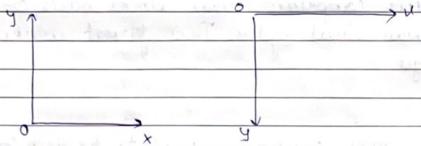
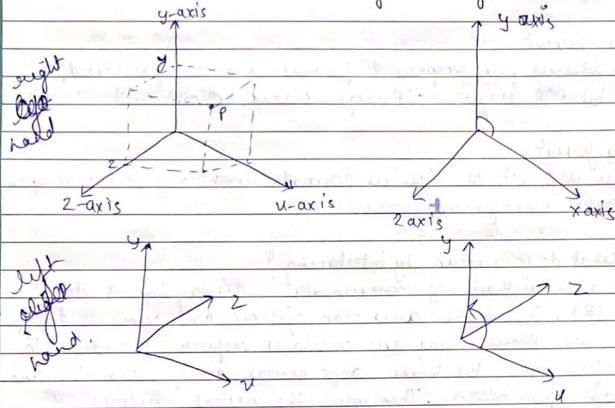


Date _____

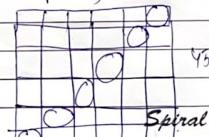
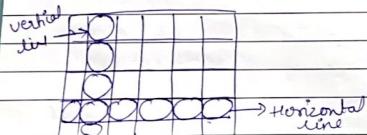
Two dimensional Cartesian Reference System



Three dimensional Cartesian Reference Systems



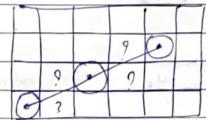
Basic concept in line drawing
The line should appear as a straight line
The line should be drawn rapidly.



Date _____

width is same
straight line
effective spacing is less
more height

straight line
width is not constant
more effective spacing
less height



line with other orientation

neither straight nor same width
we have to approximate
the pixel

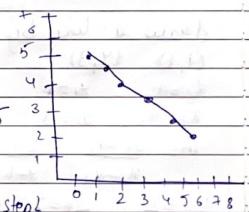
$$y = mx + c$$

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x} = \frac{dy}{dx}$$

Line drawing algorithm

Incremental Algorithm

- 1) Curposition = Start
Step = Increment
- 2) if (Curposition > End) < Accuracy goto step 5
if (Curposition < End) then goto step 3
if (Curposition > End) then goto step 4
- 3) Curposition = (Curposition + step) goto step 2
- 4) Curposition = Curposition - step goto step 2
- 5) STOP



Spiral

Digital Differential Analyzer (DDA) Algorithm

slope of straight line

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

$$\Delta y = m \cdot \Delta x = \frac{y_2 - y_1}{x_2 - x_1} \cdot \Delta x$$

$$\Delta x = \frac{\Delta y}{m} = \frac{y_2 - y_1}{y_2 - y_1}$$

q < k

$$\begin{aligned} \text{if } \Delta x \geq \Delta y \\ \text{then } \Delta x = 1 \\ x_{i+1} &= x_i + \Delta x \\ &= x_i + 1 \\ y_{i+1} &= y_i + \Delta y \\ &= y_i + m \cdot \Delta x \\ &= y_i + m \end{aligned}$$

$$\begin{aligned} \text{if } \Delta x \leq \Delta y \\ \text{then } \Delta y = 1 \\ x_{i+1} &= x_i + \Delta x \\ &= x_i + 1 \\ y_{i+1} &= y_i + 1 \\ &= y_i + 1 \end{aligned}$$

draw a line by DDA algo

$$(1, 1) \quad (4, 3)$$

$$x_1 = 1 \quad y_1 = 1$$

$$x_2 = 4 \quad y_2 = 3$$

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3 - 1}{4 - 1} = 2 = 0.67$$

$$\Delta y = 2 \quad \Delta x = 3$$

y > x

$$\therefore \Delta x = 1 \quad x_{i+1} = x_i + 1 = 1 + 1 = 2$$

$$\begin{aligned} y_{i+1} &= y_i + m \\ &= 1 + 0.67 = 1.67 \end{aligned}$$

Date _____

$$\begin{aligned} x_{i+1} &= x_i + 1 = 2 \\ y_{i+1} &= y_i + m = 1.67 + 0.67 \\ &= 2.34 \end{aligned}$$

(1, 1) (2, 1.67) (3, 2.34) (4, 3.01)

- Solution

Date _____

x_i	y_i	x_{next}	y_{next}
1	1	2	1.67
2	1.67	3	2.34
3	2.34	4	3.01

Step 1: Start

Step 2: declare $x_1, x_2, y_1, y_2, dx, dy, x, y$

Step 3: enter value of x_1, x_2, y_1, y_2

Step 4: calculate $dx = x_2 - x_1$

Step 5: calculate $dy = y_2 - y_1$

Step 6: if $|dx| > |dy|$

then step = abs(dx)

else

Step 7: $x_{inc} = dx$

Step

$y_{inc} = dy$

Step

assign $v = x_1$

assign $y = y_1$

set pixel (v, y)

$x = x + v_{inc}$

$y = y + y_{inc}$

Step 8: set pixels (Round(v), Round(y))

Step 9: Repeat step 8 until $x = x_2$

Step 10: Stop

Spiral

Spiral

Program to implement DDA

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>

Void main()
{
    int gd=DETECT, gm;
    float x, y, dx, dy, steps;
    int v0, v1, y0, y1;
    initgraph(&gd, &gm, "C:\TC\BG");
    setcolor(WHITE);
    x0 = 100, y0 = 200, v1 = 500, y1 = 300;
    dx = (float)(x1 - x0);
    dy = (float)(y1 - y0);
    if (dx >= dy)
        steps = dx;
    else
        steps = dy;
    dx = dx / steps;
    dy = dy / steps;
    x = x0;
    y = y0;
    i = 1;
    while (i <= steps)
    {
        putpixel(v, y, RED);
        x += dx;
        y += dy;
        i++;
    }
    getch();
    closegraph();
}
```

Date _____

Date _____

Advantages of DDA Algorithm

- Simplest algorithm
- Does not requires special skills for implementation
- Faster method to calculate pixel position
- Does not use multiplication theorem

Disadvantages of DDA Algorithm

- Floating point arithmetic is a time-consuming process
- Orientation dependent ∴ poor accuracy.
- Suitable for generating lines using software and less suited for hardware implementation

Spiral

Spiral

Date _____

Bresenham's line algorithm

$$M = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{dy}{dx}$$

$$\text{error term } e = 0$$

$$e = D_B - D_A$$

$$e < 0 \quad e \geq 0$$

$$D_B < D_A \quad D_B \geq D_A$$

find slope m and decision parameter

$$P_0 = 2dy - dx$$

$P < 0$	$P \geq 0$	$P > 0$
$x_{i+1} = x_i + 1$	$y_{i+1} = y_i + 1$	$\left\{ \begin{array}{l} y_{i+1} = \text{No change} \\ (y_i) \end{array} \right.$
$y_{i+1} = \text{No change}$	$y_{i+1} = y_i + 1$	$\left\{ \begin{array}{l} y_{i+1} = y_i + 1 \\ (y_i) \end{array} \right.$
$P_1 = P_0 + 2dy$	$P_1 = P_0 + 2dy - 2dx$	$P_1 = P_0 + 2dx - 2dy$

$$m < 1$$

$$m \geq 1$$

Numerical (1,1) (5,3)

$$\begin{matrix} x_1 = 1 & y_1 = 1 & m = \frac{1}{2} \\ x_2 = 5 & y_2 = 3 & \end{matrix}$$

$$P_0 = 2(2) - 4 = 0$$

$$x_{i+1} = 1 + 1 = 2$$

$$y_{i+1} = 1 + 1 = 2$$

$$P_1 = 0 + 4 - 8 = -4$$

$$P_1 = -4 + 4 = 0$$

$$P_1 = 0 + 4 - 8 = -4$$

Spiral

	(1,1)	(2,2)	(3,2)	(4,3)	P	x _i	y _i	x _{i+1}	y _{i+1}
	(5,3)				0	1	1	2	2
					-4	2	2	3	2
					0	3	2	4	3
					-4	4	3	5	9

Steps

- 1 Start quad (x₁, y₁) (x₂, y₂)
- 2 declare x_i, y_i, y_{i+1}, y_{i+2}, d, i, i₁, du, dy
- 3 $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$
- 4 $x = x_1$
- 5 $y = y_1$
- 6 $e = 2 + \Delta y - \Delta x$
- 7 $i = 1$
- 8 Plot (x, y)
- 9 while ($e \geq 0$)
 - 10 $y = y + 1$
 - 11 $e = e - 2 + \Delta y$
 - 12 $x = x + 1$
 - 13 $e = e + 2 + \Delta x$
- 14 i++
- 15 if ($i \leq \Delta x$) goto step 6
- 16 Stop

Spiral

Date _____

```

Program
#include <stdio.h>
#include <graphics.h>
void drawline (int x0, int y0, int x1, int y1)
{
    int dx, dy, p, x, y;
    dx = x1 - x0;
    dy = y1 - y0;
    x = x0;
    y = y0;
    p = 2 * dy - dx;
    while (x < x1)
    {
        if (p >= 0)
            putpixel (x, y, 7);
        y++;
        p = p + 2 * dy - 2 * dx;
    }
    else
        putpixel (x, y, 7);
    p = p + 2 * dy;
    y++;
}
int main ()
{
    int gdInit = DETECT, gmode, error, x0, y0, x1, y1;
    initgraph (&gdInit, &gmode, "C:\T\BGI");
    drawline (x0, y0, x1, y1);
    return 0;
}
note accept x0, y0,
      x1, y1

```

Spiral

Date _____

Advantage

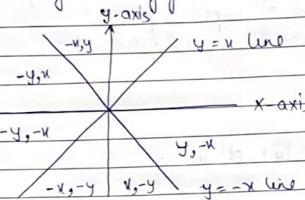
- it involves only integer arithmetic ... simple
- avoids generation of duplicate points
- can be implemented using hardware
- Faster

Disadvantage

Meant for basic line drawing

Spiral

Concept of circle drawing
A circle is a symmetric figure



A set of all points in a plane which are at a fixed distance from a fixed point

Circle equation

when Center (h, k)

$$(x-h)^2 + (y-k)^2 = r^2$$

when Center $(0, 0)$

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

Representation of a Circle

There are two standard methods of mathematically representing a circle centered at the origin.

► Polynomial Method

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

where x and y are coordinates
 r radius of circle

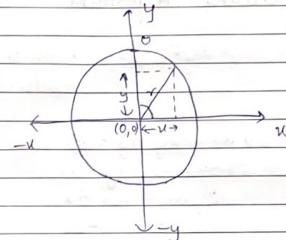
y coordinate is found for the known x coordinate

$$y = \sqrt{r^2 - x^2} \quad x \text{ to } 0$$

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

Spiral

Date _____
This will generate $\frac{1}{8}$ portion ($90^\circ \rightarrow 45^\circ$) of the circle.
Remaining is generated by reflection on 8-way symmetry of circle.



This method is an inefficient method as for each point both x and y must be squared and r^2 must be subtracted from x^2 then the square root of the result must be found out.

⇒ Trigonometric Method

$$x = r \cos \theta$$

$$y = r \sin \theta$$

where θ = current angle

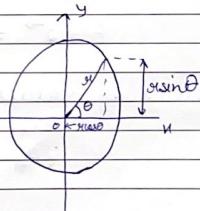
r = radius of circle

x and y coordinates

$$\theta = 0 \rightarrow \pi/4 \text{ radian}$$

each value of x and y is calculated

This method is more inefficient than polynomial because computation of sine and cosine is even more time-consuming



Spiral

Circle Drawing Algorithms

1) DDA circle drawing Algorithm

We know that $x^2 + y^2 = r^2$ with origin as the center of circle

$$2x \frac{dx}{du} + 2y \frac{dy}{du} = 0$$

$$2x \frac{dx}{du} = -2y \frac{dy}{du}$$

$$2y \frac{dy}{du} = -2x \frac{dx}{du}$$

$$\frac{dy}{du} = -\frac{x}{y}$$

Now we can construct the circle by using incremental X value $\Delta x = \epsilon y$ and incremental Y value $\Delta y = -\epsilon x$ where ϵ is calculated by radius of circle.

$$2^{n-1} \leq r \leq 2^n$$

$$\epsilon = 2^{-n}$$

eg

$$r = 50$$

$$2^{n-1} \leq 50 \leq 2^n$$

$$2^{6-1} \leq 50 \leq 2^6$$

$$32 \leq 50 \leq 64$$

$$\epsilon = 2^{-6} = 0.0156$$

next x & y values

$$x_{n+1} = x_n + \epsilon y_n$$

$$y_{n+1} = y_n - \epsilon x_{n+1}$$

Date _____

Algorithm

Date _____

Steps

- 1
- 2
- 3
- 4

Description

Read r and calculate the value of ϵ

$x_1 = start_x$ $y_1 = start_y$

$x_1 = start_x$ $y_1 = start_y$

do {

$$x_2 = x_1 + \epsilon y_1$$

$$y_2 = y_1 - \epsilon x_2$$

Plot ($\text{int}(x_2)$, $\text{int}(y_2)$)

$$x_1 = x_2$$

$$y_1 = y_2$$

} while $(y_1 - start_y) < \epsilon$ or $(start_x - x_1) > \epsilon$

5 Stop

Spiral

Spiral

2) Bresenham's Circle Drawing Algorithm
algorithm to plot 1/8 of the circle

Steps	Description
1	read x
2	calculate $d = 3 - 2x$
3	$x = 0$ $y = 0$
4	do { PLOT (x, y) if ($d < 0$) then $d = d + 4y + 6$ } else { $d = d + 4(x-y) + 10$ $y = y - 1$ $x = x + 1$ } until $(x > y)$
5	STOP

Date _____

Clipping

Date _____

Point Clipping

Point Clipping is used to determine whether the point is inside the window or not.

Clipping is a procedure of identifying those portions of a picture that are either inside or outside of viewing pane. In case of point clipping, we only show/point points on our window which are in range of our viewing pane, other point which are outside the range are discarded.

Algorithm

- i) get the minimum and maximum coordinates of both viewing pane
 $x_{max}, x_{min}, y_{max}, y_{min}$
- ii) get the coordinates for a point (x, y)
- iii) check whether given input lies between minimum and maximum coordinate of viewing pane.
 $x \leq x_{max} \quad x \geq x_{min} \quad y \leq y_{max} \quad y \geq y_{min}$
- iv) If yes display the point which lies inside the region otherwise discard it.

Spiral

Spiral

line Clipping

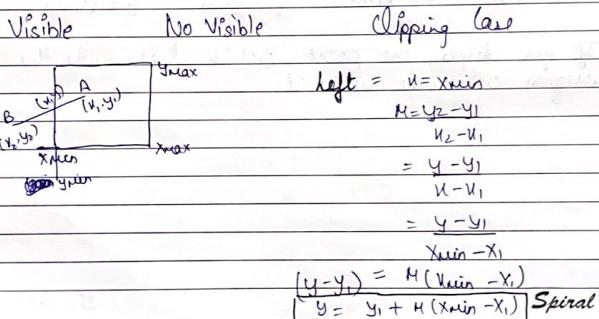
Cohen-Sutherland Algorithm

In this algo, it is detected whether line lies inside the screen or it is outside the screen.

There are 9 regions on the screen. Out of which one region is of the window and rest 8 are around it given by 4 digit binary number

TBRL		
	Top	
Left	0001 0000 window	0010
Bottom	0101 0100	0110

The central part is the viewing region or window all the lines which lie within the region are completely visible.



Date _____

Date _____

Right $x = x_{max}$

$$u = y_2 - y_1$$

$$M = y_2 - y_1$$

$$= y - y_1$$

$$= \frac{y - y_1}{x_{max} - x_1}$$

$$M(x_{max} - x_1) = y - y_1$$

$$y = y_1 + M(x_{max} - x_1)$$

Top $y = y_{max}$

$$M = y_2 - y_1$$

$$= y - y_1$$

$$= \frac{y - y_1}{x_{max} - x_1}$$

$$M(x_{max} - x_1) = y - y_1$$

$$y = y_1 + M(x_{max} - x_1)$$

Bottom $y = y_{min}$

$$M = y_2 - y_1$$

$$= y - y_1$$

$$= \frac{y - y_1}{x_{max} - x_1}$$

$$M(x_{max} - x_1) = y - y_1$$

$$y = y_1 + M(x_{max} - x_1)$$

Algorithm

Step1: Calculate positions of both endpoints of the line

Step2: Perform OR operation on both of these end points

Step3: If OR = 0000

then line is visible

else perform AND operation

If AND ≠ 0000

then line is invisible

else AND = 0000

line is partially inside the window and consider for clipping

Step4: Clipped case, find an intersection

$$m = (y_2 - y_1) / (x_2 - x_1)$$

where m =

if bit 1 is '1' line intersect with left boundary

$$y = y_1 + M(x_{min} - x_1)$$

if bit 2 is '1' line intersect with right boundary

$$y = y_1 + M(x_{max} - x_1)$$

if bit 3 is '1' line intersect with bottom boundary

$$x = x_1 + (y_{min} - y_1) / m$$

if bit 4 is '1' line intersect with top boundary

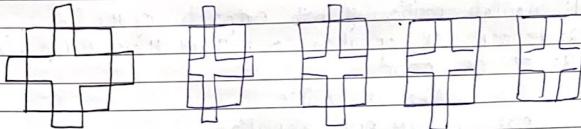
$$x = x_1 + (y_{max} - y_1) / m$$

Spiral

Polygon Clipping

Sutherland Hodgeson Algorithm

It is performed by processing the boundary of polygon against each window corner or edge. First of all entire polygon is clipped against one edge, then resulting polygon is considered, then the polygon is considered against the second edge, so on for all four edges.



Four possible solutions while processing

1) If the first vertex is outside the window, the second vertex is inside the window then the second vertex is added to the output list. The point of intersection of window boundary and polygon side (edge) is also added to the output list.

$O \rightarrow I$

(X B) save

2) If both vertices are inside window boundary Then only second vertex is added to the output list

$T \rightarrow I$

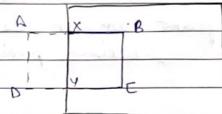
(D & C) save

3) If the first vertex is inside the window and second is an outside window. The edge which

Spiral

Date _____
Intersects with window is added to output list
 $I \rightarrow O$ (Y)

4) If both vertices are the outside window, then nothing is added to output list
 $O \rightarrow O$ (Ignore)



Spiral

Filling

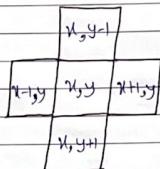
Filling is the process to fill color in objects of the computer screen.

A seed point or seed pixel is the location from where coloring starts after selecting the seed pixel we need to find its neighbors and color them if they are inside the boundary of polygon.

After selecting the seed point we can use any of the two models to find the neighboring pixels.

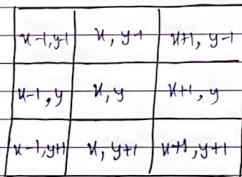
1) Four connected model

For a given seed point the four neighboring pixels top, bottom, right and left are used for filling and they can be calculated as this



2) Eight connected model

In this eight connected pixels of the seed pixels are used for filling the color.



Date _____

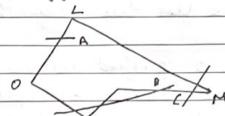
To determine a point lies inside or polygon or not in CG there are two methods

Date _____

Inside / Outside Test

Odd / Even Test (odd-parity rule)

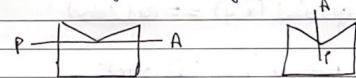
If the number of intersection is odd then we say the point is inside the polygon and if the number of intersection is even then we say the point is outside the polygon.



- (A) crosses single edge : it is inside
- (B) crosses three edges : it is inside
- (C) crosses two edges : it is outside

But this even-odd test fails when the intersection point is vertex

To handle this case few modifications are required



AP is a line which meet at the vertex. If these point lies on the same side of the constructed line then the intersection point counts as an even number of intersection. But if they lie on the opposite side of constructed line the intersection points counts as a single intersection.

Winding Number Method (Non Zero Winding no)

In this method we consider the polygon either clockwise or anticlockwise and mark all the sides in any of the orientation. After this count the upward direction as +1 downward as -1 if the total of the direction value is non-zero the point is interior else exterior.

Spiral

Date _____

Flood fill Algorithm

In this method a point or seed which is inside region is selected. This point is called a seed point. Then 4-connected or 8-connected approach is used to fill with specified color.

When boundary is of many colors and interior is to be filled with one color we use this algorithm.

In fill algo we start from a specified interior point (x, y) and assign all pixel values are currently set to a given interior color with the desired color using either a 4-connected or 8-connected approaches.

`void floodfill (x, y, fillcolor, oldcolor)`

{
 if (getpixel (x, y) == oldcolor)
 {

 putpixel (x, y, fillcolor);
 floodfill (x+1, y, fillcolor, oldcolor);
 floodfill (x, y+1, fillcolor, oldcolor);
 floodfill (x-1, y, fillcolor, oldcolor);
 floodfill (x, y-1, fillcolor, oldcolor);
 floodfill (x+1, y+1, fillcolor, oldcolor);
 floodfill (x+1, y-1, fillcolor, oldcolor);
 floodfill (x-1, y+1, fillcolor, oldcolor);
 floodfill (x-1, y-1, fillcolor, oldcolor);
 }

}

Spiral

Date _____

Boundary fill Algorithm

This algo uses recursive method.

First, a starting pixel called as the seed is considered. Checks boundary pixel or adjacent pixels are colored or not. If the adjacent pixel is already filled or colored then leave it, otherwise fill it. The filling is done using 4-connected ~~and~~ 8-connected approaches.

4-connected approach is more suitable than 8-connected

```
void fill (x, y, c, c1)
{
    int color;
    color = getpixel (x, y);
    if (color != c) & (color != c1)
        {
            putpixel (x, y, c)
            fill (x+1, y, c, c1);
            fill (x-1, y, c, c1);
            fill (x, y+1, c, c1);
            fill (x, y-1, c, c1);
        }
}
```

3
3

Spiral

Date _____

Scan-line polygon fill Algorithm

It is basically filling up of polygons using horizontal lines or scanlines.

Purpose is to fill the interior pixels of a polygon given only the vertices of figures.

Scanning is done using raster scanning concept on display device.

The beam starts scanning from top-left till bottom-right corner as end point.

The algo finds the point of intersection of line with polygon while moving from left to right and top to bottom. The various points of intersection are stored in frame buffer. The intensities of such points is high.

Concept of coherence property is used. Acc. to this, a pixel is inside the polygon, then its next pixel will be inside the polygon.

Steps

- 1) Locate the intersection points of the scan lines with the polygon edges
- 2) Pairing of intersection points
- 3) Move closer side as per scan line & sort all pairs
- 4) All pairs are sorted from Y_{max} to Y_{min}
- 5) sides get sorted on intersection point bases.
- 6) area filling starts now

Coherence Property

Utilizing property of one scene to another part of screen

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \Delta y / \Delta x \quad \text{As } \Delta y = 1 \text{ always}$$

$$x_{k+1} = \frac{1}{m} + x_k \quad (x_2 = x_{k+1}, y_2 = y_1)$$

