# Scan Conversion

AMARTYA KUNDU DURJOY
Lecturer, CSE, UGV

# Output Primitives

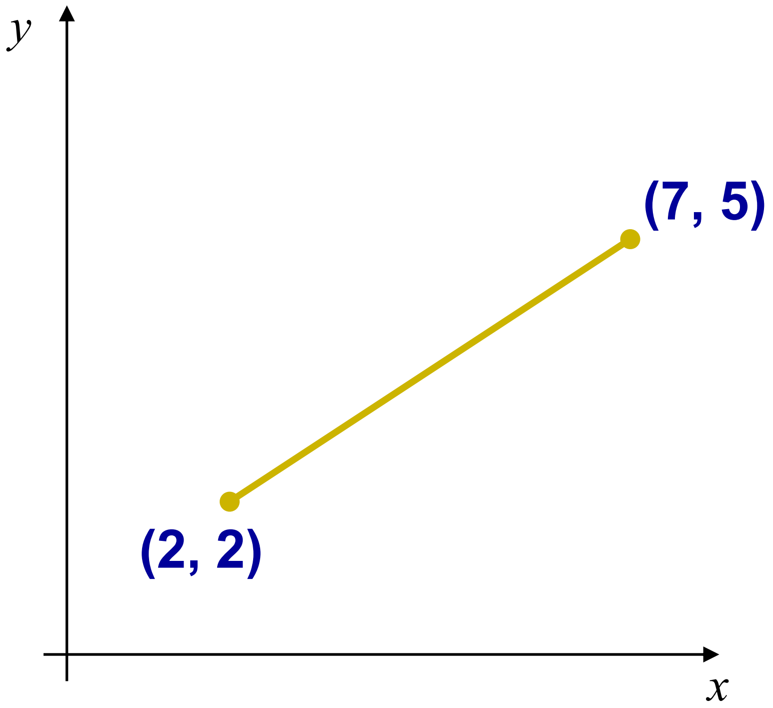
- Basic geometric structures used to describe scenes.
- Can be grouped into more complex structures.
- Each one is specified with input coordinate data and other information about the way that object is to be displayed.
- Examples: point, line and circle each one with specified coordinates.
- Construct the vector picture.

# Digital Representation

- Display screen is divided into scan lines and columns.
- Pixels positions are referenced according to scan line number and column number (columns across scan lines).
  - Scan lines start from 0 at screen bottom, and columns start from 0 at the screen left side.
- Screen locations (or pixels) are referenced with integer values.
- The frame buffer stores the intensities temporarily.
- Video controller reads from the frame buffer and plots the screen pixels.

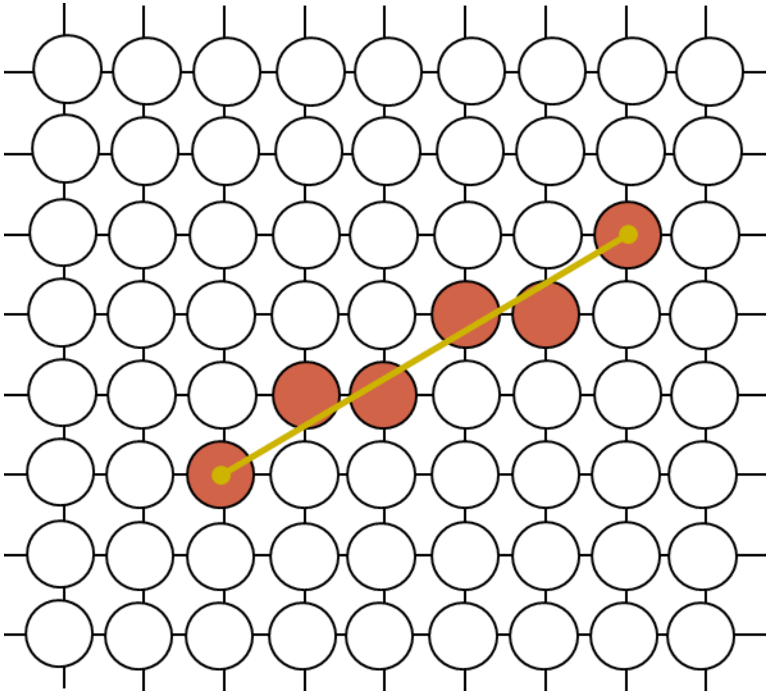
# Digital Representation(cont.)

- A line segment in a scene is defined by the coordinate positions of the line end-points.



# Digital Representation(cont.)

- But what happens when we try to draw this on a pixel based display?



- How do we choose which pixels to turn on?

# Scan Conversion

- The process of representing continuous graphics object as a collection of discrete pixels is called Scan Conversion.
- For e.g a line is defined by its two end points & the line equation, where as a circle is defined by its radius, center position & circle equation.
- It is the responsibility of graphics system or the application program to convert each primitive from its geometric definition into a set of pixels that make up the primitive in image space. This conversion task is generally referred to as a scan conversion or rasterization.

# Scan Conversion(cont.)

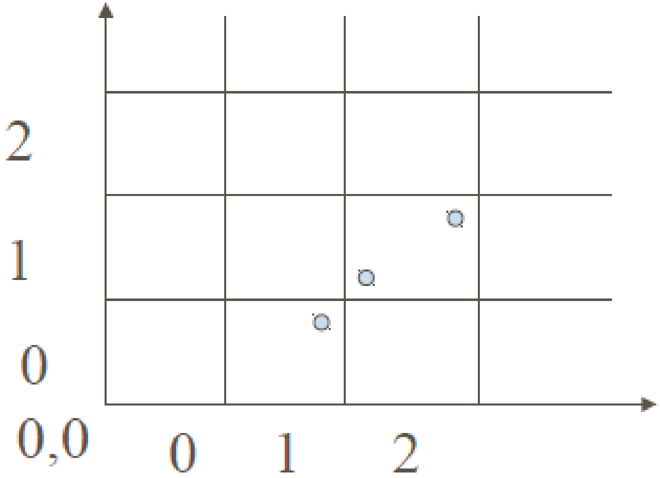
- Drawing lines, circles, and etc. on a grid implicitly involves approximation.
- Ideally, the following properties should be considered
  - smooth
  - continuous
  - pass through specified points
  - uniform brightness
  - efficient

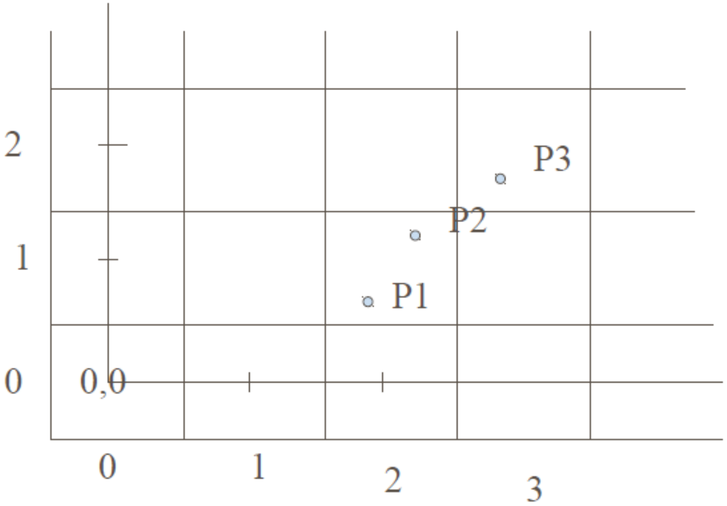
# Scan Converting a Point

- A mathematical point (x,y) where x & y are real numbers within an image area, needs to be scan-converted to a pixel at location (x',y').
- Can be done by making x' & y' to be the integer part of x & y.
  - x'= Floor(x) and y'= Floor(y)
  - P1(1.7,0.8) is represented by pixel (1,0)
  - P2(2.2,1.3) and P3(2.8,1.9) are both represented by pixel (2,1).

# Scan Converting a Point(cont.)

- Another approach is to scan convert (x,y) by making
  - x’ = Floor(x + 0.5) and y’ =Floor(y+0.5).
  - Points P1 and P2 are now both represented by pixel (2,1) whereas point P3 is represented by pixel (3,2).
- This essentially places the origin of the coordinate system for (x,y) at the center of pixel(0,0).





# Scan Converting a Line

- A line is defined by its two end points & the slope intercept equation for a line:
  - y = mx + b, m = Slope of the line, b = the y intercept of a line
- Line drawing is done by:
  - Calculating intermediate positions between the endpoints.
  - Directing the output device to fill in the calculated positions as in the case of plotting single points.

# Scan Converting a Line(cont.)

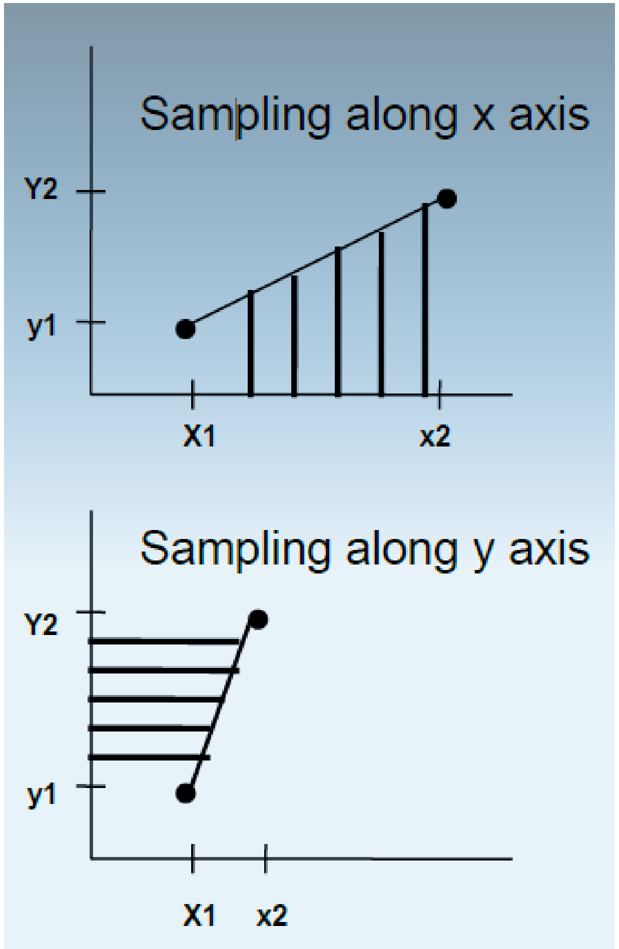
- Plotted positions may be only approximations to the actual line positions between endpoints.
  - A computed position (10.48, 20.51) is converted to pixel (10,21).
- This rounding causes the lines to be displayed with a stairstep appearance.
- Stairsteps are noticeable in low resolution systems, it can be improved by:
  - Displaying lines on high resolution systems.
  - Adjusting intensities along line path.

# Scan Converting a Line(cont.)

- The Cartesian intercept equation for a straight line:
  - $y= m. x +b$
- For line segment starting in (x1,y1) and ending in (x2,y2), the slop is:
  - m= (y1-y2)/(x1-x2)
  - b= y1- m.x1
- For any given x interval Δx, we can compute the corresponding y interval Δy:
  - Δy= m . Δx
- Or x interval Δx from a given Δy:
  - Δx= Δy/m

# Scan Converting a Line(cont.)

- On raster systems, lines are plotted with pixels, and step sizes in the horizontal and vertical directions are constrained by pixel separations.
- Scan conversion process samples a line at discrete positions and determine the nearest pixel to the line at each sampled position. (Incremental Fashion)



# Line Drawing - Algorithm 1

- A Straightforward Implementation

```
DrawLine(int x1,int y1, int x2,int y2, int color)
{
    float y;
    int x;

    for (x=x1; x<=x2; x++)
    {
         y = y1 + (x-x1)*(y2-y1)/(x2-x1)
         WritePixel(x, Round(y), color );
    }
}
```

# Line Drawing - Algorithm 2

- A Better Implementation

```
DrawLine(int x1,int y1,int x2,int y2, int color)
{
    float m,y;
    int dx,dy,x;
    dx = x2 - x1;
    dy = y2 - y1;
    m = dy/dx;
    y = y1 + 0.5;
    for (x=x1; x<=x2; x++)
    {
        WritePixel(x, Floor(y), color );
        y = y + m;
    }
}
```

# Line Drawing Algorithm Comparison

- Advantages over Algorithm 1
  - eliminates multiplication
  - improves speed
- Disadvantages
  - round-off error builds up
  - get pixel drift
  - rounding and floating point arithmetic still time consuming
  - works well only for $|m| < 1$
  - need to loop in y for $|m| > 1$
  - need to handle special cases

# Reference

- **Computer Graphics**
  - R. Plastock, Zhigang Xiang
  - (Schaum’s Outline Series)  McGraw Hill