_K + y_{K+1}^2 + \tfrac{1}{4} - y_{K+1} - y_K^2 + y_K - \tfrac{1}{4}$$

$$P_{K+1} = P_K + 2x_K + 3 + y_{K+1}^2 - y_{K+1} - y_K^2 + y_K$$

$P_K < 0 \quad y_{K+1} = y_K$

$$\boxed{P_{K+1} = P_K + 2x_K + 3 + (y_K^2) - y_K^2 - y_K + y_K}$$

If $P_K > 0 \Rightarrow y_{K+1} = y_K - 1$

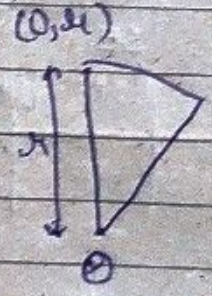
$$P_{K+1} = P_K + 2x_K + 3 + (y_K - 1)^2 - y_K^2 - (y_K - 1) + y_K$$

$$\boxed{P_{K+1} = P_K + 2x_K - 2y_K + 5}$$

$P_0 \rightarrow$ Initial Decision Parameter

$x_K = 0 \quad y_K = r$

$$\boxed{P_0 = \frac{5}{4} - r} \quad \underbrace{1.25 - r}_{\approx 1 - r}$$



NUMERICAL :-

Plot 1st Octant of circle centre at origin having radius 10 units

1) Plot $(0, r) \rightarrow (0, 10) \rightarrow (x_0, y_0)$

$p_0 = 1 - r = 1 - 10 = -9$ $\quad \boxed{(x_1, y_1) = (1, 10)}$

1.) $p_0 < 0 \rightarrow p_1 = p_0 + 2x_0 + 3 = -9 + 3 = -6$

2.) $p_1 < 0 \rightarrow p_2 = p_1 + 2x_1 + 3$

$= -6 + 2x1 = -1$ $\quad \boxed{(x_2, y_2) = (2, 10)}$

3.) $p_2 < 0 \rightarrow p_3 = p_2 + 2x_2 + 3 = -1 + 2x2 + 3 = 6$

$\boxed{(x_3, y_3) = (3, 10)}$

4) $p_3 > 0$ $p_4 = p_3 + 2(x_3 - y_3) + 5 = -3$

$\boxed{(x_4, y_4) = (4, 9)}$

5) $p_4 < 0$ $p_5 = p_4 - 3 + 8 + 3 = 8$ $\quad \boxed{(x_5, y_5) = (5, 9)}$

6) $p_5 > 0$ $p_6 = 8 + 2(5 - 9) + 5 = 5$ $\quad \boxed{(x_6, y_6) = (6, 8)}$

7) $p_6 > 0 \rightarrow p_7 = 5 + 2(6 - 8) + 5 = 6$ $\quad \boxed{(x_7, y_7) = (7, 7)}$

this. or if $x > y$ stop one step before

Bresenham's Circle Drawing Algo :-

1. Determine the radius (r) of circle from given information
2. Ensure the centre of circle lies at origin (0,0)
3. Plot first pixel of the first Octant as (0,r) [i.e. x=0 y=r]
4. Calculate initial ~~value~~ decision parameter as $\boxed{d = 3 - 2r}$
5. Repeat till $x \leq y$:
   (i) If $d_k < 0$ then :
   - $d_{k+1} = d_k + 4x_k + 6$
   - $x_{k+1} = x_k + 1 \rightarrow y_{k+1} = y_k$

   ii) Else if ($d_k \geq 0$) then :-
   - $d_{k+1} = d_k + 4(x_k - y_k) + 10$
   - $x_{k+1} = x_k + 1$
   - $y_{k+1} = y_k - 1$

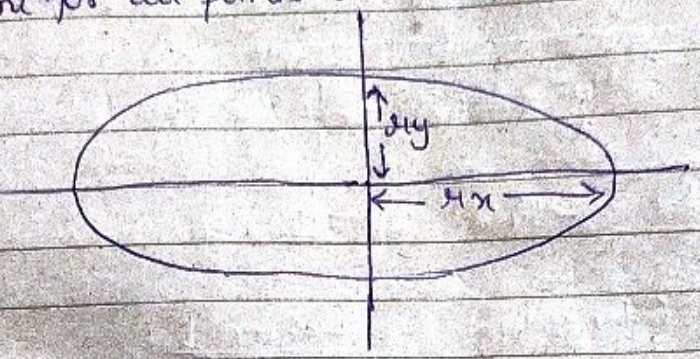
6. Plot (x_{k+1}, y_{k+1})
7. Determine & Plot other octant as well

If Centre is at (x_c, y_c) then

$(x_{k+1}, y_{k+1}) \rightarrow (x_{k+1} + x_c, y_{k+1} + y_c)$

ELLIPSE :-

An ellipse is defined as the set of points such that the sum of the distance from two fixed positions (foci) is same for all points.



$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ centre at $(0, 0)$

Major axis $= 2a = 2r_x$ Semi-$a = r_x$

Minor axis $= 2b = 2r_y$ Semi-$b = r_y$

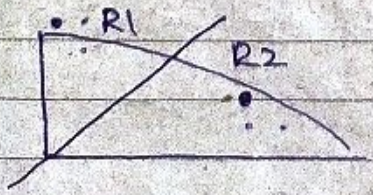
$\Rightarrow b^2x^2 + a^2y^2 - a^2b^2 = 0$ OR

$$\boxed{r_y^2x^2 + r_x^2y^2 - r_x^2r_y^2 = 0}$$

$= 0$ ON Ellipse < 0 Inside > 0 Outside

Quadrant Symmetry 4 way Symmetry

2 different Octant of a Quadrant to Plot



Algorithm :-

- Read radii r_x & r_y
- Initialise starting point of Region 1 as $x=0$ $y=r_y$
- Calculate $\boxed{p1_0 = r_y^2 - r_x^2 r_y^2 + \frac{1}{4} r_x^2}$
- Calculate $dx = 2r_y^2 x$ $dy = 2r_x^2 y$
- Repeat while $(dx < dy)$ (Region 1)
  - → Plot (x, y)
  - → if $(p1 < 0)$
    - $x = x+1$
    - Update dx → $2r_y^2 x$ (y same)
    - $p1 = p1 + 2r_y^2 x + r_y^2$ ($2r_y^2 x$ = dx)
  - else → $p1 \geq 0$
    - $x = x+1$, $y = y-1$
    - Update dx → $2r_y^2 x$
    - Update dy → $2r_x^2 y$
    - $p1 = p1 + dx - dy + r_y^2$
- when $(dx \geq dy)$ Plot Region 2 at :
  - → Find $\boxed{p2_0 = r_y^2 \left(x+\frac{1}{2}\right)^2 + r_x^2 (y-1)^2 - r_x^2 r_y^2}$
  - → Repeat till $(y > 0)$
    - → Plot (x, y)
    - → if $(p2 > 0)$
      - $x = x$ $y = y-1$
      - Update dy : $2r_x^2 y$
      - $p2 = p2 - dy + r_x^2$

else { x = x + 1
y = y - 1

new dy = dy - $2r_x^2$ OR $2r_x^2$ new r_x r_y
dx = dx + $2r_y^2$ or $2r_y^2$
p2 = p2 + dx - dy + r_x^2 }

Ex $r_x = 8$ $r_y = 6$

(x_k, y_k)	Decision Param: $P1_0$ or P_2	(x_{k+1}, y_{k+1})	$2x_{k+1} r_y^2$ dx	$2y_{(k+1)} r_x^2$ dy
P1			$r_x = 8$	$r_y = 6$
1. (0,6)	$P1_0 = -332$	(1, 6)	(2(1)(36)) = 72	2(6)(64) = 768
2. (1,6)	$P1_1 = -332 + 72 + 36 = -224$	(2,6)	4(36) = 144	= 768
3. (2,6)	$P1_2 = -224 + 144 + 36 = -44$	(3,6)	6(36) = 216	768
4. (3,6)	$P1_3 = -44 + 216 + 36 = 208$	(4,5)	8(36) = 288	640
5. (4,5)	$P1_4 = 208 + 288 + 36 - 640 = -108$	(5,5)	10(36) = 360	640
6. (5,5)	$P1_5 = -108 + 360 + 36 = 288$	(6,4)	12(36) = 432	8(64) = 512
7. (6,4)	$P1_6 = 244$	(7,3)	(504)	(384)
P2				
0. (7,3)	$P2_0 = 36(7 + \frac{1}{2})^2 + 64(2)^2 - 36(64) = -23$	(8, 2)	16(36) = 576	256
1. (8,2)	$P2_1 = -23 + 576 + 64 - 256 = 361$	(8,1)	576	2(64) = 128
2. (8,1)	$P2_2 = 361 - 128 + 64 = 297$	(8,0)	-	-

FILLED AREA PRIMITIVES:-

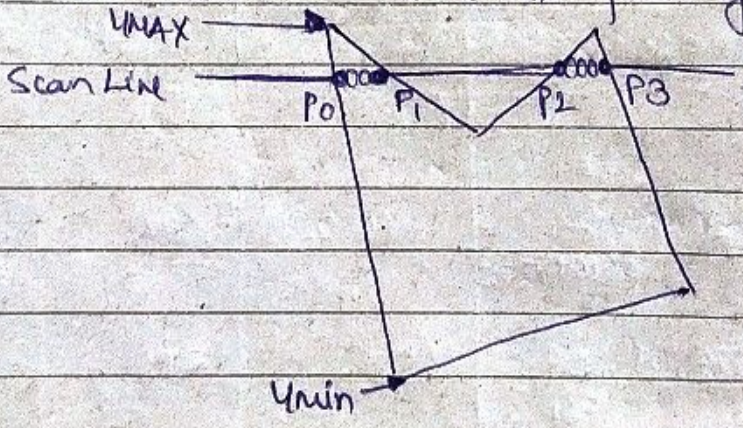
→ In practical we often use polygon which are filled with some color or pattern inside it.

→ Two basic approach to area filling on raster System
- One way is to determine the overlap intervals for scan line that cross the area.
- Another way is to fill the area is to start from given interior position & paint outwards from this point until we encounter boundary.

SCAN-LINE POLYGON FILL ALGORITHM -

this algo works by intersecting scanline with polygon edges & fills the polygon between pairs of intersections.

Step 1 - Find out Ymin & YMAX from given polygon.



Step 2. Scanline intersects with each edge of polygon from Ymin to YMAX Name each intersection point of the polygon. As per fig they are name P0, P1, P2, P3

Step 3. Sort the intersection point in the increasing order of X coordinate i.e P0, P1, P1, P2, & P2, P3.

Step 4. Fill all those pair of coordinates that are inside polygons & ignore the alternate pairs.

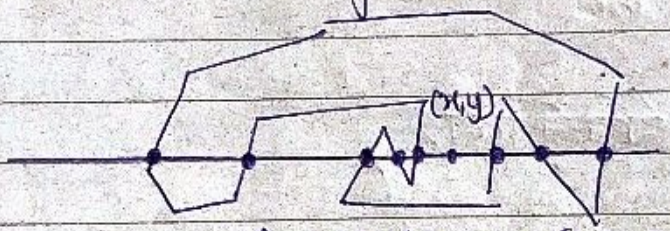
INSIDE OUTSIDE TEST

this method is also known as counting no. method. While filling an object, we often need to identify whether particular point is inside the object or outside it. There are two methods by which we can identify whether particular point is inside or outside an object.

- Odd - Even Rule
- Non zero winding number Rule

Odd Even Rule - Also known as odd parity rule or ~~even~~

In this technique,
we count the edge crossing along the line from any point x,y to Infinity.
If no. of interaction is odd, then point x,y is an interior point
& If no. is even, then x,y is an exterior point

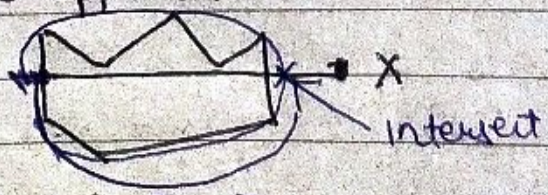


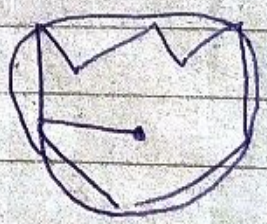
No of intercections point on left side is 5 & on right is 3 from both end the no. of inter-section point is odd, so the point is considered within object.

NON ZERO WINDING RULE :-

this method is also used with simple polygons to test given point is interior or not.

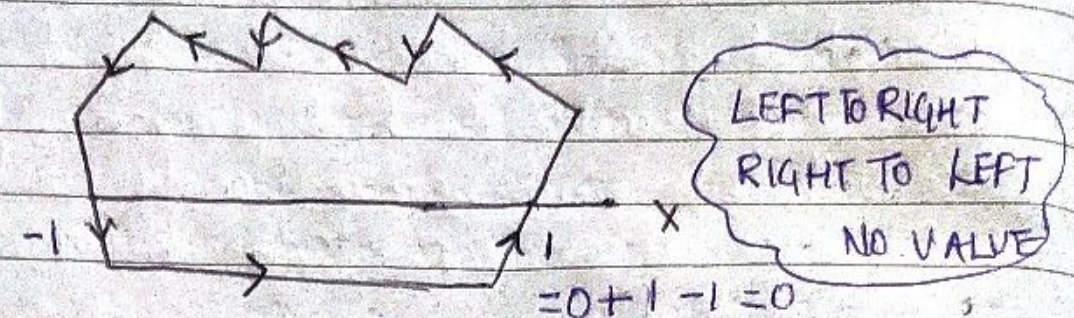
Pin elastic approach :-





No line cut

~~Point OUTSIDE IF NO. OF TIMES INSIDE~~



$= 0 + 1 - 1 = 0$

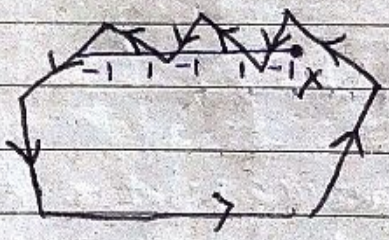
Initial Winding No $= 0$

→ If Line crosses an edge directed bottom to top add 1 to winding no

→ TOP - BOTTON ADD -1 to W.no.

→ POINT OUTSIDE IF WNo. is Zero

→ Otherwise Inside



$0 - 1 + 1 - 1 + 1 - 1 = -1$

Other than zero

Compairson B/w odd even & Non zero Winding :-

for Simple Object gives same result

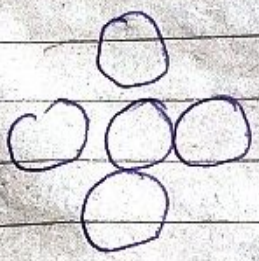
but for complicated shape both rule gives different result

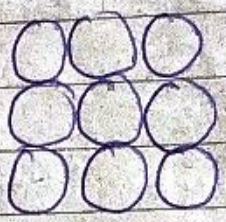
BOUNDARY FILLED ALGO :—

This Algo uses recursive method. first of all, a starting pixel called as the seed is considered.

→ The Algo. checks boundary pixel or adjacent pixels are colored or not. If pixel is already filled then leave it, otherwise fill it.

Filling is done using 4 connect. or 8 connected approaches.





Color specified in parameter fill color (f-color)

Boundary specified with parameter boundary color (b-color)

Procedure :-

```
boundary-fill4 (x, y, f-color, b-color)
{ if (getpixel (x,y) != b-color && get pixel (x,y) != f-color)
  {
      putpixel (x, y, f-color)
      boundary-fill4(x+1, y, f-color, b-color);
          "       "  (x, y+1,  "  ,  "  );
          "       "  (x-1, y ,  "  ,  "  );
          "       "  (x, y-1,  "  ,  "  );
  }
}
```

Same procedure is used for 8-connected by including 4 additional statements to test diagonal positions, such as (x+1, y+1)

FLOOD FILL ALGO :-

In this method, a point or seed which is inside region is selected. The four connected or eight connected approaches are used to fill with Specific color.

In this, we start from a specific interior point (x,y) & reassign all pixel values are currently set to a given interior color with desired color.

Disadvantage :-

Slow, May be fail for large polygon, Initial pixel requires more knowledge about surrounding pixel

ALGO :-

Procedure :-

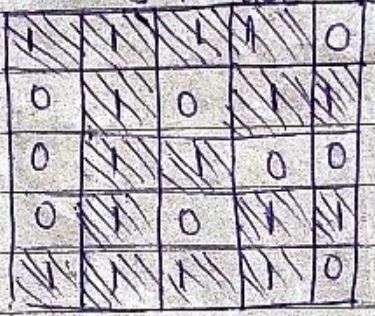
```
flood_fill4( x,y, new-color, old-color )
{
    if(getpixel(x,y) == = old-color)
    {  putpixel (x,y, new-color)
        flood-fill4(x+1,y, new-color, old-color);
                 "      (x, y+1, "  ,  "  );
                 "      (x-1, y, "  "  );
                 "      (x, y-1, "  ,  "  );
    }
}
```

CHARACTER GENERATION :-

→ We can display letter & no in variety of size & ~~fonts~~ style.
→ Typeface - The overall design style for the set of character.
→ Originally font was referred to a set of cast metal character forms in particular size & format. But now, the term font & typeface are often used interchangably.
→ Two different representations are used for storing computer fonts :-

Bitmap Font

Simple Method for representing character shapes in particular type face is to use rectangular grid pattern.

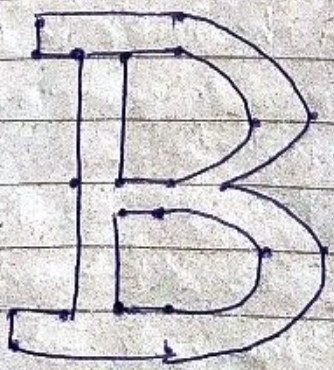


B

→ Simplest to design & display as character grid only need to maintain frame Buffer Position.
→ It require more space as each variation must be stored in a font cache.

Outline Font -

→ In this method character is generated using curve section & straight line as combine assembly



→ To display character we need to fill interior region of character.
→ This method require less storage as each variation does not required a distinct font

Date / /

Page No.

Date / /

Page No.

ype, its dimension & its color
also be displayed using

- Solid
- Dashed

or Index(lc)

↳ const Specifying color

↳ line is displayed

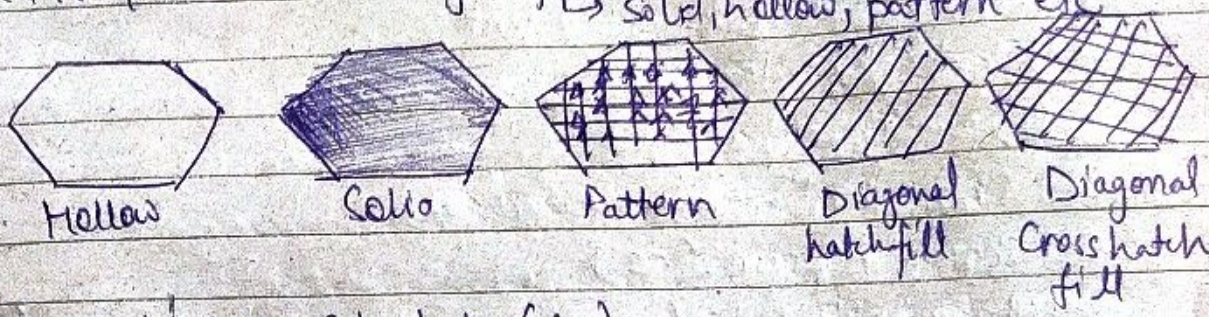
on.

AREA FILL ATTRIBUTES

→ for filling area we have choices between solid color or patter.
→ Area can be painted by various brushes & styles.

fill Style :

→ In PHIGS setInteriorStyle(fs) ↳ Solid, hollow, pattern etc



→ for setInteriorColorIndex(fc)

Pattern fill :-

setInteriorStyleIndex(pi)

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

SOFT-FILL :- (It is modified Boundary & flood fill algo)

It is used to recolor or repaint so that we can obtain layer of multiple color & get new color combination.

→ One use of this algo is soften the full boundary so that blurred effect will reduce the aliasing effect.

for ex :-

pixel → $p = tF + (1-t)B$

where t amount of foreground color
F is foreground color
B is background color

If we use RGB component then :

$p = (p_r, p_g, p_b)$ $f = (f_r, f_g, f_b)$ $b = (b_r, b_g, b_b)$

then we calculate, $t = \frac{P_k - B_k}{F_k - B_k}$

is controlled by attributes

ntation

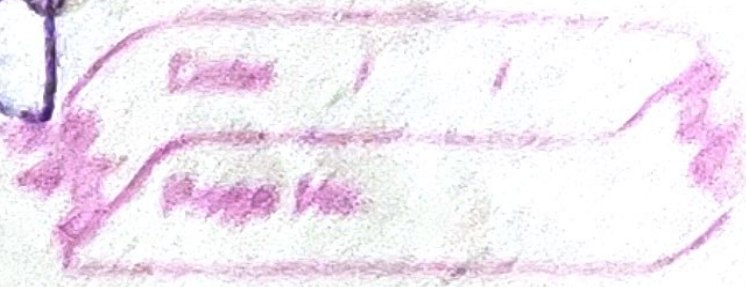
audio signals in time
lly sampled signal

Date / /
Page No.

erence of jagged
ized image can

hrically occurs due

considering region:



a, & compute pixel
sunels objects

shades of gray based

→ Tw

→ the

→ ...

of

Two drawbacks of this method :-

→ There is a technical & economic limit for increasing the resolution of virtual image.

→ ∵ the frequency of image can extend to infinity, it just reduces aliasing by raising the Nyquist limit - shift the effect of the frequency spectrum.