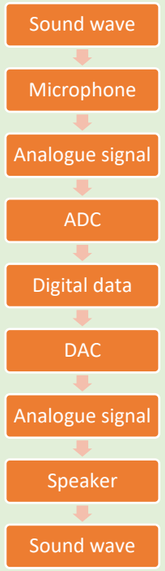
Representing Sound

Analogue and Digital

- Analogue data is the sound wave.
- The microphone (transducer) is a sensor that converts the sound wave into a continuous electrical analogue signal.
- To store the signal on the computer it needs to be converted to digital data using an ADC (Analogue to Digital Converter)
- The digital data is stored as a sequence of discrete values that are used to encode the signal.
- To listen to the digital sound that is stored in the computer a DAC (Digital to Analogue Converter) is used to create an electrical analogue signal.
- A vibrating speaker then converts the electrical signal into a sound wave.



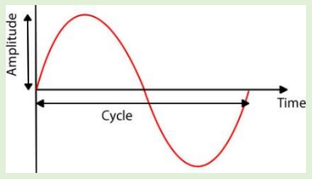
Analogue to digital conversion

For sound to be stored digitally on a computer it needs to be converted from its continuous analogue form into a discrete binary values.

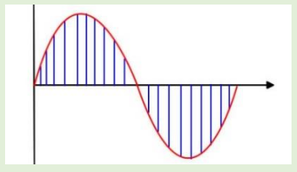
1. The analogue signal is sampled at regular intervals.
2. The samples are approximated to the nearest integer (quantised).
3. Each integer is encoded as a binary number with a fixed number of bits.

```
0001 0010 0011 0100 0100
```

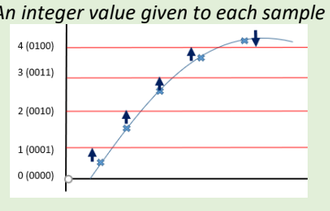
Original analogue signal



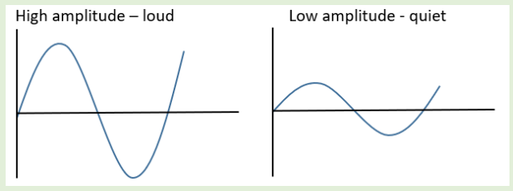
Signal sampled at regular intervals



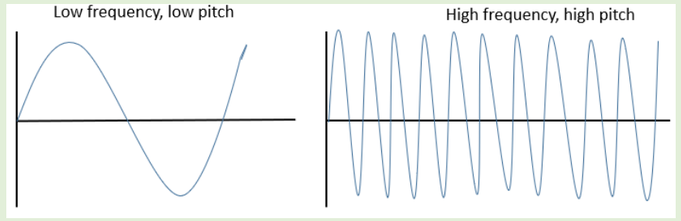
An integer value given to each sample



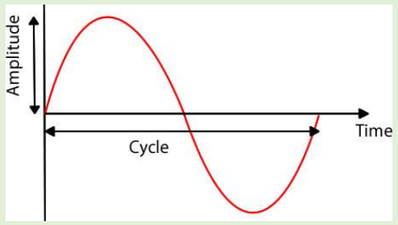
Amplitude - Represents the size of the wave and impacts the volume. The bigger the amplitude the bigger the volume.



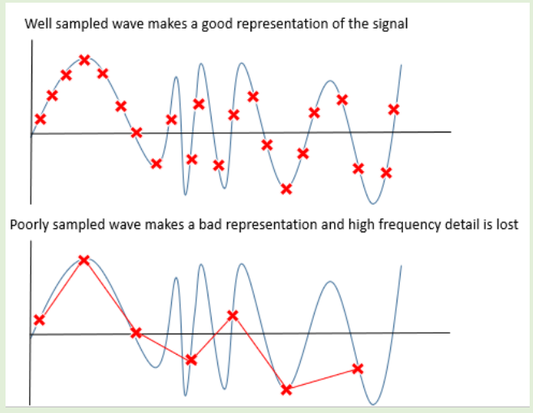
Frequency - Number of cycles per second is measured in Hertz (Hz). Frequency impacts the pitch, so the higher the frequency the higher the pitch.



Sample - Measure of amplitude at a given point in time

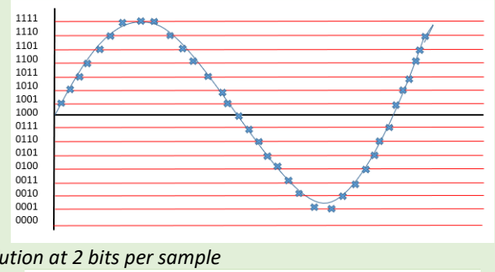


Sample rate is number of samples per second

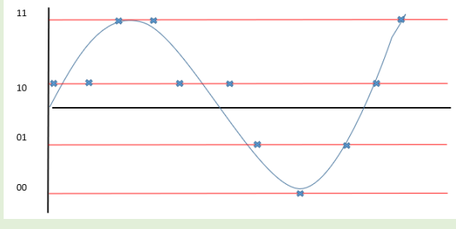


Sampling resolution is the number bits per sample

Sample resolution at 4 bits per sample



Sample resolution at 2 bits per sample



The **size of sound files** can be calculated using the following formula:

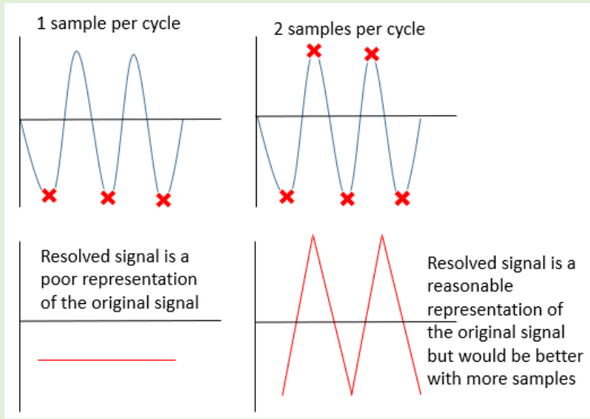
```
File size = length x sample rate x sampling resolution
```

where length is in seconds

Worked example: What is the size of a 30 second sound clip in kilobytes that is sampled at 8000 samples per second with a sample resolution of 16 bits?

30 x 8000 x 16 = 3840000 bits / 8x1000 = **480 Kbytes**

Nyquist theorem states that we need to sample at least at twice the rate of the highest frequency. That is we need to sample twice per cycle in order for us to resolve that frequency.



Musical Instrument Digital Interface (MIDI)

MIDI is synthesised music.

A MIDI file is a set of instructions that contains messages (or events) on how to produce a sound for a digital device.

MIDI is not stored music.

Information contained in a MIDI event

- Note-on
- Note-off (Both note on and note off give the event duration)
- Key pressure (Aftertouch – how hard a key is pressed
- Pitch
- Velocity
- Vibrato
- Volume
- Pitch Bend

This is not an exhaustive list, other information is also contained in a MIDI event.

A collection of events taken together allow us to produce a piece of music

Advantages of MIDI

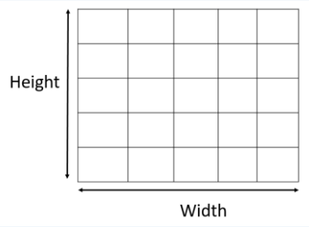
- MIDI files are generally small.
- Can play using different instruments
- Can easily edit MIDI files so there is no need for re-recording
- Can change the key of a piece of music

Bitmap Graphics

Bitmap images are made up from tiny dots called **pixels** (picture elements). Each pixel has a colour associated with it. An image can then be constructed from many of pixels which will have different colours arranged in rows and columns.

File Dimensions

Total number of pixels = height x width



Colour depth is the number of bits used to represent each pixel in an image. A black and white image has two colours. Each pixel can be represented by a single bit where 0 is black and 1 is white.

To represent more colours we can use more bits. For instance if we have 2-bits we can represent 4 colours because we know have 4 binary code combinations (00, 01, 10, 11) where each code represents a different colour.

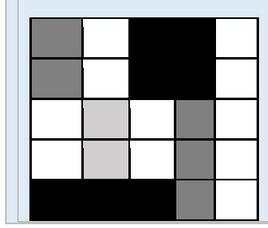
If we want to have 8 greyscale colours between black and white we will need 3 bits per pixel i.e. $2^{3.}$

With 8 bits we can represent 256 ($256=2^8$) colours (or different shades of grey)

Image resolution is the number of pixels per area of image and is often given in dots per inch (DPI). The more pixels the better the quality of the image. Lower resolution images may be pixelated.

Low resolution image, pixilation is clearly evident	*High resolution image, quality is much better*
	

Pixilation occurs when the image is overstretched. In these situations, the image has a blocky and blurred appearance. This arises when the image is presented at too large a size and there are not enough pixels to reproduce the details in the image at this larger size.

2-bit image with corresponding values	00 black 10 light grey 01 dark grey 11 white 01 11 00 00 11 01 11 00 00 11 11 10 11 01 11 11 10 11 01 11 00 00 00 01 11
	

Metadata is information about the images and is stored within the image itself. Metadata includes information on the:

- dimensions such as width and height
- colour depth in bits
- Resolution

The size of a bitmap image as stored on a file is calculated by multiplying the length, width and colour depth of the image.

File size in bits = width x height x colour depth (number bits per pixel)

File size in bytes = (width x height x colour depth) / 8

Worked Example
If the width of a bitmap image is 1000 pixels, the height is 2000 pixels and the colour depth is 8 bits, estimate the size of the image in Mbytes.

Total number of pixels: 1000 X 2000 = 2,000,000
Colour depth is 8 bits = 1 byte.
The size of the file is: 1 X 2,000,000 bytes
To convert to Mb we divide by 1,000,000
The file size in Mb is: **2Mb**

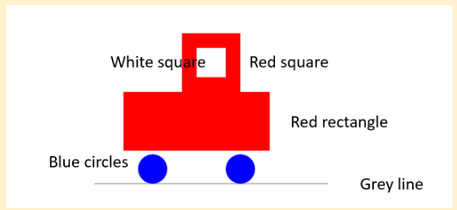
Vector Graphics

Vector graphics are made up of primitive shapes such as point, lines and polygons. Complex shapes can be constructed by combining basic primitive shapes.

Vector graphics are scalable meaning whatever size the images are represented at they do not lose their quality like bitmap graphics.

Vector graphics are made up of lists of primitive shapes that have various properties.

Note how the graphic below is made up of different shapes.



The properties of the shapes are stored in a list.

```
rect width="80" height="80" x="150" y="70" fill="red"
rect width="40" height="40" x="170" y="90" fill="white"
rect width="200" height="80" x="70" y="150" fill="red"
line x1="30" y1="275" x2="350" y2="275" stroke="grey"
circle cx="110" cy="255" r="20" fill="blue"
circle cx="230" cy="255" r="20" fill="blue"
```

The properties include the size, position and colour of the object.

Properties for a rectangle

width="80"	dimension
height="80"	dimension
x="150"	position
y="70"	position
fill="red"	fill colour

Properties for a circle

radius="50"	dimension
x="150"	position
y="70"	position
fill="red"	fill colour

Properties for a point

x="150"	position
y="70"	position
color="red"	fcolour

Properties for a line

x1="150"	position
y1="70"	position
x2="100"	position
y2="30"	position
colour="red"	colour
width="red"	line width

Vector Versus Bitmap Graphics

	Bitmap graphics Fundamentally made of pixels.	**Vector graphics** Made up of primitive shapes such as point, lines and polygons.
Advantages	Suitable for representing photographs of complex scenes with a multitude of colours, such as photograph of landscapes	Suitable for simple graphics with a limited range of colours such as logos and cartoons. Much smaller in size than bitmap graphics Vector graphics are scalable meaning that they retain their quality regardless of the scale at which they are viewed. Much better for editing. Can modify individual objects easily and supports layering in graphics packages.
Disadvantages	Pixilation occurs when the image becomes overstretched. The quality of bitmap image degrades as you zoom in.	Poor at representing images with lots of colours

Data Compression

The purpose of data compression is to make the files smaller which means that:

- Less time / less bandwidth to transfer data
- Take up less space on the disk

Data compression can be applied to all sorts of files including images, sound and text.

Lossy versus Lossless Compression

Lossless compression - The original uncompressed representation can be recreated exactly from the compressed image. That is, we can reverse the compression and the uncompressed data file will be exactly (down to the last bit) the same as the original file.

Lossy compression reduces the size of the file but the original uncompressed data will not be able to be fully recreated. Lossy compression only approximates the uncompressed data and the original data will be irrecoverable. Some of the quality of the original file will be lost. The benefit is that files using lossy compression will be smaller than those using lossless compression.

Run Length Encoding (RLE)

Run Length Encoding is a compression method where sequences of the same values are stored in pairs of the value and the number of those values.

RLE is a lossless compression method.

For instance, the sequence:

```
0 0 0 1 1 0 1 1 1 1 0 1 1 1 1
```

would be represented as:

```
3 0 2 1 1 0 4 1 1 0 4 1
```

Given that there are 7 bits per ASCII character, the uncompressed size of an ASCII phrase is:

size = number of characters (including spaces) x 7

Suppose we have a sequence of the following ASCII characters:

```
Goooooooaaaaallll
```

Uncompressed size: 17 x 7 = 119 bits

With RLE compression this would be: `1G, 7o, 5a, 4l`

Suppose we store the number of each character in 3 bits and we the ASCII characters are 7 bits then we have:

```
001 1000111 111 1101111 101 1100001 100 1101100
```

RLE compressed: `7x4 + 3x4 = 40 bits`

Dictionary based methods

With dictionary-based compression methods we use a dictionary to encode and decode the message.

We are going to compress the following chorus from a song by the Beatles:
"We all live in a yellow submarine yellow submarine yellow submarine"

Each ASCII character is 7 bits and there is a total of 67 characters including spaces so the total message is 67 x 7 bits which is 469 bits.

Using the dictionary method there are 4 by 11 = 44 bits. Ignoring the dictionary, this is more than ten times compression on the original message. Of course using a repetitive message helps compress the message.

For a short message we are not reducing the volume of data because the dictionary itself will have volume and in fact we may even increase the volume of data compared with the original uncompressed message however for long messages the dictionary method of compression will save us space.

Dictionary

word	Encoding
We	0000
all	0001
live	0010
in	0011
a	0100
yellow	0101
submarine	0110

Complete encoding

word	Encoding
We	0000
all	0001
live	0010
in	0011
a	0100
yellow	0101
submarine	0110
yellow	0101
submarine	0110
yellow	0101
submarine	0110

WARNING: Sometimes compression will increase the size of the file. For instance, perform RLE compression on the following sequence:

```
1 0 1 0 1 0 1 0 1 0 1 0 1 (13 bits)
```

This would be:

```
11 01 11 01 11 01 11 01 11 01 11 01 11 (26 bits)
```

The size of the file has been doubled by applying a compression algorithm!

Encryption

When transferring sensitive information from one place to another it is necessary to **encrypt** the message. **Encryption** means that the message is scrambled from its original **plaintext** into **ciphertext.** A **key** is required in order to understand the original message.

Plaintext is the original message

Ciphertext is the encrypted message

Given enough time, ciphertext and computational power nearly all methods of encryption can be cracked using **cryptoanalysis**.

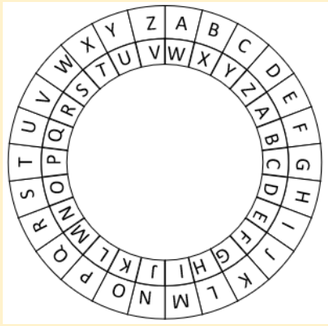
Cryptoanalysis is the breaking of encrypted codes without being given the key. Only the **Vernam** cipher has proven to be unbreakable.

Caesar Cipher
The Caesar cipher is a shift substitution cipher. It works by replacing a letter with another letter that is shifted along in the alphabet by a set number of places. The key is a letter and lets us know the number of letters we shift the alphabet by. So for instance a key of *E* means that we would shift by 4 places.

In the example below we are using a key W so we shift by 22 places to the left.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V

This can be represented as a wheel. The letters on the outer wheel represent the original letters, the letters on the inside are the corresponding encrypted letter.



- The word MARCH would be encrypted as IWNYD
- To decrypt the message we use reverse the process. So XASWNA would be BEWARE

Cryptoanalysis of Caesar Cipher

- The Caesar cypher is not a secure method of encryption and is easy to crack using two methods.
- Try all 25 shifts and apply it to the cipher text. When you have a sensible plaintext message then you have cracked the encryption and found the key.
- Alternatively, if you had a long enough piece of cyphertext you could use frequency analysis of letters. E is the most commonly used letter, so the most commonly letter in the cyphertext will likely be a E and from there is it a trivial step to calculate the key

Vernam (one-time pad) cipher
With a Vernam cipher we have perfect security. It is the only cipher that has proven impossible to break using cryptoanalysis. To help achieve this the Vernam cipher has three features:

- The key is only used once
- The length of the key is at least long as the message that we want to send.
- It is truly random

Applying the Vernam Cipher

Original Message	Hello
Convert to ASCII plain text	1001000 1100101 1101100 1101100 1101111
Generate a random key	!d7sY
Key in ASCII	0100001 1100100 0110111 1110011 1011001
Apply bitwise XOR to key and plain text	1001000 1100101 1101100 1101100 1101111 0100001 1100100 0110111 1110011 1011001 1101001 0000001 1011011 0011111 0110110
Encrypted message (cypher text)	1101001 0000001 1011011 0011111 0110110
To decrypt apply the XOR operator to the cypher text and the key	1101001 0000001 1011011 0011111 0110110 0100001 1100100 0110111 1110011 1011001 1001000 1100101 1101100 1101100 1101111
Plain text	1001000 1100101 1101100 1101100 1101111