26 – Coding systems

Tuesday, February 5, 2019 9:09 AM

This chapter covers:

- Character codes
- Error checking and correction
- Data representation (images sound)

ASCII and Unicode

The American Standard Code for Information Interchange (ASCII) was originally implemented as 7 bits but was extended to 8-bits – allowing 256 different characters to be represented.

It has limitations:

- 256 characters is not enough to represent all different symbols;
- It was initially developed in English did not account for different languages;

This led to a Unicode emerging which follows the same principles as ASCII, but accounts for UTF-8 and UTF-16 with the latter using 16-bits per character – permitting the usage of many more symbols.

Error checking and correction

There are different methods for error checking and/or correction:

- A parity bit.
- Majority voting.
- Check digits

Bit-mapped graphics

The amount of memory allocated to each pixel is referred to as the colour depth.

Resolution = width * height (in pixels).
Storage size = resolution * colour depth.

Vector graphics

These describe the edges of an image with vectors. In general, they use less memory and can be zoomed in indefinitely. However, they lack realistic shading and artistic affects that can be achieved with bitmap graphics.

Analogue and digital signals

An example of converting from analogue to digital (ADC) is in a Musical Instrument Digital Interface (MIDI), e.g. one that fits beneath the strings of an acoustic guitar, that reads data from a musical instrument and converts this data into discrete, digital data.

Sound sampling and synthesis

We can sample (take frequent measurements) of an analogue symbol and store these amplitudes as digital (binary) data.

Storage size = sample rate (Hz) * length of recording (seconds) * sampling resolution (bits).

To then listen back to the sound, e.g., we must use a digital to analogue converter (DAC) to synthesise an analogue signal. This could be achieved through using pulse-width modulation and a low-pass analogue filter.

Data compression

Compression is the process of representing some data in fewer bits so that the file takes up less memory.

Lossless compression can be achieved through run-length encoding, or dictionary-based encoding. Run-length encoding works by replacing repeated blocks of data with a single unit of that data and a number representing how many times that unit repeats. E.g. "BBBBB" may be encoded as "B5". Dictionary-based compression works by assigning short tokens to longer strings of text that appear multiple times in the file. These are stored in a dictionary. E.g. "tion" comes up a lot at the ends of many words, so all these occurrences can be replaced with a token and then that token can be stored alongside "tion" in a dictionary (key) somewhere in the encoded file.

Lossy compression is used when lossless compression does not reduce the file size sufficiently – either due to storage space reasons or data transmission speeds. It works as you would expect – trying to guess what parts of the data can be removed without having a significant impact on the quality of the item, e.g. video, that is being represented. E.g. in JPEG compression, the images are split up into chunks which are individually evaluated to see how important they are to the image as a whole – e.g. with the sky, the slight changes in blue is insignificant so can be compressed/"smoothed over", but a sharp edge between a house and the sky is significant so should be kept.