Windows will then ignore the background color and not fill in the gaps. You can obtain the current background mode (either TRANSPARENT or OPAQUE) by calling *GetBkMode*.

Drawing Modes

The appearance of lines drawn on the display is also affected by the drawing mode defined in the device context. Imagine drawing a line that has a color based not only on the color of the pen but also on the color of the display area where the line is drawn. Imagine a way in which you could use the same pen to draw a black line on a white surface and a white line on a black surface without knowing what color the surface is. Could such a facility be useful to you? It's made possible by the drawing mode.

When Windows uses a pen to draw a line, it actually performs a bitwise Boolean operation between the pixels of the pen and the pixels of the destination display surface, where the pixels determine the color of the pen and display surface. Performing a bitwise Boolean operation with pixels is called a "raster operation," or "ROP." Because drawing a line involves only two pixel patterns (the pen and the destination), the Boolean operation is called a "binary raster operation," or "ROP2." Windows defines 16 ROP2 codes that indicate how Windows combines the pen pixels and the destination pixels. In the default device context, the drawing mode is defined as R2_COPYPEN, meaning that Windows simply copies the pixels of the pen to the destination, which is how we normally think about pens. There are 15 other ROP2 codes.

Where do these 16 different ROP2 codes come from? For illustrative purposes, let's assume a monochrome system that uses 1 bit per pixel. The destination color (the color of the window's client area) can be either black (which we'll represent by a 0 pixel) or white (represented by a 1 pixel). The pen also can be either black or white. There are four combinations of using a black or white pen to draw on a black or white destination: a white pen on a white destination, a white pen on a black destination, a black pen on a white destination, and a black pen on a black destination.

What is the color of the destination after you draw with the pen? One possibility is that the line is always drawn as black regardless of the pen color or the destination color. This drawing mode is indicated by the ROP2 code R2_BLACK. Another possibility is that the line is drawn as black except when both the pen and destination are black, in which case the line is drawn as white. Although this might be a little strange, Windows has a name for it. The drawing mode is called R2_NOTMERGEPEN. Windows performs a bitwise OR operation on the destination pixels and the pen pixels and then inverts the result.

The table on the facing page shows all 16 ROP2 drawing modes. The table indicates how the pen (P) and destination (D) colors are combined for the result. The column labeled "Boolean Operation" uses C notation to show how the destination pixels and pen pixels are combined.

Pen (P):	***1***	***1***	***0***	***0***	***Boolean***	
Destination (D):	***1***	***0***	***1***	***0***	***Operation***	***Drawing Mode***
Results:	0	0	0	0	0	R2_BLACK
	0	0	0	1	~(P ¦ D)	R2_NOTMERGEPEN
	0	0	1	0	~P & D	R2_MASKNOTPEN
	0	0	1	1	~P	R2_NOTCOPYPEN
	0	1	0	0	P & ~D	R2_MASKPENNOT
	0	1	0	1	~D	R2_NOT
	0	1	1	0	P ^ D	R2_XORPEN
	0	1	1	1	~(P & D)	R2_NOTMASKPEN
	1	0	0	0	P & D	R2_MASKPEN
	1	0	0	1	~(P ^ D)	R2_NOTXORPEN
	1	0	1	0	D	R2_NOP
	1	0	1	1	~P ¦ D	R2_MERGENOTPEN
	1	1	0	0	P	R2_COPYPEN (default)
	1	1	0	1	P ¦ ~D	R2_MERGEPENNOT
	1	1	1	0	P ¦ D	R2_MERGEPEN
	1	1	1	1	1	R2_WHITE

You can set a new drawing mode for the device context by calling

```
SetROP2 (hdc, iDrawMode) ;
```

The *iDrawMode* argument is one of the values listed in the "Drawing Mode" column of the table. You can obtain the current drawing mode by using the function:

```
iDrawMode = GetROP2 (hdc) ;
```

The device context default is R2_COPYPEN, which simply transfers the pen color to the destination. The R2_NOTCOPYPEN mode draws white if the pen color is black and black if the pen color is white. The R2_BLACK mode always draws black, regardless of the color of the pen or the background. Likewise, the R2_WHITE mode always draws white. The R2_NOP mode is a "no operation." It leaves the destination unchanged.

We've been examining the drawing mode in the context of a monochrome system. Most systems are color, however. On color systems Windows performs the bitwise operation of the drawing mode for each color bit of the pen and destination pixels and again uses the 16 ROP2 codes described in the previous table. The R2_NOT drawing mode always inverts the destination color to determine the color of the line, regardless of the color of the pen. For example, a line drawn on a cyan destination will appear as magenta. The R2_NOT mode always results in a visible pen except if the pen is drawn on a medium gray background. I'll demonstrate the use of the R2_NOT drawing mode in the BLOKOUT programs in Chapter 7.

DRAWING FILLED AREAS

The next step up from drawing lines is filling enclosed areas. Windows' seven functions for drawing filled areas with borders are listed in the table below.

Function	***Figure***
Rectangle	Rectangle with square corners
Ellipse	Ellipse
RoundRect	Rectangle with rounded corners
Chord	Arc on the circumference of an ellipse with endpoints connected by a chord
Pie	Pie wedge defined by the circumference of an ellipse
Polygon	Multisided figure
PolyPolygon	Multiple multisided figures

Windows draws the outline of the figure with the current pen selected in the device context. The current background mode, background color, and drawing mode are all used for this outline, just as if Windows were drawing a line. Everything we learned about lines also applies to the borders around these figures.

The figure is filled with the current brush selected in the device context. By default, this is the stock object called WHITE_BRUSH, which means that the interior will be drawn as white. Windows defines six stock brushes: WHITE_BRUSH, LTGRAY_BRUSH, GRAY_ BRUSH, DKGRAY_BRUSH, BLACK_BRUSH, and NULL_BRUSH (or HOLLOW_BRUSH). You can select one of the stock brushes into the device context the same way you select a stock pen. Windows defines HBRUSH to be a handle to a brush, so you can first define a variable for the brush handle:

```
HBRUSH hBrush ;
```

You can get the handle to the GRAY_BRUSH by calling *GetStockObject*:

```
hBrush = GetStockObject (GRAY_BRUSH) ;
```

You can select it into the device context by calling *SelectObject*:

```
SelectObject (hdc, hBrush) ;
```