Playing With Pictures

Teacher Notes to support Tenderfoot Unit 4: Bits And Bytes – The digital advantage

An introduction to bitmap images, resolution, hexadecimal and html colour codes with practical challenges manipulating pixel values to change images. A look at some of the technology behind digitising and rendering images to introduce Bresenham's and Demosaicing algorithms.

Preparation required:
Digitising Images Resource Sheets for each person.
Pixel Spreadsheet available on all computers (no installation needed).
Binary Crossbin and Tarzia puzzles available as exemplars.

Graphic Design

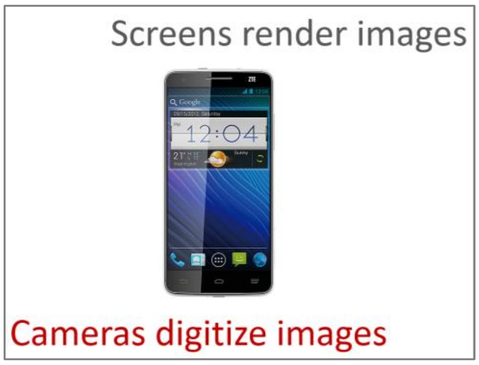
Selecting
Enhancing
Layering
Feathering

Graphic design can be stunning. It can provoke a lot of interest, with children asking just 'how do they do that?' An interesting article with links to good galleries of examples can be found at goo.gl/Q92bnR. If we take a simple picture, with just a grasp of the basics, children should be able to create similar effects in any image manipulation application. If you don't currently use one in school, GIMP is a powerful, free package similar to Photoshop (www.gimp.org/) and is used in a later session. Decomposing a picture is a useful activity. The example picture in the presentation has been built up using 4 basic techniques (selection, layering, enhancing and feathering). Firstly the handshake has been selected and cut from the original photograph. The image has been enhanced through applying a tint to different areas and it has been layered on top of a background, again created from a green rectangle with a circle removed and the edge feathered. Mastering these simple techniques allows for artistic creativity. The theory of how digital pictures are stored sits very well with creative image manipulation. We don't need to throw the IT baby out with the bathwater of curriculum change but understand the basics of digitisation and digital rendering help deepen their understanding of their creative endeavours. In this session we start by outlining the basics, introduce hexadecimal and html colour codes before looking at an activity to manipulate pixel values.

Understanding Images

Phones are a good starting point with children; a subject dear to their hearts. The presentation is designed to get pupils to consider how to evaluate a screen amidst many confusing measurements.

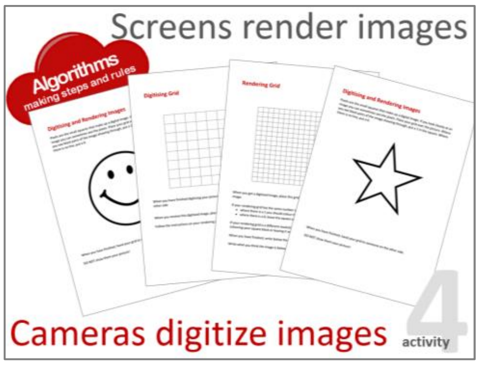
The Galaxy has the biggest screen, but is it the best? Ascertain whether pupils know how the measurement is derived (across the diagonal). In the second example the Lumia has the biggest number, but what does PPI stand for? (Pixels per inch). Only on the third example slide do we have the actual resolution of the display.



We also need to distinguish between the resolution of a display, and the resolution of the images captured. The presentation unpicks what 'megapixels' are, and how often these aren't a precise measurement. The number of pixels in an image can be derived from multiplying the height and width of the resolution. In our case, we demonstrate a 10 megapixel image being captured by the camera.

The key initial distinction to make is between screens that render images, and cameras which digitize images.

Digitising Images



A short activity, based on one first developed by CSInside (an outreach project from Glasgow University) is for children to try to digitize images at different resolutions. The class is split into two groups on either side of the room. One half receive the Star picture, the other half the face. They need to keep the image hidden from the other side. They each receive two grids. The activity can work on two levels.

As a simple activity with young children, they are tasked with digitising an image. Placing a grid over their picture they label each square with a 1 or 0 reflecting the image underneath.

They then swap with someone on the other side who has a similar resolution grid. Each tries to render the other's image. When everyone has rendered the coded image, the results for the group can be compared. This makes the simple point that the higher the resolution, the clearer the representation.

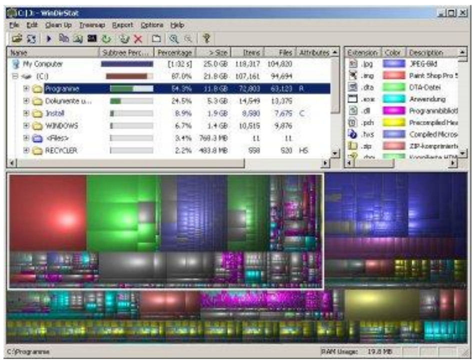
This can lead to a discussion of how to best digitise an image. Is it always best to code a 1 where black appears, even if it is a tiny proportion of the square? Can we represent curves better? Perhaps there are better rules to adopt? Let them decide the rules, but insist it is applied consistently. The process is known as rasterization. In graphics software digitising a line / curve etc. is usually achieved using a variant of Bresenham's algorithm. The maths involved is too much for KS3 but there is a nice online demonstration of rasterization at goo.gl/BxXJqe which makes the point. Many techniques have been developed to try to represent detail in limited resolution. For example, where shades of grey are possible, these too can be employed to represent part of a pixel (anti-aliasing), but this often leads to a blurred effect.

A more complex adaptation of the activity is to swap digitised images randomly. In this case the recipient will receive one of three possibilities:

- an image digitised at the same resolution as they intend to render.
- an image digitised at a lower resolution than their rendering grid or
- an image digitised at a higher resolution than they can render.

The first two present no further challenges – the child has no choice in how they render the data. The point can be made that an image cannot be displayed at a higher resolution than it was digitised. In the third case though, children are presented with the challenge of how to render clusters of 4 pixels. This is worthy of a class discussion. Again, there are no simple answers, other than a consistent approach being called for. However, it can be used to highlight real challenges in computer graphics, often referred to as LOD (Level of Detail) or Mipmapping. As a figure, for example, moves into the distance in a game, the pixel representation of it has to be reduced, presenting a similar challenge to the one we are considering.

Once resolution is understood, we can introduce the idea of file size. Which would be bigger – a picture or an entire novel? Estimate 12 words per line, 7 or 8 letters on average in a word, and about 40 lines per page. That's around 90 bytes (characters) per line, so 3,600 bytes per page. A thousand pages would be 3,600,000 bytes – around 3.5 MB. An uncompressed 10 megapixel picture would be about 8 or 9 times bigger. In simple form, a moving image can be considered like a 'flick book' – a series of images. Multimedia files, which include sound as well, will be even larger.



At this point, pupils may well ask how video can be streamed. Transmission of sound, pictures and video would not be possible without clever compression algorithms, explored later.

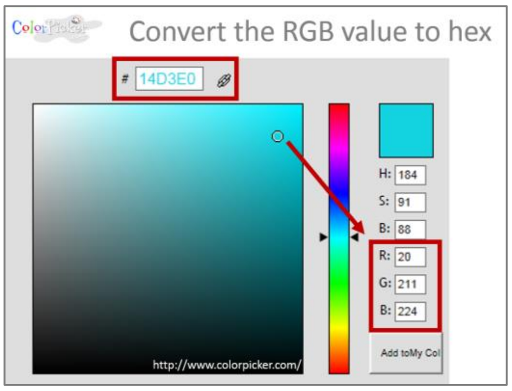
WinDirStat (windirstat.info) is a free utility that displays files visually by size and type. It can be very useful for children to begin to appreciate varying file sizes associated with different media. There are versions available for Mac and Linux too.

Understanding Colour

When an image is rendered, each pixel is lit by a combination of 3 colours; red, green and blue (RGB). RGB is an additive colour model; that is the light is added to a black screen. By contrast, when a picture is printed on white paper, light has to be removed to render the correct colours. Three translucent colours, Cyan, Magenta and Yellow (CMYK – the K represents black) are combined in this subtractive colour model.

A slide gives an explanation of full colour representation. Ask how many bits (or bytes) are required to store each pixel value. 1 byte will be needed for each colour (because 255 is represented in binary as 1111 1111), so 3 bytes (or 24 bits) are required. In actual fact, a 4th byte usually stores a value for transparency as well. You can use the presentation as a class teaching tool, with children being asked to fill in the binary values. The following slides have answers for each of the 3 examples.

We often see colour values written in decimal RGB values like the examples, but children may also have come across other forms. This is an opportune time to introduce hexadecimal. Decimal originated because we have ten fingers – what system might we have used if we had 16 fingers? Encourage children to come up with any sensible ideas. We want to encourage an understanding of place values. The key point is we only have ten numeric digits in our language, so we need to agree a set of symbols to represent digits equating to 10, 11, 12, 13, 14 and 15. Clearly in base16, 10 means something very different to 10 in base10. The presentation introduces hexadecimal and initially some practice converting to / from binary. We want the children to spot a pattern in quartets of bits. The zero in hex equates to the zero in the first 4 bits of the binary number. It is the same every time it occurs. The F digit in hex equates to 15 in binary … and it is the same every time. Now consider what happens when we get to bigger numbers. Notice that each 1 in the binary '16' column signifies 1 lot of 16 in hex … and it is the same every time. The 10 in binary, is 2 in hex, and 100 is 4 in hex. Regardless of the column headings, by treating each nibble as a binary number, we can easily convert it to the hexadecimal equivalent. So the final number 1111 is 15 in binary, which equates to F in hex.

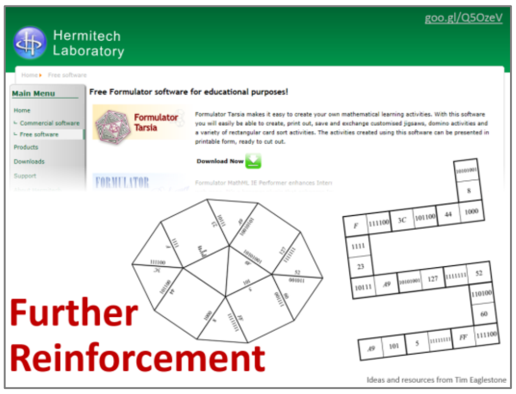

Convert the RGB value to hex

14D3E0

H: 184
S: 91
B: 88
R: 20
G: 211
B: 224

http://www.colorpicker.com/

Our blue binary pixel value is very long and not easy to remember. Hexadecimal provides a convenient shorthand for long binary numbers. Let's convert it nibble by nibble. Notice how the column values have been changed to make it easy to convert each nibble. 336666 is the hexadecimal equivalent of 001100110110011010010110. Using an online colour picker, we can easily find the RGB value. Once the students have converted it, reveal the value in the box – this is the hex equivalent, which is also the html colour code. Children are often familiar with these, but their composition is often a revelation.


goo.gl/Q5OzeV

Hermitech Laboratory

Free Formulator software for educational purposes!

Formulator Tarsia makes it easy to create your own mathematical learning activities. With this software you will easily be able to create, print out, save and exchange customised jigsaws, domino activities and a variety of rectangular card sort activities. The activities created using this software can be presented in printable form, ready to cut out.

Further Reinforcement

Ideas and resources from Tim Eaglestone

There are two good homework exercises from Gary Kacmarcik included to practice hex / binary conversion. Repeated exposure to simple binary representation and conversion is important for laying a solid foundation. Tarzia is free software from Hermitech Laboratory (http://goo.gl/Q5OzeV). It is good for making matching games that pairs or tables of students can complete. CAS Master Teacher, Tim Eaglestone has shared examples of two formats, a domino type game and a jigsaw/shape type pattern for matching. They are included in the resources. They include binary and hexadecimal conversion. Why not download the software and have a go preparing some of your own?

Pixel Spreadsheet

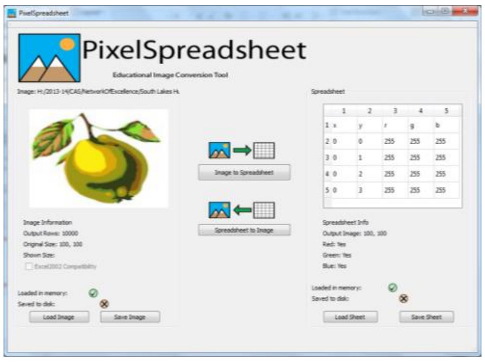


Image: H:/2013-14/CAS/NetworkOfExcellence/South Lakes H.

Spreadsheet

	1	2	3	4	5
1	x	y	r	g	b
2	0	0	255	255	255
3	0	1	255	255	255
4	0	2	255	255	255
5	0	3	255	255	255

Image Information
Output Rows: 10000
Original Size: 100, 100
Shown Size:
☐ Excel2002 Compatibility
Loaded in memory: ✓
Saved to disk: ⊗

Spreadsheet Info
Output Image: 100, 100
Red: Yes
Green: Yes
Blue: Yes
Loaded in memory: ✓
Saved to disk: ⊗

Pixel spreadsheet (http://goo.gl/Gv9Drk) is a utility developed by Mark Guzdial at Georgia Tech, which extracts the RGB pixel values from an image file and converts them into a spreadsheet. We can use the spreadsheet's functionality to alter the values, before saving as an image file. We load an image using a standard file browser interface. Note the image file details in the example – it is only a small image 100 pixels by 100, but that is still 10,000 pixels, whose RGB values are output on a separate row for each pixel.

Encourage the children to use the small image files supplied. The button converts the image file to a spreadsheet … which should be saved. The spreadsheet can then be opened and inspected. Note the column values, the first two indicating the x and y co-ordinates of the pixel, followed by the RGB values. The spreadsheet slides help you to explain how co-ordinates are numbered from the top left corner and how to manipulate cell values. In the example we set the green value to zero. As there are 10000 rows, using fill down is the easiest way to copy the value. We can then load the saved spreadsheet and convert to an image… … which should also be saved. The thumbnail shows the changed image. Notice too that we can swap colour channels simply by renaming them. The order they appear doesn't matter, it is the name of the column that is important. When it loads, the colour values will still be listed in the correct order. Try to get children to predict what the result of swapping the values on will be, before viewing the result.

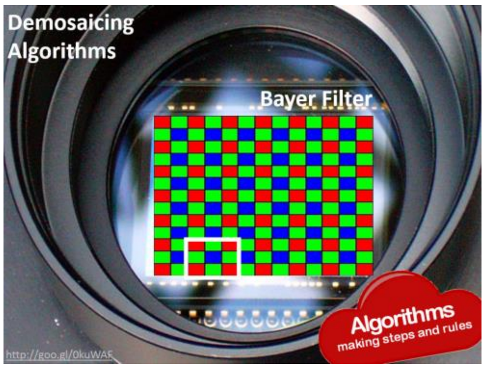
There are 5 short supporting videos. These take you through the basic operations. Three challenges are given; turning a colour image to greyscale, creating a negative and applying a filter to different quadrants of an image. All are quite challenging but the videos will provide answers if students are stuck. The first is linked in the slide, but all are available from https://goo.gl/B7iWP9.

How It Works

We've established that bitmap pictures are stored as a list of numbers, but how do screens render these as images? And how do cameras digitize the image as a series of numbers in the first place?

An excellent 2 minute video explaining how LCD screens work can be found at youtu.be/0B79dGR19Tg. Another good video for classroom use (4 minutes) emphasising what a revolution LCD was can be found at youtu.be/8ZIq5jpiY4M. The Think Maths website, initiated by Matt Parker, has a wonderful utility, whereby you upload an image and download a spreadsheet with RGB values in each cell: goo.gl/5aRdfF. One issue is that the site doesn't acknowledge your photo has uploaded. Simply select the download option to retrieve the spreadsheet. Zooming out reveals the image. To investigate how this works: Select a cell, from the *Home tab*, select *Conditional Formatting – Manage Rules*.

How is an image digitised in the first place. The technology is essentially the same as that used in solar panels. When light falls on a panel it is converted to an electrical current. In a camera, as light hits an array of Charged Coupled Devices (CCD) each of them generates a small voltage that can be measured and the number stored. The analogue image is thus converted to digital data.



Each CCD has a filter so collects only one particular colour value. Note that a Bayer Filter has twice as many green filters as red and blue. Although each pixel has only one colour value, the camera computer runs a demosaicing algorithm to average the values surrounding the pixel, creating a full colour value for each one. More details can be found at goo.gl/0kuWAF. The particulars of a demosaicing (or Bresenham's) algorithm are less important than the fact that they illustrate the primacy of algorithms in 'real world' processes.