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:

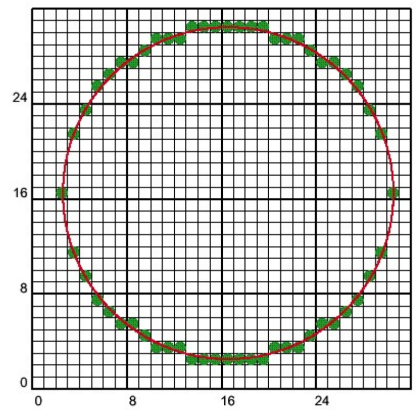
$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

Circle Generating Algorithms

- We can re-write the circle equation as:

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

- 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
  - $\mathbf{x = xc + r \cos \theta}$
  - $\mathbf{y = yc + r \sin \theta}$
- 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.

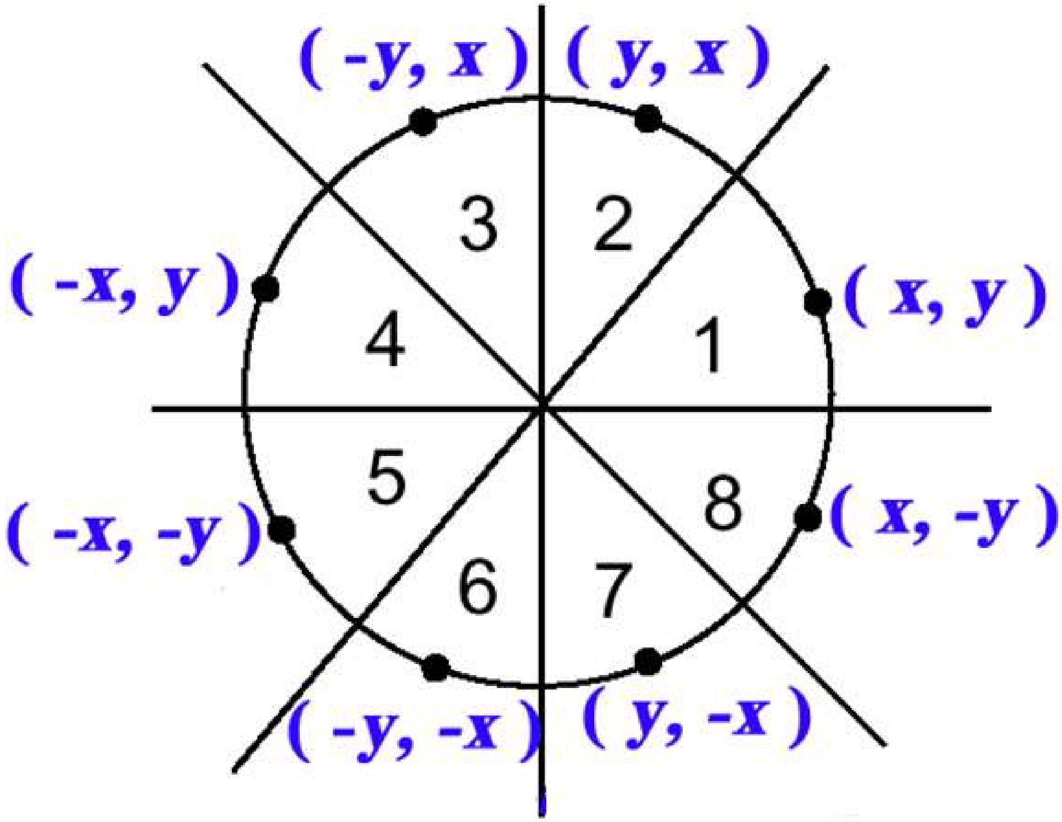
Circle Generating Algorithms

- 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)



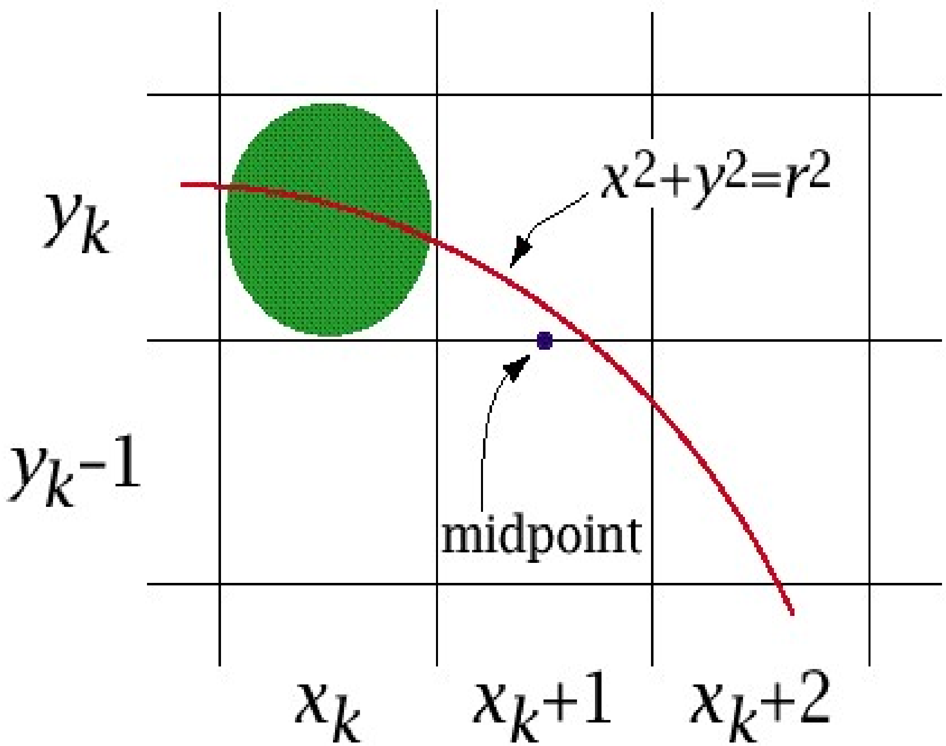
Midpoint Circle Algorithm

- 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

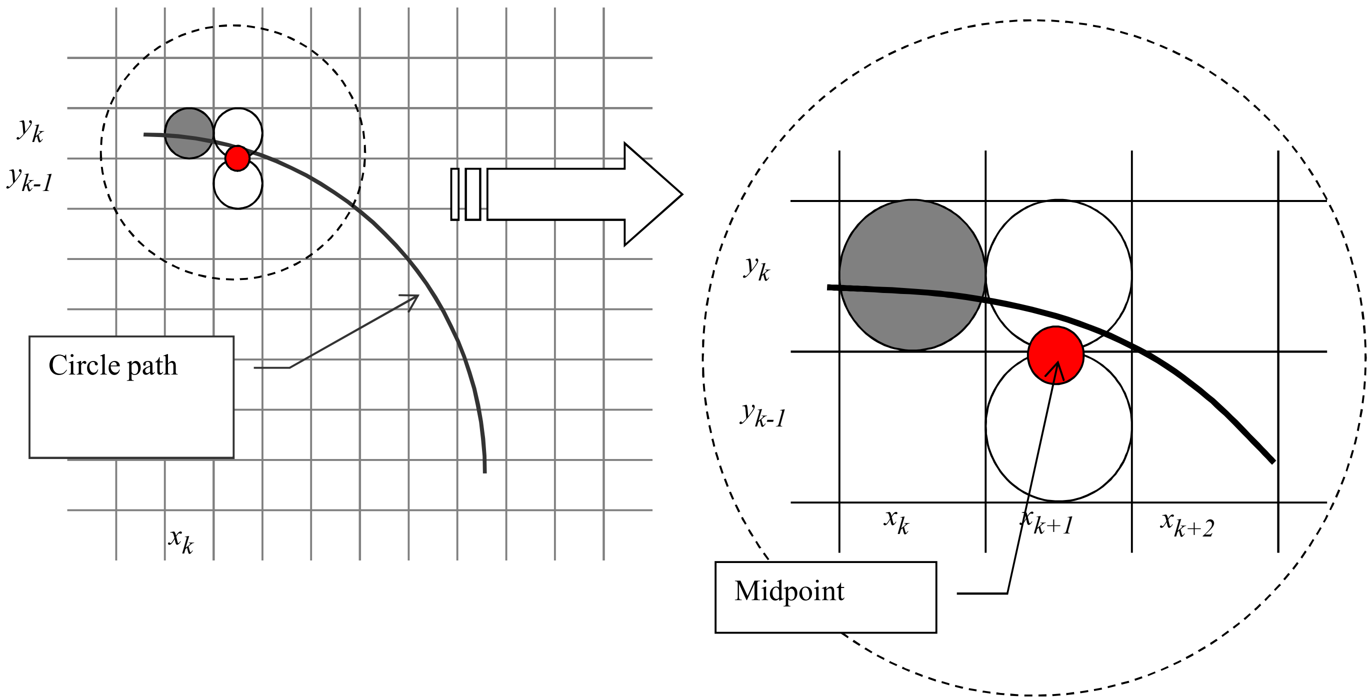


Midpoint Circle Algorithm

- 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)
- +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:

$$p_k = fcircle\,(x_k + 1, y_k - ½) = (x_k + 1)^2 + (y_k - ½)^2 - r^2$$

- *Successive decision parameters are obtained using incremental calculations.*
  - $P_{k+1} = fcircle\,(x_{k+1} + 1, y_{k+1} - ½) = [(x_k + 1) + 1]^2 + (y_{k+1} - ½)^2 - r^2$
  - $P_{k+1} = pk + 2(x_k + 1) + (y^2_{k+1} - y^2_k) - (y_{k+1} - y_k) + 1$

- *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 $\boldsymbol{x} \geq \boldsymbol{y}$.

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.

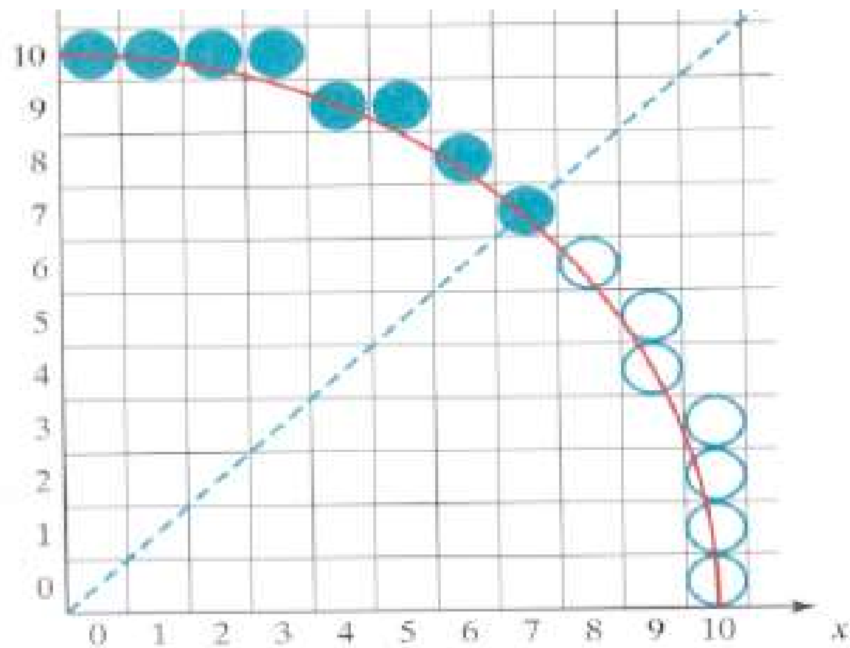
Solution:

- $p_0 = 1 - r = -9$
- 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})	$2x_{k+1}$	$2y_{k+1}$
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

Midpoint Circle Algorithm

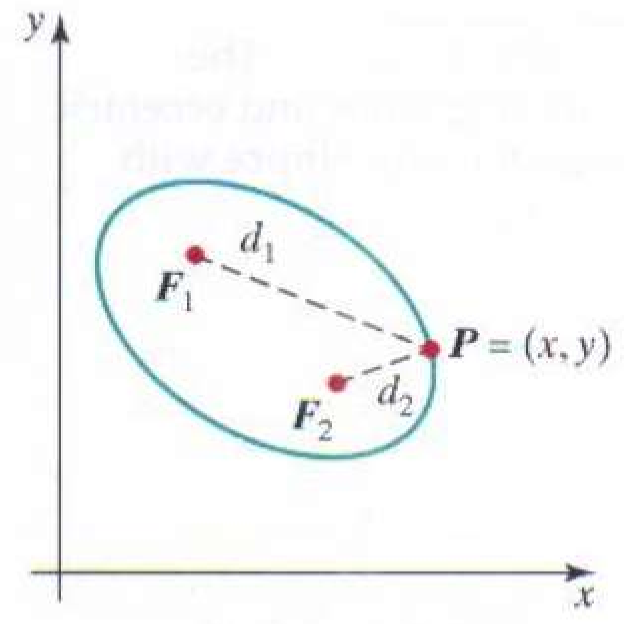


Ellipse Generating Algorithms

- 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

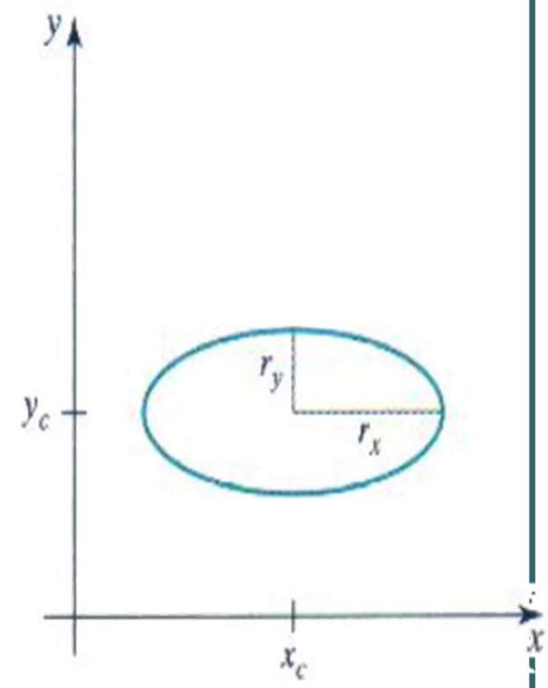
$$\sqrt{(x-x_1)^2+(y-y_1)^2}+\sqrt{(x-x_2)^2+(y-y_2)^2}=\text{consta}$$

- 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

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



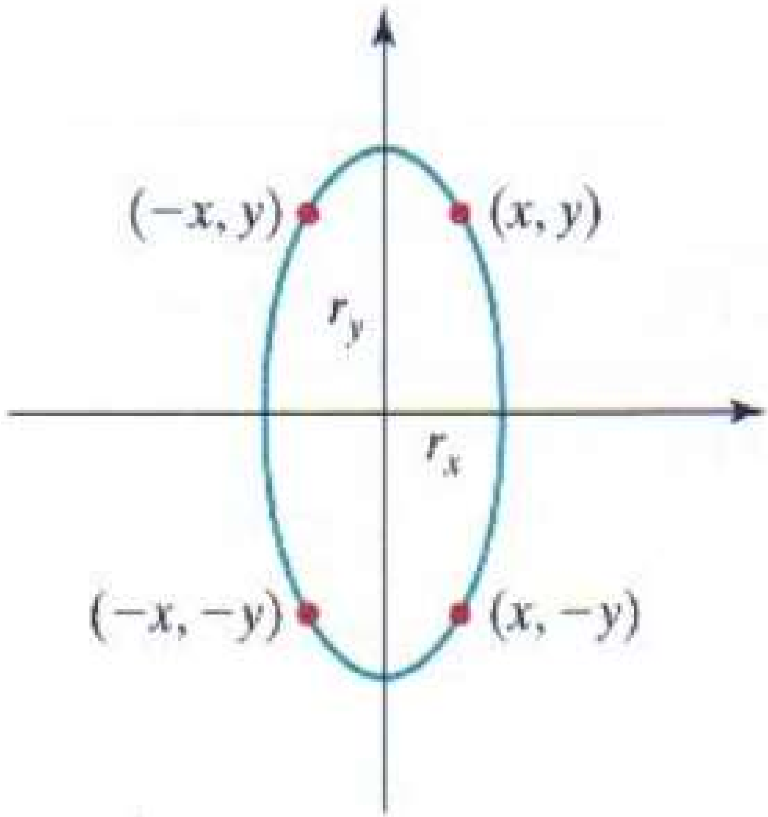
Ellipse Generating Algorithms

- 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

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

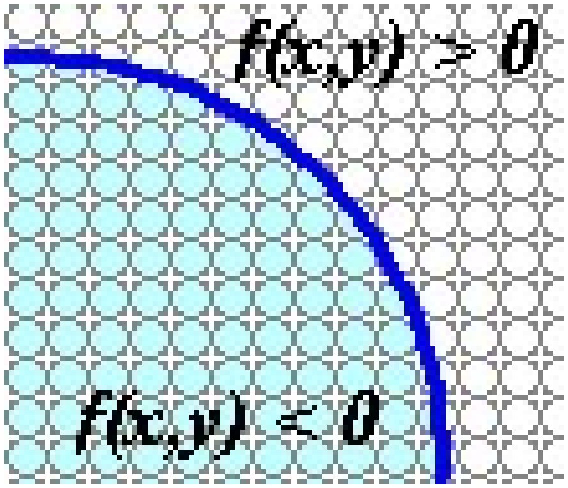
- It has the following properties:

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

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

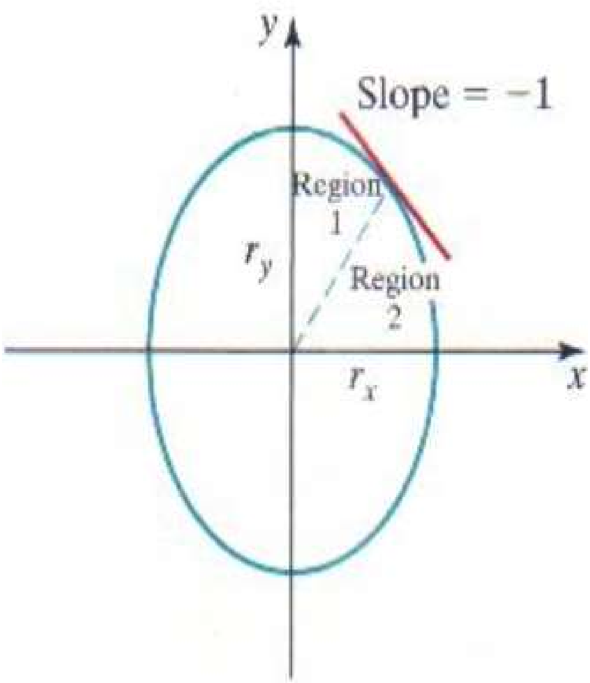
  $f_e(x,y) = 0$ for a point on the ellipse

- 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: $$p1_k = f_e\left(x_k + 1, y_k - ½\right)$$

$-ve$	• 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**:

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

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

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

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

- 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