Computer Graphics
MCA-203

UNIT-I
Scan Conversion and Transformations

Learning Objectives

In this unit, we'll cover the following:

- Graphics Primitives
- Display Devices
- Scan Conversion
  - Point
  - Line
  - Circle etc.
  - Filled-Area Primitives
- Transformations
  - Two Dimensional (2D)
  - Three Dimensional (3D)

Graphics Primitives

Contents

We'll cover the following:

- What is Computer Graphics
- Its Application
- What is Pixel
- Black & White and Colored images
- Display File
- Frame Buffer
- Aspect Ratio
- Resolution
- Color Models

What is Computer Graphics

- Refers to creation, storage and manipulation of pictures and drawings using a digital computer
- Effective tool for presenting information
- It increases the communication bandwidth between humans and machines

Applications of Computer Graphics

- Science & engineering
- Medicine
- Business
- Government
- Art
- Entertainment
- Advertising
- Education and training

Contents

We'll cover the following:

- What is Computer Graphics
- Its Application
- What is Pixel
- Black & White and Colored images
- Display File
- Frame Buffer
- Aspect Ratio
- Resolution
- Color Models

Contents

We'll cover the following:

- What is Computer Graphics
- Its Application
- What is Pixel
- Black & White and Colored images
- Display File
- Frame Buffer
- Aspect Ratio
- Resolution
- Color Models

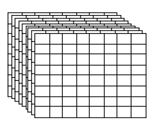
Contents

We'll cover the following:

- What is Computer Graphics
- Its Application
- What is Pixel
- Black & White and Colored images
- Display File
- Frame Buffer
- Aspect Ratio
- Resolution
- Color Models

Frame Buffers

- 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 Plane*s or *Bit Depth* is the number of bits corresponding to each pixel.
- A typical frame buffer resolution might be
  - 640 x 480 x 8
  - 1280 x 1024 x 8
  - 1280 x 1024 x 24



Contents

We'll cover the following:

- What is Computer Graphics
- Its Application
- What is Pixel
- Black & White and Colored images
- Display File
- Frame Buffer
- Aspect Ratio
- Resolution
- Color Models

Contents

We'll cover the following:

 13

Color Models

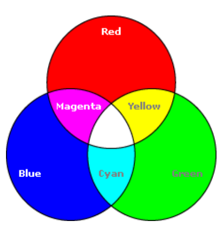
There are two most common Color models :

- RGB Color Model (Red-Green-Blue)
  - Additive model
  - Additive color models use **light** to display color
  - Colors perceived in additive models are the result of **transmitted** light
- CMYK Color Model (Cyan-Magenta-Yellow-blacK)
  - Subtractive model
  - subtractive models use printing **inks**.
  - Colors perceived in subtractive models are the result of **reflected** light

 14

RGB Color Model

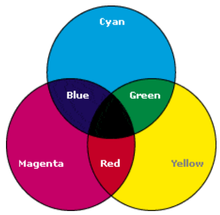
1) **RGB Color Model**:
* Additive color model.
* For computer displays.
* Uses light to display color.
* Colors result from transmitted light.
* Red + Green + Blue = White.



 15

CYMK Color Model

2) CMYK Color Model:
* Subtractive color model.
* For printed material.
* Uses ink to display color.
* Colors result from reflected light.
* Cyan + Magenta + Yellow = Black.



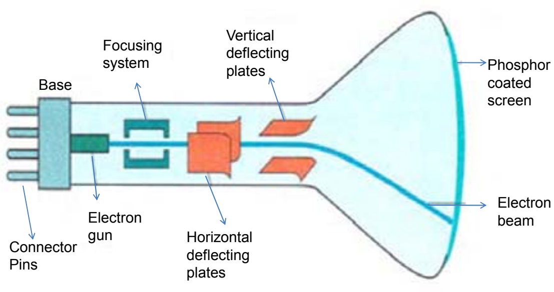
Learning Objectives

In this unit, we'll cover the following:
- Graphics Primitives
- Display Devices
- Scan Conversion
  - Point
  - Line
  - Circle etc.
  - Filled-Area Primitives
- Transformations
  - Two Dimensional (2D)
  - Three Dimensional (3D)

Display Devices

Contents

Refresh CRT



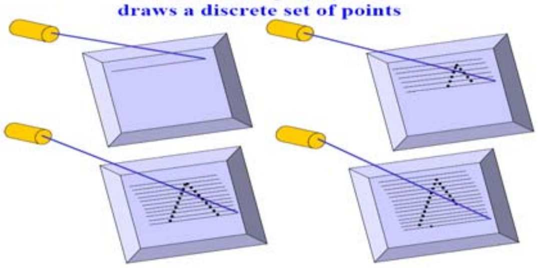
Raster Scan Displays

- The most common type of monitor employing a CRT is raster scan display.
- In Raster scan display the electron beam is swept across the screen one row at a time from top to bottom.
- Raster: A rectangular array of points or dots
- Pixel: One dot or picture element of the raster. Its intensity range for pixels depends on capability of the system
- Scan line: A row of pixels
- Picture elements are stored in a memory called frame buffer
- A special memory is used to store the image with scan-out synchronous to the raster. We call this the *frame buffer*

Disadvantage
Raster system produces jagged lines that are plotted as discrete points

Raster Scan Displays cont..

Raster-scan display system draws a discrete set of points



Raster Scan Displays cont..

Horizontal Retrace

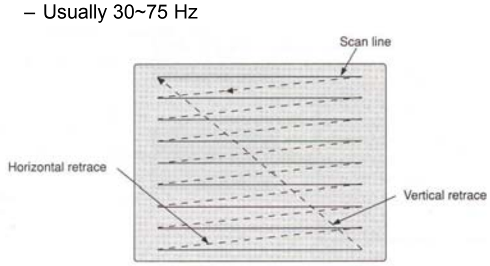
- At the end of each scan line , the electron beam returns to the left side of the screen to begin displaying the next scan line. The return to the left of the screen , after refreshing each scan line is called horizontal retrace of electron beam.

Vertical retrace

- At the end of each frame the electron beam returns to the top left corner of the screen to begin the next frame.

Raster Scan Displays cont..

- Refresh Rate
  - Usually 30~75 Hz



Raster Scan Displays cont..

Interlacing

- First, all points on the even-numbered (solid) scan-lines are displayed
- Then all points along odd-numbered (dashed) lines are displayed

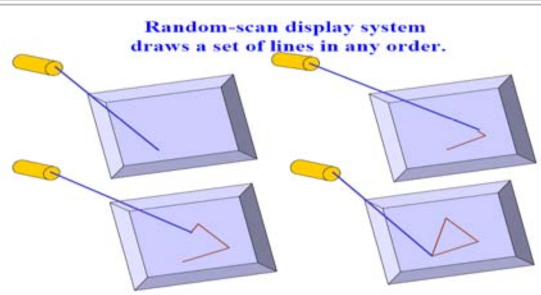
Random Scan Display

- The electron beam is directed only to the parts of the screen where a picture is to be drawn.
- Picture definition is stored as a set of line drawing commands in an area of memory referred to as refresh display file.
- To display a specified picture ,the system cycles through a set of commands in the display file , drawing each component line after processing all lines drawing commands the s/m cycle back to the first line command in the list.

Advantage

Has high resolution since picture definition is stored as line drawing commands

Random Scan Display cont...



Color CRT Monitors

2 Basic Techniques for Color Display

- Beam Penetration Method
- Shadow Mask Method

Beam-Penetration Method

•Used with random scan monitors

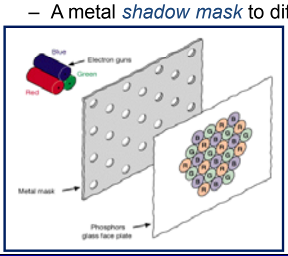
•The screen has two layers of phosphor: usually ***red and green***

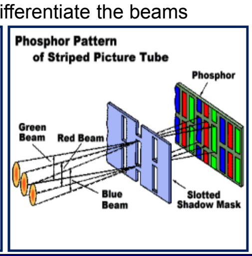
•The displayed color depends on how far the electron beam penetrates through the two layers.

•A beam of slow electrons excites only the outer of the red layer, a beam of fast electrons penetrates through the red layer and excites the inner green layer, and at intermediate beam speeds, combinations of the two colors are emitted to show other colors (yellow & orange)

Shadow-Mask Method

- Used with raster scan monitors
- Three electron guns
- A metal *shadow mask* to differentiate the beams



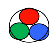


Shadow-Mask Method cont..

•The Shadow mask in the previous image is known as the ***delta-delta*** shadow-mask.

•The 3 electron beams are deflected and focused as a group onto the shadow mask, which contains a series of holes aligned with the phosphor-dot patterns.

•The 3 beams pass through a hole in the shadow mask and activate a dot triangle, which appears as a small color spot on the screen.



•A second arrangement of the 3 electron guns is ***in-line***
Where the corresponding red-green-blue color dots on the screen are aligned along one scan – line instead of a triangular pattern.

Controlling Colors in shadow mask

•Different colors can be obtained by varying the intensity levels of the three electron beams.

•Example: Simply turning off the red and green guns, we get only the color coming from the blue phosphor.

•Yellow = Green + Red
•Magenta = Blue + Red
•Cyan = Blue + Green

•White is produced when all the 3 guns possess equal amount of intensity.

Contents

Learning Objectives

In this unit, we'll cover the following:

- Graphics Primitives
- Display Devices
- Scan Conversion
  - Point
  - Line
  - Circle etc.
  - Filled-Area Primitives
- Transformations
  - Two Dimensional (2D)
  - Three Dimensional (3D)

Scan Conversion

Scan Conversion

Output Primitives

- Basic geometric structures used to describe scenes.
- Can be grouped into more complex structures
- Example : Point, straight line segments, circles and other conic sections, polygon color areas and character strings
- Construct the vector picture

Rasterization

The process of determining the appropriate pixels for representing picture or graphic object

Scan conversion

It is the final step of rasterization . It converts picture definition into a set of pixel-intensity values for storage in the frame buffer.

Scan Conversion cont..

Scan Converting a Point

- A mathematical point (x,y) needs to be scan converted to a pixel at location (x´, y´).
- With a CRT monitor, the electron beam is turned on to illuminate the screen phosphor at the selected location.

Scan Conversion cont..

Scan Converting a Line

Line drawing is accomplished by calculating intermediate positions along the line path between two specified endpoint positions.

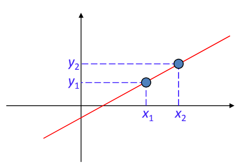
Cartesian equation:

$y = mx + c$

Where

m – slope

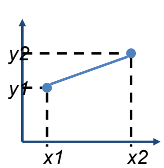
c – y-intercept



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

The Digital Differential Analyzer (DDA) Algorithm

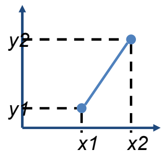
- **Digital Differential Analyzer**
  - 0 < Slope <= 1
    - Unit x interval = 1



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

The DDA Algorithm cont..

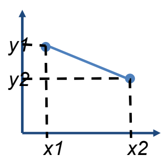
- **Digital Differential Analyzer**
  - 0 < Slope <= 1
    - Unit x interval = 1
  - Slope > 1
    - Unit y interval = 1



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

The DDA Algorithm cont..

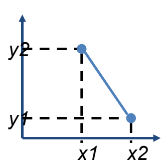
- **Digital Differential Analyzer**
  - 0 < Slope <= 1
    - Unit x interval = 1
  - Slope > 1
    - Unit y interval = 1
  - -1 <= Slope < 0
    - Unit x interval = -1



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

The DDA Algorithm cont..

- **Digital Differential Analyzer**
  - 0 < Slope <= 1
    - Unit x interval = 1
  - Slope > 1
    - Unit y interval = 1
  - -1 <= Slope < 0
    - Unit x interval = -1
  - Slope < -1
    - Unit y interval = -1



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

The DDA Algorithm cont..

DDA ALGORITHM

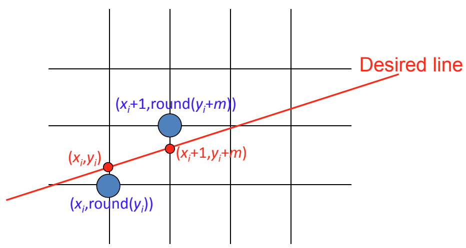
1. START
2. Get the values of the starting and ending co-ordinates i.e. ,(xa,ya) and (xb,yb).
3. Find the value of slope m

m=dy/dx=((yb-ya)/(xb-xa))
4. If lml ≤ 1 then Δx=1,Δy=mΔx

xk+1=xk+1,yk+1=yk+m
5. If lml ≥ 1 then Δy=1,Δx=Δy/m

xk+1=xk+1/m,yk+1=yk+1
6. STOP

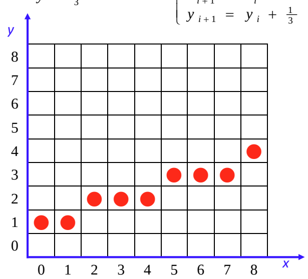
The DDA Algorithm cont..



Example (DDA)

x	y	round(y)
0	1	1
1	4/3	1
2	5/3	2
3	2	2
4	7/3	2
5	8/3	3
6	3	3
7	10/3	3
8	11/3	4

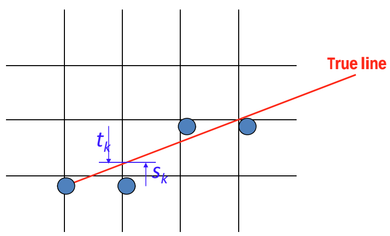
$$y = \frac{1}{3}x + 1 \qquad \begin{cases} 0 \le m \le 1 \\ x_{i+1} = x_i + 1 \\ y_{i+1} = y_i + \frac{1}{3} \end{cases}$$



The Bresenham's Line Algorithm

- The Bresenham's algorithm is another incremental scan conversion algorithm
- The big advantage of this algorithm is that it uses only integer calculations

Bresenham's Line Algorithm cont..



- For a given value of x
  - one pixel lies at distance t_k above the line, and
  - one pixel lies at distance s_k below the line

Bresenham's Line Algorithm cont..

Decision parameter

$$p_k = (s_k - t_k)$$

- If $p_k < 0$, then closest pixel is below true line (s_k smaller)
- If $p_k \geq 0$, then closest pixel is above true line (t_k smaller)

- We must calculate the new values for p_k as we move along the line.

Bresenham's Line Algorithm cont..

Algorithm
1. Input line end points
2. Load (x_0,y_0) to plot the first point.
3. Calculate dx,dy, 2dy and 2dy-2dx and obtain the starting value of the decision parameter as p_0=2dy-dx
4. At each x_k along the line, starting at k=0, perform the following test

   if p_k <0, the next point plot is (x_k+1, y_k) and

   $p_{k+1}=p_k+2dy$

   Other wise, the next point to pot is (x_k+1, y_k+1) and

   $p_{k+1}=p_k+2dy-2dx$
5. Repeat step-4 dx times.

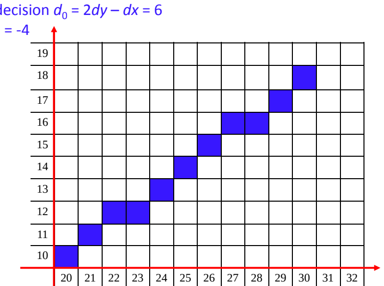
Example (Bresenham's Line Algorithm)

Draw a line from (20,10) to (30,18)

$dx = 10$ $dy = 8$ initial decision $d_0 = 2dy - dx = 6$

Also $2dy = 16$, $2(dy - dx) = -4$

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



Bresenham's Line Algorithm cont..

Special cases

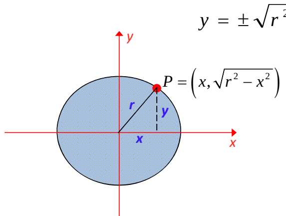
•Special cases can be handled separately
- Horizontal lines ($\Delta y = 0$)
- Vertical lines ($\Delta x = 0$)
- Diagonal lines ($|\Delta x| = |\Delta y|$)

•directly into the frame-buffer without processing them through the line-plotting algorithms.

Circle Equations (Cartesian form)

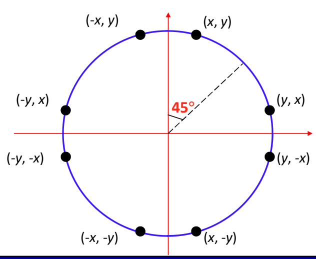
The equation for a circle is: $x^2 + y^2 = r^2$
where r is the radius of the circle
So, we can write a simple circle drawing algorithm by solving the equation for y at unit x intervals using:

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



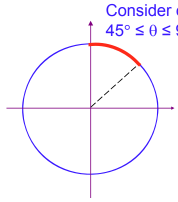
Circle Algorithms

- Use 8-fold symmetry and only compute pixel positions for the 45° sector.



Bresenham's Circle Algorithm

Consider only 45° ≤ θ ≤ 90°



General Principle

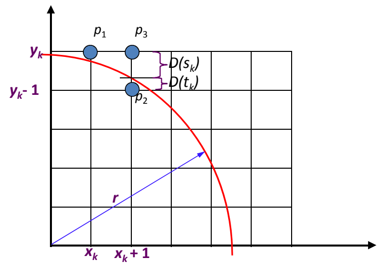
- The circle function:

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

- and

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

Bresenham's Circle Algorithm Cont..



After point p_1, do we choose p_2 or p_3?

Mid-point Circle Algorithm

MID-POINT CIRCLE ALGORITHM

1. Input radius r and circle centre (x_c, y_c), then set the coordinates for the first point on the circumference of a circle centred on the origin as:
$$(x_0, y_0) = (0, r)$$
2. Calculate the initial value of the decision parameter as:
$$p_0 = \frac{5}{4} - r$$
3. Starting with $k = 0$ at each position x_k, perform the following test. If $p_k < 0$, the next point along the circle centred on $(0, 0)$ is (x_k+1, y_k) and:
$$p_{k+1} = p_k + 2x_{k+1} + 1$$

Mid-point Circle Algorithm Cont..

Otherwise the next point along the circle is (x_k+1, y_k-1) and:
$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

4. Determine symmetry points in the other seven octants
5. Move each calculated pixel position (x, y) onto the circular path centred at (x_c, y_c) to plot the coordinate values: $x = x + x_c$ $\quad y = y + y_c$
6. Repeat steps 3 to 5 until $x >= y$

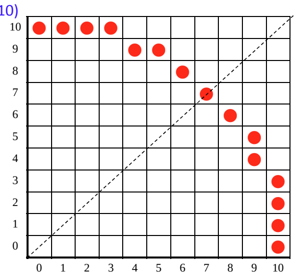
Example (Mid Point Circle Algorithm)

$r = 10$

$p_0 = 1 - r = -9$ (if r is integer round $p_0 = 5/4 - r$ to integer)

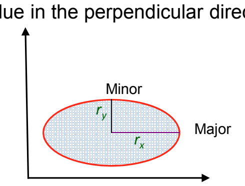
Initial point $(x_0, y_0) = (0, 10)$

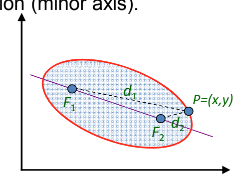
i	p_i	x_{i+1}, y_{i+1}	$2x_{i+1}$	$2y_{i+1}$
0	-9	(1, 10)	2	20
1	-6	(2, 10)	4	20
2	-1	(3, 10)	6	20
3	6	(4, 9)	8	18
4	-3	(5, 9)	10	18
5	8	(6, 8)	12	16
6	5	(7, 7)	14	14



Ellipse-Generating Algorithms

•**Ellipse** – A modified circle whose radius varies from a maximum value in one direction (major axis) to a minimum value in the perpendicular direction (minor axis).





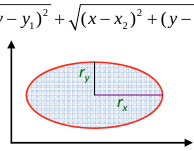
- The sum of the two distances d_1 and d_2, between the fixed positions F_1 and F_2 (called the *foci* of the ellipse) to any point P on the ellipse, is the same value, i.e.
  $d_1 + d_2$ = constant

Ellipse-Generating Algorithms Cont..

Ellipse Properties

•Expressing distances d_1 and d_2 in terms of the focal coordinates $F_1 = (x_1, x_2)$ and $F_2 = (x_2, y_2)$, we have:

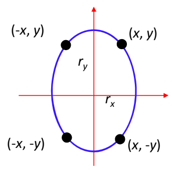
$$\sqrt{(x-x_1)^2 + (y-y_1)^2} + \sqrt{(x-x_2)^2 + (y-y_2)^2} = \text{constant}$$



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

Ellipse-Generating Algorithms Cont..

- Symmetry between quadrants
- Not symmetric between the two octants of a quadrant
- Thus, we must calculate pixel positions along the elliptical arc through one quadrant and then we obtain positions in the remaining 3 quadrants by symmetry



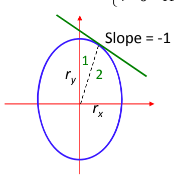
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Vaishali Joshi, Assistant Professor – Unit 1 61

Ellipse-Generating Algorithms Cont..

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

- Decision parameter:

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



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

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Vaishali Joshi, Assistant Professor – Unit 1 62

Midpoint Ellipse Algorithm

1. Input r_x, r_y, and 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 parameter in region 1 as

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

3. At each x_i position, starting at $i = 0$, if $p1_i < 0$, the next point along the ellipse centered on (0, 0) is $(x_i + 1, y_i)$ and

$$p1_{i+1} = p1_i + 2r_y^2 x_{i+1} + r_y^2$$

otherwise, the next point is $(x_i + 1, y_i - 1)$ and

$$p1_{i+1} = p1_i + 2r_y^2 x_{i+1} - 2r_x^2 y_{i+1} + r_y^2$$

and continue until

$$2r_y^2 x \geq 2r_x^2 y$$

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Vaishali Joshi, Assistant Professor – Unit 1 63

Midpoint Ellipse Algorithm cont..

4. (x_0, y_0) is the last position calculated in region 1. Calculate the initial parameter in region 2 as

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

5. At each y_i position, starting at $i = 0$, if $p2_i > 0$, the next point along the ellipse centered on (0, 0) is $(x_i, y_i - 1)$ and

$$p2_{i+1} = p2_i - 2r_x^2 y_{i+1} + r_x^2$$

otherwise, the next point is $(x_i + 1, y_i - 1)$ and

$$p2_{i+1} = p2_i + 2r_y^2 x_{i+1} - 2r_x^2 y_{i+1} + r_x^2$$

Use the same incremental calculations as in region 1. Continue until $y = 0$.

6. For both regions determine 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 coordinate values

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

Example (Midpoint Ellipse Algorithm)

$r_x = 8$, $r_y = 6$

$2r_y^2x = 0$ (with increment $2r_y^2 = 72$)

$2r_x^2y = 2r_x^2r_y$ (with increment $-2r_x^2 = -128$)

Region 1

$(x_0, y_0) = (0, 6)$ $p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2 = -332$

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

Move out of region 1 since $2r_y^2x > 2r_x^2y$

Example (Midpoint Ellipse Algorithm) cont..

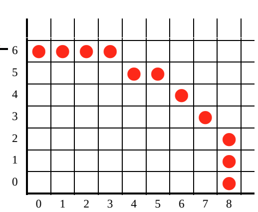
Region 2

$(x_0, y_0) = (7, 3)$ (Last position in region 1)

$p2_0 = f_{ellipse}(7 + \frac{1}{2}, 2) = -151$

Stop at $y = 0$

i	p_i	x_{i+1}, y_{i+1}	$2r_y^2x_{i+1}$	$2r_x^2y_{i+1}$
0	-151	(8, 2)	576	256
1	233	(8, 1)	576	128
2	745	(8, 0)	-	-



Filled Area Primitives

•**Polyline** - A chain of connected line segments.
•**Polygon** - When starting point and terminal point of any polyline is same, i,e when polyline is closed then it is called polygon

Filled Area Primitives cont..

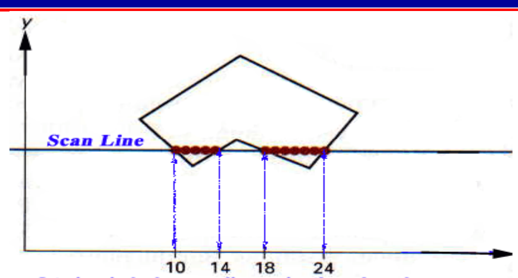
Two Basic approaches to Area-Filling –

•Scan-line Method
Determine the overlap intervals for scan lines that cross the area. It is typically used in general graphics packages to fill polygons, circles, ellipses

•Fill Method
Start from a given interior position and paint outward from this point until we encounter the specified boundary conditions. Useful with more complex boundaries and in interactive painting systems.

1. Boundary Fill
2. Flood Fill

Filled Area Primitives cont..



Interior pixels along scan line passing through a polygon area

Filled Area Primitives cont..

Boundary – Fill Algorithms

- Start at a point inside a region and paint the interior outward toward the boundary. If the boundary is specified in a single color, the fill algorithm proceeds outward pixel by pixel until the boundary color is encountered.
- It is useful in interactive painting packages, where interior points are easily selected.
- The inputs of the this algorithm are:
  - Coordinates of the interior point (x, y)
  - Fill Color
  - Boundary Color

Filled Area Primitives cont..

- Sometimes we want to fill in (or recolor) an area that is not defined within a single color boundary. We can paint such areas by replacing a specified interior color instead of searching for *a* boundary color value. This approach is called a flood-fill algorithm.

We start from a specified interior point (*x, y*) and reassign all pixel values that are currently set to a given interior color with the desired fill color.

Learning Objectives

In this unit, we'll cover the following:

- Graphics Primitives
- Display Devices
- Scan Conversion
  - Point
  - Line
  - Circle etc.
  - Filled-Area Primitives
- Transformations
  - Two Dimensional (2D)
  - Three Dimensional (3D)

2D Transformations

2D Transformation

Given a 2D object, transformation is to change the object's

- Position (translation)
- Size (scaling)
- Orientation (rotation)
- Shapes (shear)
- Reflection

Translation

•A translation moves all points in an object along the same straight-line path to new positions.

•The path is represented by a vector, called the translation or shift vector.

•We can write the components:

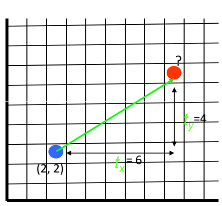
$p'_x = p_x + t_x$

$p'_y = p_y + t_y$

•or in matrix form:

P' = P + T

$$\begin{pmatrix} x' \\ Y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$



Scaling

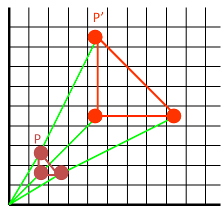
- Scaling changes the size of an object and involves two scale factors, S_x and S_y for the x- and y- coordinates respectively.
- Scales are about the origin.
- We can write the components:

$$p'_x = s_x \cdot p_x$$
$$p'_y = s_y \cdot p_y$$

or in matrix form:

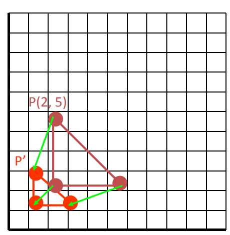
$$P' = S \cdot P$$

Scale matrix as: $S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$



Scaling cont..

- If the scale factors are in between 0 and 1 ➔ the points will be moved closer to the origin ➔ the object will be smaller.
- Example :
  - P(2, 5), Sx = 0.5, Sy = 0.5
  - Find P' ?

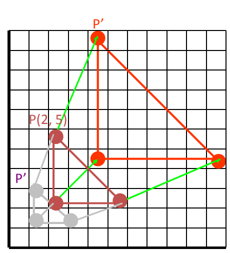


Scaling cont..

- If the scale factors are in between 0 and 1 ➔ the points will be moved closer to the origin ➔ the object will be smaller.
- Example :
  - P(2, 5), Sx = 0.5, Sy = 0.5
  - Find P' ?

•If the scale factors are larger than 1 ➔ the points will be moved away from the origin ➔ the object will be larger.

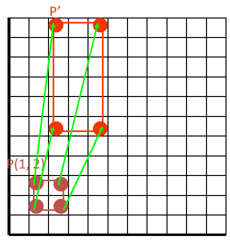
- Example :
  - P(2, 5), Sx = 2, Sy = 2
  - Find P' ?



Scaling cont..

- If the scale factors are the same, $S_x = S_y$ ➔ uniform scaling
- Only change in size (as previous example)

- If $S_x \neq S_y$ ➔ differential scaling.
- Change in size and shape
- Example : square → rectangle
  - P(1, 3), $S_x = 2$, $S_y = 5$, P' ?

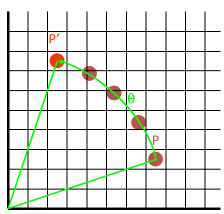


What does scaling by 1 do?
What is that matrix called?
What does scaling by a negative value do?

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Vaishali Joshi, Assistant Professor – Unit 1 79

Rotation

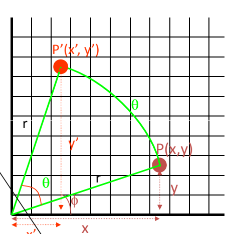
- A rotation repositions all points in an object along a circular path in the plane centered at the pivot point.
- First, we'll assume the pivot is at the origin.



© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Vaishali Joshi, Assistant Professor – Unit 1 80

Rotation cont..

- Review Trigonometry
  - => $\cos \phi = x/r$, $\sin \phi = y/r$
    - $x = r.\cos \phi$, $y = r.\sin \phi$
  - => $\cos (\phi + \theta) = x'/r$
  - $x' = r.\cos (\phi + \theta)$
  - $x' = r.\cos\phi\cos\theta - r.\sin\phi\sin\theta$
  - $x' = x.\cos \theta - y.\sin \theta$
  - =>$\sin (\phi + \theta) = y'/r$
  - $y' = r.\sin (\phi + \theta)$
  - $y' = r.\cos\phi\sin\theta + r.\sin\phi\cos\theta$
  - $y' = x.\sin \theta + y.\cos \theta$



Identity of Trigonometry

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Vaishali Joshi, Assistant Professor – Unit 1 81

Rotation cont..

- We can write the components:

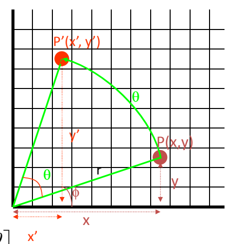
  $p'_x = p_x \cos\theta - p_y \sin\theta$

  $p'_y = p_x \sin\theta + p_y \cos\theta$

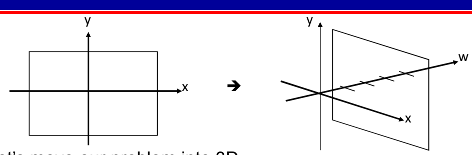
- or in matrix form:

  $P' = R \cdot P$

- θ can be clockwise (-ve) or counterclockwise (+ve as our example).
- Rotation matrix $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$



Homogeneous Coordinates



•Let's move our problem into 3D.

•Let point (*x*, *y*) in 2D be represented by point (*x*, *y*, 1) in the new space.

•Scaling our new point by any value a puts us somewhere along a particular line: (a*x*, a*y*, a).

•A point in 2D can be represented in many ways in the new space.

•(2, 4) ----------→ (8, 16, 4) or (6, 12, 3) or (2, 4, 1) or etc.

Homogeneous Coordinates cont..

- We can always map back to the original 2D point by dividing by the last coordinate
- (15, 6, 3) ---→ (5, 2).
- (60, 40, 10) -→ ?.
- Why do we use 1 for the last coordinate?
- The fact that all the points along each line can be mapped back to the same point in 2D gives this coordinate system its name – homogeneous coordinates.

Homogeneous Coordinates cont..

- Translation

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

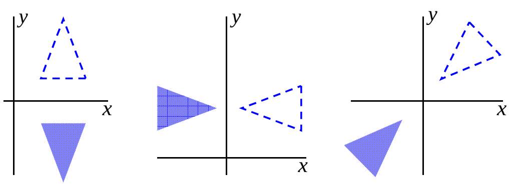
- Rotation

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

- Scaling

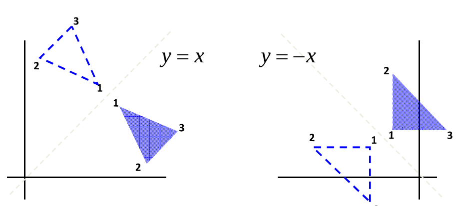
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Reflection



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

Reflection cont..



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

Shear Transformation

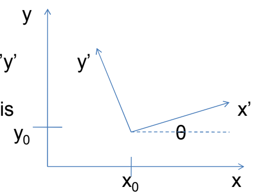
The Shear Transformation cause the image to slant. X-Shear maintains the y-coordinates but changes x values which cause the vertical line to tilt left or right. The Y-shear preserves all the x coordinate values but shifts the y coordinate.

- Y coordinates are unaffected, but x cordinates are translated linearly with y
  - $y' = y$
  - $x' = x + y * h$

Transformations between coordinate systems

This is done in two steps :

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



$M_{xyx'y'} = R(-\theta) \cdot T(-x_0,-y_0)$

Affine Transformations

Properties of affine transformations :

•Each of the transformed coordinates x' and y' is a linear function of the original coordinates x and y

•Parallel lines are transformed into parallel lines

•Finite points map to finite points

•An affine transformation involving only rotation, translation and reflection preserves angles and lengths, as well as parallel lines

3D Transformations

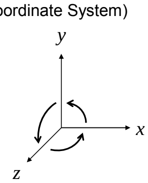
 91

3D Concepts

Everything we describe in our 3D worlds, e.g. vertices to describe objects, speed of objects, forces on objects, will be defined by 3D VECTORS i.e. triplets of 3 real values V = (x, y, z)

3D Euclidean Coordinate System

(or 3D Cartesian Coordinate System)



 92

Basic 3D Transformations

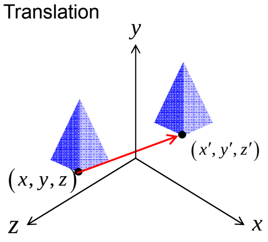
- To move/animate objects or to change the camera's position we have to transform the vertices defining our objects.
- Homogeneous coordinates: *(x,y,z)=(hx,hy,hz,h)*
- Transformations are now represented as 4x4 matrices
- Basic 3D transformations are
  - Translation
  - Rotation
  - Scaling

 93

3D Translation

Very similar to 2D. Using 4x4 matrices rather than 3x3.

Translation



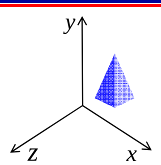
$$x' = x + t_x$$

$$y' = y + t_y$$

$$z' = z + t_z$$

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

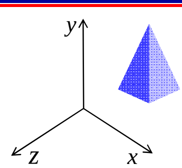
3D Scaling



$$x' = x \cdot S_x$$

$$y' = y \cdot S_y$$

$$z' = x \cdot S_z$$

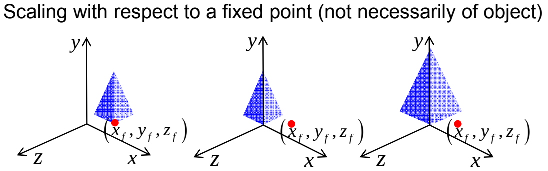


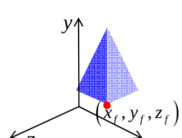
Enlarging object also moves it from origin

$$\mathbf{P}' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{S} \cdot \mathbf{P}$$

3D Scaling cont..

Scaling with respect to a fixed point (not necessarily of object)





$$\mathbf{T} \cdot \mathbf{S} \cdot \mathbf{T}^{-1} = \begin{bmatrix} S_x & 0 & 0 & (1-S_x)x_f \\ 0 & S_y & 0 & (1-S_y)y_f \\ 0 & 0 & S_z & (1-S_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

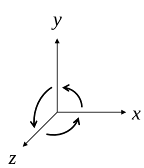
3D Rotation

- Coordinate-Axes Rotations
  - X-axis rotation
  - Y-axis rotation
  - Z-axis rotation
- General 3D Rotations
  - Rotation about an axis that is parallel to one of the coordinate axes
  - Rotation about an arbitrary axis

3D Rotation

- Positive Rotations are defined as follows:

Axis of rotation is	Direction of positive rotation is
x	y to z
y	z to x
z	x to y



Right-hand rule for rotations by positive θ

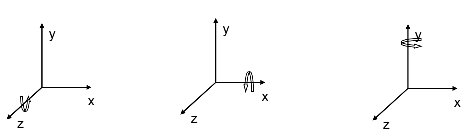
3D Rotation cont..

- Z-Axis Rotation
- X-Axis Rotation
- Y-Axis Rotation

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

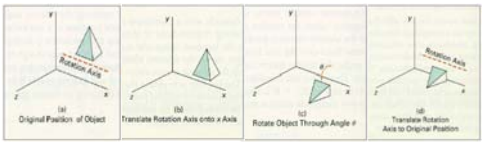
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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



General 3D Rotation

- Rotation about an Axis that is Parallel to One of the Coordinate Axes
  - **Translate** the object so that the rotation axis coincides with the parallel coordinate axis
  - Perform the specified **rotation** about that axis
  - **Translate** the object so that the rotation axis is moved back to its original position



 100

3D Reflections

Reflection

Reflection relative to xy plane.

$$\begin{pmatrix} x' & y' & z' & 1 \end{pmatrix} = \begin{pmatrix} x & y & z & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

 101

3D shears

Shearing

Z-axis shear

-Where a and b are the shear factors for x and y respectively

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

 102

Conclusion

In this unit we discuss scan conversion for Point, Line, Circle and ellipse. Along with C Programs for the Scan conversion.

Secondly we under stands the 2D and 3D transformation with some examples.

Summary

Circles and Ellipses can be efficiently and accurately scan converted using midpoint methods and taking curve symmetry into account.

The basic geometric transformation are translation, rotation and scaling. Translation moves an object in a Straight line path from one position to another. Rotation movers an object from one position to another in a circular path around a specific rotation point. Scaling Changes the dimensions of an object relative to a specified fixed point.

Review Questions

Short answer type Questions

Q1. Discuss Midpoint Circle Drawing algorithm with the help of an example.
Q2. Discuss Bresenham's algorithm for Scan converting a Line.
Q3. Discuss 2D Translation and Scaling with examples.
Q4. Compute the intermediate prints on the line drawn from (0, 0) to (5, 10) using DDA algorithm.
Q5. What is scan conversion?
Q6. What do you mean by Composite transformations? Explain with the help of an example.

Review Questions cont..

Long answer type Questions

Q1. Explain various 2D transformation with examples
Q2. Explain various 3D transformation with examples
Q3. Derive the 2D rotational transformation matrix.
Q4.What do you mean by rotation in 3D.
Q5.Find the matrix that represents rotation of an object by 30 degree about origin in 2D.
Q6.Find the transformation that scales (with respect to origin)by

a) 'a' units in the X-direction.
b) 'b' units in the Y-direction and
c) Simultaneously 'a' units in the X-direction and 'b' units in the Y direction

Suggested Reading / References

[1]. Donnald Hearn and M. Pauline Baker, "Computer Graphics", PHI.

[2]. Foley James D, "Computer Graphics", AW 2nd Ed.

[3]. Rogers, "Procedural Element of Computer Graphics", McGraw Hill.

[4]. Newman and Sproul, "Principal of to Interactive Computer Graphics", McGraw Hill.

