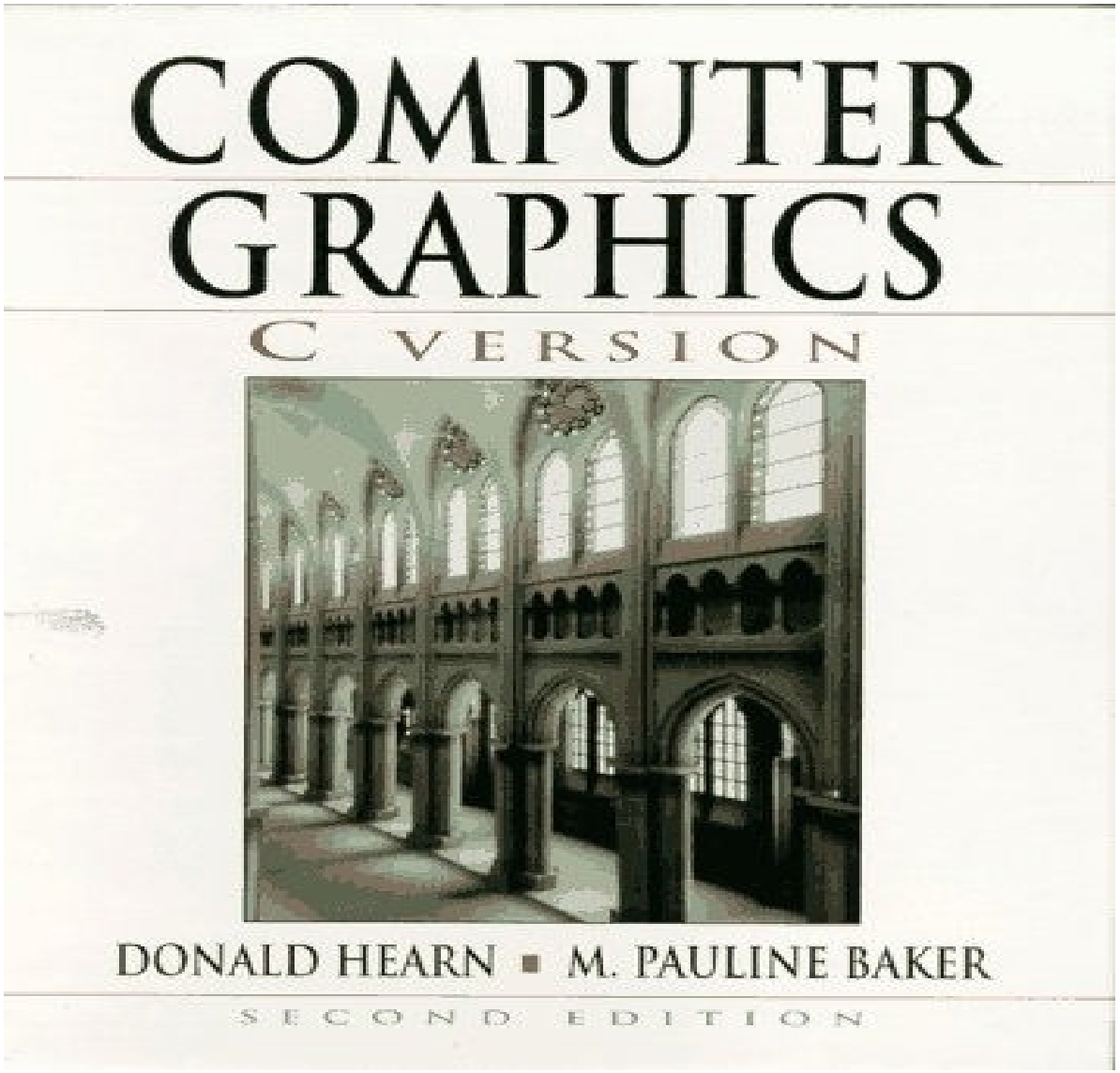
TEXT BOOK


COMPUTER
GRAPHICS
C VERSION
DONALD HEARN ■ M. PAULINE BAKER
SECOND EDITION

What is Computer Graphics??

- term **computer graphics** was coined in 1960 by William Fetter

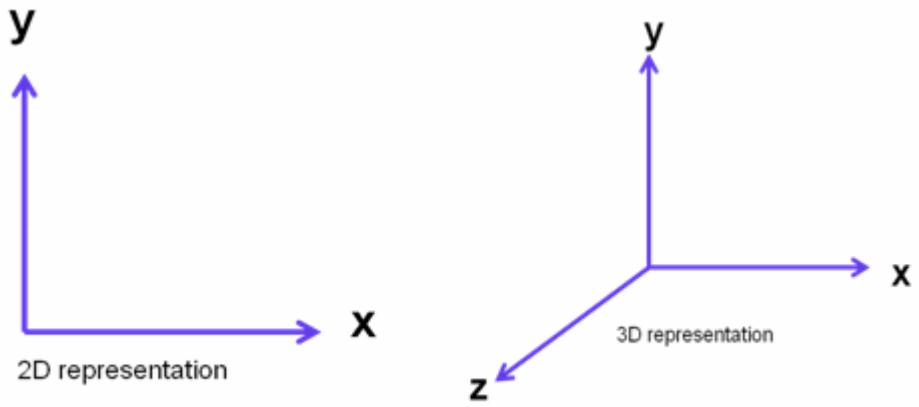
- *"Perhaps the best way to define computer graphics is to find out what it is not. It is not a machine. It is not a computer, nor a group of computer programs. It is not the know-how of a graphic designer, a programmer, a writer, a motion picture specialist, or a reproduction specialist.*

- *Computer graphics is all these – a consciously managed and documented technology directed toward communicating information accurately and descriptively."*

Computer Graphics, by William A. Fetter, 1966

COMPUTER GRAPHICS ? Contd..

- A field related to the generation of graphics using computer
- Includes the creation ,storage and manipulation of images of objects
- Objects→ Entertainment and Advertisement, Medicine, Physical, Mathematical, Engineering, Architecture
- Computer graphics- generating 2D images of a 3D world represented in a computer



MAIN TASKS

- Imaging : formation of an image
  : representation of 2D images

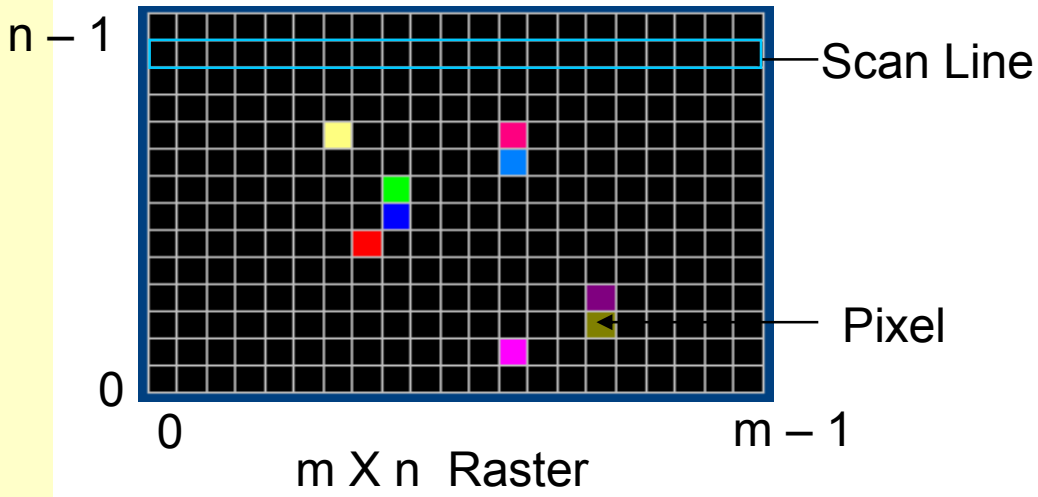
- Modeling: representing 3D images
  - creating and representing the geometry of objects in the 3D world

- Rendering : constructing 2d images from 3D models
  - generating 2D images of the objects

- Animation : simulating changes over time
  - describing how objects change in time

BASIC CONCEPTS

- **RASTER :** A rectangular array of points or dots.
- **PIXEL** (picture element) **:**One dot or picture element of the raster
- **SCAN LINE :** A row of pixels
- **BITMAP :**ones and zeros representation of the rectangular array points on screen
  - Black and white → bitmap
  - pixmap→ color (colored raster image)



We don't have any pixel like (1.2, 5.8). In raster device co-ordinates can take integer values only.

DEVELOPMENT OF COMPUTER GRAPHICS

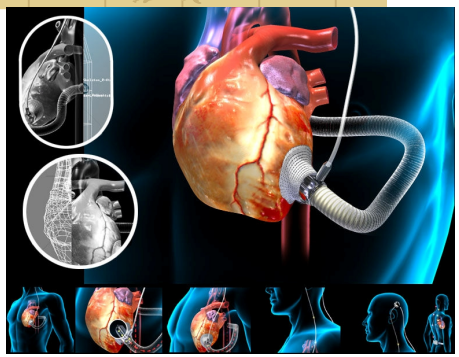
- 1951- whirlwind computer developed in MIT had computer driven CRT displays for output
- Mid 1950's- SAGE air defense system uses commands and control CRT on which operator targets with light pens. Light pens sense light emitted by objects on the screen.
- 1962- sketchpad, by Ivan Sutherland
  - Keyboard and light pen were used for drawing and making choices
  - Modern interactive graphics
- 1964- CAD and CAM-shows graphical interaction and graphical activities
  - General Motor DAC, Itek, Digitek for Lens Design
- 1968- Evans and Sutherland
  - Commercial company- flight simulators
  - High cost of graphics hardware

DEVELOPMENT OF COMPUTER GRAPHICS contd..

- 1971-Gouraud Shading- rendering method and is faster then phong shading
- 1974-1977 : Phong Shading
  - Computer graphics at NYIT –Computer animation raster graphics
- 1982- Ray Tracing (illumination based rendering method)
- 1993-Open GL- Open Graphics Library
- 1995- Microsoft, game playing API

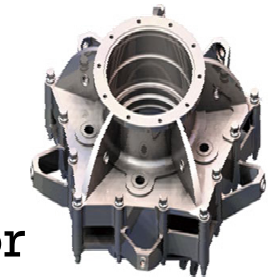
APPLICATIONS OF COMPUTER GRAPHICS

- Entertainment
  - Films-Jurasic Park
  - Video games, Formula 1 etc

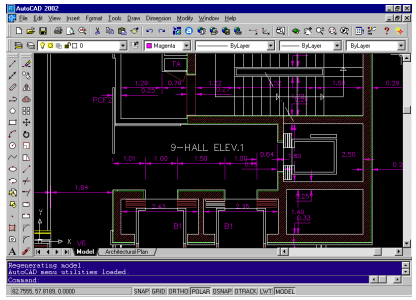


- Computer-aided design
  - Design components and system of mechanical, electrical, electronics, buildings, automobile etc
  - aeroplane

Medical Image



- Scientific Visualization and Business Visualization
  - Scientific visualization-generating computer graphics for scientific, engineering ,medical data
    - E.g. airflow inside a thunderstorm
    - Visible human
  - Business visualization- non scientific data obtained in economics

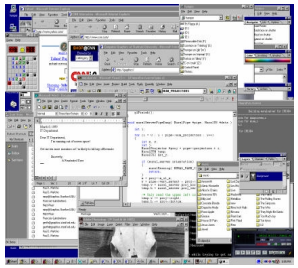


AutoCAD 2002

APPLICATIONS OF COMPUTER GRAPHICS contd…

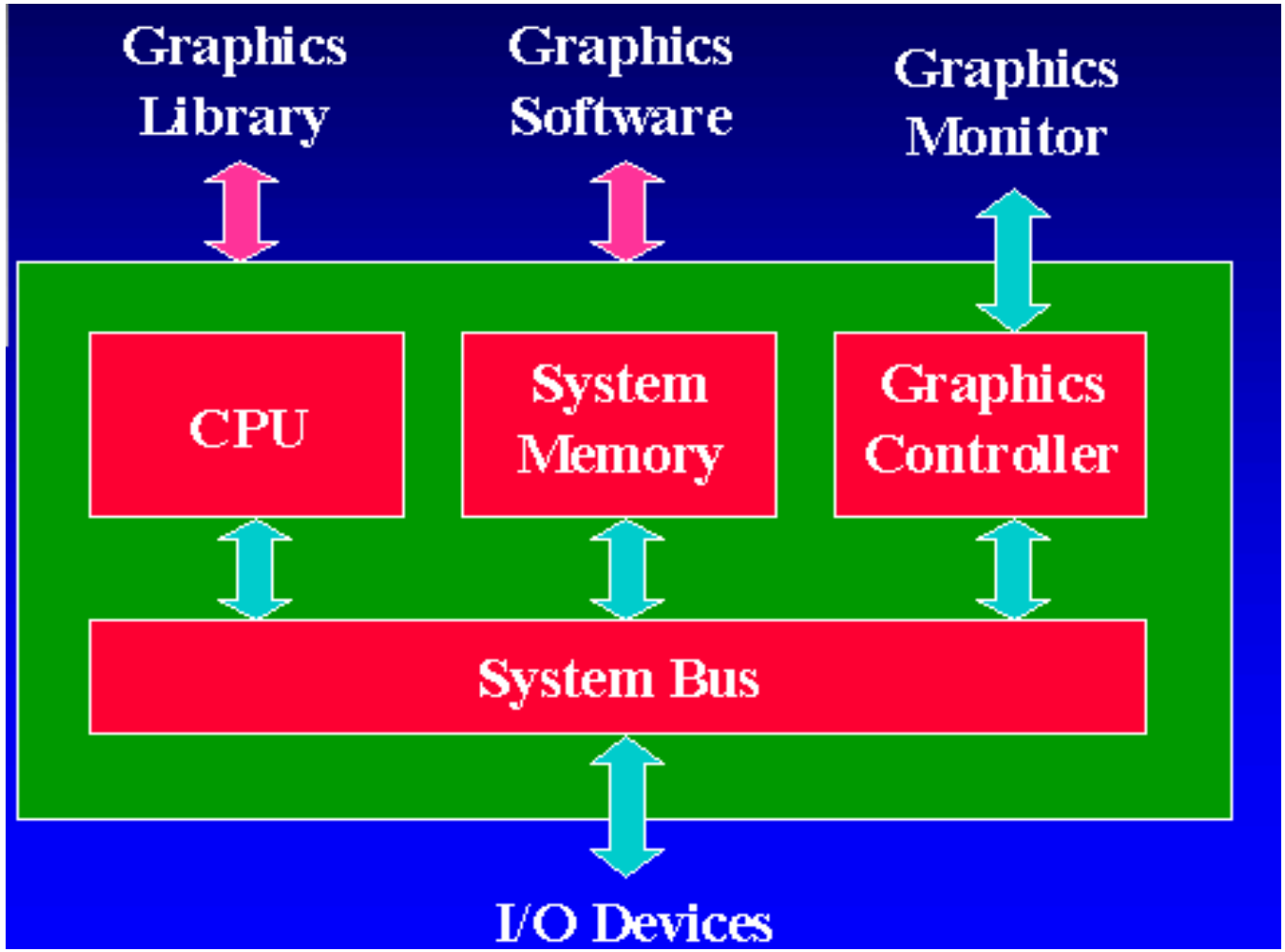
- Training
  - Driving Simulation
  - Flight Simulation
  - Desk Assembly
- Education
  - Human skeleton
  - 3D MAX
- E-commerce
  - Virtual phone store
  - Interactive kitchen planner
  - charts
- Computer Art
  - maps
- Office automation and Electronics Publishing
  - Desktop publishing

Flight Simulator



CHAPTER 2- HARDWARE CONCEPTS

Basic Graphics System



VIDEO DISPLAY DEVICE

• capable of performing a high-speed switching process and preventing of video flicker form occurring.

• comprises a plurality of display unit including numbers of unit cells.

• unit cell comprises of numerous dots arrange vertically and horizontally in orderly manner.

• Each dot is comprised of numerous display elements.

TERMINOLOGIES

1.Florescence/ phosphorescence
2.Persistence
3.Refresh Rate
4.Horizontal Scan Rate
5.Resolution
6.Aspect Ratio

FLUORESCENCE / PHOSPHORESCENCE

When the beam of electrons emitted by electron gun is strikes phosphor-coated screen of the CRT and then phosphor emits a small spot of light at each position contacted by the electron beam, such phenomenon is known as Fluorescence / Phosphorescence which last just a fraction of microseconds.

PERSISTENCE

-defined as the time from the removal of excitation to the moment when phosphorescence has decayed to 10 percent of the initial light output .

-The phosphor used for graphics display device usually have persistence of 10 to 60 microseconds.

RESOLUTION

Resolution is defined as the maximum member of points that can be displayed horizontally and vertically without overlap on a display device.

ASPECT RATIO

Aspect Ratio gives that ratio of vertical point to horizontal points necessary to produce equal length lines in both directions on the screen an aspect ratio of 3/4 means that a vertical line plotted with 3 points has the same length as a horizontal line plotted with 4 points.

HORIZONTAL SCAN RATE

The horizontal Scan Rate is the number of scan lines per seconds.

The rate is approximately the product of the refresh rate and number of scan lines.

REFRESH RATE

- The number of times the screen is redrawn each second.
- Higher refresh rates mean less flicker on the screen, which translates into less eyestrain.
- As refresh rate decreases , flicker develops because eye can no longer integrate the individual light impulses coming from pixel.
- The refresh rate above which a picture stops flickering and fuses into a steady image is called the critical fusion frequency.

FACTORS AFFECTING CRITICAL FUSION FREQUENCY

- The longer the persistence, the lower the CFF.
- Increasing the image intensity increases the CFF
- Decreasing the room light increases the CFF
- Wavelengths of emitted light
- Observer

Video display device types

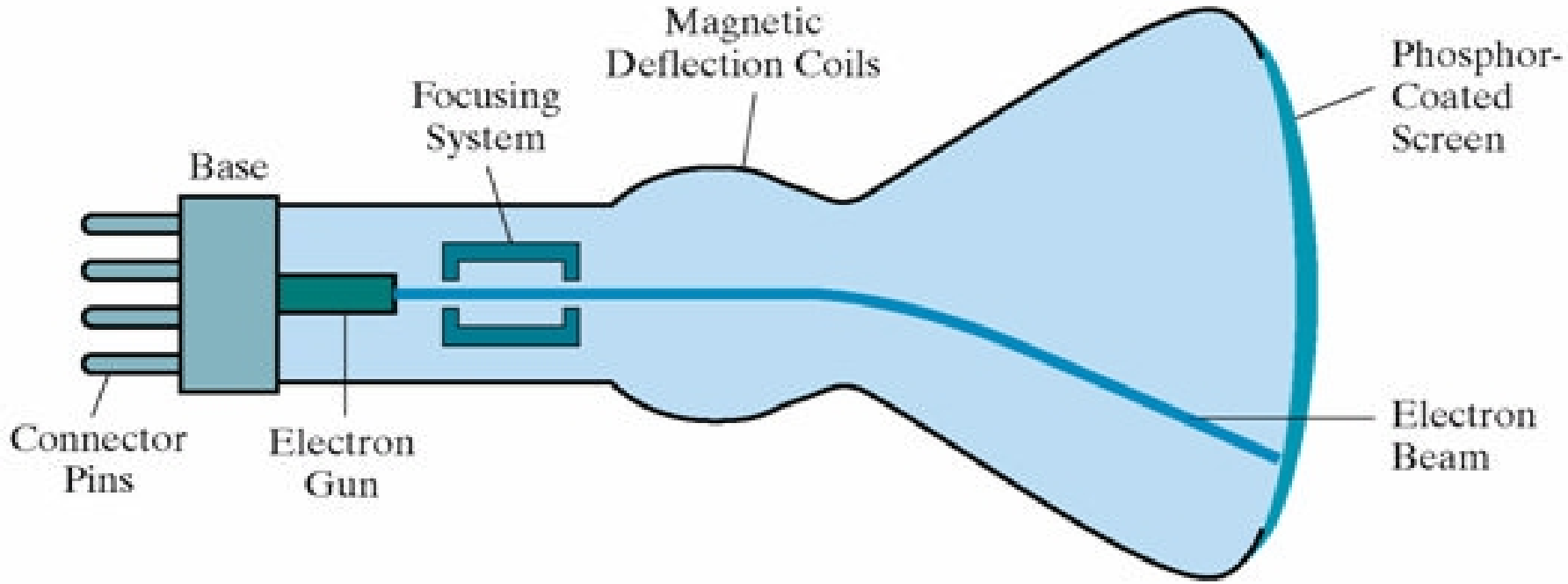
(i) Cathode Ray tube Display (CRT)

(ii) Liquid Crystal Display (LCD)

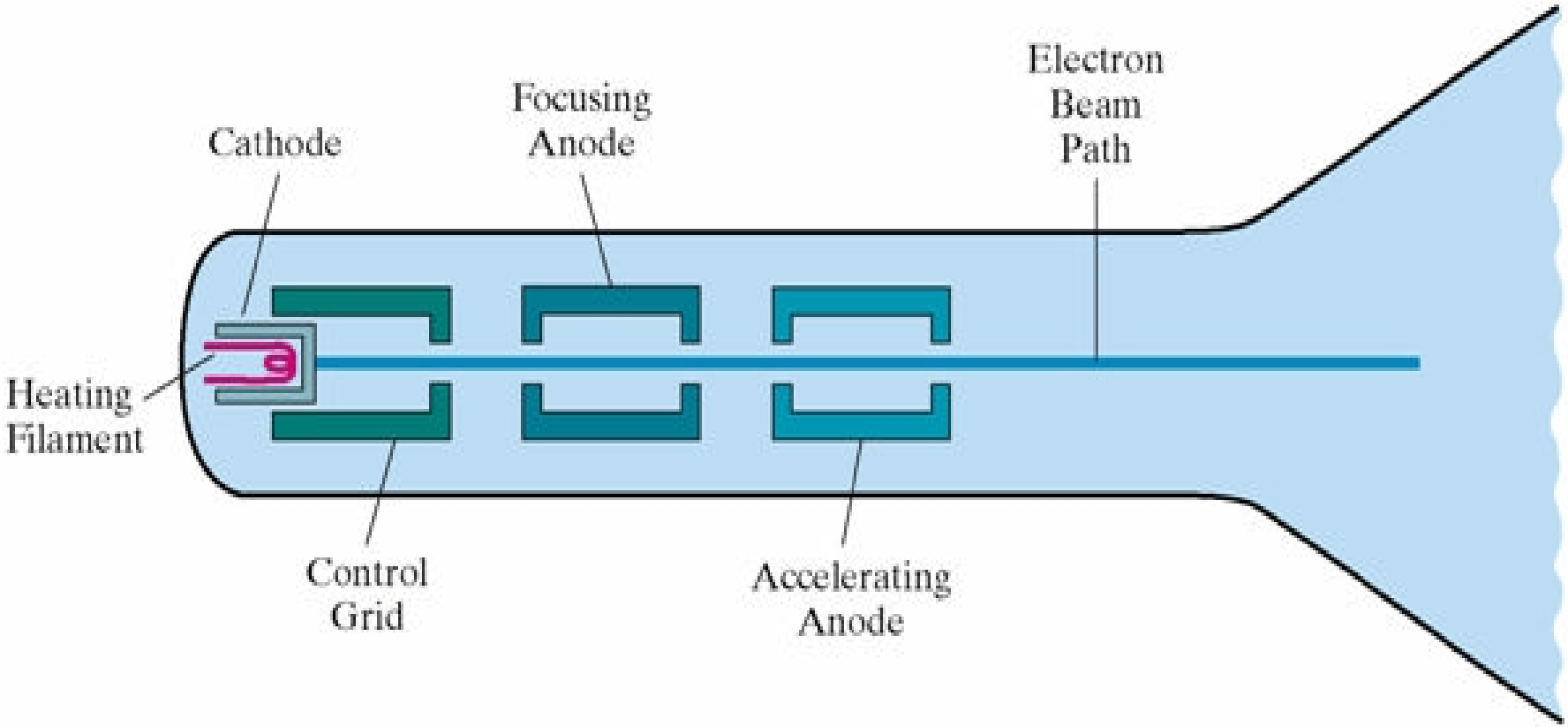
(iii) Raster Scan Display

(iv) Random Scan Display (Vector Scan Display)

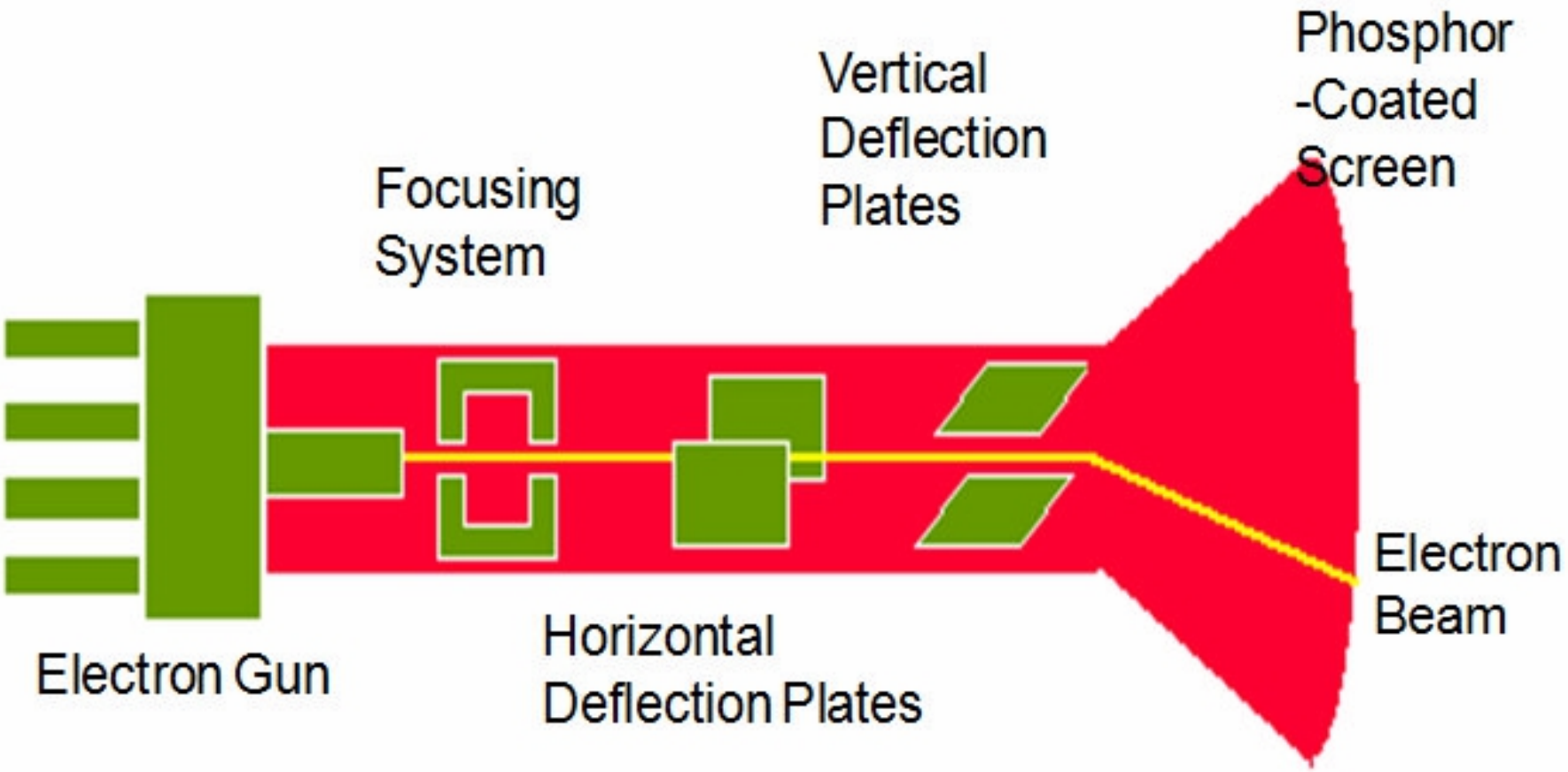
REFRESH CATHODE-RAY TUBES



Magnetic- deflection CRT



Operation of an electron gun with an accelerating anode



Electrostatic Deflection of Electron Beam in CRT

Refresh Cathode-Ray Tube

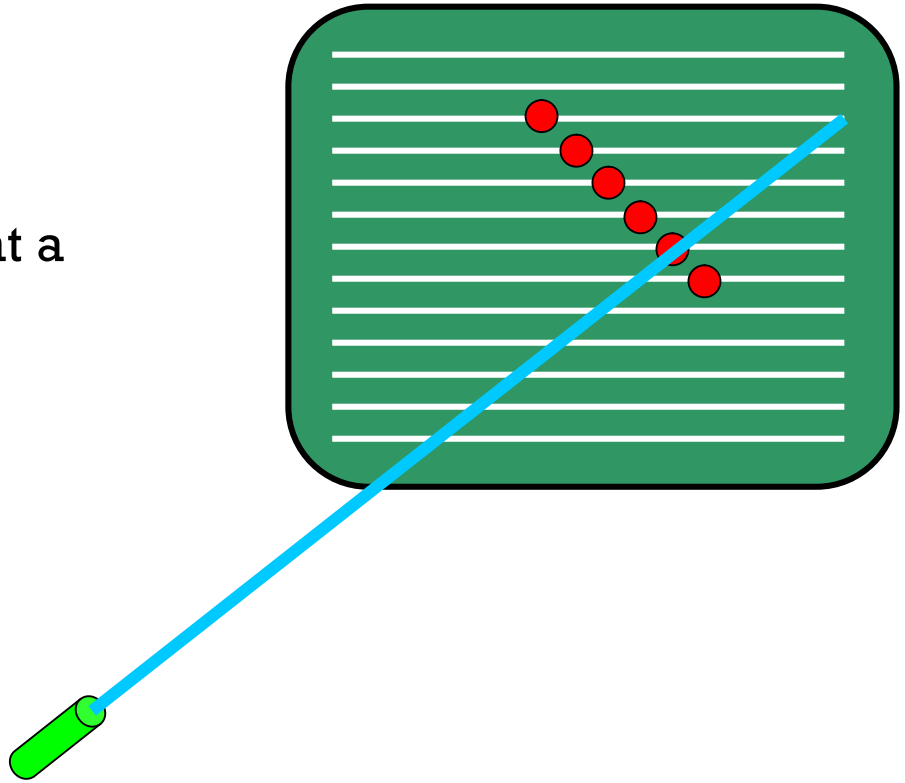
- **Focusing system**
  - Electrostatic focusing
  - Positively charged metal cylinder
- **Deflection of electron beam**
- **Persistence**
  - How long phosphors continue to emit light
  - Low persistence is useful for animation
- **Resolution**
  - Number of points per centimeter can be plotted horizontally and vertically

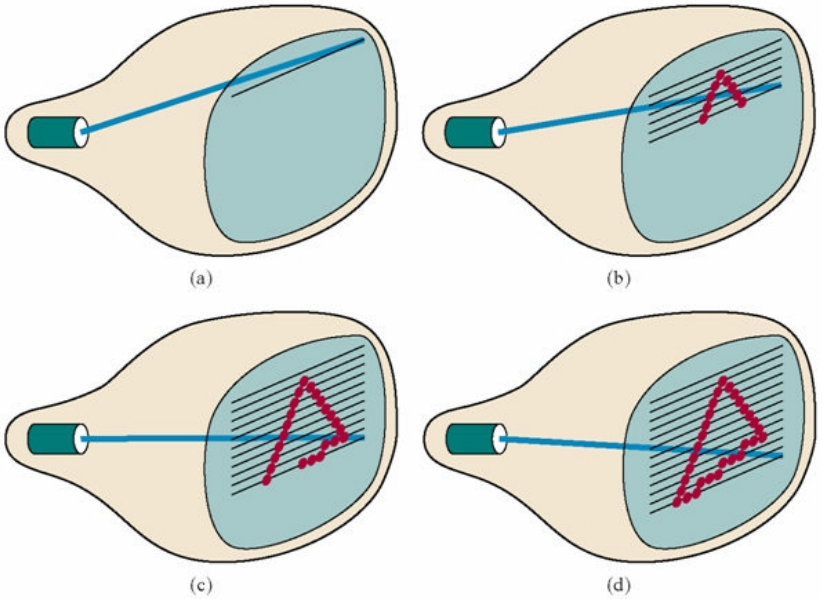
Refresh Cathode-Ray Tube

- **Intensity distribution of a phosphor spot**
  - Adjacent spots appear distinct: 60%
  - Spot size also depends on intensity
- **Deflection of electron beam**
- **Resolution of a CRT depends**
  - Type of phosphor
  - Intensity to be displayed
  - Focusing and deflection system
- **Aspect ratio**
  - The ratio of vertical points to horizontal points necessary to produce equal-length lines

Raster-Scan Display

The electron beam is swept across the screen, one row at a time from top to bottom



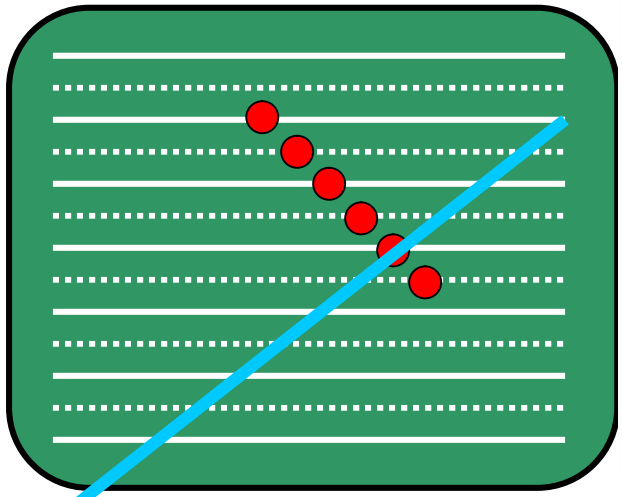


raster scan system

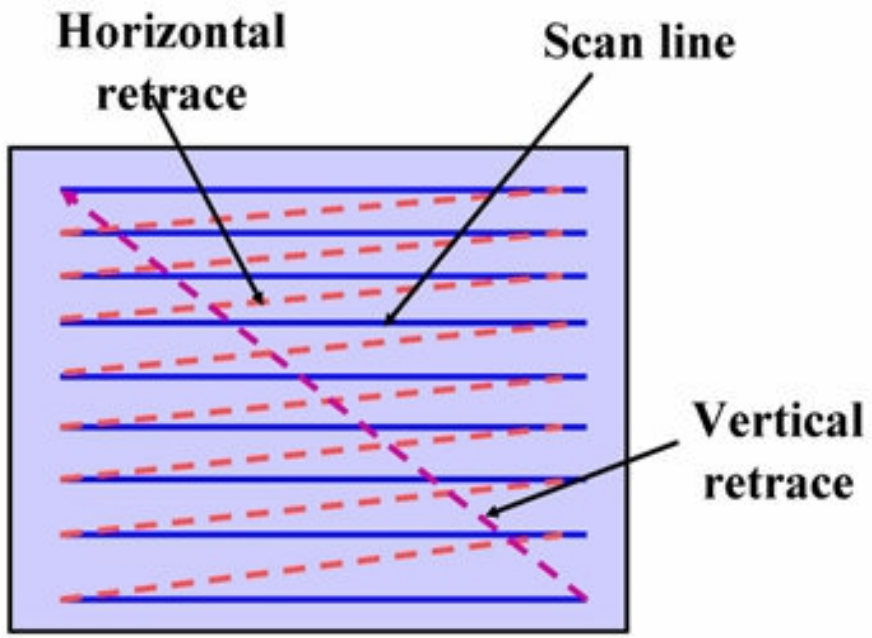
Raster-Scan Display

- Frame buffer (refresh buffer)
  Picture definition is stored in the memory area
- Pixel Picture each screen point is referred as pixel or pel
- Scan line
- Pixmap (bitmap) Picture for black-and-white system with one bit per pixel, the frame buffer is called a bitmap. For systems with multiple bits per pixel the frame buffer is called a pixmap
- Refreshing (60 - 80 frames/sec)
- Horizontal retrace
- Vertical retrace
- Interlacing each frame is displayed in two pass
- Scan conversion

Interlaced refresh



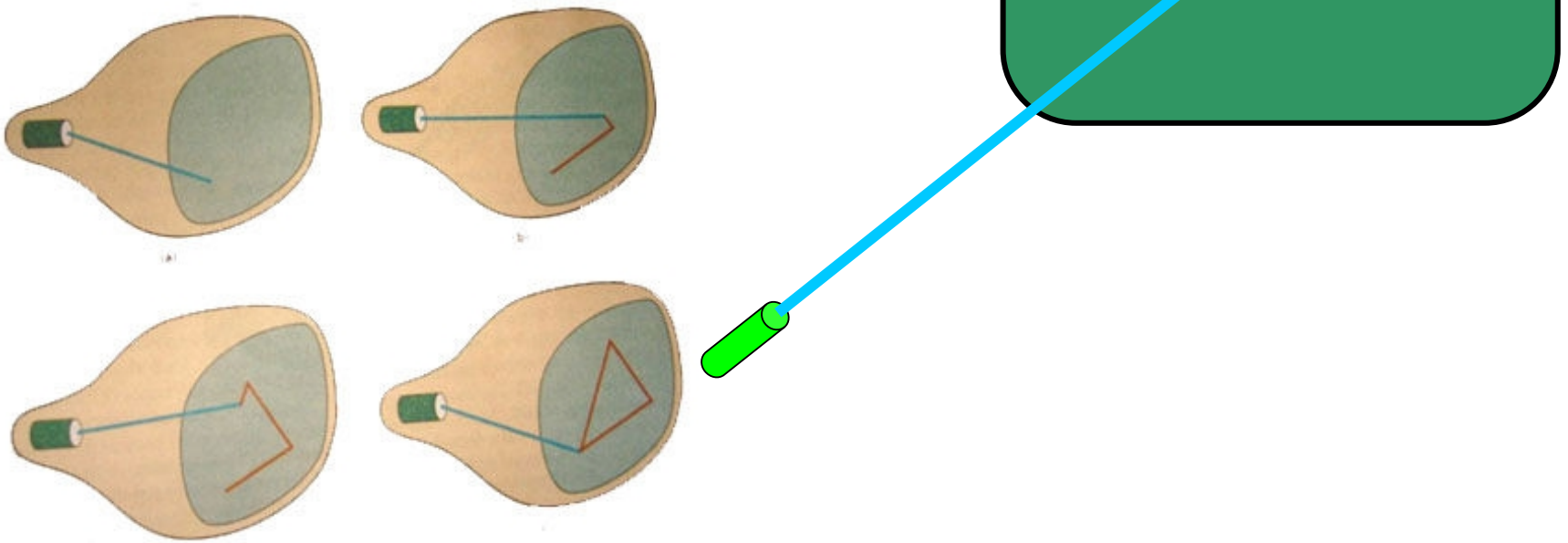
Raster Scan



Random-Scan Display

A CRT has the electron beam directed only to the parts of the screen where the picture is to be drawn

draw a picture one line.



Random-Scan Display

- Random-scan (vector, stroke-writing, calligraphic) displays
  - Draw a picture one line at a time
- Refresh rate
  - Depends on the number of lines to be displayed
- Picture definition
  - A set of line-drawing commands
  - Refresh display file
- Designed for line drawing applications
  - Can not display realistic shaded scenes

Color CRT Monitors

- **Beam-penetration method**
  - Used with random-scan monitors
  - Two layers of phosphor: red and green
  - The displayed color depends on how far the electron beam penetrates into the phosphor layers.
  - Only four colors are possible: red, green, orange, and yellow.

ADVANTAGE:

- Economical way to produce colors

LIMITATIONS:

- Generation of only four colors is possible
- Poor picture quality

Color CRT Monitors

- **Shadow-mask method**
  - Three phosphor color dots at each pixel position
  - Three electron guns
  - Color variations: Varying the intensity levels of the three electron beams
  - Full-color (true-color) system: 24 bits of storage per pixel

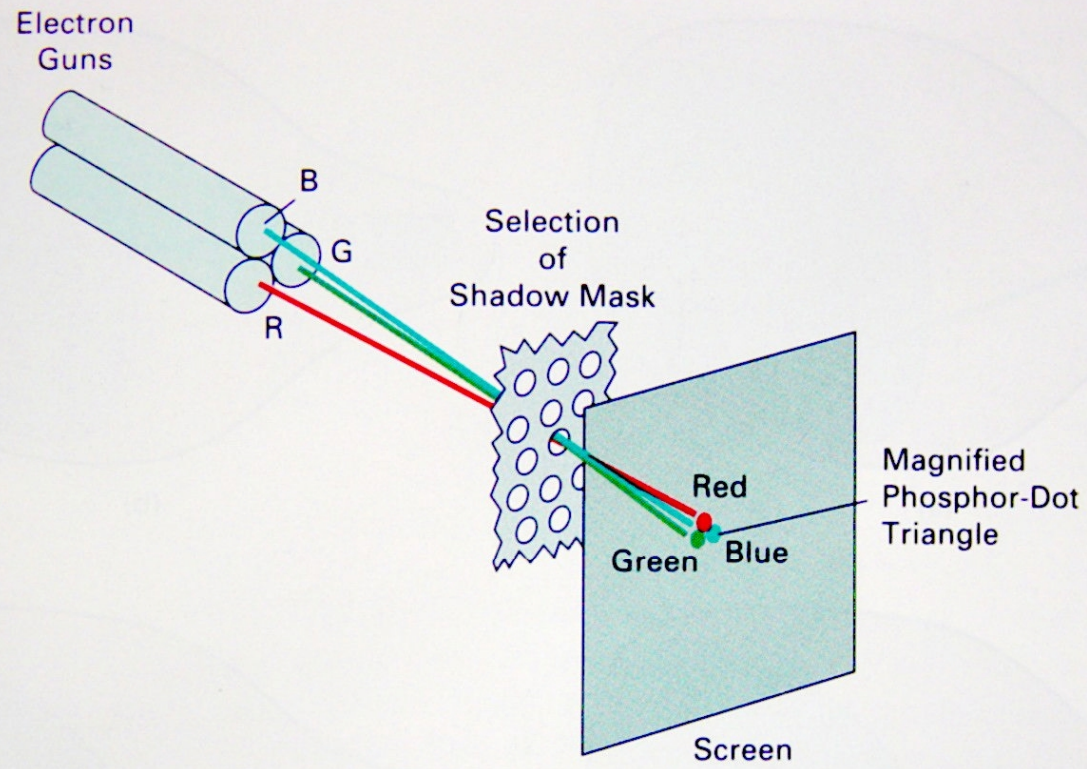


Figure 2-10

Operation of a delta-delta, shadow-mask CRT. Three electron guns, aligned with the triangular color-dot patterns on the screen, are directed to each dot triangle by a shadow mask.

Direct-View Storage Tubes

- **Store the picture information inside the CRT**
  - A charge distribution just behind the phosphor-coated screen
  - Two electron guns: primary gun and flood gun
- **Advantage**
  - Very complex pictures can be displayed at very high resolutions
- **Disadvantages**
  - Do not display color
  - Selected parts of a picture cannot be erased

Flat-Panel Displays

- **Video devices have reduced**
  - volume,
  - weight, and
  - power requirements
- **compared to a CRT.**
- **A significant feature - thinner than CRT**
- **Classified into two categories:**
  - Emissive displays
  - Nonemissive displays

LIQUID CRYSTAL DISPLAY (LCD)

- Commonly found on laptops
- Desktop versions exist
- Fluorescent lights provide illumination

TYPES

Passive matrix LCD

- Pixels arranged in a grid
- Pixels are activated indirectly
  - Row and column are activated
- Animation can be blurry

Active matrix LCD

Each pixel is activated directly

- Pixels have 4 transistors
  - One each for red, green, blue
  - One for opaqueness
- Transistors arranged in a thin film
- Animation is crisp and **clean**

Advantage

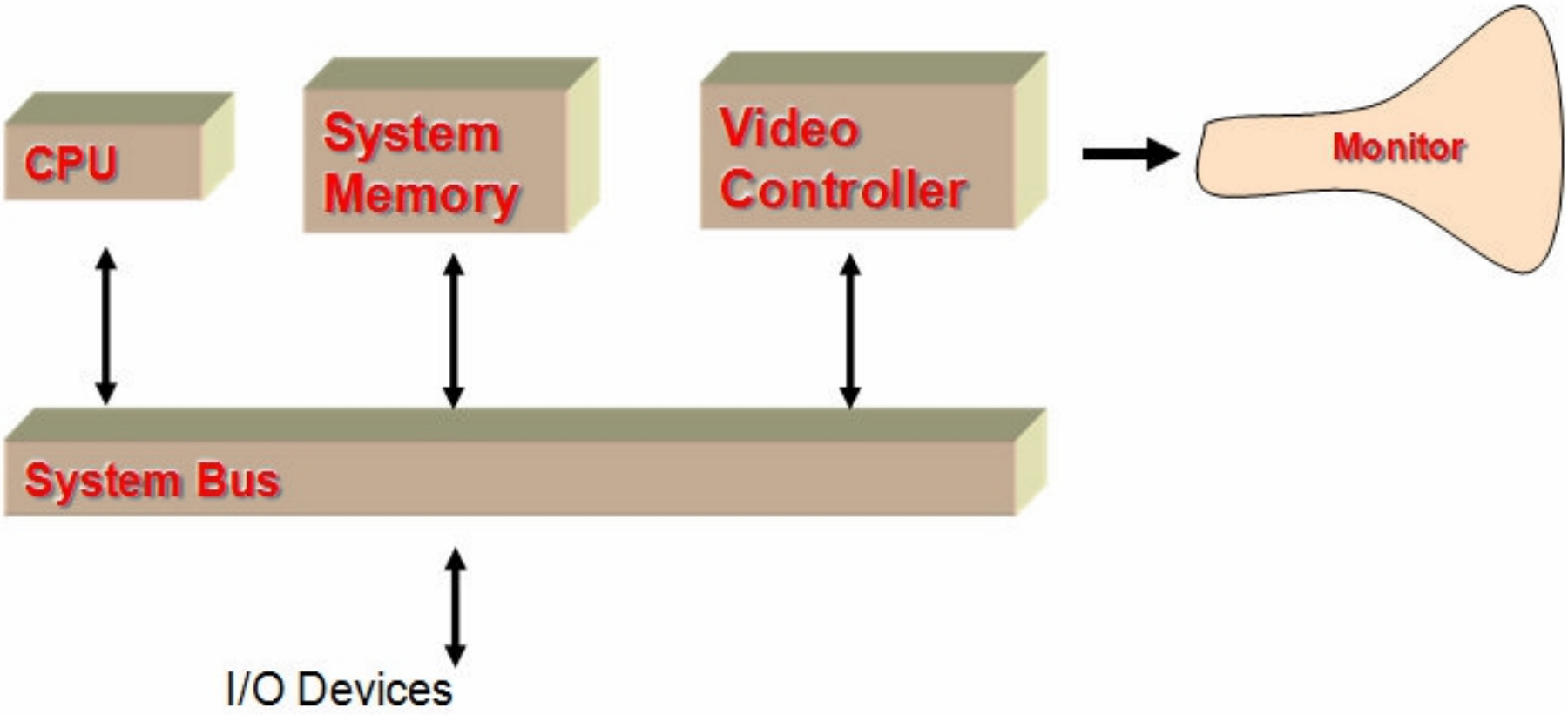
- **Physical Size**
- **Display size**
- Power Consumption
- Radiation Emission

Disadvantage

- Viewing angle
- price

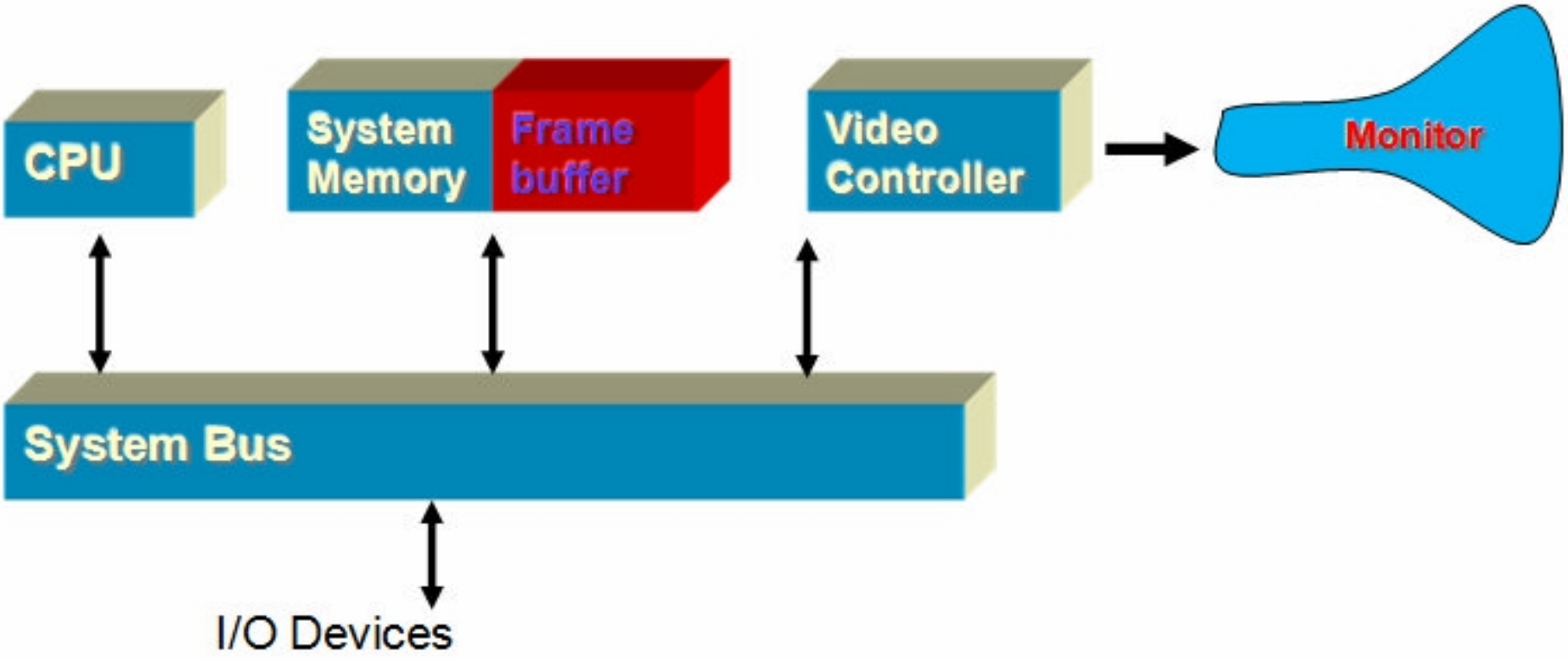
Raster-Scan Systems

Architecture of a simple raster graphics system

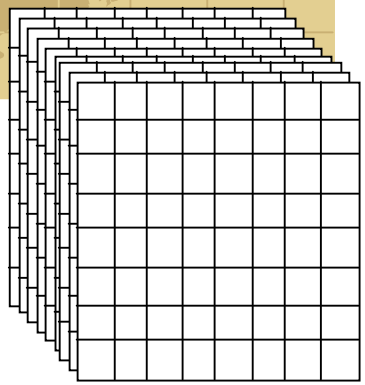


Raster-Scan Systems

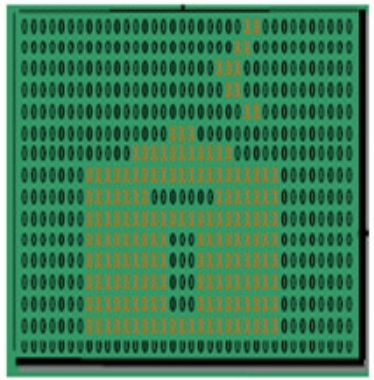
A fixed area of system reserved for frame buffer



Frame Buffer



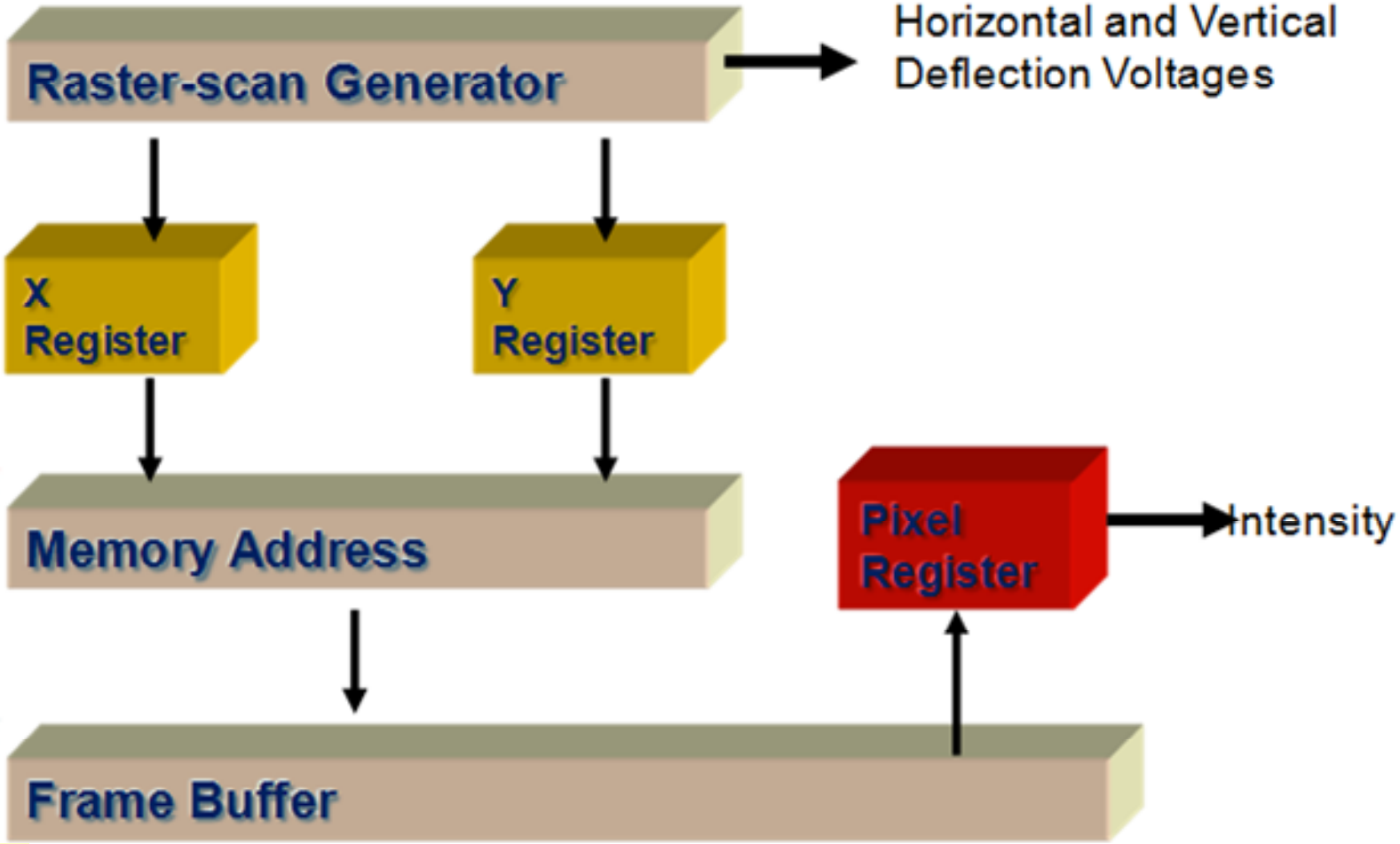
- The information in the buffer typically consists of color values for every pixel .
- A frame buffer may be thought of as computer memory organized as a two dimensional array . with each (x , y) addressable location corresponding to one pixel.
- Bit planes or bit depth is the number of bit corresponding to each pixel.



Video Controller

- **Coordinate system**
  - Frame-buffer locations are referenced in Cartesian coordinates.
  - Coordinate origin
    - Lower left screen corner
    - Upper left screen corner
- **Refresh operations of video controller**
  - Top-to-bottom, left-to right
  - x register (initial value = 0)
  - y register (initial value = y_{max})

Video Controller

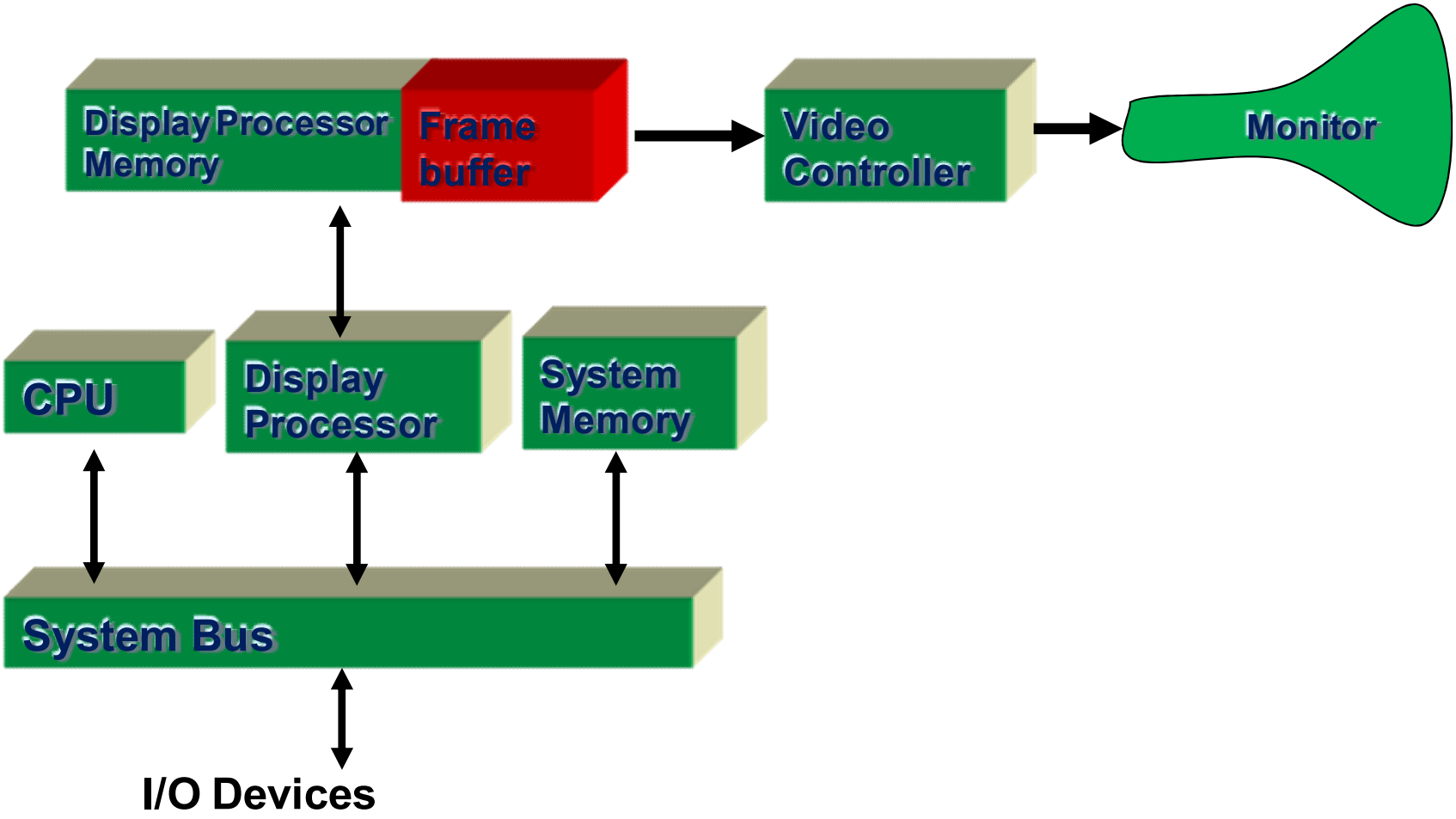


Basic video controller refresh operations

Raster-Scan Display Processor

- **Display processor (graphics controller, display coprocessor)**
  - To free the CPU from the graphics chores
  - Digitize a picture definition into a set of pixel-intensity values
- **Scan conversion: digitization process**
  - Straight-line segments
  - Curved lines and polygon outlines
  - Characters

Raster-Scan Display Processor



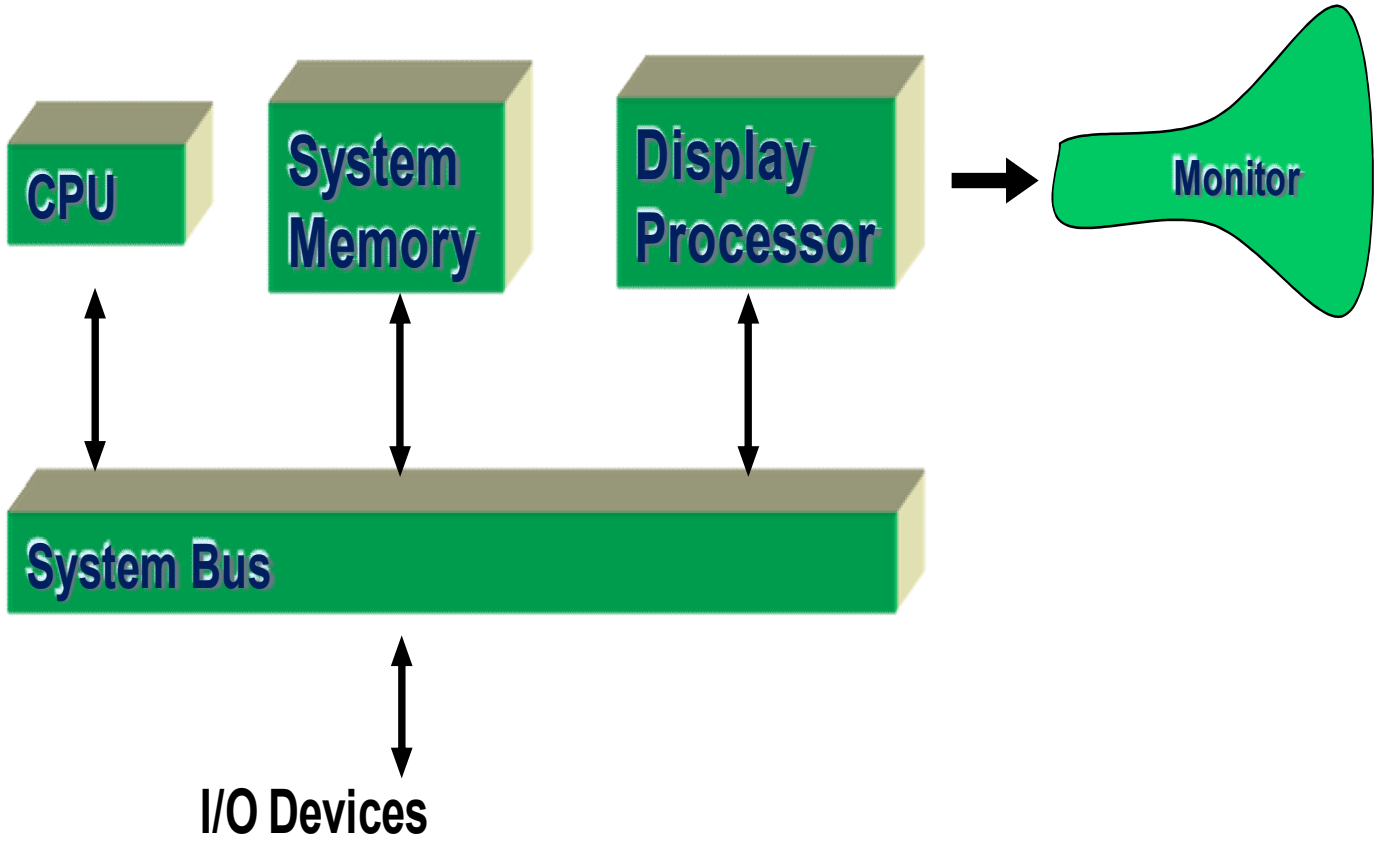
Architecture of a raster -graphics system with a display processor

Raster-Scan Display Processor

- **Display processor functions**
  - Generating various line styles
  - Displaying color areas
  - Performing certain transformations
  - Manipulations on display objects
- **Run-length encoding**
  - Reduce memory requirements
  - Organizing the frame buffer as a linked list
  - Store a scan line as a set of integer pairs
    - An intensity value
    - The number of adjacent pixels with the same intensity

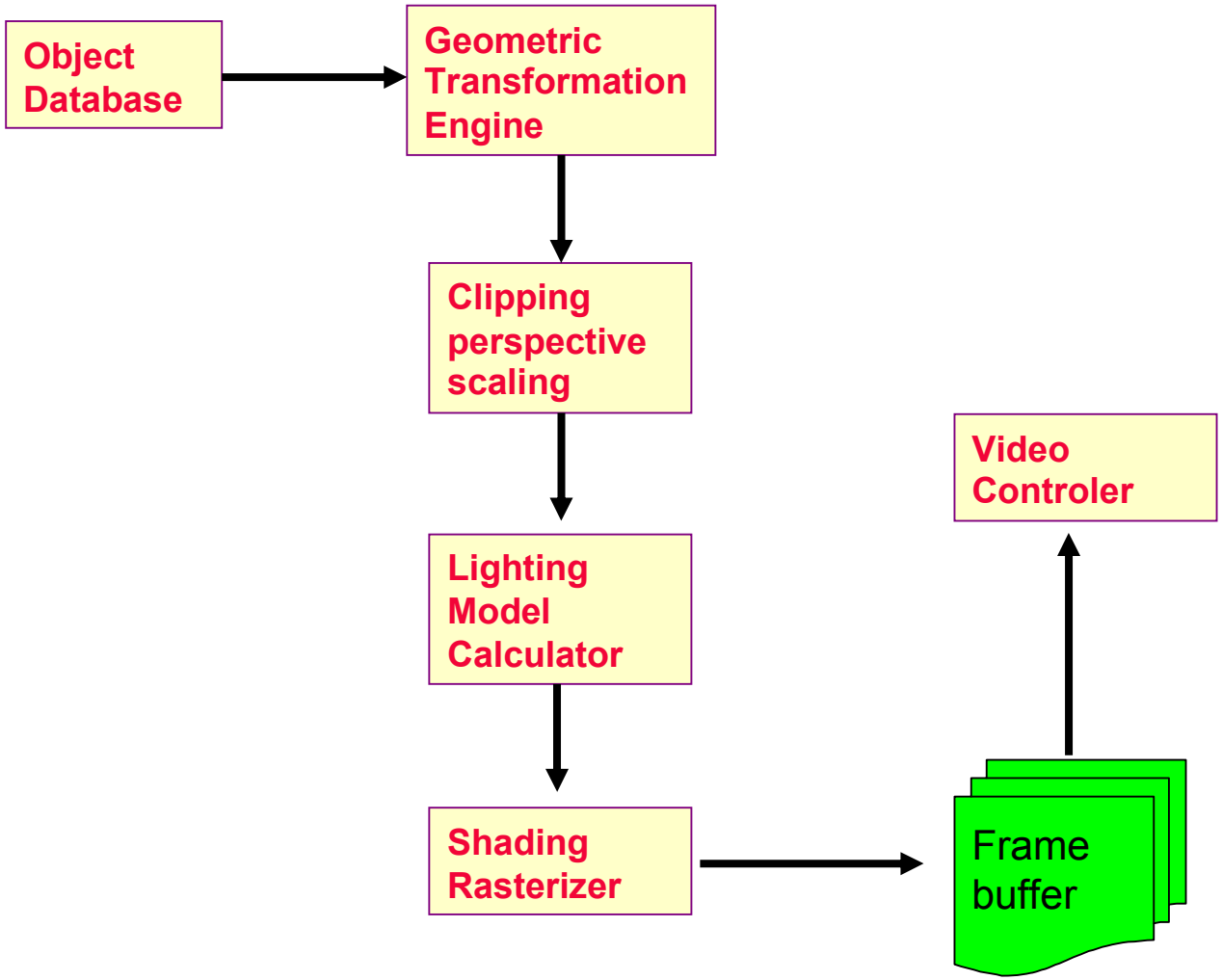
Random-Scan Systems

Graphics commands are translated into a display file stored in the system memory



Architecture of a simple random scan system

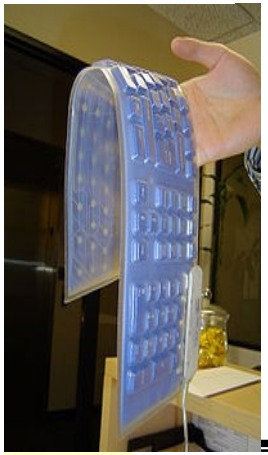
Advanced 3D Graphics Architecture



INPUT DEVICES

- piece of hardware used to enter and manipulate information on a computer

KEYBOARD



MOUSE

OPTICAL

MECHANICAL

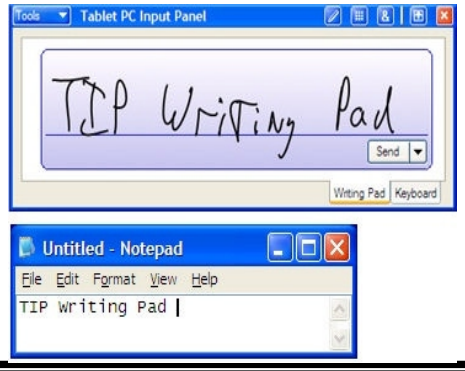
TABLET

1.No keyboard, no mouse. Instead, you have an LCD screen and a stylus.

2.You don't need to convert handwriting to text

Tablets are gaining popularity as a replacement for the computer mouse as a pointing device

Active tablets	use of electromagnetic induction technology,
Optical tablets	stylus used contains self-powered electronics
Passive tablets	small digital camera in the stylus
Acoustic tablets	sound generator was mounted in the stylus
Capacitive tablets	works by generating and detecting an electromagnetic signal
Electromagnetic tablets	use an electrostatic or capacitive signal



TOUCH PANELS

- Allow displayed objects or screen positions i.e. graphical icons to be selected with the touch of a finger.
- The key goals are to recognize one or more fingers touching a display, to interpret the command that this represents, and to communicate the command to the appropriate application.
- When a user touches the surface, the system records the change in the electrical current that flows through the display

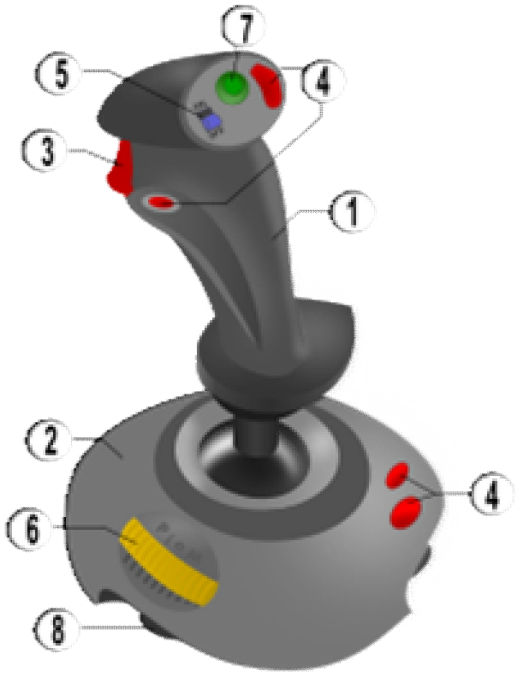
- **Optical touch panel** – employ LED's along horizontal and vertical edge of the frame, the interruption in the beam is recorded by the detectors
- **Electrical touch panel** – the voltage drop across the resistive plate to the conductive plate gives coordinate value
- **Acoustical touch panel** – sound waves generated across the glass if gets reflected by the touch of finger position and time is detected

LIGHT PEN

- a computer input device in the form of a light-sensitive wand used in conjunction with a computer's CRT TV set or monitor
- works by sensing the sudden small change in brightness of a point on the screen when the electron gun refreshes that spot
- Light pens have the advantage of 'drawing' directly onto the screen, but this can become uncomfortable, and they are not as accurate as digitizing tablets

JOYSTICK

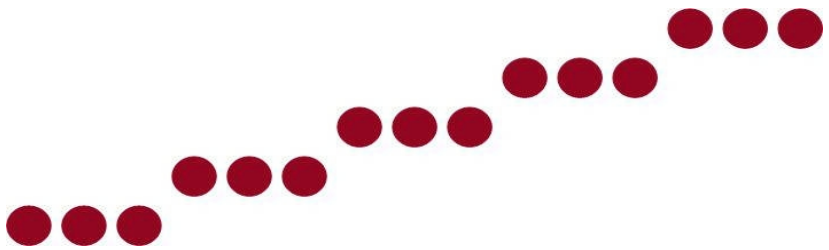
- consists of a stick that pivots on a base and reports its angle or direction to the device it is controlling



CHAPTER 3- TWO- DIMENSIONAL ALGORITHMS

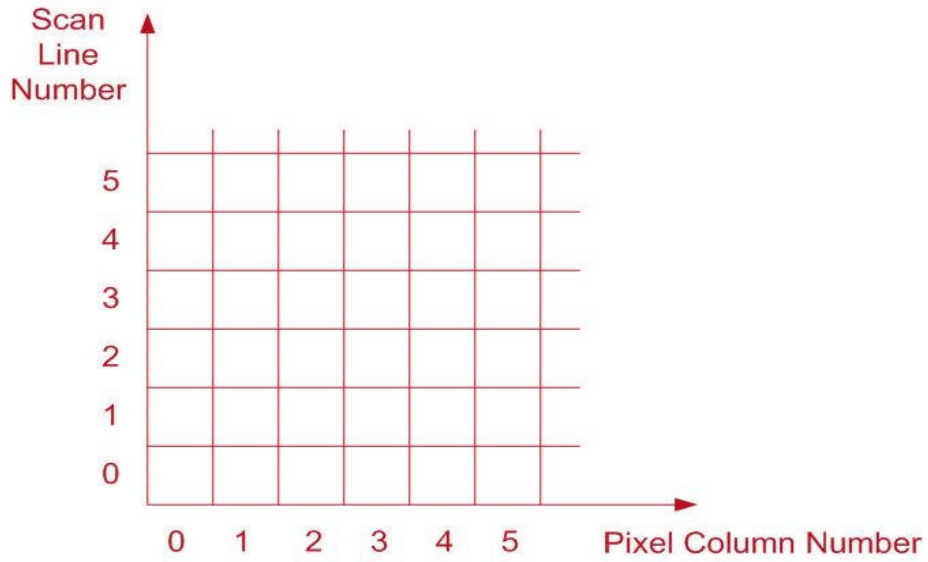
Points and Lines

- Points
  - Plotted by converting co-ordinate position to appropriate operations for the output device (e.g.: in CRT monitor, the electron beam is turned on to illuminate the screen phosphor at the selected location.)
  - Line
    - Plotted by calculating intermediate positions along the line path between two specified endpoint positions.
    - Screen locations are referenced with integer values, so plotted positions may only approximate actual line positions between two specified endpoints → "***the jaggies***". E.g.: position (10.48,20.51) → (10,21).



Pixel Position

- Pixel position: referenced by scan line number and column number



LINE DRAWING ALGORITHM

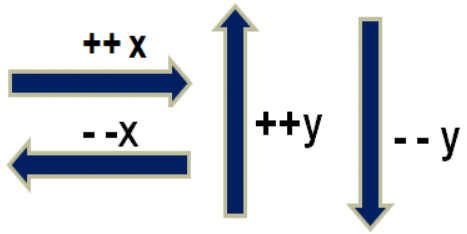
- DDA Algorithm- (Digital Differential Analyzer)
- Bresenham's Line Algorithm

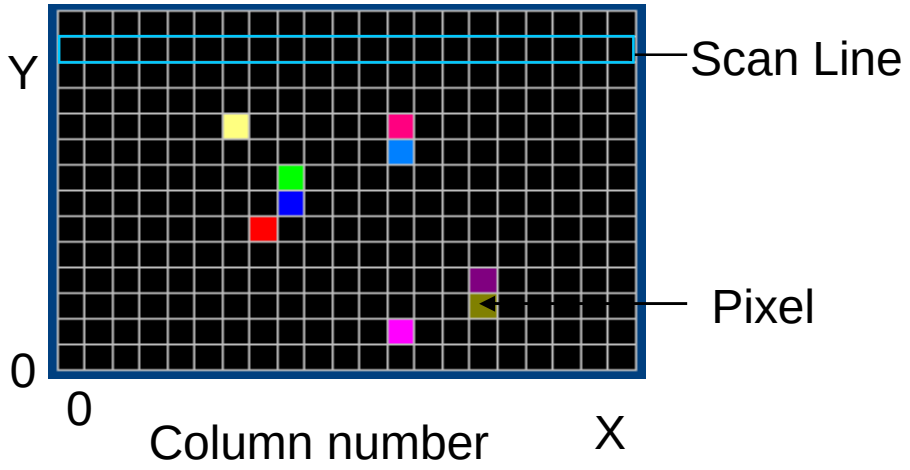
Note:

Pixel- picture element→ smallest piece of information in an image represented using dots, squares and rectangle

Components-

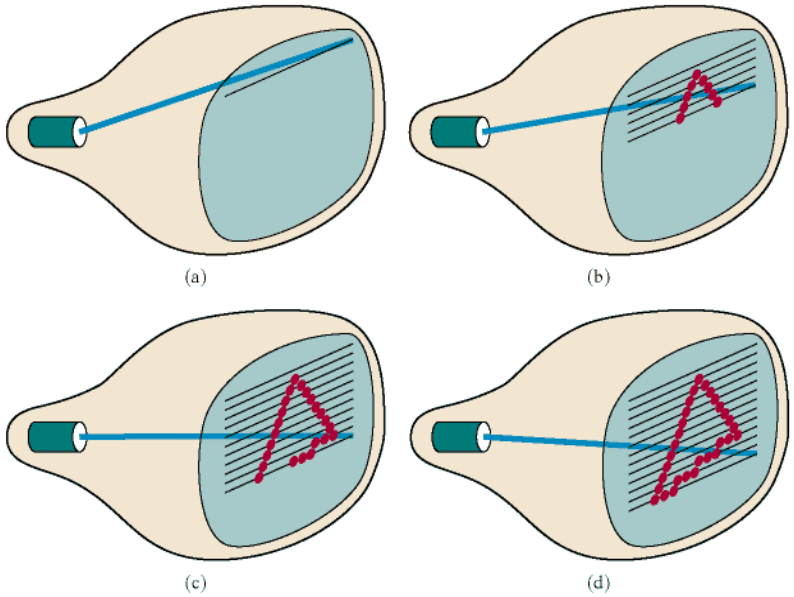
RGB
or
4 components
(cyan ,magenta,
yellow & black)



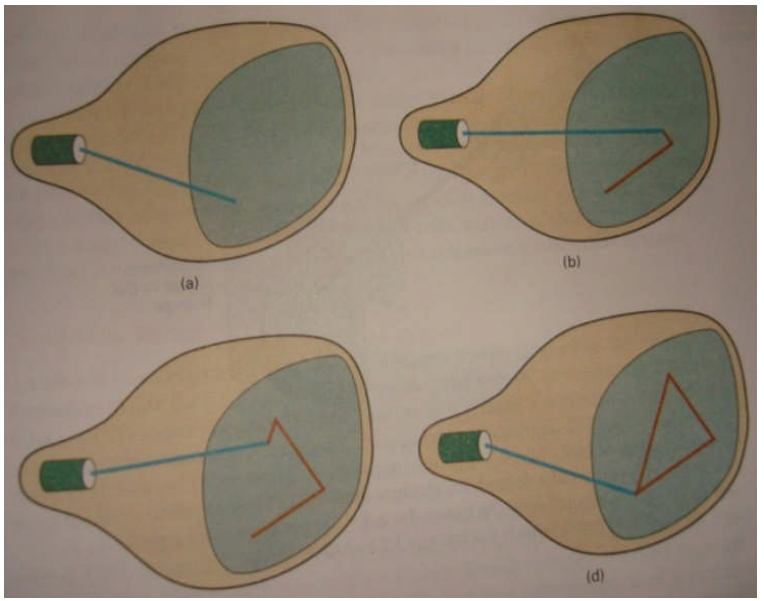


RASTER and RANDOM SCAN

- Raster Scan
  - Electron beam swept across the screen ,one row at a time from top to bottom. e.g. television



- Random or vector Scan
  - Electron beam directed only to parts of the screen where a picture is drawn. e.g. pen plotter



Straight Line Equation

- **Slope-Intercept Equation**

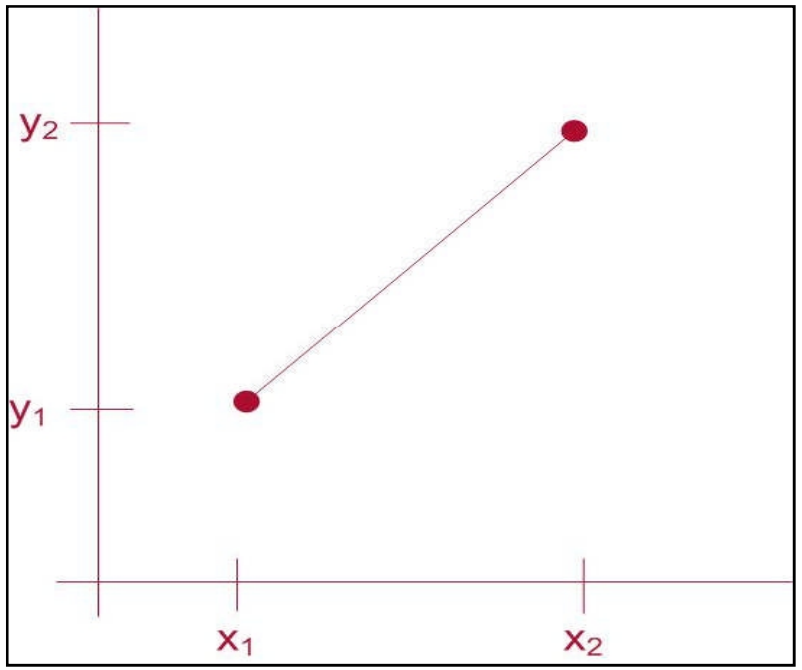
$$y = m.x + b$$

- **Slope**

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{rise}{run}$$

- **Intercept**

$$b = y_1 - m.x_1$$

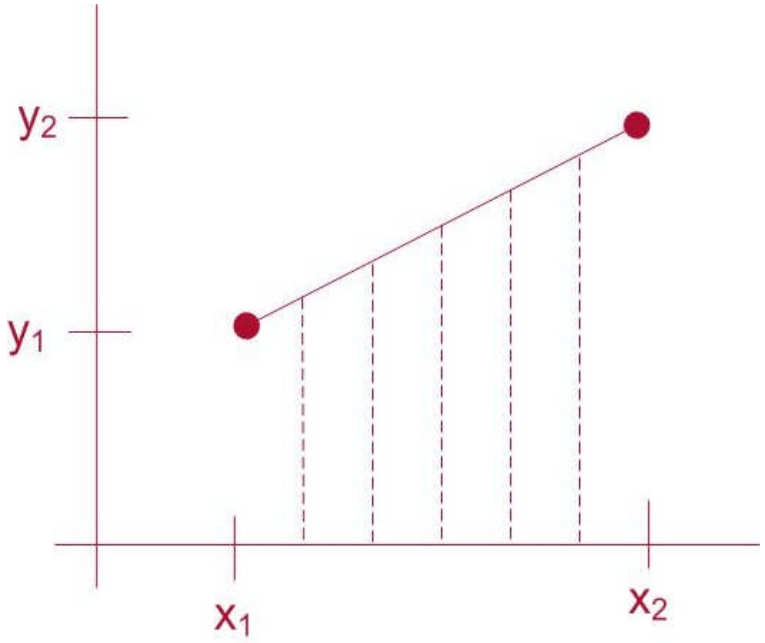


- **Interval Calculation**

$$\Delta y = m.\Delta x \qquad \Delta x = \frac{\Delta y}{m}$$

Digital System

- Sample a line at discrete positions and determine nearest pixel to the line at each sampled position



DDA Algorithm

- → Digital Differential Analyzer
  - Scan-conversion line – algorithm based on calculating either

$$\Delta x \quad \text{or} \quad \Delta y$$

  - Sample the line at unit intervals in one coordinate
  - Determine the corresponding integer values nearest the line path in another co-ordinate

CASES

- **Cases 1 → positive slope**
  - **LEFT END TO RIGHT END POINT**

a. m ≤ 1 → sampling in x-direction

$$\Delta x = 1$$
$$x_{k+1} = x_k + 1 \quad \therefore \Delta y = m\Delta x$$
$$y_{k+1} = y_k + m$$

b.m>1 → Sampling in y- direction (Reverse the role of x and y)

$$\Delta y = 1 \quad \therefore \Delta y = m\Delta x$$
$$x_{k+1} = x_k + \frac{1}{m}$$
$$y_{k+1} = y_k + 1$$

RIGHT END TO LEFT END POINT

a. m ≤ 1 → sampling in x-direction

$$\Delta x = -1$$
$$x_{k+1} = x_k - 1$$
$$y_{k+1} = y_k - m$$

b. m>1→ Sampling in y- direction

$$\Delta y = -1$$
$$x_{k+1} = x_k - \frac{1}{m}$$
$$y_{k+1} = y_k - 1$$

Case

- **Case 2→ Negative slope**
  - **LEFT END TO RIGHT END POINT**

a. |m| ≤ 1 → sampling in x-direction

$$\Delta x = 1$$

$$x_{k+1} = x_k + 1 \therefore \Delta y = m\Delta x$$

$$y_{k+1} = y_k + m$$

b. |m|>1→ Sampling in y- direction (Reverse the role of x and y)

$$\Delta y = -1$$

$$x_{k+1} = x_k - \frac{1}{m}$$

$$y_{k+1} = y_k - 1$$

Case 2

- **RIGHT END TO LEFT END POINT**

a. |m |≤ 1 → sampling in x-direction

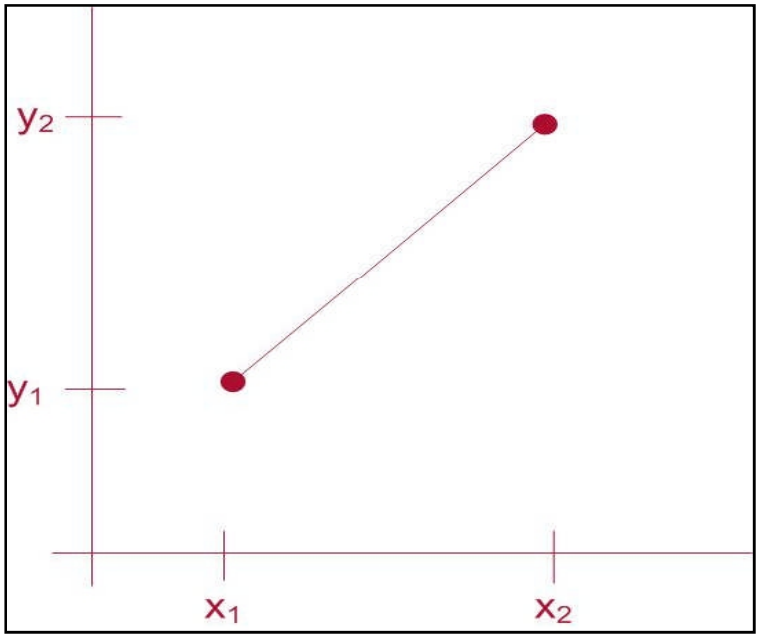
$$\Delta x = -1$$
$$x_{k+1} = x_k - 1$$
$$y_{k+1} = y_k - m$$

b. |m|>1→ Sampling in y- direction

$$\Delta y = 1 \quad \therefore \Delta y = m\Delta x$$
$$x_{k+1} = x_k + \frac{1}{m}$$
$$y_{k+1} = y_k + 1$$

DDA ALGORITHM

1. Input points (x_1, y_1) and (x_2, y_2)
2. Compute $dx = x_2 - x_1$ & $dy = y_2 - y_1$
3. If $|dx| > |dy|$ then steps $= |dx|$
   otherwise steps $= |dy|$
4. $x_{inc} = dx / steps$
   $y_{inc} = dy / steps$
5. $x = x_1$, $y = y_1$, $k = 0$
6. **do:**



- plot(round (x), round (y));
- $x = x + x_{inc}$ $y = y + x_{inc}$
- **k ++**

while (k<=steps)

Code

```
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));
    }
}
```

DDA→PROBLEMS & IMPROVEMENT

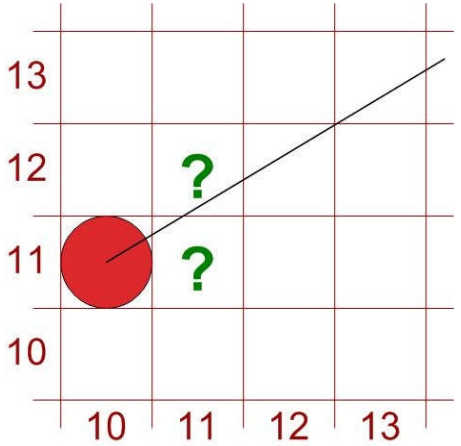
- PROBLEMS
  - Accumulations of round off error in successive additions of the floating point increment– this causes the calculated pixel drift away from the true line path for long line segments→ **ALIASING EFFECT**
  - Rounding operations and floating-point arithmetic operations are still time consuming

- IMPROVEMENT
  - Can be removed by making increments m and 1/m into integer and fractional parts so that all calculations are reduced to integer operations

Bresenham's Line Algorithm

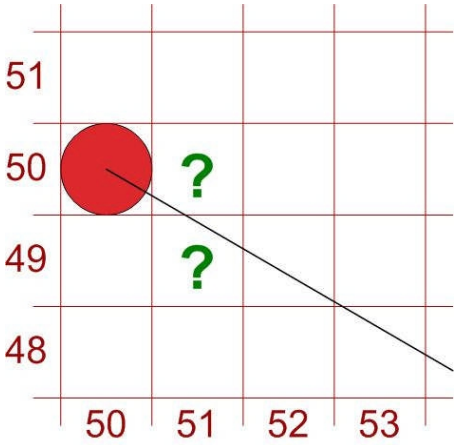
- Another incremental scan conversion algorithm
- Uses only incremental integer calculations
- Accurate and efficient line generating algorithm

<u>IDEA :</u> move across the x-axis in unit intervals and at each step choose between two y coordinates



E.g.

- Which pixel to draw ?
  - (11,11) or (11,12) ?
  - (51,50) or (51,49) ?
    - Solution→ choice is that which is closer to the original line



Bresenham's Line Algorithm contd…

- For $|m|<1$
  - i.e. $\Delta x = 1$

- Start from left end point (x_0, y_0) step to each successive column (x samples) and plot the pixel whose scan line y value is closest to the line path.

- After (x_k, y_k) the choice could be (x_k+1, y_k) or (x_k+1, y_k+1)

Bresenham's Line Algorithm contd…

y- coordinate on the mathematical line at x_{k+1} is

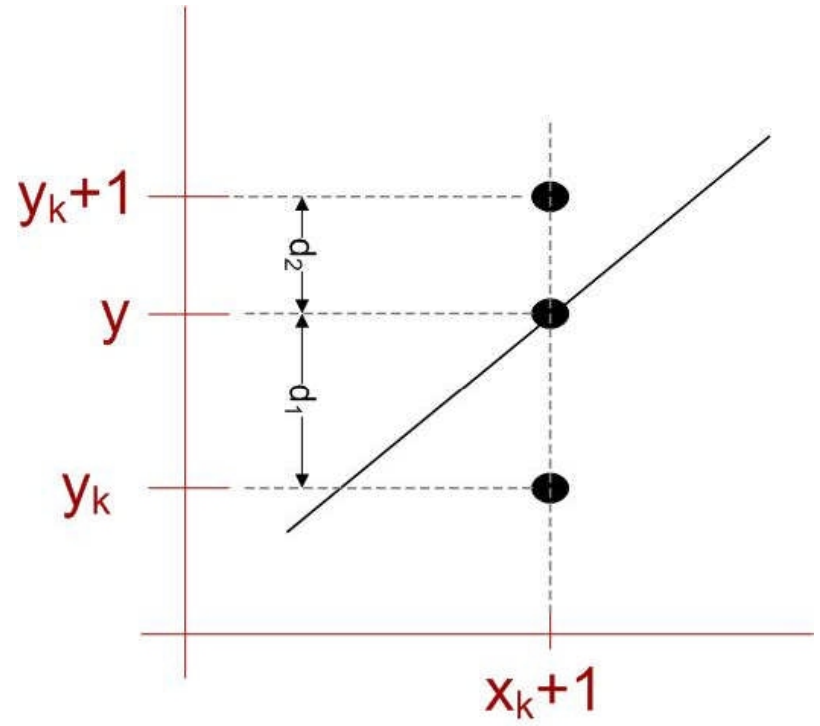
$$y = m(x_k + 1) + b$$

Then

$$d_1 = y - y_k$$
$$= m(x_k + 1) + b - y_k$$

And

$$d_2 = (y_k + 1) - y$$
$$= y_k + 1 - m(x_k + 1) - b$$



Difference between separations

$$d_1 - d_2 = 2m(x_k + 1) - 2y_k + 2b - 1$$

Bresenham's Line Algorithm contd…

Defining decision parameter

$$p_k = \Delta x(d_1 - d_2)$$
$$= 2\Delta y.x_k - 2\Delta x.y_k + c$$

Constant=2Δy + Δx(2b-1) Which is independent of pixel position

m = *Δy/Δx*

Sign of p_k is same as that of d_1-d_2 for Δx>0 (left to right sampling)

$$p_{k+1} = 2\Delta y.x_{k+1} - 2\Delta x.y_{k+1} + c$$

c eliminated here

$$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

because $x_{k+1} = x_k + 1$

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

y_{k+1}-y_k = 0 if $p_k < 0$
y_{k+1}-y_k = 1 if $p_k \geq 0$

For Recursive calculation, initially

$$p_0 = 2\Delta y - \Delta x$$

Substitute *b* = $y_0 - m.x_0$
and m = *Δy/Δx in [1]*

ALGORITHM

1. Input two points (x_1,y_1) and (x_2,y_2)
2. Compute $\Delta x = | x_2 - x_1 |$ **&** $\Delta y = |y_2 - y_1 |$
3. If $(x_2 > x_1)$ lx=1 else lx= - 1
4. If $(y_2 > y_1)$ ly=1 else ly= - 1
5. Plot first point (x_1,y_1)
6. If $(\Delta x > \Delta y)$ **{** /* i.e. when **|m|<1** */

   → calculate $p_0 = 2\Delta y - \Delta x$

   → Starting at k =0 to Δx times , repeat

   If $p_k<0$ /* next point (x_k+1,y_k) */

   $x_{k+1} = x_k+lx$, $y_{k+1}= y_k$

   $P_{k+1} = p_k + 2\Delta y$

   else /* next point $(x_k + 1, y_k+1)$ */

   $x_{k+1} = x_k+lx$, $y_{k+1}= y_k +ly$

   $P_{k+1} = p_k + 2\Delta y - 2\Delta x$

}ENDIF

ALGORITHM contd…

7. Else /* i.e. when **|m|>1** */

{

→ calculate $p_0 = 2\Delta x - \Delta y$

→ Starting at k =0 to Δy times , repeat

If $p_k<0$ /* next point (x_k, y_k+1) */

$x_{k+1} = x_k$

$y_{k+1} = y_k+1y$

$P_{k+1} = p_k + 2\Delta x$

else /* next point $(x_k + 1, y_k+1)$ */

$x_{k+1} = x_k+1x$

$y_{k+1} = y_k +1y$

$P_{k+1} = p_k + 2\Delta x - 2\Delta y$

}

Example

Digitize the line with endpoints (20,10) and (30,18)

Slope m = 0.8 $\quad \Delta x = 10, \Delta y = 8$

$P_0 = 2\Delta y - \Delta x = 6$

$2\Delta y = 16, \quad 2\Delta y - 2\Delta x = -4$

Plot $(x_0, y_0) = (20,10)$

Plot ?

k	p_k	(x_{k+1}, y_{k+1})	k	p_k	(x_{k+1}, y_{k+1})
0	6	(21,11)	5	6	(26,15)
1	2	(22,12)	6	2	(27,16)
2	-2	(23,12)	7	-2	(28,16)
3	14	(24,13)	8	14	(29,17)
4	10	(25,14)	9	10	(30,18)

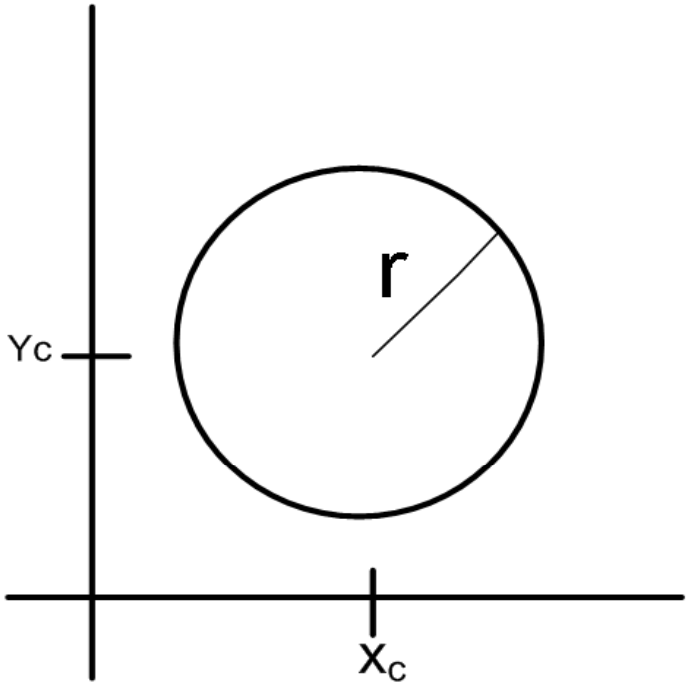
Circle

- Circle Equation:

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

- Points along circumference could be calculated by stepping along x-axis:

$$y = y_c \pm \sqrt{r^2 - (x_c - x)^2}$$



Example

- E.g. $x_c = y_c = 0$ and r

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

r=20

When x=0 then y=20

x=1 then y ≈ 20

x=2 then y ≈ 20

⋮

x=19 then y ≈ 6

x=20 then y =0

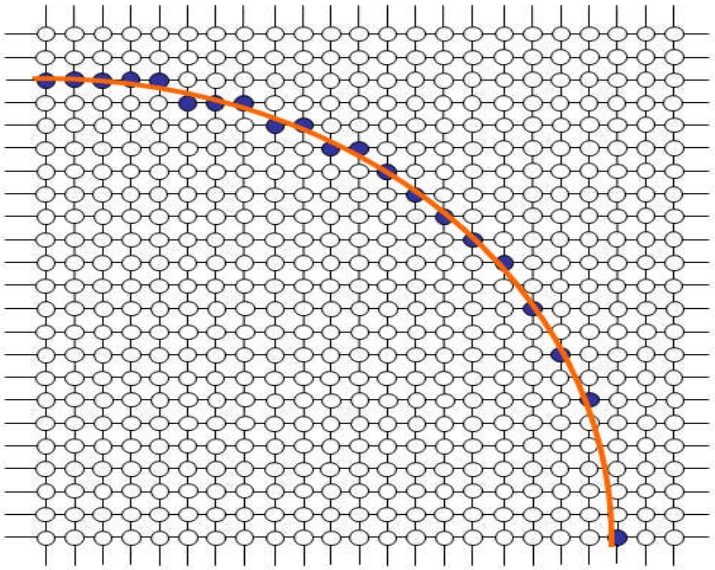
- **Problem ????**

1. Computational complexity

   →Square root and multiply operation

2. Non-uniform spacing:

   → Large gaps occurs when the



Adjustments (To fix problems)

1. Spacing problem (2 ways):
   - Interchange the role of x and y whenever the absolute value of the slope of the circle tangent > 1
   - Use polar co-ordinate:

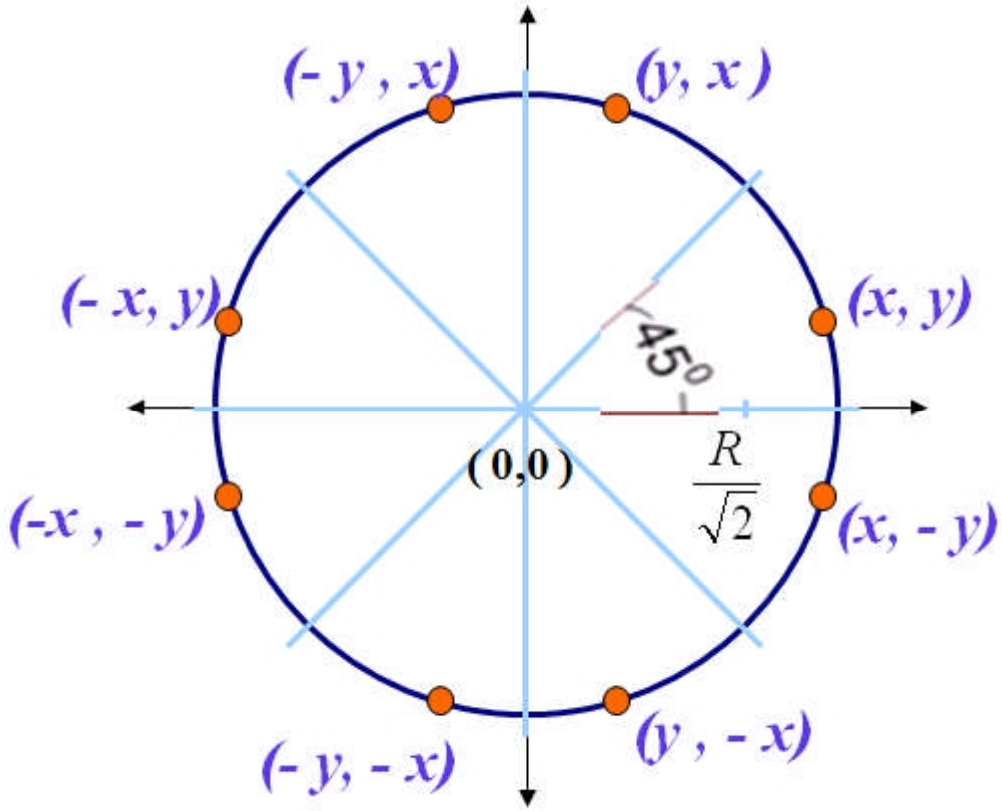
$$x = x_c + r\cos\theta$$

$$y = y_c + r\sin\theta$$

     - Equally spaced points are plotted along the circumference with fixed angular step size.
     - step size chosen for θ depends on the application and display device.
       - For more continuous boundary on a raster display, set step size at 1/r; i.e. pixel positions are approximately 1 unit apart.

2. Computation Problem:
   - Use symmetry of circle; i.e. calculate for one octant and use symmetry for others.

Eight way Symmetry of circle



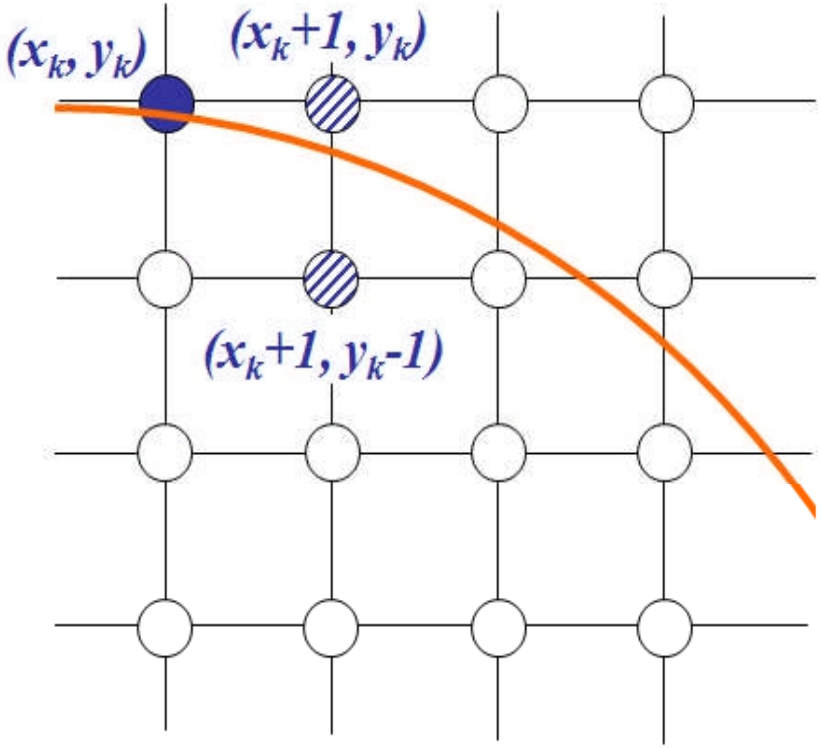
Mid-Point Circle Algorithm

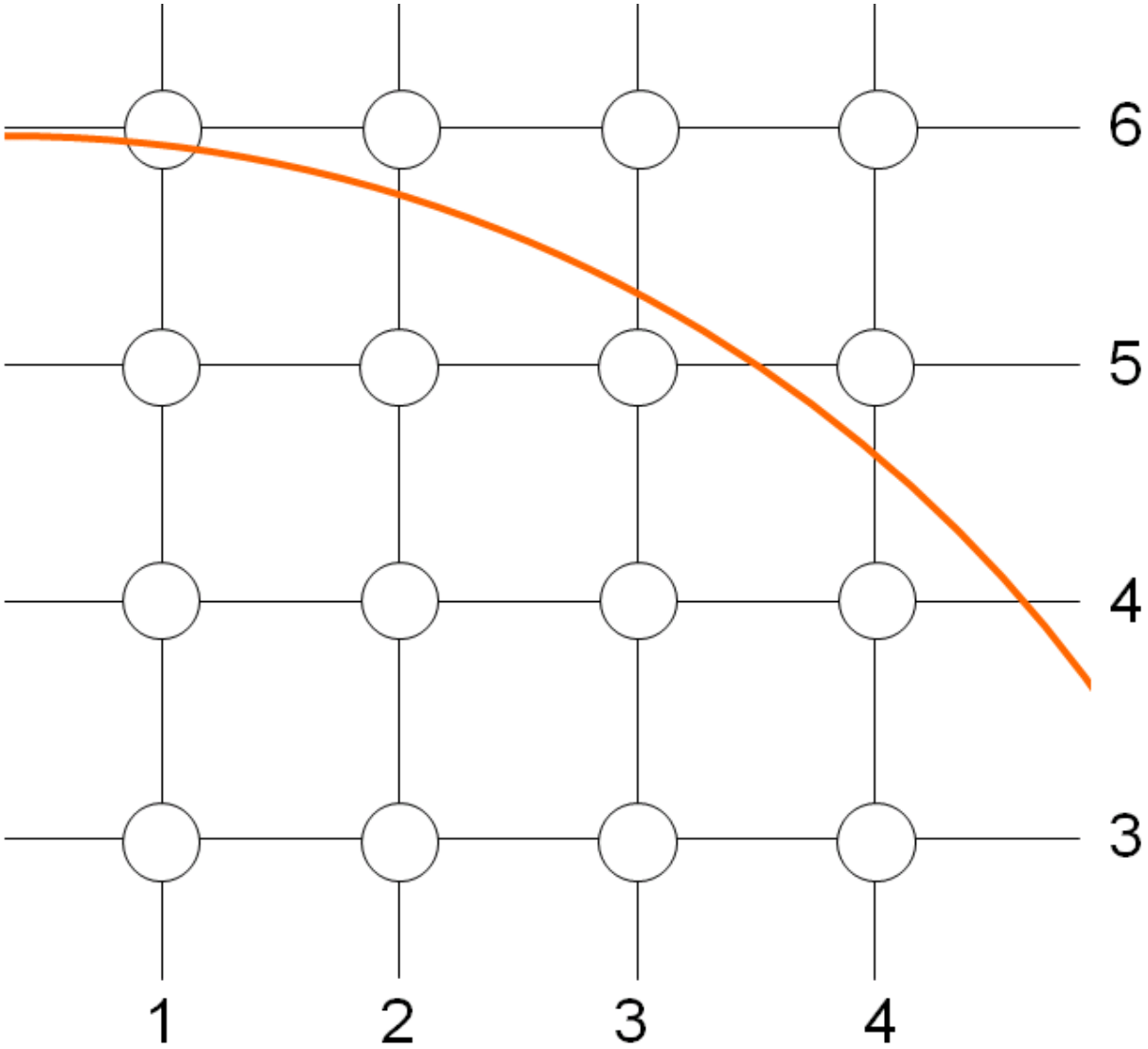
- Similarly to the case with lines, there is an incremental algorithm for drawing circles – the *mid-point circle algorithm*
- In the mid-point circle algorithm we use eight-way symmetry so only ever calculate the points for the top right eighth of a circle, and then use symmetry to get the rest of the points

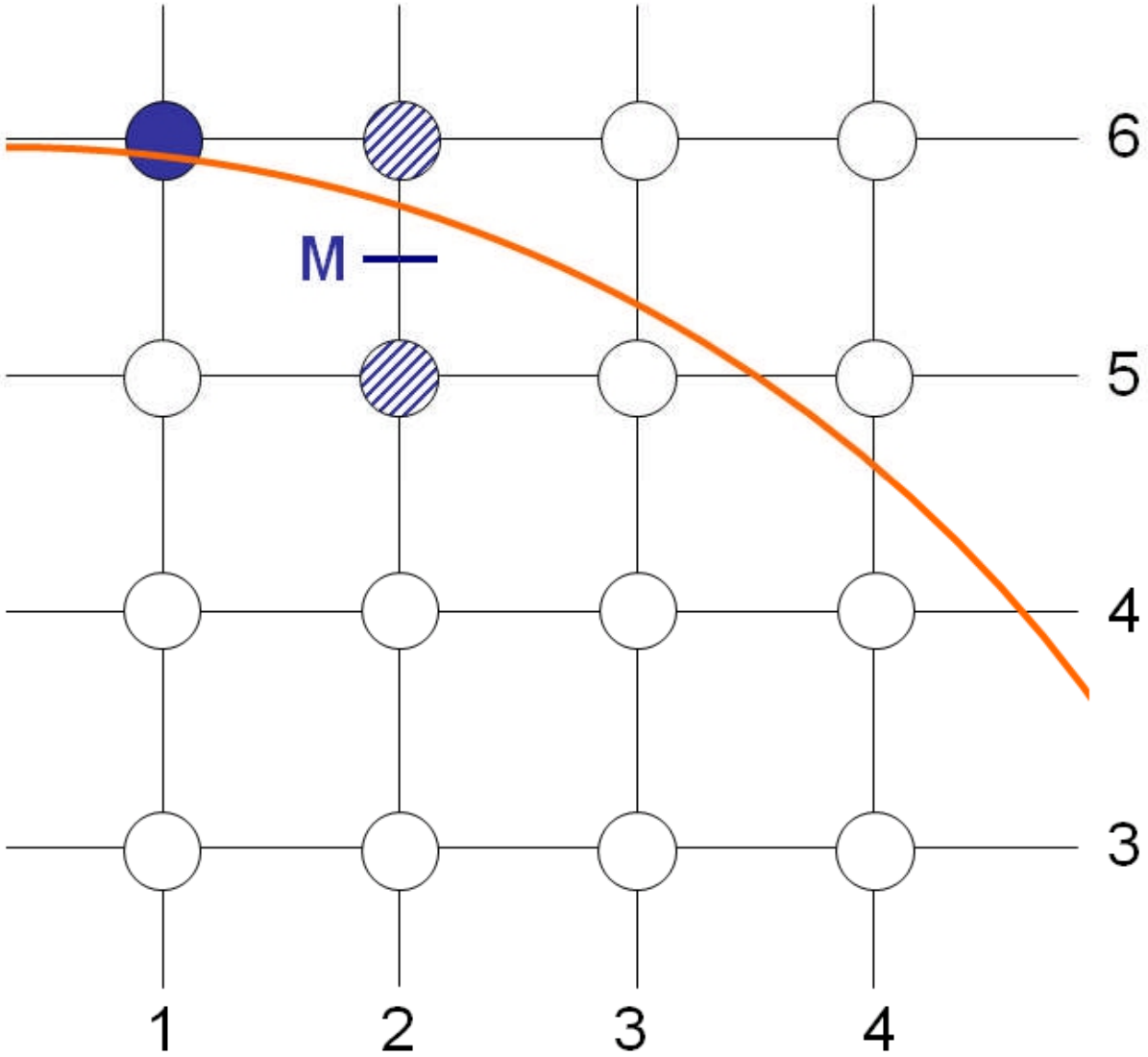
The mid-point circle algorithm was developed by Jack Bresenham, who we heard about earlier.

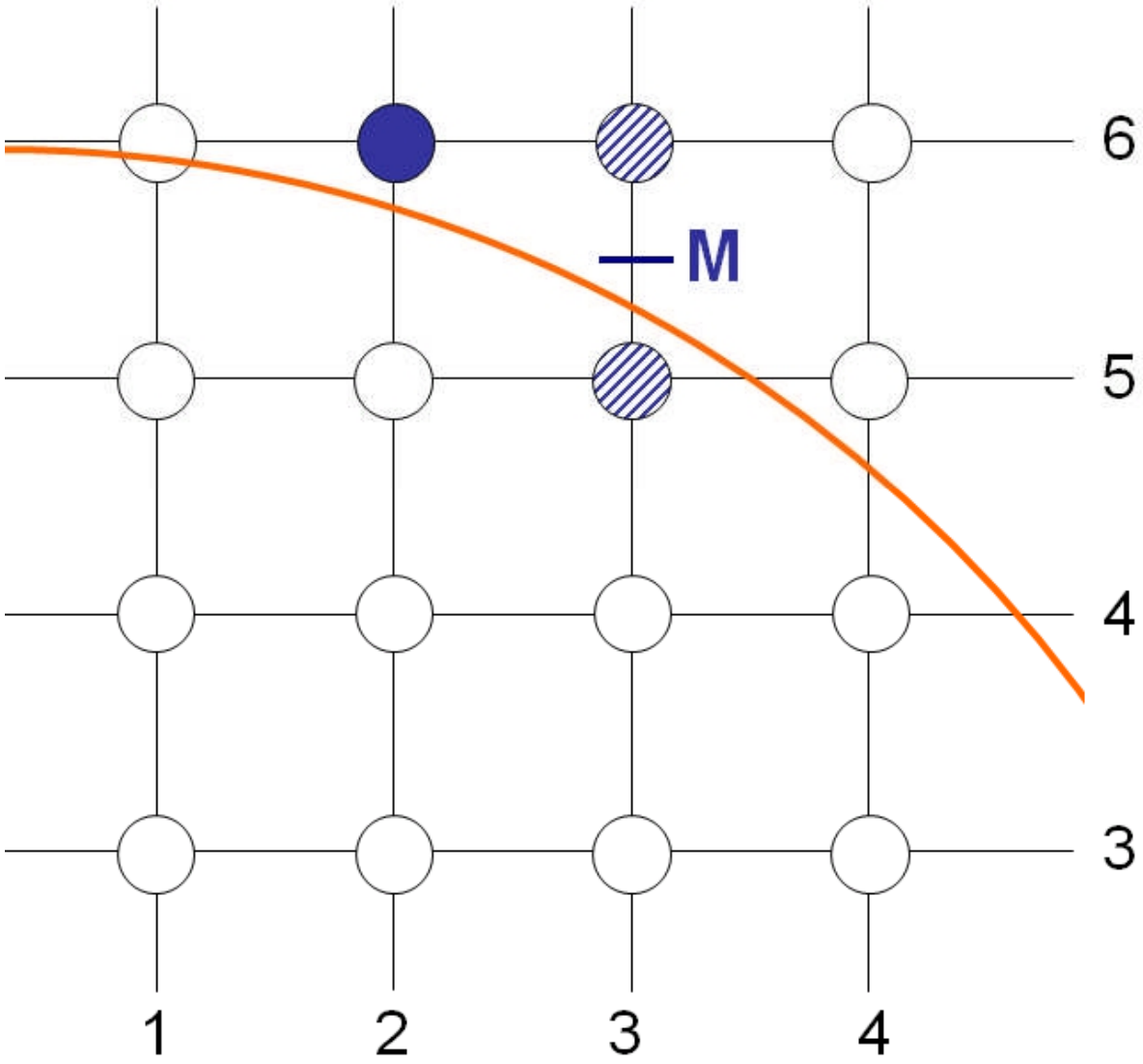
Mid-Point Circle Algorithm (cont…)

- Assume that we have just plotted point (x_k, y_k)
- The next point is a choice between (x_k+1, y_k) and (x_k+1, y_k-1)
- We would like to choose the point that is nearest to the actual circle
- So how do we make this choice?




6
5
4
3
1
2
3
4


M
6
5
4
3
1
2
3
4


6
M
5
4
3
1
2
3
4

Midpoint Circle Algorithm

- Circle function defined as:

$$f_{circle}(x, y) = x^2 + y^2 - r^2$$

- Any point (x , y) satisfies following conditions

$$f_{circle}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the circle boundary} \\ = 0, & \text{if } (x, y) \text{ is on the circle boundary} \\ > 0, & \text{if } (x, y) \text{ is outside the circle boundary} \end{cases}$$

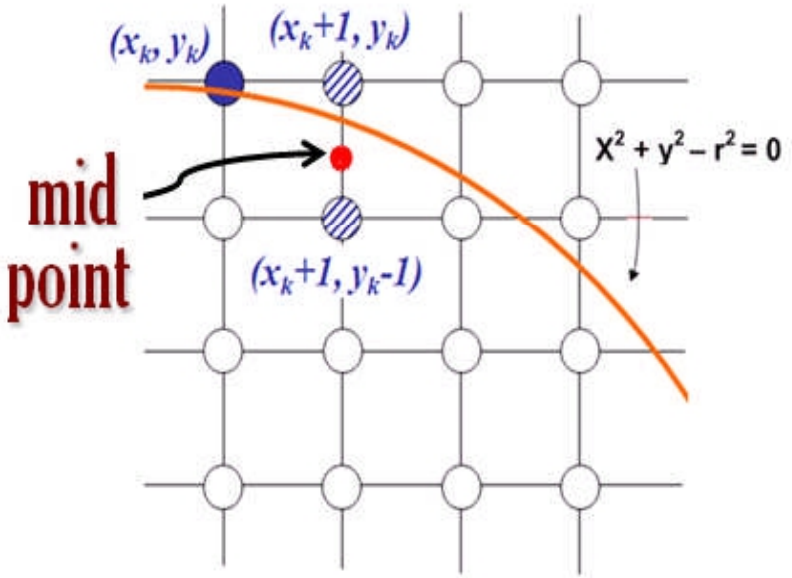
By evaluating this function at the midpoint between the candidate pixels we can make our decision

Midpoint Circle Algorithm contd….

- Decision parameter is the circle function: evaluated as:

$$p_k = f_{circle}(x_k + 1, y_k - \frac{1}{2})$$

$$= (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2$$



$$p_{k+1} = f_{circle}(x_{k+1} + 1, y_{k+1} - \frac{1}{2})$$

$$= [(x_k + 1) + 1]^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$$

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}{}^2 - y_k{}^2) - (y_{k+1} - y_k) + 1$$

$\because [p_{k+1} - p_k]$

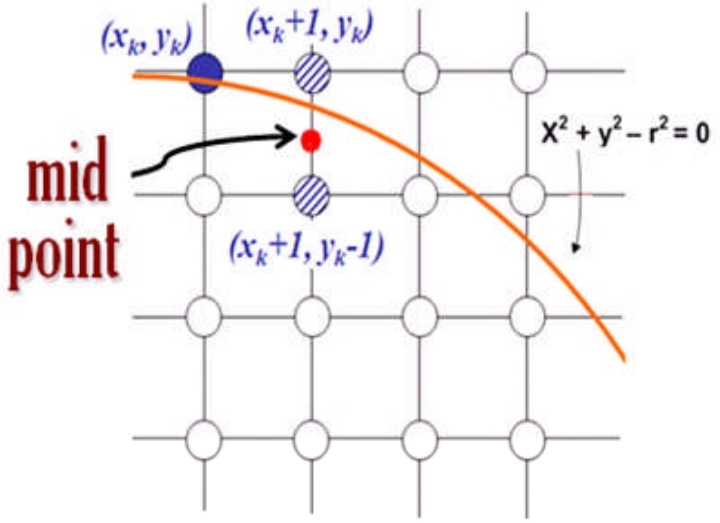
Midpoint Circle Algorithm contd….

- if $p_k < 0$

$$y_{k+1} = y_k$$
$$P_{k+1} = P_k + 2x_{k+1} + 1$$

- if $p_k > 0$

$$y_{k+1} = y_k - 1$$
$$P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$$



Note:

- Incremental evaluation of $2x_{k+1}$ and $2y_{k+1}$

$$2x_{k+1} = 2x_k + 2$$
$$2y_{k+1} = 2y_k - 2 \quad \text{if } p_k > 0$$

- At start position $(x_0, y_0) = (0, r)$

$$2x_0 = 0 \text{ and } 2y_0 = 2r$$

Midpoint Circle Algorithm contd….

- Initial decision parameter

$$p_0 = f_{circle}(1, r - \frac{1}{2})$$
$$= 1 + (r - \frac{1}{2})^2 - r^2$$
$$= \frac{5}{4} - r$$

- For r specified as an integer, round p_0 to

$$P_0 = 1-r$$

(because all increments are integers)

Algorithm

1. Input radius r and circle center (x_c, y_c) and obtain the first point on the circumference of a circle centered on the origin as

   $(x_0, y_0) = (0, r)$

2. Calculate the initial value of the decision parameter as

   $P_0 = 5/4 - r$

3. At each x_k position, starting at k = 0, perform the following test:

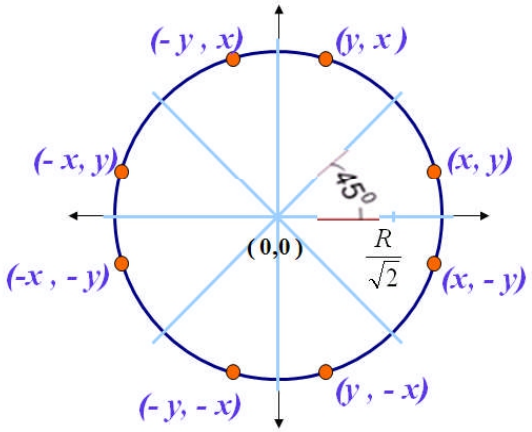
   If $p_k < 0$ /* next point (x_k+1, y_K)*/

   $$x_{k+1} = x_k + 1$$
   $$y_{k+1} = y_k$$
   $$P_{k+1} = p_k + 2x_{k+1} + 1$$

   else /*next point $(x_k+1, y_K - 1)$ */

   $$x_{k+1} = x_k + 1$$
   $$y_{k+1} = y_k - 1$$
   $$P_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

   Where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$



4. Determine the symmetry points in the other seven octants.
5. Move each calculated pixel position (x,y) onto the circular path centered on (x_c, y_c) and plot the co-ordinate values:

   $x = x + x_c$, $y = y + y_c$

6. Repeat steps 3 through 5 until $x \geq y$

Numerical

Draw a circle with radius $r = 8$, centering at (0,0). Initial point is

$$(x_0, y_0) = (0, 8)$$

Initial decision parameter is

$$p_0 = 1 - r = 7.$$

Successive pixels of the Midpoint circel are listed in the following table:

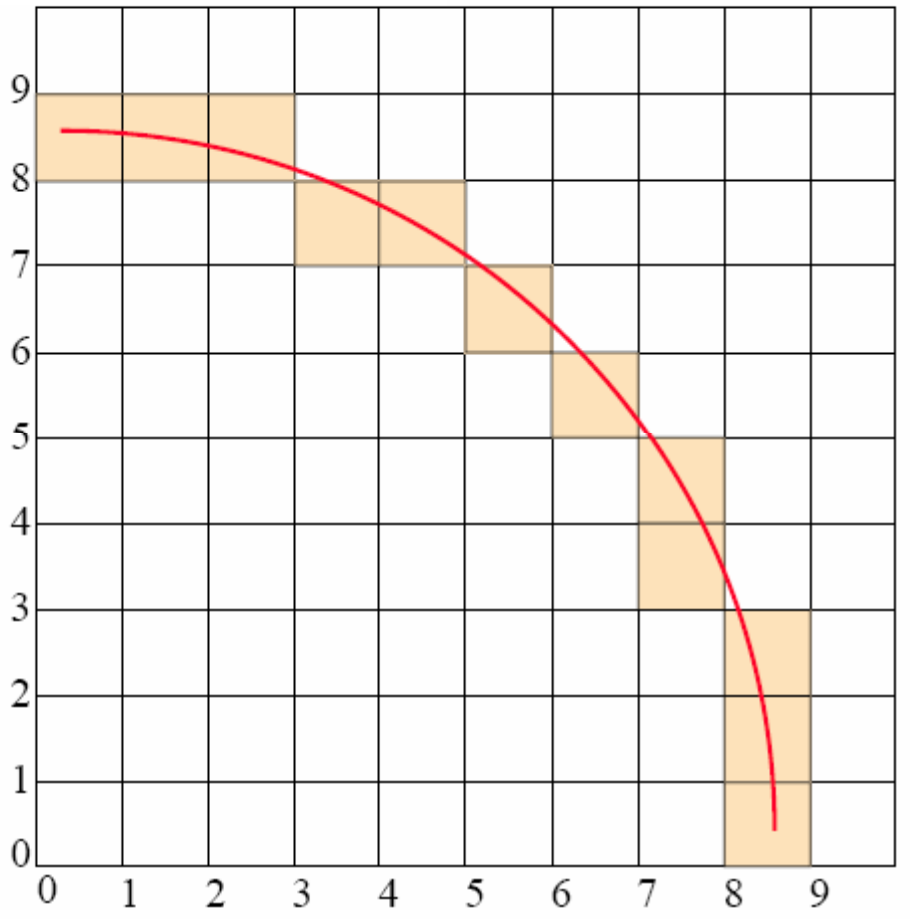
k	p_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	-7	(1,8)	2	16
1	-4	(2,8)	4	16
2	1	(3,7)	6	14
3	-6	(4,7)	8	14
4	3	(5,6)	10	12
5	2	(6,5)	12	10

Successive pixels in a Midpoint circle.

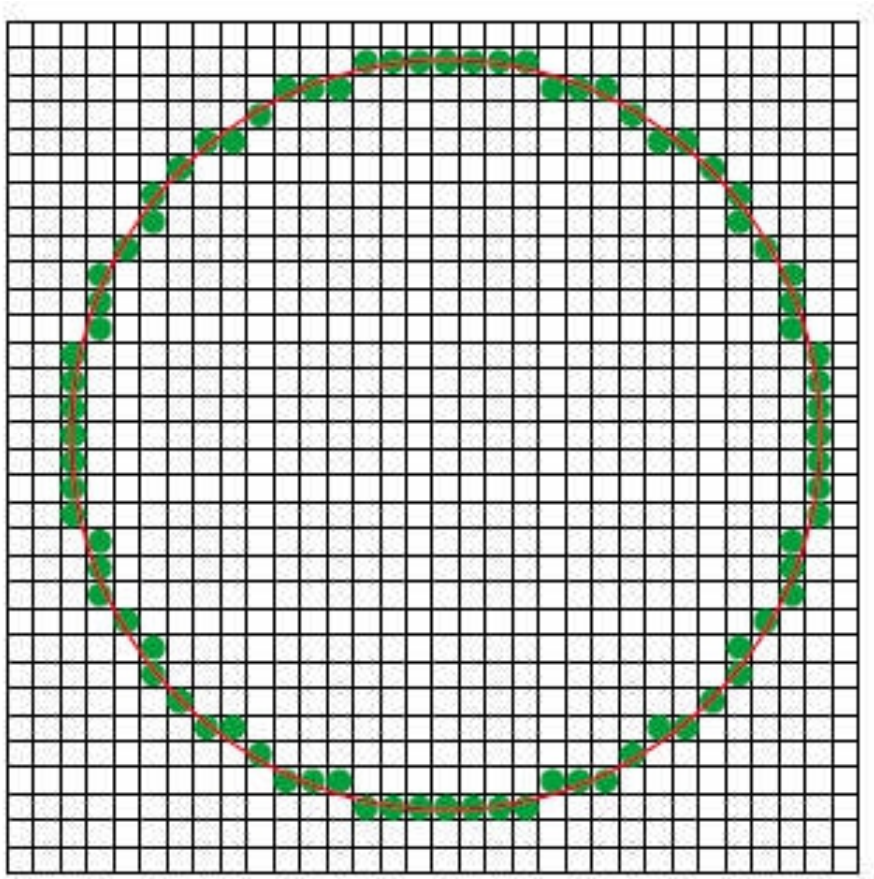
Numerical

PLOT

initial pixel → (0,8)
(1,8)
(2,8)
(3,7)
(4,7)
(5,6)
(6,5)
(7,4)
(7,3)
(8,2)
(8,1)
end pixel → (8,0)



Complete Plot



Example

Digitize the circle with r=10 and center located at origin

Initial point $(x_0,y_0)=(0,10)$ $2x_0=0$ $2y_0=20$

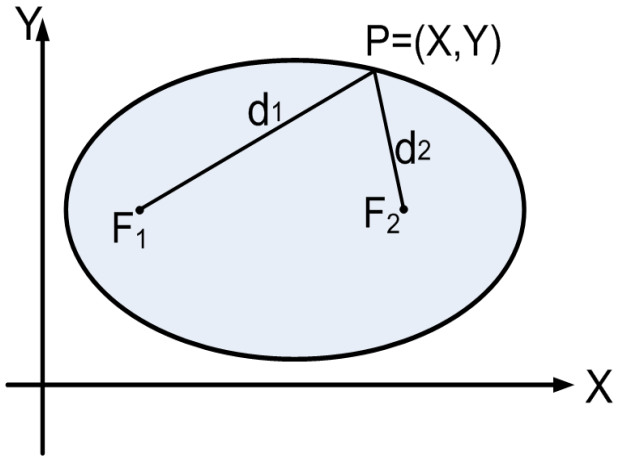
$P_0 = 1- r = 1-10 = -9$

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)	14	14

Ellipse

- Elongated circle
- Equation of ellipse:

$$d_1 + d_2 = constant$$



- F1→(x_1,y_1), F2→(x_2,y_2)

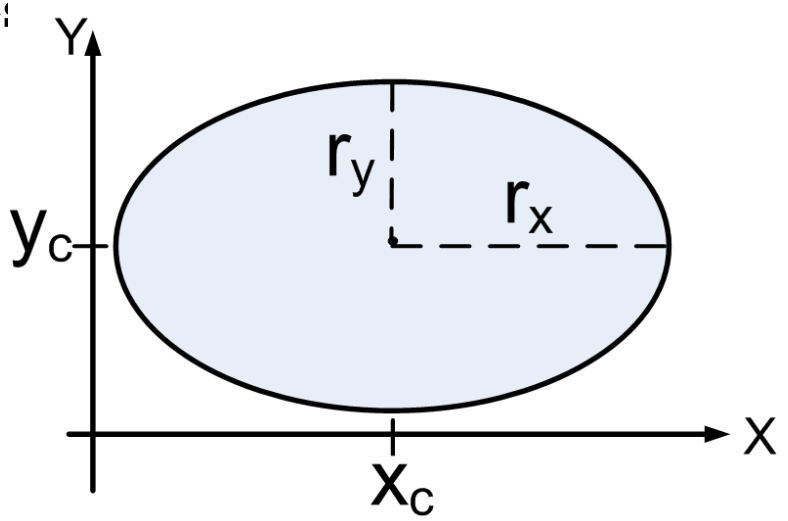
$$\sqrt{(x-x_1)^2+(y-y_1)^2}+\sqrt{(x-x_2)^2+(y-y_2)^2}=constant$$

- General Equation

$$Ax^2 + By^2 + Cxy + Dx + Ey + F = 0$$

- In terms of ellipse center coordinates

$$\left(\frac{x-x_c}{r_x}\right)^2+\left(\frac{y-y_c}{r_y}\right)^2=1$$



- In polar co-ordinate

$$x = x_c + r_x \cos\theta$$
$$y = y_c + r_y \sin\theta$$

Mid Point Ellipse Algorithm

- Ellipse function

$$f_{ellipse}(x,y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$$

$$f_{ellipse}(x,y)\begin{cases} <0, & if\ (x,y)\ is\ inside\ the\ ellipse\ boundary \\ =0, & if\ (x,y)\ is\ on\ the\ ellipse\ boundary \\ >0, & if\ (x,y)\ is\ outside\ the\ ellipse\ boundary \end{cases}$$

Mid Point Ellipse Algorithm contd…

- From ellipse tangent slope:

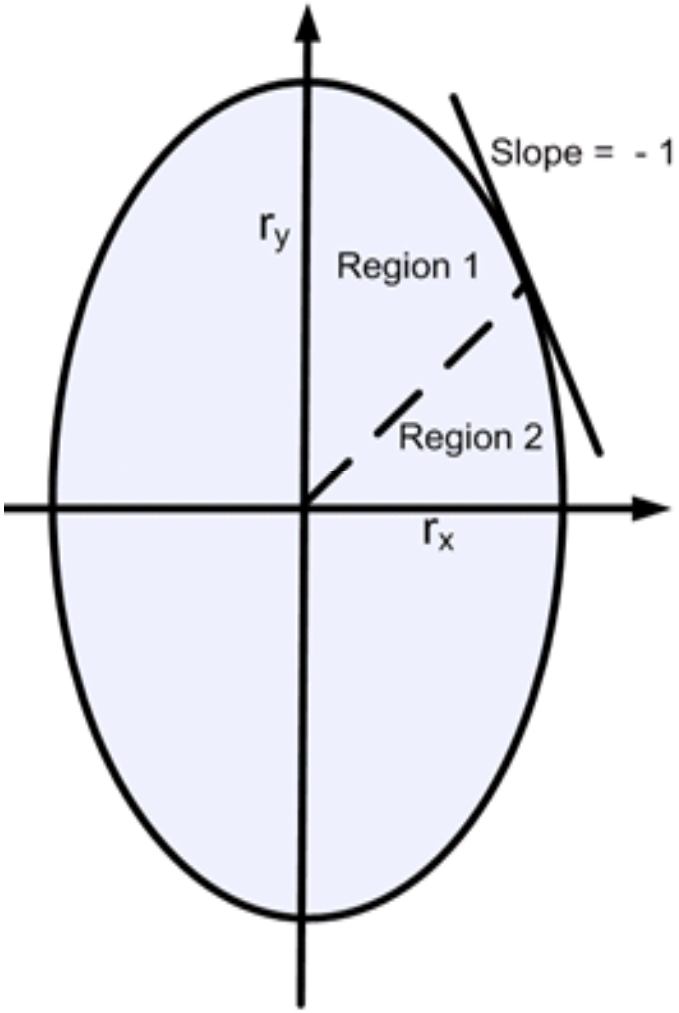
$$\frac{dy}{dx} = -\frac{2r_y^2 x}{2r_x^2 y}$$

- At boundary region (dy/dx= -1)

$$2r_y^2 x = 2r_x^2 y$$

- Start from $(0,r_y)$, take x samples to boundary between 1 and 2
- Switch to sample y from boundary between 1 and 2

  (i.e whenever $2r_y^2 x \geq 2r_x^2 y$)

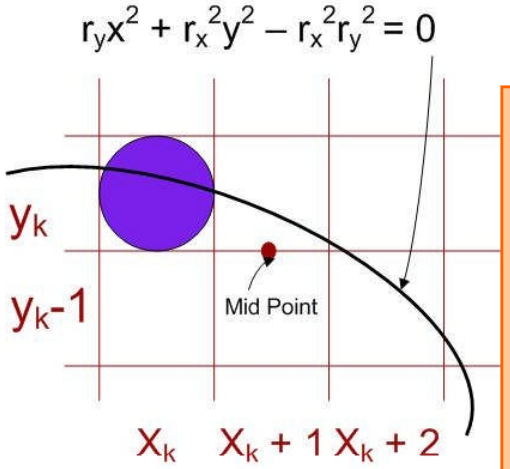


Mid Point Ellipse Algorithm contd…

- In the region 1

$$p1_k = f_{ellipse}(x_k + 1, y_k - \frac{1}{2})$$

$$= r_y^{\ 2}(x_k + 1)^2 + r_x^{\ 2}(y_k - \frac{1}{2})^2 - r_x^{\ 2} r_y^{\ 2}$$



- For next sample

$$p1_{k+1} = f_{ellipse}(x_{k+1} + 1, y_{k+1} - \frac{1}{2})$$

$$= r_y^{\ 2}[(x_k + 1) + 1]^2 + r_x^{\ 2}(y_{k+1} - \frac{1}{2})^2 - r_x^{\ 2} r_y^{\ 2}$$

$$p1_{k+1} = p1_k + 2r_y^{\ 2}(x_k + 1) + r_y^2 + r_x^2\left[\left(y_{k+1} - \frac{1}{2}\right)^2 - \left(y_k - \frac{1}{2}\right)^2\right]$$

- Thus increment

$$increment = \begin{cases} 2r_y^{\ 2} x_{k+1} + r_y^2, & if \quad p1_k < 0 \\ 2r_y^{\ 2} x_{k+1} + r_y^2 - 2r_x^{\ 2} y_{k+1}, & if \quad p1_k \geq 0 \end{cases}$$

- For increment calculation; Initially:

$$2r_y^{\ 2} x = 0$$

$$2r_x^{\ 2} y = 2r_x^{\ 2} r_y$$

- Incrementally:
  Update x by adding $2r_y^{\ 2}$ to first equation and update y by subtracting $2r_x^{\ 2}$ to second equation

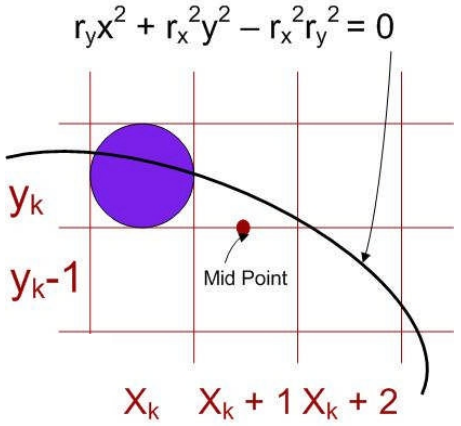
Mid Point Ellipse Algorithm contd…

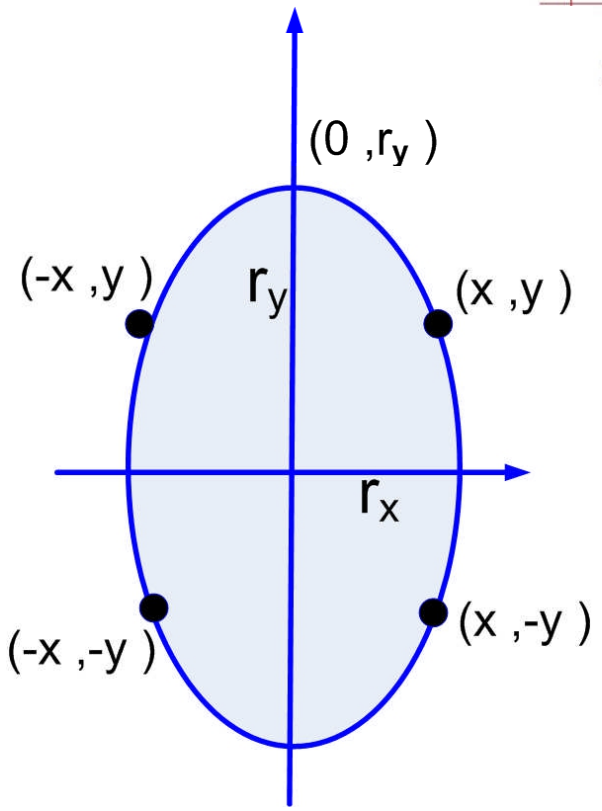
- Initial value

$$p1_0 = f_{ellipse}\left(1, r_y - \frac{1}{2}\right)$$

$$= r_y^2 + r_x^2\left(r_y - \frac{1}{2}\right)^2 - r_x^2 r_y^2$$

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4}r_x^2$$





Mid Point Ellipse Algorithm contd…

- In the region 2

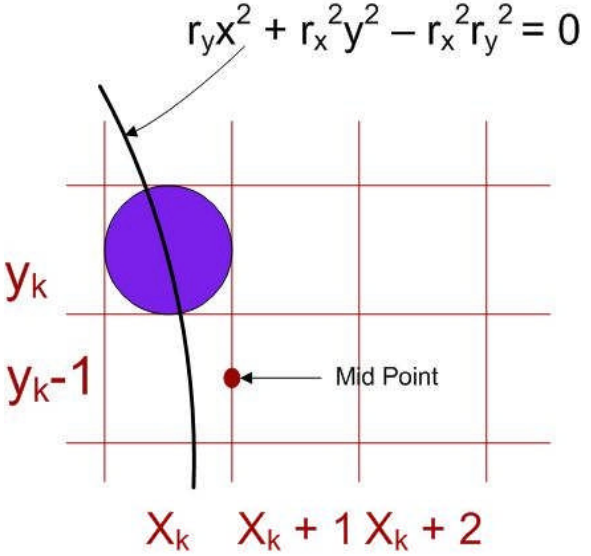
$$p2_k = f_{ellipse}(x_k + \frac{1}{2}, y_k - 1)$$
$$= r_y^2 (x_k + \frac{1}{2})^2 + r_x^2 (y_k - 1)^2 - r_x^2 r_y^2$$

- For next sample

$$p2_{k+1} = f_{ellipse}(x_{k+1} + \frac{1}{2}, y_{k+1} - 1)$$
$$= r_y^2 \left(x_{k+1} + \frac{1}{2}\right)^2 + r_x^2 \left[\left(y_k - 1\right) - 1\right]^2 - r_x^2 r_y^2$$
$$p2_{k+1} = p2_k + 2r_x^2 (y_k - 1) + r_x^2 + r_y^2 \left[\left(x_{k+1} + \frac{1}{2}\right)^2 - \left(x_k + \frac{1}{2}\right)^2\right]$$

- Initially

$$p2_0 = f_{ellipse}\left(x_0 + \frac{1}{2}, y_0 - 1\right)$$
$$p2_0 = r_y^2 \left(x_0 + \frac{1}{2}\right)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$



For simplification calculation of $p2_0$ can be done by selecting pixel positions in counter clockwise order starting at $(r_x,0)$ and unit samples to positive y direction until the boundary between two regions

Algorithm

1. Input r_x,r_y, and the ellipse center (x_c, y_c) and obtain the first point on an ellipse centered on the origin as

$$(x_0,y_0) = (0,r_y)$$

2. Calculate the initial value of the decision parameter in region 1 as

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

3. At each x_k position in region 1, starting at k = 0, perform the following test:

If $p1_k < 0$ /* next point (x_{k+1},y_k)

$$x_{k+1} = x_k + 1,\ y_{k+1} = y_k$$

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} + r_y^2$$

Else /* next point (x_k+1,y_k-1) */

$$x_{k+1} = x_k + 1,\ y_{k+1} = y_k - 1$$

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

With

$$2r_y^2 x_{k+1} = 2r_y^2 x_k + 2r_y^2, \qquad 2r_x^2 y_{k+1} = 2r_x^2 y_k - 2r_x^2$$

and continue until $2r_y^2 x \geq 2r_x^2 y$

Algorithm contd…

4. Calculate the initial value of decision parameter in region 2 using the last point (x_0,y_0) calculated in region 1 as

$$p2_0 = r_y^2\left(x_0+\frac{1}{2}\right)^2 + r_x^2(y_0-1)^2 - r_x^2 r_y^2$$

5. At each y_k position in region 2, starting at k = 0, perform the following test:

If $p2k \le 0$ /* next point (x_k+1, y_k-1)*/

$x_{k+1} = x_k+1$

$y_{k+1} = y_k-1$

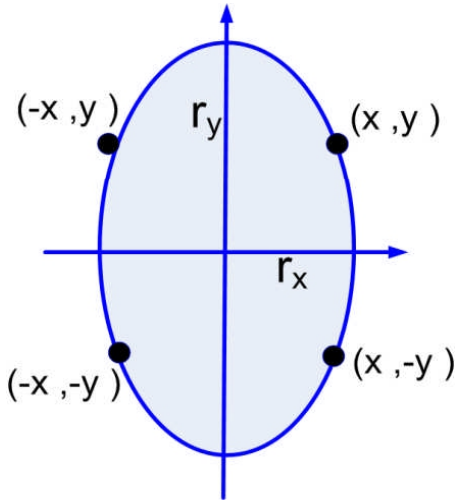
$$p2_{k+1} = p2_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

Else /*next point (x_k, y_k-1)*/

$x_{k+1} = x_k$

$y_{k+1} = y_k-1$

$$p2_{k+1} = p2_k - 2r_x^2 y_{k+1} + r_x^2$$



4. Using the same incremental calculations for x and y as in region 1. continue until y=0.
6. Determine the symmetry points in the other three quadrants.
7. Move each calculated pixel position (x,y) onto the elliptical path centered on (x_c,y_c) and plot the co-ordinate values:

$x = x + x_c$, $y = y + y_c$

Numerical

Draw an ellipse with r_x =10 and r_y = 8, centering at (0,0).

For region 1, the initial point for the ellipse centered on the origin is $(x_0, y_0) = (0, 8)$, and the initial decision parameter value is

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2 = -711.$$

Successive midpoint decision-parameter values and the pixel positions along the ellipse are listed in the following table:

k	$p1_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-711	(1,8)	128	1600
1	-519	(2,8)	256	1600
2	-199	(3,8)	384	1600
3	249	(4,7)	512	1400
4	-575	(5,7)	640	1400
5	129	(6,6)	768	1200
6	-239	(7,6)	896	1200
7	721	(8,5)	1024	1000

Table 3: Successive pixels in the region 1 of a Midpoint ellipse.

We now move out of region 1, since $2r_y^2 x > 2r_x^2 y$.

Numerical

For region 2, the initial point is $(x_0, y_0) = (8, 5)$ and the initial decision parameter is

$$p2_0 = f_{\text{ellipse}}(8 + \frac{1}{2}, 4) = -704.$$

8

The remaining positions along the ellipse path in the first quadrant are then calculated as:

k	$p2_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-704	(9,4)	1152	800
1	-252	(10,3)	1280	600
2	528	(10,2)	1280	400
3	228	(10,1)	1280	200
4	128	(10,0)	-	-

Successive pixels in the region 2 of a Midpoint ellipse.

Numerical

- rx= 8 ry=6 and center (0,0)

For region 1: The initial point for the ellipse centered on the origin is $(x_0, y_0) = (0, 6)$, and the initial decision parameter value is

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2 = -332$$

Successive decision parameter values and positions along the ellipse path are calculated using the midpoint method as

k	$p1_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	−332	(1, 6)	72	768
1	−224	(2, 6)	144	768
2	−44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	−108	(5, 5)	360	640
5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

We now move out of region 1, since $2r_y^2 x > 2r_x^2 y$.

Numerical Contd…

For region 2, the initial point is $(x_0, y_0) = (7, 3)$ and the initial decision parameter is

$$p2_0 = f\left(7 + \frac{1}{2}, 2\right) = \cdot$$

Find Yourself

The remaining positions along the ellipse path in the first quadrant are then calculated as

k	$p2_k$	(x_{k+1}, y_{k+1})
0	?	(8, 2)
1	?	(8, 1)
2	?	(8, 0)

TWO- DIMENSIONAL GEOMETRIC TRANSFORMATION

TRANSFORMATION

- process of changing orientation, shape or size
  - Translation
  - Rotation
  - Scaling
  - Reflection
  - shearing

HOMOGENOUS FORM

- Expressing position in homogeneous coordinates allows us to represent all geometric transformations equation as matrix multiplications.

(x, y) ---------- (x_h, y_h, h)

$x = x_h / h$ $y = y_h / h$

where h is any non zero value

For convenient h=1

(x,y)----------(x, y, 1).

TRANSLATION

- Every point on the object is translated by the same amount

$$x' = x + t_x, \qquad y' = y + t_y$$

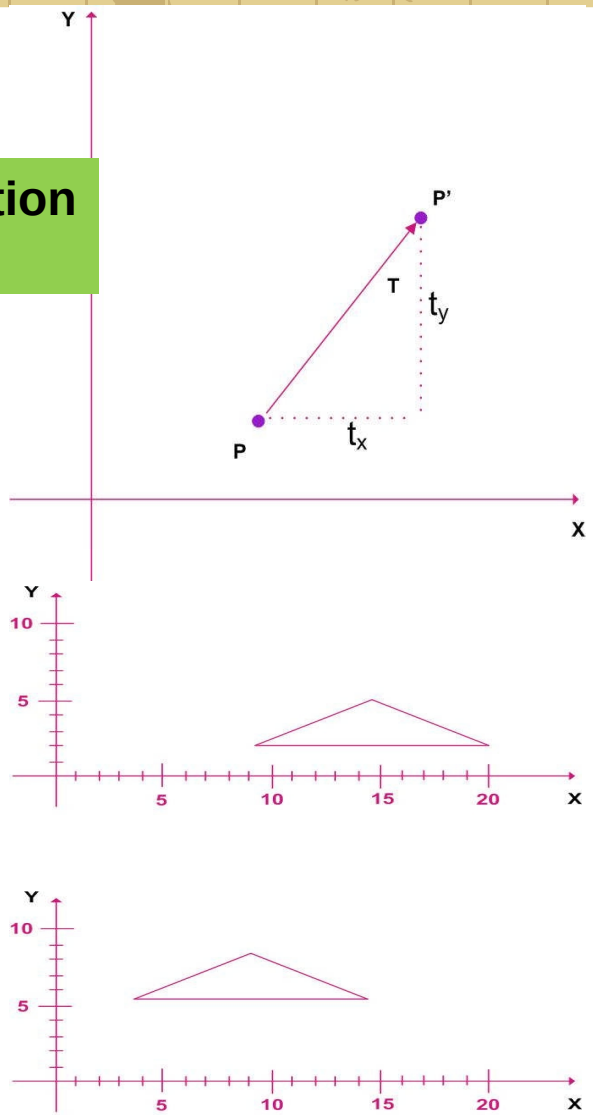
Translation vector

$$P = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \qquad p' = \begin{bmatrix} x_1' \\ y_1' \end{bmatrix}, \qquad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$P' = P + T$$

In homogeneous representation if position P = (x, y) is translated to new position P'= (x', y') then:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow P' = T(t_x, t_y).P$$



ROTATION

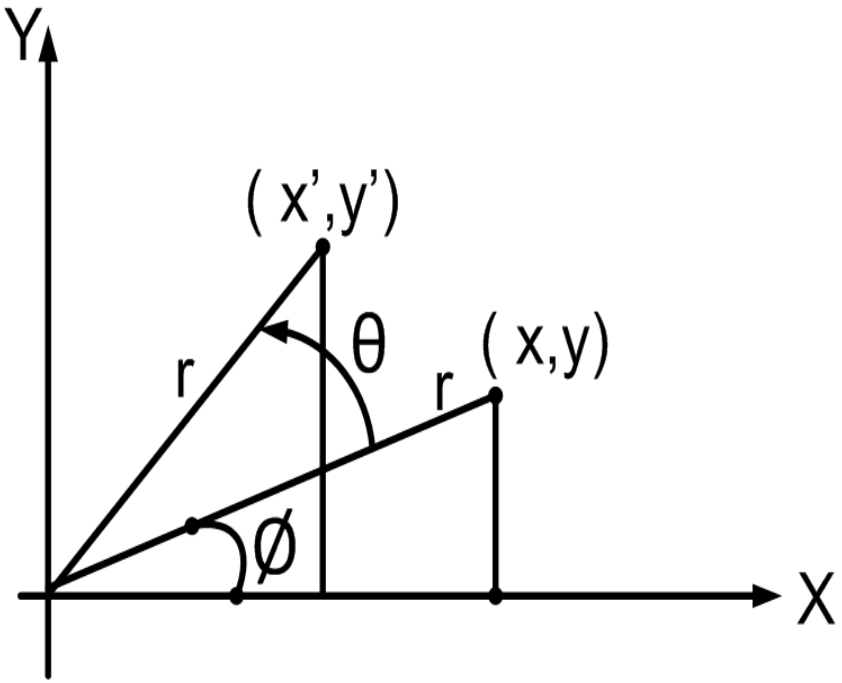
- Two- dimensional rotation is applied to an object by repositioning it along a circular path in xy plane

REQUIREMENT:

Rotation angle θ

&

pivot point



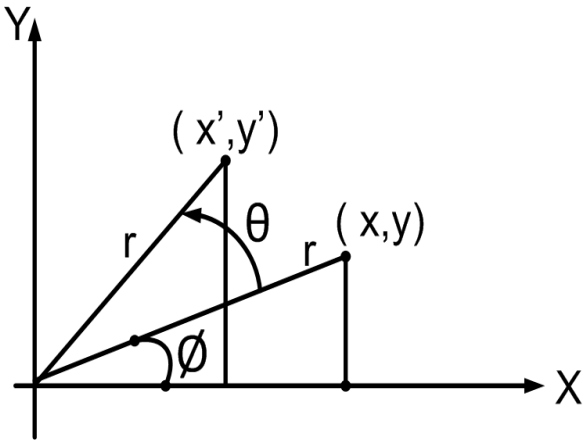
- Clockwise rotation (Negative)

- Anticlockwise rotation (positive)

ROTATION contd…

Coordinates of point (x,y) in polar form

$$x = r\cos\phi, \qquad y = r\sin\phi$$



$$x' = r\cos(\phi+\theta) = r\cos\phi.\cos\theta - r\sin\phi.\sin\theta$$
$$y' = r\sin(\phi+\theta) = r\cos\phi.\sin\theta + r\sin\phi.\cos\theta$$

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

$$P' = R.P$$

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

ROTATION MATRX

If Co-ordinates represented as row vector, Then:

$$P'^{T} = (R.P)^{T}$$
$$= P^{T}.R^{T}$$

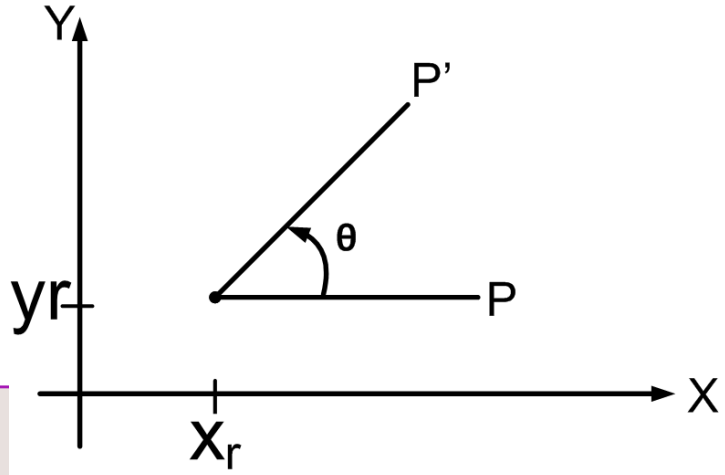
In homogeneous co-ordinate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow P' = R(\theta).P$$

ROTATION OF A POINT (X,Y) ABOUT ANY POINT (Xr, Yr)

$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$$
$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$$

Steps
1. **Translate object so as to coincide pivot to origin**
2. **Rotate object about the origin**
3. **Translate object back so as to return pivot to original position**



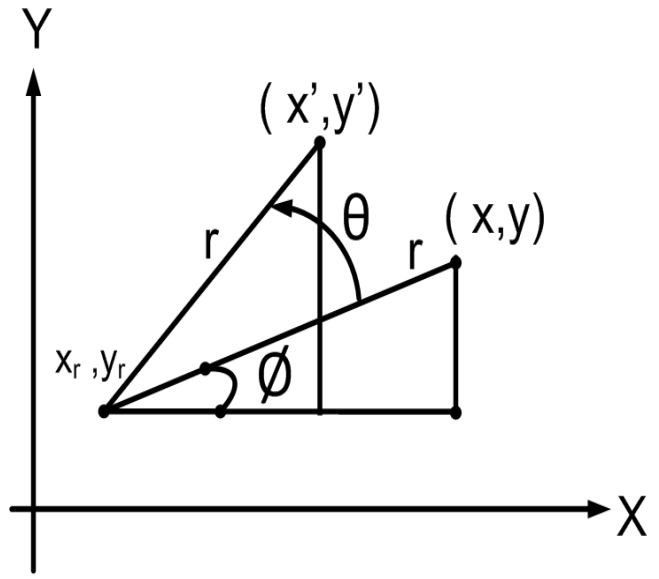
Composite Transformations

$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & x_r(1-\cos\theta) + y_r\sin\theta \\ \sin\theta & \cos\theta & y_r(1-\cos\theta) - x_r\sin\theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$T(x_r, y_r).R(\theta).T(-x_r, -y_r) = R(x_r, y_r, \theta)$$

$$T(-x_r, -y_r) = T^{-1}(x_r, y_r)$$



SCALING

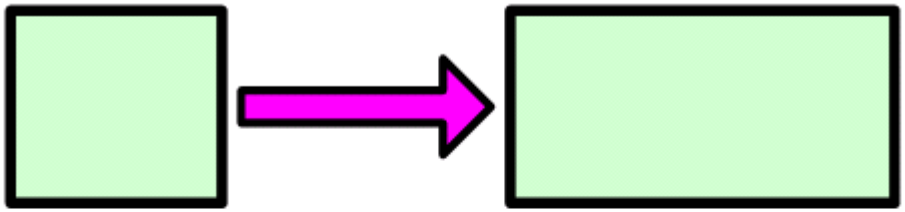
- alters the size of an object

$$x' = x.s_x, \qquad y' = y.s_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}$$

2 x 2 Scaling Matrix

$$P' = S.P$$



Square to Rectangle (Sx = 2 , Sy =1)

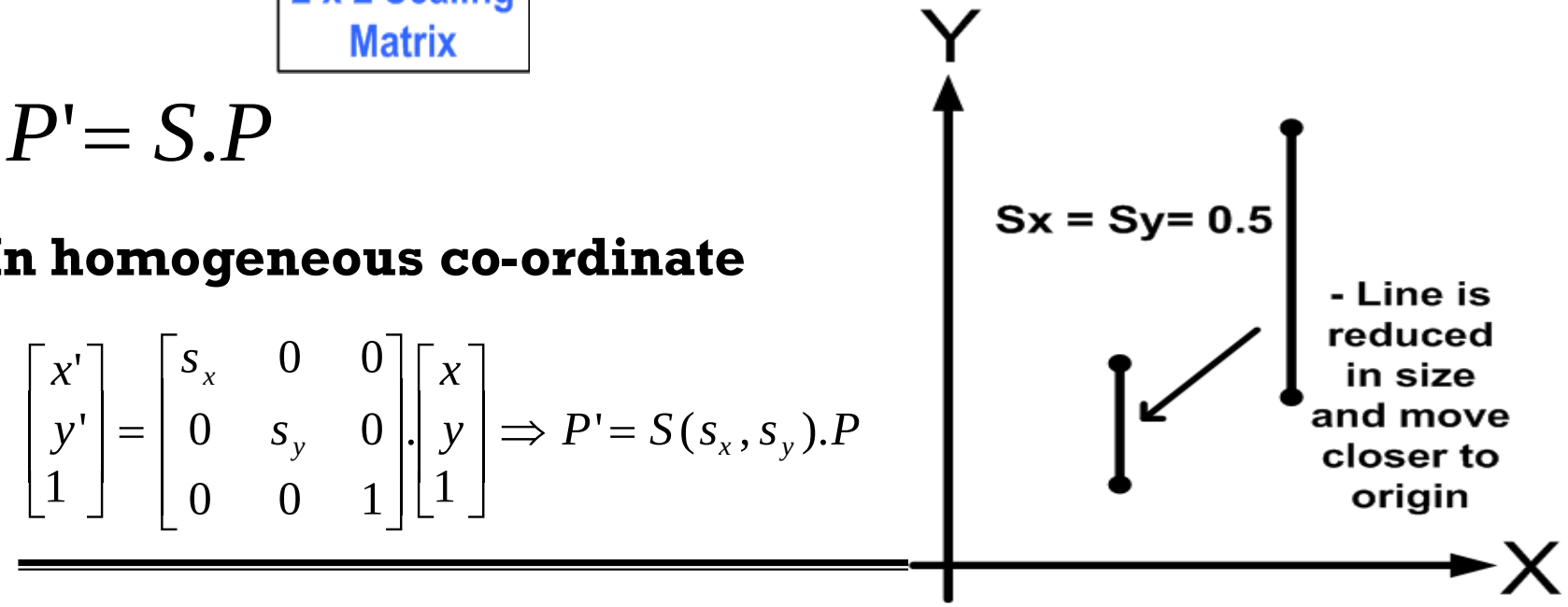
(Sx, Sy)→ Scaling factors

Sx=Sy→ Uniform Scaling

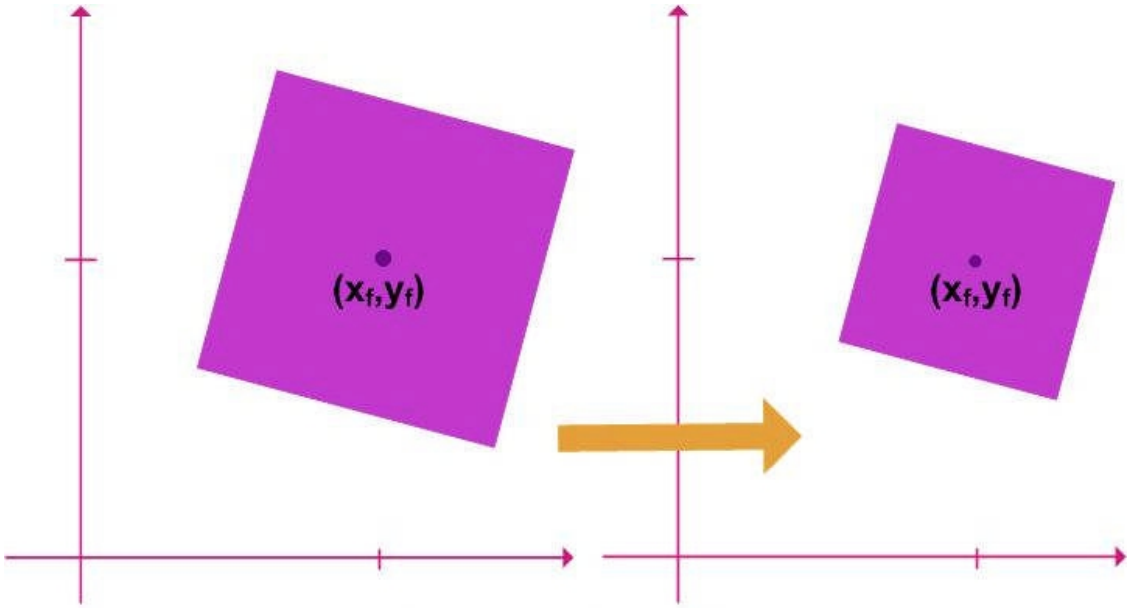
Sx≠ Sy → Differential Scaling

In homogeneous co-ordinate

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow P' = S(s_x, s_y).P$$



FIXED POINT SCALING

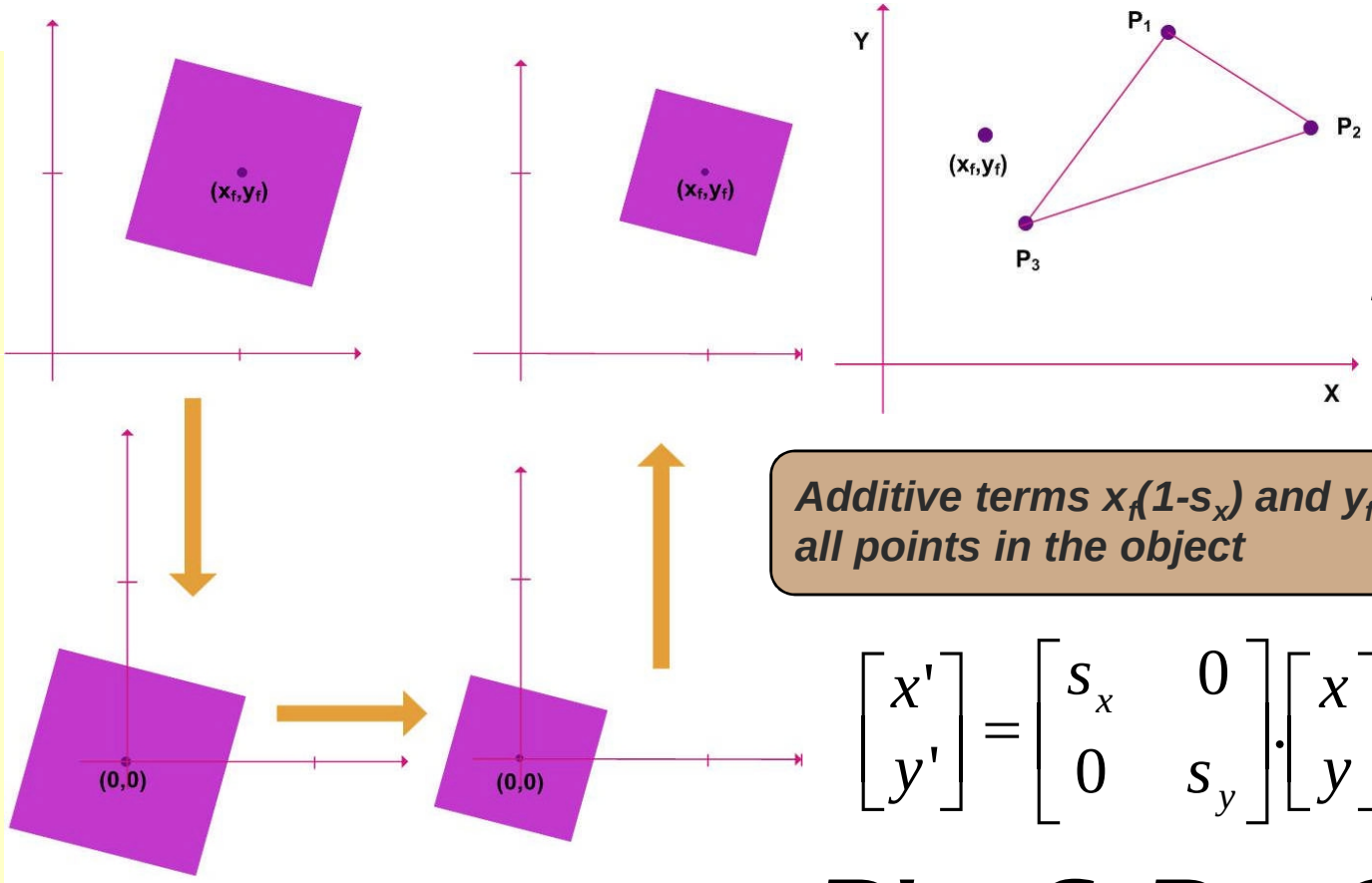


How is Done ?

For a given fixed point (x_f,y_f)

1. Shift the scaling point to origin
2. Scale
3. Restore

FIXED POINT SCALING contd…



$$x' = x_f + (x - x_f).s_x,$$
$$y' = y_f + (y - y_f).s_y$$

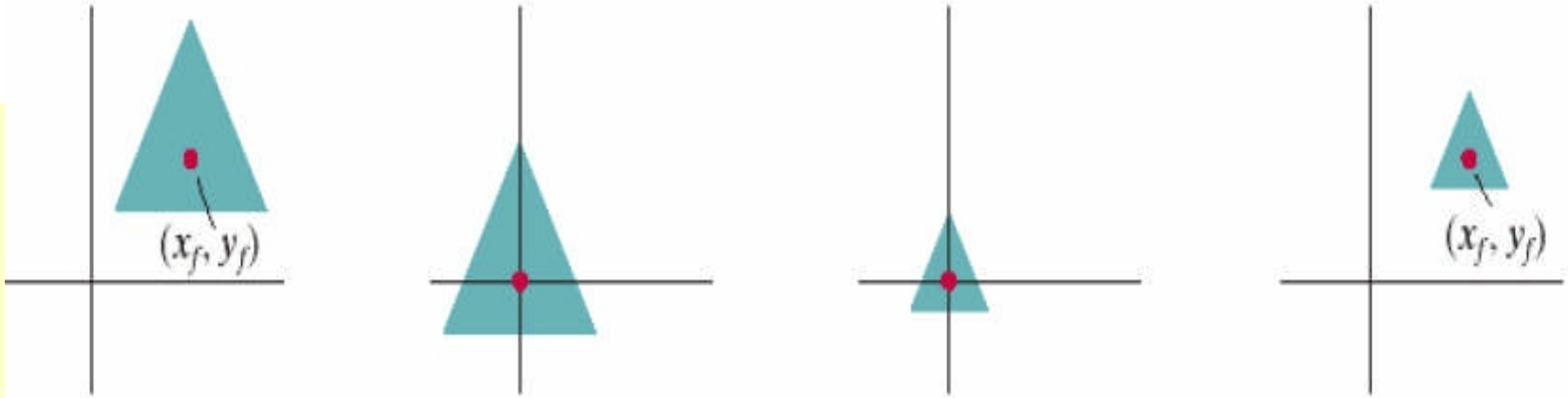
$$x' = x.s_x + x_f(1 - s_x)$$
$$y' = y.s_y + y_f(1 - s_y)$$

Additive terms $x_f(1-s_x)$ and $y_f(1-s_y)$ are constant for all points in the object

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} x_f(1 - s_x) \\ y_f(1 - s_y) \end{bmatrix}$$

$$\boldsymbol{P' = S.P + C}$$

Fixed Point Scaling contd...



$$\mathbf{T}(x_f, y_f) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}(-x_f, -y_f) = \mathbf{S}(x_f, y_f, s_x, s_y)$$

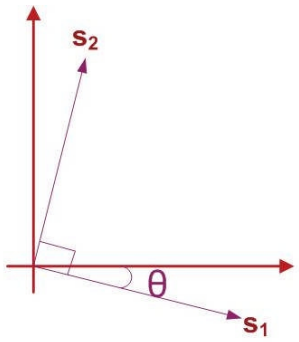
$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

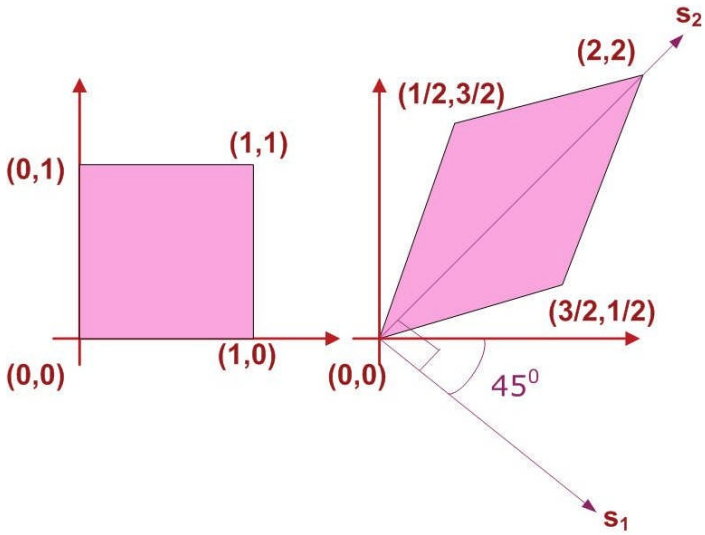
GENERAL SCALING DIRECTIONS

- Rotate the object so that x and y axes coincide with s_1 and s_2
- Apply scaling transformation in s_1 and s_2 direction
- Rotate the object in opposite direction to return points to the original orientations
- Example: square converted to parallelogram with s_1=1 and s_2 = 2 and θ = 45⁰ as shown in figure

$$R^{-1}(\theta).S(s_1, s_2).R(\theta)$$

$$= \begin{bmatrix} s_1\cos^2\theta + s_2\sin^2\theta & (s_2 - s_1)\cos\theta.\sin\theta & 0 \\ (s_2 - s_1)\cos\theta.\sin\theta & s_1\sin^2\theta + s_2\cos^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$





REFLECTION

- A transformation that produces a mirror image of an object.
- Equivalent to rotation of an object about the reflection axis.
- Reflection about x-axis (y=0)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

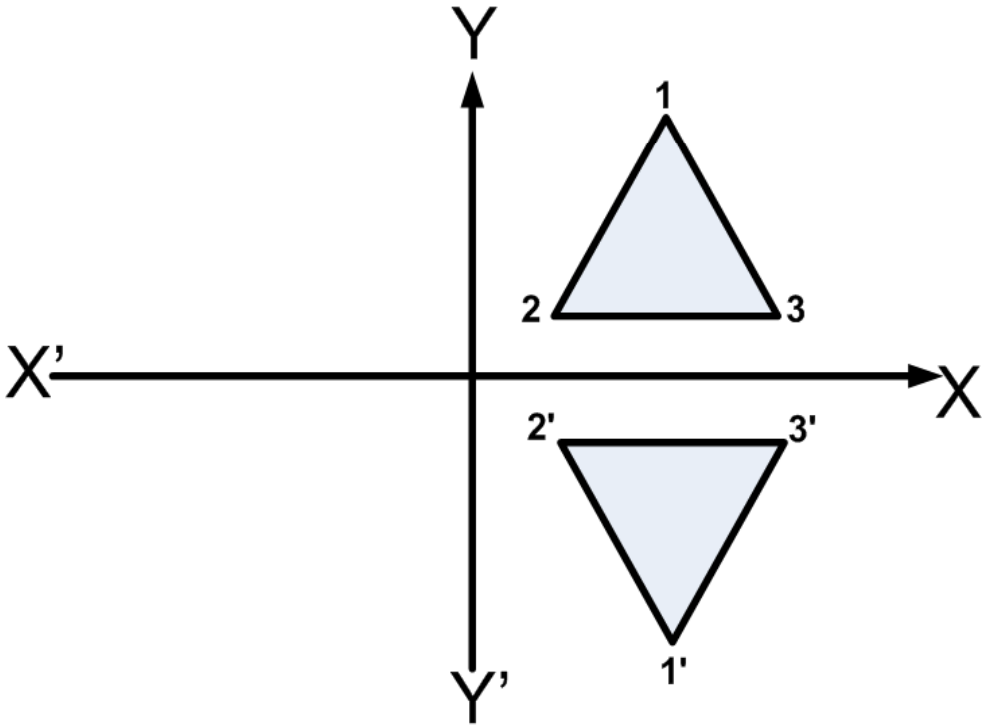


Fig: Reflection of object about x-axis (y=0)

REFLECTION contd…

- Reflection about y-axis (x=0)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

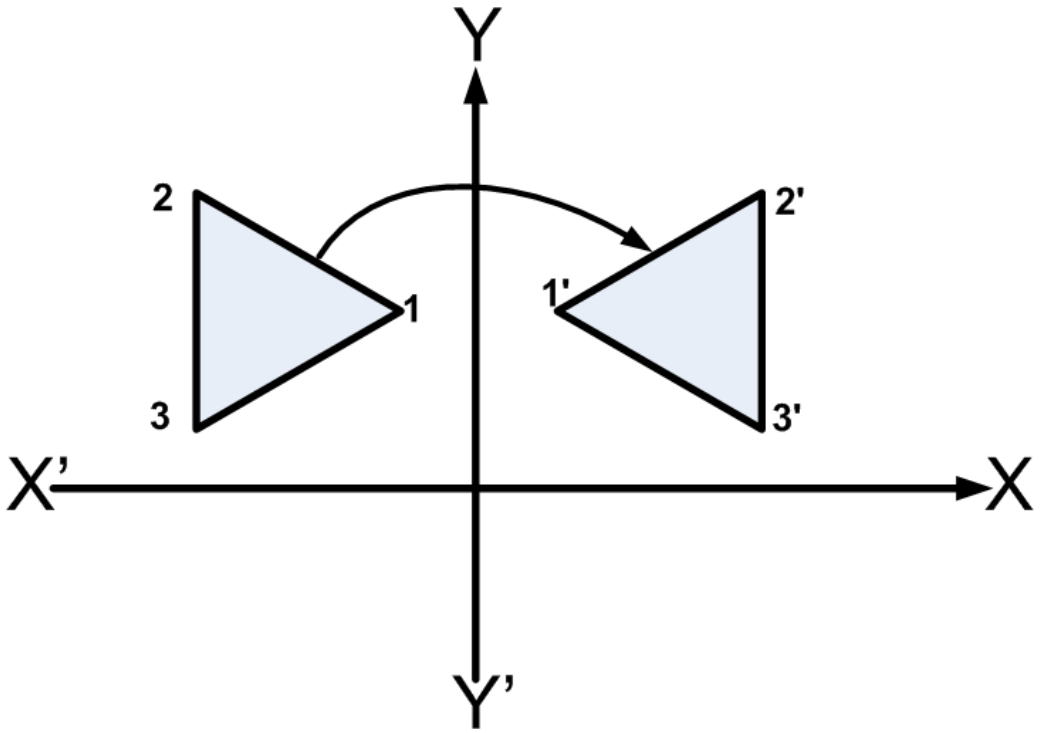


Fig: Reflection of object about y-axis (x=0)

REFLECTION contd…

- **Reflection about y = x**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Equivalent to:**
  - Reflection about x-axis
  - Rotate anticlockwise **90°**

  OR

  - Clockwise rotation 45 °
  - Reflection with x-axis
  - anticlockwise rotation 45 °

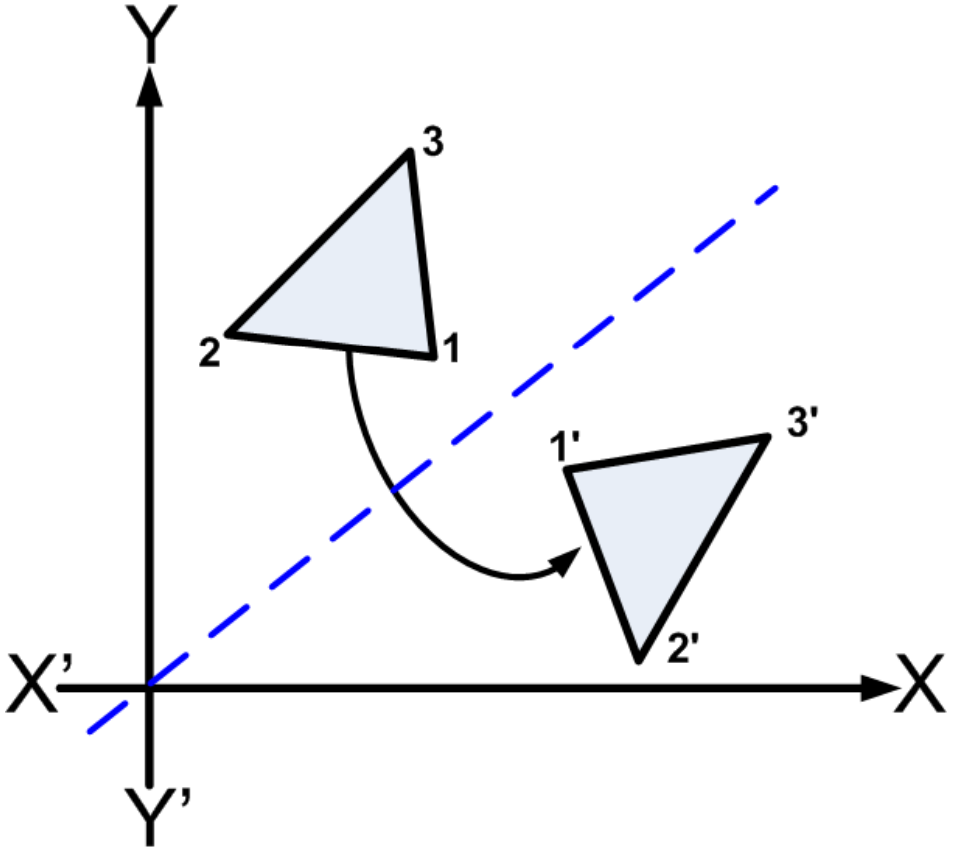


Fig: Reflection of object about y = x

REFLECTION contd…

- **Reflection about y = - x**

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

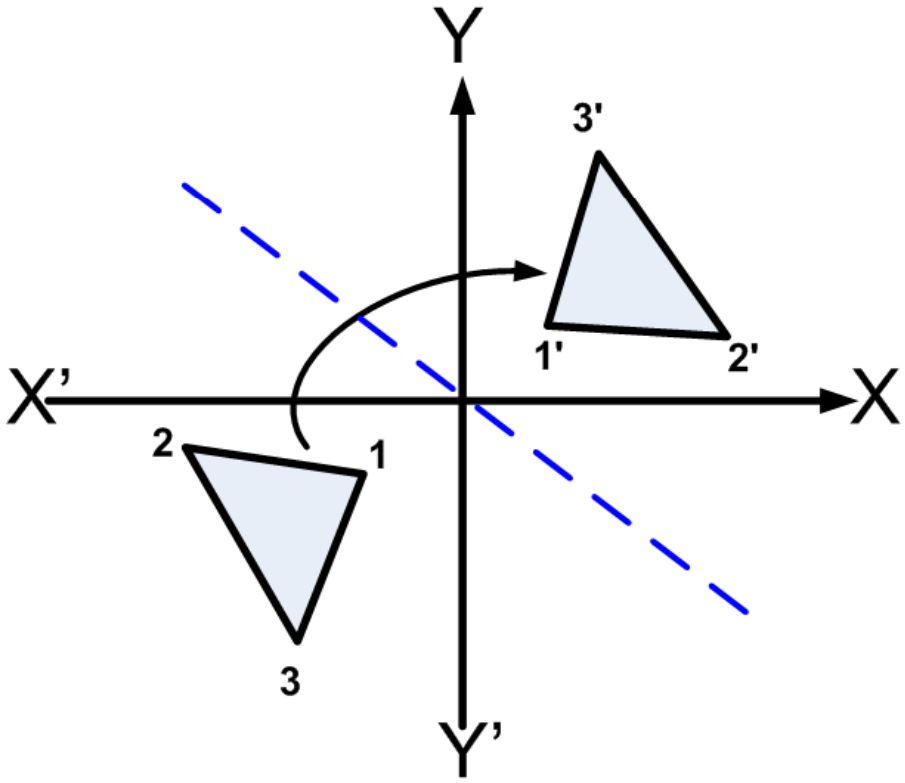
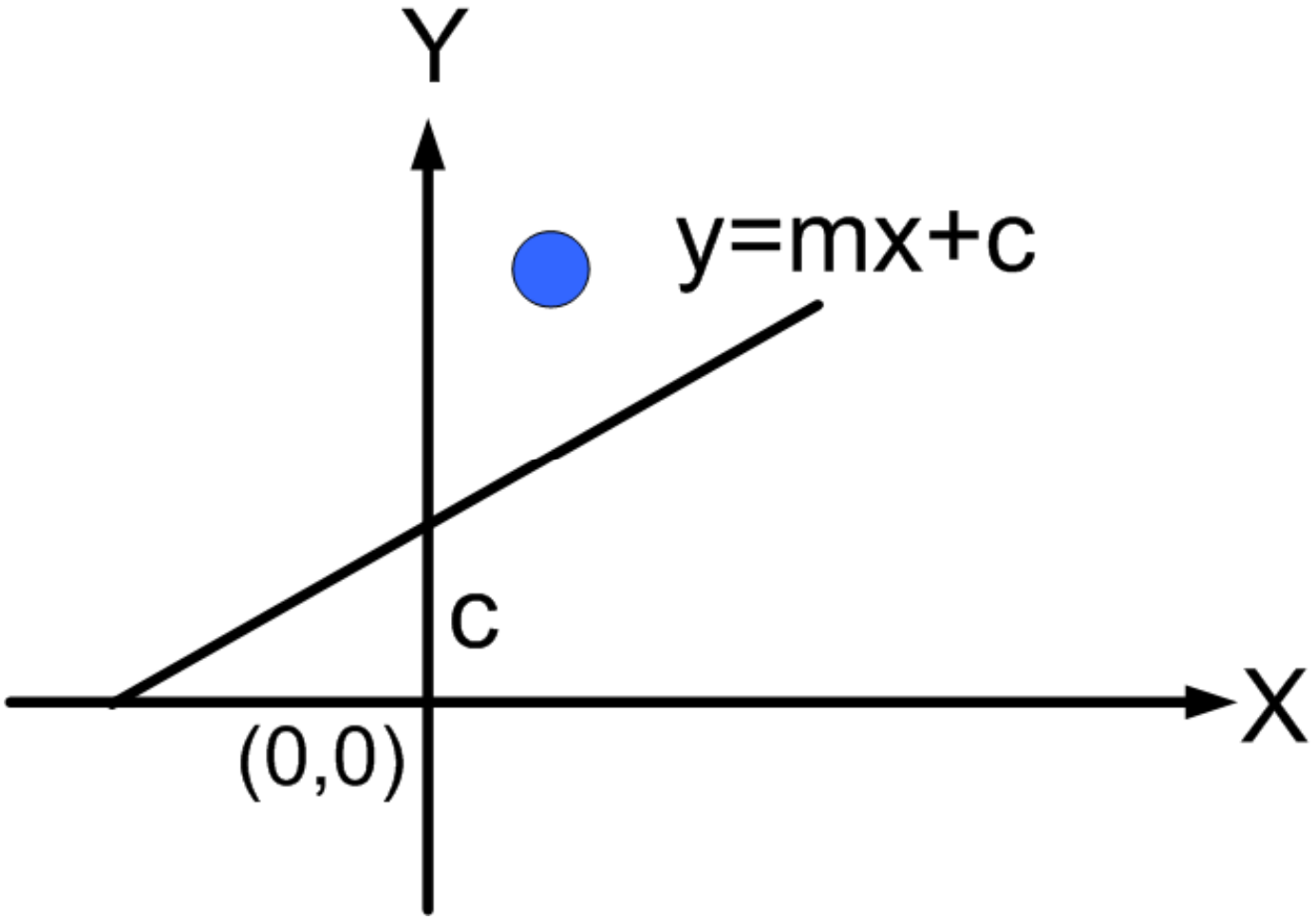


Fig: Reflection of object about y = - x

- **Equivalent to:**
  - Clockwise rotation 45 °
  - Reflection with y-axis
  - anticlockwise rotation 45 °

REFLECTION ABOUT y=m*x+c



ASSIGNMENT

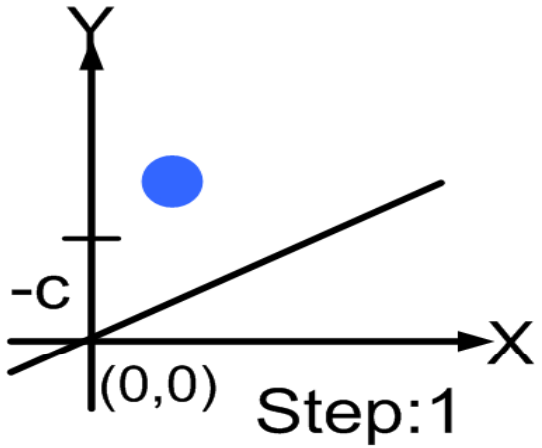
REFLECTION ABOUT y=mx+c

- Combination of translate-rotate-reflect transformation

1. First translate the line so that it passes through the origin

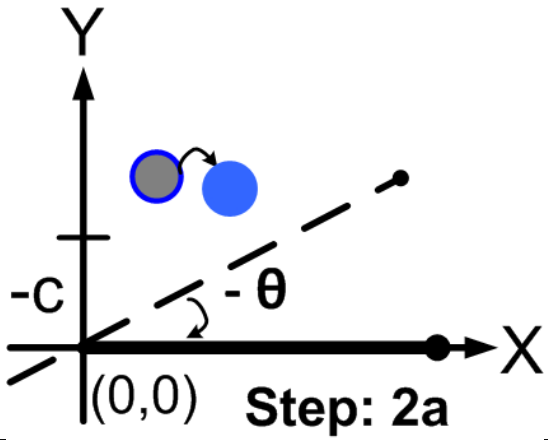
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix}$$

translatio n



2. Rotate the line onto one of the coordinate axes(say x-axis) and reflect about that axis (x-axis)

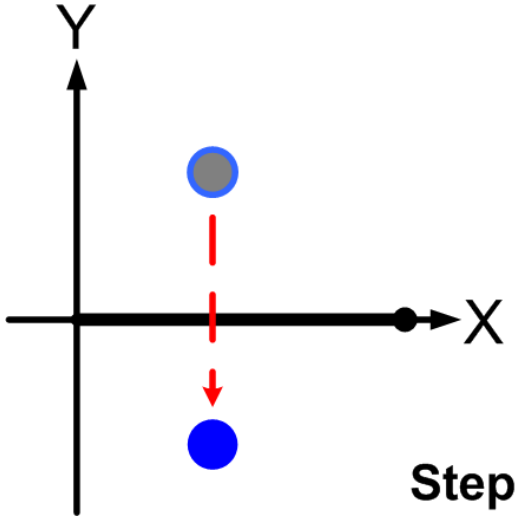
$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} --- \textit{rotation}$$



REFLECTION ABOUT y=mx+c contd…

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

reflection

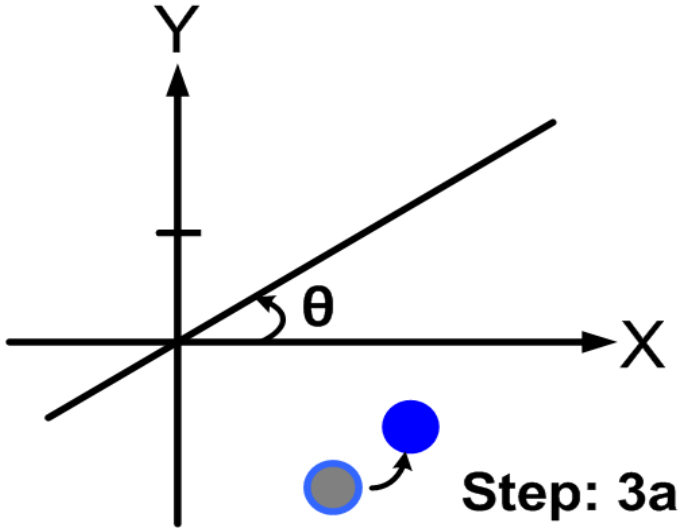


Step: 2b

3. Finally, restore the line to its original position with the inverse rotation and translation transformation.

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

rotation

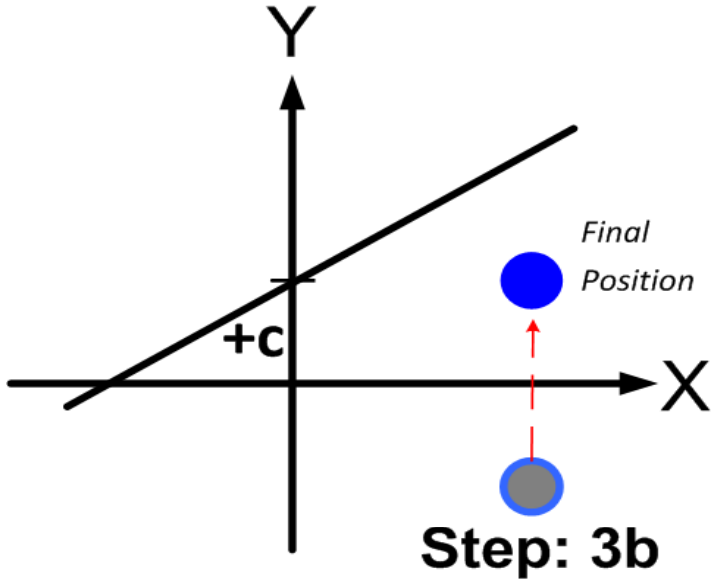


Step: 3a

REFLECTION ABOUT y=m*x+c contd…

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix}$$

translation



- Thus the composite transformation matrix for reflection about y=m*x+c is

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

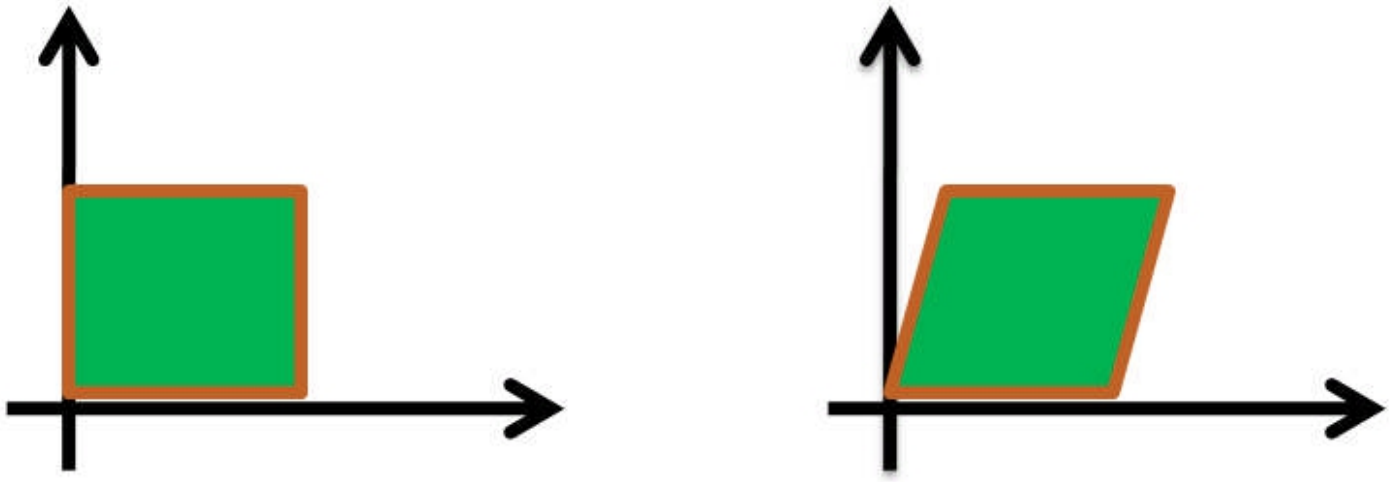
REFLECTION ABOUT y=m*x+c contd…

- **Substituting value of tanθ , sinθ & cosθ you will get the reflection matrix**

$$\begin{bmatrix} \frac{1-m^2}{1+m^2} & \frac{2m}{1+m^2} & \frac{-2cm}{1+m^2} \\ \frac{2m}{1+m^2} & \frac{m^2-1}{1+m^2} & \frac{2c}{1+m^2} \\ 0 & 0 & 1 \end{bmatrix}$$

SHEARING

- Transformation that distorts the shape of an object
- Transformed object appears as if it were composed of internal layers that had been caused to slide over each other.
- Shearing factor (Sh_x, Sh_y)



SHEARING contd…

x- direction shear relative to x-axis

x'=x+y*Shx

y'=y

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

y- direction shear relative to y-axis

x'=x

y'=y +x *Shy

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

SHEARING contd…

x- direction shear relative to other reference line $y=y_{ref}$

$x' = x + Shx(y - y_{ref})$

$y' = y$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Shx & -Shx \cdot y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

y- direction shear relative to other reference line $x=x_{ref}$

$x' = x$

$y' = y + Shy(x - x_{ref})$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Shy & 1 & -Shy \cdot X_{ref} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

COMPOSITE TRANSFORMATIONS

For Successive Translation vectors (tx1,ty1) and (tx2,ty2)

$$P' = T(t_{x2}, t_{y2}).\{T(t_{x1}, t_{y1}).P\}$$
$$= \{T(t_{x2}, t_{y2}).T(t_{x1}, t_{y1})\}.P$$

For Successive Rotations θ_1 and θ_2

$$P' = R(\theta_2).\{R(\theta_1).P\}$$
$$= \{R(\theta_2).R(\theta_1)\}.P$$

Successive Translations are additive

$$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

$$T(t_{x2}, t_{y2}).T(t_{x1}, t_{y1}) = T(t_{x1}+t_{x2}, t_{y1}+t_{y2})$$

Successive Rotations are additive.

$$R(\theta_2).R(\theta_1) = R(\theta_1+\theta_2)$$

!!!!!!! PROVE YOURSELF !!!!!!!!!

COMPOSITE TRANSFORMATIONS

Successive Scaling are multiplicative

$$S(s_{x2}, s_{y2}).S(s_{x1}, s_{y1}) = S(s_{x1}.s_{x2}, s_{y1}.s_{y2})$$

$$\begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x1}.s_{x2} & 0 & 0 \\ 0 & S_{y1}.s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Fixed Point Scaling

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

$$T(x_f, y_f).S(s_x, s_y).T(-x_f, -y_f) = S(x_f, y_f, s_x, s_y)$$

Concatenation Properties

Matrix multiplication is associative, so

A.B.C = (A.B).C = A.(B.C)

Transformation product is not commutative, so

A.B ≠ B.A

TRANSFORMATION BETWEEN COORDINATE AXES

1. Translate so that the origin (x_0, y_0) of the x'y' system is moved to the origin of the xy system
2. Rotate the x'-axis onto the x-axis

$$T(-x_0,-y_0)=\begin{bmatrix}1 & 0 & -x_0\\ 0 & 1 & -y_0\\ 0 & 0 & 1\end{bmatrix}$$

$$R(-\theta)=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$$

So , the composite matrix

$$T.R=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & -x_0\\ 0 & 1 & -y_0\\ 0 & 0 & 1\end{bmatrix}$$

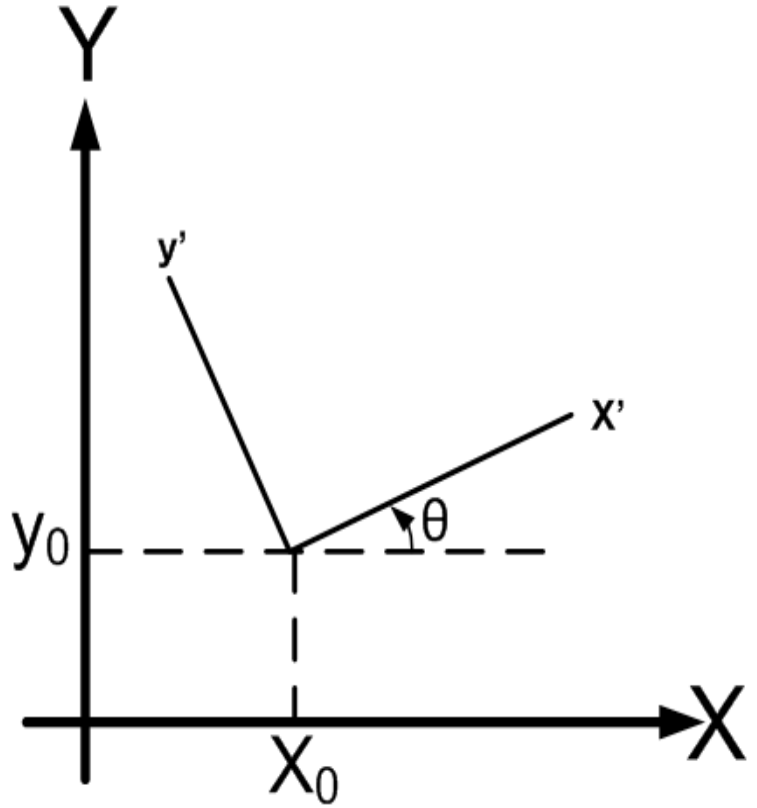


Fig: A cartesian system x'y' system positioned at (x_0,y_0) with orientation θ in xy cartesian system

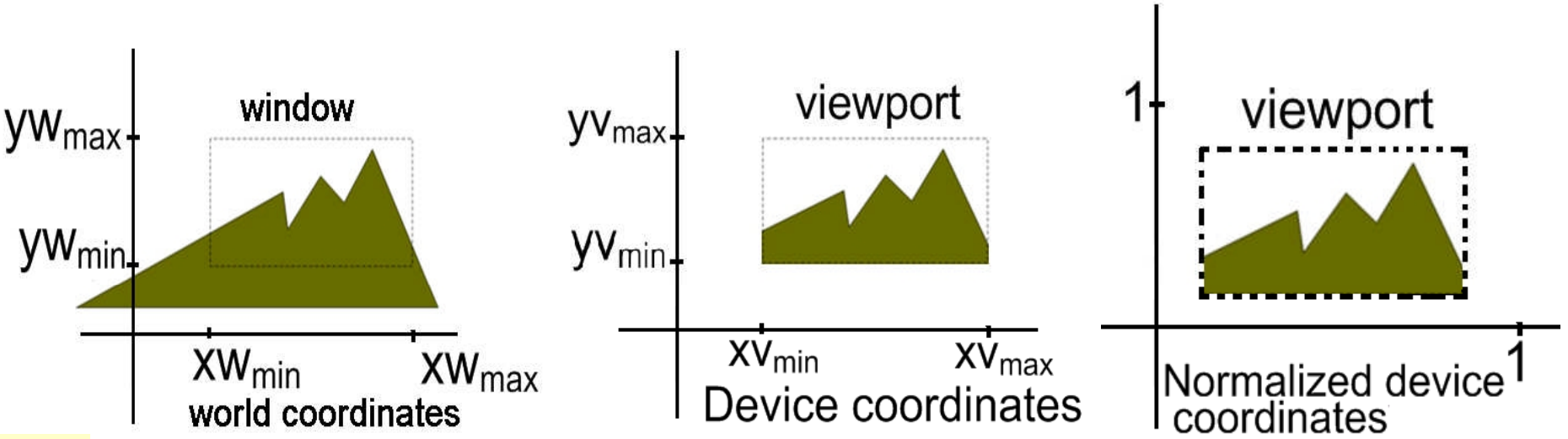
2-D VIEWING

2-D VIEWING

- **Window**
  - - a world coordinate area selected for display
  - define what is to be viewed
- **View-port**
  - An area on a display device to which a window is mapped
  - Define where it is to be displayed
- **Windows and view-port**
  - Rectangle in standard positions, with rectangle edges parallel to coordinate axes
  - Other geometric takes longer to proceed
- **Viewing transformation-** the mapping of a world coordinate scene to device coordinates
- **Transformation pipeline-**
  - takes the object coordinates through several intermediate coordinates systems before finishing with device coordinates

VIEWING TRANSFORMATION

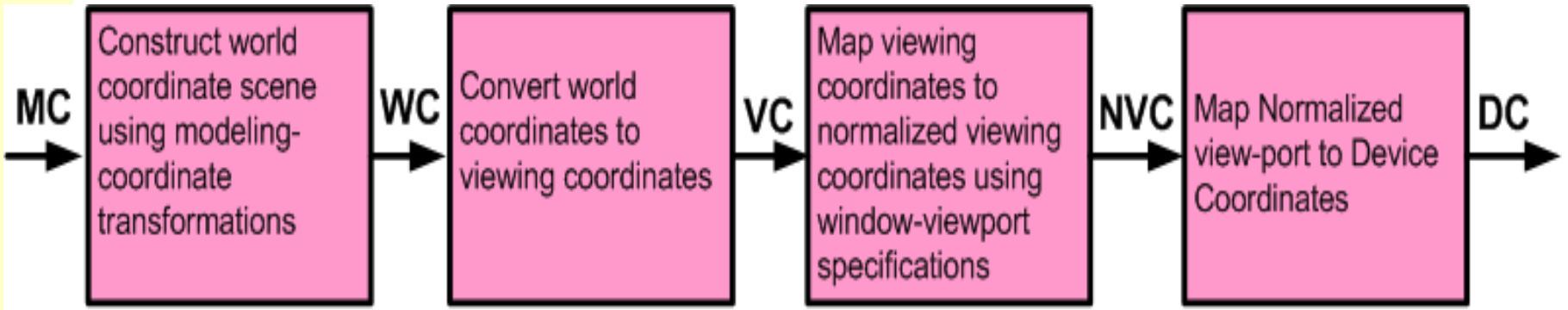
- **Transformation from world to viewport**



- **For Fixed sized viewport, zooming in effect is attained if window size is decreased and zooming out effect if window size is increased.**
- **Panning effect is attained by successively changing the position of viewing window**
- **The normalized device coordinates maps the world co-ordinate to the viewport of maximum size = unit square as shown in figure. The advantage is the mapping is display device independent**

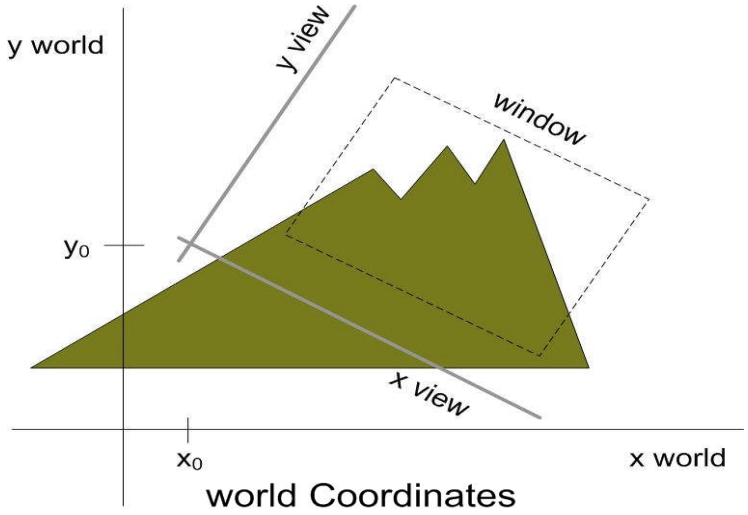
2-D VIEWING TRANSFORMATION PIPELINE

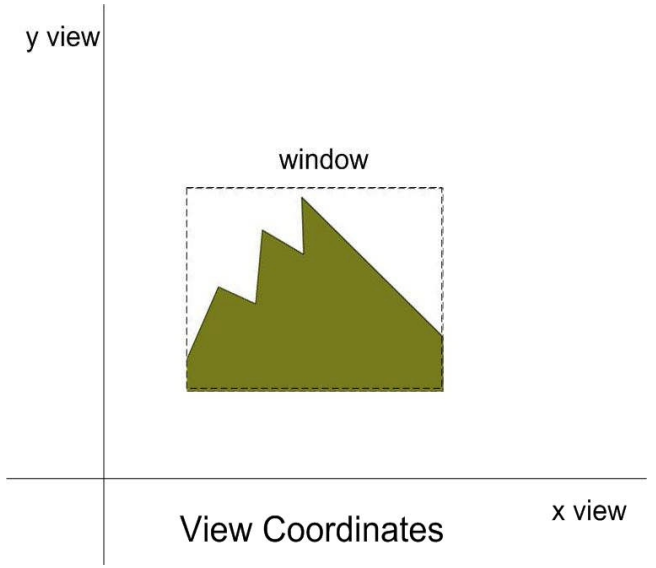
- Procedures for displaying views of a two-dimensional picture on an output device:
  - Specify which parts of the object to display (clipping window, or world window, or viewing window)
  - Where on the screen to display these parts (viewport).
- Clipping window is the selected section of a scene that is displayed on a display window.
- Viewport is the window where the object is viewed on the output device.

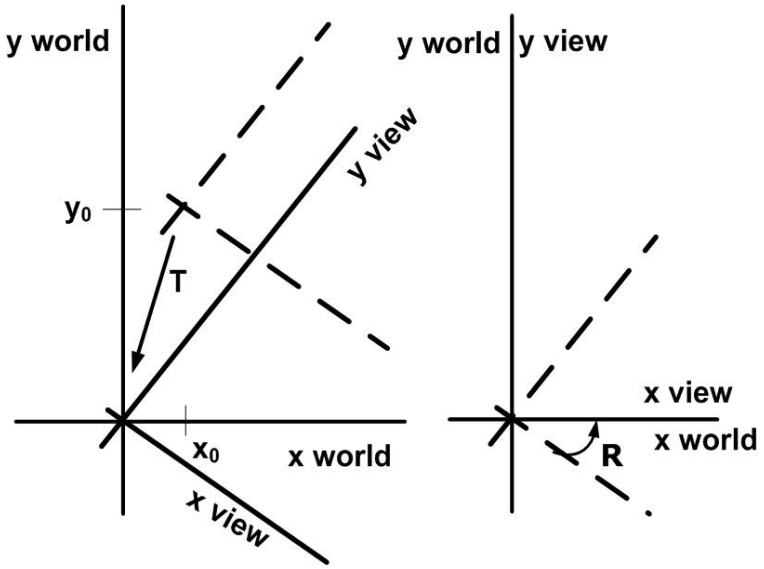


MC-Modeling Coordinates, WC-world Coordinates
VC-Viewing Coordinates NVC- Normalized Viewing Coordinates
DC- Device Coordinates

VIEWING REFERENCE FRAME







Setting up viewing Reference Frame

The matrix is obtained in two steps:

1. Translate Viewing reference frame origin to world origin
2. Rotate Viewing reference frame to coincide with world coordinate axes

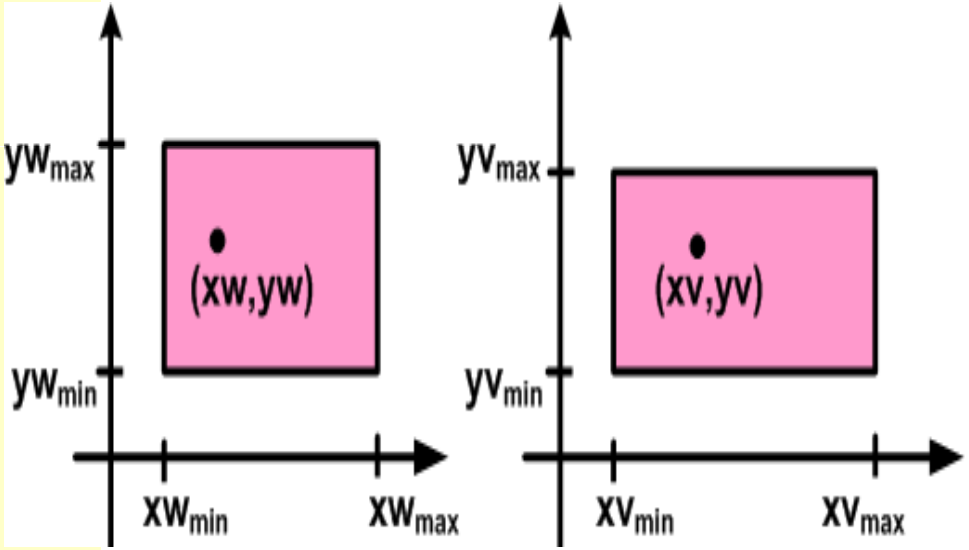
Thus we get the matrix as

$$\mathbf{M_{wc,vc} = R.T}$$

Windows To Viewport Co-ordinate Transformation

window to viewport coordinate transformation maintains the same relative placement of objects in normalized space as in viewing cordinates

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}} \quad and \quad \frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$



Solving we get,

$$xv = xv_{min} + (xw - xw_{min})sx$$

$$yv = yv_{min} + (yw - yw_{min})sy$$

where scaling factors

$$sx = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$sy = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

If sx = sy, the proportion is maintained otherwise the scene is stretched

CLIPPING

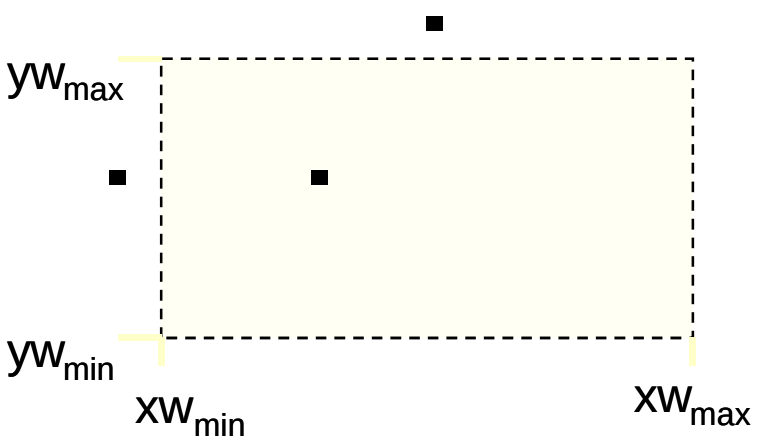
CLIPPING

- Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of a space
- A **clip window** can be polygon or curved boundaries
- World- coordinate clipping removes the primitives outside the window from further consideration; thus eliminating the processing necessary to transform these primitives to device space.
- **Clipping type-** point ,line, area, curve and text clipping

- **APPLICATIONS**
  - Extracting part of a defined scene for viewing
  - Identifying visible surfaces in 3-dimensional views
  - Antialiasing line segments or object boundaries
  - Creating objects using solid modeling procedures
  - Displaying a multiwindow environment
  - Drawing and painting operations that allow parts of a picture to be selected for copying, moving erasing or duplication.

POINT CLIPPING

Assume clipping window is Rectangle, point P=(x,y) is saved for display if following inequalities are satisfied



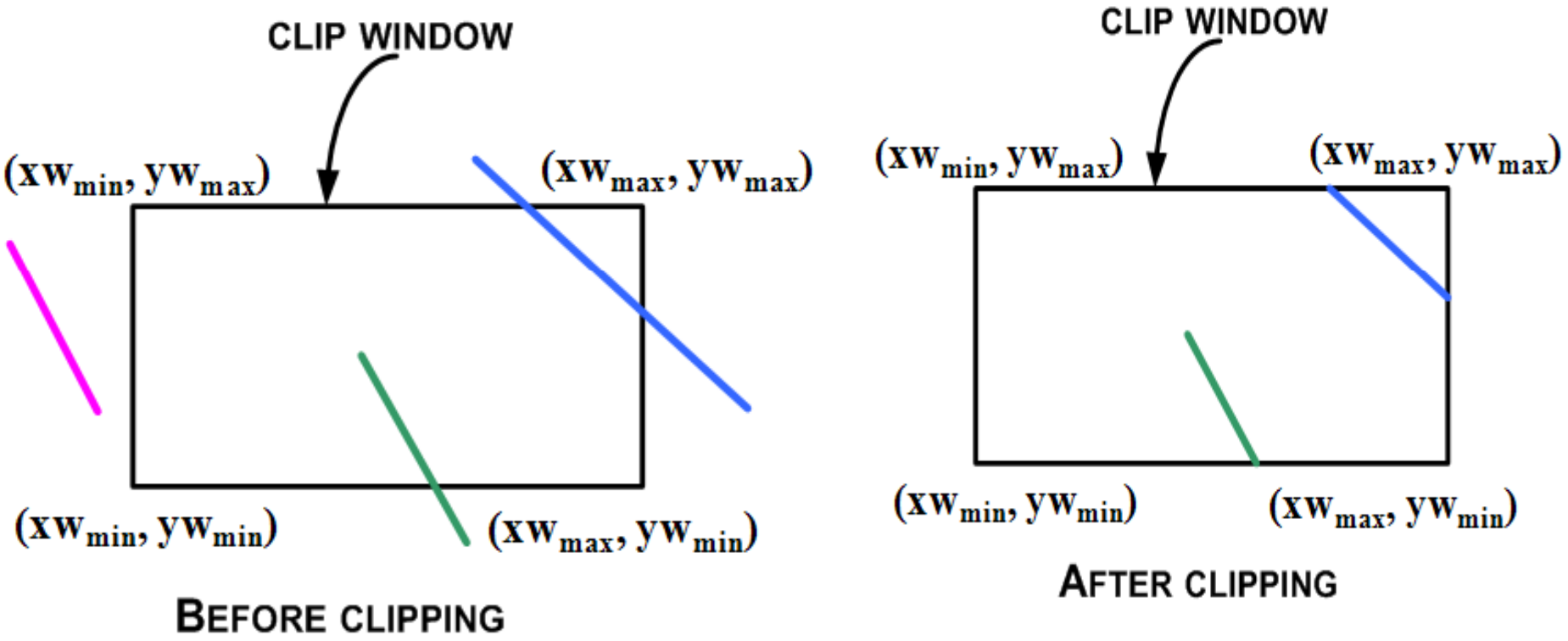
$$xw_{min} \leq x \leq xw_{max}$$
$$yw_{min} \leq y \leq yw_{max}$$

$(xw_{min}, xw_{max}, yw_{min}, yw_{max})$ → can be either world coordinate window boundaries or viewport boundaries

If all the four inequalities are satisfied for a point with co-ordinate (x,y), the point is accepted; i.e not clipped

LINE CLIPPING

"INSIDE – OUTSIDE" TESTS



What are the methods (algorithms) to perform clipping operations ?

LINE CLIPPING

- For a line segment with endpoints (x_1,y_1) and (x_2,y_2) and one or both endpoints outside the clipping rectangle, the parametric representation

$$x=x_1 + u(x_2- x_1)$$

$$y=y_1 + u(y_2- y_1) \qquad 0\leq u \leq 1$$

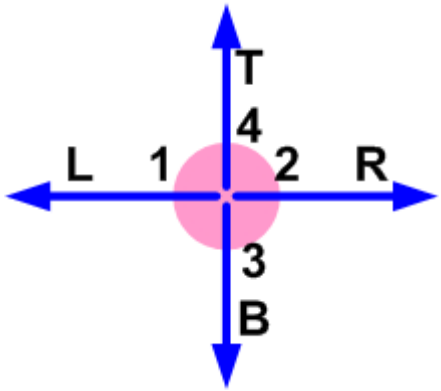
- If value of u for an intersection with a rectangle boundary edge is outside the range $0\rightarrow 1$, the line does not enter the interior of the window at that boundary
- If value of u is within the range $0\rightarrow 1$, the line segment does indeed enter into the clipping area

COHEN-SUTHERLAND LINE CLIPPING

- advantage of the algorithm is that it vastly reduces the number of line intersections that must be calculated
- World space is divided into regions based on the window boundaries
  - Each region has a unique **four bit region code**
  - Region codes indicate the position of the regions with respect to the window

4	3	2	1
Top	Below	Right	Left

TBRL



Region Code Legend

bit 4 : $y > yw_{max}$

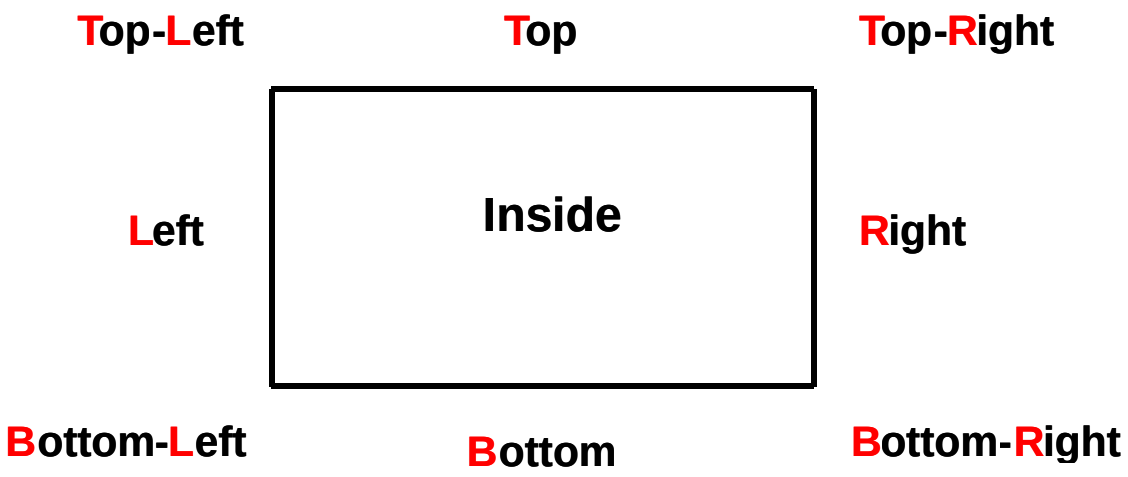
bit 3 : $y < yw_{min}$

bit 2 : $x > xw_{max}$

bit 1 : $x < xw_{min}$

1001	1000	1010
0001	0000 Window	0010
0101	0100	0110

Cohen-Sutherland line clipping



TBRL

1001	1000	1010
0001	0000 Window	0010
0101	0100	0110

Cohen-Sutherland line clipping

Region coding

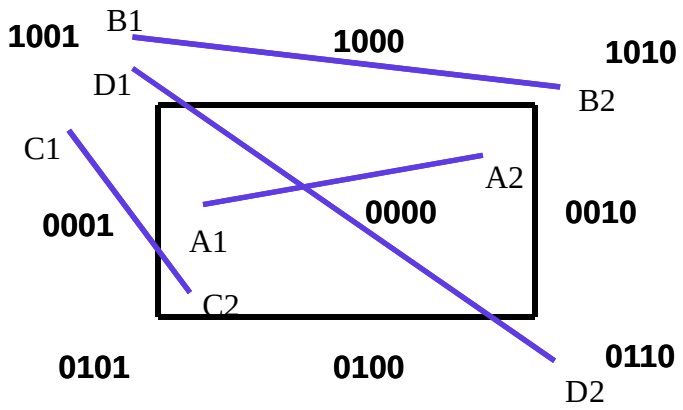
How would you decide which region an endpoint is in?

e.g

if ($x < xw_{min}$) && ($y > y_{max}$) → the point is at the **Top-Left**

Are there cases we can **trivially accept or reject**?

How would you test for those?



- **A1=0000,A2=0000**
  both (0000) → accept trivially

- **B1=1001,B2=1010**
  both NOT (0000)
  AND Operation
  B1 → 1001
  B2 → 1010
  Result 1000

  Result = NOT (0000) → reject trivially

ALGORITHM

1. Assign a region code for each endpoints.
2. If both endpoints have a **region code 0000** →**trivially accept** these line.
3. Else, perform the logical AND operation for both region codes.

   3.1 **if** the result is **NOT** 0000 → **trivially reject** the line.

   3.2 **else** (i.e. **result = 0000**, need clipping)

   3.2.1. Choose an endpoint of the line that is outside the window.

   3.2.2. Find the intersection point at the window boundary (based on region code).

   3.2.3. Replace endpoint with the intersection point and update the region code.

   3.2.4. Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.

4. Repeat step 1 for other lines.

Cohen-Sutherland line clipping

Intersection calculations:

Intersection with vertical boundary

$y = y_1 + m(x-x_1)$

Where

$x = xw_{min}$ or xw_{max}

Intersection with horizontal boundary

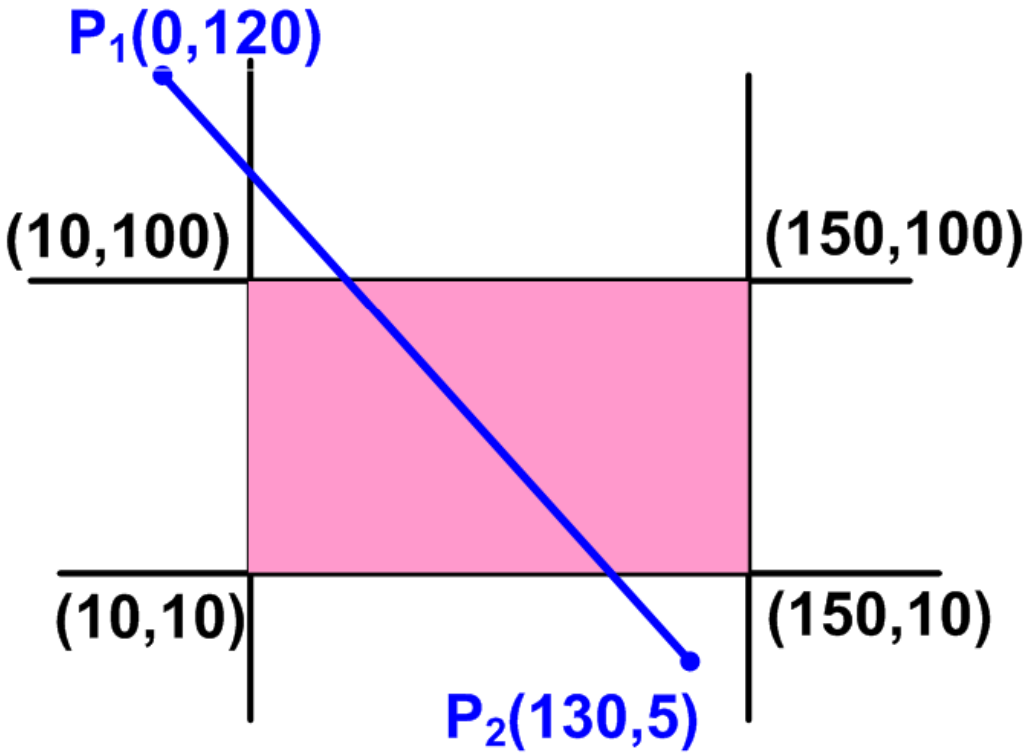
$x = x_1 + (y-y_1)/m$

Where

$y = yw_{min}$ or yw_{max}

COHEN-SUTHERLAND LINE CLIPPING contd…

- **Obtain the endpoints of line P_1P_2 after cohen-sutherland clipping**



ASSIGNMENT

COHEN-SUTHERLAND LINE CLIPPING →Numerical

1. $\mathbf{P_1}$=1001 $\mathbf{P_2}$=0100
2. Both (0000)→NO
3. And Operation

   1001

   0100

Result→0000

3.1 not(0000)→NO

3.2 0000→ yes

3.2.1 choose $\mathbf{P_1}$

3.2.2 Intersection with **LEFT** boundary

m=(5-120)/(130-0)= -0.8846

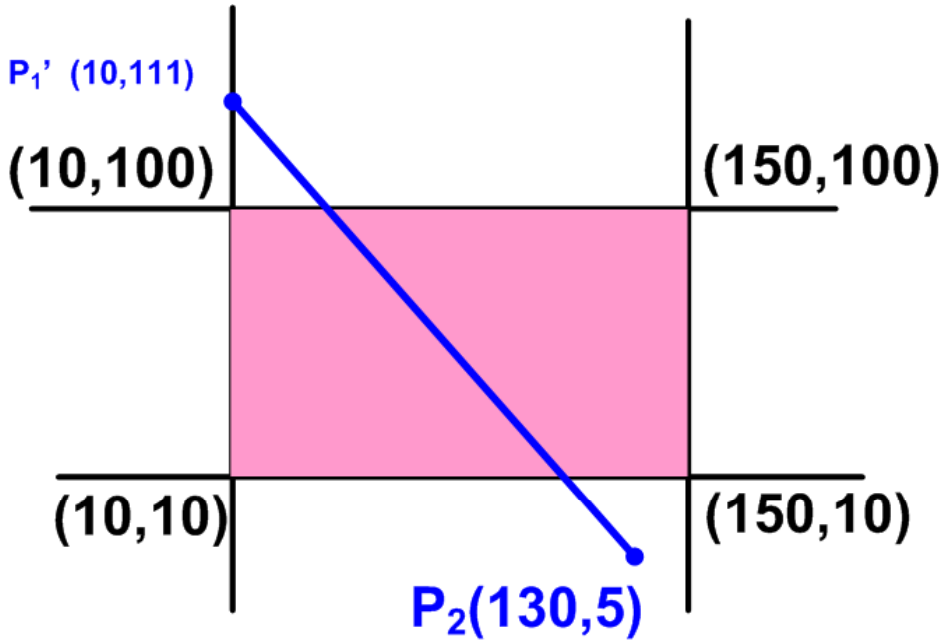
$y = y_1 + m(x-x_1)$ where x=10

y=120-0.8846(10-0)=111.15≈111

$\mathbf{P_1}$'= (10, 111)

3.2.3 Update region code $\mathbf{P_1}$'= 1000 **(TOP)**

3.2.4 Repeat step 2.



B

COHEN-SUTHERLAND LINE CLIPPING →Numerical

1. $\mathbf{P_1}'$=1000 $\mathbf{P_2}$=0100

2. Both (0000)→NO

And Operation

1000

0100

Result→0000

3.1 not(0000)→NO

3.2 0000→ yes

3.2.1 choose $\mathbf{P_1}'$

3.2.2 Intersection with **TOP** boundary

m=(5-120)/(130-0)= -0.8846

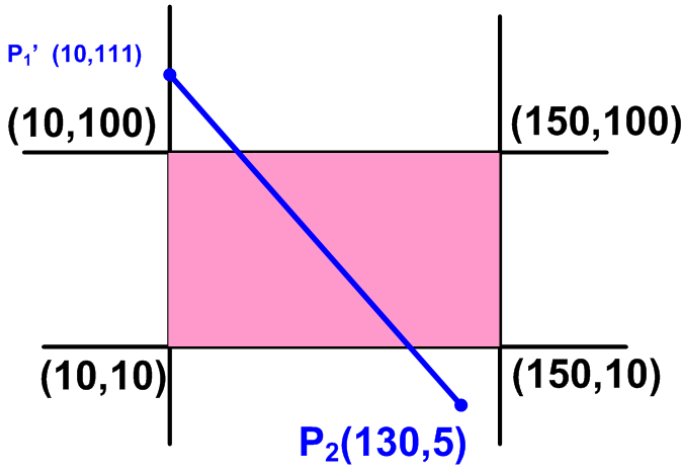
$x = x_1 + (y-y_1)/m$ where y= 100

x=10+ (100-111)/(-0.8846)=22.44 ≈22

$\mathbf{P_1}''$= (22, 100)

3.2.3 Update region code $\mathbf{P_1}''$= 0000

3.2.4 Repeat step 2.



COHEN-SUTHERLAND LINE CLIPPING →Numerical

1. $\mathbf{P_1}$''=0000 $\mathbf{P_2}$=0100

2. Both (0000)→NO

And Operation

0000

0100

Result→0000

3.1 not(0000)→NO

3.2 0000→ yes

3.2.1 choose $\mathbf{P_2}$

3.2.2 Intersection with **BOTTOM** boundary

m=(5-120)/(130-0)= -0.8846

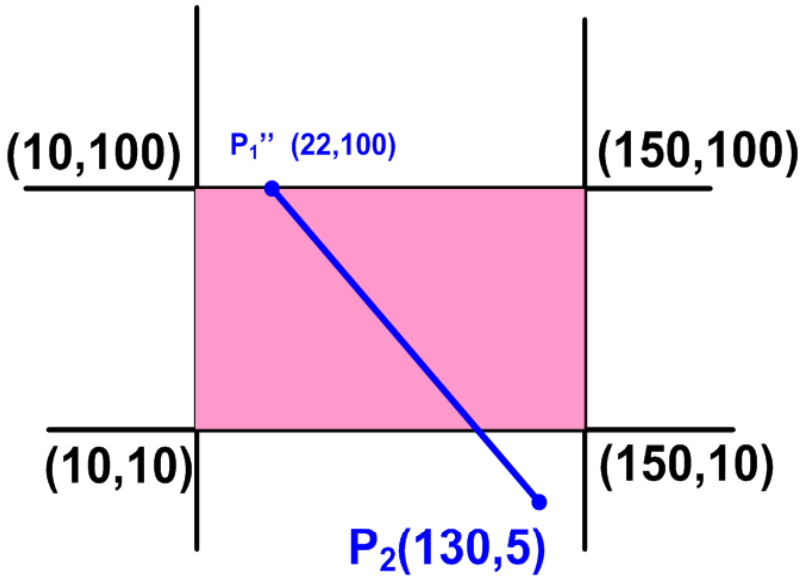
$x = x_1 + (y-y_1)/m$ where y= 10

x=130+ (10-5)/(-0.8846)=124.35 ≈124

$\mathbf{P_2}$'= (124, 10)

3.2.3 Update region code $\mathbf{P_2}$'= 0000

3.2.4 Repeat step 2.



COHEN-SUTHERLAND LINE CLIPPING →Numerical

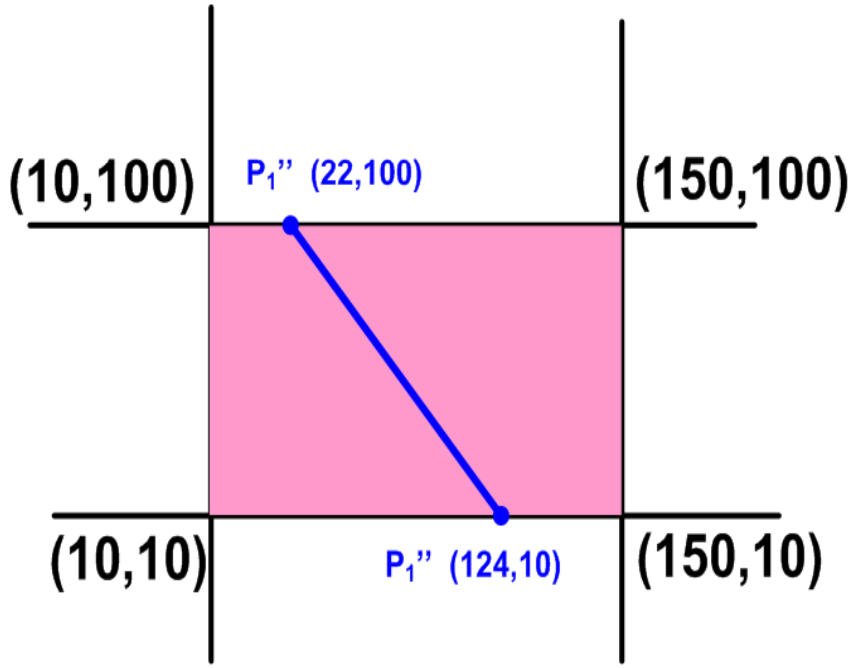
Both (0000)→YES

ACCEPT & DRAW

Thus endpoints after clipping

$\mathbf{P_1}''$= (22, 100)

$\mathbf{P_2}'$= (124, 10)



Liang-Barsky Line Clipping

- Based on parametric equation of a line:

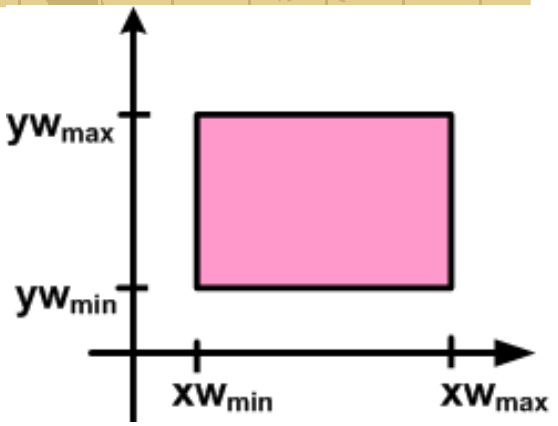
$$x = x_1 + u.\triangle x$$

$$y = y_1 + u.\triangle y$$

- Similarly, the clipping window is represented by:

$$xw_{min} \le x_1 + u.\triangle x \le xw_{max} \qquad 0 \le u \le 1$$

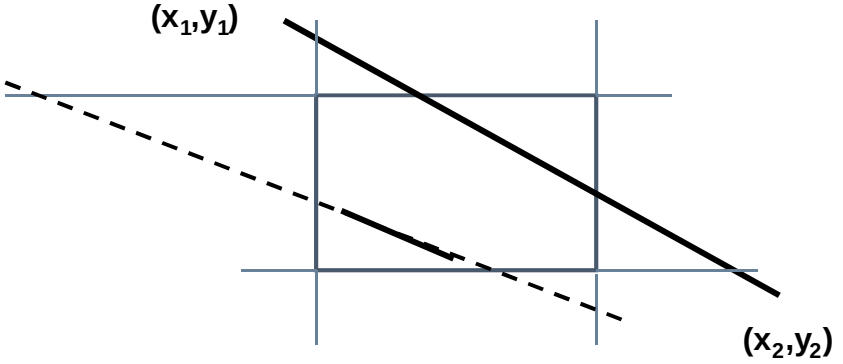
$$yw_{min} \le y_1 + u.\triangle y \le yw_{max}$$



... or,

$$u.\ p_k \le q_k \qquad k = 1, 2, 3, 4$$

where:



k = 1 (is the line inside left boundary ?) $p_1 = -\triangle x$, $q_1 = x_1 - xw_{min}$

k = 2 (is the line inside right boundary ?) $p_2 = \triangle x$, $q_2 = xw_{max} - x_1$

k = 3 (is the line inside bottom boundary ?) $p_3 = -\triangle y$, $q_3 = y_1 - yw_{min}$

k = 4 (is the line inside top boundary ?) $p_4 = \triangle y$, $q_4 = yw_{max} - y_1$

$P_k < 0 \rightarrow$ infinite extension of the line proceeds from outside to inside the infinitely extended boundary

$P_k > 0 \rightarrow$ infinite extension of the line proceeds from inside to outside of the infinitely extended boundary

Liang-Barsky Line Clipping contd…

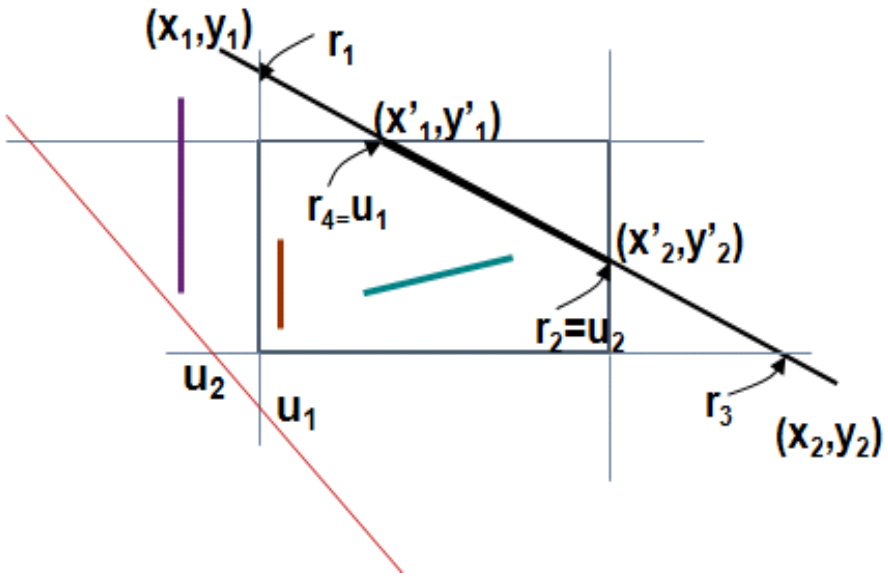
➢ **Trivial rejection :**

Reject line with $p_k = 0$ for some k and one $q_k < 0$ for these k.

➢clipped line will be

$$x_1' = x_1 + u_1 . \triangle x \quad u_1 \geq 0$$
$$y_1' = y_1 + u_1 . \triangle y$$

$$x_2' = x_1 + u_2 . \triangle x \quad u_2 \leq 1$$
$$y_2' = y_1 + u_2 . \triangle y$$



➢Calculate

$$u_k = \frac{q_k}{p_k}$$

u_1 --(For intersection with the boundaries to which line enters the boundary)
→**maximum value between 0 and u** (for $p_k < 0$), where starting value of u_1**=0**

u_2 --(For intersection with the boundaries to which line leaves the boundary)
→**minimum value between u and 1** (for $p_k > 0$), where starting value of u_2**=1**

COMPUTER GRAPHICS

EG678EX

Deel Mani Baral

deelmanibaral@gmail.com

Institute of Engineering

Pulchowk Campus

CHAPTER 6- THREE DIMENSIONAL TRANSFORMATIONS

“The art of 3D graphics is the art of fooling the

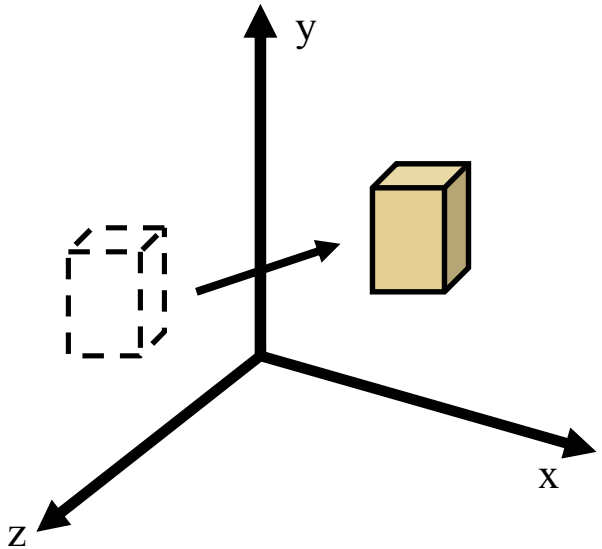
brain into thinking

that it sees a 3D object painted on a flat screen.”

3D Translation

- **Translation of a Point**

$$x' = x + t_x, \quad y' = y + t_y, \quad z' = z + t_z$$



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = T.P$$

3D Rotation

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_z(\theta).P$$

Z-Axis Rotation

$$y' = y\cos\theta - z\sin\theta$$
$$z' = y\sin\theta + z\cos\theta$$
$$x' = x$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_x(\theta).P$$

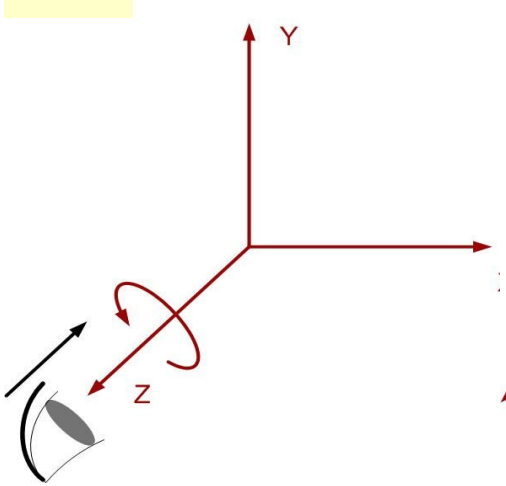
X-Axis Rotation

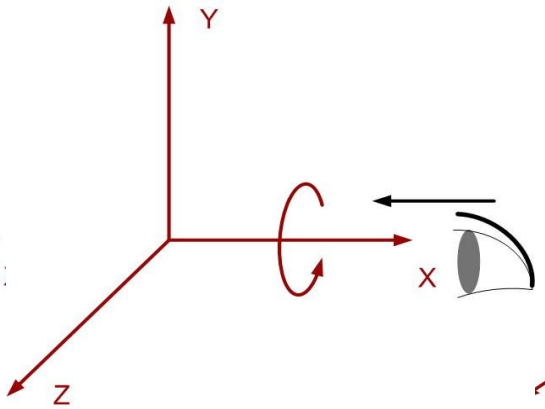
$$z' = z\cos\theta - x\sin\theta$$
$$x' = z\sin\theta + x\cos\theta$$
$$y' = y$$

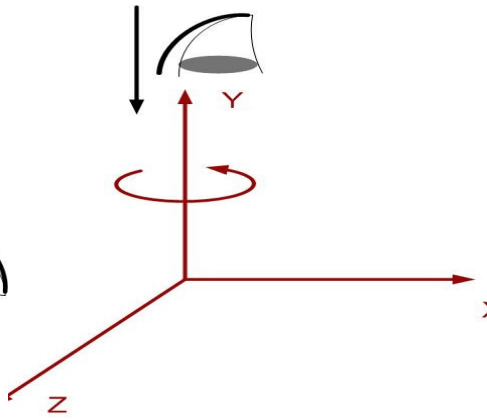
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$P' = R_y(\theta).P$$

Y-Axis Rotation

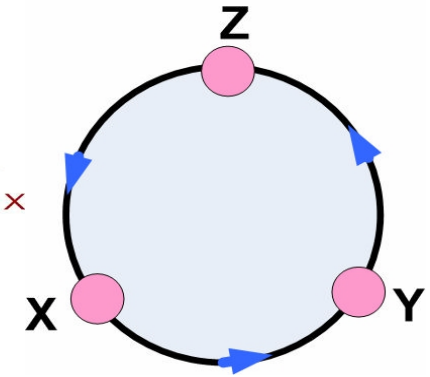






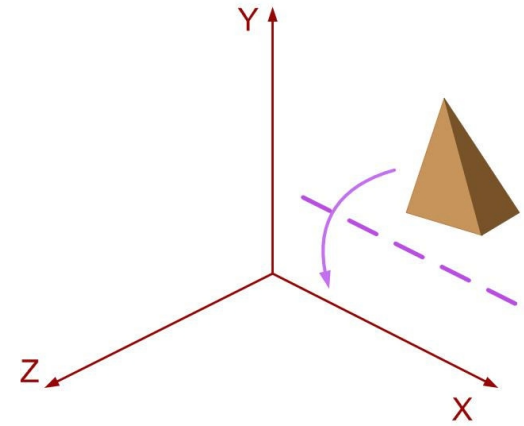
cyclic order

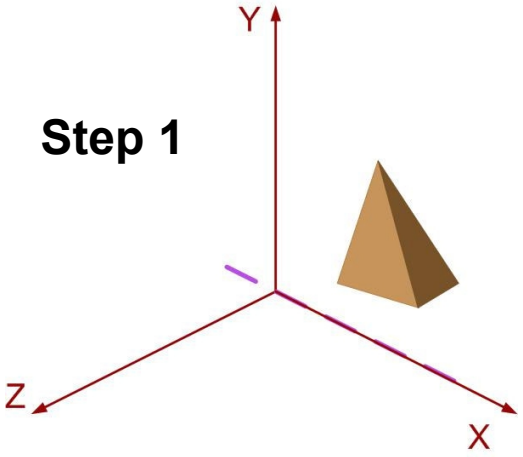
$$x \to y \to z \to x$$

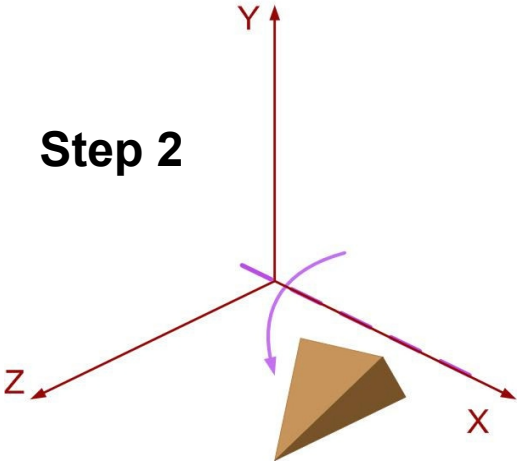


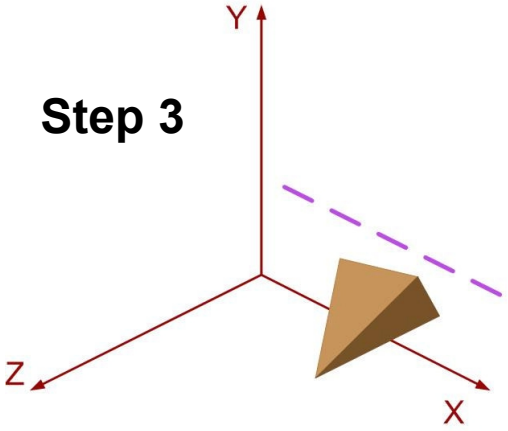
Rotation about axis parallel to co-ordinate axis

- E.g. x-axis









STEPS

1. Translate the object so as to coincide rotation axis to parallel co-ordinate axis

2. Perform the rotation about x-axis

3. Translate back the object so as to move rotation axis to original position

General 3D Rotations

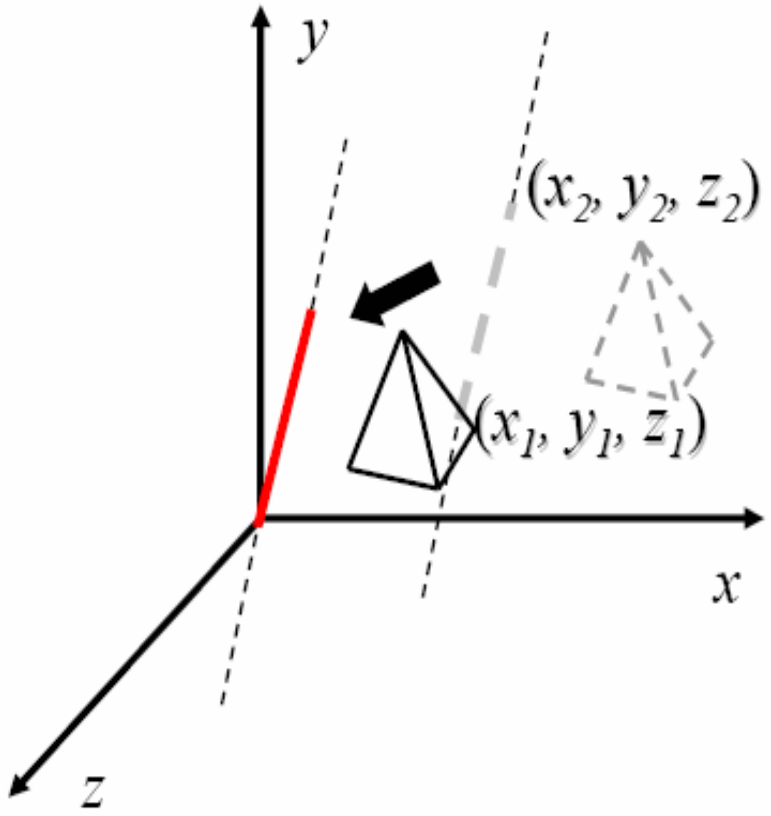
Rotation about an Arbitrary Axis

Basic Idea

1. **Translate the object so that the rotation axis passes through the coordinate origin**
2. **Rotate the object so that the axis of rotation coincides with one of the coordinate axes**
3. **Perform the specified rotation about the coordinate axis**
4. **Apply inverse rotations to bring the rotation axis back to its original orientation**
5. **Apply the inverse translation to bring the rotation axis back to its original position.**

Arbitrary Axis Rotation

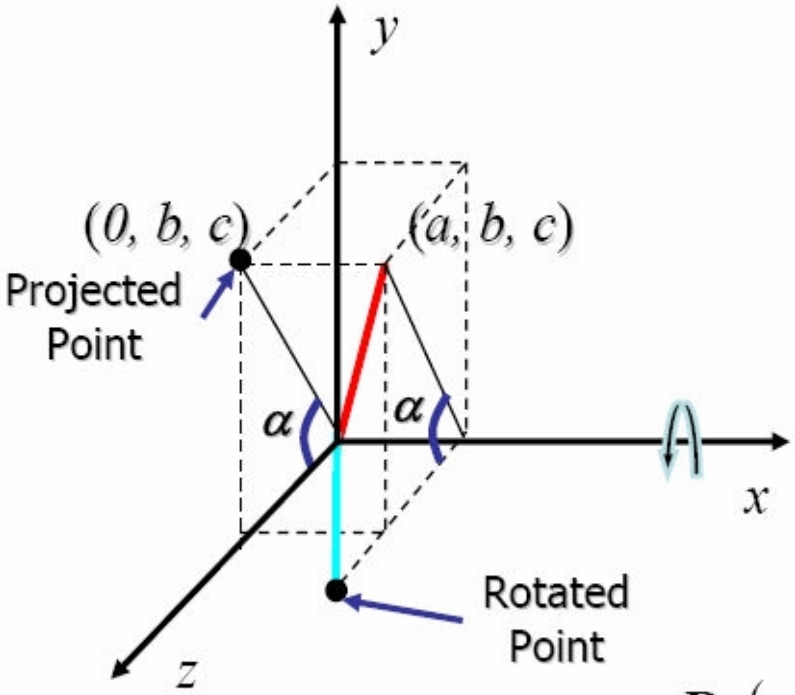
- Step 1. Translate



$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Arbitrary Axis Rotation

- Step 2. Rotate about x axis by α



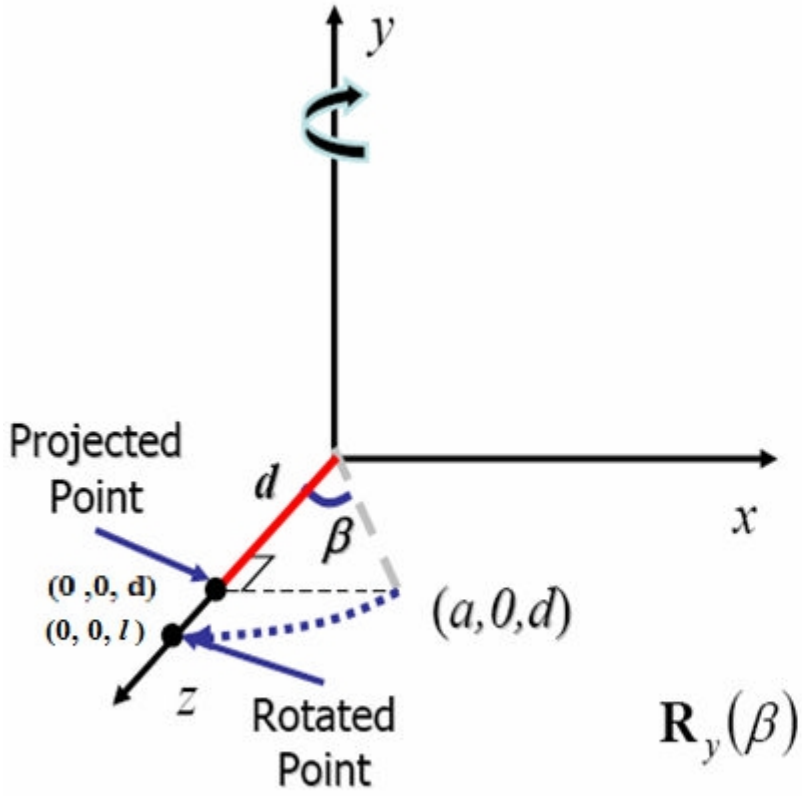
$$\sin\alpha = \frac{b}{\sqrt{b^2+c^2}} = \frac{b}{d}$$

$$\cos\alpha = \frac{c}{\sqrt{b^2+c^2}} = \frac{c}{d}$$

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Arbitrary Axis Rotation

- Step 3. Rotate about y axis by β (clockwise)



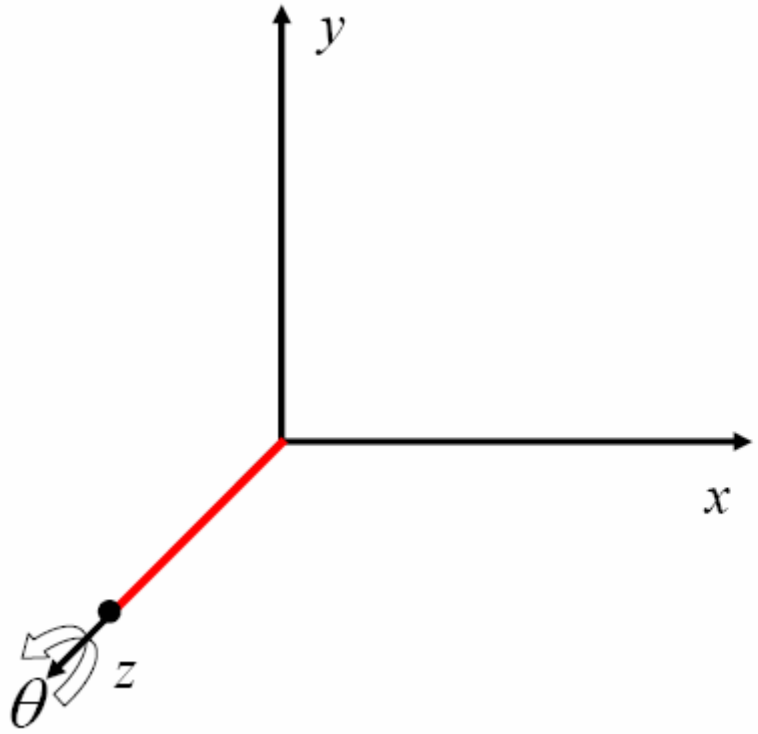
$$\sin\beta = \frac{a}{l}, \qquad \cos\beta = \frac{d}{l}$$

$$l^2 = a^2 + b^2 + c^2 = a^2 + d^2$$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} d/l & 0 & -a/l & 0 \\ 0 & 1 & 0 & 0 \\ a/l & 0 & d/l & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Arbitrary Axis Rotation

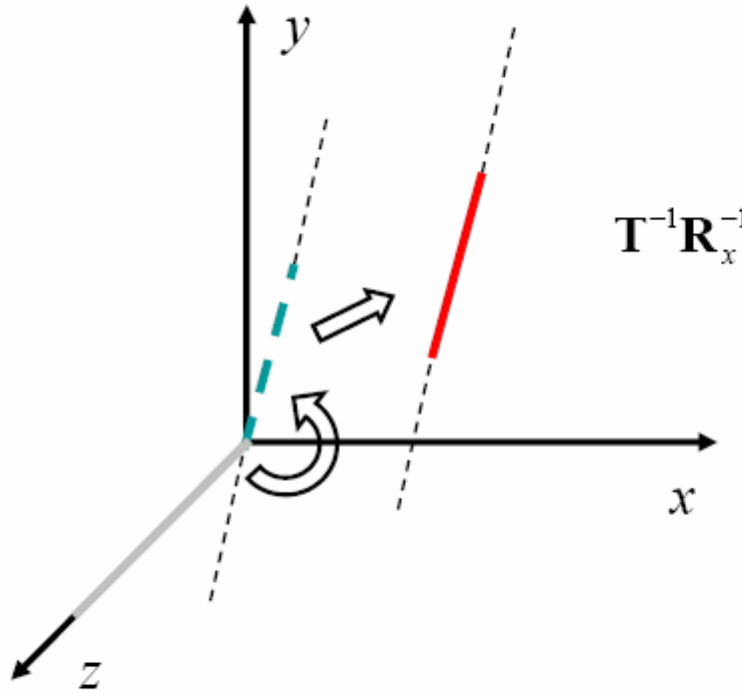
- Step 4. Rotate about z axis by the angle θ



$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Arbitrary Axis Rotation

- Step 5. Reverse transformation

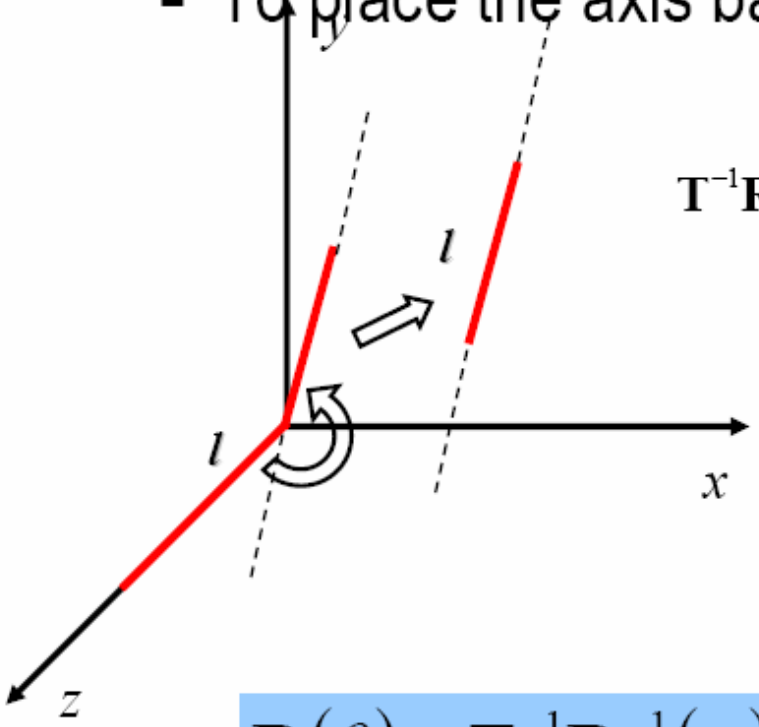


$$\mathbf{T}^{-1}\mathbf{R}_x^{-1}(\alpha)\mathbf{R}_y^{-1}(\beta)=\begin{bmatrix}1&0&0&x_1\\0&1&0&y_1\\0&0&1&z_1\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&\cos\alpha&\sin\alpha&0\\0&-\sin\alpha&\cos\alpha&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos\beta&0&\sin\beta&0\\0&1&0&0\\-\sin\beta&0&\cos\beta&0\\0&0&0&1\end{bmatrix}$$

$$\mathbf{R}(\theta)=\mathbf{T}^{-1}\mathbf{R}_x^{-1}(\alpha)\mathbf{R}_y^{-1}(\beta)\mathbf{R}_z(\theta)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha)\mathbf{T}$$

Arbitrary Axis Rotation

- Step 5. Reverse transformation
  - To place the axis back in its initial position

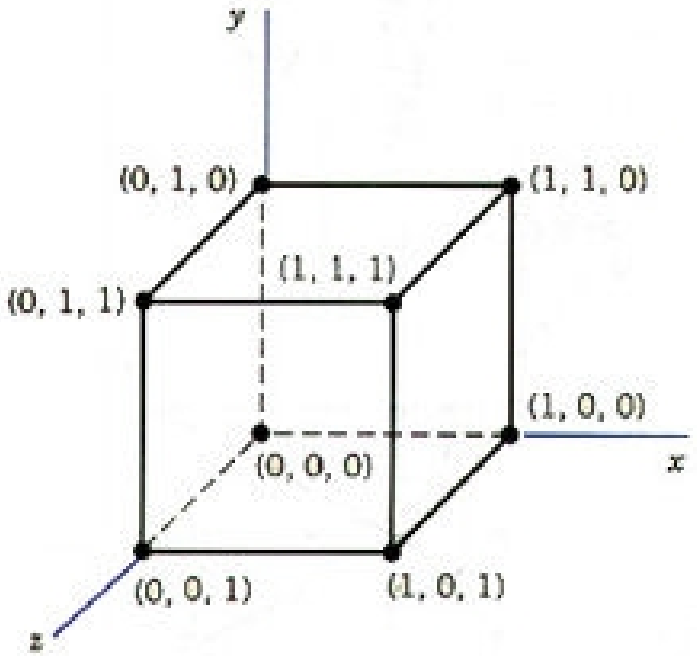


$$\mathbf{T}^{-1}\mathbf{R}_x^{-1}(\alpha)\mathbf{R}_y^{-1}(\beta)=\begin{bmatrix}1&0&0&x_1\\0&1&0&y_1\\0&0&1&z_1\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&\cos\alpha&\sin\alpha&0\\0&-\sin\alpha&\cos\alpha&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\cos\beta&0&\sin\beta&0\\0&1&0&0\\-\sin\beta&0&\cos\beta&0\\0&0&0&1\end{bmatrix}$$

$$\mathbf{R}(\theta)=\mathbf{T}^{-1}\mathbf{R}_x^{-1}(\alpha)\mathbf{R}_y^{-1}(\beta)\mathbf{R}_z(\theta)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha)\mathbf{T}$$

Example

Find the new coordinates of a unit cube 90°-rotated about an axis defined by its endpoints A(2,1,0) and B(3,3,1).

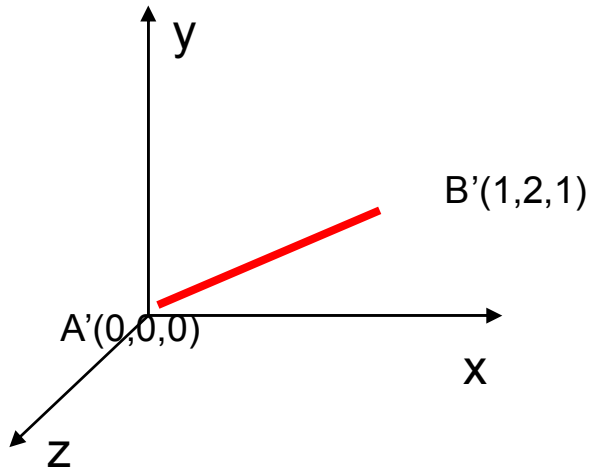


A Unit Cube

ASSIGNMENT

Example

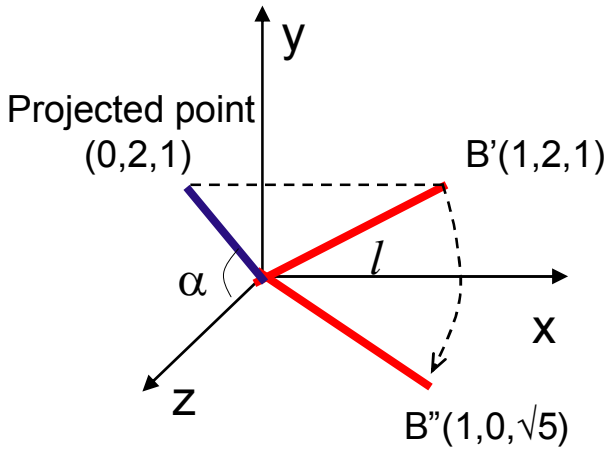
- Step1. Translate point A to the origin



$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example

- Step 2. Rotate axis *A'B'* about the x axis by and angle α, until it lies on the *xz* plane.



$$\sin\alpha = \frac{2}{\sqrt{2^2+1^2}} = \frac{2}{\sqrt{5}} = \frac{2\sqrt{5}}{5}$$

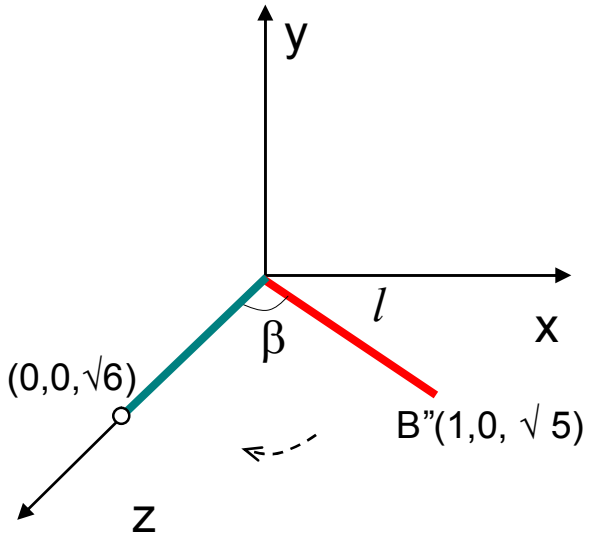
$$\cos\alpha = \frac{1}{\sqrt{5}} = \frac{\sqrt{5}}{5}$$

$$l = \sqrt{1^2+2^2+1^2} = \sqrt{6}$$

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} & 0 \\ 0 & \frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example

- Step 3. Rotate axis *A'B''* about the y axis by and angle β until it coincides with the z axis.



$$\sin\beta = \frac{1}{\sqrt{6}} = \frac{\sqrt{6}}{6}$$

$$\cos\beta = \frac{\sqrt{5}}{\sqrt{6}} = \frac{\sqrt{30}}{6}$$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \frac{\sqrt{30}}{6} & 0 & -\frac{\sqrt{6}}{6} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{6}}{6} & 0 & \frac{\sqrt{30}}{6} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example

- Step 4. Rotate the cube 90° about the z axis

$$\mathbf{R}_z(90^\circ) = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Finally, the concatenated rotation matrix about the arbitrary axis AB becomes,

$$\mathbf{R}(\theta) = \mathbf{T}^{-1}\mathbf{R}_x^{-1}(\alpha)\mathbf{R}_y^{-1}(\beta)\mathbf{R}_z(90^\circ)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha)\mathbf{T}$$

Example

$$\mathbf{R}(\theta)=\begin{bmatrix}1&0&0&2\\0&1&0&1\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&\frac{\sqrt{5}}{5}&\frac{2\sqrt{5}}{5}&0\\0&-\frac{2\sqrt{5}}{5}&\frac{\sqrt{5}}{5}&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}\frac{\sqrt{30}}{6}&0&\frac{\sqrt{6}}{6}&0\\0&1&0&0\\-\frac{\sqrt{6}}{6}&0&\frac{\sqrt{30}}{6}&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}0&-1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$\begin{bmatrix}\frac{\sqrt{30}}{6}&0&-\frac{\sqrt{6}}{6}&0\\0&1&0&0\\\frac{\sqrt{6}}{6}&0&\frac{\sqrt{30}}{6}&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&\frac{\sqrt{5}}{5}&-\frac{2\sqrt{5}}{5}&0\\0&\frac{2\sqrt{5}}{5}&\frac{\sqrt{5}}{5}&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&-2\\0&1&0&-1\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$=\begin{bmatrix}0.166&-0.075&0.983&1.742\\0.742&0.667&0.075&-1.151\\-0.650&0.741&0.167&0.560\\0&0&0&1\end{bmatrix}$$

Example

- Multiplying $\mathbf{R}(\theta)$ by the point matrix of the original cube

$$[\mathbf{P}'] = \mathbf{R}(\theta) \cdot [\mathbf{P}]$$

$$[\mathbf{P}'] = \begin{bmatrix} 0.166 & -0.075 & 0.983 & 1.742 \\ 0.742 & 0.667 & 0.075 & -1.151 \\ -0.650 & 0.741 & 0.167 & 0.560 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.650 & 1.667 & 1.834 & 2.816 & 2.725 & 1.742 & 1.909 & 2.891 \\ -0.558 & -0.484 & 0.258 & 0.184 & -1.225 & -1.151 & -0.409 & -0.483 \\ 1.467 & 1.301 & 0.650 & 0.817 & 0.726 & 0.560 & -0.091 & 0.076 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

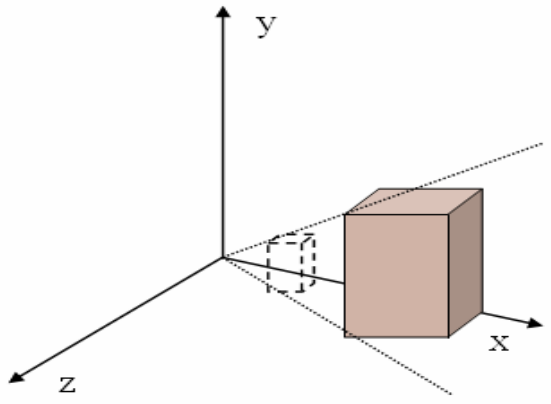
3-D SCALING

Scaling about origin

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fixed Point Scaling

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & (1-s_x)x_f \\ 0 & s_y & 0 & (1-s_y)y_f \\ 0 & 0 & s_z & (1-s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



$$S(x_f, y_f, z_f, s_x, s_y, s_z) = T(x_f, y_f, z_f).S(s_x, s_y, s_z).T(-x_f, -y_f, -z_f)$$

3-D REFLECTION

- **Performed relative to a reflection axis or reflection plane**
- **Axis reflection→ equivalent to 180° rotation about the axis in 3-D space**
- **Plane reflection → equivalent to 180° rotation in 4-D space**

- **Reflection in xy plane**

  x' = x

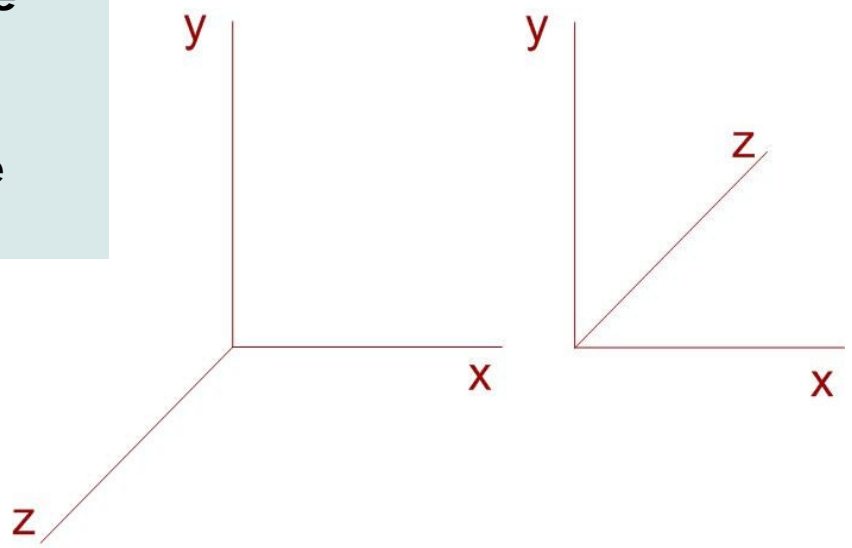
  y' = y

  z' = -z

Matrix is as

$$RF_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note :
yz plane
x→ -x
&
zx plane
y→ -y



3-D SHEAR

- Z-axis shear

  x' = x + a.z

  y' = y + b.z

  z' = z

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$SH_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

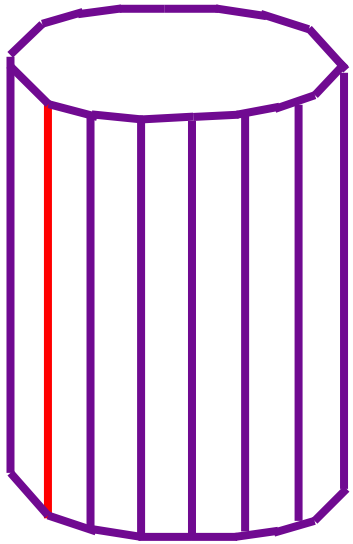
Parameter **a** and **b** can be assigned any values

→alters x and y coordinate values by an amount that is proportional to the z-value; while leaving the z-coordinate unchanged.

3-D OBJECT REPRESENTATION

3-D OBJECT REPRESENTATION

- Most commonly used representation for 3-d object
- Speeds up the surface rendering
- wired frame applications can be displayed quickly to give a general indication of surface structure.
- Greater the number of smaller surfaces, better the approximations



Polygon Surface Approximation of Cylinder

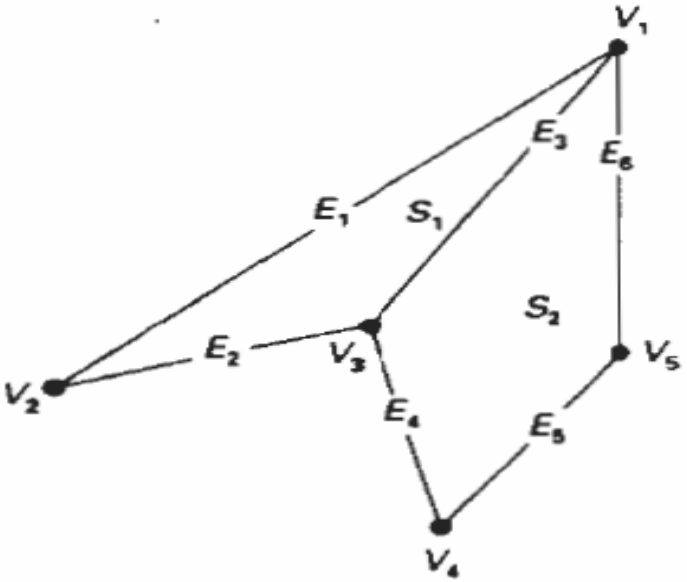
3-D OBJECT REPRESENTATION

Data Requirements for Polygon Surface Representation

- Polygon data tables– to store information
- Polygon Data Tables consists of
  - Geometric Tables- Consist of parameters that specify polygon vertices and spatial orientation of polygons
    - Vertex Table-stores the coordinates of vertices
    - Edge Table-contains the pointer back to the vertex table to identify the vertices for each polygon edge.
    - Polygon Table-contains pointer back into the edge table to identify the edges for each polygon
  - Attributes Tables- Consist of parameters that specify transparency, surface texture, color etc.

3-D Object Representation

- Convenient Organizations for storing geometric data is to create three lists
  - Vertex Table
  - Edge Table
  - Polygon Surface Table



VERTEX TABLE	
V_1:	x_1, y_1, z_1
V_2:	x_2, y_2, z_2
V_3:	x_3, y_3, z_3
V_4:	x_4, y_4, z_4
V_5:	x_5, y_5, z_5

EDGE TABLE	
E_1:	V_1, V_2
E_2:	V_2, V_3
E_3:	V_3, V_1
E_4:	V_3, V_4
E_5:	V_4, V_5
E_6:	V_5, V_1

POLYGON-SURFACE TABLE	
S_1:	E_1, E_2, E_3
S_2:	E_3, E_4, E_5, E_6

Guidelines to Generate Error Free Table

1. Every vertices listed as an end point for at least two edges
2. Every edge is part of at least one polygon
3. Every polygon is closed.
4. Every polygon has at least one shared edge
5. If edge table contains pointers to polygons, every edge referenced by a polygon pointer has a reciprocal pointer back to polygon

3-D Object Representation

- Spatial Orientation of Surface
  - Orientation of surface normal
  - How to find surface normal if surface vertices (x_1,y_1,z_1), (x_2,y_2,z_2) and (x_3,y_3,z_3) are given ?
    - Plane equation:

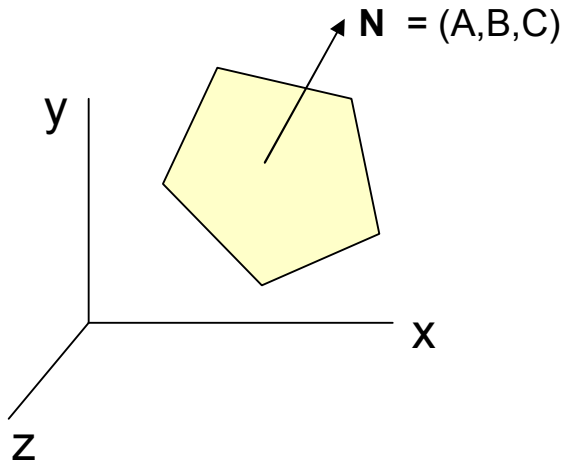
$$Ax + By + Cz + D = 0$$

    - Put the vertex coordinates (x_1,y_1,z_1), (x_2,y_2,z_2) and (x_3,y_3,z_3) in plane equation to get

Solution obtained from cramer' s rule

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \qquad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \qquad D = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$



    - Surface Normal **N** to the plane Ax+By+Cz+D=0 has the Cartesian components (A,B,C)

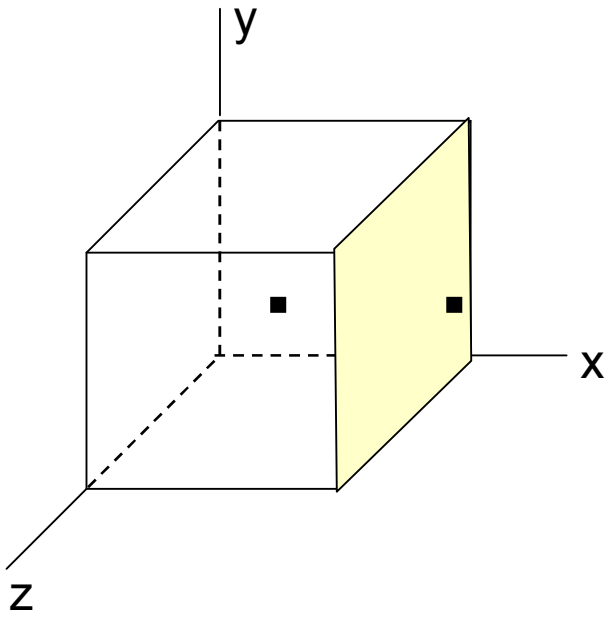
3-D Object Representation

Two sides of a surface (Inside or Outside the Plane Surface)

- For right handed cartesian system (i.e. if vertices are taken in Counter-Clockwise order to evaluate A,B,C and D)

if $Ax + By + Cz + D < 0$, the point (x, y, z) is inside the surface

if $Ax + By + Cz + D > 0$, the point (x, y, z) is outside the surface



Projections

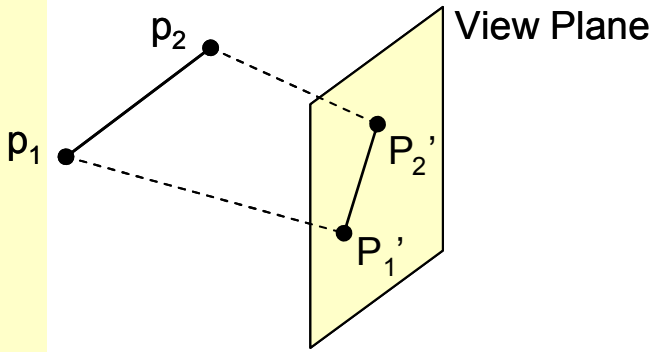
- Transformation that changes a point in n-dimensional coordinate system into a point in a coordinate system that has dimension less than n.
- Converts 3-D viewing co-ordinates to 2-D projection co-ordinates
- **TYPES OF PROJECTION**
  - **Parallel Projection**
  - **Perspective Projection**

PARALLEL PROJECTION

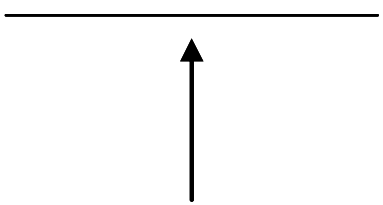
- **Orthographic**
- **Oblique**

PARALLEL PROJECTION

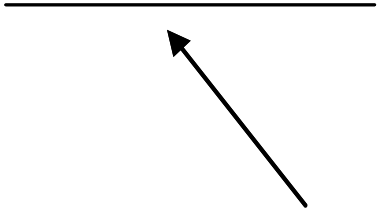
1. Parallel Projection
   - Coordinate positions are transformed to view plane along parallel lines (projection lines)
   - Preserves relative proportions of objects
   - Accurate views of various sides of an object are obtained.
   - Doesn't give realistic representation of the appearance of the 3-D object
   - **Types**
   - **Orthographic**- when the projection is perpendicular to the view plane. Used to produce Front, Side and Top view of an object
   - **Oblique** – when the projection is not perpendicular to the view plane



Parallel Projection



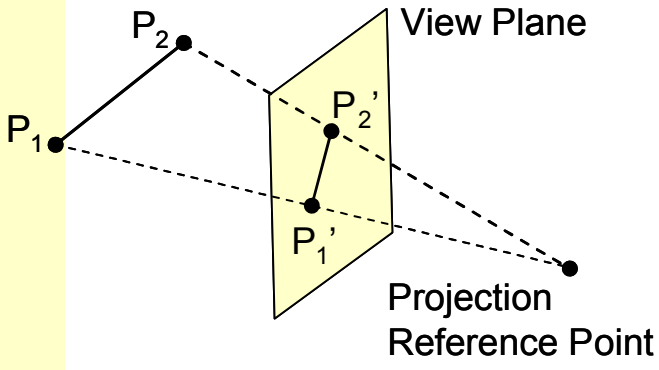
Orthographic Projection



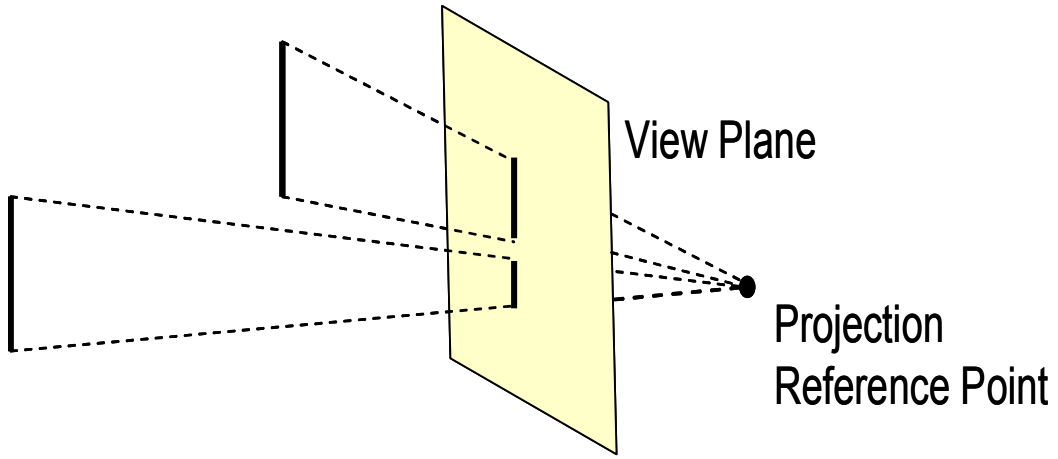
Oblique Projection

PERSPECTIVE PROJECTION

- Coordinate positions are transformed to view plane along lines (projection lines) that converges to a point called **projection reference point** (center of projection)
- Produce realistic view
- Does not preserve relative proportions
- Equal sized object appears in different size according as distance from view plane



Perspective Projection



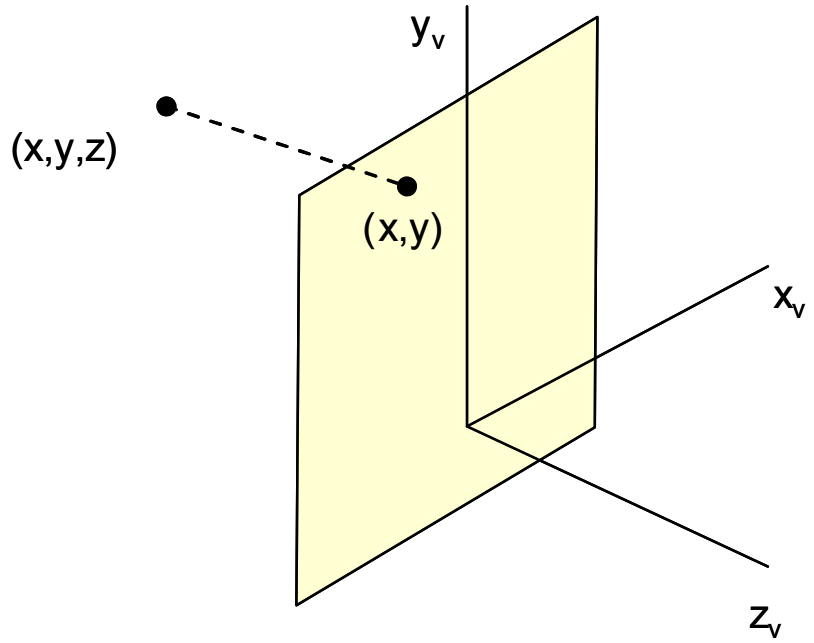
Perspective Projection of equal sized objects at different distances from the view plane

ORTHOGRAPHIC PROJECTION OF A POINT ON TO A VIEW PLANE

$x_p = x$

$y_p = y$

Note: **Z value is preserved for the depth information needed in depth culling and visible surface determination procedure**



Orthographic projection of a point onto a viewing plane

Oblique Projection

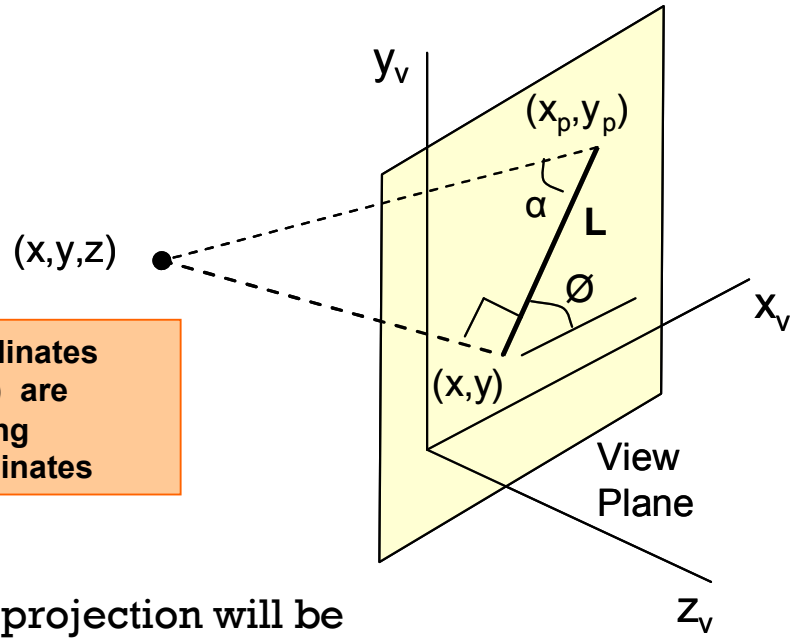
- Oblique projection vectors are specified by two angles α and Ø
- (x,y) are also the orthographic co-ordinates on view plane

Projection Matrix Calculation

$$x_p = x + L.cos\emptyset$$
$$y_p = y + L.sin\emptyset$$

$$\tan\alpha = \frac{Z}{L} \qquad , L = \frac{Z}{\tan\alpha}, L = ZL_1$$

$$L_1 = \frac{1}{\tan\alpha}$$



Coordinates (x,y,z) are viewing coordinates

$$x_p = x + z(L_1 cos\emptyset)$$
$$y_p = y + z(L_1 sin\emptyset)$$

- Final projection matrix for producing parallel projection will be

$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1\cos\phi & 0 \\ 0 & 1 & L_1\sin\phi & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Orthographic projection: When $L_1 = 0$
$x_p = x$
$y_p = y$

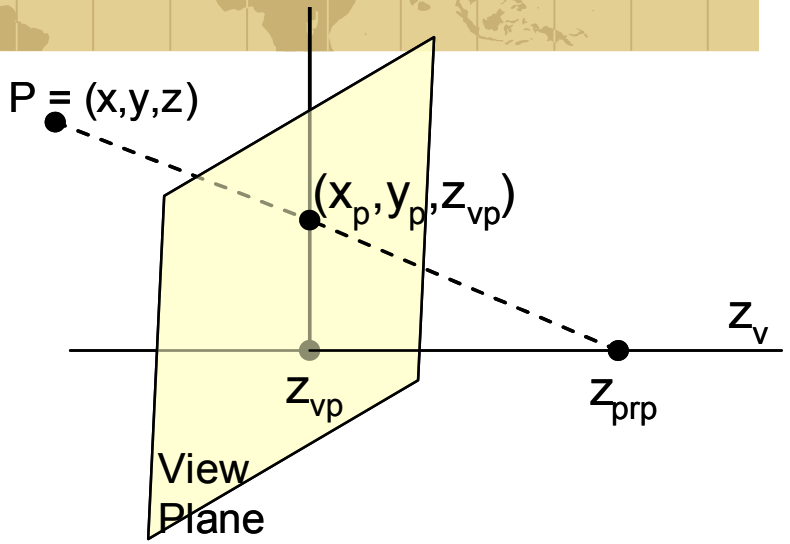
Perspective Projection

- Projection vectors meet at projection reference point z_{prp} along the z_v axis
  - $x' = x - xu$
  - $y' = y - yu$
  - $z' = z - (z - z_{prp})u$
  - u takes value between 0 and 1
  - P'(x',y',z') represents any point along the projection line
- On view plane
  - $z' = z_{vp}$; therefore
  - $u = (z_{vp} - z)/(z_{prp} - z)$
- thus projection transformation equations are

$$x_p = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) = x\left(\frac{d_p}{z_{prp} - z}\right)$$

$$y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) = y\left(\frac{d_p}{z_{prp} - z}\right)$$

Where, $h = (z_{prp} - z)/d_p$
and $x_p = x_h/h$, $y_p = y_h/h$



Perspective projection of a point P with coordinates (x,y,z) to position (xp,yp,zvp) on the view plane

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{vp}/d_p & z_{vp}(z_{prp}/d_p) \\ 0 & 0 & -1/d_p & z_{prp}/d_p \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Perspective projection

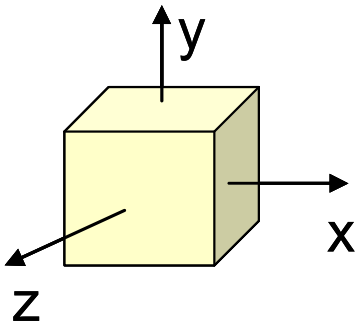
Vanishing Point: a set of parallel lines that are not parallel to view plane are projected as converging lines that appear to converge at a point called vanishing point

•a set of parallel lines that are parallel to view plane are projected as parallel lines

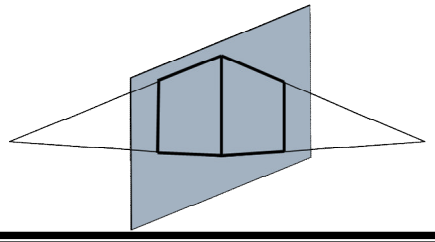
•More than one set of parallel lines form more than one vanishing points in the scene

Principal Vanishing point: Vanishing point for a set of parallel lines parallel to one of the principal axis of object

•We can control the number of principal vanishing point to one, two or three with the orientation of projection plane and classify as **one, two or three point perspective projection**



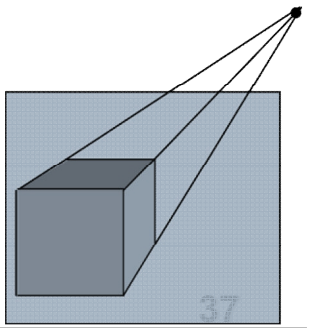
Coordinate Description



X-axis vanishing Point

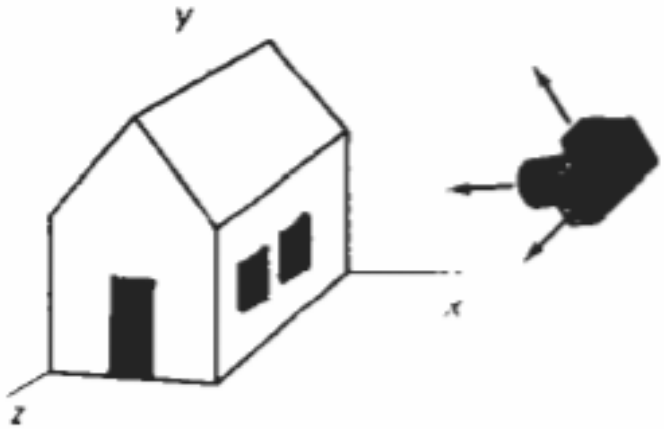
z-axis vanishing Point

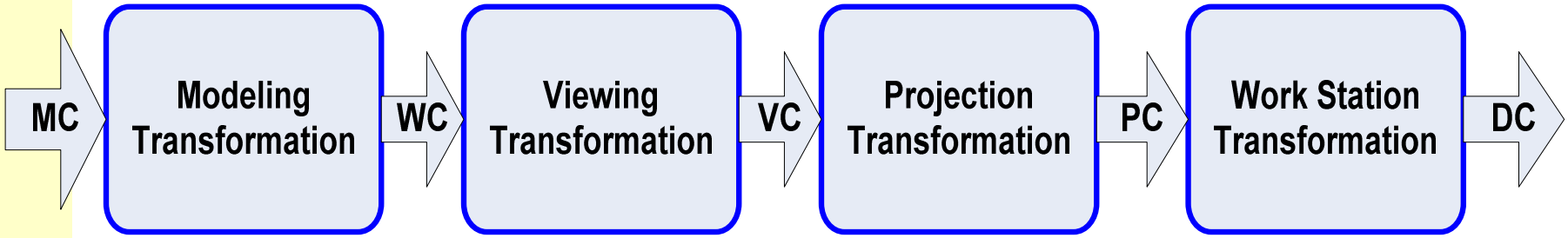
Vanishing Point



3-D Viewing Pipeline

- **3-D scene generation process analogous to taking photographs with camera**



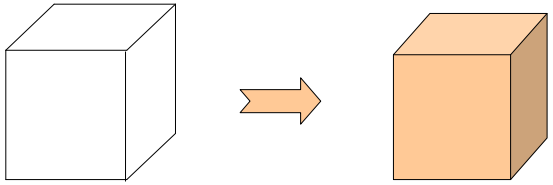


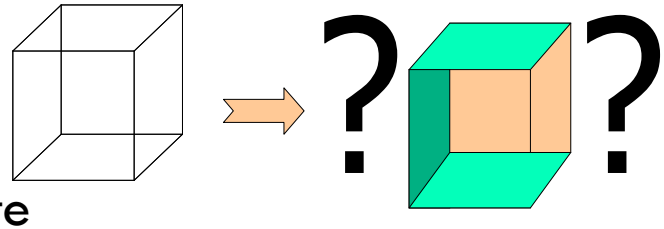
MC-Modeling Coordinates WC-World Coordinates VC-Viewing Coordinates
PC- Projection Coordinates DC-Device Coordinates

Fig: 3D Transformation Pipeline Transformation

VISIBLE SURFACE DETECTION

Visible Surface Detection Method





- Also called **hidden surface elimination method**
- Considerations
  - Memory
  - Time
  - Object Type

- **Two approaches**

Depends upon object definition or projected image

- Object-space
  - Deals with object definition directly
  - Compares objects and parts of them to each other to determine the visible object or its parts.
  - E.g. Line Display algorithms in wire-frame display

- Image Space Method
  - deals with projected image of an object
  - Visibility is decided point by point at each pixel position on projected plane
  - Can be adapted to determine visible-line detection.

Back Face Detection Method (Plane Equation Method)

- Fast and simple object-space method
- Based on "inside" or "outside" test
- Test whether view point is inside or outside the polygon surface
- Point (x',y',z') is "inside" a polygon surface with plane parameters A,B,C,D if Ax+By+Cz+D<0 i.e. the polygon surface must be a back face.

Alternatively

1. Make the center of object at origin
2. Calculate the value of D of the plane by taking three points in anticlockwise direction.
3. If D<=0 the surface is visible
   D>0 the surface in invisible.

Consider Normal Vector **N** = (A,B,C) to polygon surface and Vector **V** in viewing direction

The polygon is back face if: **V.N**>0

If **V.N<0**→ Visible Surface

If object description is converted to viewing co-ordinates then, **V** is along zv axis i.e **V** = (0,0,Vz), then:

V.N = VzC

i.e V.N > 0 → VzC

So we need to consider only sign of C

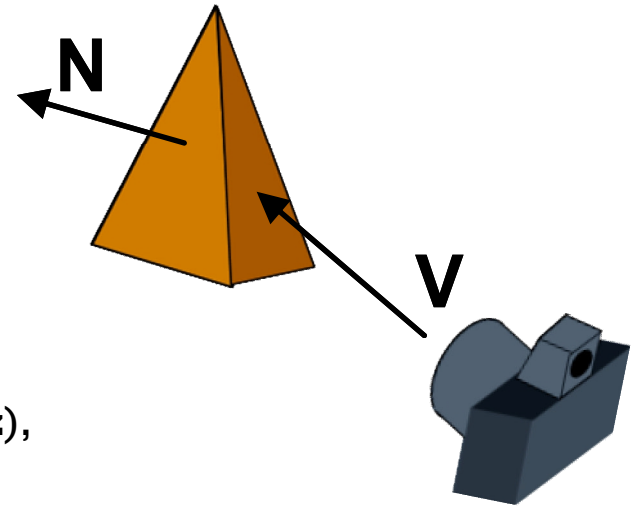


Fig: Vector V in the Viewing Direction and a back face normal Vector N of a polyhedron

Back Face Detection Method (Plane Equation Method)

- For right handed viewing system with viewing direction along negative z_v axis, if c<0 → back face
  - What if C = 0 ??
  - viewing direction is perpendicular to surface normal→ also back face

  So, **back face** if **C≤0**

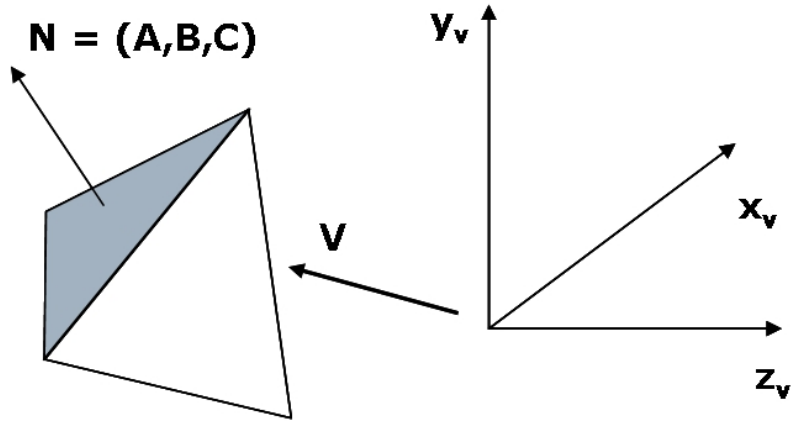


Fig: A polygon surface with C < 0 in a right-handed viewing coordinate system is identified as a back face when the viewing direction is along the negative Zv axis

- Doesn't works well for
  1. Overlapping front faces due to
     - Multiple objects
     - Concave objects
  2. Non Closed Objects
  3. Non-polygonal models

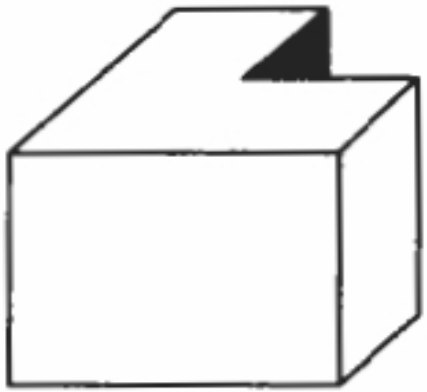


Fig: View of a concave polyhedron with one face partially hidden by other faces

Depth Buffer (Z-Buffer) method

- Commonly used Image-space method to detect visible surface
- Surface depth is compared at each pixel position on the projection plane
- Object depth is usually measured from the view plane along the z-axis of the viewing system

- For each pixel position (x,y) on projection plane, object depths can be compared by comparing z-values as each (x,y,z) position on a polygon surface corresponds to the orthographic projection point (x,y) on projection plane

- Usually applied to scene containing only polygon surfaces; but possible to apply for non-planar surfaces

- Two buffers required
  - **Depth buffer:** to store depth value for each position as surfaces are processed
    - Initially all positions are set to 0 (minimum depth)
  - **Refresh buffer**: to store intensity values of each position
    - Initially refresh buffer is initialized to background intensity

- **DRAWBACK**: Deals only with opaque surface but not with transparent surface
  - i.e. it can only find one visible surface at each pixel position

Depth Buffer Method- Algorithm

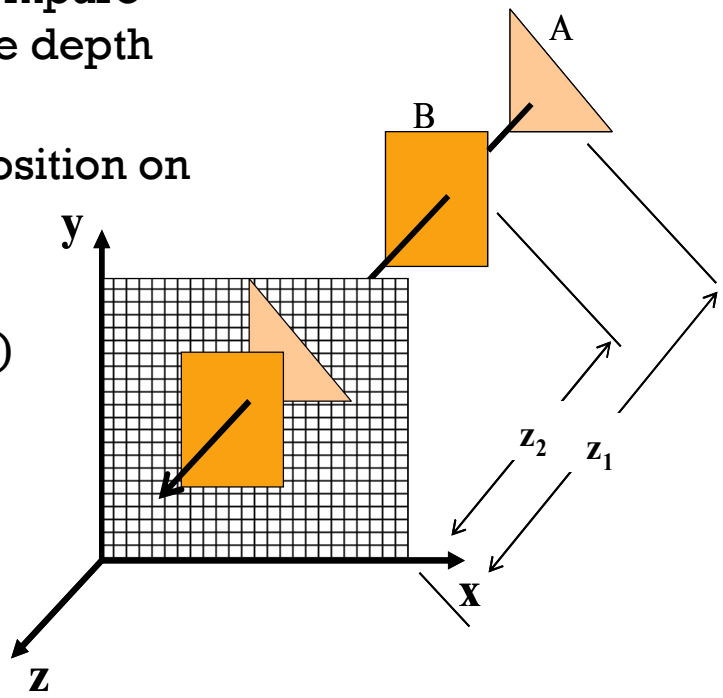
1. Initialize the depth buffer and refresh buffer so that for all buffer positions (x,y),

   depth(x,y) = 0, refresh(x,y) = $I_{backgrnd}$

2. For each position on each polygon surface, compare depth values to previously stored values in the depth buffer to determine visibility

   - Calculate the depth z for each (x,y) position on the polygon
   - If z>depth(x,y), then set

     depth(x,y)=z, refresh(x,y)=I_{surf}(x,y)



$I_{backgrnd}$ = background intensity
I_{surf}(x,y) = projected intensity value for the surface at pixel position (x,y)

3. After all pixels and surfaces are compared, draw object using x,y,z from depth and intensity refresh buffer.

Depth Buffer (Depth Computation)

- From plane equation, depth is

$$z = \frac{-Ax - By - D}{C}$$

- For the next adjacent pixel (x+1,y) in a scan line, depth is

$$z' = \frac{-A(x+1) - By - D}{C} \quad \text{or} \quad z' = z - \frac{A}{C}$$

- **Start from top scan line to bottom scan line**
- Starting from top vertex, calculate x position down the left edge of the polygon recursively as

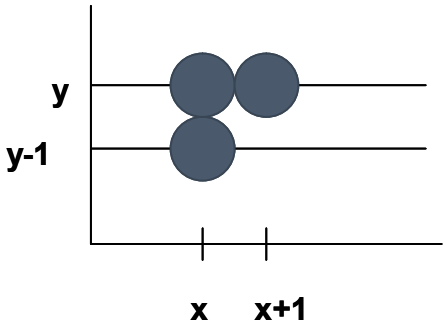
  x' = x – 1/m
- Thus depth value for the next pixel in left edge is obtained as

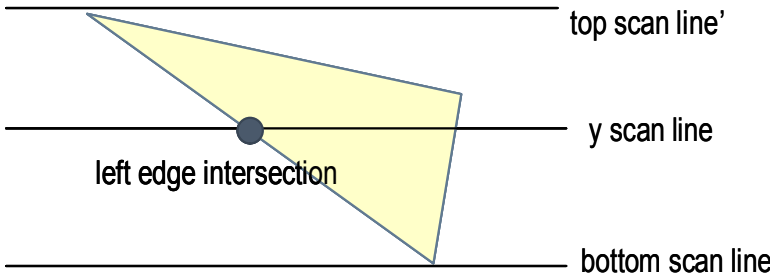
$$z' = z + \frac{\frac{A}{m} + B}{C}$$

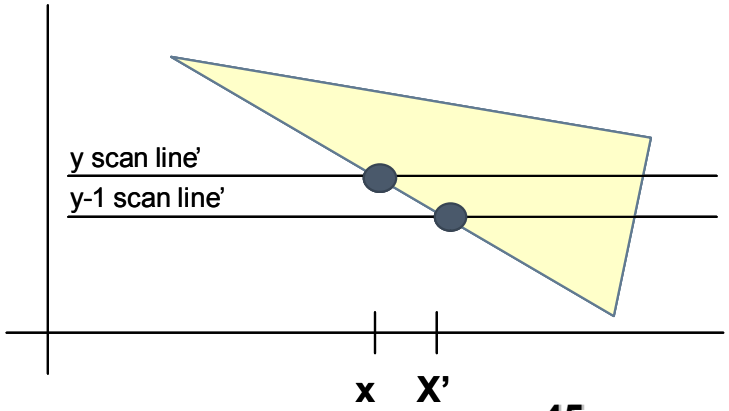
put x' = x-1/m and y' = y-1

- For vertical edge m = infinity, so:

$$z' = z + \frac{B}{C}$$

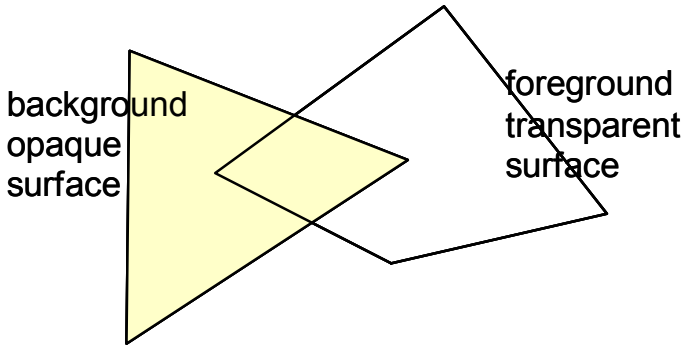




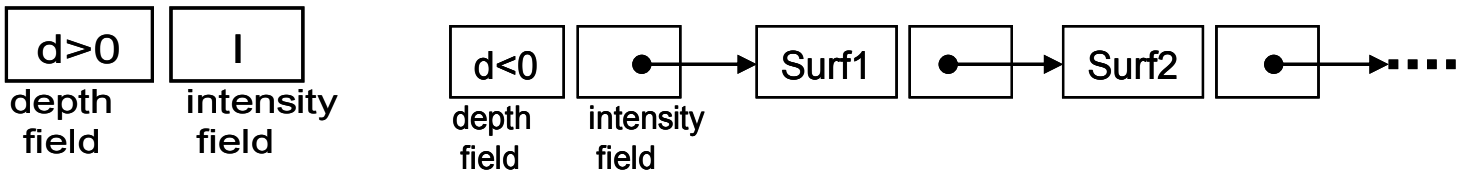


A-Buffer Method

- Extension of depth buffer method
- represents Antialiased, area-average, accumulation-buffer
- The depth buffer is expanded so that each position in the buffer can reference a linked list of surfaces thus enabling more than one surface intensity consideration



- Each pixel position in the A-Buffer has two fields
  - **Depth Field** → stores a positive or negative real number
    - Positive→ single surface contributes to pixel intensity
    - Negative → multiple surfaces contribute to pixel intensity
  - **Intensity Field** → stores surface-intensity information or a pointer value
    - Surface intensity if single surface → stores the RGB components of the surface color at that point and percent of pixel coverage
    - Pointer value if multiple surfaces



A- Buffer Method

Data for each surface in the linked list includes

- RGB intensity components
- Opacity parameter (percent of transparency)
- Depth
- Percent of area coverage
- Surface identifier
- Other surface-rendering parameters
- Pointer to next surface

- Scan lines are processed to determine surface overlaps of pixels across the individual scan lines.
- Surfaces are subdivided into a polygon mesh and clipped against the pixel boundaries
- The opacity factors and percent of surface overlaps are used to determine the pixel intensity as an average of the contribution from the overlapping surfaces

Scan-Line Method

- Extension of scan line algorithm for filling polygon interiors
- Instead of filling just one surface, we deal with multiple surfaces
- As each scan line is processed, all polygon surfaces intersecting that line are examined to determine which are visible.
  - Across each scan line , depth calculations are made for each overlapping surface to determine, which is nearest to view plane.
- When the visible surface has been determined , the intensity value for that position is entered into the refresh buffer.

DATA STRUCTURE

- **A. Edge table** containing
  - Coordinate endpoints for each line in a scene
  - Inverse slope of each line
  - Pointers into polygon table to identify the surfaces bounded by each line
- **B. Surface table** containing
  - Coefficients of the plane equation for each surface
  - Intensity information for each surface
  - Pointers to edge table
- **C. Active Edge List**
  - To keep a trace of which edges are intersected by the given scan line

Note :

- The edges are sorted in order of increasing x
- Define flags for each surface to indicate whether a position is inside or outside the surface

SCAN LINE ALGORITHM

I. Initialize the necessary data structure

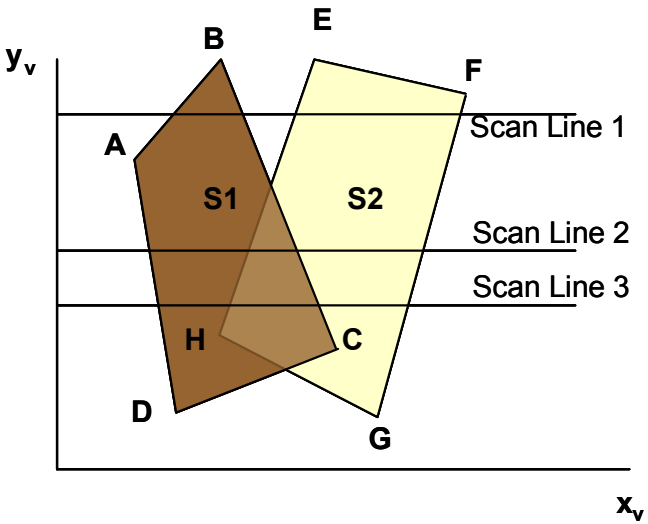
1. Edge table containing end point coordinates, inverse slope and polygon pointer.
2. Surface table containing plane coefficients and surface intensity
3. Active Edge List
4. Flag for each surface

II. For each scan line repeat

1. update active edge list
2. determine point of intersection and set surface on or off.
3. If flag is on, store its value in the refresh buffer
4. If more than one surface is on, do depth sorting and store the intensity of surface nearest to view plane in the refresh buffer

Scan-Line Method (contd…)

- For scan line 1
  - The active edge list contains edges AB,BC,EH, FG
  - Between edges AB and BC, only *flags for s1 == on* and between edges EH and FG, only *flags for s2==on*
    - no depth calculation needed and corresponding surface intensities are entered in refresh buffer
- For scan line 2
  - The active edge list contains edges AD,EH,BC and FG
  - Between edges AD and EH, only the *flag for surface s1 == on*
  - Between edges EH and BC *flags for both surfaces == on*
    - Depth calculation (using plane coefficients) is needed.
    - In this example ,say s2 is nearer to the view plane than s1, so intensities for surface s2 are loaded into the refresh buffer until boundary BC is encountered
  - Between edges BC and FG flag for s1==off and *flag for s2 == on*
    - Intensities for s2 are loaded on refresh buffer
- For scan line 3
  - Same **coherent** property as scan line 2 as noticed from active list, so no depth calculation needed between edges BC and EH

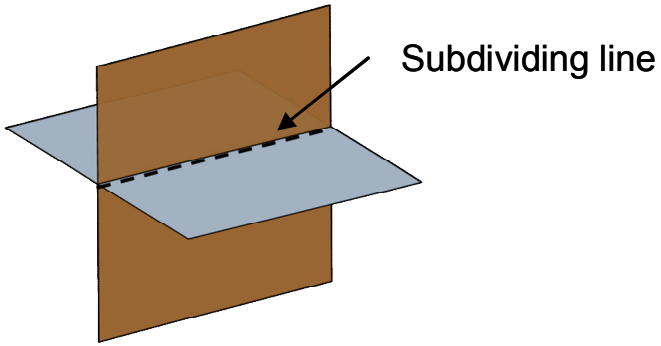


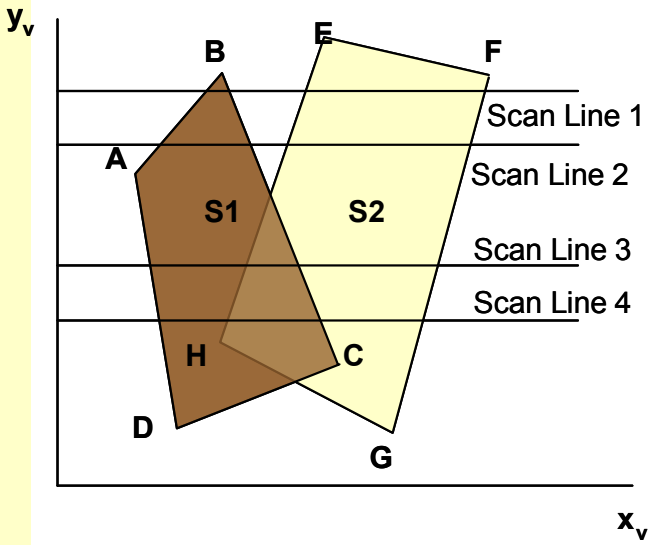
Scan-Line Method (contd…)

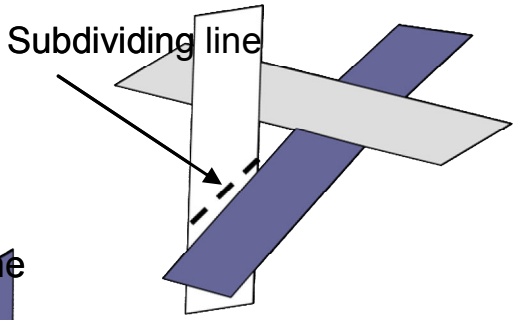
Problem:

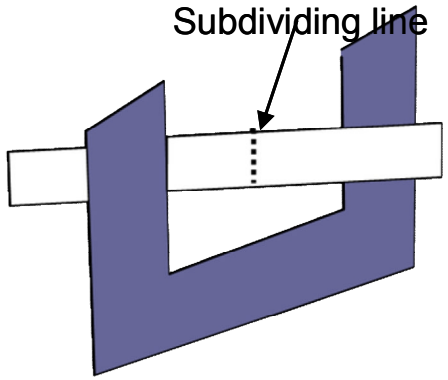
Dealing with **cut through** surfaces and **cyclic overlap** is problematic when used coherent properties

- Solution: Divide the surface to eliminate the overlap or cut through









Illumination Models & Surface Rendering Methods

Illumination and Rendering

- An ***illumination model*** in computer graphics
  - also called a ***lighting model*** or a ***shading model***
  - used to calculate the color of an illuminated position on the surface of an object
  - Approximations of the physical laws

- A ***surface-rendering method*** determine the pixel colors for all projected positions in a scene

Photorealism in Computer Graphics

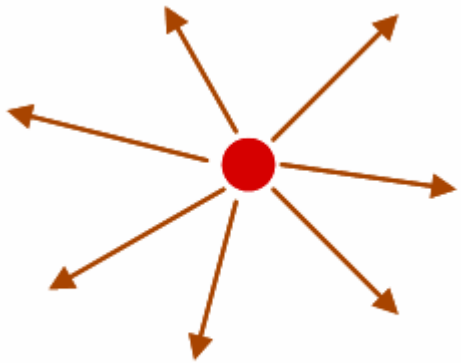
- Photorealism in computer graphics involves
  - Accurate representations of surface properties, and
  - Good physical descriptions of the lighting effects

- Modeling the lighting effects that we see on an object is a complex process, involving principles of both physics and psychology

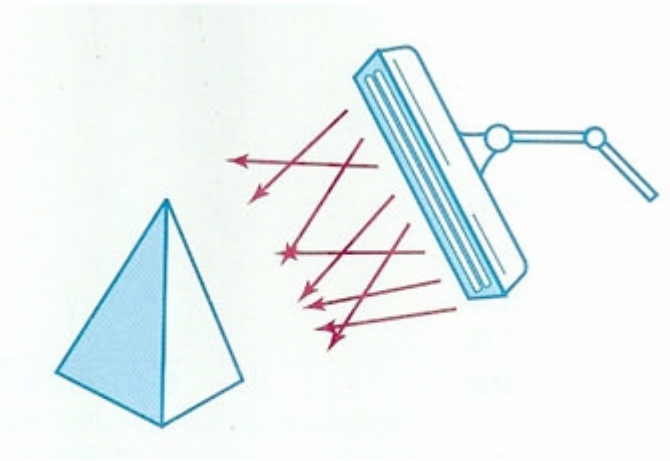
- Physical illumination models involve
  - Material properties, object position relative to light sources and other objects, the features of the light sources, and so on

Light Sources

- Total Reflected Light = Contribution from light sources
  + contribution from reflecting surfaces
- Point Source: when light source model is a reasonable approximation for sources whose dimensions are small compared to the size of objects in a scene. e.g. Sun
- Distributed Light Source: area of source is not small compared to the surfaces in the scene
  - e.g. Fluorescent lamp



Point Light Source



Distributed Light Source

Ambient Light

- Also called background light
- Multiple reflection of nearby (light-reflecting) objects yields a uniform illumination
- A form of diffuse reflection independent of the viewing direction and the spatial orientation of a surface
- Ambient illumination is constant for an object

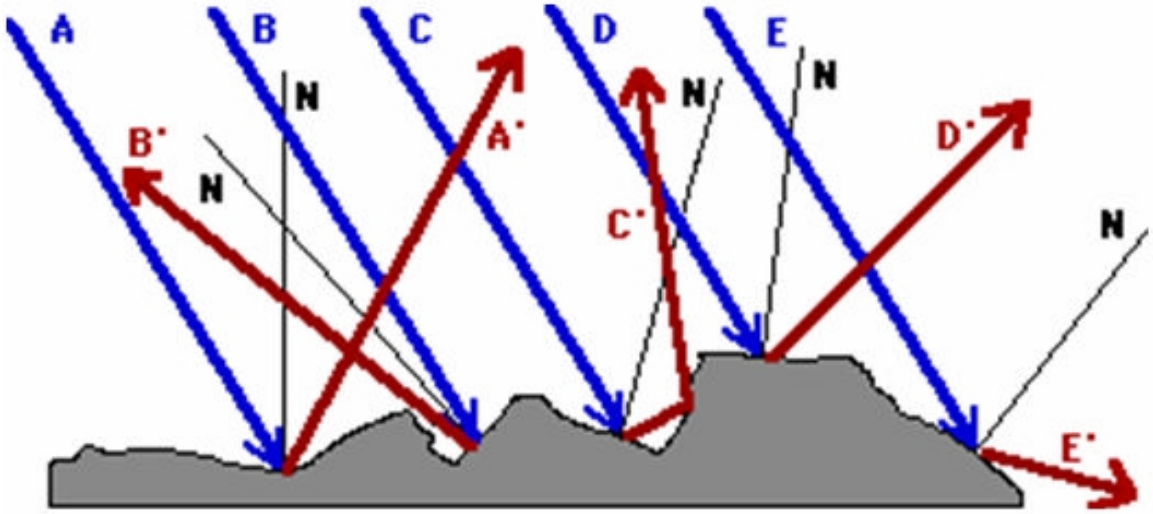
$$I_{amb} = k_a I_a$$

I_a : the incident ambient intensity

k_a : ambient reflection coefficient, the proportion reflected away from the surface

Diffuse Reflection

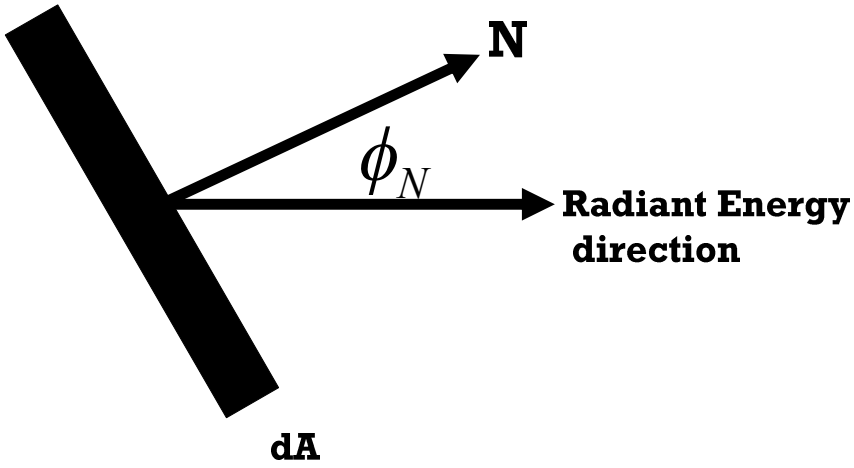
- Surfaces are rough
- Incident light is scattered with equal intensity in all directions
- Surfaces appear equally bright from all direction.
- Such surfaces are called ***ideal diffuse reflectors*** (also referred to as ***Lambertian reflectors***)



Diffuse Reflection From a Surface

Lambert's Cosine Law

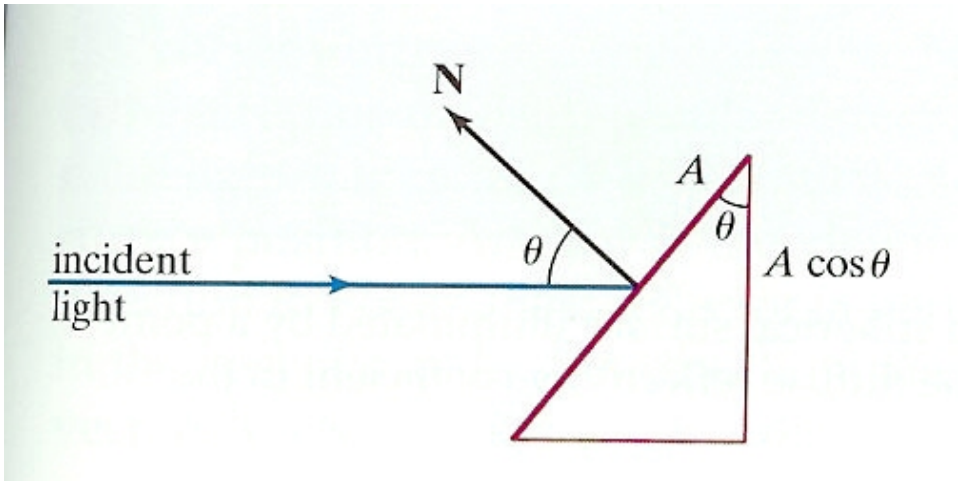
- " The radiant energy from any small surface area dA in any direction ϕ_N relative to the surface normal is proportional to $cos\phi_N$

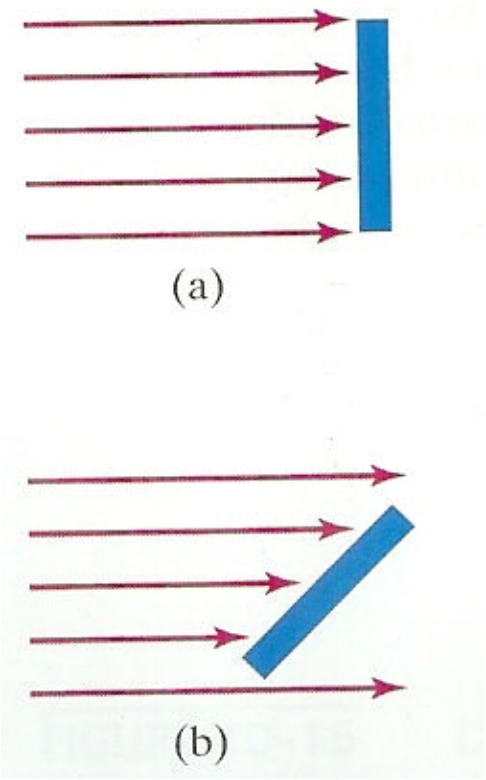


Diffuse Reflection

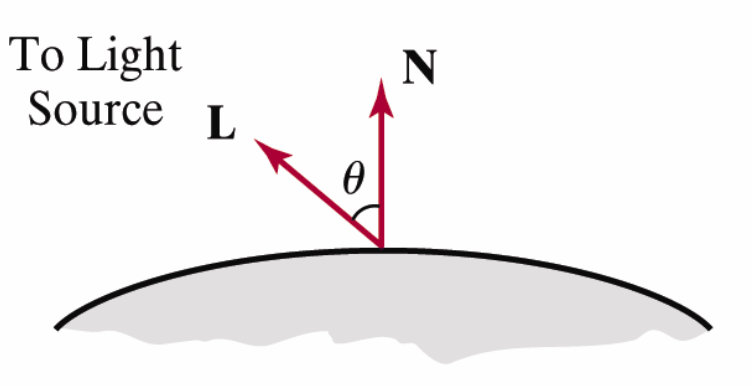
- Light intensity is independent of angle of reflection
- Light intensity depends on angle of incidence

When cosΘ is
+ve surface is illuminated
-ve light source is behind the surface





Diffuse Reflection



$$I_{diff} = k_d I_l \cos\theta = k_d I_l (N \cdot L)$$

I_l : the intensity of the light source

k_d : diffuse reflection coefficient,

N : the surface normal (unit vector)

L : the direction of light source, (unit vector)

When cosΘ is
+ve surface is illuminated
-ve light source is behind the surface

Ambient + Diffuse

$$I = \begin{cases} k_a I_a + k_d I_l (N \cdot L) & \text{if } N \cdot L > 0 \\ k_a I_a & \text{if } N \cdot L \le 0 \end{cases}$$

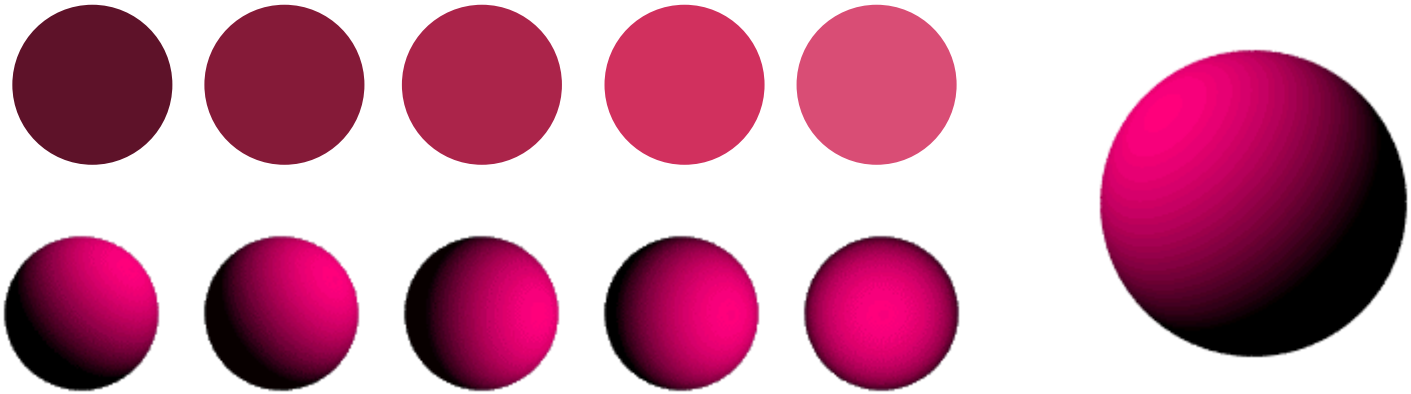
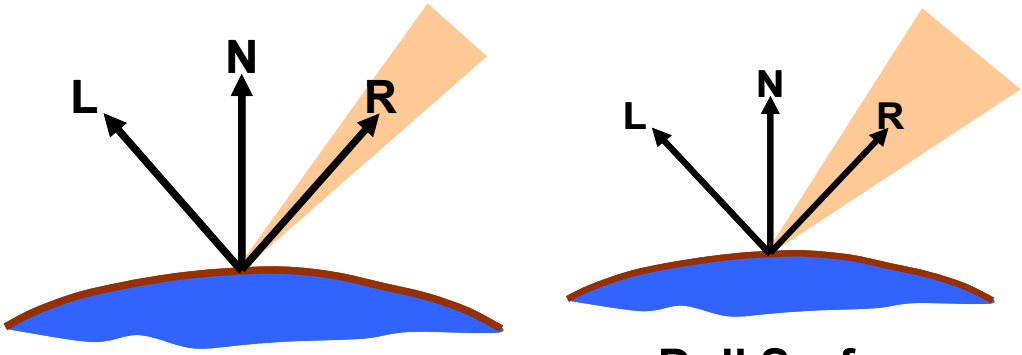


Fig: sphere illuminated with different intensity ambient light
Illuminated with varying direction light source

Specular Reflection

- Perfect reflector (mirror) reflects all lights to the direction where angle of reflection is identical to the angle of incidence
- It accounts for the ***highlight***
- Near total reflector reflects most of light over a range of positions close to the direction
- E.g. illumination on shiny surface such as polished metal, an apple or a person's forehead
- Shiny surface have a narrow specular-reflection range, and dull surfaces have a wider reflection range.

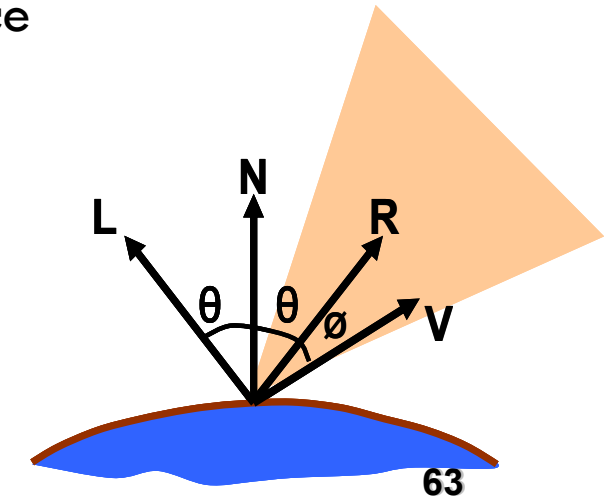


Phong Specular Reflection Model

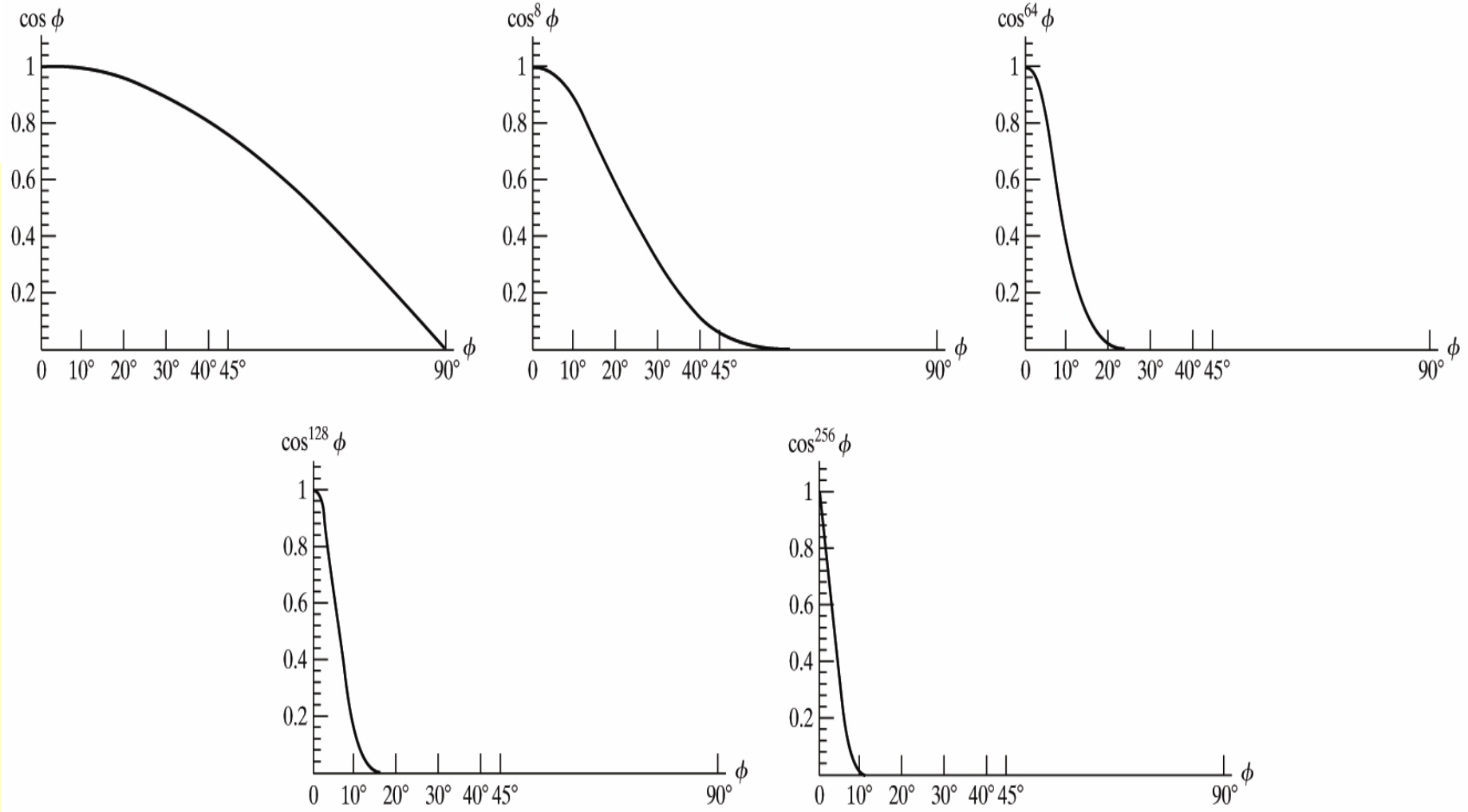
- Sets Intensity of specular reflection proportional to $\cos^{n_s}\phi$
  - n_s → specular reflection parameter (depends on surface)
  - Ø ranges from 0 to 90^0 (*cos Ø* varies from 0 to 1)
- Intensity of specular reflection depends on:
  - Material properties of surface
  - Angle of incidence θ
  - Other factors such as polarization and color of the incident light
- Monochromatic specular intensity variations can be approximated using **specular-reflection coefficient**, $W(\theta)$ for each surface

$$I_{spec} = w(\theta) I_l \cos^{n_s}\phi$$

- At $\theta = 90^0$, $w(\theta) = 1$ → all incident light is reflected



Specular Reflection



Plot For $\cos^{n_s}\phi$ and Specular Reflection Coefficient For Various Materials

Specular Reflection

$$I_{spec} = k_s I_l \cos^{n_s} \phi = k_s I_l (R \cdot V)^{n_s}$$

N = unit normal Vector

L = unit vector towards light source

R = unit vector to specular reflection direction

V = unit vector towards viewer

ϕ = angle between R and V

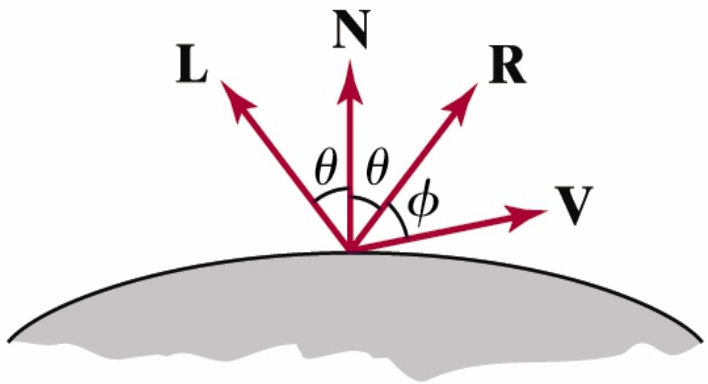


Figure 10-16

Specular reflection angle equals angle of incidence θ.

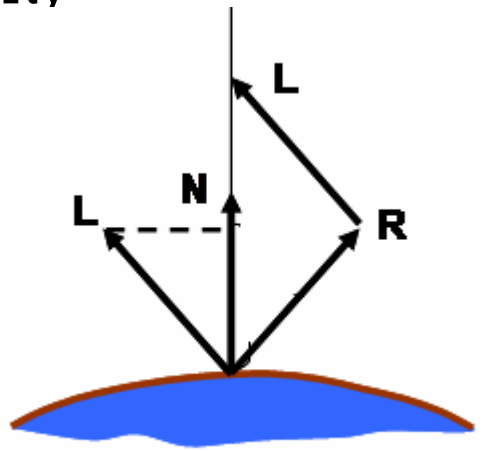
I_l : intensity of the incident light

k_s : color-independent specular coefficient

n_s : specular reflection parameter (depends upon surface)

Specular Reflection

- Specular-reflection coefficient k_s is a material property
  - For some material, k_s varies depending on θ
  - $k_s = 1$ if $\theta = 90°$

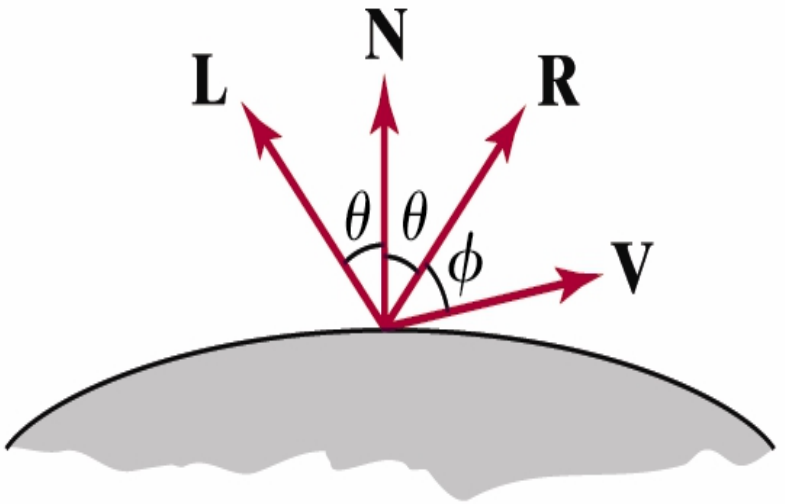


Calculation of vector R

- Calculating the reflection vector R

$$\mathbf{R} + \mathbf{L} = (2\mathbf{N} \cdot \mathbf{L})\mathbf{N}$$

$$\mathbf{R} = (2\mathbf{N}.\mathbf{L})\mathbf{N} - \mathbf{L}$$



Specular Reflection

- **Simplified Phong model using halfway vector**
  - Replacing **V**.**R** with **N**.**H** where **H** is halfway vector between **L** and **V** (i.e. **H** is unit bisector vector of angle between **L** and **V**)
  - H is constant if both viewer and the light source are sufficiently far from the surface

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|}$$

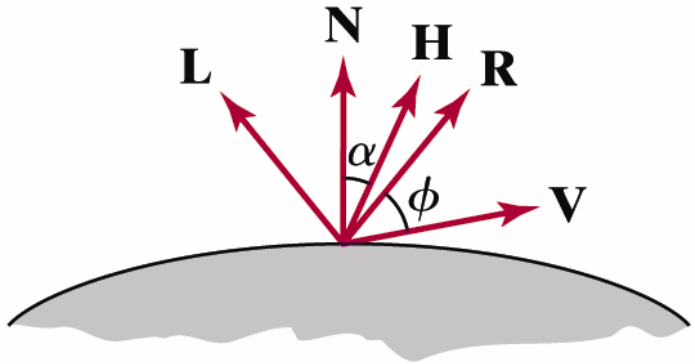


Figure 10-22

Halfway vector **H** along the bisector of the angle between **L** and **V**.

Combined Diffuse and Specular Reflection

- Point Light Source

$$I = I_{diff} + I_{spec}$$
$$= k_a I_a + k_d I_l (N.L) + k_s I_l (N.H)^{n_s}$$

- Multiple light sources (n light sources)

$$I = k_a I_a + \sum_{i=1}^{n} I_{li} \left[k_d (N.L_i) + k_s (N.H_i)^{n_s} \right]$$

Intensity Attenuation

- Intensity attenuation must be considered for producing realistic lighting effects.
- Intensity of radiant energy at a point *d* distance far from source is attenuated by $1/d^2$
- **attenuation factor** $1/d^2$ *produces* too much intensity variation
- Using inverse linear quadratic function of d for intensity attenuation as:

$$f(d) = \frac{1}{a_0 + a_1 d + a_2 d^2}$$

  - a_0 can be adjusted to prevent f(d) from becoming too large when d is very small
  - Magnitude of attenuation function is limited to 1 as

$$f(d) = \min\left(1, \frac{1}{a_0 + a_1 d + a_2 d^2}\right)$$

The Phong illumination model considering attenuation is:

$$I = k_a I_a + \sum_{i=1}^{n} f(d_i) I_{li} \left[k_d (N.L_i) + k_s (N.H_i)^{n_s} \right]$$

Color Consideration in Phong Illumination model

- For RGB description, each color in a scene is expressed in terms of R,G and B components
- Various methods:
  - Described by considering the RGB components for e.g. (k_{dR}, k_{dG}, k_{dB}) of diffuse reflection coefficient vector
    - e.g. For blue light ($k_{dR}=k_{dG}=0$)

$$I_B = k_{aB} I_{aB} + \sum_{i=1}^{n} f(d_i) I_{lBi} \left[k_{dB} (N.L_i) + k_{sB} (N.H_i)^{n_s} \right]$$

  - Described by specifying components of diffuse and specular color vectors for each surface and retaining the reflectivity (k) as a single valued constants

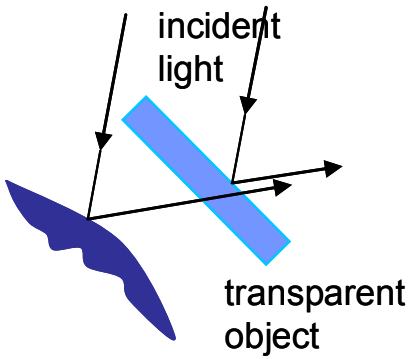
$$I_B = k_a S_{dB} I_{aB} + \sum_{i=1}^{n} f(d_i) I_{lBi} \left[k_d S_{dB} (N.L_i) + k_s S_{sB} (N.H_i)^{n_s} \right]$$

  - Described by specifying wavelength for a color specification. This specification is useful to specify color as more than three components

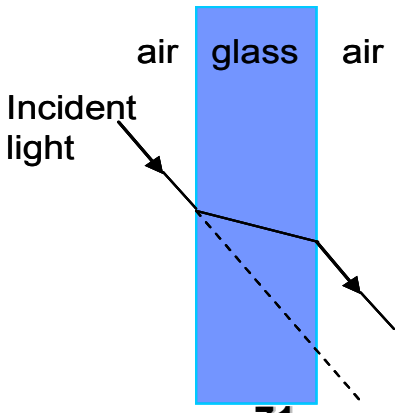
$$I_B = k_a S_{d\lambda} I_{a\lambda} + \sum_{i=1}^{n} f(d_i) I_{l\lambda i} \left[k_d S_{d\lambda} (N.L_i) + k_s S_{s\lambda} (N.H_i)^{n_s} \right]$$

Transparency

- Transparent surface produces both reflected and transmitted light
- Light intensity depends on relative transparency and position of light source or illuminated object behind or in front of the transparent surface



- To model transparent surface, intensity contribution of light from various sources (illuminated objects) that are transmitted from the surface must be considered in the intensity equation
- Both diffuse and specular reflection take place on transparent surface
- Diffuse effects are important for partially transparent surfaces such as frosted glass
- The Snell's law is used to calculate the refracted ray direction



$$\sin\theta_r = \frac{\eta_i}{\eta_r}\sin\theta_i$$

Transparency (contd…)

- Transmitted intensity I_{trans} through a transparent surface from a background object and Reflected intensity I_{refl} from the transparent surface with ***transparency coefficient*** k_t is given by

$$I = (1-k_t)I_{refl} + k_tI_{trans}$$

where $(1-k_t)$ is ***opacity factor***

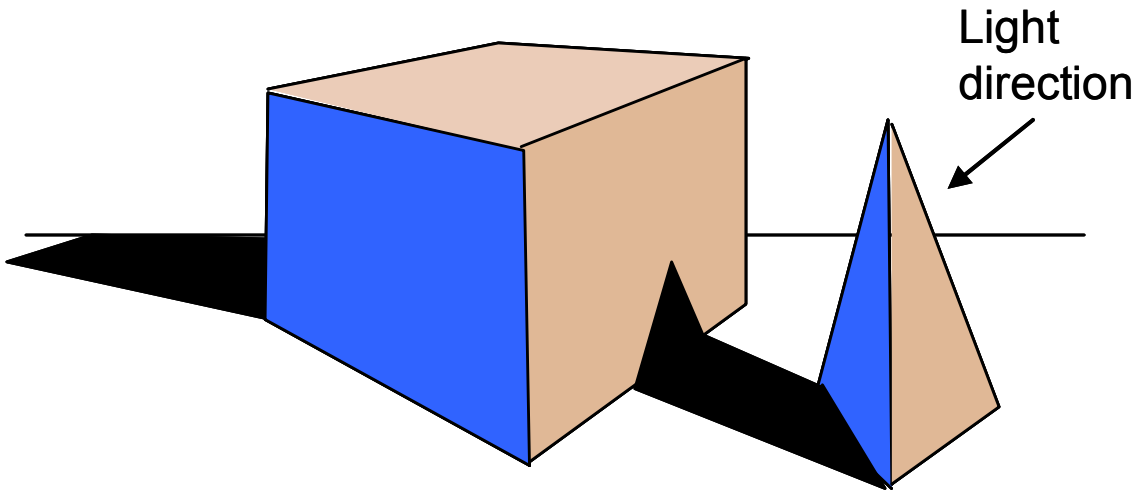
Transparency effects Implementation:

1. *Process opaque objects first to determine depths for visible opaque surfaces.*
2. *Depth positions of the transparent objects are compared to the values previously stored in the depth buffer.*
3. *If any transparent surface is visible, its reflected intensity is calculated and combined with the opaque-surface intensity previously stored in the frame buffer.*
4. *Visible transparent surfaces are then rendered by combining their surface intensities with those of the visible and opaque surfaces behind them.*

Shadow

- Hidden surface method with light source at the view position can be used

- The shadow area for all light sources are determined and these shadows could be treated as a surface pattern arrays



Assigning Intensity Level

- The intensity calculated by illumination model must be converted to the allowable intensity of a particular graphics system

- The difference between intensities 0.20 and 0.22 should be perceived same as that of 0.80 and 0.88

- The intensity level in a monitor should be spaced so that the ratio of successive intensities is constant

  for n+1 successive intensity levels

  $$I_1/I_0 = I_2/I_1 = \ldots\ldots\ldots I_n/I_{n-1} = r$$

  $$I_k = r^k I_0$$

  also $I_n = 1 \rightarrow r = (1/I_0)^{1/n}$

  so, $I_k = I_0^{(n-k)/n}$

- Lowest intensity value I_0 depends on the characteristics of the monitor (I_0 ranges from 0.005 to around 0.025)
  - the black level displayed on monitor will have some intensity

- The highest intensity value is 1
- For color system (blue color for example)

  $$I_{Bk} = r_B^k I_{B0}$$

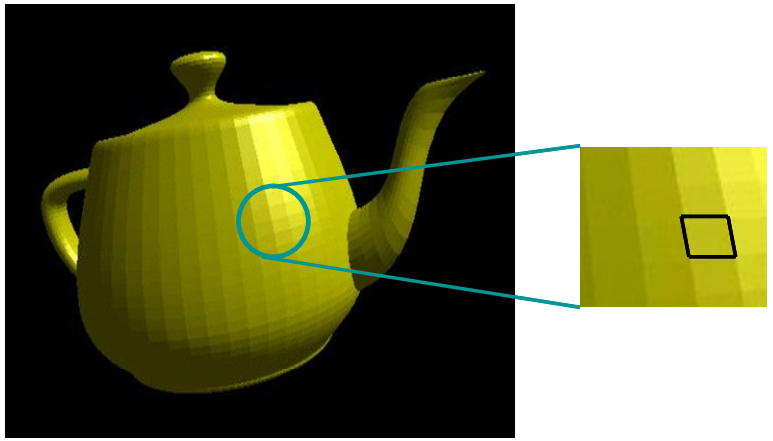
Polygon Rendering Methods

- Objects usually polygon-mesh approximation
- Illumination model is applied to fill the interior of polygons
- Curved surfaces are approximated with polygon meshes
  - But polyhedra that are not curved surfaces are also modeled with polygon meshes
- **Two ways** of polygon surface rendering:
  1. Single intensity for all points in a polygon
  2. Interpolation of intensities for each point in a polygon
- Methods:
  1. Constant Intensity Shading
  2. Gouraud Shading
  3. Phong Shading

Constant Intensity Shading or Flat Shading

-Fast and simple method for rendering an object with polygon surface

-Each polygon shaded with single intensity calculated for the polygon



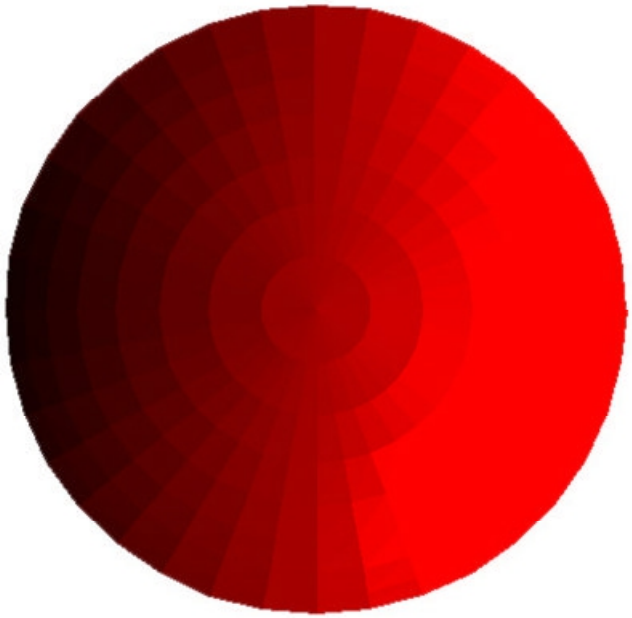
PROCEDURE

1. Take a point on the object surface and calculate the intensity
2. Render the surface with same intensity throughout the surface
3. Repeat above procedure for each polygon surface

ASSUMPTIONS

1. Object is a polyhedron
2. light sources should be sufficiently (i.e. **N.L** and attenuation function are constant)
3. Viewing position is sufficiently far (i.e. **V.R** is constant over the surface)

DRAWBACK: intensity discontinuity at the edges of polygons



Credit goes to the students

Gouraud Shading

- Intensity interpolation method
- Renders a polygon surface by linearly interpolating intensity values across the surface.
- Intensity discontinuity at the edges of polygons is eliminated by matching intensity values of each polygon with adjacent polygons

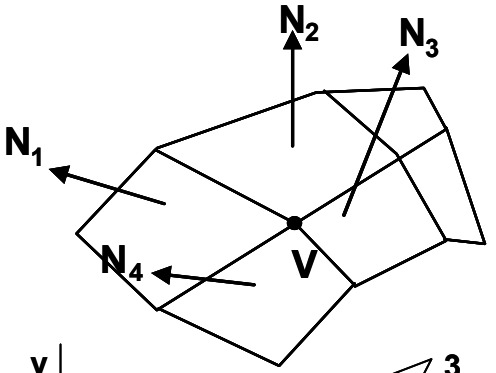
PROCEDURE

1. Determine the average unit normal vector at each polygon vertex
2. Calculate each of the vertex intensities by applying an illumination model
3. Linearly interpolate the vertex intensities over the polygon surface

Gouraud Shading (contd…)

- Average Unit Normal: Obtained by averaging the surface normals of all polygons sharing the vertex

$$N_v = \frac{\sum_{k=1}^{n} N_k}{\left| \sum_{k=1}^{n} N_k \right|}$$

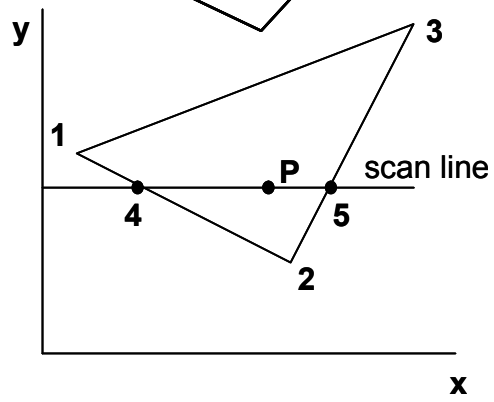


- Intensity interpolation:
  - Along the polygon edges are obtained by interpolating intensities at the edge ends

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

Recursive calculation along the edge

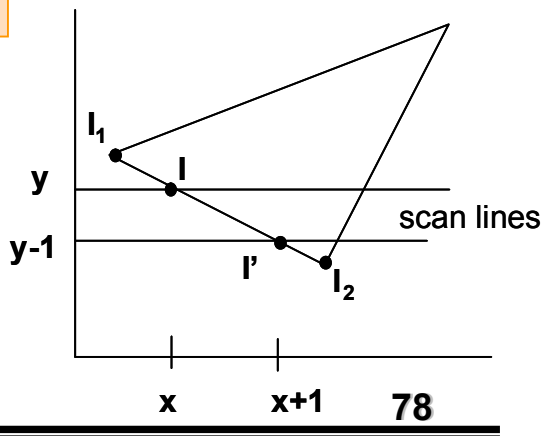
$$I' = I + \frac{I_2 - I_1}{y_1 - y_2}$$



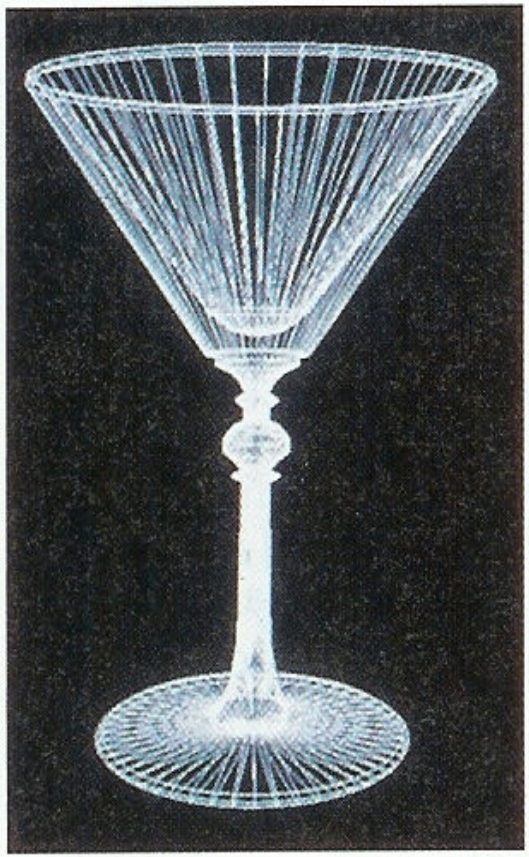
  - Along the scan line between the polygon edges are obtained by interpolating intensities at the intersection of scan line and polygon edges

$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Recursive Calculation along the scan line ??



Surface Rendering



Polygon Approximation

Flat Shading

Gouraud Shading

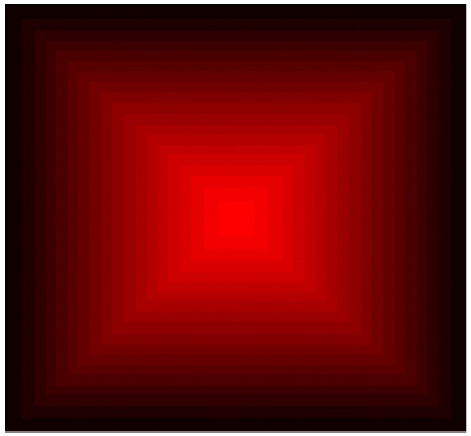
Gouraud Shading Problems

- Lighting in the polygon interior is inaccurate
- Mach band effect
  - Optical illusion
  - "Mach Bands" are not physically there. Instead, they are illusions due to excitation and inhibition in our neural processing
  - bright and dark intensity streaks caused by linear interpolation of intensities

Solution: Could be reduced by dividing the surface into large number of polygons or by using other methods, such as *Phong shading*

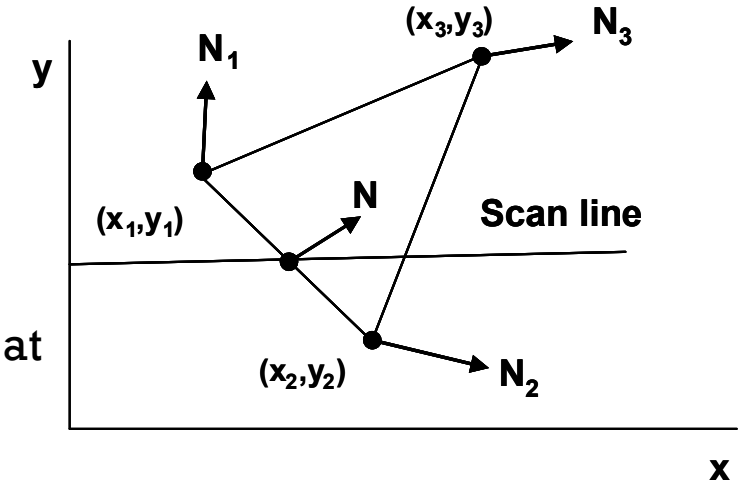
The bright bands at 45 degrees (and 135 degrees) are illusory.

The intensity of each square is the same.



Phong Shading

- More accurate method for rendering
- Interpolate normal vectors and apply illumination model to each surface point
- **PROCEDURE**



1. Determine average unit normal vectors at each polygon vertex
2. Linearly interpolate the vertex normals over the surface of the polygon
3. Apply an illumination model along each scan line to calculate projected pixel intensities for the surface points

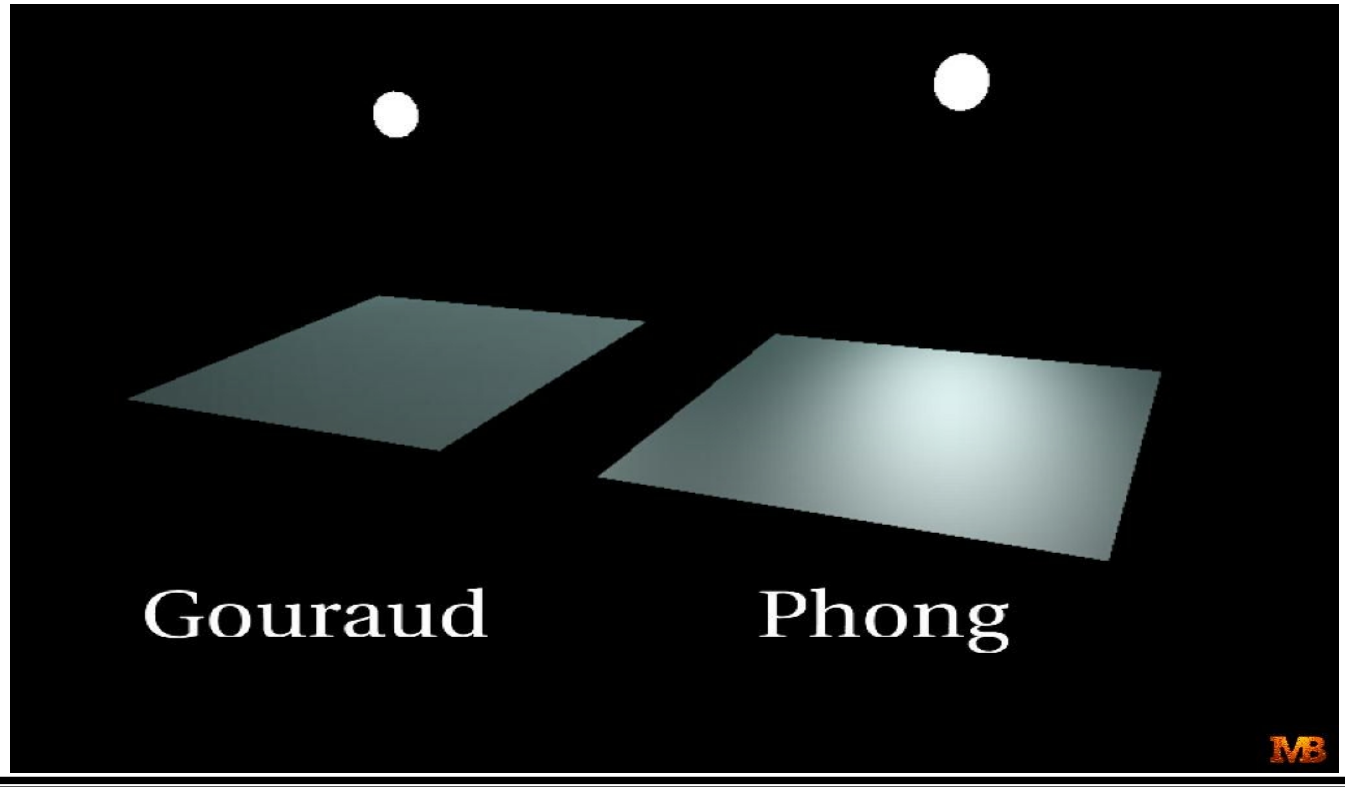
$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

- **Trade-off**: requires considerably more calculations

Note: Students are encouraged to read Fast Phong Shading which could be useful for project works

Gouraud versus Phong Shading

- Gouraud shading is faster than Phong shading
- Phong shading is more accurate



THANKS ~!!!~

MID POINT ALGORITHM PARAMETERS FOR SOME CURVES

Rectangular Hyperbola (x.y=a)

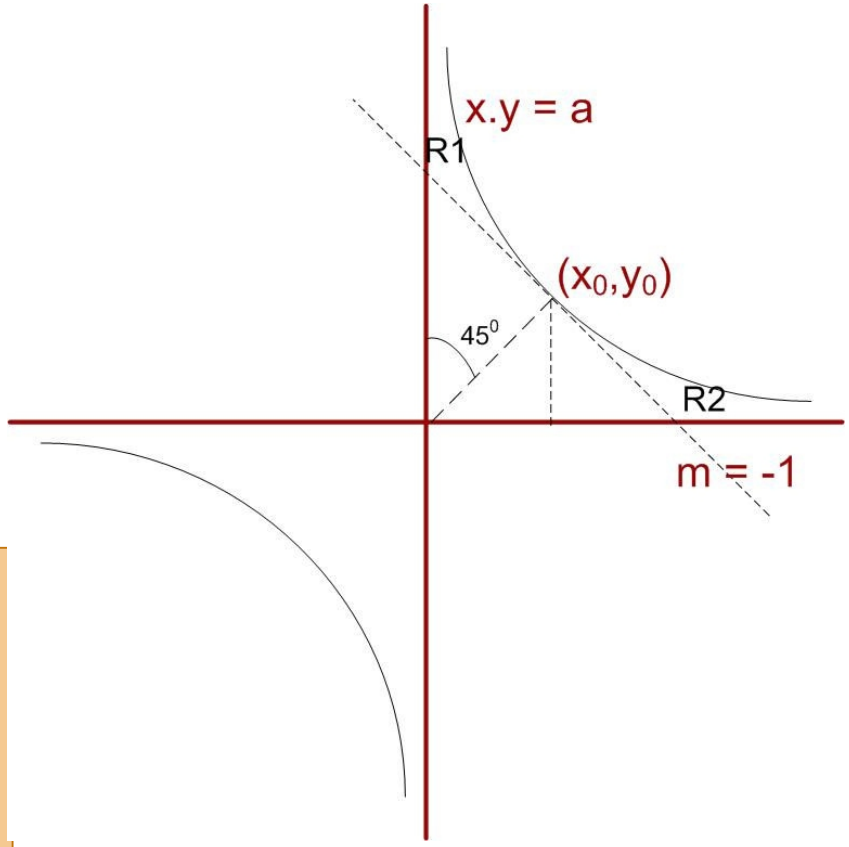
- At boundary of two regions

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

$$\frac{dy}{dx} = -1$$

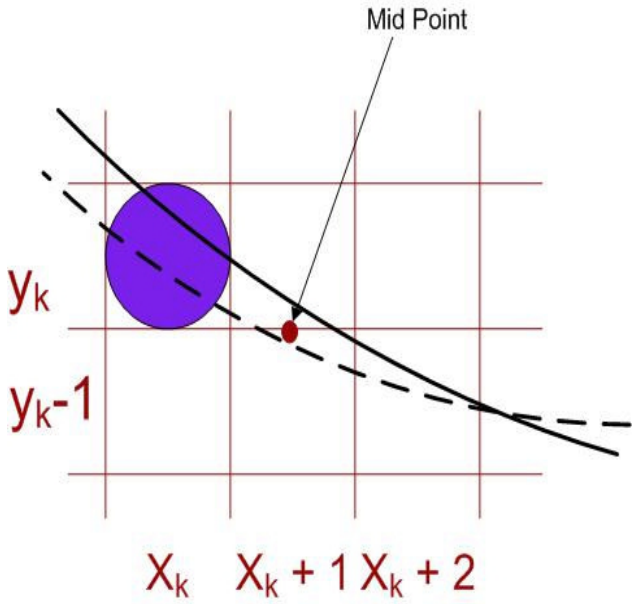
$$\Rightarrow x = \sqrt{a}$$

$$\Rightarrow (x_0, y_0) = (\sqrt{a}, \sqrt{a})$$



- |m| <1 for R2 so sample to x
- Since curve is symmetric about the point (x_0,y_0) we may take this point as starting point and calculate the pixels in R2 and find symmetric points in R1 by interchanging x and y
- The conjugate curve can be drawn by taking changing x to –x and y to -y

x.y = a



$$f(x, y) = x.y - a$$

$$f(x,y)\begin{cases} < 0, & if\ (x,y)\ \ is\ \ below\ \ the\ \ curve \\ = 0, & if\ (x,y)\ \ is\ \ on\ \ the\ \ curve \\ > 0, & if\ (x,y)\ \ is\ \ above\ \ the\ \ curve \end{cases}$$

x. y = a

$$p_k = f(x_k + 1, y_k - \frac{1}{2})$$

$$= (x_k + 1).(y_k - \frac{1}{2}) - a$$

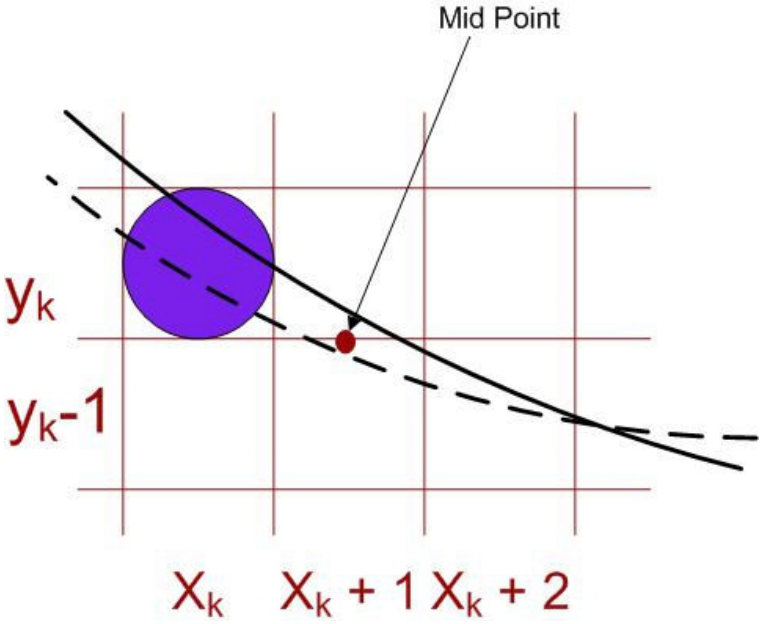
$$p_{k+1} = f(x_{k+1} + 1, y_{k+1} - \frac{1}{2})$$

$$= [(x_k + 1) + 1].(y_{k+1} - \frac{1}{2}) - a$$

$$p_{k+1} = p_k + (x_k + 1)(y_{k+1} - y_k) + (y_{k+1} - \frac{1}{2})$$

$$= p_k + (y_k - \frac{1}{2}) \quad if\ p_k < 0$$

$$= p_k - x_k + y_k - \frac{5}{2} \quad otherwise$$



$$p_0 = f(x_0 + 1, y_0 - \frac{1}{2})$$

$$= (x_0 + 1).(y_0 - \frac{1}{2}) - a$$

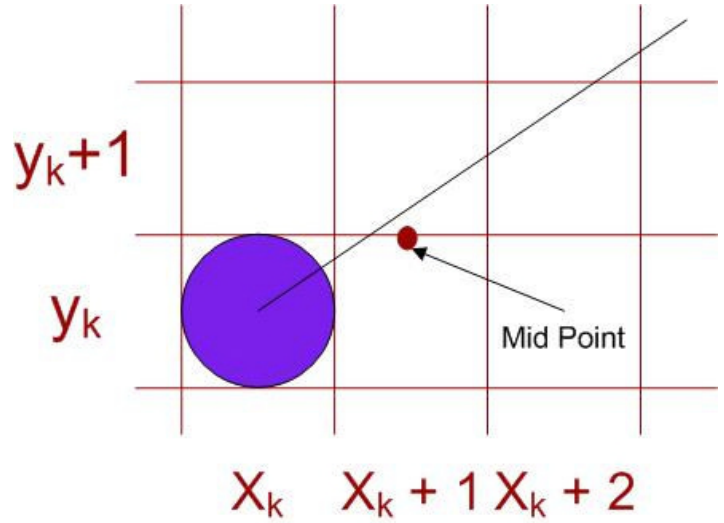
$$= (\sqrt{a} + 1).(\sqrt{a} - \frac{1}{2}) - a$$

$$= \frac{\sqrt{a}}{2} - \frac{1}{2}$$

Straight line (y = m.x + b)

- For |m|<1

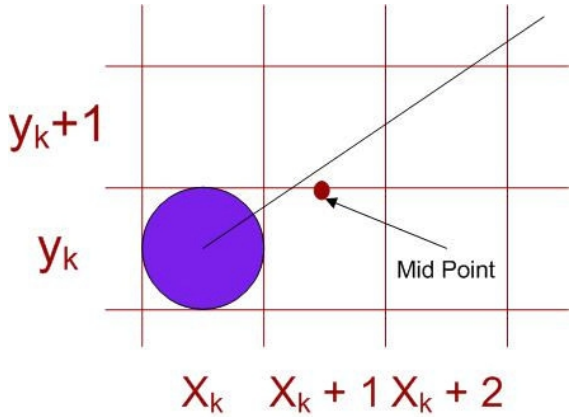
$$f(x,y) = m.x - y + b$$



$$f(x,y)\begin{cases} < 0, & if\ (x,y)\ \ is\ \ above\ \ the\ \ line \\ = 0, & if\ (x,y)\ \ is\ \ on\ \ the\ \ line \\ > 0, & if\ (x,y)\ \ is\ \ below\ \ the\ \ line \end{cases}$$

y = m.x + b

- (x_0, y_0) be starting point

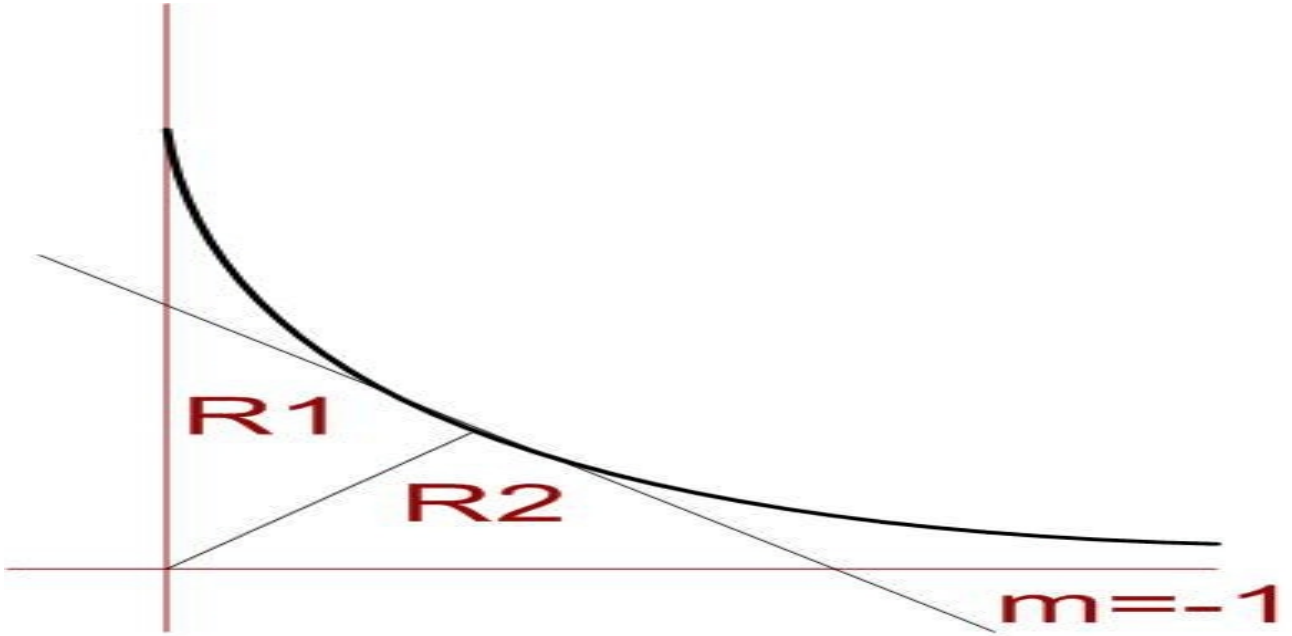


$$p_k = f(x_k + 1, y_k + \frac{1}{2})$$
$$= m.(x_k + 1) - (y_k + \frac{1}{2}) + b$$

$$p_{k+1} = f(x_{k+1} + 1, y_{k+1} + \frac{1}{2})$$
$$= m.[(x_k + 1) + 1] - (y_{k+1} + \frac{1}{2}) + b$$

$$p_{k+1} = p_k + m - (y_{k+1} - y_k)$$
$$= p_k + m \quad if \quad p_k < 0$$
$$= p_k + m - 1 \quad otherwise$$

$$p_0 = f(x_0 + 1, y_0 + \frac{1}{2})$$
$$= m.(x_0 + 1) - (y_0 + \frac{1}{2}) + b$$
$$= m - \frac{1}{2}$$



For the curve

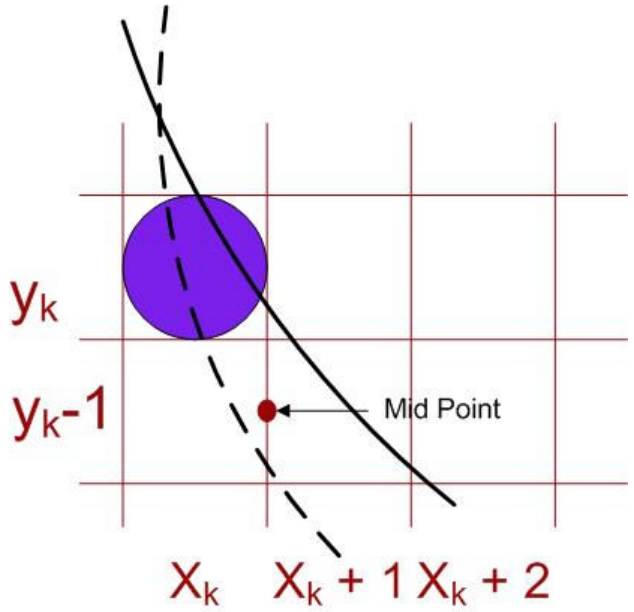
$$y = Ae^{-\alpha x}$$

- At boundary

$$\frac{dy}{dx} = -A\,\alpha\,.e^{-\alpha\,.x}$$

$$\frac{dy}{dx} = -1$$

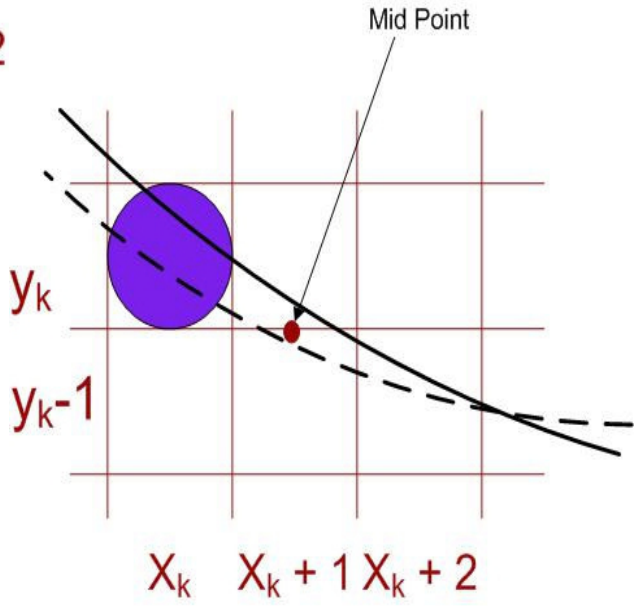
$$\Rightarrow \quad x = \frac{\ln(\quad A\quad .\alpha\quad)}{\alpha}$$



- Two regions
- Region 1 → |m|>1
- Region 2 → |m|<1

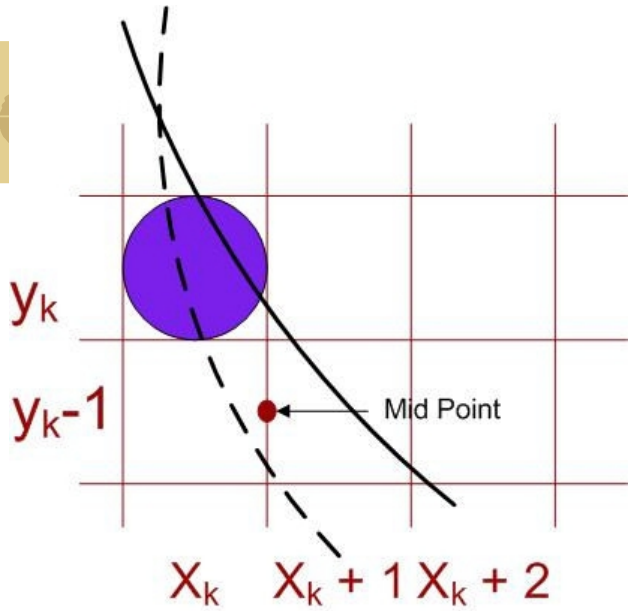
$$f(x,y) = Ae^{-\alpha x} - y$$

$$f(x,y)\begin{cases} <0, & if\,(x,y) \quad is \quad above \quad the \quad curve \\ =0, & if\,(x,y) \quad is \quad on \quad the \quad curve \\ >0, & if\,(x,y) \quad is \quad below \quad the \quad curve \end{cases}$$



Region 1

$$x < \frac{\ln(A.\alpha)}{\alpha}$$



$$p1_k = f(x_k + \frac{1}{2}, y_k - 1)$$
$$= Ae^{-\alpha(x_k + \frac{1}{2})} - (y_k - 1)$$
$$= Ae^{-\alpha/2}.e^{-\alpha.x_k} - y_k + 1$$

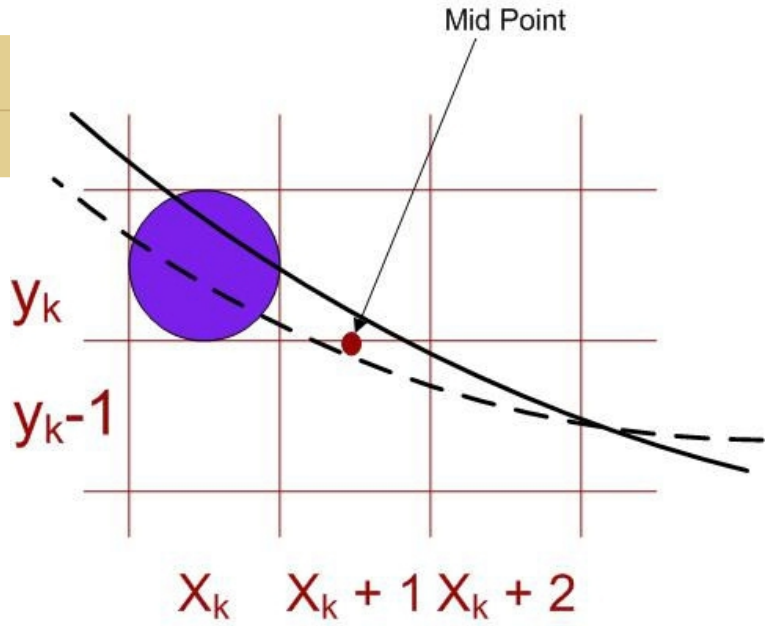
$$p1_{k+1} = f(x_{k+1} + \frac{1}{2}, y_{k+1} - 1)$$
$$= Ae^{-\alpha(x_{k+1} + \frac{1}{2})} - [(y_k - 1) - 1]$$
$$= Ae^{-\alpha/2}.e^{-\alpha.x_{k+1}} - y_k + 2$$

$$p1_0 = f(x_0 + \frac{1}{2}, y_0 - 1)$$
$$= Ae^{-\alpha(0 + \frac{1}{2})} - (A - 1)$$
$$= A(e^{-\alpha/2} - 1) + 1$$

$$p1_{k+1} = p1_k + Ae^{-\alpha/2}(e^{-\alpha.x_{k+1}} - e^{-\alpha.x_k}) + 1$$
$$= p1_k + 1 \quad if \quad p1_k < 0$$
$$= p1_k + Ae^{-\alpha/2}(e^{-\alpha.(x_k+1)} - e^{-\alpha.x_k}) + 1 \quad otherwise$$

Region 2

$$x > \frac{\ln(\ A\ .\alpha\ )}{\alpha}$$



$$p2_k = f(x_k + 1, y_k - \frac{1}{2})$$

$$= Ae^{-\alpha(x_k+1)} - (y_k - \frac{1}{2})$$

$$= Ae^{-\alpha}.e^{-\alpha.x_k} - y_k + \frac{1}{2}$$

$$p2_{k+1} = f(x_{k+1} + 1, y_{k+1} - \frac{1}{2})$$

$$= Ae^{-\alpha(x_k+2)} - (y_{k+1} - \frac{1}{2})$$

$$= Ae^{-2\alpha}.e^{-\alpha.x_k} - y_{k+1} + \frac{1}{2}$$

$$p2_0 = f(x_0 + 1, y_0 - \frac{1}{2})$$

$$= Ae^{-\alpha}.e^{-\alpha.x_0} - y_0 + \frac{1}{2}$$

$$p2_{k+1} = p2_k + A(e^{-\alpha} - 1).e^{-\alpha.(x_k+1)} - (y_{k+1} - y_k)$$

$$= p2_k + A(e^{-\alpha} - 1).e^{-\alpha.(x_k+1)} \quad if\ p2_k > 0$$

$$= p2_k + A(e^{-\alpha} - 1).e^{-\alpha.(x_k+1)} + 1 \quad otherwise$$