Design of Circle Generating Algorithms

Unit1 – Lecture 5

Simple Circle Algorithms

Since the equation for a circle on radius r centered at (0,0) is

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

an obvious choice is to plot

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

for -r <= x <= r.

This works, but is inefficient because of the multiplications and square root operations. It also creates large gaps in the circle for values of x close to R (and clumping for x near 0).

A better approach, which is still inefficient but avoids the gaps is to plot

$$x = r \cos ø$$

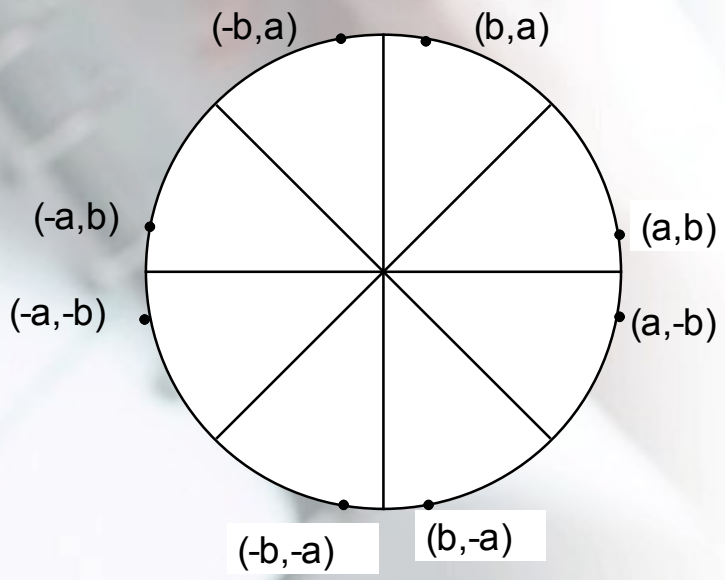
$$y = r \sin ø$$

as ø takes on values between 0 and 360 degrees.

Circle Drawing Algorithm

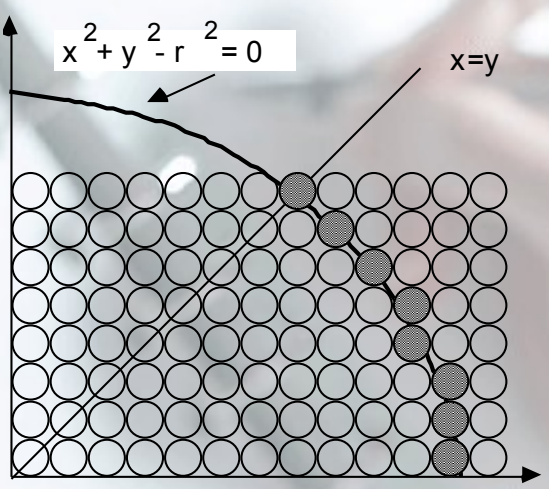
We only need to calculate the values on the border of the circle in the first octant. The other values may be determined by symmetry. Assume a circle of radius r with center at (0,0).

```
Procedure Circle_Points(x,y :Integer);
Begin
        Plot(x,y);
        Plot(y,x);
        Plot(y,-x);
        Plot(x,-y);
        Plot(-x,-y);
        Plot(-y,-x);
        Plot(-y,x);
        Plot(-x,y)
End;
```



Fast Circles

Consider only the first octant of a circle of radius r centered on the origin. We begin by plotting point (r,0) and end when x < y.



The decision at each step is whether to choose the pixel directly above the current pixel or the pixel which is above and to the left.

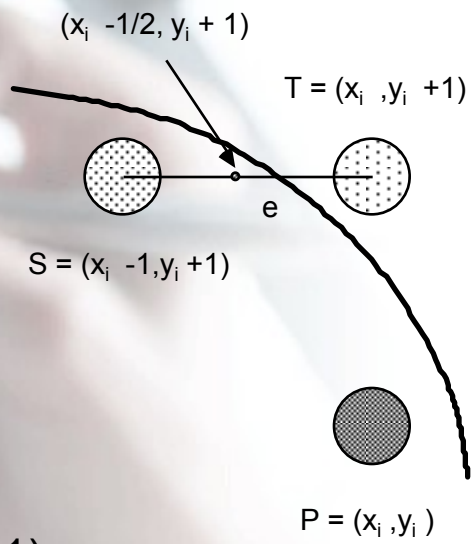
Assume $P_i = (x_i, y_i)$ is the current pixel.

$T_i = (x_i, y_i + 1)$ is the pixel directly above

$S_i = (x_i - 1, y_i + 1)$ is the pixel above and to the left.

Fast Circles - The Decision Variable

$f(x,y) = x^2 + y^2 - r^2 = 0$



$f(x_i - 1/2 + e, y_i + 1)$

$= (x_i - 1/2 + e)^2 + (y_i + 1)^2 - r^2$

$= (x_i - 1/2)^2 + (y_i+1)^2 - r^2 + 2(x_i-1/2)e + e^2$

$= f(x_i - 1/2, y_i + 1) + 2(x_i - 1/2)e + e^2 = 0$

Let $d_i = f(x_i - 1/2, y_i+1) = -2(x_i - 1/2)e - e^2$ Thus,

If $e < 0$ then $d_i > 0$ so choose point $S = (x_i - 1, y_i + 1)$.

$d_{i+1} = f(x_i - 1 - 1/2, y_i + 1 + 1) = ((x_i - 1/2) - 1)^2 + ((y_i + 1) + 1)^2 - r^2$

$= d_i - 2(x_i - 1) + 2(y_i + 1) + 1$

$= d_i + 2(y_{i+1} - x_{i+1}) + 1$

If $e >= 0$ then $d_i <= 0$ so choose point $T = (x_i, y_i + 1)$.

$d_{i+1} = f(x_i - 1/2, y_i + 1 + 1)$

$= d_i + 2y_{i+1} + 1$

Fast Circles - Decision Variable (cont.)

The initial value of d_i is

$d_0 = f(r - 1/2, 0 + 1) = (r - 1/2)^2 + 1^2 - r^2$

$= 5/4 - r$ {1-r can be used if r is an integer}

When point $S = (x_i - 1, y_i + 1)$ is chosen then

$d_{i+1} = d_i + -2x_{i+1} + 2y_{i+1} + 1$

When point $T = (x_i , y_i + 1)$ is chosen then

$d_{i+1} = d_i + 2y_{i+1} + 1$

Fast Circle Algorithm

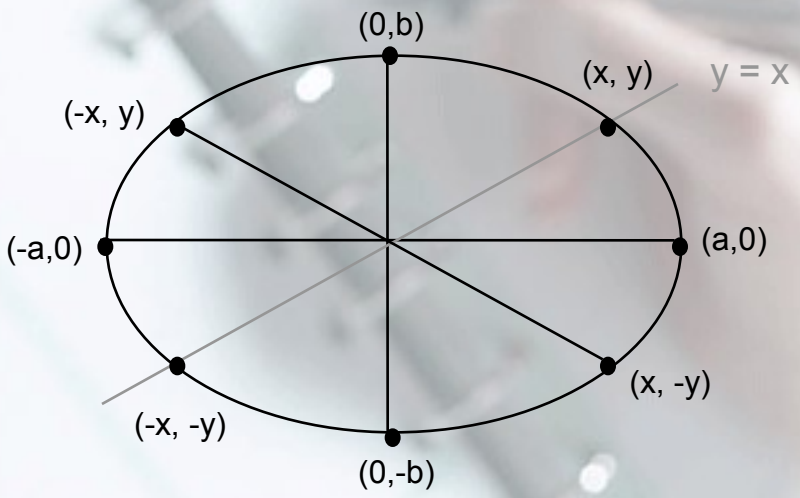
```
Begin  {Circle}
      x := r;
      y := 0;
      d := 1 - r;
      Repeat
            Circle_Points(x,y);
            y := y + 1;
            If d <= 0 Then
                  d := d + 2*y + 1
            Else
                  x := x - 1;
                  d := d + 2*(y-x) + 1
            End
      Until x < y
End;  {Circle}
```

```
Procedure Circle_Points(x,y
:Integer);
            Begin
                  Plot(x,y);
                  Plot(y,x);
                  Plot(y,-x);
                  Plot(x,-y);
                  Plot(-x,-y);
                  Plot(-y,-x);
                  Plot(-y,x);
                  Plot(-x,y)
            End;
```

Fast Ellipses

The circle algorithm can be generalized to work for an ellipse but only four way symmetry can be used.

$$F(x,y) = b^2x^2 + a^2y^2 - a^2b^2 = 0$$



All the points in one quadrant must be computed. Since Bresenham's algorithm is restricted to only one octant, the computation must occur in two stages. The changeover occurs when the point on the ellipse is reached where the tangent line has a slope of ±1. In the first quadrant, this is where the line y = x intersects the ellipses.