A Circle Drawing Algorithm

- The equation for a circle is:

$$x^2 + y^2 = r^2$$

- where r is the radius of the circle

- So, we can write a simple circle drawing algorithm by solving the equation for y at unit x intervals using:

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

Direct circle algorithm

- Cartesian coordinates

- Circle equation:

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

- Step along x axis from x_c - r to x_c + r and calculate

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

Figure 6-11 Circle with center coordinates (x_c, y_c) and radius r .

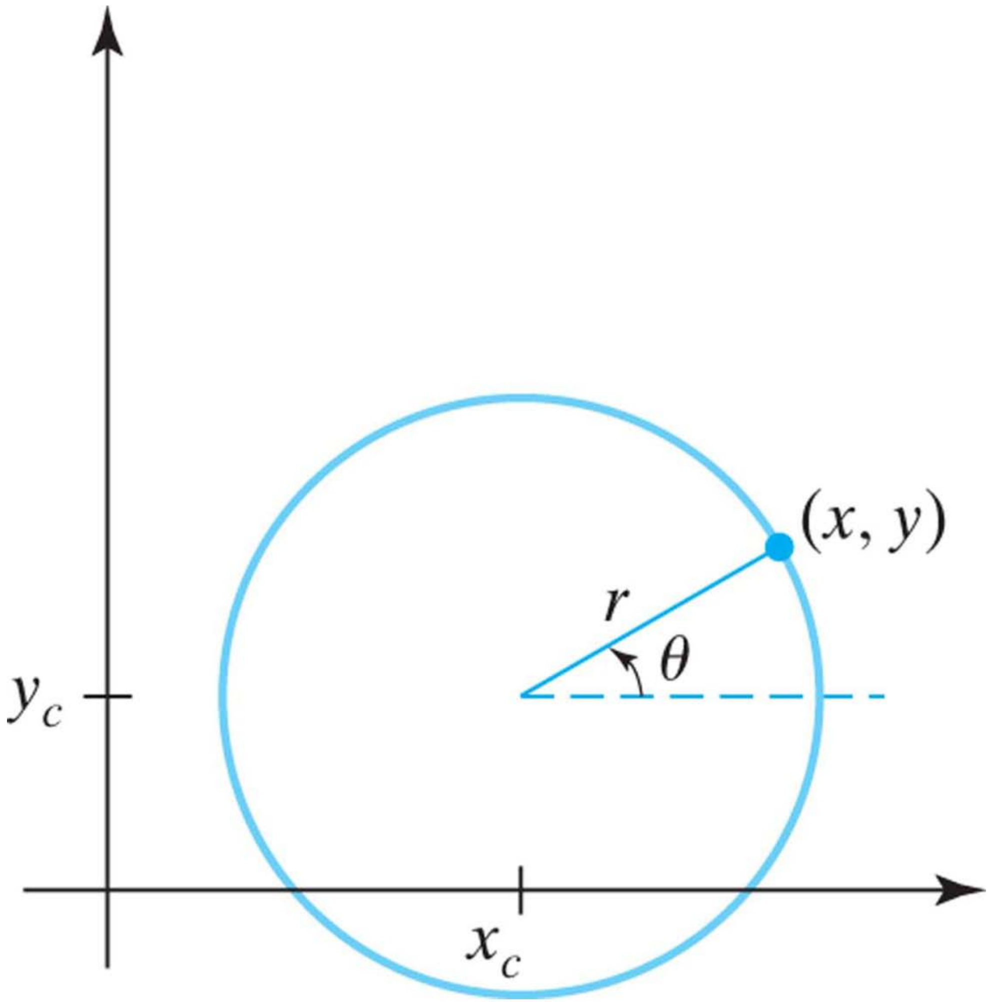
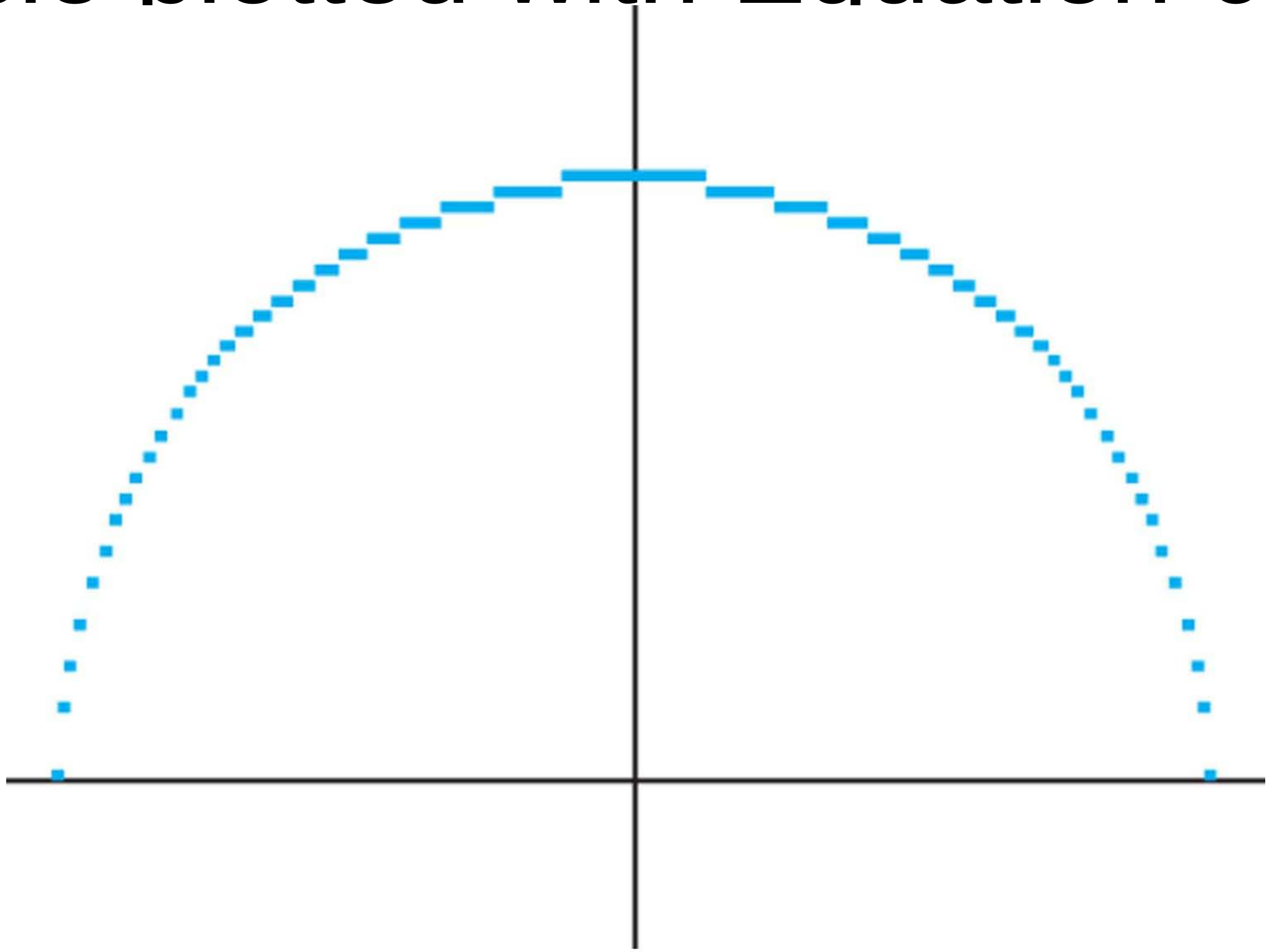
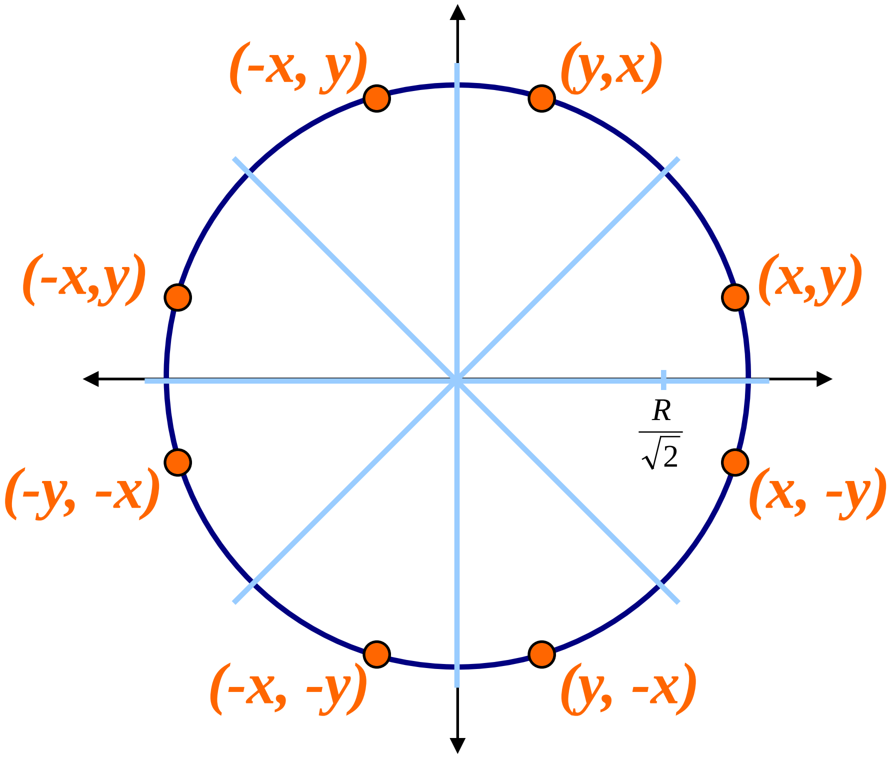


Figure 6-12 Upper half of a circle plotted with Equation 6-27



Eight-Way Symmetry

•

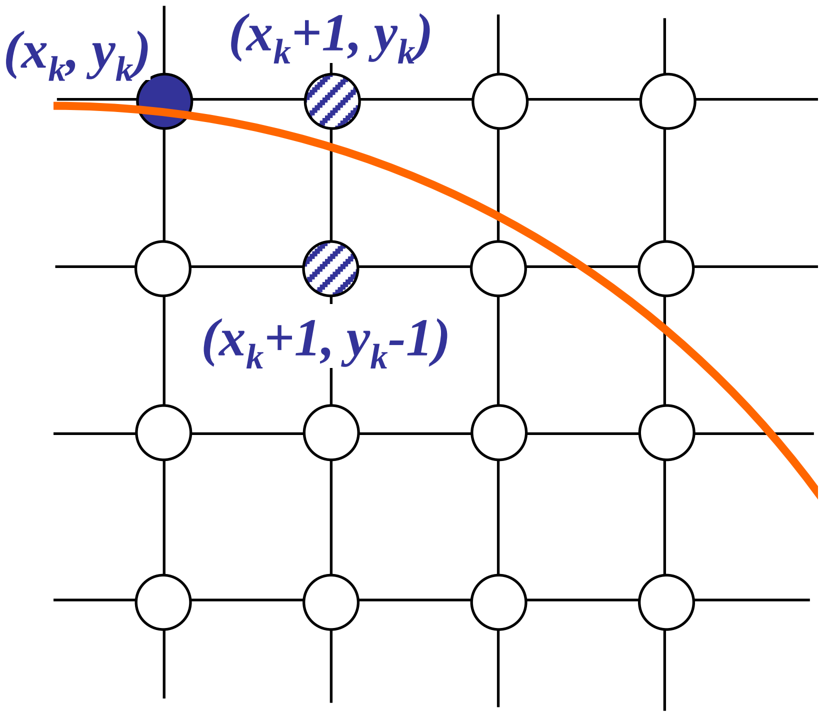


Mid-Point Circle Algorithm

In the mid-point circle algorithm we use eight-way symmetry so only ever calculate the points for the top right eighth of a circle, and then use symmetry to get the rest of the points

Mid-Point Circle Algorithm (cont…)

- Assume that we have just plotted point (x_k, y_k)
- The next point is a choice between (x_k+1, y_k) and (x_k+1, y_k-1)
- We would like to choose the point that is nearest to the actual circle



Mid-Point Circle Algorithm

- Let's re-jig the equation of the circle slightly to give us:

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

- The equation evaluates as follows:

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

- By evaluating this function at the midpoint between the candidate pixels we can make our decision

Mid-Point Circle Algorithm

- Assuming we have just plotted the pixel at (x_k,y_k) so we need to choose between (x_k+1,y_k) and (x_k+1,y_k-1)
- Our decision variable can be defined as:

$$\begin{aligned} p_k &= f_{circ}(x_k + 1, y_k - \tfrac{1}{2}) \\ &= (x_k + 1)^2 + (y_k - \tfrac{1}{2})^2 - r^2 \end{aligned}$$

- If $p_k < 0$ the midpoint is inside the circle and the pixel at y_k is closer to the circle
- Otherwise the midpoint is outside and y_k-1 is closer

Mid-Point Circle Algorithm (cont…)

- To ensure things are as efficient as possible we can do all of our calculations incrementally
- First consider:

$$p_{k+1} = f_{circ}\left(x_{k+1} + 1, y_{k+1} - \tfrac{1}{2}\right)$$

or:

$$= [(x_k + 1) + 1]^2 + \left(y_{k+1} - \tfrac{1}{2}\right)^2 - r^2$$

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

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

Mid-Point Circle Algorithm (cont…)

- The first decision variable is given as:

$$
\begin{aligned}
p_0 &= f_{circ}(1, r - \tfrac{1}{2}) \\
&= 1 + (r - \tfrac{1}{2})^2 - r^2 \\
&= \tfrac{5}{4} - r
\end{aligned}
$$

- Then if $p_k < 0$ then the next decision variable is given as:

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

- If $p_k > 0$ then the decision variable is:

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

The Mid-Point Circle Algorithm

- Input radius r and circle centre (x_c, y_c), then set the coordinates for the first point on the circumference of a circle centred on the origin as:

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

- Calculate the initial value of the decision parameter as:

$$p_0 = \tfrac{5}{4} - r$$

- Starting with $k = 0$ at each position x_k, perform the following test. If $p_k < 0$, the next point along the circle centred on $(0, 0)$ is (x_k+1, y_k) and:

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

The Mid-Point Circle Algorithm

- Otherwise the next point along the circle is (x_k+1, y_k-1) and:

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

4. Determine symmetry points in the other seven octants
5. Move each calculated pixel position (x, y) onto the circular path centred at (x_c, y_c) to plot the coordinate values:

$$x = x + x_c \qquad y = y + y_c$$

6. Repeat steps 3 to 5 until $x >= y$

Mid-Point Circle Algorithm Summary

– Eight-way symmetry can hugely reduce the work in drawing a circle

– Moving in unit steps along the x axis at each point along the circle's edge we need to choose between two possible y coordinates