COMPUTER GRAPHICS

Section – I
INTRODUCTION

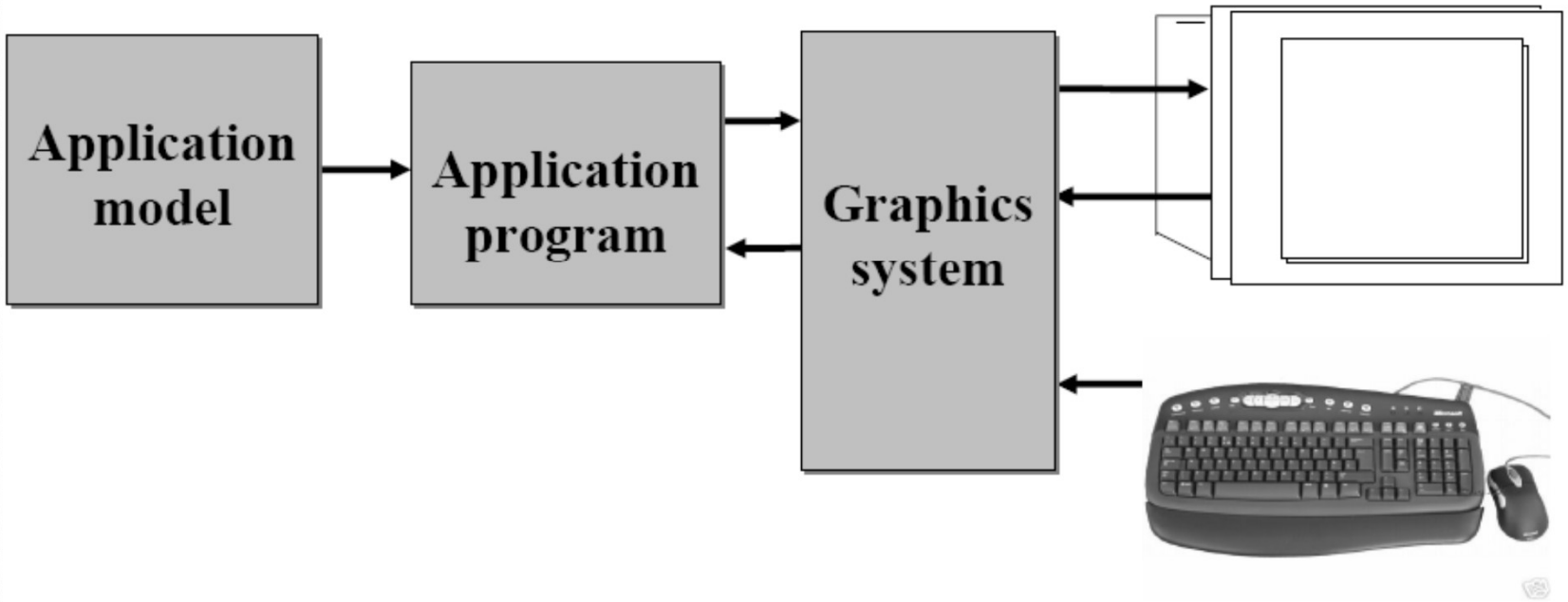
COMPUTER GRAPHICS

- ***Computer Graphics is subfield of computer science which*** **involves display, manipulation and storage of pictures, experimental data and geometric information for proper visualization using computational techniques.**

- **Typical graphics system comprises of a host computer with support of fast processor, large memory, frame buffer and**
  - **Display devices (color monitors),**
  - **Input devices (mouse, keyboard, joystick, touch screen, trackball)**
  - **Output devices (LCD panels, laser printers, color printers. Plotters etc.)**
  - **Interfacing devices such as, video I/O,TV interface etc.**

- **Computer graphics used in diverse areas such as science, engineering, medicine ,business, industry, government, art, entertainment , advertising education and training.**

Conceptual framework for interactive graphics



Graphics Applications

- **Typical applications areas are**
  - Entertainment
  - Computer-aided design
  - Scientific visualization
  - Training & Education
  - Computer art
  - Image Processing
  - GUI

Entertainment

•Animations are frequently used in advertising and television commercials.

•Frame by frame motion is used.

• Commonly used computer graphics method is Morphing

  •Object is transferred into another object

  •Eg: An automobile into a tiger.

•Computer Graphics methods are commonly used in making motion pictures, music videos and television shows

• Graphics objects can combined with live action, actors and live scenes.

Computer-aided design

•Computer Aided Design (CAD) used in design of buildings , automobiles, aircraft, watercraft textiles etc.

•Wireframe models are used.(outline forms)

- •Show overall shape and internal features of objects.
- •Helps to watch the behavior of inner components during motion.
- • Useful for testing the performance of the system or vehicles.

•Animations in Virtual Reality Environment (VR).

- •Determines how vehicle operators affected by certain motions.
- •Helps designer to explore various positions that obstruct operators view.

Los Angeles Airport

Boeing 777 Airplane

Gear Shaft Design

Computer-aided design

Network Communications:

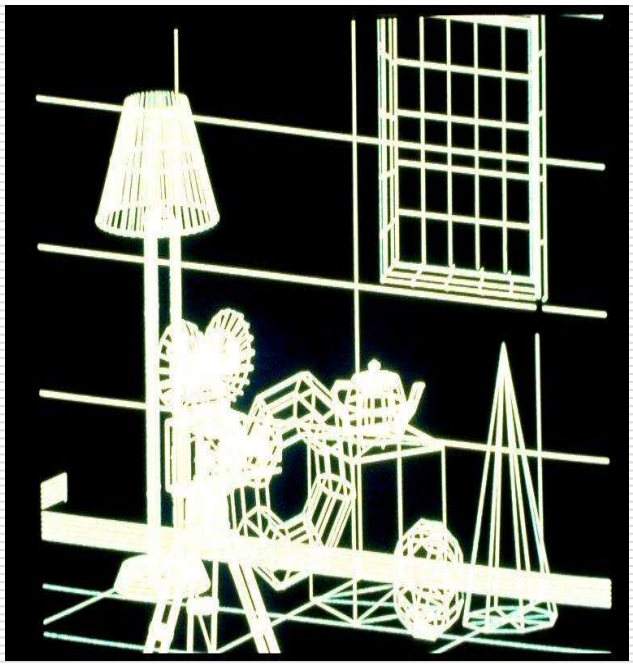
•Designed by placing components into the layout using graphics packages by providing connections.

Architectural Designs

•Architects use interactive graphics to lay floor plans,Positioning of rooms, doors, windows, stairs and other buliding features.

•Electrical Enginer can try out arrangements for wiring , electrical layouts and fire warning systems

•Realistic displays of architectural designs done.



Wireframe models

Scientific visualization

•Scientists,engineers,medical personnel analysts

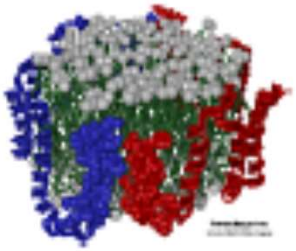
•Satellite cameras producing large data files can be interpreted and converted in the visual form

•Producing graphical representation for scientific engineering and medical datasets and processes is generally referred as **scientific visualization**.

Display of complex structures, complex molecular structures such as proteins and DNA, weather forecasting, complex mathematical objects.

•Techniques such as contour plots,graphs and charts,surface renderings

Airflow Inside a Thunderstorm



Apo A-1

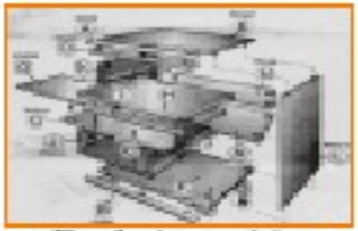
Visible Human

Education & Training

- Computer generated models are used as educational aids.
- Specialized systems like simulators for training of ship captains, aircraft pilots, heavy –equipment operators are used.
- Output of Automobile simulator is used to investigate the behavior of drivers in critical situations and optimizing the vehicle design.

Driving Simulation

Flight Simulation



Desk Assembly

Computer art

- Variety of computer methods including special purpose hardware , specially developed software like mathematics packages(mathematica),CAD packages , desktop publishing packages and animation packages provide facilities for designing object shapes and motions.
- CG methods are also applied in commercial art for logos and other designs.

Image Processing

- Image processing (IP) applies techniques to modify or interpret existing pictures.
  - Improves image quality
  - machine perception of visual information.
- Methods like retouching and rearranging of sections of photographs used to analyze satellite photos of earth and galaxies.
- IP and CG are combined in many medical applications
  - to design artifical limbs
  - to model and study phyiscal functions,
  - Computer aided surgery.

Image Composition

Image Morphing

Graphical User Interface(GUI)

- Software packages provide graphics interface.
- Window manger allows a user to display multiple-window areas.
- Each window contain different graphical or nongraphical displays.
- **Typical Components Used:**
  - **Menus ,Icons , Cursors , Dialog Boxes, Scroll Bars ,Buttons**
  - **Valuators ,Grids, Sketching , 3-D Interface**

Graphics packages and Platforms

- Various application packages and standards are available:
  - Core graphics
  - **GKS** Graphics Kernel System by ISO (International Standards Organization) & ANSI (American National Standards Institute)
  - **SRGP** Simple Raster Graphics Package
  - **PHIGS** Programmers Hierarchical Interactive Graphics System
  - **OpenGL** (with ActiveX and Direct3D)
- On various platforms, such as
  - DOS, Windows,
  - Linux, OS/2,
  - SGI, SunOS,
  - Solaris, HP-UX,
  - Mac, DEC-OSF.

- Thank you

COMPUTER GRAPHICS

Section – II

Computer Graphics Devices

Computer Graphics Devices

- **Examples of Computer Graphics Devices:**
  - Plotters, data matrix, laser printers, Films,
  - Flat panel devices, scanners,
  - LCD panels, keyboard, joystick, mouse,
  - Touch screen, track ball, etc.
- **The most commonly used display device is the**
  - CRT monitor

Video Display Devices

- Primary output of the graphics system is a video monitor.
- The operation of most video monitors is based on standard cathode ray tube design.
- **Types of CRT display devices**
  - **DVST (Direct View Storage Tube)**
  - **Calligraphic or Random Scan display system**
  - **Refresh and Raster scan display system**

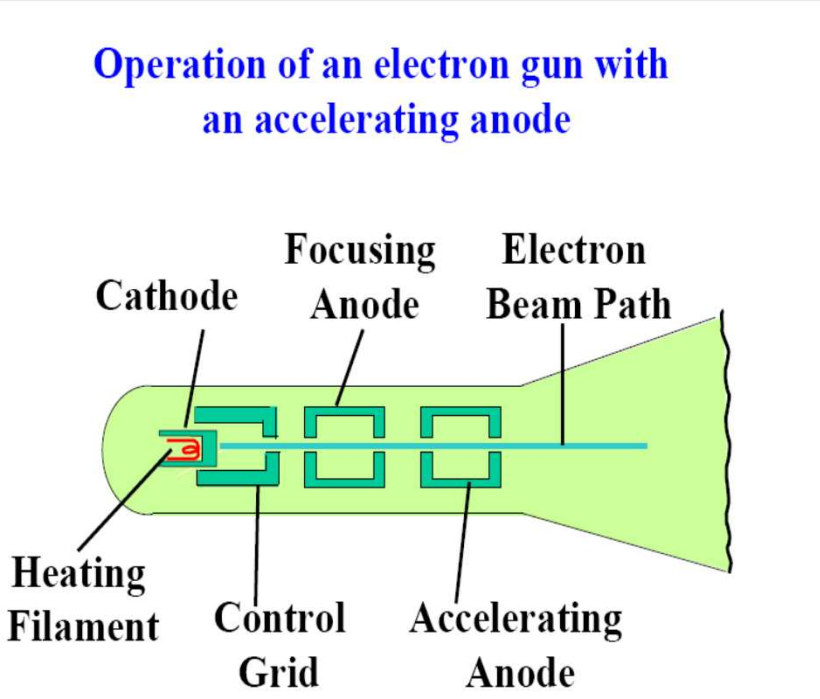
Cathode Ray Tube(CRT)

•Contains a filament , when heated, emits a stream of electrons.

•Electrons are focused with an electromagnet into a sharp beam and directed to a specific point of the face of the picture tube

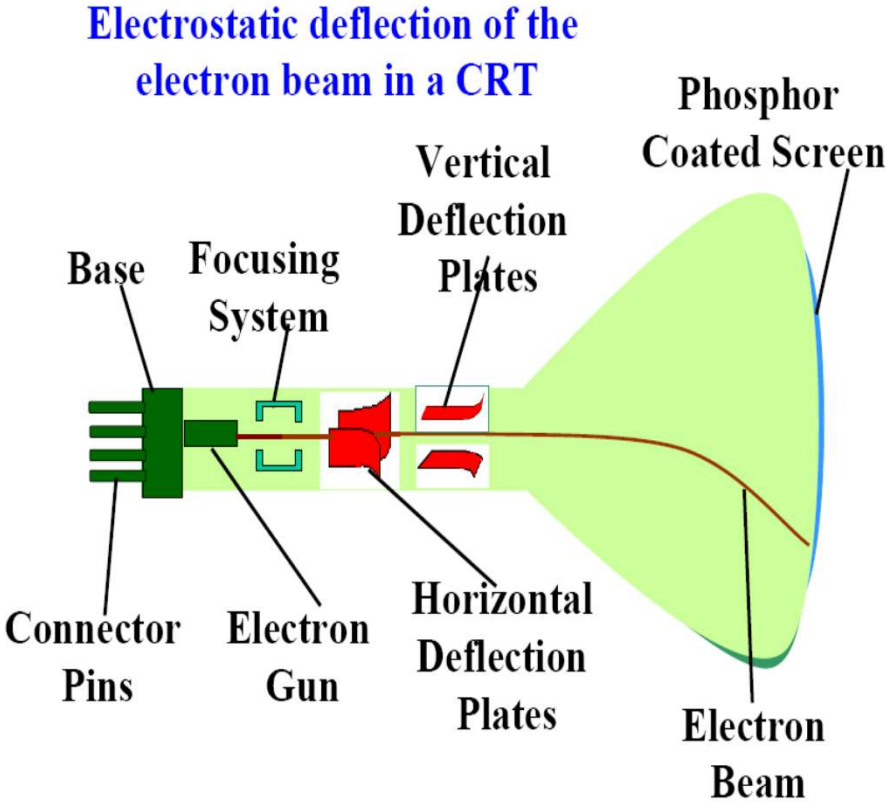
•The front surface of the picture tube is coated with small phosphor dots.

•When the beam hits a phosphor dot it glows with a brightness proportional to the strength of the beam and how long it takes to hit



CRT

- Intensity & Brightness controlled by voltage levels.
- Electron beam has to be focused only at the centre (Focusing system)
- Deflection directs the electron beam horizontally and vertically at any point on the screen
- Deflection controlled by pair of deflection plates.
- CRT beam energy +phosphor=light spot.
- Fading = Excited phosphor drop to ground state after some time.



CRT

- **Persistence**: Time taken by the emitted light to decay to $1/10^{th}$ of its original intensity.
  - Lower persistence requires high refresh rates to maintain a picture without flicker.
  - High persistence phosphor is useful for displaying complex static pictures.
- **Resolution**: The total no of points that can be displayed without overlap on a CRT.
  - Depends on type of phosphor, the intensity to be displayed and focusing and deflection systems.
  - High definition systems has a resolution 1280*1024.
- **Aspect Ratio**: The ratio of horizontal points to vertical points needed to produce equal length lines in both directions on the screen.
- **Refresh Rate:** Rate at which the screen is refreshed.

CRT characteristics

- How can one measure CRT capabilities?
  - Size of tube
  - Brightness of phosphors vs. darkness of tube
  - Speed of electron gun
  - Width of electron beam
  - Pixels

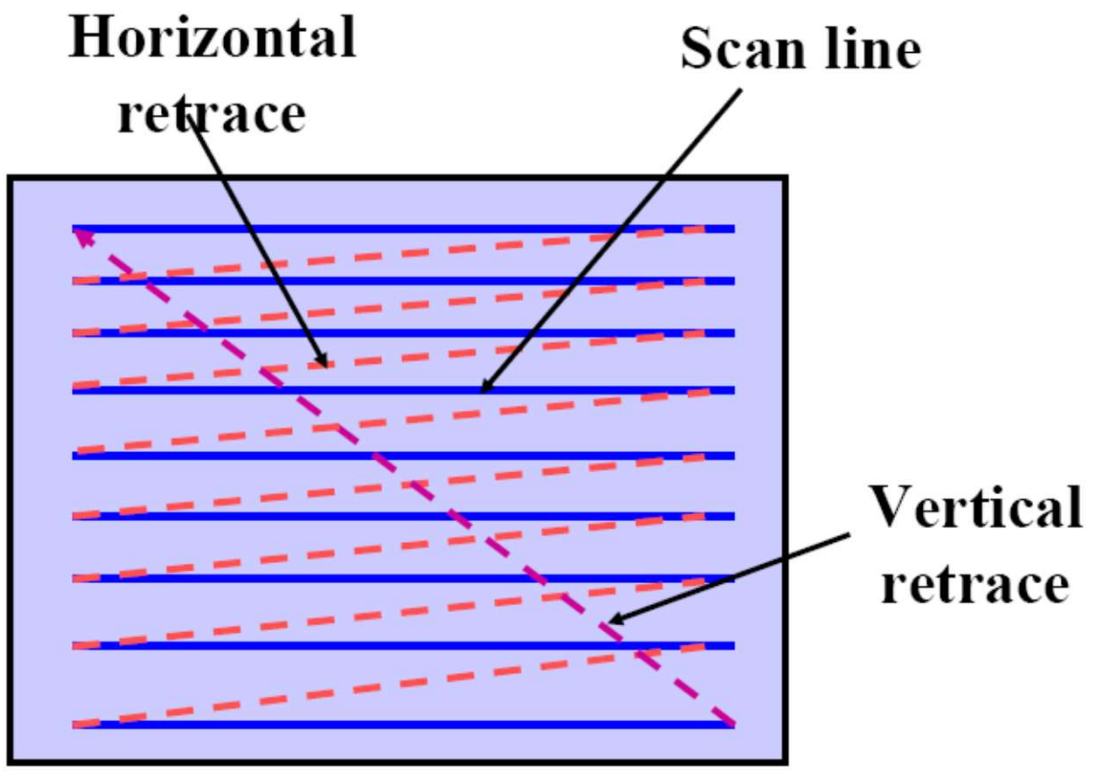
Refresh and Raster scan display system

- Refresh CRT is a point-plotting device.
- The electron beam is swept across the screen, one row at a time from top to bottom.
- Each row is called as scan line.
- As it moves across each scan line, the intensity is turned on and off to create pattern of illuminated spots.
- Raster displays store the display primitives (lines, characters, shaded and patterned areas) in a refresh buffer
- Refresh buffer or Frame buffer stores the drawing primitives in terms of points and pixels components as intensity values.
- Used in television screens and printers.

Refresh and Raster scan display system

- B/W system : 1 bit/pixel – intensity control
  - 1-> beam turned on
  - 0-> beam turned off
  - Frame buffer : Bit map
- Additional bits per pixel for color system.
  - Frame buffer :Pix map
- Needs 60 to 80 frames per second.
- **Horizontal Retrace**: After refreshing each scan line the electron beam return to the left of the screen .
- **Vertical Retrace**: At the end of each frame , beam returns to the top left corner of the screen to begin the next frame.

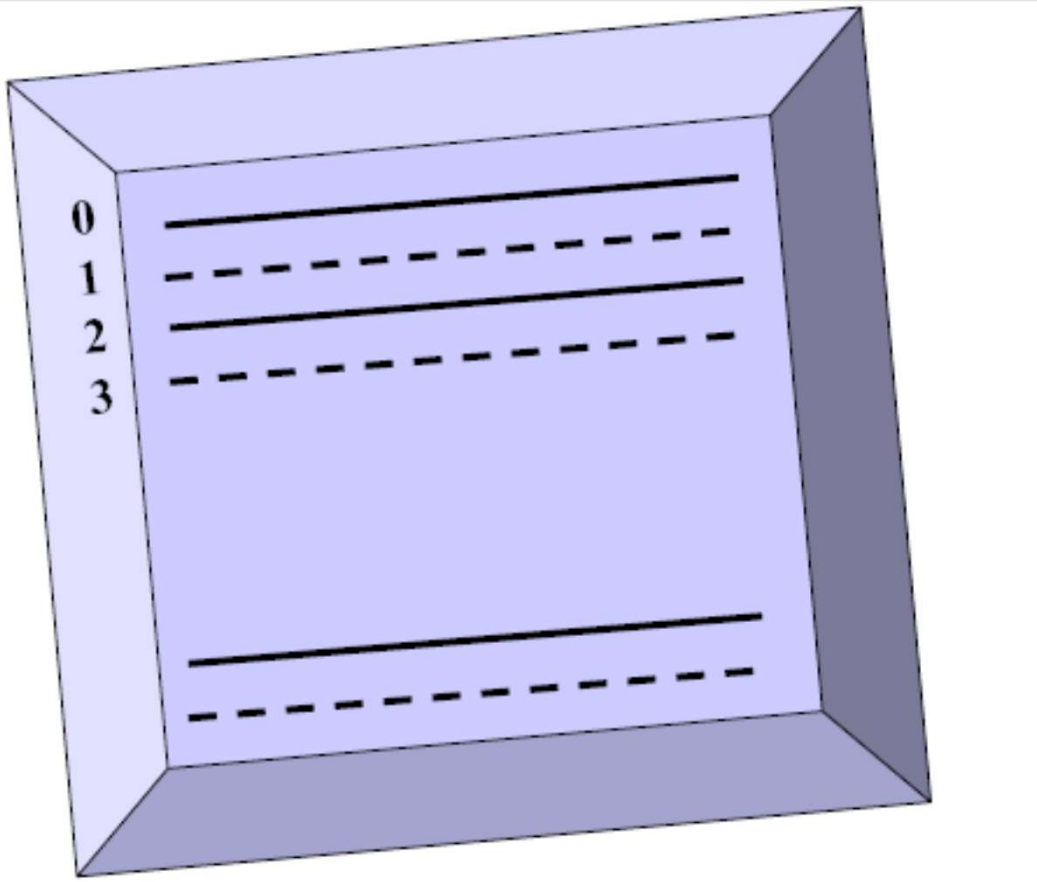
Refresh and Raster scan display system



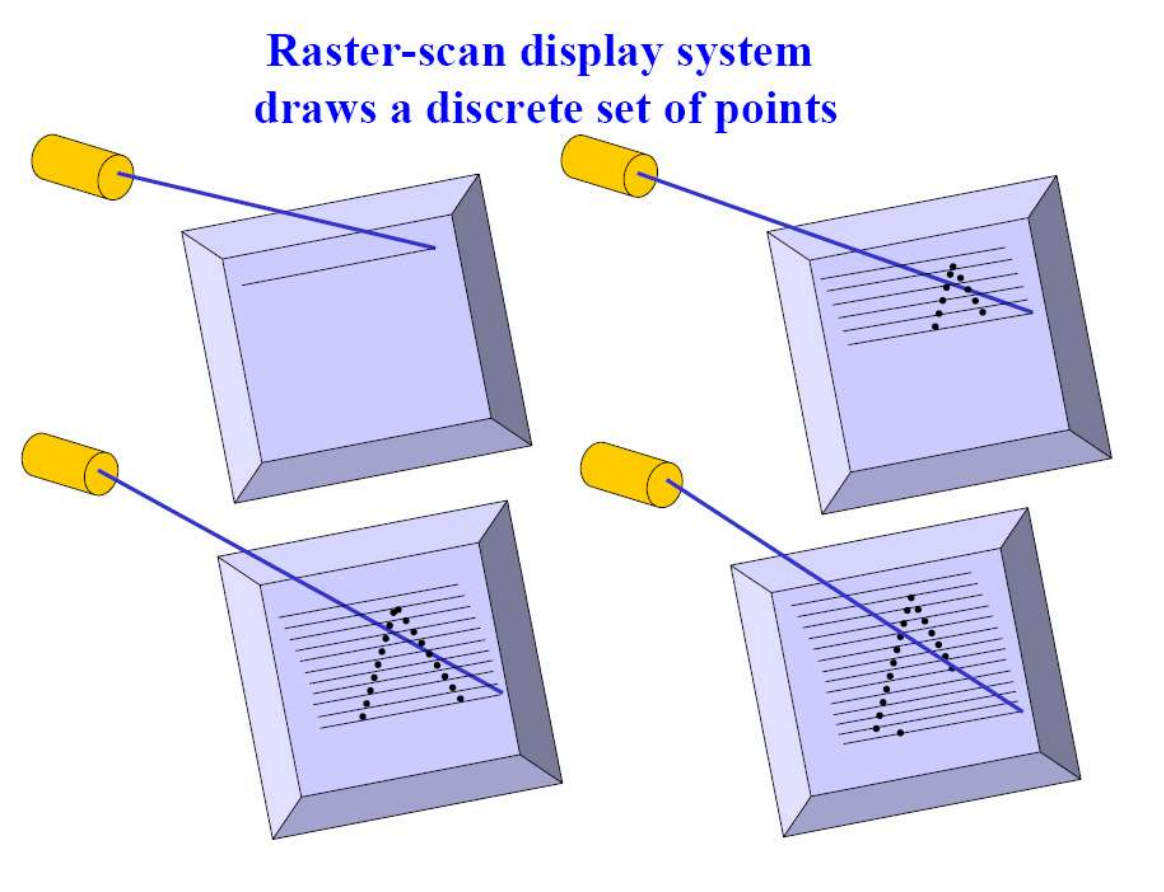
Refresh and Raster scan display system

- **Interlacing**: Allows us to see the entire screen displayed in ½ the time it would have taken to sweep across all the lines at once from top to bottom.
- **Interlaced Refresh procedure:**
  - To reduce flicker, divide frame into two "fields" of odd and even lines.
  - Divides into two passes.
    - 1st pass : Beam sweeps across every even scan lines from top to bottom.
    - 2nd pass : After vertical retrace, beam sweeps remaining odd scan lines.
  - Used with slow refresh rates.

Interlacing



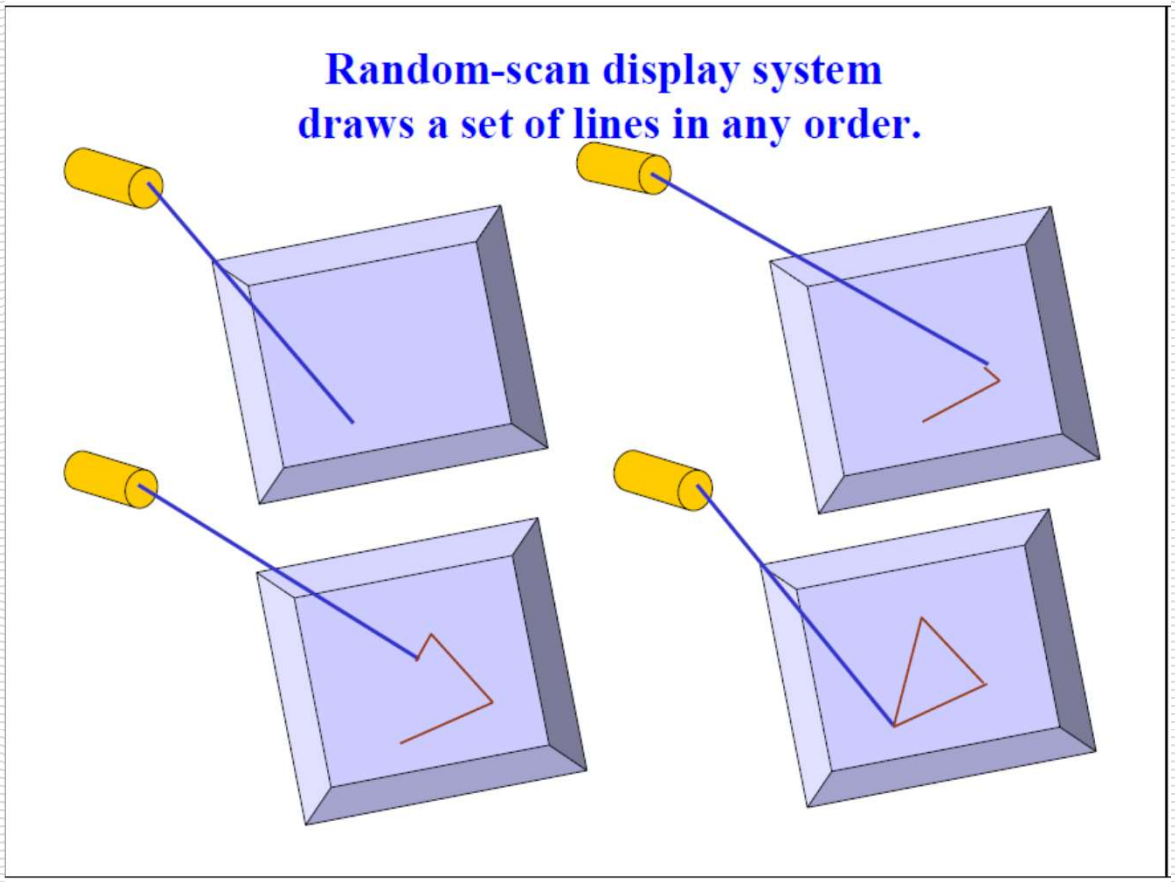
Raster-scan display system draws a discrete set of points



Random Scan display system

- Random scan display unit directs the electron beam to the parts of the screen where the picture is to be drawn.
- Draws a picture one line at a time referred as vector displays.
- Refresh rate depends on number of lines to be displayed.
- Picture definition stored as line drawing commands in refresh display file/refresh buffer.
- To display a picture, system cycles through a set of commands in the display file drawing each component line in a turn.
- Draws lines of a picture in 30 – 60 times/sec.
- Random scan - Order of deflection is dictated by the arbitrary order of the display commands

Random scan Display system



Random scan Systems

- An application program input stored in the system memory along with graphics package.
- Graphics commands in the application program are translated into display file.
- Display file accessed by the display processor to refresh screen.
- Display processor cycles through each command in the display file program.
- Patterns drawn on the screen by directing the electron beam along the component lines of the picture.
- Lines are defined by their coordinate endpoints, these values are converted into deflection voltages.
- Scene is drawn one line at a time by positioning the beam to fill in the line between specified end points.

Raster Vs Random Displays

Raster Display	Random Display
Picture definition stored as discrete point.	Picture definition stored as line drawing commands.
Produces jagged lines plotted as discrete point sets.	Produces smooth lines.
CRTs, Printers, Home TV sets etc.	Pen plotters, Asteroids, CAD/CAM etc.
Decreasing memory costs have made raster systems popular.	Random-scan system's are generally costlier
Refresh time is not dependent on image complexity	Refresh time is dependent on image complexity.

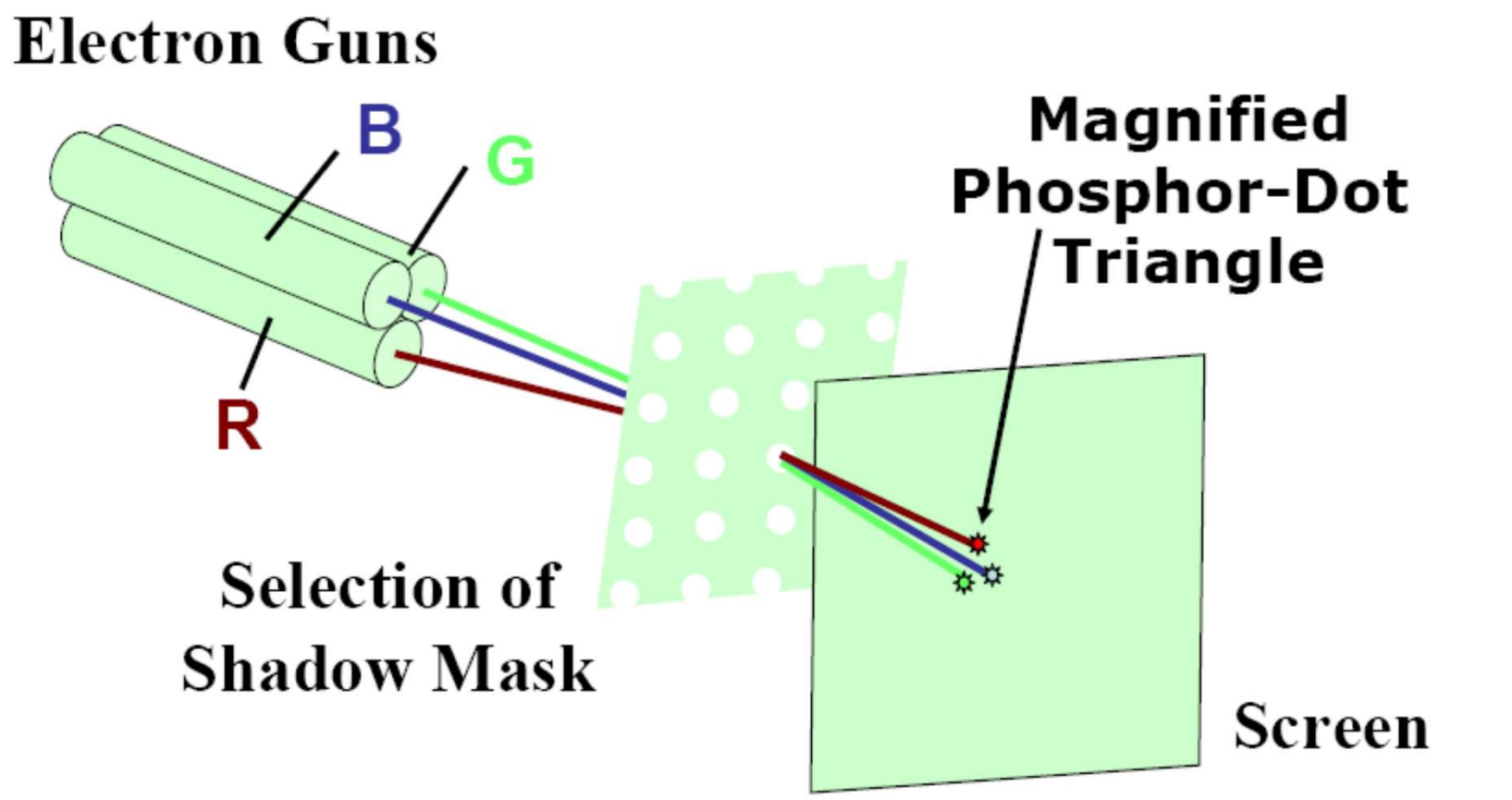
Color CRT Monitors

- Displays color pictures by using combination of phosphors that emit different colored light.
- By combining emitted light a range of colors generated.
- Basic Techniques:
  - Beam penetration
  - Shadow mask method.
- Beam penetration method:
  - Uses random-scan monitors.
  - Red and green phosphor coated inside the CRT screen.
  - Display depends upon how far beam penetrates the phosphor layers.
  - Fast electrons penetrates the red and excites the green
  - Intermediate speed beam produces combinations of red and green => orange,yellow
  - Produces only four colors , quality not good.

Color CRT Monitors
Shadow Methods

- Three electron guns, aligned with the triangular color-dot patterns on the screen.
- Directed to each dot triangle by a shadow mask.
- When beam pass through the hole in the shadow mask they activate a dot triangle (small color spot)
- The phosphor dots in the triangles are arranged so the beam can activate its corresponding color.
- Color variations obtained by turning off the corresponding gun.
- Commonly used in color CRTs.
- Several millions colors can be generated.

Color CRT Monitors
Shadow Methods



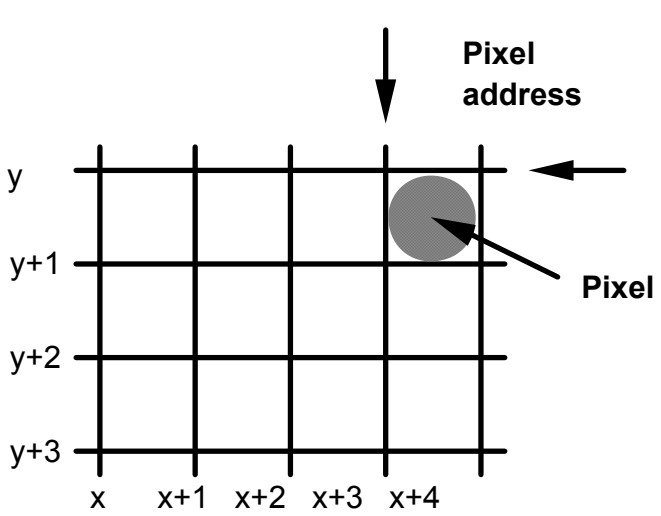
- Thank you

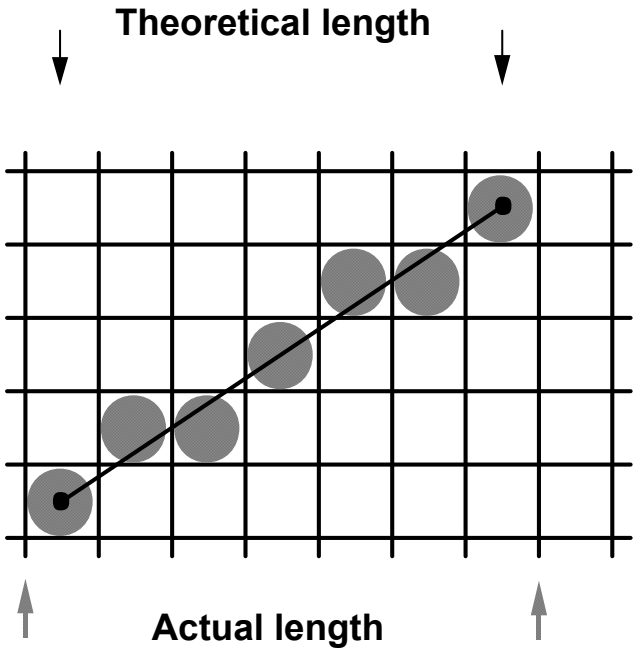
Raster Conversion Algorithms

DDA line algorithm

Bresenham line algorithm

Pixel Addressing in Raster Graphics

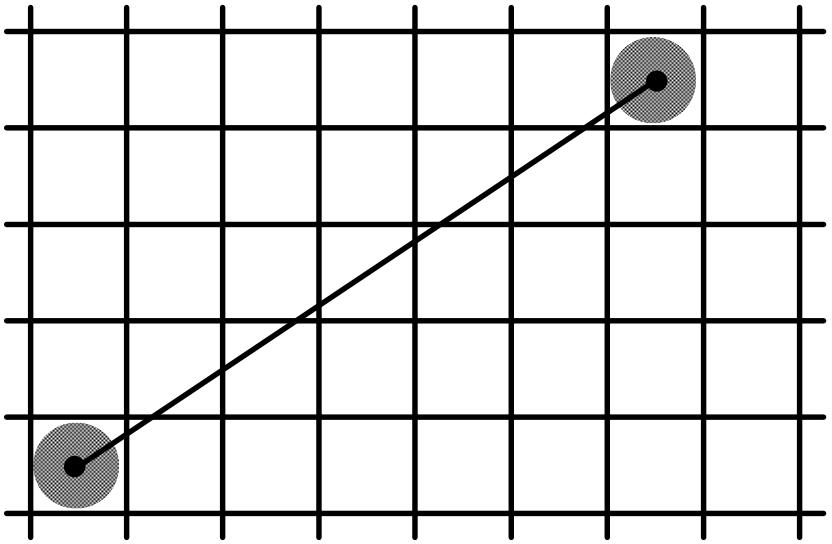




Raster conversion Algorithms: Requirements

- visual accuracy
- spatial accuracy
- speed

Line – Raster Representation



Basis for Line Drawing Algorithms

- The Cartesian slope-intercept for a straight line is
  - **$y = m.x + b$** ----------→ 1
    - **m= Slope of the line**
    - **b= y intercept**
- The slope of the line is defined as

  $m=(y_2-y_1)/(x_2-x_1)$ ----------→ 2
- The y intercept b is defined as

  $b=y1- m . x1$----------→3
- Algorithms for displaying straight lines are based on the above 3 eqns.
- For any given x interval Δx along a line, we compute the y-interval Δy.
  - **$\Delta y=m . \Delta x$** ----------→4
  - Similarly we can obtain Δx .
    - **$\Delta x= \Delta y/m$** ----------→5
  - The eqns 4 and 5 form the basis for determining the deflection voltage in analog devices.

Cases to handle

- Case 1:**Slope |m| < 1**
  - Δx=small horizontal deflection voltage
  - Δy is calculated proportional to slope.(Δy =m. Δx)
- Case 2:**Slope |m| >1**
  - Δy=small vertical deflection voltage
  - Δx is calculated proportional to slope.(Δx =Δy/m)
- Case 3:**Slope m = 1**
  - Δx=Δy (Both the voltages are same)

Line drawing – DDA algorithm

- (DDA) is a scan-conversion line algorithm based on calculating either Δy or Δx.
  - $\Delta y = m * \Delta x$
  - $\Delta x = \Delta y / m$
- we have many cases based on sign of the slope, value of the slope, and the direction of drawing.
  - Slope sign: positive or negative.
  - Slope value: <= 1 or >1.
  - Direction: (left – right) or (right – left)

DDA – case 1

- **Positive slope and left to right:**

- If slope <= 1 then:

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

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

- If slope > 1 then:

  $\mathbf{x_{k+1} = x_k + 1/m}$

  $\mathbf{y_{k+1} = y_k + 1}$

DDA – case 2

- **Positive slope and right to left:**

- If slope <= 1 then:

  $\mathbf{x_{k+1} = x_k - 1}$

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

- If slope > 1 then:

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

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

DDA – case 3

- **Negative slope and left to right:**

- If |m | <= 1 then:

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

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

- If |m | > 1 then:

  $\mathbf{x_{k+1} = x_k + 1/m}$

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

DDA – case 4

- **Negative slope and right to left:**

- If |m | <= 1 then:

  $\mathbf{x_{k+1} = x_k - 1}$

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

- If |m | > 1 then:

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

  $\mathbf{y_{k+1} = y_k + 1}$

Digital Differential Algorithm

- Input line endpoints, (x_1,y_1) and (x_2, y_2)
- set pixel at position (x_1,y_1)
- calculate slope $m=(y_2-y_1)/(x_2-x_1)$
- **For +ve slope (left to right)**
  - **Case $|m|\leq 1$**: Sample at unit x intervals and compute each successive y.
  - Repeat the following steps until (x_2, y_2) is reached:

    $\mathbf{y_{k+1} = y_k + m}$ where$(m=\Delta y/ \Delta x)$

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

    set pixel at position $\mathbf{(x_{k+1},Round(y_{k+1}))}$
  - **Case $|m|>1$**: Sample at unit y intervals and compute each successive x.
  - Repeat the following steps until (x_2, y_2) is reached:

    $\mathbf{x_{k+1} = x_k + 1/m}$

    $\mathbf{y_{k+1} = y_k + 1}$

    set pixel at position $\mathbf{(Round(x_{k+1}), y_{k+1})}$

Digital Differential Algorithm

- **For +ve slope (right endpoint to left endpoint)**
  - **Case $|m| \leq 1$**: Sample at unit x intervals and compute each successive y.
  - Repeat the following steps until (x_2, y_2) is reached:

    $\mathbf{y_{k+1} = y_k - m}$ where$(m = \Delta y / \Delta x)$

    $\mathbf{x_{k+1} = x_k - 1}$

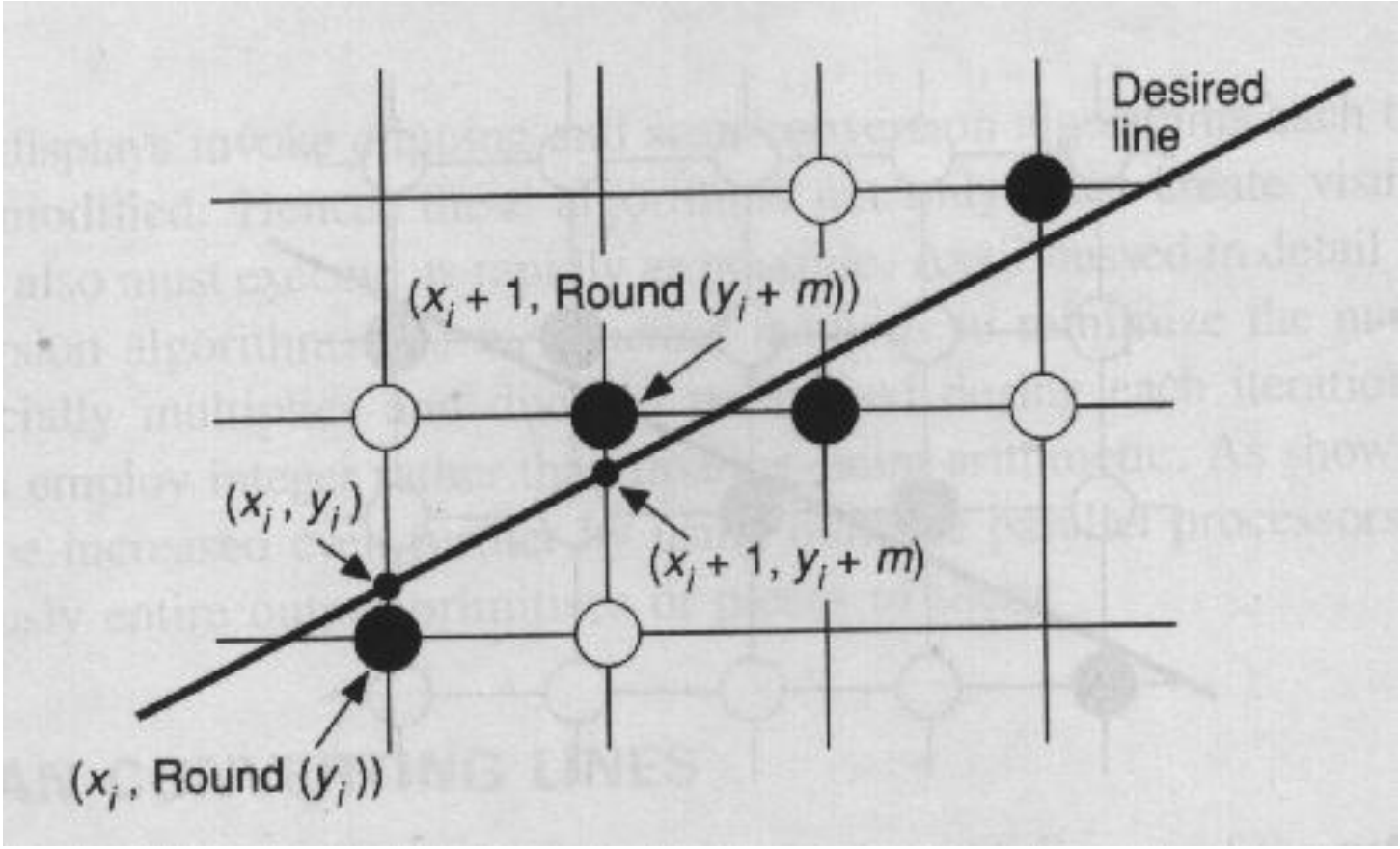
    set pixel at position $(x_{k-1}, \text{Round}(y_{k-1}))$
  - **Case $|m| > 1$**: Sample at unit y intervals and compute each successive x.
  - Repeat the following steps until (x_2, y_2) is reached:

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

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

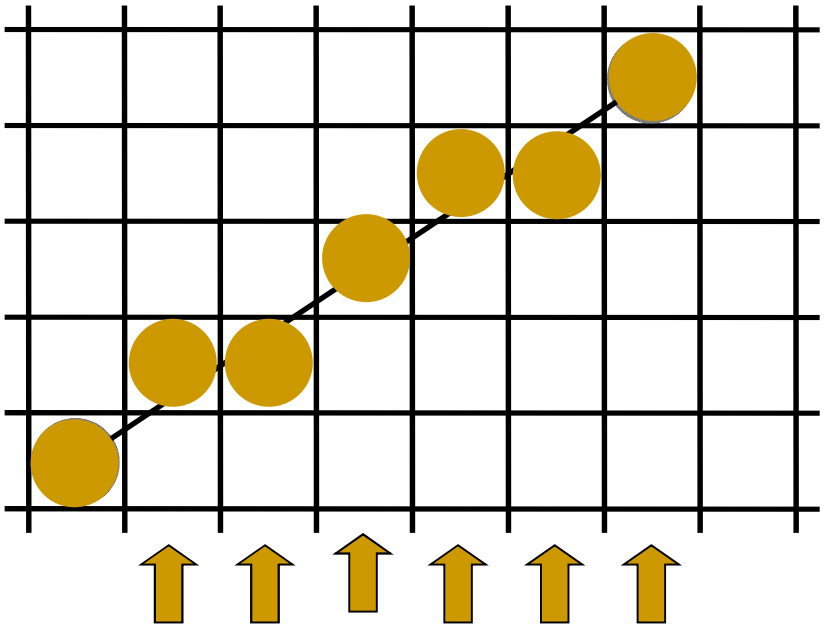
    set pixel at position $(\text{Round}(x_{k-1}), y_{k-1})$

Scan Conversion Process



DDA (Digital Differential Algorithm)

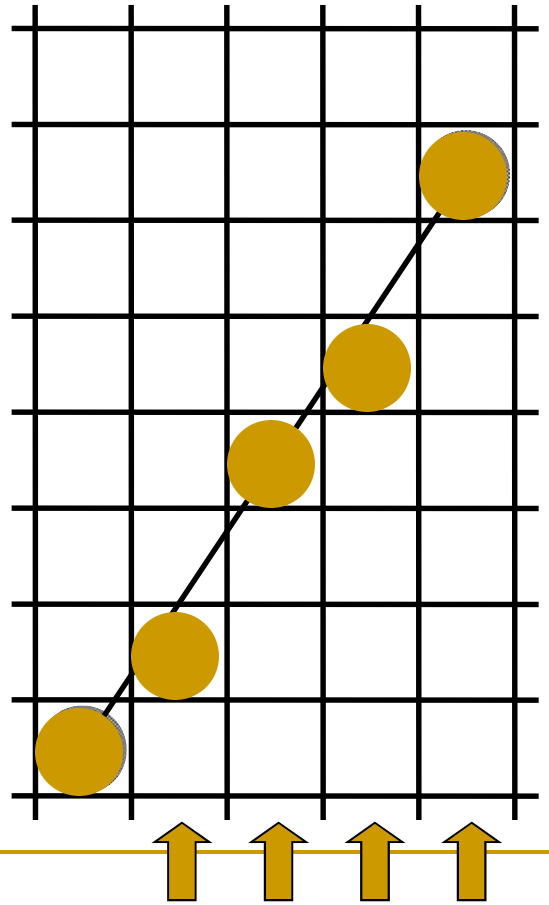
m < 1



- Δx=small horizontal deflection voltage (Δx=1)
- Δy proportional to slope m (Δy= Δx . m)

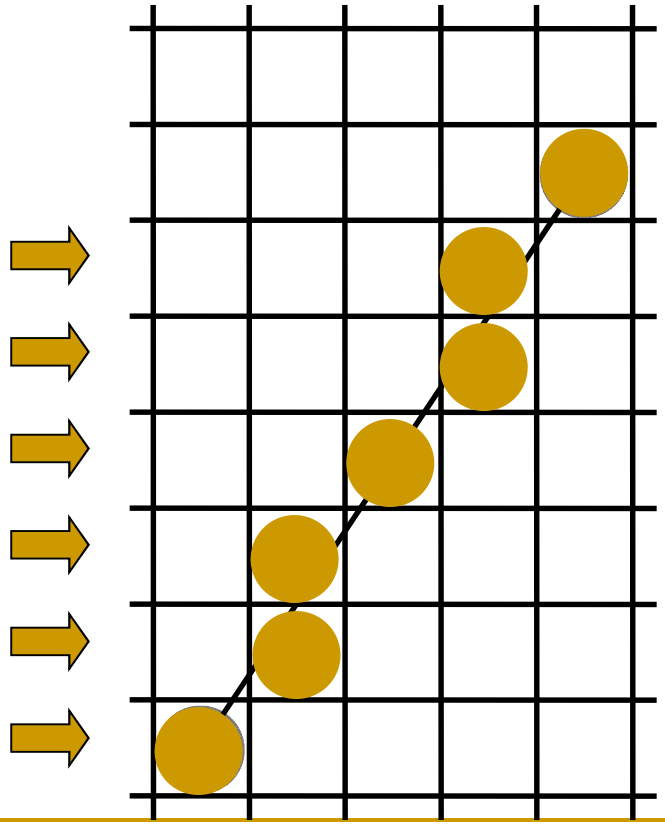
DDA (Digital Differential Algorithm)

For m < 1 , If we sample along x



DDA (Digital Differential Algorithm)

For m > 1, we sample along y



- Δy=small horizontal deflection voltage (Δy=1)
- Δx proportional to slope m (Δx= Δy/ m)

DDA Line algorithm - Procedure

- Procedure lineDDA(xa,xb,ya,yb:integer);
- Var
  - dx,dy,steps,k : integer;
  - xIncrement, yIncrement, x , y: real;
  - Begin
    - dx:=xb-xa;
    - dy:=yb-ya;
    - if abs(dx) >abs(dy) then steps:=abs(dx)
    - else steps:=abs(dy)
    - xIncrement :=dx/steps;
    - yIncrement :=dy/steps;
    - x:=xa;
    - y:=ya;

DDA Line algorithm – Procedure contd.

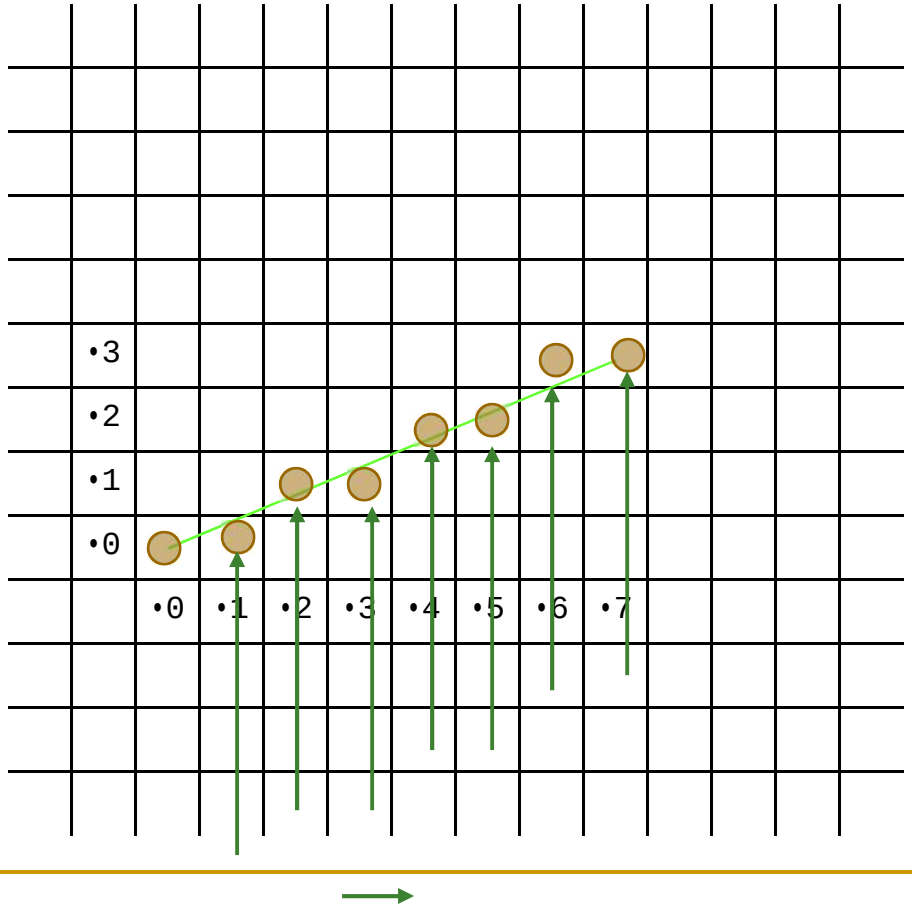
- setPixel(round(x),round(y),1);
- for k:=1 to steps do
- Begin
  - x:=x+xIncrement;
  - y:=y+yIncrement;
  - setPixel(round(x), round(y),1);

  End

End {lineDDA}

Example

- **Consider endpoints:** P1 (0,0) P2 (7,3)
- m = (3 – 0)/(7 - 0)
  = 0.429 (m<1)(+ve slope)
- dx = 1
- $x_0 = 0, y_0 = 0$
- $x_1 = x_0 + 1 = 1$
- $y_1 = y_0 + 0.429$
  $= 0.429 \approx 0$
- $x_1 = 1, y_1 = 0.429$
- $x_2 = x_1 + 1 = 2$
- $y_2 = y_1 + 0.429$
  $= 0.859 \approx 1$



Example contd.

- **$x_2 = 2, y_2 = 0.858$**
- $x_3 = x_2 + 1 = 3$
- $y_3 = y_2 + 0.429$
  $= 1.287 \approx 1$
- **$x_3 = 3, y_3 = 1.287$**
- $x_4 = x_3 + 1 = 4$
- $y_4 = y_3 + 0.429$
  $= 1.716 \approx 2$
- **$x_4 = 4, y_4 = 2$**
- $x_5 = x_4 + 1 = 5$
- $y_5 = y_4 + 0.429$
  $= 2.145 \approx 2$

Example contd.

- **$x_5 = 5, y_5 = 2$**
- $x_6 = x_5 + 1 = 6$
- $y_6 = y_5 + 0.429$

  $= 2.574 \approx 3$
- **$x_6 = 6, y_6 = 3$**
- $x_7 = x_6 + 1 = 7$
- $y_7 = y_6 + 0.429$

  $= 3.003 \approx 3$

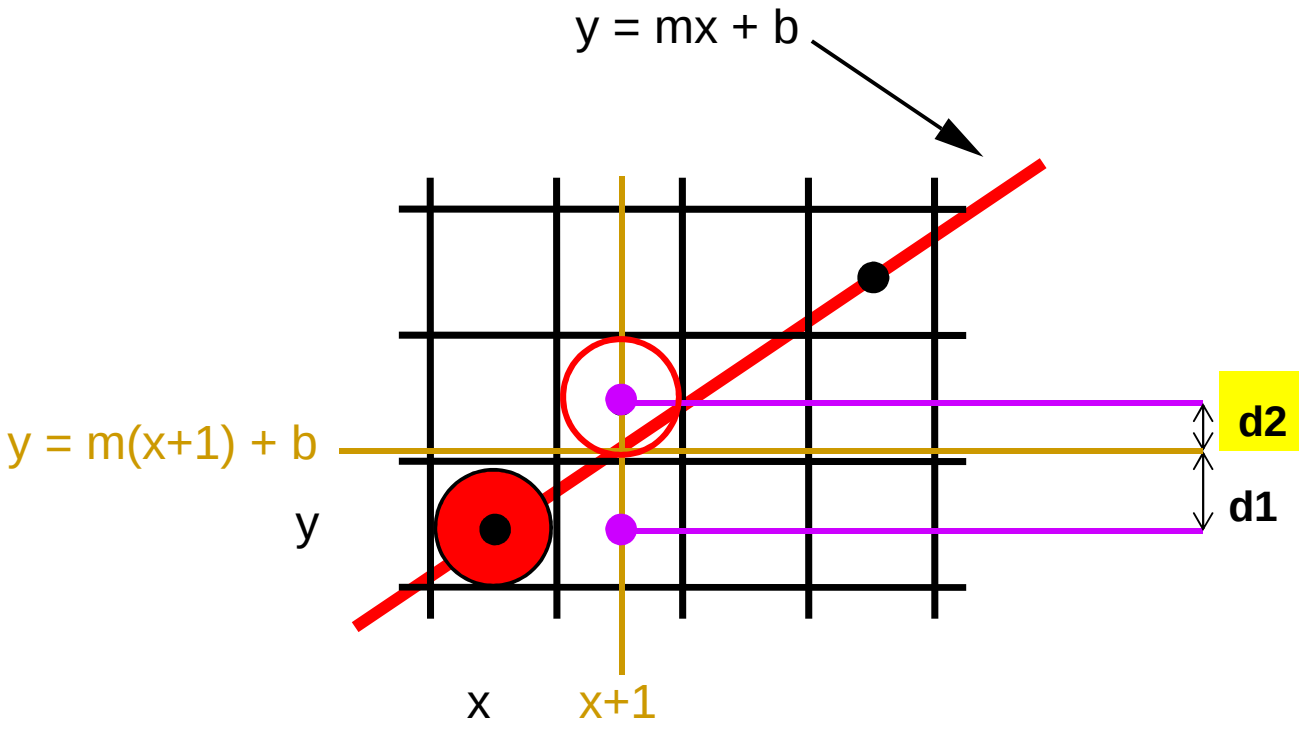
$x_7=7, y_7=3$

Disadvantages of DDA algorithm

- DDA works with floating point arithmetic
- Rounding to integers necessary

Bresenham's Line Algorithm

- **An accurate and efficient raster line generating-algorithms developed by Bresenham.**
- **Scan converts lines using only incremental integer calculations.**



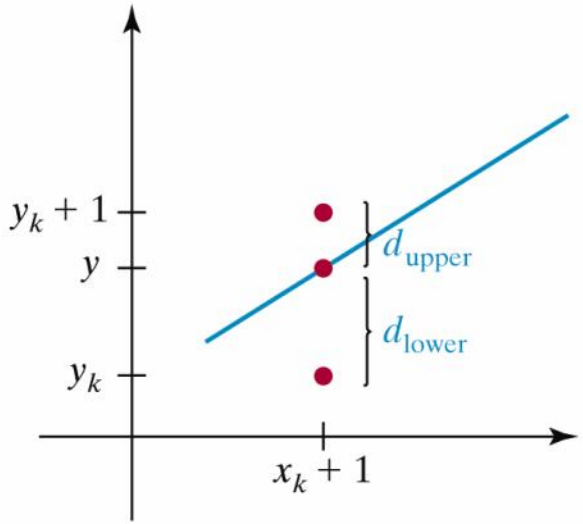


Figure 3-11

Vertical distances between pixel positions and the line y coordinate at sampling position $x_k + 1$.

Bresenham's line algorithm (slope ≤ 1)

- Input line endpoints, (x_0,y_0) and (x_n, y_n)
- calculate $\Delta x = x_n - x_0$ and $\Delta y = y_n - y_0$
- Assuming the pixel (x_k , y_k) is to displayed
  - Determine the positions whether at **(x_k+1 , y_k) and (x_k+1 , y_k+1)**
- From x_k+1 we label vertical pixel separations from line as d1 and d2
- The y coordinate at pixel column position x_k+1 is calculated as
  - **$y=m(x_k+1)+b.$**
  - **Then**
    - **$d1=y-y_k=m(x_k+1)+b-y_k$**

and

- **$d2=y_k+1-y=y_k+1-$** $m(x_k+1)-b.$

Bresenham's line algorithm (slope ≤ 1)

- d1-d2=2 **$m(x_k+1)-2y_k+2b-1$**
- Decision parameter p_k for the kth step in line algorithm obtained by rearranging the above equations.
- Substitute m=Δy/ Δx, where Δy, Δx are horizontal and vertical seperations.
  - $P_{k=}$ Δx(d1-d2)== **$2\,\Delta y.\, x_k - 2\,\Delta x\,.\, y_k + c$----------→1**
  - C is constant has the value 2 Δy+ Δx(2b-1)
  - When P_k is negative plot the lower pixel else plot the upper pixel.
- Coordinate changes along the line occur in unit steps in x and y directions.
- The values of successive decision parameter can be evaluated from
  - $P_{k+1=}$**$2\,\Delta y.\, x_{k+1} - 2\,\Delta x\,.\, y_{k+1} + c$----------→2**
- Subtracting 1 & 2 we get
  - $P_{k+1}=P_k+$ **$2\,\Delta y- 2\,\Delta x\,(y_{k+1}- y_k)$ (since $x_{k+1=}x_k+1$)**
  - The term $(y_{k+1}- y_k)$ is either 1 or 0 depending on the sign of parameter p_k
  - The recursive calculation of decision parameters is performed at each integer x position.
- The parameter p_0 is evaluated from eq1 $p_0=2\,\Delta y- \Delta x$

Bresenham's Line Algorithm

- Input line endpoints, (x_0,y_0) and (x_n, y_n)
- Load (x_0,y_0) into the frame buffer that is first point
- Calculate the constants Δx, Δy,$2\Delta y$ and $2\Delta y - 2\Delta x$
- calculate parameter $p_0 = 2\ \Delta y - \Delta x$
- Set pixel at position (x_0,y_0)
- repeat the following steps until (x_n, y_n) is reached:
- if $p_k < 0$
- set the next pixel at position $(x_k +1, y_k)$
- calculate new $p_{k+1} = p_k + 2\ \Delta y$
- if $p_k \geq 0$
- set the next pixel at position $(x_k +1, y_k + 1)$
- calculate new $pk_{+1} = p_k + 2(\Delta y - \Delta x)$
- Repeat last step Δx times.

Advantages of Bresenham's Line Algorithm

- Bresenham’s algorithm uses integer arithmetic
- Constants need to be computed only once
- Bresenham’s algorithm generally faster than DDA

Bresenham's line drawing Procedure

- procedure lineBres (xa, ya, xb, yb : integer)
- var
  - dx, dy, x, y, xEnd, p: integer;
- begin
  - dx :=abs(xa-xb);
  - dy :=abs(ya-yb);
  - p:=2 * dy – dx;
  - if xa > xb then
    - begin
      - x:= xb;
      - y:= yb;
      - xEnd := xa;
  - else
  - begin
    - x:=xa;
    - y:=ya;

Contd…

- xEnd := xb;
- end;
- setPixel (x,y,1);
- while x < xEnd do
- begin
  - x:= x+1;
  - if p < 0 then p:= p+2 * dy
  - else

    begin

        y:=y+1;

    p:=p+2 * (dy-dx)

    end;

    setPixel (x,y,1);

    end

    end; {lineBres}

Bresenham's Line Algorithm (Example)

- Note: Bresenham's algorithm is used when slope is <= 1.

- using Bresenham's Line-Drawing Algorithm, Digitize the line with endpoints (20,10) and (30,18).

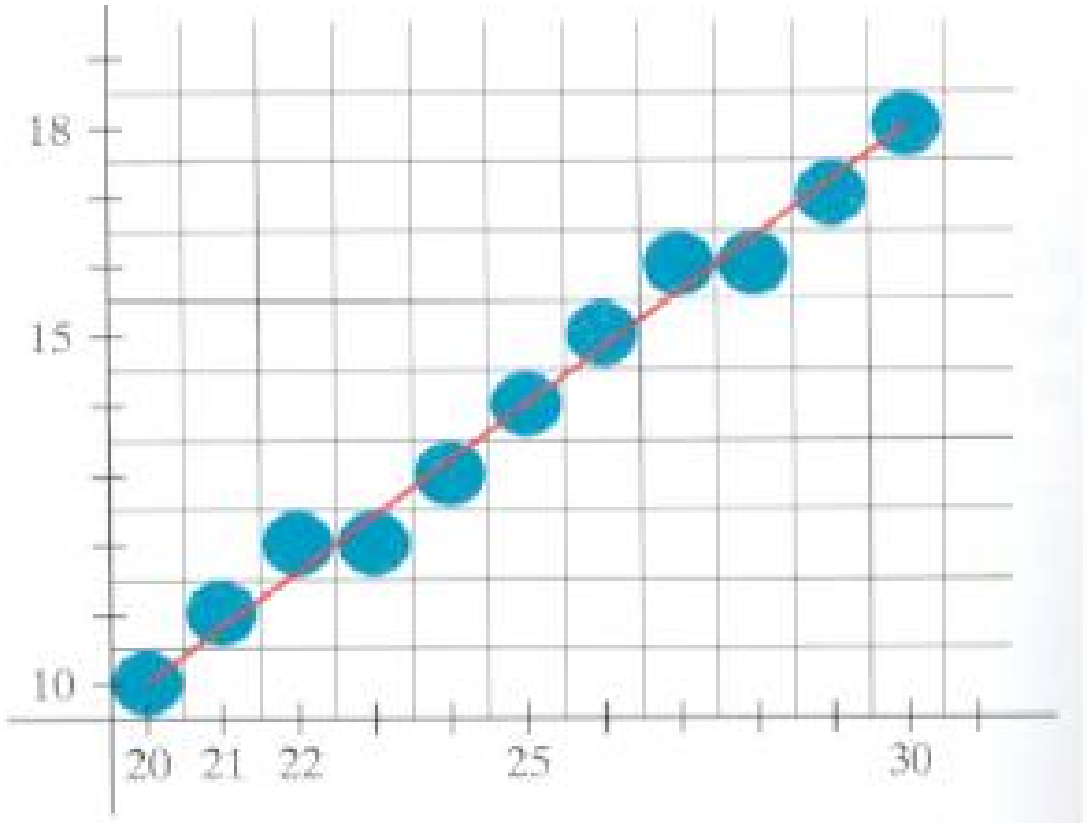
- $\Delta y = 18 - 10 = 8$
- $\Delta x = 30 - 20 = 10$
- $m = \Delta y / \Delta x = 0.8$

- plot the first point (x0, y0) = (20, 10)
- $p0 = 2 * \Delta y - \Delta x = 2 * 8 - 10 = 6$, so the next point is (21, 11)

Example (cont.)

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)

Example (cont.)



- Thank you

Circle Generating Algorithms

- Circle is a frequently used components in pictures and graphs , a procedure for generating either full circles or circular arcs is included in most graphics packages.

- Circle is a set of points that are all at a given distance r from center position (x_c, y_c).

- The distance relationship equation of a circle is expressed by the Pythagorean theorem in Cartesian coordinates as:

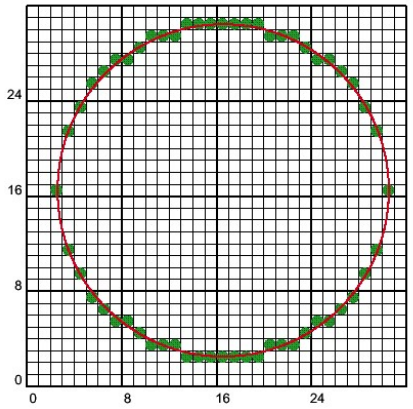
$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

Circle Generating Algorithms

- We can re-write the circle equation as:

$$y = y_c \pm (r^2 - (x - x_c)^2)^{0.5}$$

- By substitution with x , x_c and y_c we can get y.
- Two problems with this approach:
  - it involves considerable computation at each step.
  - The spacing between plotted pixel positions is not uniform.



Circle Generating Algorithms

- Polar coordinates **(r and θ)** are used to eliminate the unequal spacing shown above.
- Expressing the circle equation in parametric polar form yields the pair of equations
  - **$x = xc + r \cos \theta$**
  - **$y = yc + r \sin \theta$**
- When a circle is generated with these equations using a fixed angular step size, a circle is plotted with equally spaced points along the circumference.
- The step size chosen θ depends on the application and the display device.
- Larger angular separations along the circumference can be connected with straight lines.

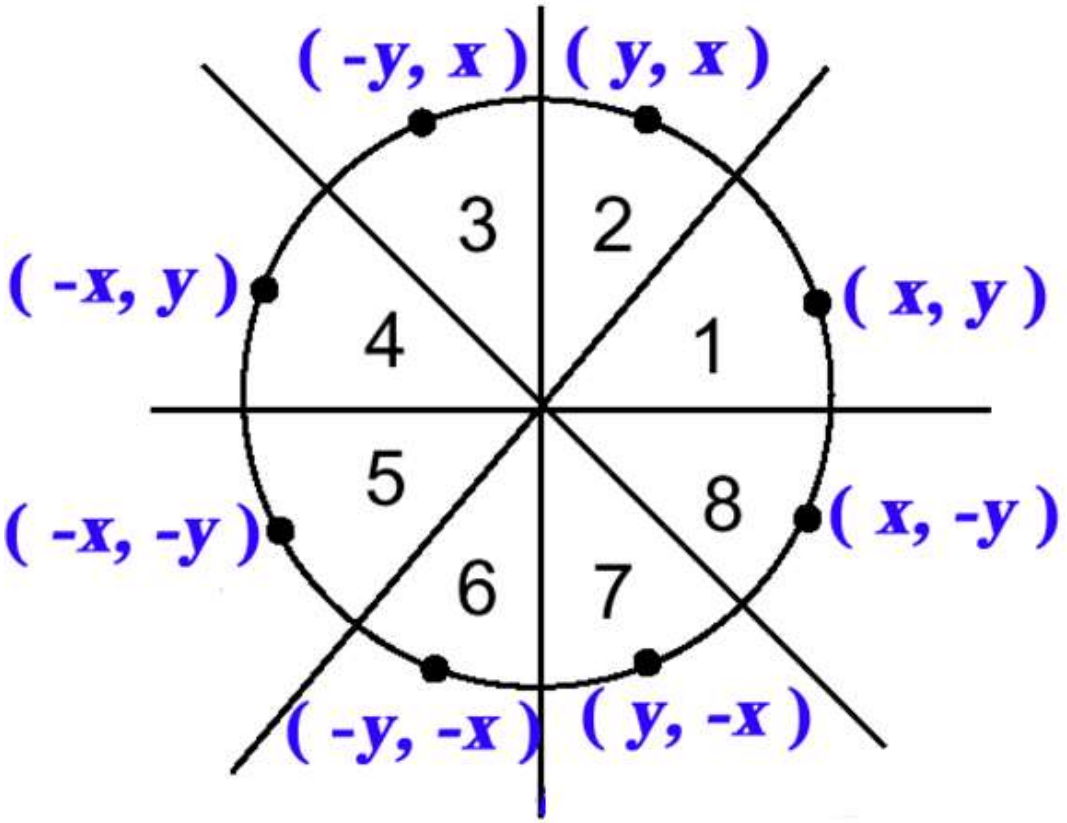
Circle Generating Algorithms

- The Cartesian equation involves multiplications and square root calculations.
- Parametric equations contain multiplications and trigonometric calculations.
- Efficient circle algorithms are based on incremental calculations of decision parameters which involves only integer calculations.

Midpoint Circle Algorithm (BASICS)

- Computation can be reduced by considering the symmetry of circles. The shape of the circle is similar in each quadrant.
- There is also symmetry between octants.
- Adjacent octant within one quadrant are symmetric with 45º line dividing the two octants.
- We can generate all pixel positions around a circle by calculating only the points within the sector from x=0 to x=y

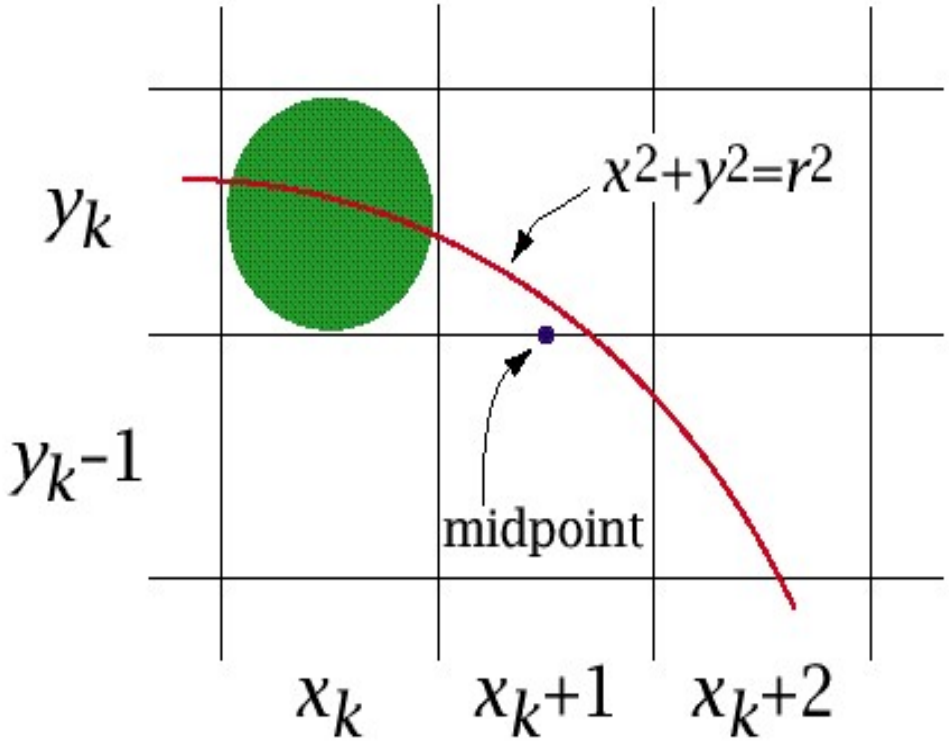
Midpoint Circle Algorithm (BASICS)



Midpoint Circle Algorithm

- A method for direct distance comparison is to test the halfway position between two pixels to determine if this midpoint is inside or outside the circle boundary.
- This method is more easily applied to other conics, and for an integer circle radius.
- we sample x at unit intervals and determine the closest pixel position to the specified circle path at each step.

Midpoint Circle Algorithm



Midpoint Circle Algorithm

- For a given radius r and screen center position (x_c, y_c), we can first set up our algorithm to **calculate pixel positions** around a circle path centered at the coordinate origin (0, 0).

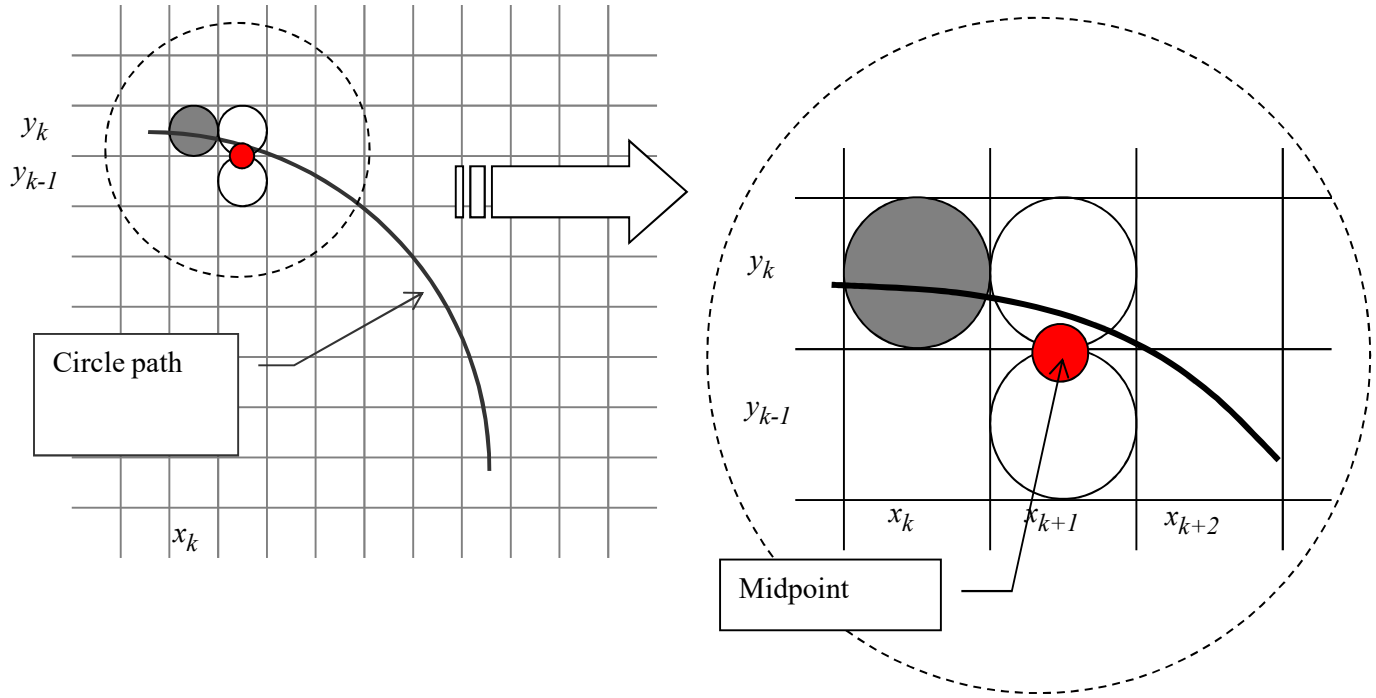
- Then each calculated position (x, y) is moved to its proper screen position by adding x_c to x and y_c to y.

- Along the circle section from x = 0 to x = y in the first quadrant, the slope of the curve varies from 0 to - 1.

- Therefore, we can take unit steps in the positive x direction over this octant and use a decision parameter to determine which of the two possible y positions is closer to the circle path at each step.

Mid-point Circle Algorithm

- Consider current position (xk, yk)
- Next point position is (xk+1, yk) or (xk+1, yk-1)?



Mid-point Circle Algorithm

- Our decision parameter is the earlier circle function evaluated at the mid point between the 2 pixels

- p_k < 0: midpoint is inside the circle; plot (xk+1, yk)
- p_k +ve: midpoint is outside the circle; plot (xk+1, yk-1)

- Successive decision parameters are obtained using incremental calculation

Midpoint Circle Algorithm

- Positions in the other seven octants are then obtained by symmetry.
- To apply the midpoint method. we define a circle function:

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

- Any point (x, y) on the boundary of the circle with radius r satisfies the equation $f_{circle}(x, y) = 0$.
- If $f_{circle}(x, y) < 0$, the point is inside the circle boundary ,
  If $f_{circle}(x, y) > 0$, the point is outside the circle boundary,
  If $f_{circle}(x, y) = 0$, the point is on the circle boundary.

Mid-point Circle Algorithm - Calculating p_k

- First, set the pixel at (x_k, y_k), next determine whether the pixel $(x_k + 1, y_k)$ or the pixel $(x_k + 1, y_k - 1)$ is closer to the circle using:

$$p_k = fcircle\,(x_k + 1, y_k - \tfrac{1}{2}) = (x_k + 1)^2 + (y_k - \tfrac{1}{2})^2 - r^2$$

- *Successive decision parameters are obtained using incremental calculations.*
  - $P_{k+1} = fcircle\,(x_{k+1} + 1, y_{k+1} - \tfrac{1}{2}) = [(x_k + 1) + 1]^2 + (y_{k+1} - \tfrac{1}{2})^2 - r^2$
  - $P_{k+1} = pk + 2(x_k + 1) + (y^2_{k+1} - y^2_k) - (y_{k+1} - y_k) + 1$

- *If pk<0 this midpoint is inside the circle select* y_k
  - $P_{k+1} = p_k + 2(x_k + 1) + 1$ **or** $p_{k+1} = p_k + 2x_{k+1} + 1$,

else mid position is outside or on the circle boundary select $y_k - 1$

- $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$

  where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$. Depending upon the sign of p_k.

Mid-point Circle Algorithm - Calculating p_0

- The initial decision parameter is obtained by evaluating the circle function at the start position $(x_0,y_0)=(0,r)$
  - **$p_0 = fcircle\ (1, r - ½) = 1 + (r - ½)^2 - r^2$**
  - **$p_0 = 5/4 - r$**

- **If the radius r is specified as an integer, simply round p_0 to**
  - **$P_{0=}\ 1 - r$**

Midpoint Circle Algorithm

1. Input radius **r** and circle center (x_c, y_c). set the first point $(x_0, y_0) = (0, r)$.

2. Calculate the initial value of the decision parameter as $p_0 = 1 - r$.

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

   If $p_k < 0$,

   plot $(x_k + 1, y_k)$ and $p_{k+1} = p_k + 2x_{k+1} + 1$,

   Else,

   plot $(x_k + 1, y_k - 1)$ and $p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$,

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

Midpoint Circle Algorithm

4. Determine symmetry points on 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 coordinate values: $\boldsymbol{x = x + x_c}$, $\boldsymbol{y = y + y_c}$

6. Repeat steps 3 though 5 until $\boldsymbol{x \geq y}$.

7. For all points, add the center point (x_c, y_c)

Midpoint Circle Algorithm

- Now we drew a part from circle, to draw a complete circle, we must plot the other points.
- We have $(x_c + x , y_c + y)$, the other points are:
  - $(x_c - x , y_c + y)$
  - $(x_c + x , y_c - y)$
  - $(x_c - x , y_c - y)$
  - $(x_c + y , y_c + x)$
  - $(x_c - y , y_c + x)$
  - $(x_c + y , y_c - x)$
  - $(x_c - y , y_c - x)$

Midpoint Circle Algorithm

- Given a circle radius r = 10, demonstrate the midpoint circle algorithm by determining positions along the circle octant in the first quadrant from x = 0 to x = y.

Solution:

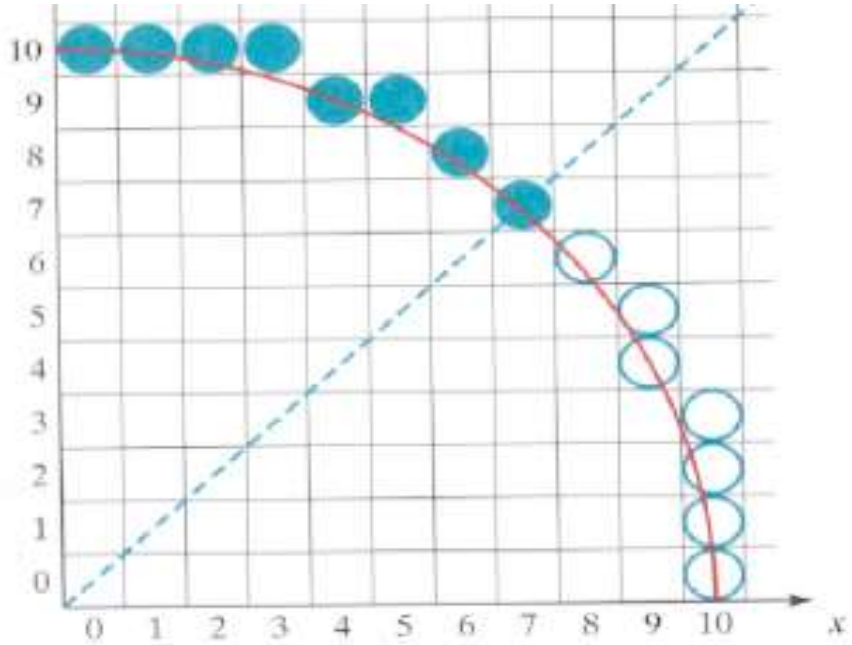
- $p_0 = 1 - r = -9$
- Plot the initial point $(x_0, y_0) = (0, 10)$,

- $2x_0 = 0$ and $2y_0 = 20$.
- Successive decision parameter values and positions along the circle path are calculated using the midpoint method as appear in the next table:

Midpoint Circle Algorithm

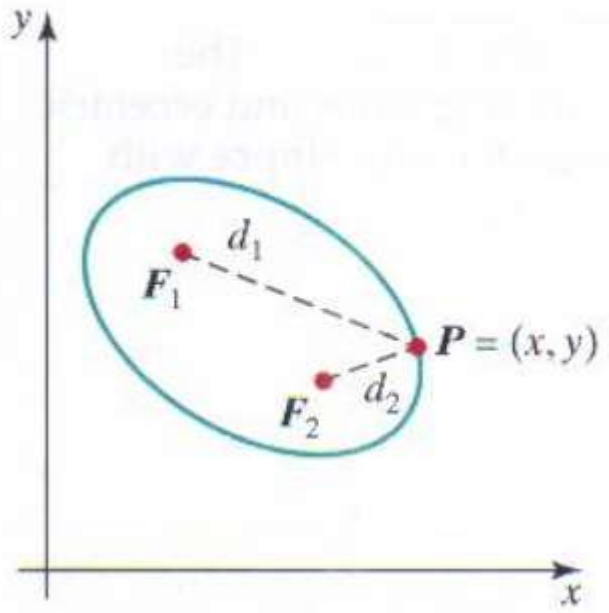
K	P_k	(x_{k+1}, y_{k+1})	$2\,x_{k+1}$	$2\,y_{k+1}$
0	– 9	(1, 10)	2	20
1	– 6	(2, 10)	4	20
2	– 1	(3, 10)	6	20
3	6	(4, 9)	8	18
4	– 3	(5, 9)	10	18
5	8	(6,8)	12	16
6	5	(7,7)	14	14

Midpoint Circle Algorithm



Ellipse Generating Algorithms

- Ellipse – an elongated circle.
- A modified circle whose radius varies from a maximum value in one direction to a minimum value in the perpendicular direction.
- A precise definition in terms of distance from any point on the ellipse to two fixed position, called the foci of the ellipse.
- The sum of these two distances is the same value for all points on the ellipse.



Ellipse Generating Algorithms

- If the distance to the two foci from any point P=(x,y) on the ellipse is labeled as d1 and d2 then the general equation
  - $d_1 + d_2 = constant$
- Expressing the distances in terms of the focal coordinates F1=(x1,y1) and F2=(x2,y2) we have

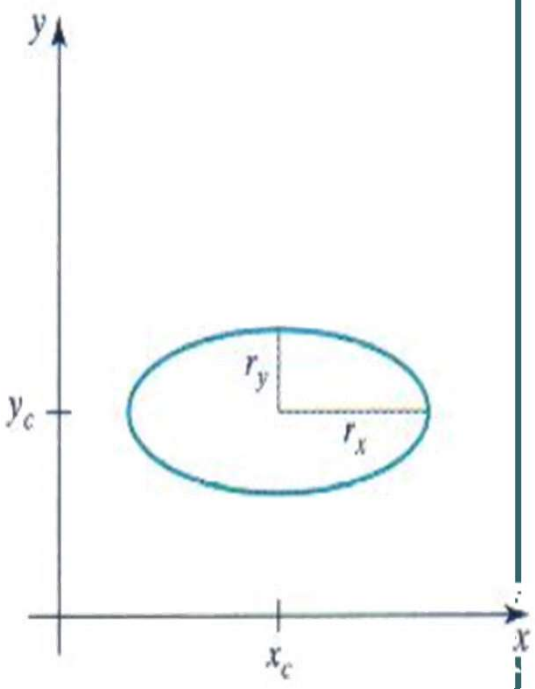
$$\sqrt{(x-x_1)^2+(y-y_1)^2}+\sqrt{(x-x_2)^2+(y-y_2)^2}=\text{consta}$$

- Ellipse has two axes major and minor axes.
- Major axes is a straight line segment extending from one side of the ellipse to the other side through foci

Ellipse Generating Algorithms

- Minor axis spans the shorter dimensions of the ellipse bisecting the major axis at the halfway position between the two foci.
- we will only consider 'standard' ellipse in terms of the ellipse center coordinates and parameters r_x and r_y

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



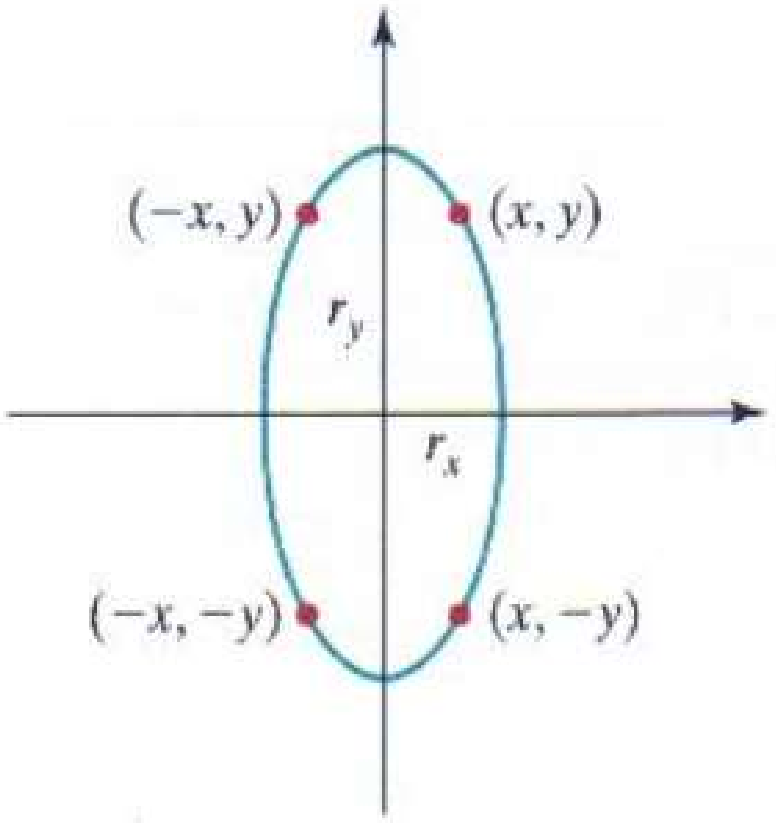
Ellipse Generating Algorithms

- An ellipse only has a 2-way symmetry
- Calculation of a point (x,y) in one quadra yields the ellipse points shown for the other three quadrants
- Consider an ellipse centered at the origin

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

- What is the discriminator function?

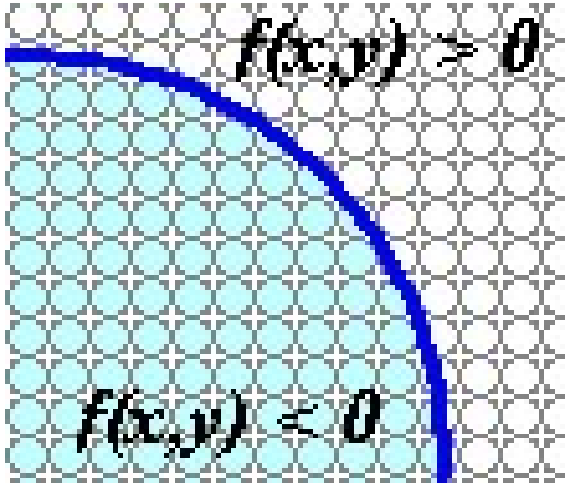
$$f_e(x,y) = r_y^{\,2}x^2 + r_x^{\,2}y^2 - r_x^{\,2}r_y^{\,2}$$



Midpoint Ellipse Algorithms

- We define the Ellipse function as

$$f_e(x,y) = r_y^{\,2}x^2 + r_x^{\,2}y^2 - r_x^{\,2}r_y^{\,2}$$



- It has the following properties:

  $f_e(x,y) < 0$ for a point inside the ellipse

  $f_e(x,y) > 0$ for a point outside the ellipse

  $f_e(x,y) = 0$ for a point on the ellipse

- The ellipse function $f_e(x,y)$ serves as the decision parameter in the midpoint algorithm..
- At each sampling position select the next pixel along the ellipse path according to the sign of the ellipse function.

Ellipse Generating Algorithms

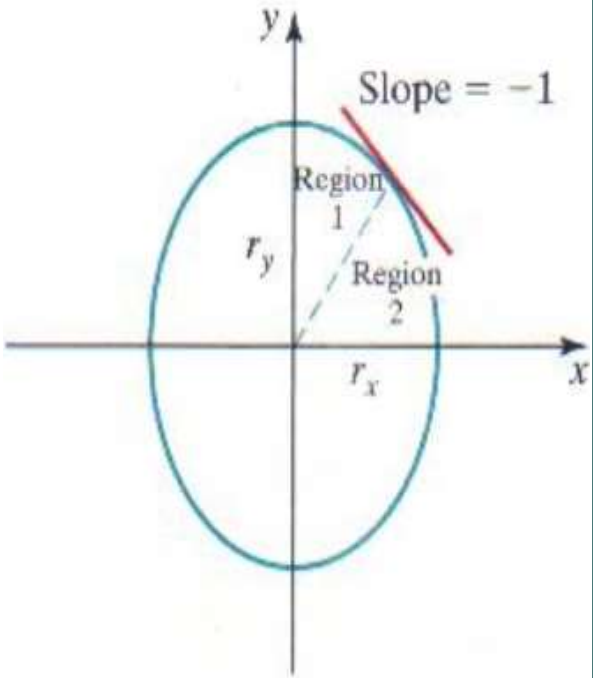
•Ellipse is different from circle.
•Similar approach with circle, different is sampling direction.
•The midpoint ellipse method is applied throughout the first quadrant in two parts.
•The division of the first quadrant according to the slope of an ellipse with $r_x < r_y$

Region 1:

- •Sampling is at *x* direction
- •Choose between (x_{k+1}, y_k), or (x_{k+1}, y_{k-1})
- •Move out if 2r2yx >= 2r2xy

Region 2:

- •Sampling is at *y* direction
- •Choose between (x_k, y_{k-1}), or (x_{k+1}, y_{k-1})



Midpoint Ellipse Algorithms (Decision parameters)

- Region 1:

$$p1_k = f_e\left(x_k + 1, y_k - \tfrac{1}{2}\right)$$

–ve	• midpoint is inside • choose pixel (x_k+1, y_k)
+ve	• midpoint is outside • choose pixel (x_k+1, y_k-1)

Midpoint Ellipse Algorithms

Region 2

$$p2_k = f_e\left(x_k + \tfrac{1}{2}, y_k - 1\right)$$

–ve	• midpoint is inside • choose pixel (x_k+1, y_k-1)
+ve	• midpoint is outside • choose pixel (x_k, y_k-1)

Midpoint Ellipse Algorithms

1. Input r_x, r_y and ellipse center (x_c, y_c). First point on the similar ellipse centered at the origin is $(0, r_y)$.

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

2. Initial value for decision parameter at **region 1**:

$$p1_0 = f_{ellipse}(1, r_y - 1/2)$$

$$= r^2_y - r^2_x (r_y - 1/2)^2 - r^2_x r^2_y$$

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

Midpoint Ellipse Algorithms

3. At each x_k in region 1, starting from k = 0, test $p1_k$:
   If $p1_k < 0$, next point (x_{k+1}, y_k) and

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

   else, next point (x_k+1, y_k-1) and

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

With $2r^2_y x_k + 1 = 2r^2_y x_k + 2r^2_y$, $2r^2_x y_{k+1} = 2r^2_x y_k - 2r^2_x$

- Determine symmetry points in the other 3 octants.
- Get the actual point for ellipse centered at (x_c, y_c) that is $(x + x_c, y + y_c)$.

Midpoint Ellipse Algorithms

4. Repeat step 3 - 6 until $2r_y^2x \geq 2r_x^2y$.

5. Initial value for decision parameter in **region 2**:

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

6. At each yk in region 2, starting from k = 0, test p2k:If p2k > 0, next point is (xk, yk-1) and

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

Midpoint Ellipse Algorithms

- else, next point is (x_k+1, y_k-1) and

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

9. Determine symmetry points in the other 3 octants.
10. Get the actual point for ellipse centered at (x_c, y_c) that is $(x + x_c, y + y_c)$.
11. Repeat step 8 - 10 until $y = 0$.

Midpoint Ellipse Algorithms

```
inline int round (const float a)  { return int (a + 0.5); }

  /*  The following procedure accepts values for an ellipse
   *  center position and its semimajor and semiminor axes, then
   *  calculates ellipse positions using the midpoint algorithm.
   */
  void ellipseMidpoint (int xCenter, int yCenter, int Rx, int Ry)
  {
    int Rx2 = Rx * Rx;
    int Ry2 = Ry * Ry;
    int twoRx2 = 2 * Rx2;
    int twoRy2 = 2 * Ry2;
    int p;
    int x = 0;
    int y = Ry;
    int px = 0;
    int py = twoRx2 * y;
    void ellipsePlotPoints (int, int, int, int);
```

Midpoint Ellipse Algorithms

```
/* Region 1 */
    p = round (Ry2 - (Rx2 * Ry) + (0.25 * Rx2));
    while (px < py) {
      x++;
      px += twoRy2;
      if (p < 0)
        p += Ry2 + px;
      else {
        y--;
        py -= twoRx2;
        p += Ry2 + px - py;
      }
      ellipsePlotPoints (xCenter, yCenter, x, y);
    }
```

Midpoint Ellipse Algorithms

```
/* Region 2 */
    p = round (Ry2 * (x+0.5) * (x+0.5) + Rx2 * (y-1) * (y-1) - Rx2 * Ry2);
    while (y > 0) {
      y--;
      py -= twoRx2;
      if (p > 0)
        p += Rx2 - py;
      else {
        x++;
        px += twoRy2;
        p += Rx2 - py + px;
      }
      ellipsePlotPoints (xCenter, yCenter, x, y);
    }
  }
```

Midpoint Ellipse Algorithms

```
void ellipsePlotPoints (int xCenter, int yCenter, int x, int y);
  {
    setPixel (xCenter + x, yCenter + y);
    setPixel (xCenter - x, yCenter + y);
    setPixel (xCenter + x, yCenter - y);
    setPixel (xCenter - x, yCenter - y);
  }
```

- Thank you

PIXEL ADDRESSING AND OBJECT GEOMETRY

Pixel addressing and object geometry

- When an object is scan converted into the frame buffer, the input description is transformed to pixel coordinates.
- So, the displayed image may not correspond exactly with the relative dimensions of the input object.
- To preserve the specified geometry of world objects, we need to compensate for the mapping of mathematical input points to finite pixel area, we use one of the two ways:

Pixel addressing and object geometry (cont.)

1) Adjust the dimensions of displayed objects to account for the amount of overlap of pixel areas with the object boundaries.

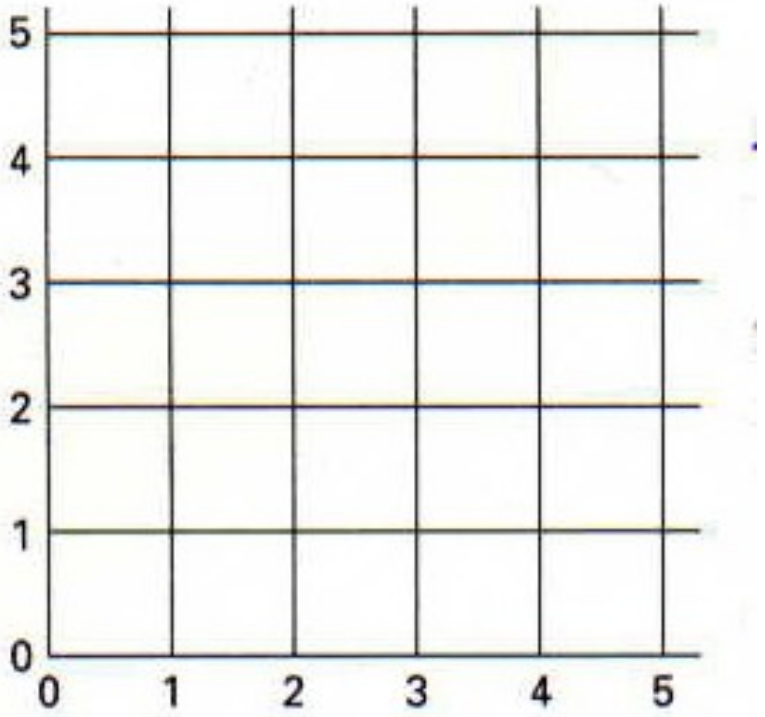
(i.e. a rectangle with 40 cm width, will be displayed in 40 pixel)

2) Map world coordinates onto screen positions between pixels, so that we align objects boundaries with pixel boundaries instead of pixel centers._

Pixel addressing and object geometry (cont.)

Screen Grid Coordinates:

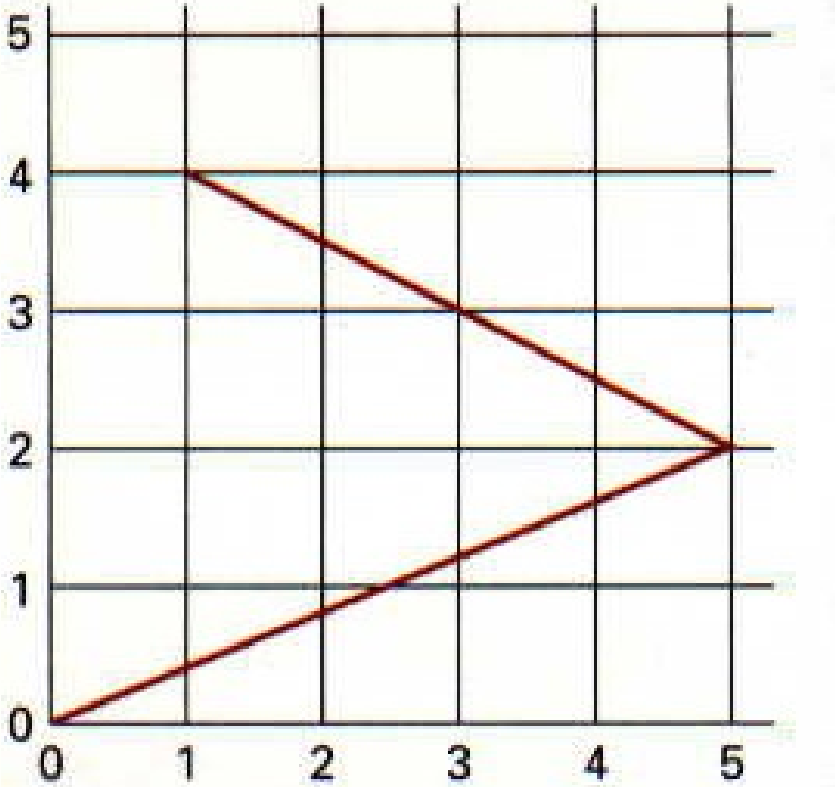
An alternative to addressing display positions in terms of pixel centers is to reference screen coordinates with respect to the grid of horizontal and vertical pixel boundary lines spaced one unit a part.



Pixel addressing and object geometry (cont.)

- Screen coordinate position is then the pair of integer values identifying a grid intersection position between two pixels.

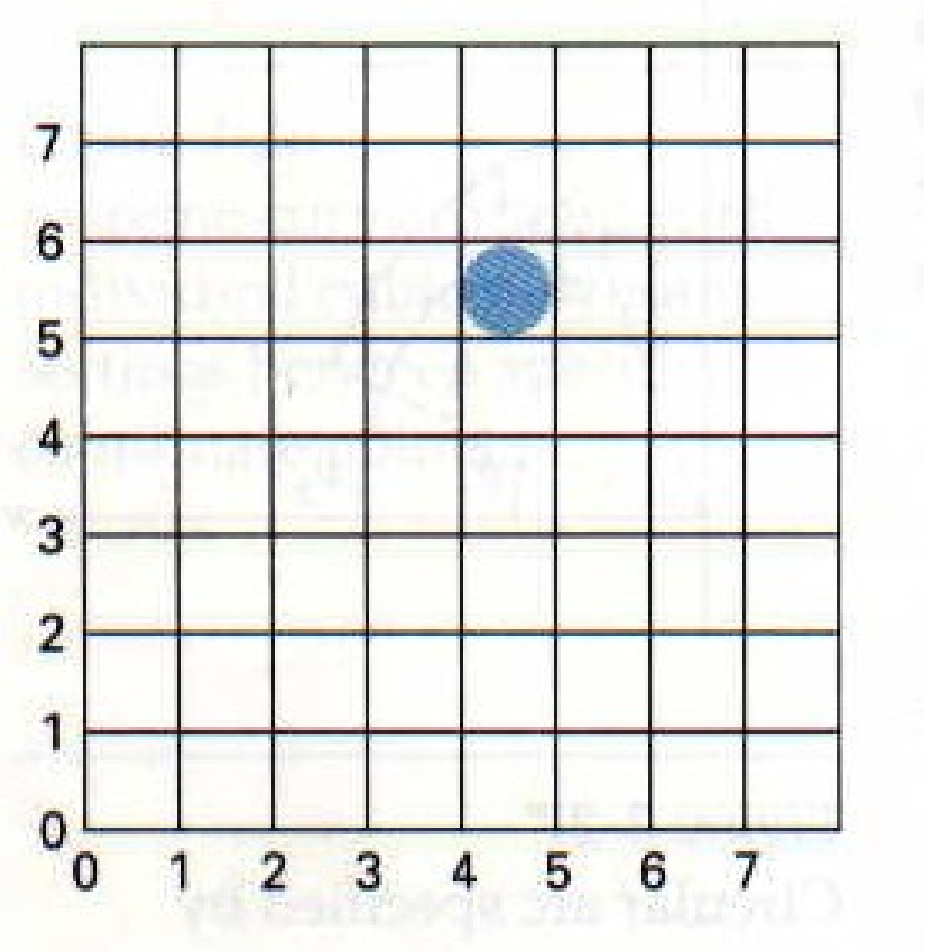
- For example, the mathematical line path for a polyline with screen endpoints (0, 0), (5, 2), and (1,4) is shown beside.



Pixel addressing and object geometry (cont.)

- With the coordinate origin at the lower left of the screen, each pixel area can be referenced by the integer grid coordinates of its lower left corner.

- The following figure illustrates this convention for an 8 by 8 section of a raster, with a single illuminated pixel at screen coordinate position (4, 5).



Pixel addressing and object geometry (cont.)

- In general, we identify the area occupied by a pixel with screen coordinates (x, y) as the unit square with diagonally opposite corners at (x, y) and ($x + 1$, $y + 1$).

- This pixel addressing scheme has several advantages:
  1. It avoids half-integer pixel boundary
  2. It facilitates precise object representations.
  3. Simplifies the processing involved in many scan conversion algorithms and in other raster procedures

Pixel addressing and object geometry (cont.)

Notes:

1) The previous algorithms for drawing line, circle, …etc are still valid when applied to input positions expressed as screen grid coordinates.

2) The decision parameter P_k is a measure of screen grid separation differences rather than separation differences from pixel centers.

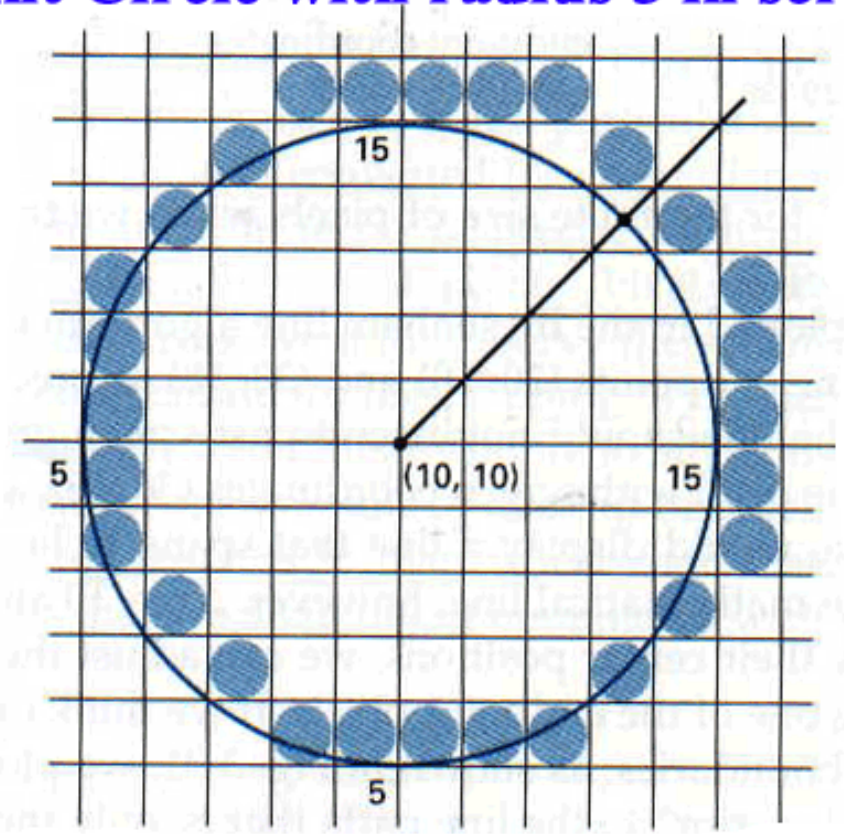
Pixel addressing and object geometry (cont.)

- A circle of radius 5 and center position (10, 10), for instance, would be displayed by the midpoint circle algorithm using screen grid coordinate positions.

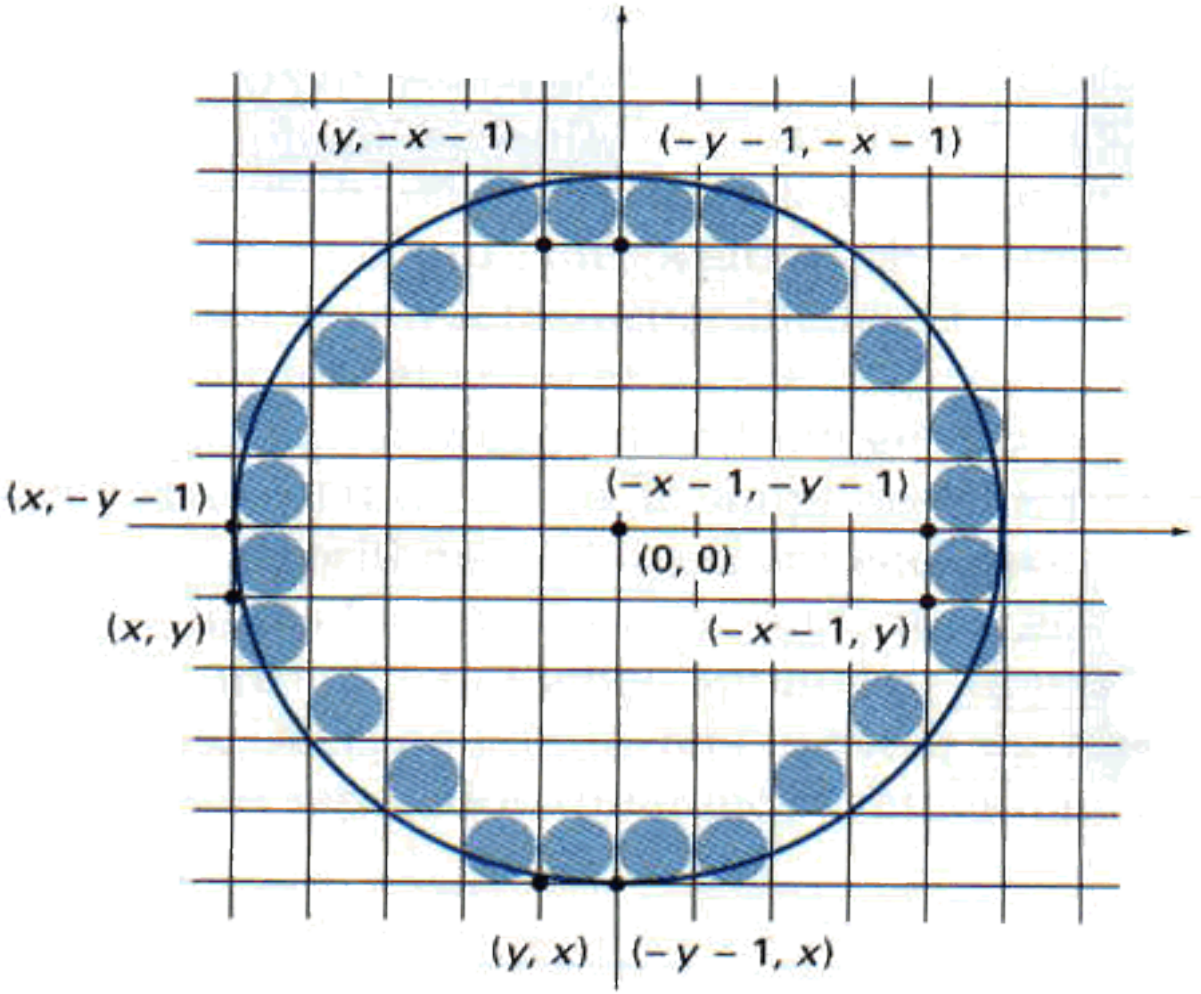
- But the plotted circle has a diameter of 11, To plot the circle with the defined diameter of 10, we can modify the circle algorithm to shorten each pixel scan line and each pixel column.

Pixel addressing and object geometry (cont.)

Midpoint Circle with radius 5 in screen coordinates



Pixel addressing and object geometry (cont.)



Modification of the circle path