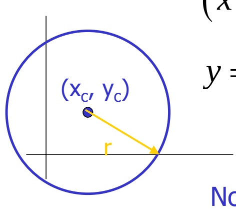
Drawing Circles and Arcs

- Similar to line drawing
  - But non-linear
- Algorithms
  - Simple equation implementation
  - Optimized (Bresenham approach)

1

Circle Equations



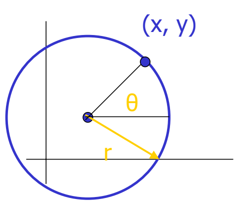
$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

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

Not a very good method, since slope changes dramatically.

2

Polar Coordinate Form



$$x = x_c + r\cos\theta$$

$$y = y_c + r\sin\theta$$

$$\Delta s = r\theta$$

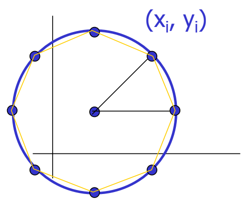
$$\Delta s \approx 1$$

$$\Delta\theta \approx \frac{1}{r}$$

Simple method: plot directly from parametric equations

3

Polygon Approximation

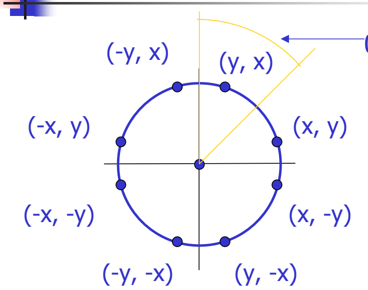


Calculate polygon vertices from polar equation; connect with lines

Can use larger Δθ, fewer trig computations

4

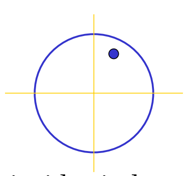
Symmetry Optimization



Calculate points for one octant; replicate in other seven

5

Bresenham's Circle Algorithm



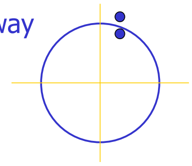
$$f_{circle}(x,y) = x^2 + y^2 - r^2$$

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

6

Circle Decision Parameter

Calculate f_{circle} for point midway between candidate pixels



$$p_k = f_{circle}\left(x_k + 1, y_k - \frac{1}{2}\right)$$

$$= (x_k + 1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2$$

$$= x_k^{\,2} + 2x_k + 1 + y_k^{\,2} - y_k + \frac{1}{4} - r^2$$

7

Calculating p_{k+1}

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

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

$$= x_k^{\,2} + 4x_k + 4 + y_{k+1}^{\,2} - y_{k+1} + \frac{1}{4} - r^2$$

8

Recurrence Relation for p_k

$$p_k = x_k^{\,2} + 2x_k + 1 + y_k^{\,2} - y_k + \frac{1}{4} - r^2$$

$$p_{k+1} = x_k^{\,2} + 4x_k + 4 + y_{k+1}^{\,2} - y_{k+1} + \frac{1}{4} - r^2$$

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

$$= p_k + 2x_{k+1} + 1 + \left(y_{k+1}^{\,2} - y_k^{\,2}\right) - \left(y_{k+1} - y_k\right)$$

9

One Last Term …

$$\left(y_{k+1}^2 - y_k^2\right) - \left(y_{k+1} - y_k\right) = ?$$

If $y_{k+1} = y_k$, then:

$$\left(y_k^2 - y_k^2\right) - \left(y_k - y_k\right) = 0$$

If $y_{k+1} = y_k - 1$, then:

$$\left(\left(y_k - 1\right)^2 - y_k^2\right) - \left(\left(y_k - 1\right) - y_k\right) =$$

$$\left(y_k^2 - 2y_k + 1 - y_k^2\right) - \left(-1\right) =$$

$$-2y_k + 2 = -2\left(y_k - 1\right)$$

10

Initial Values

$$\left(x_0, y_0\right) = \left(0, r\right)$$

$$p_0 = f_{circle}\left(0+1, r - \frac{1}{2}\right)$$

$$= 1 + \left(r - \frac{1}{2}\right)^2 - r^2$$

$$= 1 + r^2 - r + \frac{1}{4} - r^2$$

$$= \frac{5}{4} - r \approx 1 - r$$

Round to nearest integer. Why?

11

Bresenham Algorithm Summary

At each point:

If $p_k < 0$: Plot$\left(x_k + 1, y_k\right)$

$$p_{k+1} = p_k + 2\left(x_k + 1\right) + 1$$

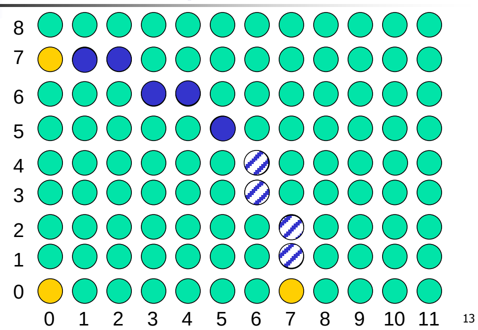
If $p_k >= 0$: Plot$\left(x_k + 1, y_k - 1\right)$

$$p_{k+1} = p_k + 2\left(x_k + 1\right) + 1 - 2\left(y_k - 1\right)$$

12

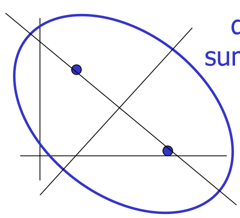
Circle Example 1

r = 7



13

General Ellipse Equation

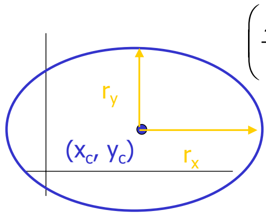


In general, ellipse described by constant sum of distances from foci

Difficult to solve equations and plot points (use parametric polar form?)

14

Special Case Ellipses



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

$$x = x_c + r_x\cos\theta$$
$$y = y_c + r_y\sin\theta$$

Modified midpoint algorithm possible

15

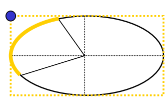
Drawing Ellipses

- General ellipses
  - Polygon approximation
    - Rotation (later)
- Aligned axes
  - Constrained (no rotation)
  - Midpoint algorithm

16

Drawing an Arc (Qt)

```
void QPainter::drawArc
  (int x,
   int y,
   int w,
   int h,
   int a,
   int alen);
void QPainter::drawArc
  (const QRect& r,
   int a,
   int alen);
```



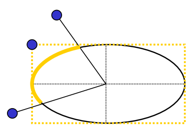
Angles in 1/16-degree units; applied to circle, then deformed; CCW from 3 o'clock

17

Drawing an Arc (MS Windows)

```
BOOL CDC::Arc
  (LPCRECT lpRect,
   POINT ptStart,
   POINT ptEnd );

BOOL CDC::Ellipse
  (LPCRECT lpRect);
```



Arc angles determined by additional points; CCW drawing

18