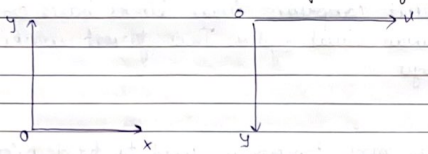
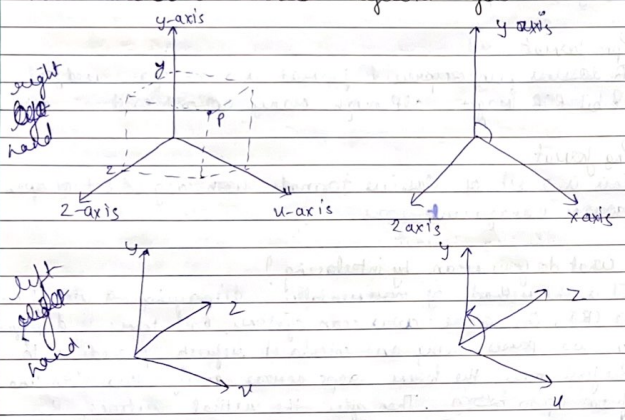
Date

Two dimensional Cartesian Reference System



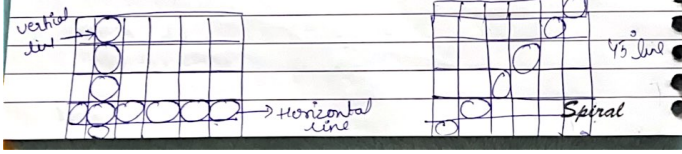
Three dimensional Cartesian Reference System



Basic Concept in line drawing

The line should appear as a straight line

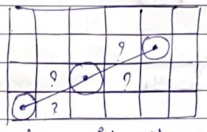
The line should be drawn rapidly.



Spiral

Date

width is same	straight line
straight line	width is not constant
effective spacing is less	more effective spacing
more bright	less bright



neither straight not same width

we have to approximate the pixel

line with other orientation

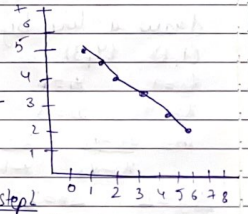
$$y = mx + c$$

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

Line drawing algorithm

Incremental Algorithm.

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



Spiral

Date

Digital Differential Analyzer (DDA) Algorithm

slope of straight line

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

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

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

if $\Delta x \geq \Delta y$
then $\Delta x = 1$

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

$y_{i+1} = y_i + \Delta y$
$= y_i + m \cdot \Delta x$
$= y_i + m$

if $\Delta x \leq \Delta y$
then $\Delta y = 1$

$x_{i+1} = x_i + \Delta x$
$= x_i + \frac{\Delta y}{m}$
$= x_i + \frac{1}{m}$

$y_{i+1} = y_i + \Delta y$
$= y_i + 1$

draw a line by DDA algo
(1,1) (4,3)

$x_1 = 1$ $y_1 = 1$
$x_2 = 4$ $y_2 = 3$

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

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

if $\Delta x > \Delta y$
$\therefore \Delta x = 1$ $x_{i+1} = x_i + 1 = 1 + 1 = 2$

$y_{i+1} = y_i + m$
$= 1 + 0.67 = 1.67$

Spiral

Date

$x_{i+1} = 2 + 1 = 3$
$y_{i+1} = y_i + m = 1.67 + 0.67$

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

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 ABS(dx) > ABS(dy)
then step = abs(dx)
else
Step 7: $x_{inc} = \frac{dx}{step}$
$y_{inc} = \frac{dy}{step}$
assign $x = x_1$
assign $y = y_1$
Step 8: set pixel (x, y)
step 9: $x = x + x_{inc}$
$y = y + y_{in}$
set pixels (Round(x), Round(y))
Step 10: Repeat step 9 until $x = x_2$
Step 11: stop

Spiral

Date

Program to implement DDA

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

void main() {
 int gd = DETECT, gm, i;
 float x, y, dx, dy, steps;
 int x0, x1, y0, y1;
 initgraph (&gd, &gm, "C:\\TC\\BGI");
 setbkcolor (WHITE);
 x0 = 100, y0 = 200, x1 = 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 (x, y, RED);
      x += dx;
      y += dy;
      i++;
 getch(); closegraph ();
}
```

Spiral

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 line using software and less suited for hardware implementation

Spiral

Date

Bresenham's line Algorithm

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

error term = e

$e = D_B - D_A$

$e < 0$	$e \geq 0$
$D_B < D_A$	$D_B \geq D_A$

find slope m and decision parameter

$P_0 = 2dy - dx$

$P < 0$	$P \geq 0$
$x_{i+1} = x_i + 1$	$x_{i+1} = x_i + 1$
$y_{i+1} =$ No change (y_i)	$y_{i+1} = y_i + 1$
$P_1 = P_0 + 2dy$	$P_1 = P_0 + 2dy - 2dx$

$m < 1$

$P < 0$	$P \geq 0$
$x_{i+1} =$ No change (x_i)	$x_{i+1} = x_i + 1$
$y_{i+1} = y_i + 1$	$y_{i+1} = y_i + 1$
$P_1 = P_0 + 2dx$	$P_1 = P_0 + 2dx - 2dy$

$m \geq 1$

Numerical (1,1) (5,3)

$x_1 = 1 \quad y_1 = 1 \quad m = \frac{1}{2}$

$x_2 = 5 \quad y_2 = 3$

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

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

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

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

$P_1 = -4 + 4 = 0$

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

Spiral

Date

(1,1) (2,2) (3,2) (4,3) (5,3)

P	x_i	y_i	x_{i+1}	y_{i+1}
0	1	1	2	2
-4	2	2	3	2
0	3	2	4	3
-4	4	3	5	3

Steps	Description
1	Start
2	~~Declare read (x_1,y_1) (x_2,y_2) $x_1, x_2, y_1, y_2, d, i_1, i_2, dx, dy$~~
3	$\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$
4	$x = x_1$
	$y = y_1$
5	$e = 2 * \Delta y - \Delta x$
6	$i = 1$
7	Plot (x,y)
8	while $(e \geq 0)$
	{
	$y = y + 1$
	$e = e - 2 * \Delta x$
	}
	$x = x + 1$
	$e = e + 2 * \Delta y$
9	$i + +$
10	if $(i \leq \Delta x)$ goto step 6
11	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; }
            x++
      }
}
int main ()
{
  int gdriver = DETECT, gmode, errors, x0, y0, x1, y1;
  initgraph (&gdriver, &gmode, "C:\\TC\\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

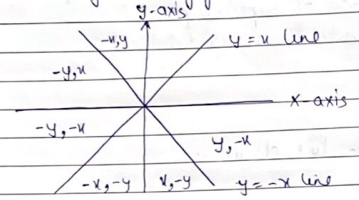
Disadvantage
Meant for basic line drawing

Spiral

Date

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 coordinate
r radius of circle

y coordinate is found for the known x coordinate

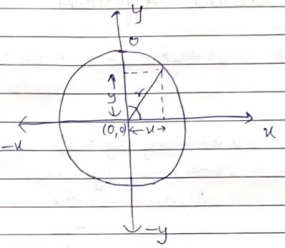
$$y = \sqrt{r^2 - x^2}$$

0 to $x = r/\sqrt{2}$

Spiral

Date

This will generate 1/8 portion (90° → 45°) of the circle Remaining is generated by reflection or 8-way symmetry of circle.



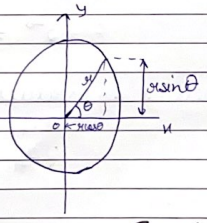
This method is an inefficient method as for each point both x and r must be squared and x^2 must be subtracted from r^2 then the square root of the result must be found out.

◁ Trigonometric Method

$x = r\cos\theta$ where θ = current angle
$y = r\sin\theta$ r = radius of circle
x and y coordinate

$\theta = 0 \rightarrow \pi/4$ radian
each value of x and y is calculated

This method is more inefficient than polynomial because computation of sinθ and cosθ is even more time-consuming



Spiral

Date

Circle Drawing Algorithms

▷ DDA circle drawing Algorithm

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

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

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

$$2^{n-1} \leq r < 2^n$$
$$\varepsilon = 2^{-n}$$

eg $r = 50$

$$2^{n-1} \leq 50 \leq 2^n$$
$$2^{6-1} \leq 50 < 2^6$$
$$32 \leq 50 < 64$$

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

next x & y values

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

Spiral

Date

Algorithm

Steps	Description
1	Read r and calculate the value of ε
2	start_x = 0 start_y = r
3	x_1 = start_x y_1 = start_y
4	do { $x_2 = x_1 + \varepsilon y_1$ $y_2 = y_1 - \varepsilon x_2$ Plot (int(x_2), int(y_2)) $x_1 = x_2$ $y_1 = y_2$ } while (y_1 − start_y) < ε or (start_x − x_1) > ε
5	Stop

Spiral

Date

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

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

Spiral

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 / outside of viewing pane. In case of point clipping, we only show / print points on our window which are in range of our viewing pane, others point which are outside the range are discarded.

Algorithm
1) get the minimum and maximum coordinates of both viewing pane
x_{max} , x_{min} , y_{max} , y_{min}

2) get the coordinate for a point (x, y)

3) check whether given input lies between minimum and maximum coordinate of viewing pane.
$x \leq x_{max}$ $x \geq x_{min}$ $y \leq y_{max}$ $y \geq y_{min}$

4) If yes display the point which lies inside the region otherwise discard it

Spiral

Date

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

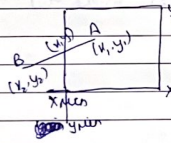
TBRL

	Top	
1001	1000	1010
0001 (Left)	0000 window	0010 (Right)
0101	0100	0110

Bottom

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

Visible | No Visible | Clipping Case



Left $x = x_{min}$

$$M = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y - y_1}{x - x_1} = \frac{y - y_1}{x_{min} - x_1}$$

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

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

Spiral

Date

Right $x = x_{max}$

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

$$M = \frac{y - y_1}{x - x_1}$$

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

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

Top $y = y_{max}$

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

$$\boxed{x = x_1 + \frac{(y_{max} - y_1)}{m}}$$

Bottom $y = y_{min}$

$$\boxed{x = x_1 + \frac{(y_{min} - y_1)}{m}}$$

Algorithm

Step 1: Calculate positions of both endpoints of the line

Step 2: Perform OR operation on both of these end points

Step 3: 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

Step 4: Clipped case, find an intersection

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

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

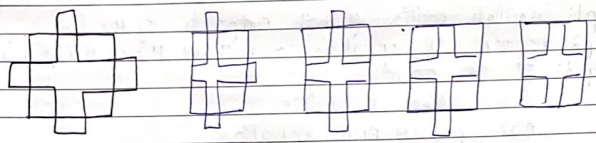
Spiral

Date

Polygon Clipping

Sutherland Hodgeman 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 an 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 → I
(X B) save

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

I → I
(2nd 'C') save

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

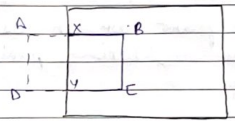
Date

intersects with window is added to output list

I → O (Y)

4) If both vertices are the outside window, then nothing is added to output list

O → O (Ignore)



Date

Filling

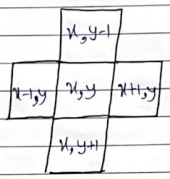
Filling is the process to fill color in objects of the computer screen

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

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

1) Four connected models

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.

x-1,y+1	x, y+1	x+1, y-1
x-1, y	x, y	x+1, y
x-1,y+1	x, y+1	x+1, y+1

Spiral

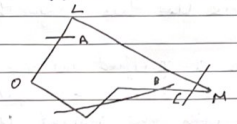
To determine a point lies inside a polygon or not in CG there are two methods

Date

Inside/Outside Test

Odd/Even Test (odd-parity rule)

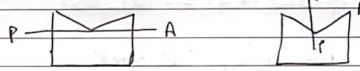
If the number of intersection is odd then we say the point is inside the polygon and if the number of intersection is even then we say the point is outside the polygon.



A cross single edge ∴ it is inside
B cross three edges ∴ it is inside
C cross 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 sight 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 reassign 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, fill_color, old_color)
{
    if ( getpixel (x,y) == old_color)
    {
        putpixel (x,y, fill_color);
        floodfill (x+1, y, fill_color, old_color);
        floodfill (x, y+1, fill_color, old_color);
        floodfill (x-1, y, fill_color, old_color);
        floodfill (x, y-1, fill_color, old_color);
        floodfill (x-1, y-1, fill_color, old_color);
        floodfill (x-1, y+1, fill_color, old_color);
        floodfill (x+1, y-1, fill_color, old_color);
        floodfill (x+1, y+1, fill_color, old_color);
    }
}
```

4 { } 8

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 & 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);
    }
}
```

Spiral

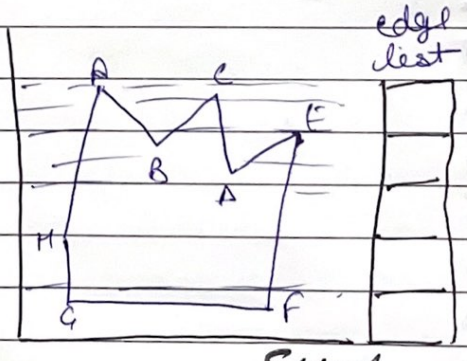
Date

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 line with the polygon edge
2) Pairing of intersection points
3) Move down 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

relating property of one scene to another part of screen

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

$x_{k+1} = \frac{1}{m} + x_k$ $(x_2 = x_{k+1}, x_k = x_1)$

Spiral