Faculty of German Engineering and Industrial Management Education - FDIBA

Introduction to Computer Graphics

Drawing Lines

Assoc. Prof. Stoyan Maleshkov

Technical University of Sofia

Line Drawing

- Draw a line on a raster screen between two
- points.
- What's wrong with the statement of the
- problem?
  - it doesn't say anything about which points are allowed as endpoints
  - it doesn't give a clear meaning to "draw"
  - it doesn't say what constitutes a "line" in the raster world
  - it doesn't say how to measure the success of a proposed algorithm

Line Drawing

- **Problem Statement**
- Given two points P and Q in the plane, both with integer coordinates, determine which pixels on a raster screen should be on in order to make a picture of a unit-width line segment starting at P and ending at Q.

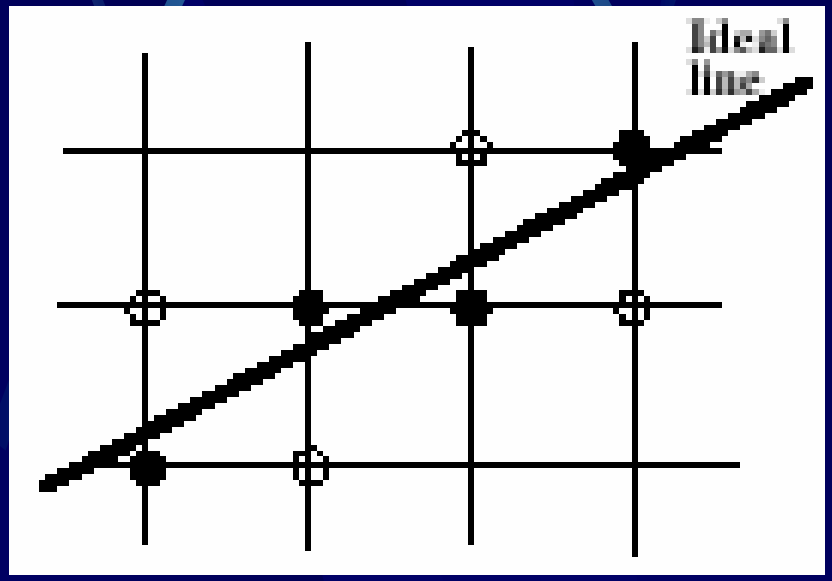
Basic Problem

- Scan Conversion or *rasterization*
- Due to the scanning nature of raster displays
- Transforming the continuous into this discrete (sampling)
- Algorithms are fundamental to both 2-D and 3-D computer graphics
- **Most of the early scan-conversion algorithms developed for plotters can be attributed to one man, Jack Bresenham**

Drawing Lines

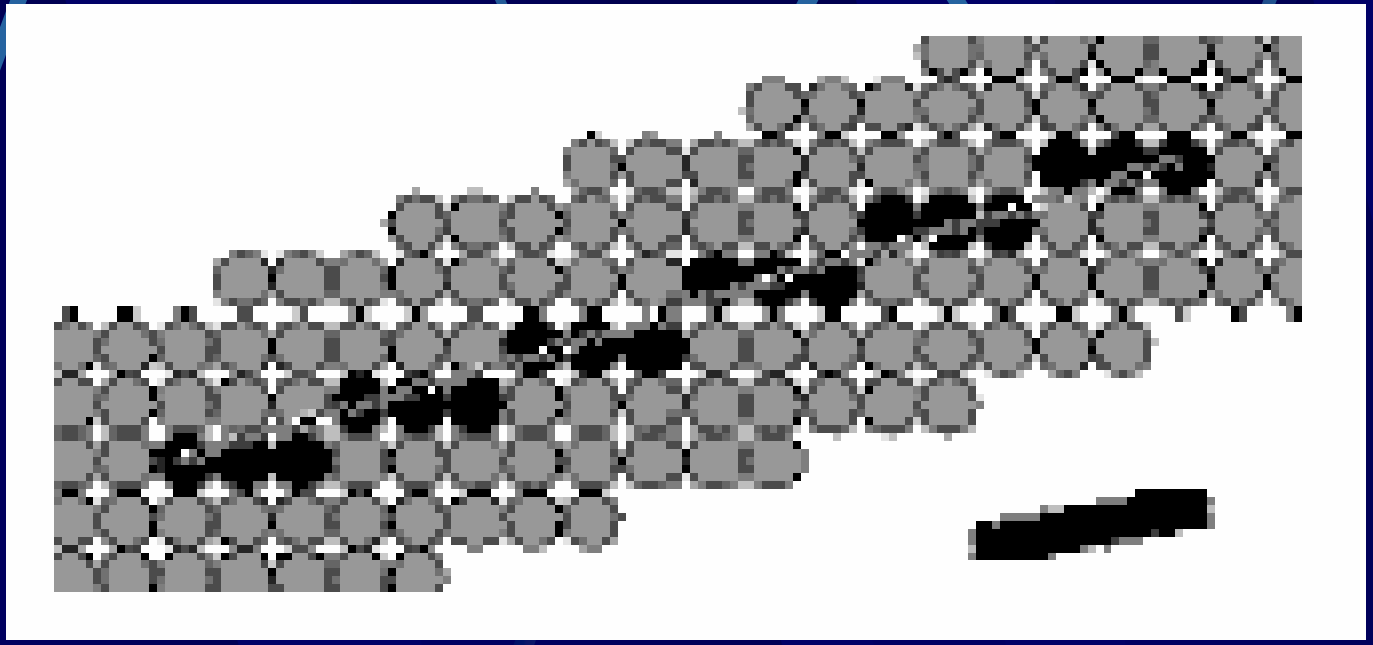
For horizontal, vertical and diagonal lines all pixels lie on the ideal line: special case

• For lines at arbitrary angle, pick pixels closest to the ideal line (Bresenham's = midpoint "scan conversion" algorithm)



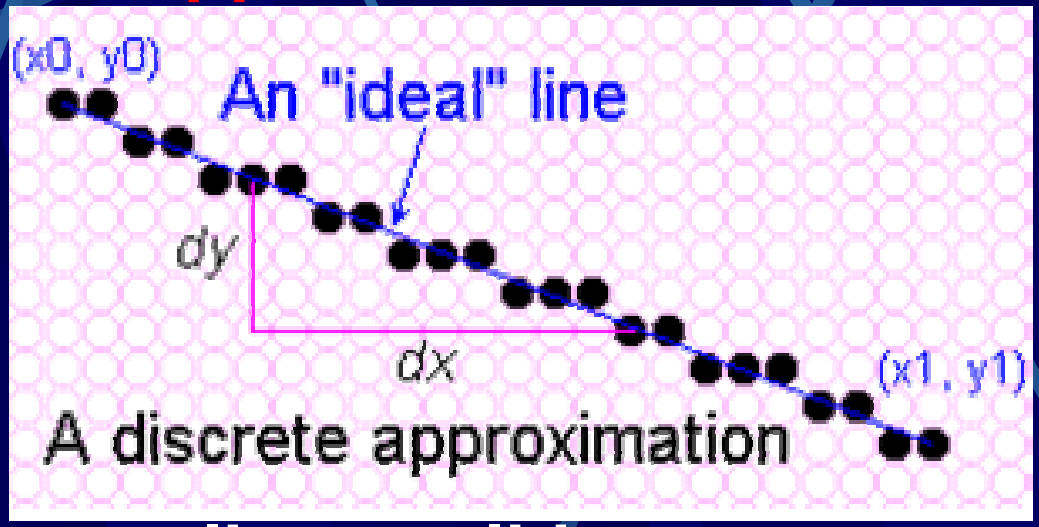
Drawing Lines (2)

- For thick lines, use multiple pixels in each column or fill a rotated rectangle



Quest for the *Ideal Line*

- **The best we can do is a discrete approximation of an ideal line**



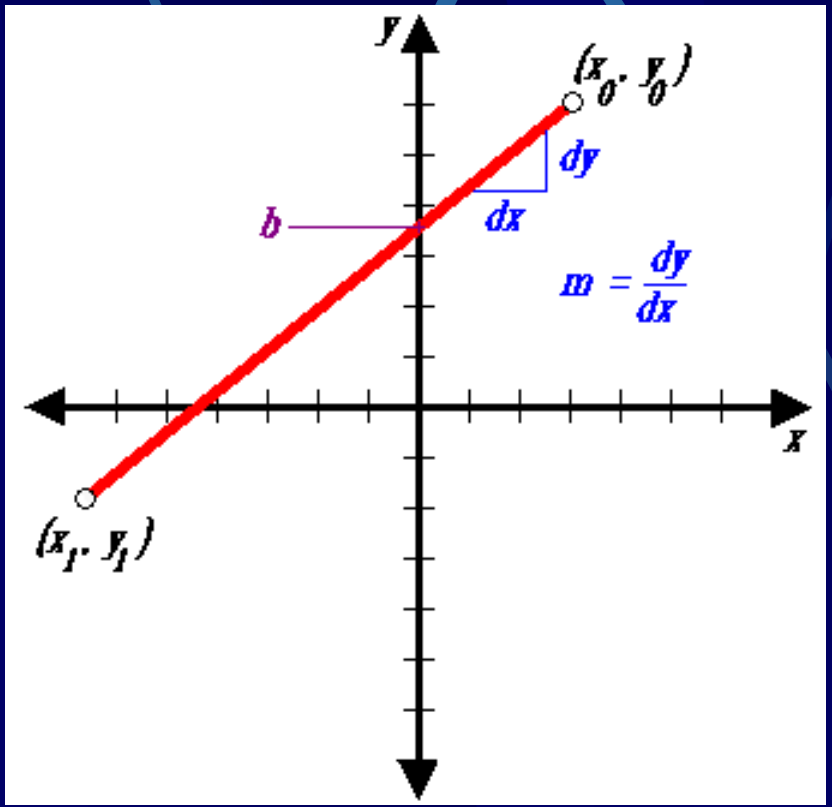
Important line qualities:

- Continuous appearance
- Uniform thickness and brightness
- Turn on the pixels nearest the ideal line
- How fast is the line generated

Simple Line

- **Based on the simple *slope-intercept algorithm* from algebra**

$$y = m x + b$$



The Basic Algorithm:

- Find the equation of the line that connects the two points P and Q.
- Starting with the leftmost point P, increment x_i by 1 to calculate $y_i = Ax_i + B$,
  where A = slope, B = y intercept.
- Intensify the pixel at $(x_i, \text{Round}(y_i))$,
  where $\text{Round}(y_i) = \text{Floor}(0.5 + y_i)$

Simple Line Algorithm

```
public void lineSimple(int x0, int y0, int x1,
    int y1, Color color)
{ int pix = color.getRGB();
   int dx = x1 - x0; int dy = y1 - y0;
    raster.setPixel(pix, x0, y0);
if (dx != 0) {
   float m = (float) dy / (float) dx;
   float b = y0 - m*x0;
   dx = (x1 > x0) ? 1 : -1;
while (x0 != x1) {
    x0 += dx;
   y0 = Math.round(m*x0 + b);
   raster.setPixel(pix, x0, y0); } } }
```

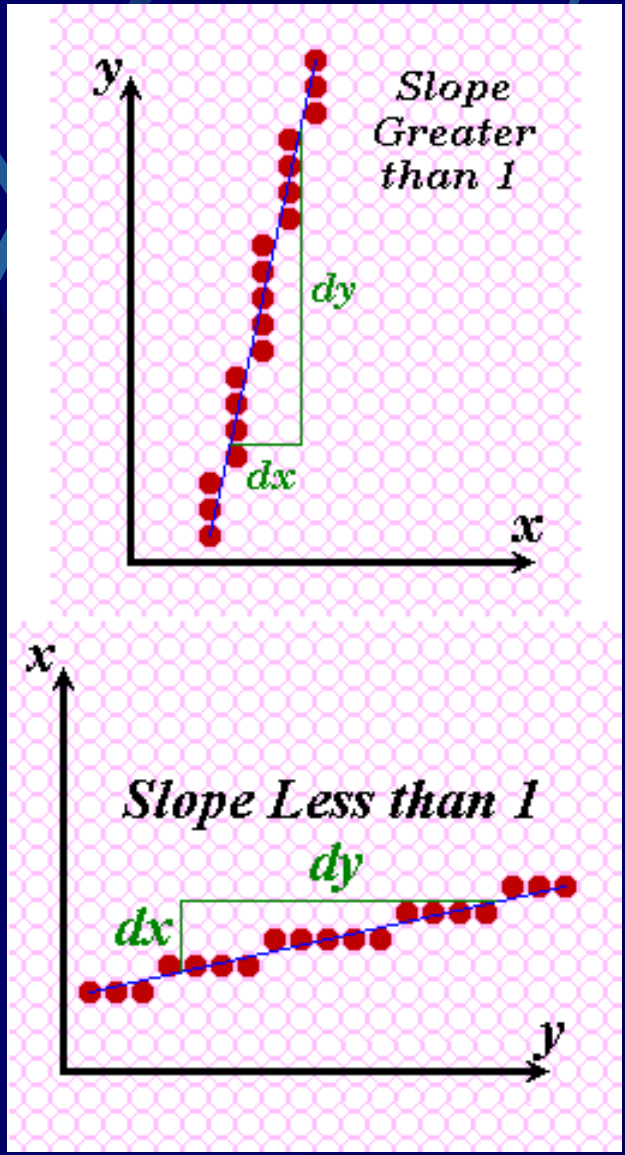
Simple Line Algorithm

- Demonstration

Improved Line Algorithm

- **Problem:**

lineSimple( ) does not give satisfactory results for slopes > 1



Solution: symmetry

```
if (Math.abs(dx) > Math.abs(dy)) { // slope < 1
// Draw the line incrementing x0  …
}
} else if (dy != 0) { // slope >= 1
float m = (float) dx / (float) dy; // compute
   slope
float b = x0 - m*y0;
dy = (dy < 0) ? -1 : 1;
while (y0 != y1) {
y0 += dy;
raster.setPixel(pix, Math.round(m*y0 + b),
   y0); }
```

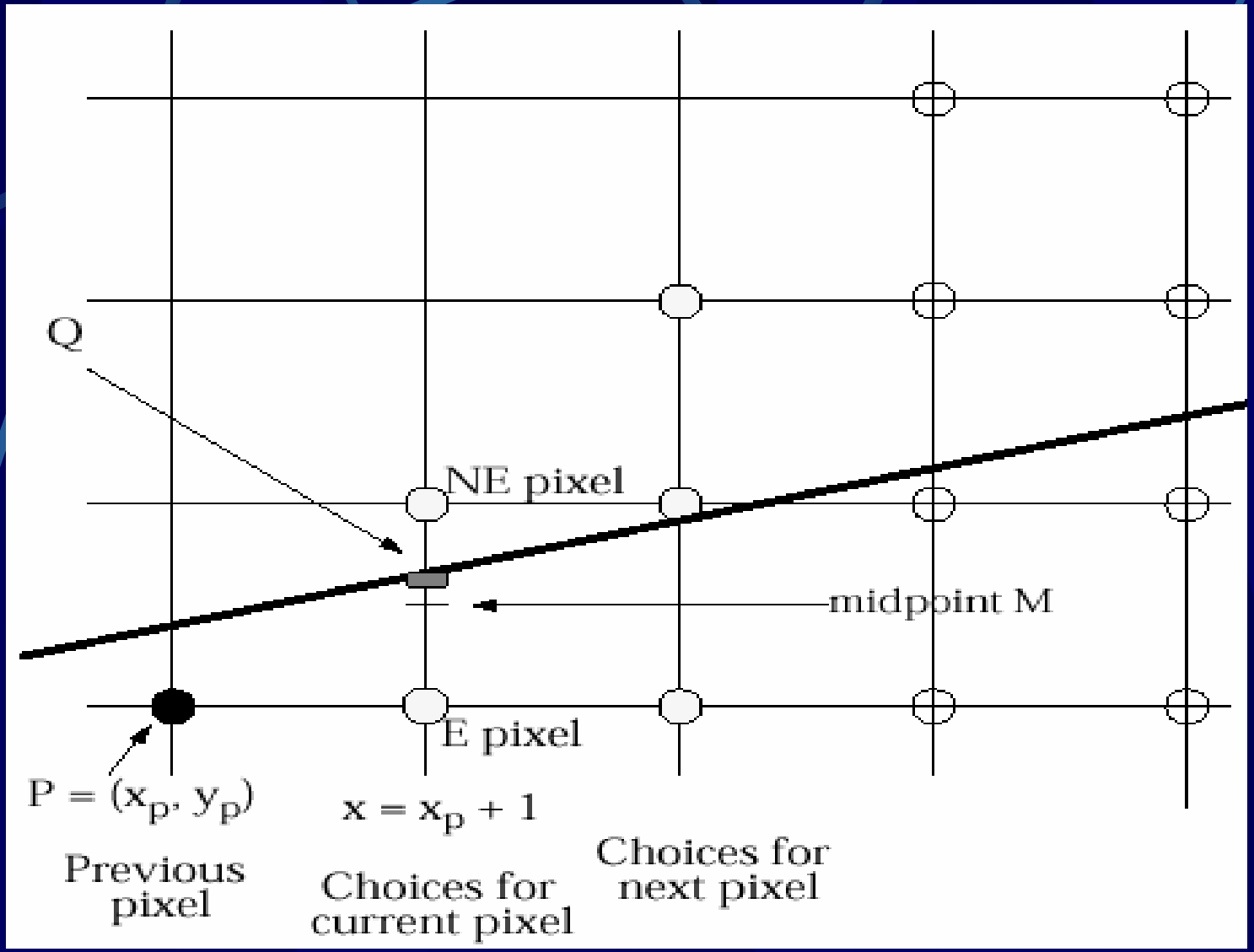
Improved Line Algorithm

- Demonstration

Optimized Line Algorithm

- Perform all calculations with **integer** variables
- Rewrite the algorithm, that it use only addition
- **Result: Bresenham Algorithm**

Bresenham Algorithm

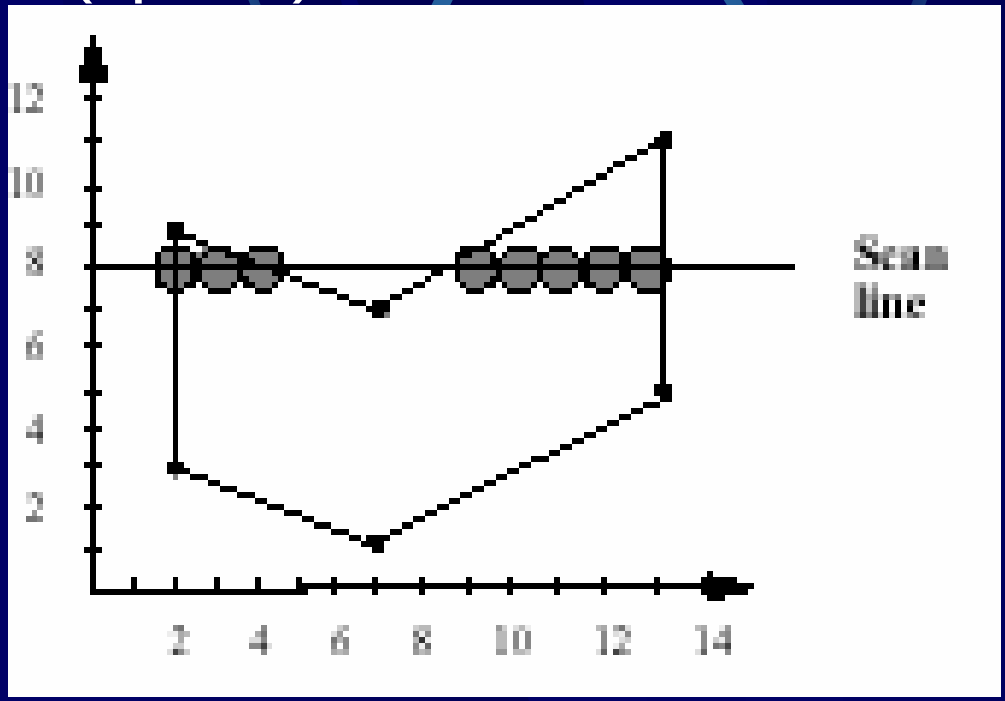


Benchmarking

- Demonstration

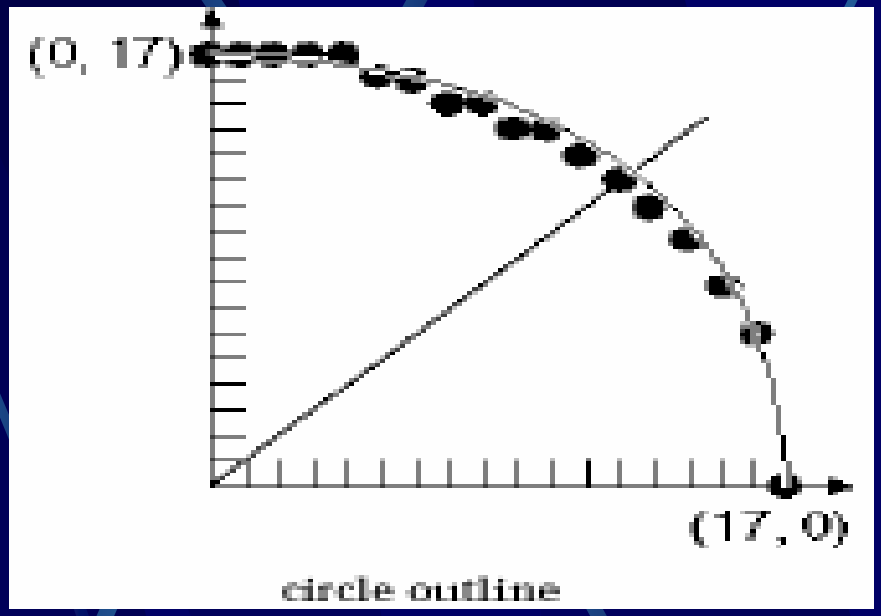
Drawing Filled Polygons

1. Find intersection of scanline with polygon edges
2. Sort intersections by increasing x
3. Fill the polygon between pairs of intersections (spans)

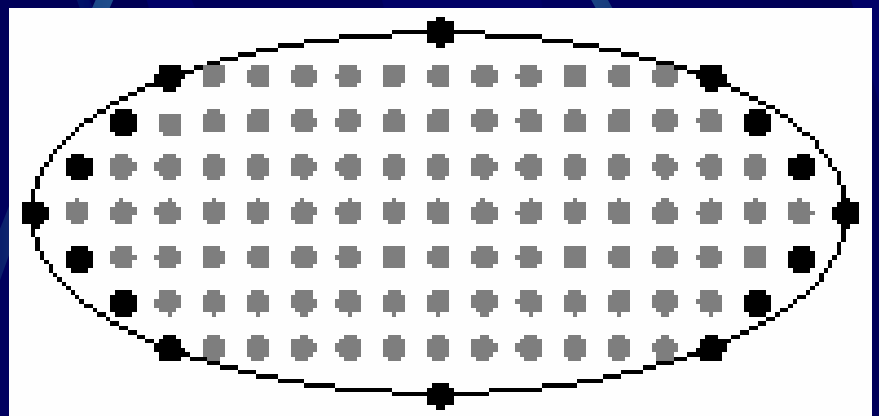


Drawing Circles and Ellipses

- Use symmetry



circle outline


filled ellipse

Thickness Attribute

- Thick circle drawn by tracing a rectangular pen