Graphics

One of the main types of data that is stored on computers, and transmitted over the internet, is graphical data. Pictures. There are many different ways of storing graphical data, but they all follow similar principles. The most important of which is that the image must be represented as a set of binary codes.

Pixel	Picture Element. The smallest addressable part of an image. The image is divided up into dots, each of which can have a different colour, or level of brightness. These are pixels.
Resolution	Usually measured in dpi (Dots per inch). This is a measure of how sharp and detailed an image is. The higher the resolution, the more pixels per inch, and the greater the detail.

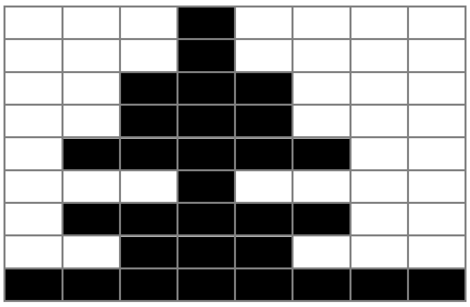
Each pixel is represented by a binary code, of one or more bits. How many bits depends upon the number of colours available on the system (The size of the palette.), as we shall see below.

Monochrome (2 colour) Image

Each pixel can be either black or white. This means that we only need one bit per pixel.

0	White
1	Black

So we can store the state of 8 bits in each byte of memory. The diagram below shows a small image of 8 pixels x 9 pixels, and how it might be stored in memory locations with a word length of 1 byte.



First set of 8 pixels	00010000
Second set	00010000
Third set	00111000
	00111000
	01111100
	00010000
	01111100
	00111000
	11111111

This is called a BITMAP

If we have a monochrome VDU which displays images of 320 x 200 pixels:

The number of pixels on the screen is 320 x 200 = 64,000.

With 1 bit per pixel:

The number of bits of memory needed to hold the image is 64,000 bits.

Which is equivalent to 8000 bytes.

4 Colour Image

Each pixel can be one of four colours. (This isn't a realistic example). This means that we will need 4 different binary codes, and that means that we will need to use two bits to represent the state of each pixel.

00	White
01	Green
10	Red
11	Blue

so now our small image might look like this:

SEE DIAGRAM ONE

So, with the same VDU, which can display images of 320 x 200 pixels:

The number of pixels on the screen is 320 x 200 = 64,000

With 2 bits per pixel:

The number of bits of memory needed to hold the image is 64,000 x 2 = 128,000

Which is equivalent to 16,000 bytes.

So the amount of memory needed has doubled though the size of the image is still the same.

16 Colour Image

The early PCs had what were called VGA displays. Each pixel could be set to one of 16 colours. This meant that 16 binary codes were needed, and therefore each pixel was represented by a 4 bit code. This is convenient from a systems programming point of view, because the codes for two pixels fit nicely into one byte of memory.

Repeating the calculation for the VDU mentioned in the other examples:

The number of pixels on the screen is 320 x 200 = 64,000

With 4 bits per pixel:

The number of bits of memory needed to hold the image is 64,000 x 4 = 256,000

Which is equivalent to 32,000 bytes.

256 Colours

SVGA displays are available which use 8 bits to store the state of each pixel. This allows the user to use any one of 2^8 = 256 colours, which is enough for reasonably realistic photographic images.

The 320 x 200 VDU would require 320 x 200 x 8 = 512,000 bits of memory (64,000 bytes) to hold the image.

True Colour

Computers which use 24 bits (3 bytes) to store the state of each pixel. This allows the use of 2^{24} = 16,777,216 different colours, which is enough for very realistic photographic quality images.

The 320 x 200 VDU would require 320 x 200 x 24 = 1,536,000 bits of memory (192,000 bytes) to store the image.

Video Cards

Of course, most VDUs on the market have much higher resolutions than 320 x 200. Displays of 1,280 x 1,024 are available. Because there are more pixels, this clearly means that more memory will be needed to store the image.

If we want to have True Colour on a 1,280 x 1,024 display, it is going to require:

1280 x 1024 x 24 = 31,457,280 bits = 3,932,160 bytes = 3.75 Megabytes

Which is quite a bit of memory. Nowadays, the main memory of the computer (RAM) is not used to hold the screen display image. When you buy a video card for your computer, one or more of the chips on it will be the memory which is set aside for the exclusive use of the VDU.

Saving Graphical Data in Files

When you save a graphic in a file, on disk, it is essentially just a long stream of bits, 1s and 0s. In order to enable the graphics software to reconstruct the image, there are certain things it needs to know. They include:

- Size of file (in bytes)
- Length of image in pixels
- Width of image in pixels

These can be stored at the head of the file, before the actual binary image data. Other information which is needed is:

- The number of bits per pixel
- Which binary codes represent which colours

A standard graphics format will probably be in use, which specifies this sort of information, and it can therefore be written into the software. JPEG and GIF are well known examples. If an image is stored as a GIF file, for example, each pixel will be stored as an eight bit binary code, with specified codes for each colour.

Graphic Manipulation Packages

These packages allow you to load and save graphical images in a number of file formats. They also give you fairly similar facilities for manipulating the images. These include:

Change View	-	Zoom in / out
Change Image	-	Mirror image, Rotate, Shear, Resize, Enhance edges, Blur, Sharpen
Change Colours	-	Contrast, Brightness, Change Red, Change Blue, Change Green, Convert to Greyscale
Drawing tools	-	Pen, Brush, Spraycan, Flood, Eraser, Lines, Circles, Squares, Freehand

Don't get confused between the image as it is held in the main memory (or on disk), and the image as it is held in the Display memory on the Video card. The Zoom facility is a good way of showing the difference. When the graphics package has loaded a graphics file from the disk, it copies it into a suitable data structure. Usually a two-dimensional array. It then renders (copies) the data into the video memory, and you see the image on the screen.

If you then use the zoom function on your graphics package, you see an enlarged image. The original image has not changed. You can zoom out again, and there it is, at the original size. When it was rendering the image, though, what it did was, for every pixel in the original image, write four pixels of the same colour in the video memory, like this:

SEE DIAGRAM TWO

This gives you a magnified view of the image. Of course, if you zoom too much, the image starts to look a bit 'blocky', like a mosaic. That is because the original image only has a certain amount of detail. There is no point enlarging beyond that point.

What about the opposite process. When you are zooming out, the software has to replace each block of 4 pixels with one pixel. The colour of that pixel will hopefully give some idea of the colours of the original four.

SEE DIAGRAM THREE

The left hand blocks are easy. eg Four green pixels are rendered as one green pixel. The middle one consists of two red and two yellow. If the software is sophisticated enough, it may detect that the block is next to a solid red one, and so try to compensate for the effect that that block will have on the eye by replacing this block with a yellow pixel in the video memory. The top right hand block is all blues. The white pixel sort of counter balances the dark blue one, so it is a question of choosing which of the available mid-blues best matches the effect that all four of those pixels would have on the eye from a distance. The lower right hand block is almost all magenta, so the one black pixel is basically ignored.

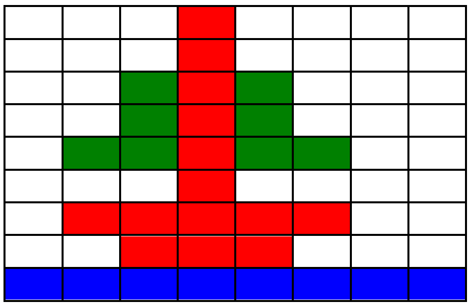
This process of approximating colours is known as **DITHERING**, and it can involve some very sophisticated mathematical algorithms.

Remember that the original image is still unchanged. It is perfectly possible to have a full 24 bit colour image stored in a file, but the graphics software may have to dither the image because the VDU can only display 8 bit colour images (due to the amount of memory on the video card).

Or maybe the user wants to print out the image. The printer might be able to handle the full number of colours, but maybe it is necessary to reduce the size of the image by a half, to fit it on a sheet of A4. The software will have to dither the image for exactly the same reasons (except that instead of rendering the image into the video memory, it will be copied to the printer buffer.)

There are many different graphics formats. Some of them are designed to make image manipulation easy. Some of them are designed to take up as little memory/disk space as possible. Which one you wish to choose depends on what you want to do with the images.

DIAGRAM ONE



1st 4 pixels row 1	00000010	row 5	01010000
2nd 4 pixels row 1	00000000	row 6	00000010
1st 4 pixels row 2	00000010	row 6	00000000
2nd 4 pixels row 2	00000000	row 7	00101010
1st 4 pixels row 3	00000110	row 7	10100000
2nd 4 pixels row 3	01000000	row 8	00001010
1st 4 pixels row 4	00000110	row 8	10000000
2nd 4 pixels row 4	01000000	row 9	11111111
1st 4 pixels row 5	00010110	row 9	11111111

DIAGRAM TWO

Main memory

Video memory

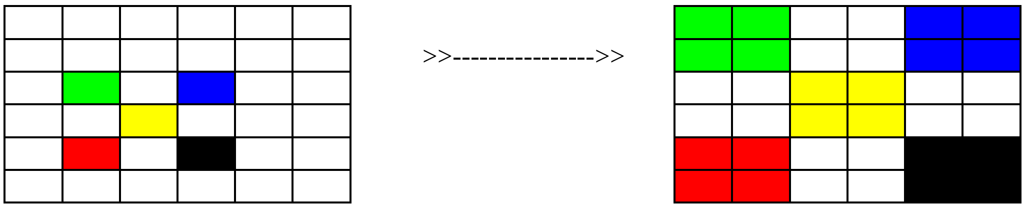


DIAGRAM THREE

Main memory

Video memory