CS50

Images

Overview

From social media to cancer screenings, newspapers to comic books, images are important in our lives. Images are stored as files, a series of bytes, just like the programs, word docs, and text files you're used to writing. Common image file formats include bitmaps (.bmp), JPGs (.jpg), PNGs (.png), TIFFs (.tiff), and GIFs (.gif).

Key Terms

- pixel
- hexadecimal
- header
- lossy
- lossless

Bitmaps

1	1	0	0	0	0	1	1
1	0	1	1	1	1	0	1
0	1	0	1	1	0	1	0
0	1	1	1	1	1	1	0
0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1

Our entire screen is composed of **pixels**, little dots with programmable color and brightness values. A bitmap describes a pattern of all the pixel values that make up an image: when our screen's pixels become those values, the image will appear. A bitmap takes some number of bits per pixel (bpp). More bits can be used to create a more detailed color palette. Back when screens were black and white, just a single bit was used per pixel, with `0 = black`, `1 = white`, as shown at left.

RGB Triples

The bitmaps we've worked with contain three bytes for each pixel in a color image. Each byte specifies a number between `0` and `255`, or, in **hexadecimal**, `0x00` to `0xff`. These three numbers detail how much red, green, and blue to put in that pixel. We can make sense of how this information comes together by thinking about painting, where mixing different combinations of just a few colors can produce many, many different shades. In this case, red, green, and blue can be combined to make the entire rainbow with varying levels of brightness.

Bitmap Headers

Bitmaps also have a **header** – a few bytes at the beginning of the file that tell the display program how to interpret the bits in that file. For the common .bmp Microsoft file extension, we use the struct at right to specify its header. These fields specify the size of the image in bytes, its width and height in pixels, and more. Having this information bundled together ensures that our display program knows exactly how to format the image.

```
typedef struct
{
    DWORD biSize;
    LONG biWidth;
    LONG biHeight;

    (...)

} __attribute__((__packed__))
BITMAPINFOHEADER;
```

Other Image File Formats

Do we need to store all of the information corresponding to every pixel in our image file? After all, many images feature a lot of repetition and redundancy (see what we did there?). Is there a way to encode this repetition to create smaller file sizes?

The answer is, typically, yes. Notice the four horizontal repetitions of 0 and 1, respectively, in the example at right. In the file below it, we encoded `(pixel value, repeat number)` pairs. What we've just done here is compressed the original file, resulting in a new file of smaller size.

GIF compression

0	0	0	0
1	1	1	1

⟷

0	4
1	4

There are two main types of file compression: lossy and lossless. In **lossy** file compression, files, such as JPGs, are compressed in such a way that data is lost, meaning that the original file cannot be completely recovered. On the other hand, in **lossless** compression, which is used with PNGs and GIFs, no data is lost and the original file can be exactly reconstructed. The compression in the example above falls into this category.