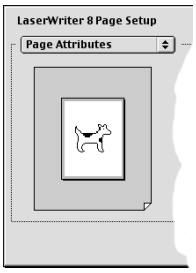
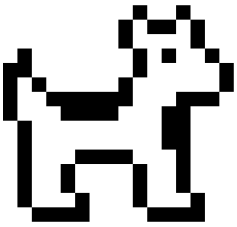


Adventures with Clarus, the Dogcow
or
How Computer Images are Displayed

http://developer.apple.com/products/techsupport/dogcow/index.html

Introduction—If you've ever used a printer driver made by Apple, you undoubtedly have been introduced to Clarus *(not "Claris")* when you opened the Page Setup window. Clarus is the cute little "moof" that shows you how your printing will be oriented on the page before you print it.

In this document, I'll use the Clarus graphic to show how computer images are displayed and the difference between ***raster*** and ***vector*** based presentations. But first, let's reduce Clarus to a tiny little graphic about 1/4 inch square.



Using Pixels—Now, let's see how an image is displayed using pixels.

When we "blow up" that little square, the "dots" in the image can be seen quite clearly, as shown at the left. Each "dot" is called a ***pixel*** and is about 1/72nd of an inch wide and high on a 14 inch, 640x480 screen display.

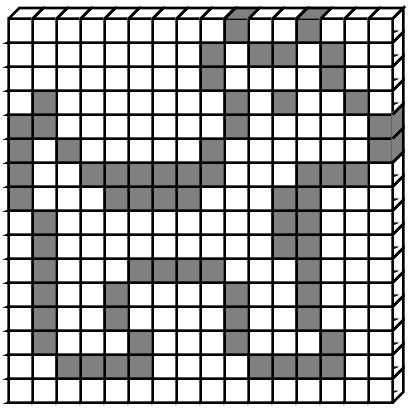
We have used 256 pixels (16x16) in our graphic. As you can see, some pixels are white and some are black.

The term "640x480" means the computer monitor screen itself is made up of that many pixels—307,200 little dots. Newer Macintoshes can display a lot more pixels (up to 1280x960) but, the pixels are generally physically smaller than 1/72nd of an inch because even the newest monitors are seldom larger than 21 inches.

Raster Based—Your computer screen is called a ***raster-based*** display. The "electron gun" in your monitor starts at the top-left corner of the screen and moves to the right to paint a thin line across the top. Then, it quickly returns to a point just below the top-left corner, and paints another thin line. This process continues until the screen is completely filled with thin horizontal lines; the physical image produced is commonly referred to as a ***raster***.

It takes a lot of these thin lines to produce a pixel, but we still consider the pixel to be the smallest element of the "picture" your computer stores and the monitor draws.

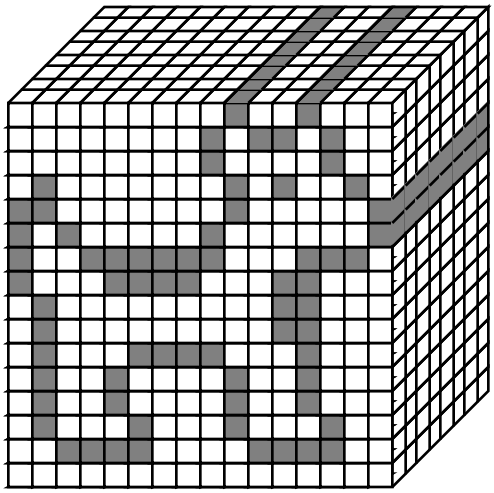
Bits and Bytes—Here is another way to look at our picture, as if it were made from a series of building blocks, each block representing a pixel. We'll start with only two colored blocks, black and white.



Your computer uses a ***bit*** (a binary-digit) to represent two states, 0 and 1. In this case, we can assign "0" to the color black and "1" to the color white.

In this way, we can store the whole image in 256 bits (16x16) or just ***thirty-two bytes***. In your Mac, a ***byte*** consists of eight bits and is the smallest storage element used to define the amount of memory or disk storage space a file occupies.

Adding Color or Grayscale—Now, let's make our set of building blocks larger so we can add some colors or shades of gray to our picture.



If each pixel is to represent a color or grayscale, it must use more than one building block. The common **GIF** file format uses up to 256 colors or shades of gray, and the eight bits necessary to represent each pixel would stack-up to look something like this. Our tiny little 1/4" square Clarus must now occupy a minimum of 2048 bits or ***256 bytes*** for a 16x16 pixel drawing.

Adding More Grayscale—Up to 256 shades of gray are enough to faithfully reproduce grayscale. In fact, less than half that number of shades are generally all that are needed for excellent black-and-white reproduction.

Adding More Color—Color, however, is another thing. The human eye is capable of seeing ***millions of colors!*** Modern computers have devised ways of showing those colors and all it takes is more—lots more memory and disk space to display and store them. The **TIFF** and **JPEG** file formats can accept up to millions of colors.

In summary, if our little 16x16 pixel example were to display the same sized drawing in

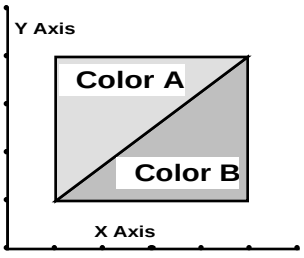
- 2 colors (B&W), each pixel would occupy 1 bit of memory or 32 bytes overall.
- 256 colors, each pixel would occupy 8 bits or 1 byte of memory or 256 bytes overall.
- 65 thousand colors, each pixel would occupy 16 bits or 2 bytes of memory or 512 bytes overall.
- 16 million colors, each pixel would occupy 24 bits or 3 bytes of memory or 768 bytes overall.

If I attempted to modify the drawing above to show pixels containing millions of colors, I'd have to make a drawing with 24 building blocks behind each pixel going out behind the graphic.

Vector Based—A vector is a line drawn from one point to another. If you think about it, that's the way we draw–by drawing groups of short and long lines that represent something.

Since a computer is very fast at arithmetic, it can quickly make mathematical representations of the length and position of the lines you draw, and store them for display.

Here is a simple example of a vector graphic. It is a flag made from two triangles and two colors. The computer simply calculates and stores the position of the lines, their widths, their beginning and end points, and the colors of the lines and the fill colors between them.



We learned at the beginning of this document that your computer can only display raster-based images. This means it must also spend some time making a raster-based image so you can see what you are doing.

Why Vector-Based Graphics?—The important things about vector-based images are storage space and quality of presentation.

- **Storage Space** is generally ***quite small*** because the computer only records the equations for the lines (in this example, five equations) and the colors of the lines and fills.
- **Quality** is ***extremely high*** because, if the image is enlarged, the pixels don't get enlarged and produce a "jagged" graphic—the symbol of poor quality! You can enlarge a vector-based graphic as big as a billboard and the results will still be very, very good.

There is still a lot for all of us to learn about graphic displays, but I hope this short tutorial has been of some help to your understanding of the subject. Ted Finch <tedfinch@techie.com>