

Cambridge International AS Level **Computer Science** **9608**

Unit 1 Theory Fundamentals

1.1 Information representation

Contents

1.1.1 Number representation.....	3
Show understanding of the basis of different number systems and use the binary, denary and hexadecimal number system	3
Convert a number from one number system to another	3
Express a positive or negative integer in 2's complement form	6
Show understanding of, and be able to represent, character data in its internal binary form depending on the character set used.	7
Express a denary number in Binary Coded Decimal (BCD) and vice versa	8
Describe practical applications where BCD is used	8
1.1.2 Images	9
Show understanding of how data for a bitmapped image is encoded.....	9
Use the terminology associated with bitmaps: pixel, file header, image resolution, screen resolution.....	11
Perform calculations estimating the file size for bitmapped images of different resolutions	11
Show understanding of how data for a vector graphic is represented and encoded	11
Use the terminology associated with vector graphics: drawing object, property and drawing list.	11
Show understanding of how typical features found in bitmapped and vector graphics software are used in practice.....	12
Justify where bitmapped graphics and/or vector graphics are appropriate for a given task.....	13
1.1.3 Sound	14
Show understanding of how sound is represented and encoded	14
Use the associated terminology: sampling, sampling rate, sampling resolution	15
Show understanding of how file sizes depend on sampling rate and sampling resolution	15
Show understanding of how typical features found in sound-editing software are used in practice	15
1.1.4 Video	16
Show understanding of the characteristics of video streams.....	16
1.1.5 Compression techniques.....	19
Show understanding of how digital data can be compressed, using either 'lossless' (including runtime encoding – RTE) or 'lossy' techniques.....	19
Bibliography	20

1.1.1 Number representation

Show understanding of the basis of different number systems and use the binary, denary and hexadecimal number system

Convert a number from one number system to another

http://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving_Programming_Data_Representation_and_Practical_Exercise/Fundamentals_of_Data_Representation/Binary_number_system

In base 10 we have some important numbers we give names to: for example 10x10x10 is 1000 which we call a thousand; and 1 thousand multiplied by 1 thousand which we call 1 million; and so on. In base 10 we are also used to the metric system which uses kilo to mean a thousand, for example kilometre or kilogram. In binary we also have names to describe key values.

The basic unit is 0 or 1 – this is a binary digit or a bit.

A group of 8 bits is called a byte and half a byte (4 bits) is called a nibble.


We also use the kilo prefix to represent the same sort of scale as we do in base 10. But 1000 is not correct in binary which is 2^{10} or $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 1024$. Using this approximation to 1000 we can now define a whole set of names commonly used to describe binary numbers

8 bits	1 byte
1024 bytes	1 kilobyte
1024 kilobyte	1 megabyte
1024 megabytes	1 gigabyte
1024 gigabytes	1 terabyte

Computers are made up hardware that stores and processes data. If you break a computer down into its most basic components you have millions of circuits that either allow electricity to flow, or not. The computer uses electronic circuits to store one of two values using a switch – the switch is either on (1) or off (0). Using a number of these switches provides us with many possible combinations of 1s and 0s which we can use to represent numeric values. This is called Binary.

Denary: A system of numbers using ten digits, 0 and 1-9 (also called base-10 system)

Binary: A system of numbers using only 2 digits, 0 and 1 (also called base-2 system)



128	64	32	16	8	4	2	1
$2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$	$2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$	$2 \times 2 \times 2 \times 2 \times 2 \times 2$	$2 \times 2 \times 2 \times 2$	$2 \times 2 \times 2$	2×2	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	1^0

To convert Binary to Denary

To convert the binary 111001 into a denary number use the column headings.

128	64	32	16	8	4	2	1
0	0	1	1	1	0	0	1

The number is $32+16+8+1 = 57$ (add up the column headings where there is a 1)

To convert Denary to Binary

Use the same column headings. You need to find the biggest column heading that you can take away from the number and start there:

Let's convert 57 into binary:

The biggest column heading we can take out of 57 is 32 (the next one is 64, which is too big) Write a 1 under the column heading 32. That leaves us with $57-32 = 25$. Write a 1 under the column heading 16 (because we can take 16 out of 25. $25-16$ leaves 9) You should be able to see now that 9 is an 8 and a 1 so we end up with:

128	64	32	16	8	4	2	1
0	0	1	1	1	0	0	1

Always double check by adding the columns up at the end. They should give you the number you started with $32+16+8+1=57$

Hexadecimal

Humans are not very good at remembering long strings of numbers so, to make it easier for us, we can represent every group of 4 bits with a single digit.

The smallest value you can have with 4 bits is $0000 = 0$. The largest value is $1111 = 15$. In base 10 we have 10 symbols 0 to 9, if we use the 16 symbols for 0 to 15 we can use a system based on place values of 16 rather than 2 or 10. We call this hexadecimal (or hex for short). We do, however need to have symbols for the numbers 10, 11, 12, 13, 14 and 15. We use the letters A, B, C, D, E and F to represent these values.

Allowed Hexadecimal digits are:

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

To convert Hexadecimal to Denary

Converting hex to denary is the same process we have used before with column values, using the values 1 and 16.

Let's convert Hex 23 to denary:

Place the first Hex digit under the column heading 16 and the second Hex digit under the column heading 1. Complete the calculations to find out the denary value.

16	1
2	3
2 X 16 = 32	3 X 1 = 3
32 + 3 = 35 in denary	

To convert Denary to Hexadecimal

Converting denary to hex is the same process we have used before with column values, using the values 1 and 16 except this time we need to work out how many groups of 16 there are in the denary number.

Let's convert 182 to hex:

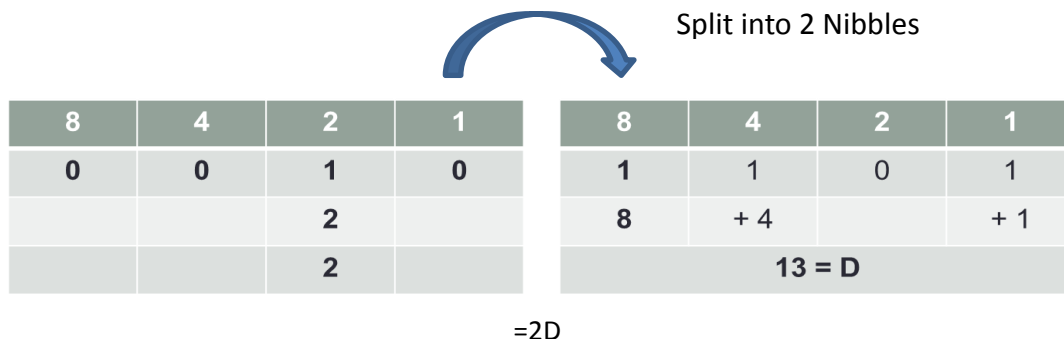
$$182/16 = 11 \text{ remainder } 6$$

Represent this using the table and replace the denary with its corresponding Hex value.

16	1
11	6
B	6
B6	

To convert a Binary number to Hex

To convert from Binary to Hex you first need to represent the binary using two nibbles. For example, the binary number 00101101 can be represented in 2 nibbles as shown below.



Then replace the values with the corresponding Hex digits.

To convert a Hex to Binary

To convert from Hex to Binary you need to split the Hex digits into two nibbles and then represent each digit in binary as shown below.

A5
Split it up into 2 nibbles and
represent using numbers using
binary

A = 10				5			
8	4	2	1	8	4	2	1
1	0	1	0	0	1	0	1

Express a positive or negative integer in 2's complement form

If a computer system uses one byte to store a number three problems arise:

- The biggest number that can be represented is 255. This is solved by using more than one byte to represent a number. Most computer systems – in particular, high-level programming languages – use either two or four bytes to store integers. There is still a limit on the size of integer that can be represented, but it is now much larger.
- Fractions cannot be represented.
- Negative numbers cannot be represented. This section considers two methods for representing negative integers.

The denary number 117 becomes 01110101 in binary. If we want to be able to store +117 and -117, the number needs a second piece of information to be stored, namely the sign. There are two ways of doing this.

Sign and magnitude

The first bit in the byte – i.e. the most significant bit (MSB) – will represent the sign. 0 represents positive and 1 represents negative. This means that:

+117 = 01110101 and -117 = 11110101

We say the most significant bit is being used as a “sign bit”.

Note that the range of possible integers represented by one byte is now -127 to +127 because we only have seven bits to store the size or “magnitude” of the integer.

The byte does not represent just the size of the integer but also its sign. This makes arithmetic unpredictable so there is still a major issue with using the “sign-and-magnitude” method.

Two's complement

The MSB still represents a number but is assumed to be negative. This means that the column headings, i.e. the “place values” of the number, become:

-128	64	32	16	8	4	2	1

The denary number +117 does not need to use the MSB, so it still becomes 01110101.

However, -117 must be thought of as -128 + 11:

-128	64	32	16	8	4	2	1
1	0	0	0	1	0	1	1

So the denary number -117 becomes 10001011.

Note that the range of possible integers represented by one byte is now -128 to $+127$. Two's complement has the major advantage over sign-and-magnitude representation that addition and subtraction of two numbers in two's complement form produces the correct result – as long as the result is within the permitted range of possible integers.

The following algorithm is another way of calculating the two's complement value of a negative number:

1. Work out the binary value of the positive number (make sure you write down all the leading zeros).
2. Change all the digits, 0 for 1 and 1 for 0.
3. Add 1.

This is usually remembered as: "Write down the positive number – flip the bits – add 1".

Let's apply this algorithm to our example of -117 :

1. Work out the binary value of the positive number:
01110101.
2. Change all the digits: 10001010.
3. Add 1: 10001011.

This result matches the value we got using the method above.

http://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Fundamentals_of_Data_Representation/Two%27s_complement

Show understanding of, and be able to represent, character data in its internal binary form depending on the character set used.

Many different types of data have to be stored in computers, including numbers and character data. We consider how characters are stored in a computer.

Values are stored inside a computer as a series of 0s and 1s. A single 0 or 1 is called a binary digit or *bit*. Any group of 0s and 1s can be used to represent a specific character, for example, a letter. The number of bits used to store one character is called a **byte**. The complete set of characters that the computer uses is known as its **character set**.

Each of the characters in a character set must have its own binary value, which is the code by which the computer recognises it. For example, "A" could be represented as 000, "B" as 001, "C" as 010 and so on. However, there are only eight possible codes using three bits, so we could store the letters A to H but not the rest of the alphabet. We also need to represent the lower case letters, punctuation marks, digits and so on. The computer can store as many characters as necessary simply by using sufficient bits in the code.

The size of the character set depends on what the computer is meant to be able to do. Some systems don't need to be able to recognise a lot of characters so they use only a few bits for each character. A good example of this is an ATM (a cash machine) which needs very few characters to operate the interface.

A good way of thinking about the character set for a normal computer is to have a look at the characters that are available on its keyboard. For this character set eight bits are used to store a single character. Using a byte (eight bits) for a character, we have 256 codes (i.e. 256 different eight-bit binary values). This is enough to represent each of the characters on a standard keyboard.

Imagine that one computer stores "A" as 01000001 and another computer stores "A" as 01000010. Any document created on one of the computers will not make sense on the other,

because they will interpret the codes differently. Computers can only communicate if they can understand each other's codes. In the 1960s, a meeting in the USA agreed a standard set of codes known as the American Standard Code for Information Interchange (**ASCII**). Most computer systems today use the ASCII coding system, so you can be fairly sure that when you type in "A" on any computer, it is stored in the computer's memory as 01000001 and will be read as "A" by any other computer.

The ASCII coding system uses seven bits to represent each character and the eighth bit as a means of checking the rest (we discuss this in Chapter 1.5). This means that 128 different characters can be represented in the standard ASCII character set. (As this is the most common character set, people generally consider a byte to be eight bits).

Unicode is a more recent, 16-bit code that uses two bytes. Using 16 bits makes it possible to represent over 65,000 characters. This means that all the characters used by the world's languages can be represented in Unicode. It is widely used to handle documents, particularly if a single document needs to be written in, for example, English, Arabic and Chinese. Unicode also allows the localisation of software, where standard software can be adapted for use by different cultures by modifying the layout and features of the interface.

Lots of different character sets are available to a computer for different tasks. Consider how many bits are needed for:

- the ANSI set, which includes graphical symbols, lines and shapes
- the standard Chinese character set, which has thousands of different characters.

Each applications program determines how to interpret a binary pattern by the context in which the value is used.

ASCII American Standard Code for Information Interchange – widely used 7-bit coding system used for character data

Unicode 16-bit character coding system

http://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Fundamentals_of_Data_Representation/ASCII

http://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Fundamentals_of_Data_Representation/Unicode

Express a denary number in Binary Coded Decimal (BCD) and vice versa Describe practical applications where BCD is used

Some numbers are not proper numbers because they don't behave like numbers. For example, a barcode looks like a number but if the barcode for a chocolate bar is added to the barcode for a sponge cake the result is meaningless. Values that are written as a string of digits but do not behave like numbers are often stored in **binary coded decimal (BCD)**.

Each digit is changed into a four-bit binary number that is then placed after one another in sequence.

Decimal Digit	BCD 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

For example, to convert 398 into BCD, we convert the 3 to 0011, the 9 to 1001 and the 8 to 1000. Thus, in BCD, 398 is represented by 001110011000.

Note that it is essential to retain the leading zeros.

The hexadecimal number BD in BCD gives B = 1011 and D = 1101. This gives the binary number 10111101.

Binary coded decimal (BCD) Representation of a number that uses a group of four binary digits to represent each digit in a denary number, e.g. 78 in denary is 0111 1000 0011 in BCD.

1.1.2 Images

A large part of using modern computers involves sending pictures and films to each other, along with using a graphical user interface. All of this involves computers saving and processing images. This section will cover the two main image types: vector and bitmap, along with some compression techniques.

Show understanding of how data for a bitmapped image is encoded

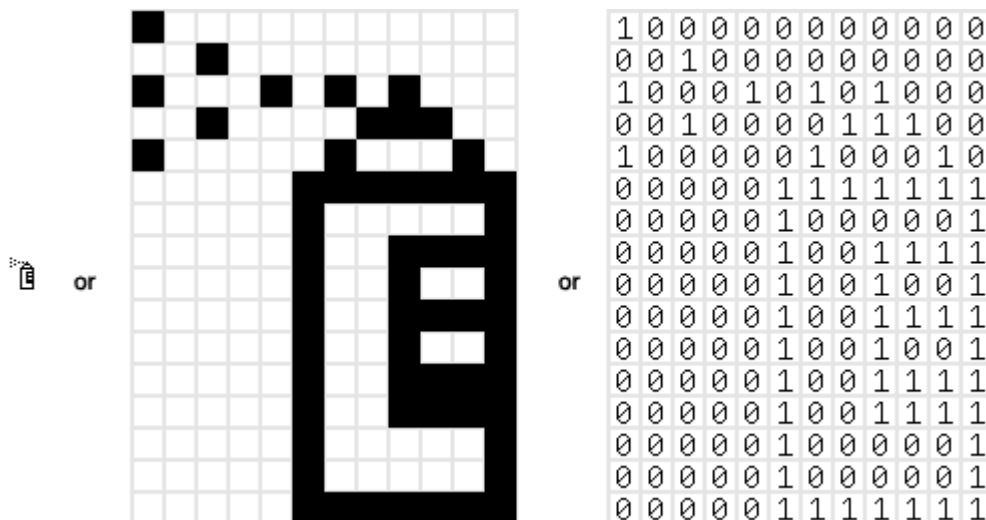
Bitmap graphic Image made up of a number of small “picture elements” called pixels.

In **bitmap graphics**, the picture is made up of a series of square pixels of different colours.

The colours available determine the type of bitmap:

- black and white (monochrome)
- grey-scale
- 16-colour
- 256-colour
- true colour – where millions of colours are possible.

Imagine the whole of the computer screen being made up of many thousands of pixels. Each pixel may be set to ‘on’ or ‘off’ depending on whether the value of the pixel in the computer’s memory is 1 or 0.



This graphic show how graphics are stored in the computer's memory; the picture is drawn on a 16 x 12 grid. Grid squares which are 'on' are represented by a '1' and grid squares that are 'off' are represented by a '0'. The amount of memory required to store this graphic would be 16 x 12 bits, which is 192 bits.

The quality of the picture is determined by the **resolution** of the graphics available. The smaller the size of the pixels the finer the detail that can be displayed on the screen. Small pixels mean high resolution. Large pixels mean low resolution.

In general, the higher the definition and the more colours which are used, the greater the final file size of the image.

A problem can arise when we attempt to increase the size of a bitmap image because each of the pixels increases in size, losing definition. If the size increase is too great, the individual pixels can become visible. This is known as the "staircase effect".

http://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Fundamentals_of_Data_Representation/Bitmaps

Colour depth - The number of bits used to represent the colour of a single pixel.

With 1 bit we can have just 2 colours, black or white, for example.

With 2 bits we can have 2² or 4 colours

With 3 bits we can have 2³ or 8 colours and so on

8 bits will give us 2⁸ or 256 colours

16 bits will give us 2¹⁶ or 65536 colours

So the more the bits per pixel, the greater the colour depth or range of colours in the image, we achieve.

It seems pretty obvious that the higher the colour depth, the closer the picture will look to reality. Why then don't we just ramp up the colour depth on every image that we make? The answer should be obvious, for a fixed resolution, the higher the resolution, the larger the file size.

Use the terminology associated with bitmaps: pixel, file header, image resolution, screen resolution

Pixel - the smallest possible addressable area defined by a solid colour, represented as binary, in an image

File Header - contains information about the type, size, and layout of a file

Image Resolution - how many pixels an image contains per inch/cm

Screen Resolution - the number of pixels per row by the number of pixels per column

Perform calculations estimating the file size for bitmapped images of different resolutions

The storage requirements of bitmapped images can be calculated using the following formula:

*Storage requirements =
total number of pixels used in the image x bit depth/colour depth*

Example:

Suppose the graphic to be stored is 1 inch by 2 inches, and the resolution is 90dpi in 256 colours.

The total number of pixels used in the image is $1 \times 2 \times 90 \times 90 = 16200$.

The number of bits used to represent 256 colours is 8 bits.

The storage requirements would therefore be 16200×8 , which is 129600 bits, 16200 bytes or 15.82 Kilobytes.

Show understanding of how data for a vector graphic is represented and encoded

Vector graphic Image made up of numerous vectored shapes that can be re-sized without a reduction in resolution.

In **vector graphics**, a drawing is composed from a toolbox of available shapes – rectangle, straight line and thousands of others – called “drawing objects”. The properties of each object are stored as a set of mathematical equations, coordinates, etc. Hence when an object is re-sized, none of its definition is lost as the re-sizing is achieved by re-calculating all the properties.

http://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Fundamentals_of_Data_Representation/Vectors

Use the terminology associated with vector graphics: drawing object, property and drawing list

Drawing object - Vector graphics are made up of objects and their properties. An object is a mathematical or geometrically defined construct such as a rectangle, line or circle.

Property - Each of these objects has properties to tell you the size, colour, position etc. Take a look at the next example to see how drawing lists are built from objects and properties.

Drawing list - a set of commands used to define a vector image

Show understanding of how typical features found in bitmapped and vector graphics software are used in practice

Different software packages create graphics in different ways. Microsoft Paint creates bitmap graphics while Microsoft Visio creates vector graphics.

Image Editor Features

Listed below are some of the most used capabilities of the better graphic manipulation programs. The list is by no means all inclusive. There are a myriad of choices associated with the application of most of these features.

Selection

One of the prerequisites for many of the applications mentioned below is a method of selecting part(s) of an image, thus applying a change selectively without affecting the entire picture. Most graphics programs have several means of accomplishing this, such as:

- a marquee tool for selecting rectangular or other regular polygon-shaped regions,
- a lasso tool for freehand selection of a region,
- a magic wand tool that selects objects or regions in the image defined by proximity of colour or luminance,
- vector-based pen tools,

The border of a selected area in an image is often animated with the marching ants effect to help the user to distinguish the selection border from the image background.

Layers

Another feature common to many graphics applications is that of Layers, which are analogous to sheets of transparent acetate (each containing separate elements that make up a combined picture), stacked on top of each other, each capable of being individually positioned, altered and blended with the layers below, without affecting any of the elements on the other layers. This is a fundamental workflow which has become the norm for the majority of programs on the market today, and enables maximum flexibility for the user while maintaining non-destructive editing principles and ease of use.

Image size alteration

Image editors can resize images in a process often called image scaling, making them larger, or smaller. High image resolution cameras can produce large images which are often reduced in size for Internet use.

Cropping an image

Digital editors are used to crop images. Cropping creates a new image by selecting a desired rectangular portion from the image being cropped. The unwanted part of the image is discarded. Image cropping does not reduce the resolution of the area cropped. Best results are obtained when the original image has a high resolution. A primary reason for cropping is to improve the image composition in the new image.

Histogram

Image editors have provisions to create an image histogram of the image being edited. The histogram plots the number of pixels in the image (vertical axis) with a particular brightness value (horizontal axis). Algorithms in the digital editor allow the user to visually adjust the brightness value of each pixel and to dynamically display the results as adjustments are made. Improvements in picture brightness and contrast can thus be obtained.

Noise reduction

Image editors may feature a number of algorithms which can add or remove noise in an image. Some JPEG artefacts can be removed; dust and scratches can be removed and an

image can be de-speckled. Noise reduction merely estimates the state of the scene without the noise and is not a substitute for obtaining a "cleaner" image.

Selective colour change

Some image editors have colour swapping abilities to selectively change the colour of specific items in an image, given that the selected items are within a specific colour range.

Image orientation

Image orientation (from left to right): original, -30° CCW rotation, and flipped. Image editors are capable of altering an image to be rotated in any direction and to any degree.

Perspective control and distortion

Perspective control: original (left), perspective distortion removed (right). Some image editors allow the user to distort (or "transform") the shape of an image.

Enhancing images

In computer graphics, the process of improving the quality of a digitally stored image by manipulating the image with software. It is quite easy, for example, to make an image lighter or darker, or to increase or decrease contrast. Advanced photo enhancement software also supports many filters for altering images in various ways. Programs specialized for image enhancement are sometimes called image editors.

Sharpening and softening images

Graphics programs can be used to both sharpen and blur images in a number of ways, such as unsharp masking or deconvolution.^[2] Portraits often appear more pleasing when selectively softened (particularly the skin and the background) to better make the subject stand out. This can be achieved with a camera by using a large aperture, or in the image editor by making a selection and then blurring it.

Change colour depth

An example of converting an image from colour to grayscale

Contrast change and brightening

Image editors have provisions to simultaneously change the contrast of images and brighten or darken the image. Underexposed images can often be improved by using this feature.

Colour adjustments

The colour of images can be altered in a variety of ways. Colours can be faded in and out, and tones can be changed using curves or other tools.

Vector graphics editors typically allow rotation, movement (without rotation), mirroring, stretching, skewing, affine transformations, changing of z-order (loosely, what's in front of what) and combination of primitives into more complex objects.

More sophisticated transformations include set operations on closed shapes (union, difference, intersection, etc.).

Justify where bitmapped graphics and/or vector graphics are appropriate for a given task

As a general rule bitmaps are typically used to depict lifelike images whereas vector graphics are more often used for abstract images such as logos. There are however numerous exceptions to this rule. It is often impossible to determine whether an image is a bitmap or a vector file just by looking at it.

Vector graphics are ideal for simple or composite drawings that need to be device-independent, or do not need to achieve photo-realism. For example, the PostScript and PDF page description languages use a vector graphics model.

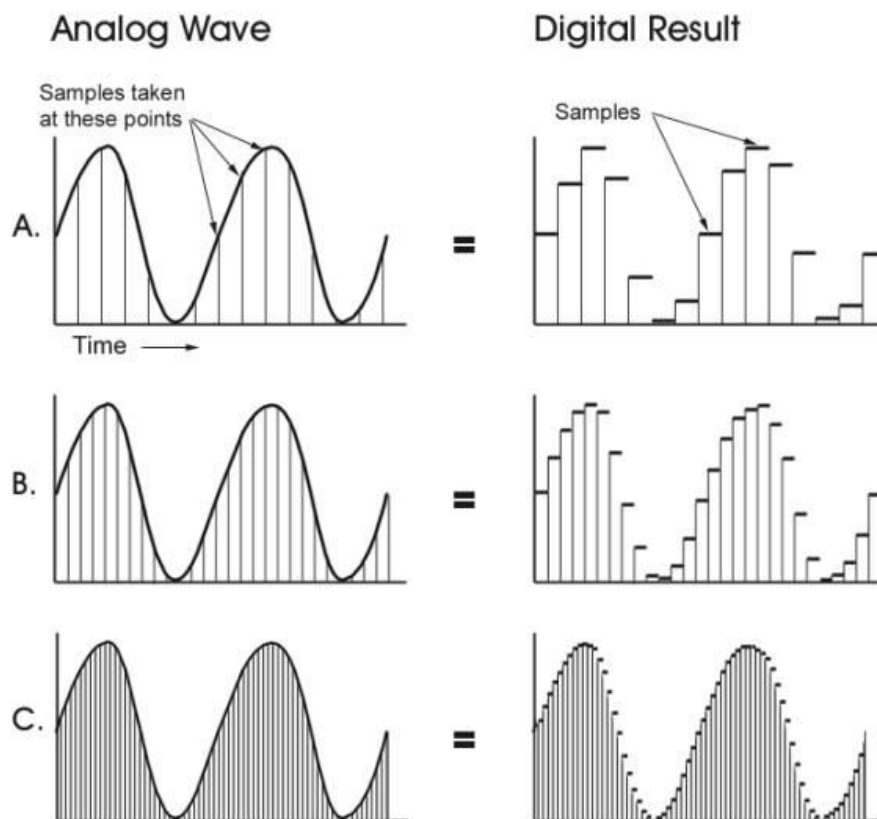
1.1.3 Sound

Show understanding of how sound is represented and encoded

Sound is analogue in nature: it is a continuously varying quantity. Computers work in digital quantities. In order to input sound into a computer the sound must be changed from analogue into digital. This process is called analogue to digital conversion or ADC. A microphone or another sound source is connected to the computer's sound card in order to capture the sound. The sound card carries out the ADC in a process called sampling. Sampling is a form of digitising. The sampling rate is the frequency at which samples of the sound are taken and is usually twice the maximum frequency of the sound being sampled. The sampling resolution is the number of bits used for data storage, which is 16 bits for so-called CD quality sound.

When sampling a sound, the analogue signal is chopped into a number of slices per second. At each slice, the amplitude (or height) of the signal is measured and rounded to the nearest available digital value. The more chops per second (sampling rate) and the finer the values which may be assigned to the amplitude (sampling resolution), the better the digital representation of the original analogue sound.

Increasing Sample Rates



The image shows the principle of sampling an analogue audio signal. You can see that as the sampling rate increases, the digitised sample becomes closer to the original signal. Close inspection of the sampling process part of the diagram should also show that each part of the analogue signal is given its nearest digital value. The consequence of this is that sampling can never exactly reproduce the original. The benefit of sampling is that it reduces the quantity of data that requires to be stored, as compared to the analogue version.

Use the associated terminology: sampling, sampling rate, sampling resolution

Sampling - Sampling is a form of digitising

Sampling Rate - The sampling rate is the frequency at which samples of the sound are taken and is usually twice the maximum frequency of the sound being sampled.

Sampling Resolution - The sampling resolution is the number of bits used for data storage

Show understanding of how file sizes depend on sampling rate and sampling resolution

To create digital sound as close to the real thing as possible you need to take as many samples per second as you can. When recording MP3s you'll normally use a sampling rate between 32,000, 44,100 and 48,000Hz (samples per second). That means that for a sampling rate of 44,100, sound waves will have been sampled 44,100 times per second! Recording the human voice requires a lower sampling rate, around 8,000Hz. If you speak to someone on the phone it may sound perfectly acceptable, but try playing music down a telephone wire and see how bad it sounds.

The sampling resolution allows you to set the range of volumes storable for each sample. If you have a low sampling resolution then the range of volumes will be very limited, if you have a high sampling resolution then the file size may become unfeasible. The sampling resolution for a CD is 16 bits used per sample.

Bit rate - the number of bits required to store 1 second of sound

To work out the size of a sound sample requires the following equation:

File size = Sample Rate * Sample Resolution * Length of sound

This is the same as saying:

File size = Bit Rate * Length of sound

Show understanding of how typical features found in sound-editing software are used in practice

Editors designed for use with music typically allow the user to do the following:

- The ability to import and export various audio file formats for editing.
- Record audio from one or more inputs and store recordings in the computer's memory as digital audio
- Edit the start time, stop time, and duration of any sound on the audio timeline
- Fade into or out of a clip (e.g. an S-fade out during applause after a performance), or between clips (e.g. crossfading between takes)
- Mix multiple sound sources/tracks, combine them at various volume levels and pan from channel to channel to one or more output tracks

- Apply simple or advanced effects or filters, including compression, expansion, flanging, reverb, audio noise reduction and equalization to change the audio
- Playback sound (often after being mixed) that can be sent to one or more outputs, such as speakers, additional processors, or a recording medium
- Conversion between different audio file formats, or between different sound quality levels

Typically these tasks can be performed in a manner that is both non-linear and non-destructive.

http://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Fundamentals_of_Data_Representation/Sounds

1.1.4 Video

Show understanding of the characteristics of video streams:

- ***the frame rate (frames/second)***
- ***interlaced and progressive encoding***
- ***video interframe compression algorithms and spatial and temporal redundancy***
- ***multimedia container formats***

Number of frames per second

Frame rate, the number of still pictures per unit of time of video, ranges from six or eight frames per second (*frame/s*) for old mechanical cameras to 120 or more frames per second for new professional cameras. PAL standards (Europe, Asia, Australia, etc.) and SECAM (France, Russia, parts of Africa etc.) specify 25 frame/s, while NTSC standards (USA, Canada, Japan, etc.) specify 29.97 frames. Film is shot at the slower frame rate of 24 frames per second, which slightly complicates the process of transferring a cinematic motion picture to video. The minimum frame rate to achieve a comfortable illusion of a moving image is about sixteen frames per second.

Interlaced vs progressive

Video can be interlaced or progressive. Interlacing was invented as a way to reduce flicker in early mechanical and CRT video displays without increasing the number of complete frames per second, which would have sacrificed image detail to remain within the limitations of a narrow bandwidth. The horizontal scan lines of each complete frame are treated as if numbered consecutively, and captured as two *fields*: an *odd field* (upper field) consisting of the odd-numbered lines and an *even field* (lower field) consisting of the even-numbered lines.

Analogue display devices reproduce each frame in the same way, effectively doubling the frame rate as far as perceptible overall flicker is concerned. When the image capture device acquires the fields one at a time, rather than dividing up a complete frame after it is captured, the frame rate for motion is effectively doubled as well, resulting in smoother, more lifelike reproduction (although with halved detail) of rapidly moving parts of the image when viewed on an interlaced CRT display, but the display of such a signal on a progressive scan device is problematic.

NTSC, PAL and SECAM are interlaced formats. Abbreviated video resolution specifications often include an *i* to indicate interlacing. For example, PAL video format is often specified as *576i50*, where *576* indicates the total number of horizontal scan lines, *i* indicates interlacing, and *50* indicates 50 fields (half-frames) per second.

In *progressive scan* systems, each refresh period updates all scan lines in each frame in sequence. When displaying a natively progressive broadcast or recorded signal, the result is optimum spatial resolution of both the stationary and moving parts of the image. When displaying a natively interlaced signal, however, overall spatial resolution is degraded by simple line doubling—artefacts such as flickering or "comb" effects in moving parts of the image appear unless special signal processing eliminates them. A procedure known as deinterlacing can optimize the display of an interlaced video signal from an analogue, DVD or satellite source on a progressive scan device such as an LCD Television, digital video projector or plasma panel. Deinterlacing cannot, however, produce video quality that is equivalent to true progressive scan source material.

Video compression method (digital only)

Uncompressed video delivers maximum quality, but with a very high data rate. A variety of methods are used to compress video streams, with the most effective ones using a Group Of Pictures (GOP) to reduce spatial and temporal redundancy. Broadly speaking, spatial redundancy is reduced by registering differences between parts of a single frame; this task is known as *intraframe compression* and is closely related to image compression. Likewise, temporal redundancy can be reduced by registering differences between frames; this task is known as *interframe compression*, including motion compensation and other techniques. The most common modern standards are MPEG-2, used for DVD, Blu-ray and satellite television, and MPEG-4, used for AVCHD, Mobile phones (3GP) and Internet.

Video capture card

A video picture is made up of a series of images that change approximately 26 times per second. A **video capture card** is an analogue-to-digital converter which reads the analogue signals from the video camera or video tape and digitises them ready for storage in a computer. Many video cameras are now digital cameras and consequently do not need a video capture card, but video capture cards are still used to provide other functions such as colour correction capabilities. The algorithm which processes the signal and prepares it for storage and use is called the "codec". There are many different codecs that use different algorithms to do the same job. This means that video which was prepared using one codec can only be re-created by using the same codec.

A video capture card fits into one of the expansion slots of a computer system and allows the processor to store the values of the screen pixels for a specific picture. In other words, it allows the action to be frozen. A typical example of the use of a video capture card is the selection and storage of a single frame taken from a video file. If a computer is able to receive and display television pictures, the video capture card could store a single picture frame.

Streaming

If the file is a video clip, transmission time does not matter if the file is going to be watched after the recipient has done some other work. It begins to matter if the file is a video that the recipient wants to watch as it is being transmitted. This technique is called **streaming**. For streaming, it is important that the file can be transmitted quickly enough to let the person

watch the video as it is being transmitted. In other words, the data must arrive quickly enough to compose and display the media. If the bit rate is not high enough then the video will appear to judder, or even to freeze, while the computer waits for the next download of data. This difference between the files that makes the baud rate so important is not the size of the file, but how it is used. If a file is going to be watched in real time as it is received then the data are said to be "time sensitive". Time-sensitive files need a high baud rate, not necessarily just large files. Much recent research has been done on streaming as computers are increasingly used for the download and viewing of media files.

Multimedia container formats

The container file is used to identify and interleave different data types. Simpler container formats can contain different types of audio formats, while more advanced container formats can support multiple audio and video streams, subtitles, chapter-information, and meta-data (tags) — along with the synchronization information needed to play back the various streams together. In most cases, the file header, most of the metadata and the synchro chunks are specified by the container format. For example, container formats exist for optimized, low-quality, internet video streaming which differs from high-quality DVD streaming requirements.

Container format parts have various names: "chunks" as in RIFF and PNG, "atoms" in QuickTime/MP4, "packets" in MPEG-TS (from the communications term), and "segments" in JPEG. The main content of a chunk is called the "data" or "payload". Most container formats have chunks in sequence, each with a header, while TIFF instead stores offsets. Modular chunks make it easy to recover other chunks in case of file corruption or dropped frames or bit slip, while offsets result in framing errors in cases of bit slip.

Flexible containers can hold many types of audio and video, as well as other media. The most popular multi-media containers are:

- 3GP (used by many mobile phones; based on the ISO base media file format)
- ASF (container for Microsoft WMA and WMV, which today usually do not use a container)
- AVI (the standard Microsoft Windows container, also based on RIFF)
- DVR-MS ("Microsoft Digital Video Recording", proprietary video container format developed by Microsoft based on ASF)
- Flash Video (FLV, F4V) (container for video and audio from Adobe Systems)
- IFF (first platform-independent container format)
- Matroska (MKV) (not limited to any codec or system, as it can hold virtually anything. It is an open standard and open source container format).
- MJ2 - Motion JPEG 2000 file format, based on the ISO base media file format which is defined in MPEG-4 Part 12 and JPEG 2000 Part 12
- QuickTime File Format (standard QuickTime video container from Apple Inc.)
- MPEG program stream (standard container for MPEG-1 and MPEG-2 elementary streams on reasonably reliable media such as disks; used also on DVD-Video discs)
- MPEG-2 transport stream (a.k.a. MPEG-TS) (standard container for digital broadcasting and for transportation over unreliable media; used also on Blu-ray Disc video; typically contains multiple video and audio streams, and an electronic program guide)
- MP4 (standard audio and video container for the MPEG-4 multimedia portfolio, based on the ISO base media file format defined in MPEG-4 Part 12 and JPEG 2000 Part 12) which in turn was based on the QuickTime file format.
- Ogg (standard container for Xiph.org audio format Vorbis and video format Theora)

- RM (RealMedia; standard container for RealVideo and RealAudio)

1.1.5 Compression techniques

Show understanding of how digital data can be compressed, using either 'lossless' (including runtime encoding - RTE) or 'lossy' techniques

When data other than text is being transmitted, e.g. on the Internet, it is important to limit the amount of data that needs to be sent to stop the time taken to download the data being unreasonably long. The amount of data can be limited by reducing the file size of pictures so that they take up only a small part of the screen or restricting them to a few colours. Speeding up the transmission of the data is achieved by reducing the amount of data that is sent. This is known as **file compression**.

Compression can be either lossy or lossless. Lossless compression means that no data is lost.

Lossy Compression

Lossy compression involved sacrificing some of the data in order to reduce the file size. Lossy compression techniques reduce the quantity of data in two ways. First by using complex mathematical encoding and secondly by deliberately losing some types of visual information that our eyes and brain usually ignore (this is called *quantization*).

For example, the video frame rate may be reduced from the normal 25 frames per second down to around 15 frames per second before there is a perceptible loss in quality. The frames themselves may be treated as separate still images, and compressed individually using JPEG compression. Different areas of a frame may be compressed by different degrees – an area of blue sky which lacks detail might be compressed by 25:1, whereas a person's face might only be compressed by 5:1. Depending on the amount of action in the video, only some areas of each frame may change from one frame to the next and only the changed data need be stored. The size of the picture may also be reduced, reducing the overall quantity of pixels to be stored.

If lossy compression is taken to an extreme, it can result in a significant loss of picture quality. The higher the compression ratio, the worse the resulting image. For instance, colour fidelity fades and the edges of objects become very obvious, until eventually the results is unwatchable.

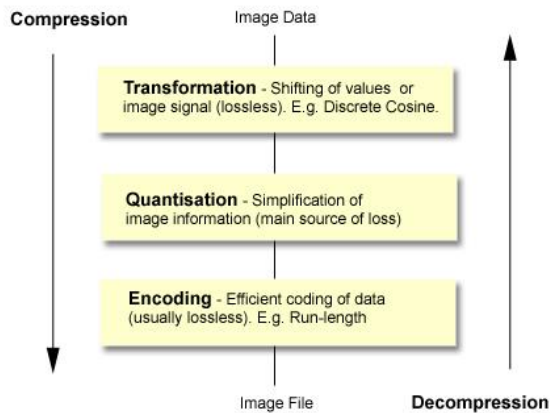
JPEG image compression works in part by rounding off nonessential bits of information. There is a corresponding trade-off between preserving information and reducing size. A number of popular compression formats exploit these perceptual differences, including those used in music files, images, and video.

Lossy image compression can be used in digital cameras, to increase storage capacities with minimal degradation of picture quality. Similarly, DVDs use the lossy MPEG-2 Video codec for video compression.

Runtime Encoding (RTE)/Run-length Encoding (RLE)

Run-length encoding (RLE) is a very simple form of data compression in which *runs* of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs: for example, simple graphic images such as icons, line drawings, and animations. It is not useful with files that don't have many runs as it could greatly increase the file size.

Run-length encoding performs lossless data compression and is well suited to palette-based bitmapped images such as computer icons. It does not work well at all on continuous-tone images such as photographs, although JPEG uses it quite effectively on the coefficients that remain after transforming and quantizing image blocks.



Lossless Compression

Lossless compression uses mathematical techniques such as Huffman coding or Discrete Cosine Transformation (DCT), to reduce the quantity of information to be stored, while still being capable of reproducing the original image without any loss in quality.

Lossless data compression algorithms usually exploit statistical redundancy to represent data more concisely without losing information, so that the process is reversible. Lossless compression is possible because most real-world data has statistical redundancy. For example, an image may have areas of colour that do not change over several pixels; instead of coding "red pixel, red pixel, ..." the data may be encoded as "279 red pixels". This is a basic example of run-length encoding; there are many schemes to reduce file size by eliminating redundancy.

Compression standards include MPEG, M-JPEG, Cinepak, Intel's Indeo Video Interactive (IVI), Apple's Quicktime, Microsofts Directshow

Bibliography

- Numerous pages from www.wikipedia.org e.g. http://en.wikipedia.org/wiki/Multimedia_Container_Format
- New Higher Computing by John Walsh (Hodder & Stoughton)
- Cambridge International AS and A Level Computing coursebook by Leadbetter, Blackford and Piper (Cambridge University Press)
- A Level Computing 5th Edition by Heathcote & Langfield (Payne-Gallway Publishers Ltd)
- Revise AS Computing – The ultimate study guide by Roger Legg (Letts)
- Numerous sections referenced throughout these notes http://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise