Scan Conversion Definition

It is a process of representing graphics objects a collection of pixels. The graphics objects are continuous. The pixels used are discrete. Each pixel can have either on or off state.

The circuitry of the video display device of the computer is capable of converting binary values (0, 1) into a pixel on and pixel off information. 0 is represented by pixel off. 1 is represented using pixel on. Using this ability graphics computer represent picture having discrete dots.

Any model of graphics can be reproduced with a dense matrix of dots or points. Most human beings think graphics objects as points, lines, circles, ellipses. For generating graphical object, many algorithms have been developed.

Advantage of developing algorithms for scan conversion

1. Algorithms can generate graphics objects at a faster rate.
2. Using algorithms memory can be used efficiently.
3. Algorithms can develop a higher level of graphical objects.

Examples of objects which can be scan converted

1. Point
2. Line
3. Sector
4. Arc
5. Ellipse
6. Rectangle
7. Polygon
8. Characters
9. Filled Regions

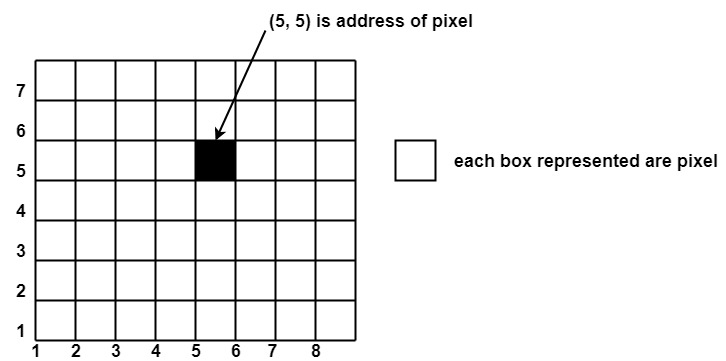
The process of converting is also called as rasterization. The algorithms implementation varies from one computer system to another computer system. Some algorithms are implemented using the software. Some are performed using hardware or firmware. Some are performed using various combinations of hardware, firmware, and software.

Pixel or Pel:

The term pixel is a short form of the picture element. It is also called a point or dot. It is the smallest picture unit accepted by display devices. A picture is constructed from hundreds of such pixels. Pixels are generated using commands. Lines, circle, arcs, characters; curves are drawn with closely spaced pixels. To display the digit or letter matrix of pixels is used.

The closer the dots or pixels are, the better will be the quality of picture. Closer the dots are, crisper will be the picture. Picture will not appear jagged and unclear if pixels are closely spaced. So the quality of the picture is directly proportional to the density of pixels on the screen.

Pixels are also defined as the smallest addressable unit or element of the screen. Each pixel can be assigned an address as shown in fig:



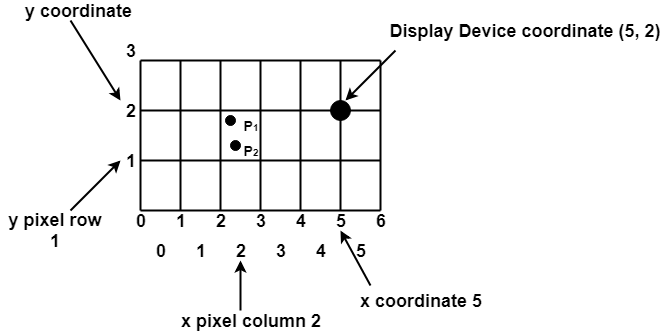
Different graphics objects can be generated by setting the different intensity of pixels and different colors of pixels. Each pixel has some co-ordinate value. The coordinate is represented using row and column.

P (5, 5) used to represent a pixel in the 5th row and the 5th column. Each pixel has some intensity value which is represented in memory of computer called a **frame buffer**. Frame Buffer is also called a refresh buffer. This memory is a storage area for storing pixels values using which pictures are displayed. It is also called as digital memory. Inside the buffer, image is stored as a pattern of binary digits either 0 or 1. So there is an array of 0 or 1 used to represent the picture. In black and white monitors, black pixels are represented using 1's and white pixels are represented using 0's. In case of systems having one bit per pixel frame buffer is called a bitmap. In systems with multiple bits per pixel it is called a pixmap.

Scan Converting a Point

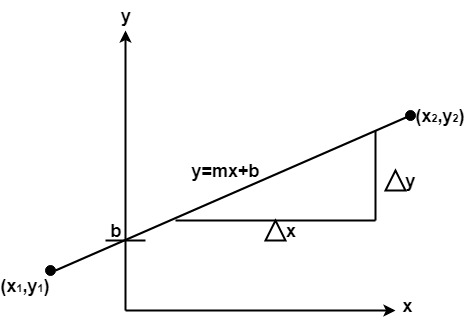
Each pixel on the graphics display does not represent a mathematical point. Instead, it means a region which theoretically can contain an infinite number of points. Scan-Converting a point involves illuminating the pixel that contains the point.

**Example:** Display coordinates points Scan Converting a Pointas shown in fig would both be represented by pixel (2, 1). In general, a point p (x, y) is represented by the integer part of x & the integer part of y that is pixels [(INT (x), INT (y).



Scan Converting a Straight Line

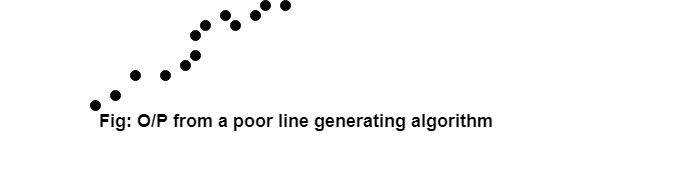
A straight line may be defined by two endpoints & an equation. In fig the two endpoints are described by (x1,y1) and (x2,y2). The equation of the line is used to determine the x, y coordinates of all the points that lie between these two endpoints.



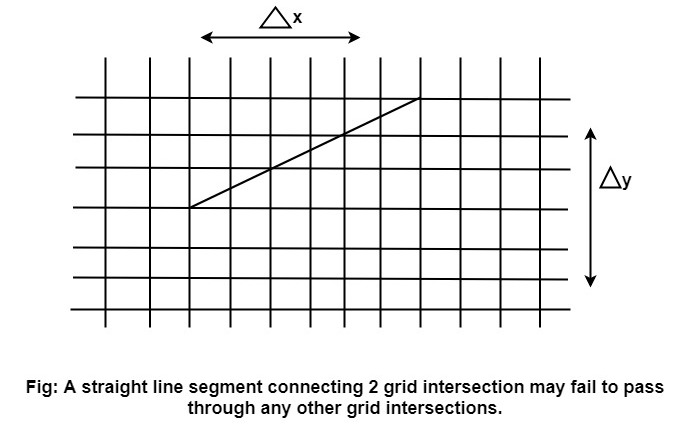
Using the equation of a straight line, y = mx + b where m = Scan Converting a Straight Line & b = the y interrupt, we can find values of y by incrementing x from x =x1, to x = x2. By scan-converting these calculated x, y values, we represent the line as a sequence of pixels.

Properties of Good Line Drawing Algorithm:

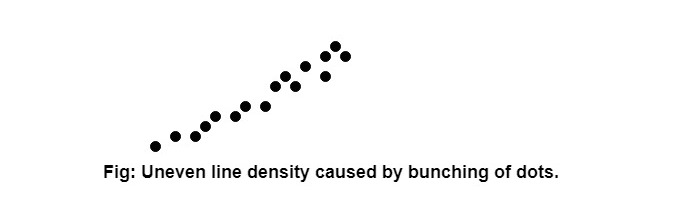
**1. Line should appear Straight:** We must appropriate the line by choosing addressable points close to it. If we choose well, the line will appear straight, if not, we shall produce crossed lines.



The lines must be generated parallel or at 45° to the x and y-axes. Other lines cause a problem: a line segment through it starts and finishes at addressable points, may happen to pass through no another addressable points in between.



**2. Lines should terminate accurately:** Unless lines are plotted accurately, they may terminate at the wrong place.



**3. Lines should have constant density:** Line density is proportional to the no. of dots displayed divided by the length of the line.

To maintain constant density, dots should be equally spaced.

**4. Line density should be independent of line length and angle:** This can be done by computing an approximating line-length estimate and to use a line-generation algorithm that keeps line density constant to within the accuracy of this estimate.

**5. Line should be drawn rapidly:** This computation should be performed by special-purpose hardware.

Algorithm for line Drawing:

1. Direct use of line equation
2. DDA (Digital Differential Analyzer)
3. Bresenham's Algorithm

Direct use of line equation:

It is the simplest form of conversion. First of all scan P1 and P2 points. P1 has co-ordinates (x1',y1') and (x2' y2' ).

Then       m = (y2',y1')/( x2',x1') and b = Scan Converting a Straight Line

If value of |m|≤1 for each integer value of x. But do not consider Scan Converting a Straight Line

If value of |m|>1 for each integer value of y. But do not consider Scan Converting a Straight Line

**Example:** A line with starting point as (0, 0) and ending point (6, 18) is given. Calculate value of intermediate points and slope of line.

**Solution:** P1 (0,0) P7 (6,18)

              x1=0  
              y1=0  
              x2=6  
              y2=18  
              Scan Converting a Straight Line

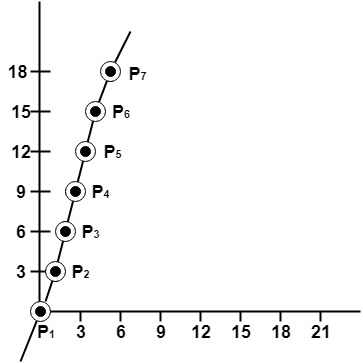
We know equation of line is  
              y =m x + b  
              y = 3x + b..............equation (1)

put value of x from initial point in equation (1), i.e., (0, 0) x =0, y=0  
              0 = 3 x 0 + b  
              0 = b ⟹ b=0

put b = 0 in equation (1)  
              y = 3x + 0  
              y = 3x

Now calculate intermediate points  
    Let x = 1 ⟹ y = 3 x 1 ⟹ y = 3  
    Let x = 2 ⟹ y = 3 x 2 ⟹ y = 6  
    Let x = 3 ⟹ y = 3 x 3 ⟹ y = 9  
    Let x = 4 ⟹ y = 3 x 4 ⟹ y = 12  
    Let x = 5 ⟹ y = 3 x 5 ⟹ y = 15  
    Let x = 6 ⟹ y = 3 x 6 ⟹ y = 18

So points are P1 (0,0)  
              P2 (1,3)  
              P3 (2,6)  
              P4 (3,9)  
              P5 (4,12)  
              P6 (5,15)  
              P7 (6,18)



Algorithm for drawing line using equation:

**Step1:** Start Algorithm

**Step2:** Declare variables x1,x2,y1,y2,dx,dy,m,b,

**Step3:** Enter values of x1,x2,y1,y2.  
              The (x1,y1) are co-ordinates of a starting point of the line.  
              The (x2,y2) are co-ordinates of a ending point of the line.

**Step4:** Calculate dx = x2- x1

**Step5:** Calculate dy = y2-y1

**Step6:** Calculate m = Scan Converting a Straight Line

**Step7:** Calculate b = y1-m\* x1

**Step8:** Set (x, y) equal to starting point, i.e., lowest point and xendequal to largest value of x.

              If dx < 0  
                  then x = x2  
              y = y2  
                        xend= x1  
        If dx > 0  
              then x = x1  
                  y = y1  
                        xend= x2

**Step9:** Check whether the complete line has been drawn if x=xend, stop

**Step10:** Plot a point at current (x, y) coordinates

**Step11:** Increment value of x, i.e., x = x+1

**Step12:** Compute next value of y from equation y = mx + b

**Step13:** Go to Step9.

# DDA Algorithm

DDA stands for Digital Differential Analyzer. It is an incremental method of scan conversion of line. In this method calculation is performed at each step but by using results of previous steps.

Suppose at step i, the pixels is (xi,yi)

The line of equation for step i  
              yi=mxi+b......................equation 1

Next value will be  
              yi+1=mxi+1+b.................equation 2  
              m =DDA Algorithm  
              yi+1-yi=∆y.......................equation 3  
              yi+1-xi=∆x......................equation 4  
              yi+1=yi+∆y  
              ∆y=m∆x  
              yi+1=yi+m∆x  
              ∆x=∆y/m  
              xi+1=xi+∆x  
              xi+1=xi+∆y/m

**Case1:** When |M|<1 then (assume that x1<x2)  
              x= x1,y=y1 set ∆x=1  
              yi+1=y1+m,     x=x+1  
              Until x = x2</x

**Case2:**When |M|<1 then (assume that y1<y2)  
              x= x1,y=y1 set ∆y=1  
              xi+1=DDA Algorithm,     y=y+1  
              Until y → y2</y

### Advantage:

1. It is a faster method than method of using direct use of line equation.
2. This method does not use multiplication theorem.
3. It allows us to detect the change in the value of x and y ,so plotting of same point twice is not possible.
4. This method gives overflow indication when a point is repositioned.
5. It is an easy method because each step involves just two additions.

### Disadvantage:

1. It involves floating point additions rounding off is done. Accumulations of round off error cause accumulation of error.
2. Rounding off operations and floating point operations consumes a lot of time.
3. It is more suitable for generating line using the software. But it is less suited for hardware implementation.

## DDA Algorithm:

**Step1:**Start Algorithm

**Step2:**Declare x1,y1,x2,y2,dx,dy,x,y as integer variables.

**Step3:** Enter value of x1,y1,x2,y2.

**Step4:** Calculate dx = x2-x1

**Step5:** Calculate dy = y2-y1

**Step6:** If ABS (dx) > ABS (dy)  
            Then step = abs (dx)  
            Else

**Step7:** xinc=dx/step  
            yinc=dy/step  
            assign x = x1  
            assign y = y1

**Step8:** Set pixel (x, y)

**Step9:** x = x + xinc  
            y = y + yinc  
            Set pixels (Round (x), Round (y))

**Step10:** Repeat step 9 until x = x2

**Step11:** End Algorithm

# Bresenham's Line Algorithm

This algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly.

In this method, next pixel selected is that one who has the least distance from true line.

The method works as follows:

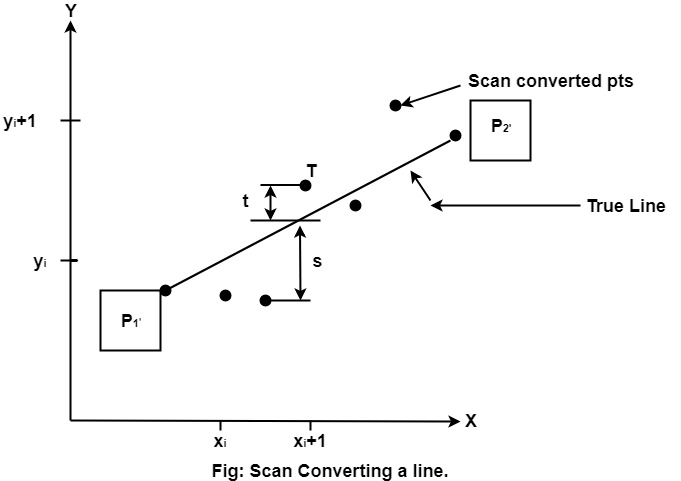
Assume a pixel P1'(x1',y1'),then select subsequent pixels as we work our may to the night, one pixel position at a time in the horizontal direction toward P2'(x2',y2').

Once a pixel in choose at any step

The next pixel is

1. Either the one to its right (lower-bound for the line)
2. One top its right and up (upper-bound for the line)

The line is best approximated by those pixels that fall the least distance from the path between P1',P2'.



### Advantage:

1. It involves only integer arithmetic, so it is simple.

2. It avoids the generation of duplicate points.

3. It can be implemented using hardware because it does not use multiplication and division.

4. It is faster as compared to DDA (Digital Differential Analyzer) because it does not involve floating point calculations like DDA Algorithm.

### Disadvantage:

1. This algorithm is meant for basic line drawing only Initializing is not a part of Bresenham's line algorithm. So to draw smooth lines, you should want to look into a different algorithm.