

Recommended reading

- 1) For general discussions on graphics system, read Chapters 1 and 4;
- 2) for line drawing algorithms, read Section 3.2;
- 3) For Mandelbrot fractals, read Section 20.4;
- 4) For area filling, read Section 3.6;
- 5) For interaction devices and interface design, read Chapter 8.

Area filling

Area filling is to fill in a 2D region with a specified solid color or a pattern. There are two approaches to this problem: *scanline conversion* and *boundary fill*.

Scanline conversion is suitable for regions with their boundaries defined by polygons or other regular curves. We first consider a polygon region.

The two main steps are

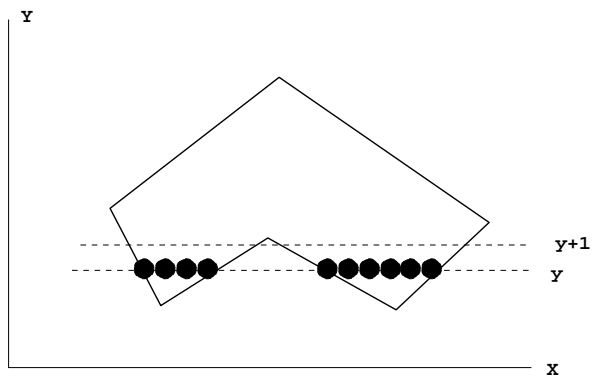
- 1) The intervals on each scanline that fall inside the polygon are identified.
- 2) The pixels in internal intervals are colored.

Scanline coherence of a polygon is used to speed up computing the intersections between a scanline and a polygon. In most cases, a new intersection point can be obtained by simple increments from an intersection point on the preceding scanline.

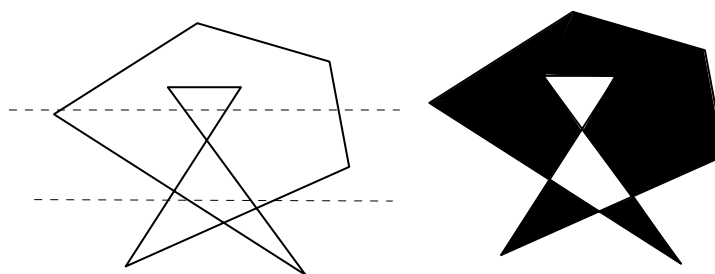
The *parity* of intersections on a scanline is used to identify overlapping intervals. *There are always an even number of intersections between a scanline and a closed polygon.* However, parity check works only for simple (i.e. *non-self-intersecting*) polygons.

Self-intersecting 2D polygons can arise as projections of 3D non-planar polygons. The interior of such 2D polygons can be defined by nonzero winding

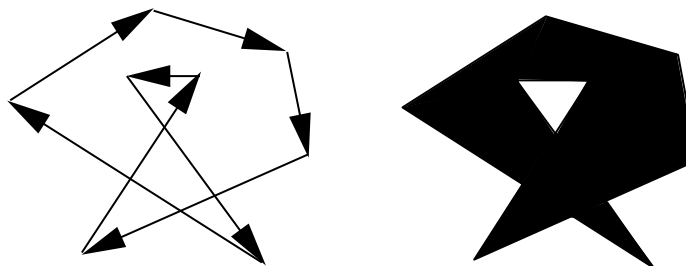
numbers.



Scanline conversion for area-filling



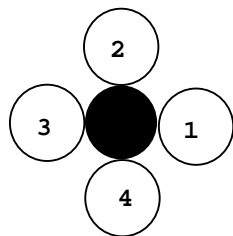
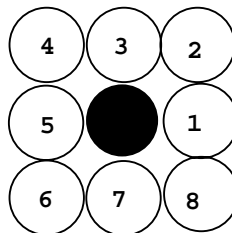
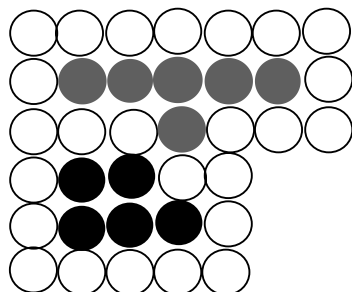
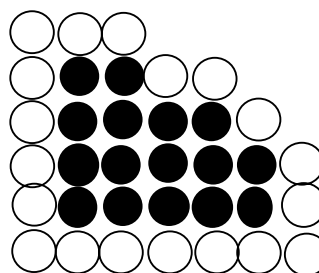
Odd Parity Check



Non-Zero Winding Number

Boundary filling

Boundary filling is suitable for regions with complex boundaries, such as boundaries drawn by hand using a mouse, which are usually represented as a sequence of pixels.

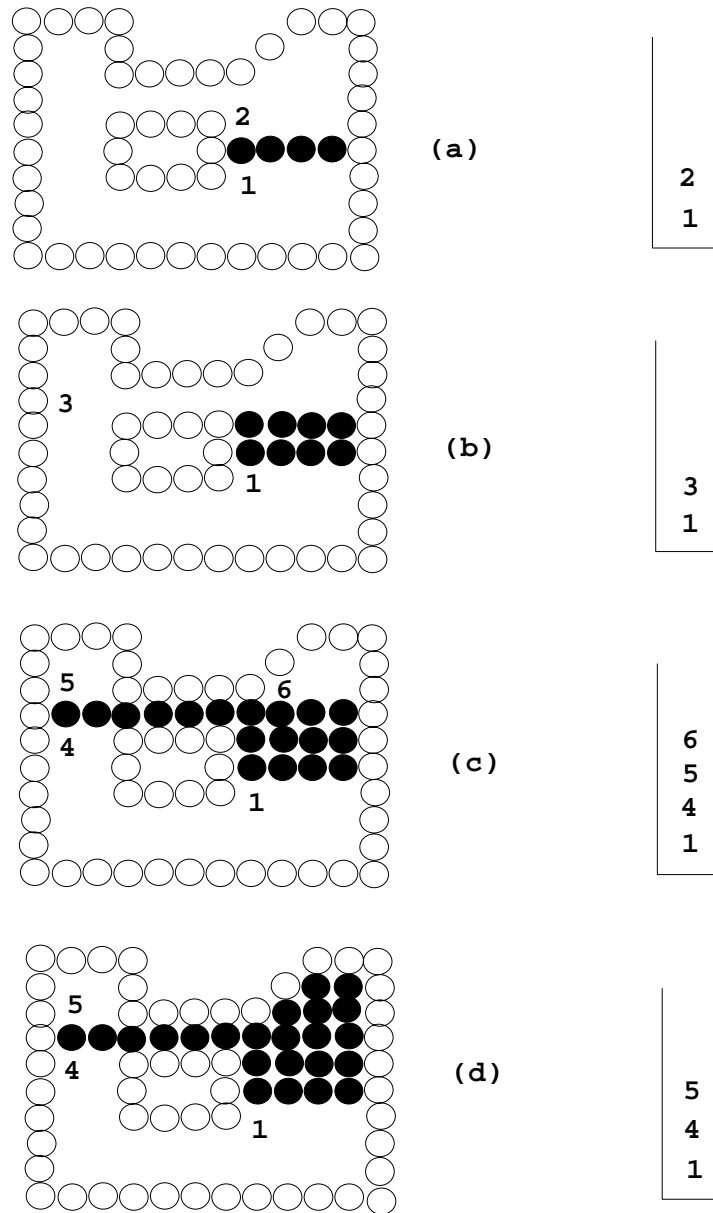
**4-Connectivity****8-Connectivity****(A)****(B)**

Which connectivity should be used?

The main steps are:

- 1) A seed pixel inside the region is selected as the current pixel and set to the specified color.
- 2) The current pixel's neighboring interior pixels that are yet to be processed are set to the specified color.
- 3) Set all the above neighboring pixels, in turn, as the current pixel, and repeat step 2.

Neighbor relationship can be 4-connected or 8 connected. Each has its advantage and limitations.



A recursive implementation

Interior pixels on a scanline can be organized into *spans*, or groups of consecutive interior pixels. Each span is rendered from the leftmost starting point.

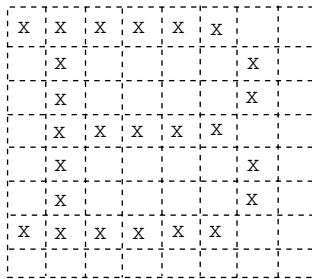
When a span is rendered, the starting pixels of its neighboring and unprocessed spans are identified and pushed into a stack. All spans are processed in a recursive depth-first order.

Character Generation

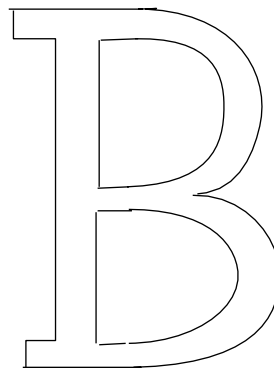
There are two ways to represent characters: **bitmap fonts** and **outline fonts**.

Bitmap fonts are stored as a fixed 2D array of color values. Copying a character into the frame buffer is fast, but fonts of different sizes need different bitmaps.

The outline font uses line segments and curves to define the outline of a character. It is relatively slow to fill in the interior of the character; usually scanline filling is used. But fonts of different sizes can be derived easily by scaling up or down the same representation.



Bitmap font

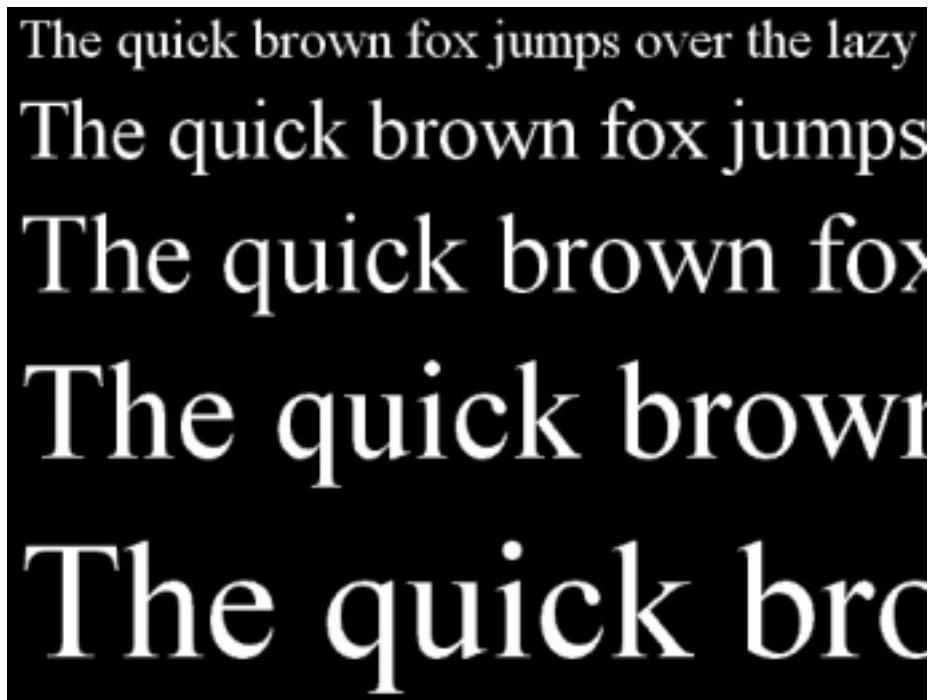


Outline font

The next two images show the screen display of bitmap fonts and outline fonts, respectively.



The quick brown fox jumps over the lazy
The quick brown fox jumps
The quick brown fox
The quick brown
The quick bro



The quick brown fox jumps over the lazy
The quick brown fox jumps
The quick brown fox
The quick brown
The quick bro