Output Primitives

CSC309A - Computer Graphics
B. Tech. 2015

Course Leader:

Deepak V.

deepak.cs.et@msruas.ac.in

Objectives

- At the end of this lecture, student will be able to
  - Define characteristics of lines, conics and curves
  - Comprehend rasterization algorithms for line and conics

Contents

- Straight lines plotting algorithms
- Conics and curves plotting algorithms

Straight Line

- **Point plotting**
  - A single coordinate position is converted into appropriate operations for the output device in use

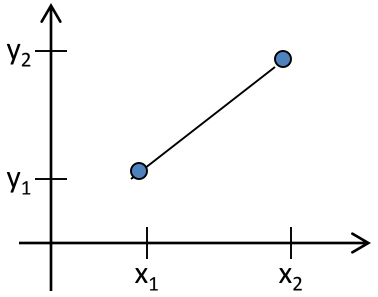
- **Line drawing**
  - It is done by calculating intermediate points along the line path between two specified endpoint. The output device is then directed to fill in these points.
    - Discrete coordinate points are calculated from the equation of the line
    - Computed points are converted to pixel positions by rounding of to nearest integer values (as screen locations are referenced with integer values)
  - Stair-step effect

Note:-Screen locations are referenced with integer values, so plotted position may only approximate actual line position for example computed line position (10.48, 20.51) would be converted to pixel position (10, 21) . This rounding of values to integers causes line to be displayed with stair case effect

Line Drawing Algorithm

- **Raster graphics device**
  - Lines are plotted with pixels
  - Step sizes in the horizontal and vertical directions are constrained by pixel separations.
  - Sampling positions along x axis

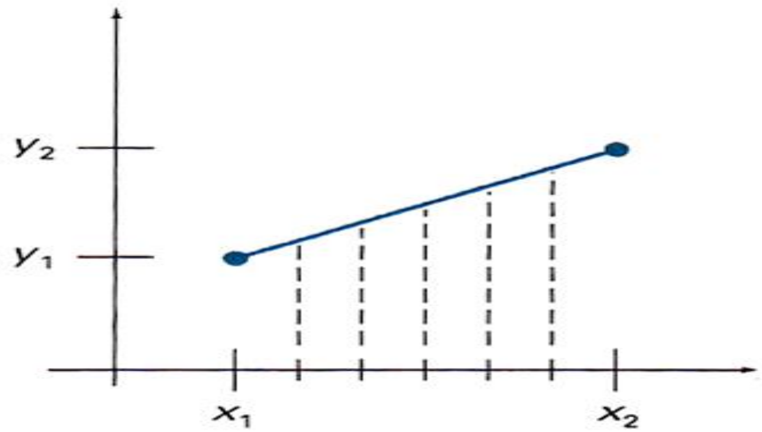
- **Line equation**
  - The Cartesian slope-intercept equation for a straight line is $y = m . x + b$
    where 'm' is slope of the line and 'b' is the y-intercept
  - m and b can be calculated as:
    $m = y_2 - y_1 / x_2 - x_1$
    $b = y_1 - m.x_1$



Line Drawing Algorithm

- Line equation
  - x interval $\Delta x \leftrightarrow$ y interval Δy

    $\Delta x = \Delta y / m \qquad \Delta y = m \cdot \Delta x$
  - These equations determine deflection voltages in analog device

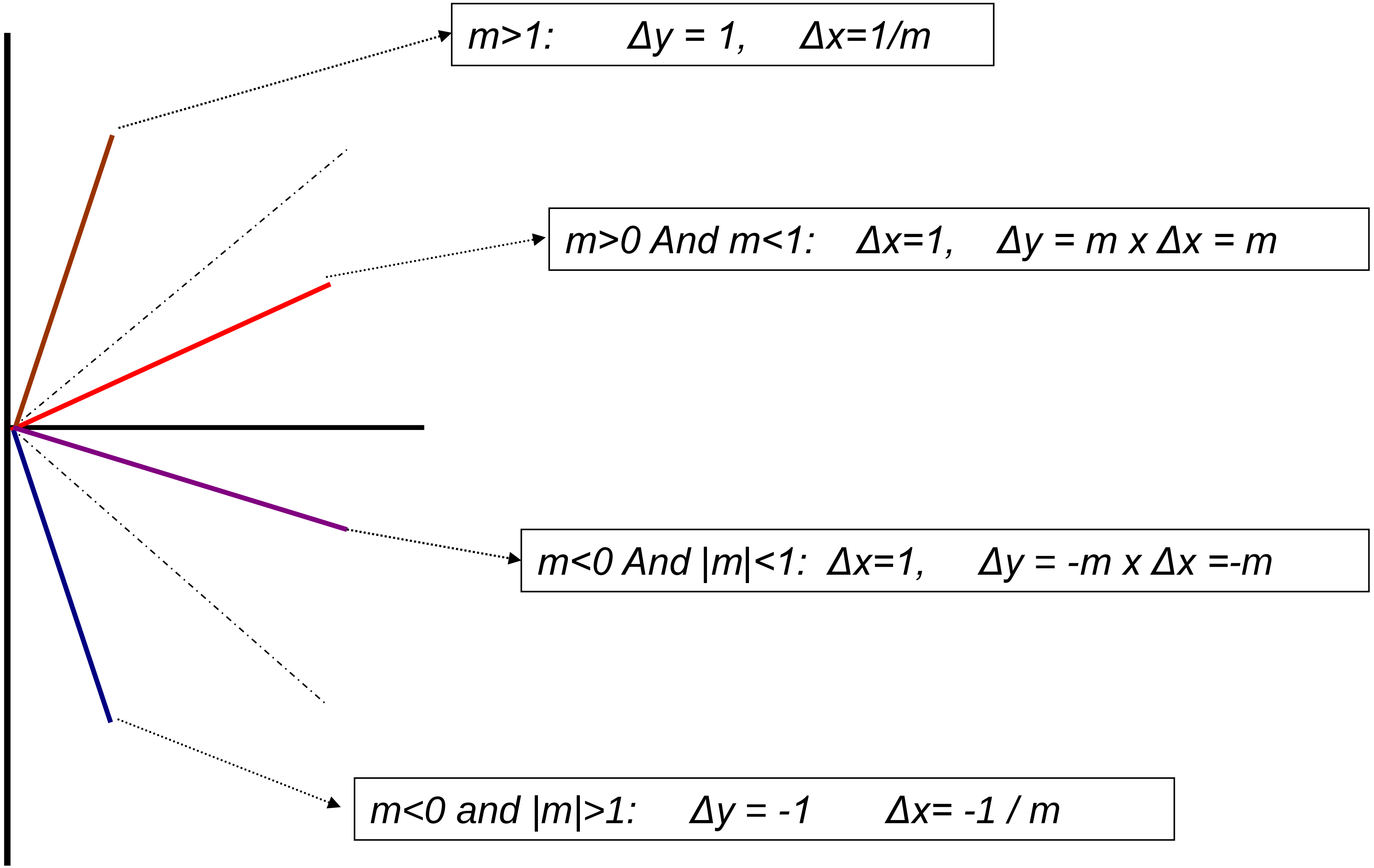


Straight line segment with five sampling positions along the x-axis between x_1 and x_2

Line Drawing Algorithm

- Line equation
  - If $|m|<1$
    - Δx can be set proportional to a small horizontal deflection voltage
    - $\Delta y = \Delta x \cdot m$
  - Else if $|m|>1$
    - Δy can be set proportional to a small vertical deflection voltage
    - $\Delta x = \Delta y / m$
  - Else the horizontal and vertical deflections voltages are equal.

Line Drawing Algorithm



Note: Endpoint is at the right

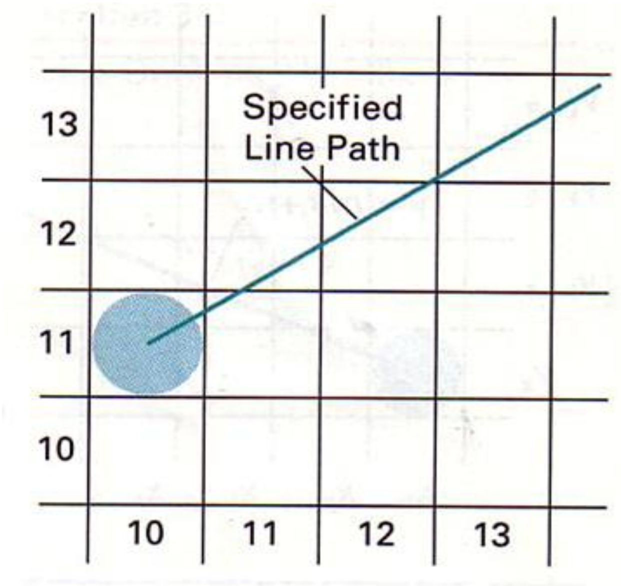
Line Drawing Algorithm

- Digital Differential Analyzer (DDA) algorithm
  - Scan-conversion line algorithm based on calculating either Δy or Δx
  - Assumption: Processed from the left endpoint to the right endpoint.
  - Though a faster method for calculating pixel position, accumulation of round-off error can cause the calculated pixel position to drift away from the true line path in case of long line segment

```
Δx = x_b-x_a ;  Δy = y_b-y_a ;
x = x_a, y = y_a ;
if  abs(Δx)> abs(Δy)
        step = abs(Δx)
else  step = abs(Δy)
Xincerement = Δx/step;
Yincerement = Δy/step
Setpixel ( x, y ) ;
for k=1:step
        x += Xincerement ;
        y += Yincerement ;
        Setpixel ( x, y ) ;
```

Line Drawing Algorithm

- Bresenham's algorithm
  - This algorithm can be applied to both lines and curves
  - To decide which of two possible pixel position closer to the line path at each sample step

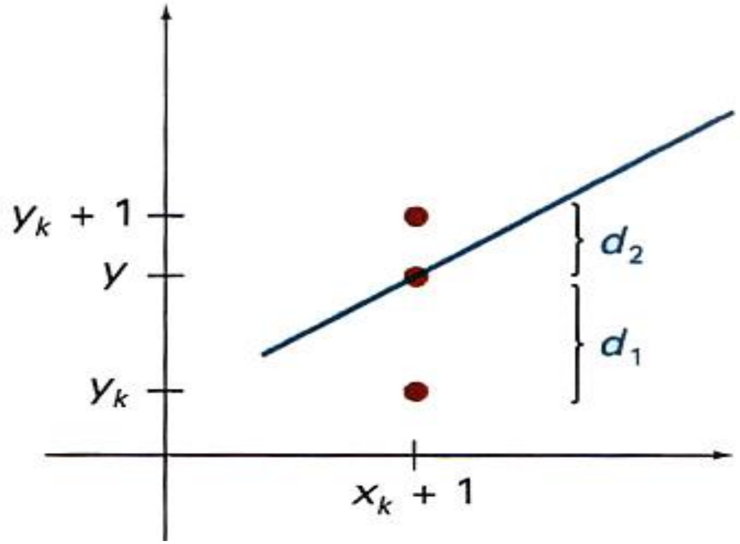


Section of a display screen where a st. line segment is to be plotted, starting from the pixel at column 10 on scan line 11

Next sample position : whether to plot the pixel at position (11, 11) or the one at (11, 12)?

Line Drawing Algorithm

- Bresenham's algorithm
  - Plot the pixel whose y value is closet to the line path
  - If $d_1 > d_2$ then (x_k+1, y_K+1) is plotted else (x_k+1, y_K) is plotted



Distances between pixel positions and the line y coordinate at sampling Position x_k+1

$$y = m(x_k+1) + b$$
$$d1 = y - y_k = m(x_k+1) + b - y_k$$
$$d2 = (y_k+1 - y) = y_k+1 - m(x_k+1) - b$$

$$d1 - d2 = 2m(x_k+1) - 2y_k + 2b - 1$$

$$m = \Delta y / \Delta x$$

Decision parameter Pk for kth step,

$$P_k = \Delta x\,(d1 - d2)$$
$$= 2\Delta y x_k - 2y_k \Delta x + \underbrace{(2\Delta y + \Delta x(2b-1))}_{\text{Constant: c}}$$

Line Drawing Algorithm

- Bresenham's algorithm
  - The sign of p_k is the same as the sign of d_1-d_2

$$p_{K+1} = 2\,\Delta y\, x_{k+1} - 2\, y_{k+1}\,\Delta x + C$$

$$p_{K+1} - p_K = 2\,\Delta y\,\underbrace{(x_{k+1} - x_k)}_{1} - 2\,\Delta x\,\underbrace{(y_{k+1} - y_k)}_{\text{1 or 0 depends on d1\& d2}}$$

If $p_K > 0 \rightarrow d_1 > d_2$
Both x & y to be incremented
Next point $\rightarrow x_k + 1,\ y_k + 1$

$p_{K+1} = p_k + 2\Delta y - 2\Delta x$

If $p_K < 0 \rightarrow d_1 < d_2$
Only x to be incremented
$x_k + 1,\ y_k$

$p_{K+1} = p_k + 2\Delta y$

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

Line Drawing Algorithm

- Bresenham's algorithm - steps
  1. Input the two line endpoints and store the left endpoint in (x_0, y_0)
  2. Load (x_0, y_0) into the frame buffer; that is, plot the first point
  3. Calculate constants Δy, Δx, 2Δy, 2Δy - 2Δx and obtain the starting value for the decision parameter as $p_0 = 2\Delta y - \Delta x$
  4. At each x_k along the line, starting at k = 0, perform the following test:
     1. If $p_k < 0$, the next point to plot is (x_k+1, y_k) and $p_{k+1} = p_k + 2\Delta y$
     2. Otherwise, the next point to plot is (x_k+1, y_k+1) and $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
  5. Repeat step 4 Δx times

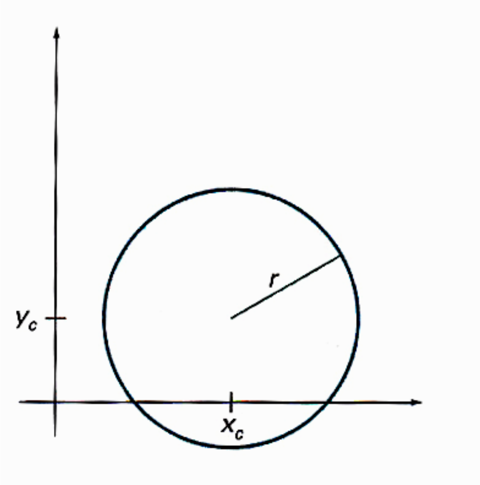
Line Drawing Algorithm

- Bresenham's algorithm - example
  - To illustrate the algorithm, let's digitize the line with endpoints (20, 10) and (30, 18).
  - This line has a slope of 0.8, with $\Delta y=8$ $\Delta x=10$

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)

Conics and Curves

- Circle
  - Defined as the set of points that are all at a given distance r from a center position (x_c, y_c)
  - $(x - x_c)^2 + (y - y_c)^2 = r^2$
  - Position of points on circumference is calculated by stepping along the x axis in unit steps from $x_c - r$ to $x_c + r$ and calculating the corresponding 'y' values at each position as: $y = y_c \pm (r^2 - (x - x_c)^2)^{1/2}$



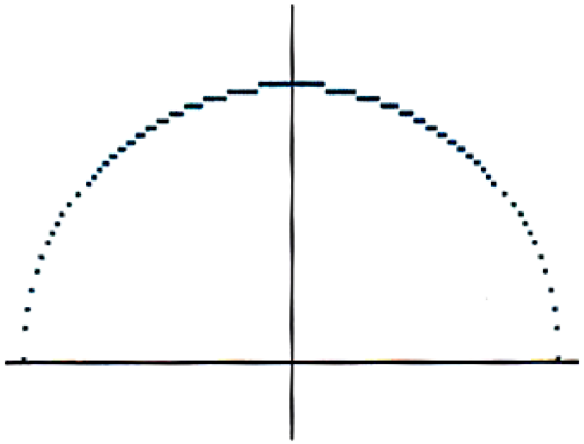
Conics and Curves

- Circle
  - Issues : considerable computation and non-uniform spacing between plotted pixel position
  - One way to eliminate unequal spacing is use of polar form

    $x=x_c + r \cos \theta$

    $y=y_c + r \sin \theta$

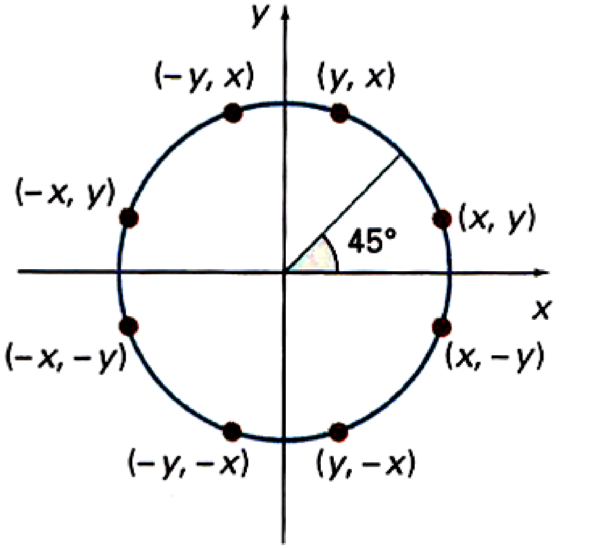
    We can set the step size 1/r



Positive half of a circle plotted with equation given on previous slide with $(x_c, y_c) = (0,0)$

Conics and Curves

- Midpoint circle algorithm
  - Computation can be reduced by considering the symmetry of circle
  - Bresenham's algorithm is adapted to circle generation: midpoint circle algorithm
  - $f_{circle}(x,y) = x^2 + y^2 - r^2$
    - < 0 if (x,y) is inside circle boundary
    - $= 0$ if (x,y) is on the circle boundary
    - > 0 if (x,y) is outside circle boundary



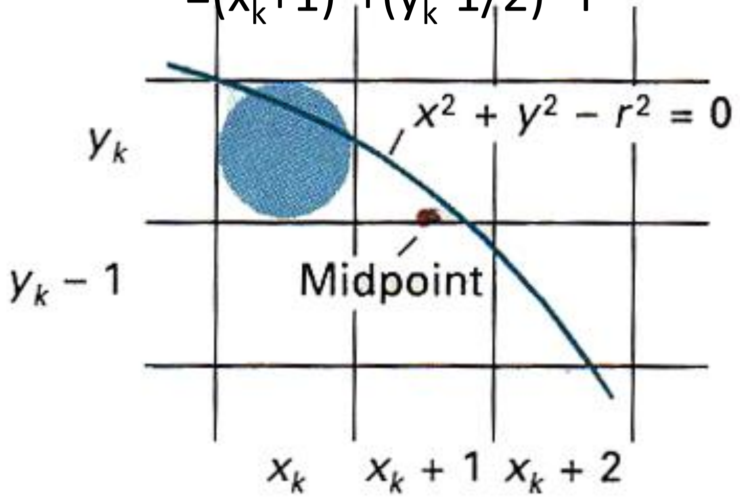
Symmetry of a circle : Calculation of a circle point (x, y) in one octant yields the circle points shown for the other seven octants

Conics and Curves

- Midpoint circle algorithm
  - The midpoint between the two candidate pixels at sampling position x_k+1 → (x_k+1, y_k) or (x_k+1, y_k-1) ?
  - Decision parameter is the circle function evaluated at the midpoint between these two pixels

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

$=(x_k+1)^2+(y_k-1/2)^2-r^2$



Midpoint between candidate pixels at sampling position X_k+1 along a circular path

Conics and Curves

- Midpoint circle algorithm
  - If $p_k < 0$, this midpoint is inside the circle and the pixel on scan line y_k is closer to the circle boundary. Otherwise, the mid-position is outside or on the circle boundary, and the pixel on scan line $y_k - 1$ will be selected.
  - Successive decision parameters are obtained using incremental calculations. For sampling position of $x_{k+1}+1 = x_k+2$, decision parameter is:

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

Where y_{k+1} is either y_k or y_{k-1} depending on the sign of P_k

If pk <0 ($y_{k+1}=y_k$) and $p_{k+1}=2x_{k+1}+1$

else ($y_{k+1}=y_{K-1}$) and $p_{k+1}=2x_{k+1}+1-2y_{k+1}$

Conics and Curves

- Midpoint circle algorithm
  - The initial decision parameter is obtained by evaluating the circle function at the start position $(x_0, y_0) = (0, r)$.

    $P_0 = f_{circle}(1, \; r-1/2)$

    $P_0 = 1 + (r-1/2)^2 - r^2$

    $\quad = 5/4 - r$

    $P_0 = 1 - r$ (if r is an integer)

Conics and Curves

- Midpoint circle algorithm - steps

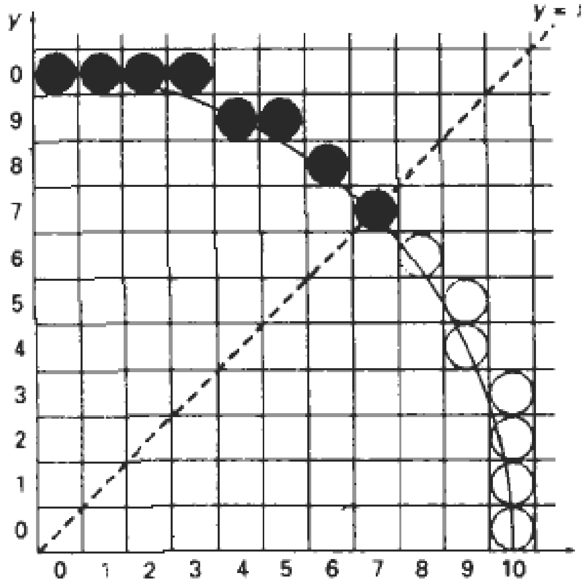
1. Input radius r, circle center (x_c, y_c) & obtain the first point on circumference of a circle centered on origin as $(x_0,y_0)=(0,r)$
2. Calculate the initial value of the decision parameter as p_0.
3. At each x_k position, starting at $k = 0$, perform following test
    1. If $p_k < 0$, the next point along the circle centered on (0,0) is (x_k+1, y_k) and $p_{k+1}= p_k + 2x_{k+1}+1$
    2. Otherwise, the next point along the circle is $(x_k+1, y_k - 1)$ and $p_{k+1}= p_k + 2x_{k+1} +1 – 2y_{k+1}$

       where $2x_{k+1}= 2x_k + 2$ and $2y_{k+1}= 2y_k - 2$.
4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and
6. Plot the coordinate values: $x = x + x_c$, $y = y + y_c$
7. Repeat steps 3 through 5 until $x \geq y$.

Conics and Curves

- Midpoint circle algorithm – example
  - Radius r = 10,
  - $P_0 = 1\text{-}r = -9$

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



Selected pixel positions (solid circles) along a circle path with radius r=10 centered on the origin. Open circles show the symmetry position in the first quadrant

Conics and Curves

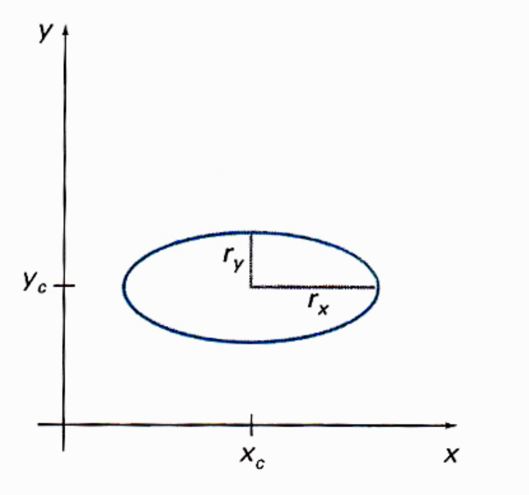
- Ellipse
  - Parametric equation using polar coordinates:

    $x = x_c + r_x \cos\theta$

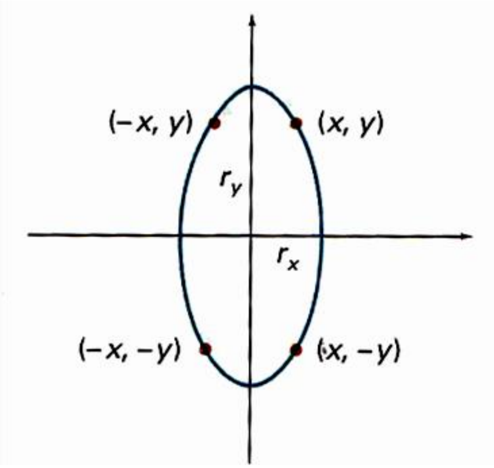
    $y = y_c + r_y \sin\theta$

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

  - An ellipse in standard position is symmetric between quadrants but it is not symmetric between the two octants of a quadrant

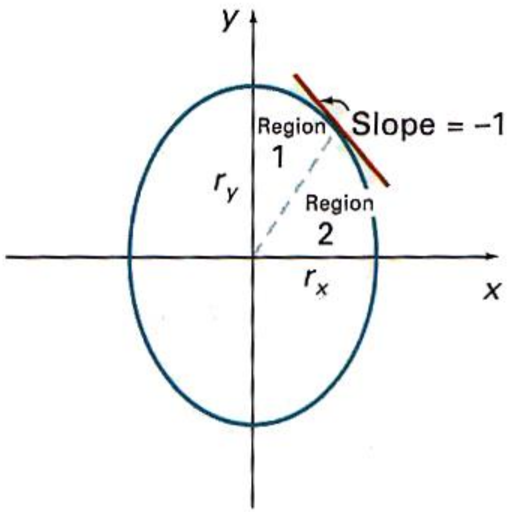


Ellipse centered at (x_c, y_c) with semi-major axis r_x and semi-minor axis r_y



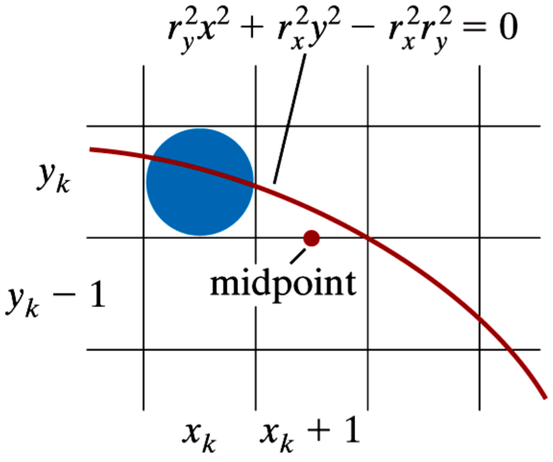
Conics and Curves

- Midpoint ellipse method
  - The first quadrant is processed by taking unit steps in the x-direction where slope of the curve is <1 and taking unit steps in the y-direction where the slope is > 1.
  - $f_{ellips}(x,y) = r^2_y x^2 + r^2_x y^2 - r^2_x r^2_y$
    - < 0 if (x,y) is inside ellipse boundary
    - = 0 if (x,y) is on the ellipse boundary
    - > 0 if (x,y) is outside ellipse boundary

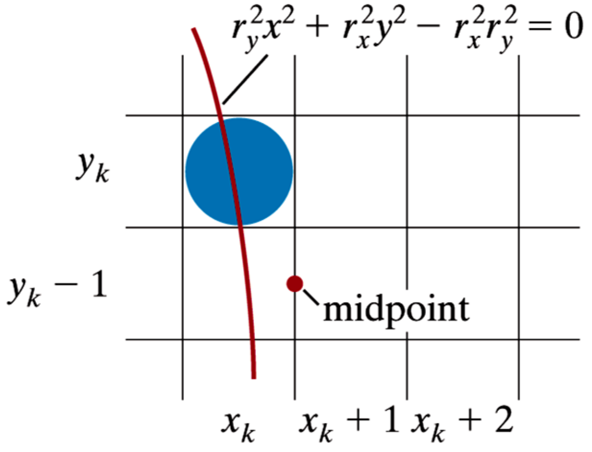


Conics and Curves

- Midpoint ellipse method
  - At each sampling position, the next pixel along the ellipse path is selected according to the sign of the ellipse function evaluated at the midpoint between the two candidate pixels



Midpoint between candidate pixels at sampling position x_k+1 along an elliptical path



Midpoint between candidate pixels at sampling position y_k-1 along an elliptical path

Conics and Curves

- Polynomial curves
  - A polynomial function of n^{th} degree in *x* is defined as

$$y = \sum_{k=0}^{n} a_k x^k = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + a_n x^n$$

  - Designing object shapes or motion paths is typically done by specifying a few points to define the general curve contour, then fitting
    - The selected points with a polynomial
    - One way to accomplish the curve fitting is to construct a cubic polynomial curve section between each pair of specified points

Conics and Curves

- Polynomial curves
  - Each curve section is then described in parametric form as $x=ax_0+ax_1u+ax_2u^2+ax_3u^3$

    $y=ay_0+ay_1u+ay_2u^2+ay_3u^3$
  - Parameter ***u*** varies over the interval **0** to 1.
  - Boundary conditions
    - Two adjacent curve sections have the same coordinate position at the boundary
    - To match the two curve slopes at the boundary so that one continuous, smooth curve can be obtained
  - Continuous curves that are formed with polynomial pieces are called spline curves or splines

Lecture Summary

- Bresenham's Algorithm gives good rasterization for line
- Lines are plotted with pixels
- Mid-point algorithm are adapted from Bresenham's Algorithm, and are best suited for curves and conics
- Continuous curves that are formed with polynomial pieces are called spline curves or splines