

2-4 Circle Generation—Bresenham's Algorithm

In addition to rasterizing straight lines, it is necessary to rasterize other more complicated functions. Considerable attention has been given to conic sections, i.e., circles, ellipses, parabolas, hyperbolas (see [Pitt67; Jord73; Bels76; Ramo76; Vana84, 85; Fiel86]). The circle has, of course, received the greatest attention. (See also [Horn76; Badl77; Doro79; Suen79].) One of the most efficient and easiest to derive of the circle algorithms is due to Bresenham [Bres77]. To begin, note that only one octant of the circle need be generated. The other parts are obtained by successive reflections. This is illustrated in Fig. 2-10. If the first octant (0 to 45° ccw) is generated, the second octant is obtained by reflection through the line $y = x$ to yield the first quadrant. The results in the first quadrant are reflected through the line $x = 0$ to obtain those in the second quadrant. The combined results in the upper semicircle are reflected through the line $y = 0$ to complete the circle. Figure 2-10 gives the appropriate two-dimensional reflection matrices.

To derive Bresenham's circle generation algorithm, consider the first quadrant of an origin-centered circle. Notice that if the algorithm begins at $x = 0$, $y = R$, then for clockwise generation of the circle y is a monotonically decreasing function of x in the first quadrant (see Fig. 2-11). Similarly, if the algorithm begins at $y = 0$, $x = R$, then for counterclockwise generation of the circle x is a monotonically decreasing function of y . Here, clockwise generation starting at $x = 0$, $y = R$ is chosen. The center of the circle and the starting point are both assumed located precisely at pixel elements.

Assuming clockwise generation of the circle, then for any given point on the circle there are only three possible selections for the next pixel which best represents the circle: horizontally to the right, diagonally downward to the right and

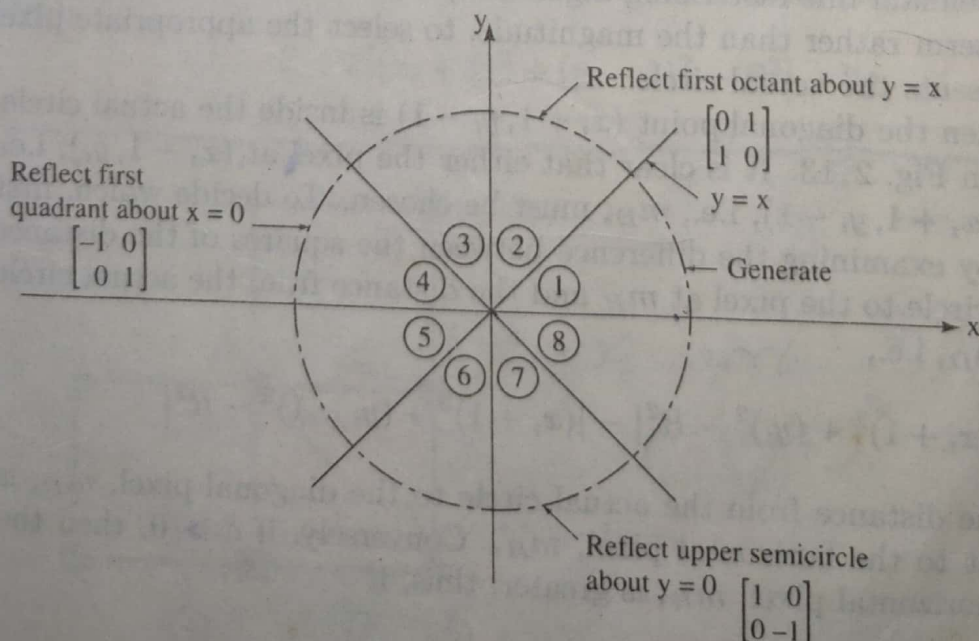


Figure 2-10 Generation of a complete circle from the first octant.

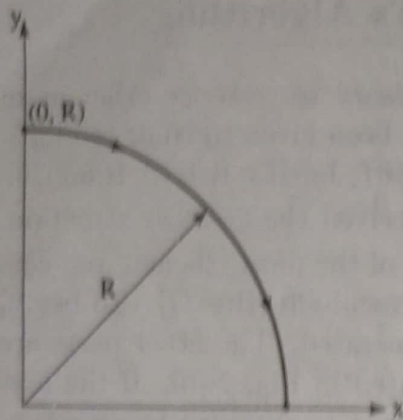


Figure 2-11 First quadrant of a circle.

vertically downward. These are labeled m_H , m_D , m_V , respectively, in Fig. 2-12. The algorithm chooses the pixel which minimizes the square of the distance between one of these pixels and the true circle, i.e., the minimum of

$$m_H = |(x_i + 1)^2 + (y_i)^2 - R^2|$$

$$m_D = |(x_i + 1)^2 + (y_i - 1)^2 - R^2|$$

$$m_V = |(x_i)^2 + (y_i - 1)^2 - R^2|$$

The calculations are simplified by noting that there are only five possible types of intersections of the circle and the raster grid in the vicinity of the point (x_i, y_i) . These are shown in Fig. 2-13.

The difference between the square of the distance from the center of the circle to the diagonal pixel at $(x_i + 1, y_i - 1)$ and the distance to a point on the circle R^2 is

$$\Delta_i = (x_i + 1)^2 + (y_i - 1)^2 - R^2$$

As with the Bresenham line rasterizing algorithm, it is desirable to use only the sign of an error term rather than the magnitude, to select the appropriate pixel which best represents the actual circle.

If $\Delta_i < 0$, then the diagonal point $(x_i + 1, y_i - 1)$ is inside the actual circle, i.e., case 1 or 2 in Fig. 2-13. It is clear that either the pixel at $(x_i + 1, y_i)$, i.e., m_H , or that at $(x_i + 1, y_i - 1)$, i.e., m_D , must be chosen. To decide which, first consider case 1 by examining the difference between the squares of the distance from the actual circle to the pixel at m_H and the distance from the actual circle to the pixel at m_D , i.e.,

$$\delta = |(x_i + 1)^2 + (y_i)^2 - R^2| - |(x_i + 1)^2 + (y_i - 1)^2 - R^2|$$

If $\delta < 0$, then the distance from the actual circle to the diagonal pixel, m_D , is greater than that to the horizontal pixel, m_H . Conversely, if $\delta > 0$, then the distance to the horizontal pixel, m_H , is greater; thus, if

$$\begin{array}{ll} \delta \leq 0 & \text{choose } m_H \text{ at } (x_i + 1, y_i) \\ \delta > 0 & \text{choose } m_D \text{ at } (x_i + 1, y_i - 1) \end{array}$$

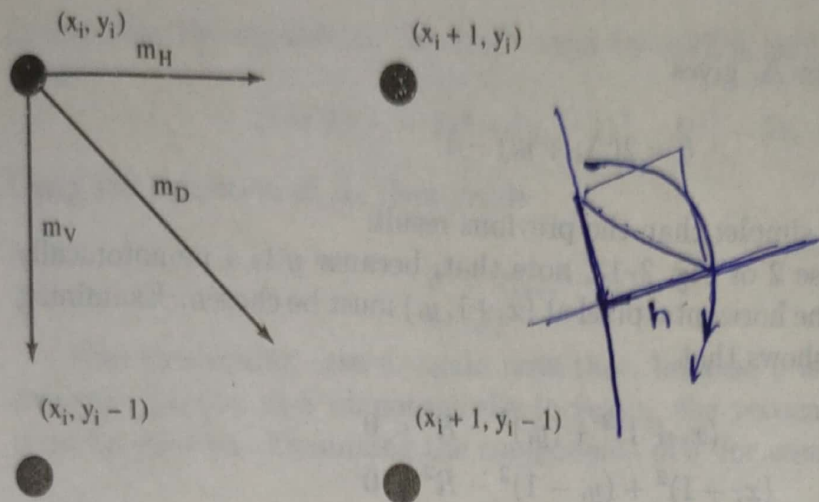


Figure 2-12 First quadrant pixel selections.

The horizontal move is selected when $\delta = 0$, i.e., when the distances are equal.

The work involved in evaluating δ is reduced by noting that for case 1

$$(x_i + 1)^2 + (y_i)^2 - R^2 \geq 0$$

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 < 0 \quad \text{which pixel is set is less than}$$

because the diagonal pixel at $(x_i + 1, y_i - 1)$ is always inside the circle, and the horizontal pixel at $(x_i + 1, y_i)$ is always outside the circle. Thus, δ can be evaluated as

$$\delta = (x_i + 1)^2 + (y_i)^2 - R^2 + (x_i + 1)^2 + (y_i - 1)^2 - R^2$$

Completing the square for the $(y_i)^2$ term by adding and subtracting $-2y_i + 1$ yields

$$\delta = 2[(x_i + 1)^2 + (y_i - 1)^2 - R^2] + 2y_i - 1$$

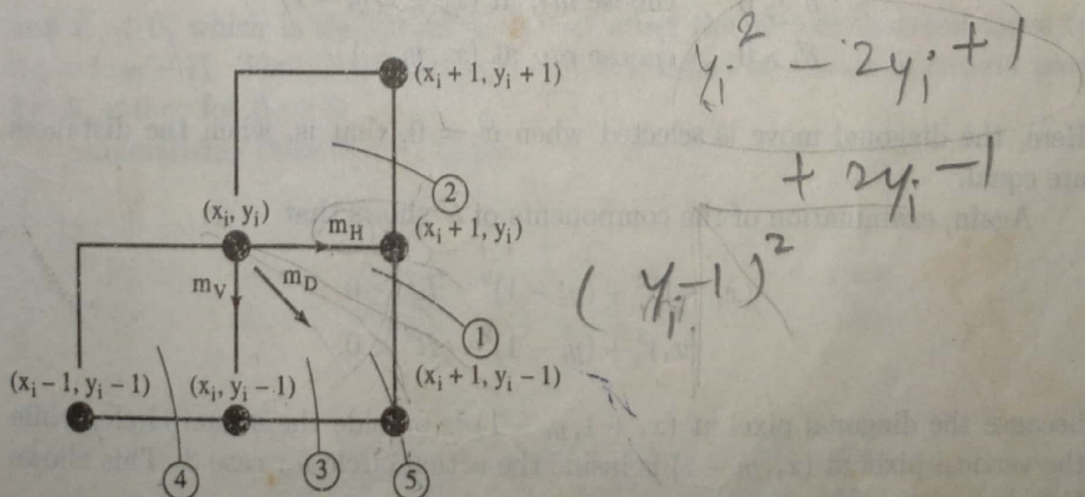


Figure 2-13 Intersection of a circle and the raster grid.

Using the definition for Δ_i gives

$$\delta = 2(\Delta_i + y_i) - 1$$

which is considerably simpler than the previous result.

In considering case 2 of Fig. 2-13, note that, because y is a monotonically decreasing function, the horizontal pixel at $(x_i + 1, y_i)$ must be chosen. Examining the components of δ shows that

$$\begin{aligned}(x_i + 1)^2 + (y_i)^2 - R^2 &< 0 \\ (x_i + 1)^2 + (y_i - 1)^2 - R^2 &< 0\end{aligned}$$

because the horizontal pixel at $(x_i + 1, y_i)$ and the diagonal pixel at $(x_i + 1, y_i - 1)$ both lie inside the actual circle for case 2. Hence, $\delta < 0$; and the correct pixel at $(x_i + 1, y_i)$ is selected, using the same criteria as in case 1.

If $\Delta_i > 0$, then the diagonal point $(x_i + 1, y_i - 1)$ is outside the actual circle, i.e., case 3 or 4 in Fig. 2-13. Here, it is clear that either the pixel at $(x_i + 1, y_i - 1)$, i.e., m_D , or that at $(x_i, y_i - 1)$, i.e., m_V , must be chosen. Again, the decision criteria is obtained by first considering case 3 and examining the difference between the squares of the distance from the actual circle to the diagonal pixel at m_D , and the distance from the actual circle to the pixel at m_V , that is

$$\delta' = |(x_i + 1)^2 + (y_i - 1)^2 - R^2| - |(x_i)^2 + (y_i - 1)^2 - R^2|$$

If $\delta' < 0$, then the distance from the actual circle to the vertical pixel at $(x_i, y_i - 1)$ is larger and the diagonal move m_D to the pixel at $(x_i + 1, y_i - 1)$ is chosen. Conversely, if $\delta' > 0$, then the distance from the actual circle to the diagonal pixel is greater, and the vertical move to the pixel at $(x_i, y_i - 1)$ is chosen. Thus, if

$$\begin{aligned}\delta' \leq 0 & \quad \text{choose } m_D \text{ at } (x_i + 1, y_i - 1) \\ \delta' > 0 & \quad \text{choose } m_V \text{ at } (x_i, y_i - 1)\end{aligned}$$

Here, the diagonal move is selected when $\delta' = 0$, that is, when the distances are equal.

Again, examination of the components of δ' shows that

$$\begin{aligned}(x_i + 1)^2 + (y_i - 1)^2 - R^2 &\geq 0 \\ (x_i)^2 + (y_i - 1)^2 - R^2 &< 0\end{aligned}$$

because the diagonal pixel at $(x_i + 1, y_i - 1)$ is outside the actual circle, while the vertical pixel at $(x_i, y_i - 1)$ is inside the actual circle for case 3. This allows δ' to be written as

$$\delta' = (x_i + 1)^2 + (y_i - 1)^2 - R^2 + (x_i)^2 + (y_i - 1)^2 - R^2$$

Completing the square for the $(x_i)^2$ term by adding and subtracting $2x_i + 1$ yields

$$\delta' = 2[(x_i + 1)^2 + (y_i - 1)^2 - R^2] - 2x_i - 1$$

Using the definition of Δ_i then yields

$$\delta' = 2(\Delta_i - x_i) - 1$$

Now considering case 4, again note that, because y is a monotonically decreasing function as x monotonically increases, the vertical pixel at $(x_i, y_i - 1)$ must be selected. Examining the components of δ' for case 4 shows that

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 > 0$$

$$(x_i)^2 + (y_i - 1)^2 - R^2 > 0$$

because both the vertical and diagonal pixels are outside the actual circle. Hence, $\delta' > 0$; and the correct choice of m_V is selected using the same criteria developed for case 3.

It remains only to examine case 5 of Fig. 2-13, which occurs when the diagonal pixel at $(x_i + 1, y_i - 1)$ lies on the actual circle, i.e., for $\Delta_i = 0$. Examining the components of δ shows that

$$(x_i + 1)^2 + (y_i)^2 - R^2 > 0$$

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 = 0$$


Hence, $\delta > 0$; and the diagonal pixel at $(x_i + 1, y_i - 1)$ is selected. Similarly, the components of δ' are

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 = 0$$

$$(x_i)^2 + (y_i - 1)^2 - R^2 < 0$$

and $\delta' < 0$, which is the condition for selecting the correct diagonal move to $(x_i + 1, y_i - 1)$. Thus, the case of $\Delta_i = 0$ is satisfied by the same criteria used for $\Delta_i < 0$ or for $\Delta_i > 0$.

Summarizing these results yields

	$\Delta_i < 0$		
	$\delta \leq 0$	choose the pixel at $(x_i + 1, y_i)$	$\rightarrow m_H$
	$\delta > 0$	choose the pixel at $(x_i + 1, y_i - 1)$	$\rightarrow m_D$
	$\Delta_i > 0$		
	$\delta' \leq 0$	choose the pixel at $(x_i + 1, y_i - 1)$	$\rightarrow m_D$
	$\delta' > 0$	choose the pixel at $(x_i, y_i - 1)$	$\rightarrow m_V$
	$\Delta_i = 0$	choose the pixel at $(x_i + 1, y_i - 1)$	$\rightarrow m_D$

Simple recursion relationships which yield an incremental implementation of the algorithm are easily developed. First, consider the horizontal movement, m_H , to the pixel at $(x_i + 1, y_i)$. Call this next pixel location $(i + 1)$. The coordinates of the new pixel and the value of Δ_i are then

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i$$

$$\begin{aligned}\Delta_{i+1} &= (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2 \\ &= (x_{i+1})^2 + 2x_{i+1} + 1 + (y_i - 1)^2 - R^2 \\ &= (x_i + 1)^2 + (y_i - 1)^2 - R^2 + 2x_{i+1} + 1 \\ &= \Delta_i + 2x_{i+1} + 1\end{aligned}$$

Similarly, the coordinates of the new pixel and the value of Δ_i for the move m_D to $(x_i + 1, y_i - 1)$ are

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i - 1$$

$$\Delta_{i+1} = \Delta_i + 2x_{i+1} - 2y_{i+1} + 2$$

Those for the move m_V to $(x_i, y_i - 1)$ are

$$x_{i+1} = x_i$$

$$y_{i+1} = y_i - 1$$

$$\Delta_{i+1} = \Delta_i - 2y_{i+1} + 1$$

A pseudocode implementation of the Bresenham circle algorithm is given below.

Bresenham's incremental circle algorithm for the first quadrant

all variables are assumed integer

initialize the variables

$$x_i = 0$$

$$y_i = R$$

$$\Delta_i = 2(1 - R)$$

$$\text{Limit} = 0$$

while $y_i \geq \text{Limit}$

 call setpixel(x_i, y_i)

 determine if case 1 or 2, 4 or 5, or 3

 if $\Delta_i < 0$ then

$$\delta = 2\Delta_i + 2y_i - 1$$

 determine whether case 1 or 2

 if $\delta \leq 0$ then

 call mh(x_i, y_i, Δ_i)


```

else
    call md( $x_i, y_i, \Delta_i$ )
end if
else if  $\Delta_i > 0$  then
     $\delta' = 2\Delta_i - 2x_i - 1$ 
    determine whether case 4 or 5
    if  $\delta' \leq 0$  then
        call md( $x_i, y_i, \Delta_i$ )
    else
        call mv( $x_i, y_i, \Delta_i$ )
    end if
else if  $\Delta_i = 0$  then
    call md( $x_i, y_i, \Delta_i$ )
end if
end while
finish

```

move horizontally

subroutine mh(x_i, y_i, Δ_i)

$x_i = x_i + 1$

$\Delta_i = \Delta_i + 2x_i + 1$

end sub

move diagonally

subroutine md(x_i, y_i, Δ_i)

$x_i = x_i + 1$

$y_i = y_i - 1$

$\Delta_i = \Delta_i + 2x_i - 2y_i + 2$

end sub

move vertically

subroutine mv(x_i, y_i, Δ_i)

$y_i = y_i - 1$

$\Delta_i = \Delta_i - 2y_i + 1$

end sub

The limit variable is set to zero to terminate the algorithm at the horizontal axis. This yields the circle in the first quadrant. If only a single octant is desired, then setting $\text{Limit} = \text{Integer}(R/\sqrt{2})$ yields the second octant (see Fig. 2-10). Reflection about $y = x$ then yields the first quadrant. A flowchart is given in Fig. 2-14.

