

# Bresenham's algorithm for circles

Let's say we want to scan-convert a circle centered at  $(0,0)$  with an integer radius  $R$  (Figure 1). We'll see that the ideas we previously used for line scan-conversion can be made work for this task. First of all, notice that the interior of the circle is characterized by the inequality  $D(x,y) := x^2 + y^2 - R^2 < 0$ . We'll use  $D(x,y)$  to derive our decision variable.

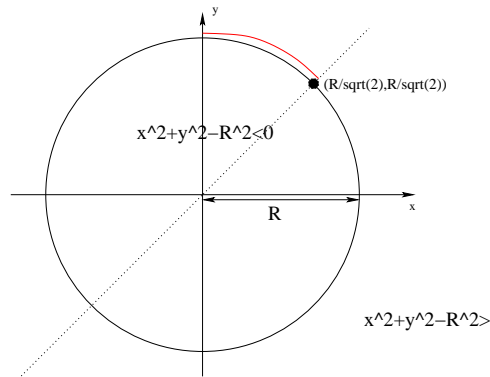


Figure 1: A circle and the description of its interior and exterior as two quadratic inequalities.

First, let's think how to plot pixels close to the 1/8 of the circle marked red in the figure. The range of the  $x$  coordinates for such pixels is from 0 to  $R/\sqrt{2}$ . We'll go over vertical scanlines through the centers of the pixels and, for each such scanline, compute the pixel on that line which is the closest to the scanline-circle intersection point (black dots in Figure 2). All such pixels will be plotted by our procedure.

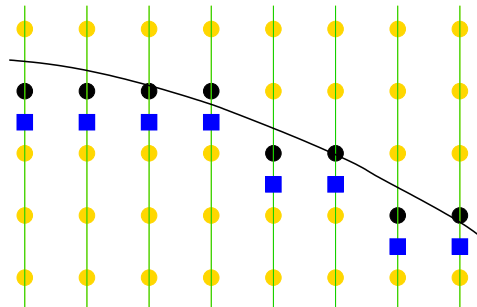


Figure 2: A circle and the description of its interior and exterior as two quadratic inequalities.

Notice that each time we move to the next scanline, the  $y$ -coordinate of the plotted point either stays the same or decreases by 1 (the slope of the circle there

is between  $-1$  and  $0$ ). To decide what needs to be done, we'll use the decision variable, which will be the value of  $D(x, y)$  evaluated at the blue square, i.e. the midpoint between the plotted pixel and the pixel immediately below.

The first pixel plotted is  $(0, R)$  and therefore the initial value of the decision variable should be

$$D(0, R - .5) = (R - .5)^2 - R^2 = .25 - R.$$

The  $y$  variable, holding the second coordinates of the plotted pixels, will be initialized to  $R$ . Let's now think what happens after a point  $(x, y)$  is plotted. First, we'll pretend that we need to move the plotted point to the right (no change in  $y$ ) and check if this keeps the decision variable negative (we don't want any blue squares outside the circle!). If  $(x, y)$  is the last plotted point, the decision variable is  $D(x, y - .5)$ . After we move to the right, it becomes  $D(x + 1, y - .5)$ . Simple arithmetic shows that it increases by  $D(x + 1, y - .5) - D(x, y - .5) = 2x + 1$ . If this increase makes it positive, we'd better move down by 1 pixel. This puts the blue square at  $(x + 1, y - 1.5)$  and means that we need to increase the decision by the previous  $2x + 1$  plus  $D(x + 1, y - 1.5) - D(x + 1, y - .5) = 2 - 2y$ .

This leads to the algorithm in Figure 3. Clearly, to make the decision variable integer, we need to scale it by a factor of 4. All the multiplications by powers of two (when updates of the decision variables are made) can be done efficiently using shifts ( $\ll$  operator in C). Eight-way symmetry is used to go from 1/8-th of the circle to the full circle.

```

y := R;
d := 1/4 - R;

for x:=0 to ceil(R/sqrt(2)) do
  plot_points(x,y);
  d += 2x+1;
  if (d>0)
    d += 2-2y;
    y--;
  plot_points(x,y) :
    plot(x,y);
    plot(x,-y);
    plot(-x,y);
    plot(-x,-y);
    plot(y,x);
    plot(-y,x);
    plot(y,-x);
    plot(-y,-x);

```

Figure 3: Bresenham's algorithm for circles.