APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

STUDY MATERIALS

06/08/18

Module II

Line Drawing Algorithms

1) DDA (Digital Differential Analyzer)

Algorithm

```
#define ROUND(a) ((int)(a+0.5))
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).
  xincrement = dx/float(steps).
  yincrement = dy/float(steps).
  setpixel(ROUND(x), ROUND(y)).
  for(k=0 ; k<steps; k++)
  { x+= xincrement.
    y+= yincrement
    setpixel(ROUND(x), ROUND(y))
  }
}
```

eg:- $(x_a, y_a) = (20, 10)$, $(x_b, y_b) = (30, 18)$

$dx = 30 - 20 = 10$

$dy = 18 - 10 = 8$

$x = 20, y = 10$

if $(20 > 8)$ ✓ $\Rightarrow$ steps = 10

xincrement = 10/10 = 1

yincrement = 8/10 = 0.8

1st point → (20.5, 10.5)

k	x	y	Point to be plotted
0	21	10.8	(21.5, 11.3)
1	22	11.6	(22.5, 12.1)
2	23	12.4	(23.5, 12.9)
3	24	13.2	(24.5, 13.7)
4	25	14	(25.5, 14.5)
5	26	14.8	(26.5, 15.3)
6	27	15.6	(27.5, 16.1)
7	28	16.4	(28.5, 16.9)
8	29	17.2	(29.5, 17.7)
9	30	18	(30.5, 18.5)

Advantage

* Calculation is simple (only addition is used).

Disadvantage

* Round off errors occurs.

2) Bresenham's Line Drawing Algorithm

- Used when lines have a slope less than 1.

Algorithm

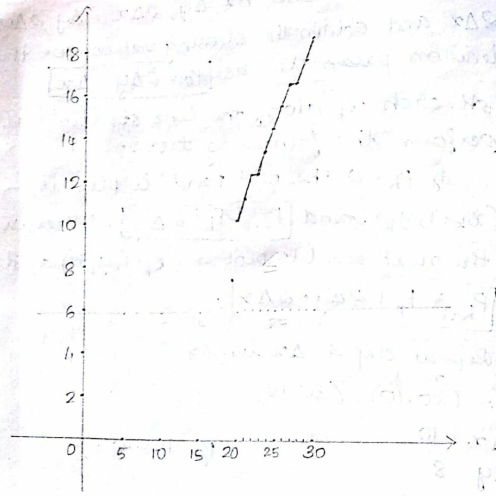
1. Input the 2 line endpoints and store the left endpoints in (x_0, y_0).
2. Load (x_0, y_0) into the frame buffer, ie, plot the 1st point.
3. Calculate constants $\Delta x, \Delta y, 2\Delta x, 2\Delta y, 2\Delta y - 2\Delta x$ and obtain the starting value for the decision parameter as $\boxed{P_0 = 2\Delta y - \Delta x}$.
4. At each x_k along the line starting at k=0, perform the following test:-
   If $P_k < 0$, the next point to plot is, (x_{k+1}, y_k) and $\boxed{P_{k+1} = P_k + 2\Delta y}$. Otherwise, the next point to plot is (x_{k+1}, y_{k+1}) and $\boxed{P_{k+1} = P_k + 2\Delta y - 2\Delta x}$.
5. Repeat step 4 Δx times.

eg:- (x_0, y_0) (20,10), (30,18)

$\Delta x = 10$
$\Delta y = 8$
$2\Delta x = 20$
$2\Delta y = 16$
$2\Delta y - 2\Delta x = -4$

$P_0 = 16 - 10 = 6$

k	P_k	(x_{k+1}, y_{k+1})	
0	6	(21, 11)	$P_1 = 6 - 4 = 2$ //
1	2	(22, 12)	$P_2 = 2 - 4 = -2$ //
2	−2	(23, 12)	$P_3 = -2 + 16 = 14$ //
3	14	(24, 13)	$P_4 = 14 - 4 = 10$ //
4	10	(25, 14)	$P_5 = 10 - 4 = 6$ //
5	6	(26, 15)	$P_6 = 6 - 4 = 2$ //
6	2	(27, 16)	$P_7 = 2 - 4 = -2$
7	−2	(28, 16)	$P_8 = -2 + 16 = 14$ //
8	14	(29, 17)	$P_9 = 14 - 4 = 10$ //
9	10	(30, 18)	



08/08/18

Circle generation Algorithms

1) Midpoint Circle Algorithm.

Algorithm

1. Input radius r and circle centre (x_c, y_c) and obtain the 1st point on the circumference of a circle centred on the origin as $(x_0, y_0) = (0, r)$
2. Calculate the initial value of the decision parameter as $P_0 = \frac{5}{4} - r \left(\sim (1-r)\right)$.
3. At each x_k position, starting at $k=0$, perform the following test :-

   If $P_k < 0$, the next point along the circle centred on $(0,0)$ is (x_{k+1}, y_k) and $\boxed{P_{k+1} = P_k + 2x_{k+1} + 1}$

   Otherwise, the next point along the circle is (x_{k+1}, y_{k-1}) and $\boxed{P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}}$.

4. Determine symmetry points in the other ~~7~~ octants.
5. Move each calculated pixel position (x,y) onto the circular path centred on (x_c, y_c) and plot the coordinate values $x = x + x_c$, $y = y + y_c$.
6. Repeat steps 3 through 5 until $x \geq y$.

eg:- Radius $r = 10$, centre by default $= (0,0)$

$\therefore (x_0, y_0) = (0, 10)$

$P_0 = 1 - r = 1 - 10 = -9$

k	P_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	-9	(1,10)	2	20
1	-6	(2,10)	4	20
2	-1	(3,10)	6	20
3	6	(4,9)	8	18
4	-3	(5,9)	10	18
5	8	(6,8)	12	16
6	5	(7,7)		

2) Bresenham's Incremental Circle Algorithm.

we get a quadrant.

Algorithm

All variables are assumed integers

Initialise the variables $x_i = 0$, $y_i = R$, $\Delta i = 2(1-R)$

limit = 0.

while $(y_i \geq \text{limit})$.

{ setpixel (x_i, y_i).

//determine if case 1 or 2/4 or 5/3.

if $(\Delta i < 0)$ then

$\delta = 2\Delta i + 2.y_i - 1$

```
if (δ ≤ 0) then
    call mh (x_i, y_i, Δ_i)
else
    call md (x_i, y_i, Δ_i)
end if
else if (Δ_i > 0) then
    δ' = 2Δ_i - 2x_i - 1
    if (δ' ≤ 0) then
        call md (x_i, y_i, Δ_i).
    else
        call mv (x_i, y_i, Δ_i)
    end if
else if (Δ_i = 0) then
    call md (x_i, y_i, Δ_i)
end if
end while
finish
```

Move Horizontally

```
subroutine mh (x_i, y_i, Δ_i)
    x_p = x_i + 1
    Δ_i = Δ_i + 2x_p + 1
end sub.
```

Move Diagonally

```
subroutine md (x_i, y_i, Δ_i)
    x_i = x_p + 1
    y_i = y_i - 1
    Δ_i = Δ_i + 2x_p - 2y_p + 2
end sub
```

Move Vertically

```
subroutine mv (x_i, y_i, Δ_i)
    y_i = y_i - 1
    Δ_i = Δ_i - 2y_i + 1
end sub.
```

eg:- Consider the origin centred circle of radius 8.
$x_p = 0, y_i = 8, \Delta_i = 2(1-R)$
$= 2(1-8) = \underline{-14}$, limit = 0.

Setpixel	Δ_i	δ	δ'	(x,y)
~~(0,8)~~	-14	-13		(0,8)
(0,8)	-11	-7		(1,8)
	-6	3		(2,8)
	-12	-11		(3,7)
	-3	7		(4,7)
	-3	5		(5,6)
	1		-11	(6,5)
	-6			(6,4)
	9		3	(7,4)
	21		25	(8,3)
	37		55	(9,2)
	57			(10,1)
	9		3	(7,4)
	4		-7	(7,3)
	18		18	(8,2)

17 17

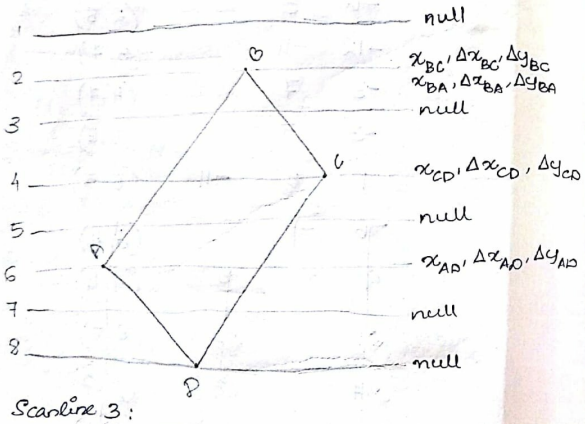
Scan Conversion

Process of organising the picture into precise pattern required by the graphics display.

1) Real time Scan conversion

(i) A simple active edge list using pointers.

ii) Sorted active edge list.



Scanline 3:

$x_{BC} + \Delta x_{BC}, \Delta x_{BC}, \Delta y_{BC} - 1$

$x_{BA} + \Delta x_{BA}, \Delta x_{BA}, \Delta y_{BA} - 1$

Scanline 5:

$x_{BA} + 3\Delta x_{BA}, \Delta x_{BA}, \Delta y_{BA} - 3$

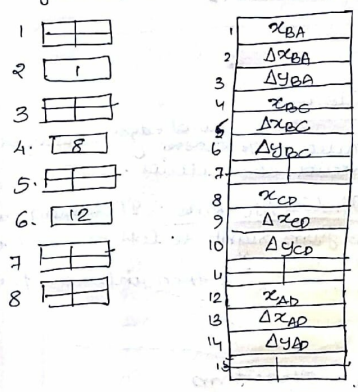
$x_{CD} + \Delta x_{CD}, \Delta x_{CD}, \Delta y_{CD} - 1$

Scanline 7:

$x_{CD} + 3\Delta x_{CD}, \Delta x_{CD}, \Delta y_{CD} - 3$

$x_{AD} + \Delta x_{AD}, \Delta x_{AD}, \Delta y_{AD} - 1$

13/08/18



2) Solid Area Scan Conversion (Polygon filling / Contour filling)

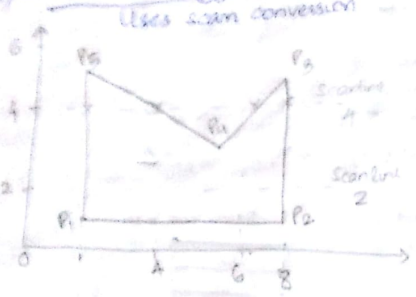
2 Methods:-

1) Scan Conversion :- Find whether the points are inside / outside the polygon till boundary.
2) Seed fill :- Assume that a point (seed point) is interior. Find its adjacent points and find its intensity. If intensity is same as seed point, it is interior. Otherwise exterior. till background or boundary.

Polygon Filling Algorithms

1) Scan Line Polygon fill algorithm

Uses scan conversion



Inside-Outside Test

1) **Odd-Even Rule** :- If no. of edges crossing is odd, then point is interior, else exterior.

2) **Non-Zero Winding No. Rule** :- If crossing an edge from right moves from right to left then, WNo = WNo + 1. Else, WNo = WNo - 1.
If WNo is non-zero, then interior point, else exterior.

2) Boundary fill Algorithm

- Seed fill algorithm.

4-connected (top, bottom, left, right) pixels are adjacent (Partial fill)

8-connected pixels (topleft, top, topright, bottom left, bottom, bottom right, left, right) (Complete fill)

Boundary Fill Algorithm

```
void BoundaryFill (int x,int y,int fill,int boundary)
{
  int current;
  current = getpixel (x,y);
  if(current ≠ boundary && current ≠ fill)
  {
    setcolour (fill);
    setpixel(x,y);
    BoundaryFill(x+1, y, fill, boundary);
    BoundaryFill(x-1, y, fill, boundary);
    BoundaryFill(x, y+1, fill, boundary);
    BoundaryFill(x, y-1, fill, boundary);
  }
}
```

3) Flood Fill Algorithm

Used to recolor. Also uses seed fill algorithm.

```
void FloodFill (int x,int y, int fillcolor, int oldcolor)
{ if(getpixel(x,y) = oldcolor)
  {
    setcolor (fillcolor)
    setpixel(x,y)
    FloodFill(x+1, y, fillcolor, oldcolor)
    FloodFill(x-1, y, fillcolor, oldcolor)
    FloodFill (x, y+1, fillcolor, oldcolor)
    FloodFill (x, y-1, fillcolor, oldcolor)
  }
}
```

try it now

A KTU STUDENTS PLATFORM

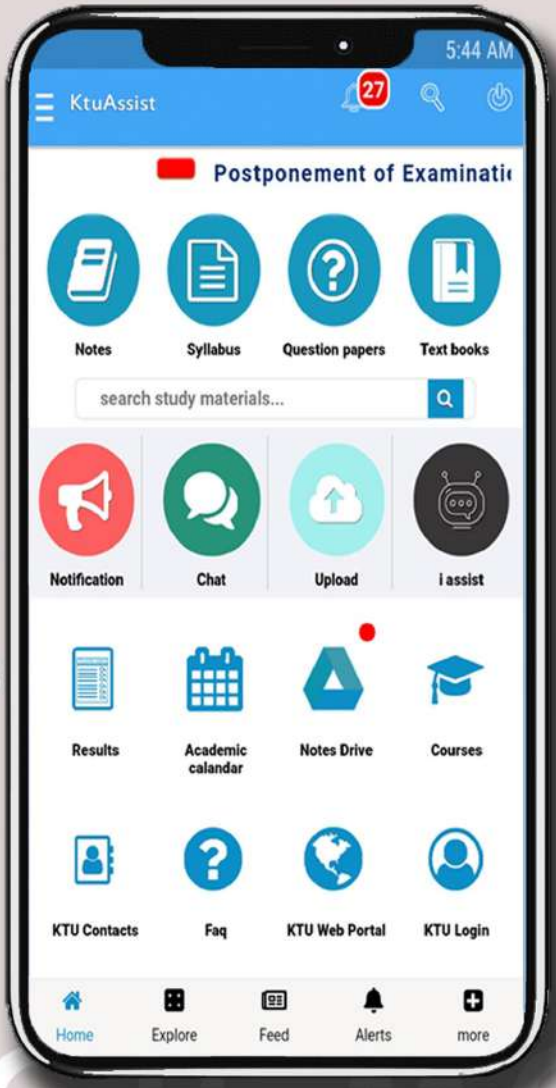
SYLLABUS

NOTES

TEXT BOOKS

QUESTION PAPERS

KTU NOTIFICATION



DOWNLOAD IT FROM GOOGLE PLAY

CHAT

FAQ

LOGIN

CALENDAR

MUCH MORE

DOWNLOAD APP

ktuassist.in

instagram.com/ktu_assist

facebook.com/ktuassist