

# Everything is numbers

## Effectiveness of data representation

For each of the activities, you may want to draw from the discussion points below to provide students with a greater appreciation of the way data is represented in each case. Note that this isn't intended to be an exercise in providing 'correct' answers: these discussions should pique student curiosity and encourage them to investigate things further, or formulate their own opinions.

### Count the dots

#### Understanding base 2 (binary)

One of the easiest ways to see if students have come to terms with how numbers are presented in binary is to ask them questions about how quickly numbers grow as they get larger.

A common misunderstanding is that adding an extra digit allows the number to get a little bit bigger (ie by 2 or 4), when in fact, it doubles in size. So, when your number is already quite big (say, 8 096), adding another bit to the number means you can now store double the number of value (16 192).

Ask students what the implications of this might have been when we moved from 32-bit computers, where the processor can perform operations on 2 32-bit values simultaneously, to 64-bit computing. It's a pretty big jump!

#### Why base 10?

Ask students why they think we, as humans, use the base 10 number system. Just as computers detect the presence or absence of electricity (and therefore only have that as a frame of reference), why might we have chosen ten as the base for our number system?

There's a great opportunity to explore some of the historical development of mathematics through this question, and there's plenty of information online that a quick search will uncover. Surprise answer – there is some conjecture that it has to do with us having ten fingers, but there is a lot of debate on the topic, especially when we consider that Ancient civilizations used a range of bases including 12 and 60.

You can read a quick summary of the debate at the Maths Stack Exchange: [Why have we chosen our number system to be decimal \(base 10\)?](http://math.stackexchange.com/questions/8734/why-have-we-chosen-our-number-system-to-be-decimal-base-10)<sup>1</sup>

#### Big vs small: are larger bases better?

What are the implications for using a higher base (like our base 10) over a base 2 system? If we completely ignore the practicalities of computing (ie the electrical signal detection), we can begin to think about the advantages and disadvantages of using larger bases.

Higher bases require more values to be represented: you need 10 symbols for base 10, but only 2 for base 2, which has implications for the complexity of the alphabet/language in use. You need to consider this alongside the size and complexity of the numbers generated – compare 23 to 10111.

Then there is the 'cost' of performing operations on these values. Adding numbers that consist of only 1s and 0s is simpler than adding numbers that contain 10 or more different symbols.

---

<sup>1</sup> <http://math.stackexchange.com/questions/8734/why-have-we-chosen-our-number-system-to-be-decimal-base-10>

## Colour by numbers

### Important relationships: computing power and colours

A 24-bit colour palette can represent over 16 million different colours by reserving 8 bits for each of the three additive colour channels (Red, Green and Blue).

The Wikipedia article [Color depth](https://en.wikipedia.org/wiki/Color_depth)<sup>2</sup> provides an explanation of how the RGB channels on a computer work in a range of different colour palettes. Modern computers generally use 32-bit colour (which adds a channel for transparency), but even with 8 bits of colour information (which allows us to represent 256 different colours) we can still generate quite good quality images – see the example 8-bit vs 24-bit image in the Wikipedia article linked above. Many people probably couldn't see the difference between the two pictures unless they looked very closely.

When computers could only process 16 bits at a time, it made sense to use as few colours as possible since you didn't want to slow processing time down to break the data up and reconstruct it again, but now we can process 64-bits at a time, allowing us to present higher quality images with in the same (or less!) time.

This is why older games and gaming systems have 'bad' graphics – to enable the computer to run the programs fast enough, colour information had to be taken away so that the total amount of data stored was smaller.

### Compression: why you can't just scale up a jpeg

Ever noticed how when you grab an image off the Internet and drop it into a document, that sometimes when you scale it up it looks rubbish? This is because some type of 'lossy' compression has been used to make the image smaller and easier to transfer.

A common image format that uses lossy compression is JPEG (.jpg or .jpeg, [JPEG](https://en.wikipedia.org/wiki/JPEG)<sup>3</sup>). The algorithm is fantastic for reducing the amount of data necessary to display an image at a given size, but as soon as you attempt to scale the picture, the loss of information means it is hard to reproduce the original image because there are so many bits missing.

To overcome this, 'lossless' compression algorithms can be used - the PNG ([Portable Network Graphics](https://en.wikipedia.org/wiki/Portable_Network_Graphics)<sup>4</sup>) image format is in common use and is an example of a lossless compression technique that still generates 'raster' (or bitmap) images.

An alternative to raster graphics is to use vector graphics ([Vector graphics](https://en.wikipedia.org/wiki/Vector_graphics)<sup>5</sup>). This type of image stores its data as a series of polygons, defined using control points that determine where on the canvas their edges are; the thickness of their strokes; and the colour of their fill (among other details). Since the polygons are defined as continuous paths, these scale very well without losing quality, and are typical of high quality images used for large format printing and fine detail.

So why not just use vector graphics? The answer is that sometimes, particularly when we are dealing with images such as photos where there are many colours and gradients at play, it isn't easy to define the entire image as mathematical paths.

---

<sup>2</sup> [https://en.wikipedia.org/wiki/Color\\_depth](https://en.wikipedia.org/wiki/Color_depth)

<sup>3</sup> <https://en.wikipedia.org/wiki/JPEG>

<sup>4</sup> [https://en.wikipedia.org/wiki/Portable\\_Network\\_Graphics](https://en.wikipedia.org/wiki/Portable_Network_Graphics)

<sup>5</sup> [https://en.wikipedia.org/wiki/Vector\\_graphics](https://en.wikipedia.org/wiki/Vector_graphics)

Instead, identifying the gradual change in colour from one pixel to the next is better achieved through changing the values of colour channels in individual pixels, so the decision about which type of graphic to use must consider the use case (eg printing vs. publishing online) and the type of image (eg logo that is likely to be used for small prints on letterheads and large banners as opposed to photographic print).

## Codes in a song

### Sound as a transmitter

The activity uses modems as the device for transmitting data through sound signals, but for many students, the modem is a piece of ancient technology that is unfamiliar. Today's modems (eg ADSL2 modems) use sounds that are outside of the range of human hearing, and so they appear to be silent.

The use of high–low signals is easy to understand when every bit of data being encoded can be represented as a bit-string value (such as the text examples used in this activity), but what happens when data becomes more complex? How do we take a song, made up of many different individual instruments playing simultaneously with different notes and pitches, and turn that into a bit stream?

Students will realise that there is a need to work out how that data is first going to be represented in some form of encoding, which would allow it to then be transferred through other means.

Given that all sound travels in waves, what we need is a means for representing a wave in some kind of bit-string encoding. The Sampling Sound activity demonstrates one way this can be done.

### Transmission using other media

The same concepts observed in this task can be applied to other media such as optical fibre. Where the modem uses low/high tones as a way of sending a series of '0s and 1s', optic fibre uses pulses of light (on for 1, off for 0) to transmit data.

Given that light travels so much faster than sound:

- What does this mean for the rate at which data can be sent over optic fibre?
- What about interference? Is sound or light going to be more robust to external interference, particularly when the cable is shielded?

The answer, of course, is that fibre is a much more robust medium for data transmission, but it is also comparatively more expensive. Light also needs to be channelled through glass, whereas sound can be sent through the air. This explains why fast wi-fi speeds are still well below the transfer rates of fibre.

## Sampling sound

### Representing sound

The technique used to represent audio in this activity is called [Pulse-Code Modulation](https://en.wikipedia.org/wiki/Pulse-code_modulation)<sup>6</sup> (PCM) and is used to produce .wav files – a commonly found audio file type on most computers.

In addition to this technique, modern audio data storage uses compression techniques to further reduce the file sizes of the data being stored while maintaining as much information (ie the quality) as possible. This is what allows CD-quality audio to be stored in small files, such as mp3, m4a or ogg files.

---

<sup>6</sup> [https://en.wikipedia.org/wiki/Pulse-code\\_modulation](https://en.wikipedia.org/wiki/Pulse-code_modulation)

Both bit depth and sample rates can have a significant impact on the size and quality of the reproduced audio. The most noticeable improvements occur as you move from very low values to higher ones – the move from a bit depth of 8 bits to 16 bits is much more noticeable in most audio samples than it is from 16 bits to 24-bits. Similarly, increasing the sample rate from 22 kHz to 44 kHz produces a far more noticeable improvement in audio quality than it does moving from 44 kHz to 96 kHz.

PCM is a lossy algorithm. Converting from the true, analogue audio signal to the digital representation will, by necessity, result in a loss of some of the original data. This is because sound waves are continuous, whereas PCM is a representation of audio at discrete, sampled intervals.

There is much debate about whether or not the loss of data is perceptible to the human ear ([Comparison of analog and digital recording](#)<sup>7</sup>), which is further complicated by the fact that modern audio recording techniques generally involve an element of digital processing that must, by its nature, cause some audio loss.

Today almost all audio, including professionally produced music, is recorded and distributed using digital means.

---

<sup>7</sup> [https://en.wikipedia.org/wiki/Comparison\\_of\\_analog\\_and\\_digital\\_recording](https://en.wikipedia.org/wiki/Comparison_of_analog_and_digital_recording)