COMPSCI 111/111G
Digital Images and Vector Graphics

Lecture 13 SS 2018

The Seine and La Grande Jatte – Springtime
George Seurat 1888

Learning Outcomes

Students should be able to:

- Describe the differences between bitmap graphics and vector graphics
- Calculate the size in bytes of a bitmap image
- Compare and contrast different compression methods (jpeg, gif and png)

Bitmap Graphics

Storing pictures digitally

- Sample the image (divide into dots)
- Image resolution (number of dots)

200 x 250

40 x 50

20 x 25

http://en.wikipedia.org/wiki/Raster_graphics

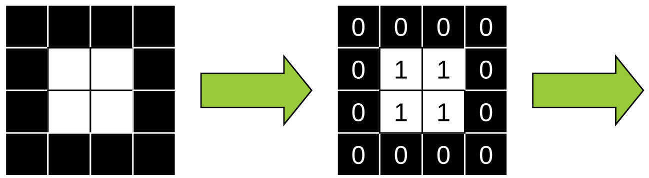
Black and White pictures

Digital Pictures consist of small dots

- Each dot is called a picture element (pixel)

Storing information

- Black and White are only two states
- Use bits to represent pixels (0 = OFF, 1 = ON)
- One to one mapping, so known as Bitmap



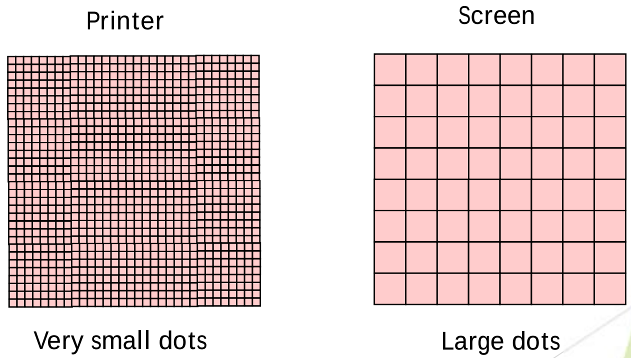
0000011001100000

http://en.wikipedia.org/wiki/Pixel

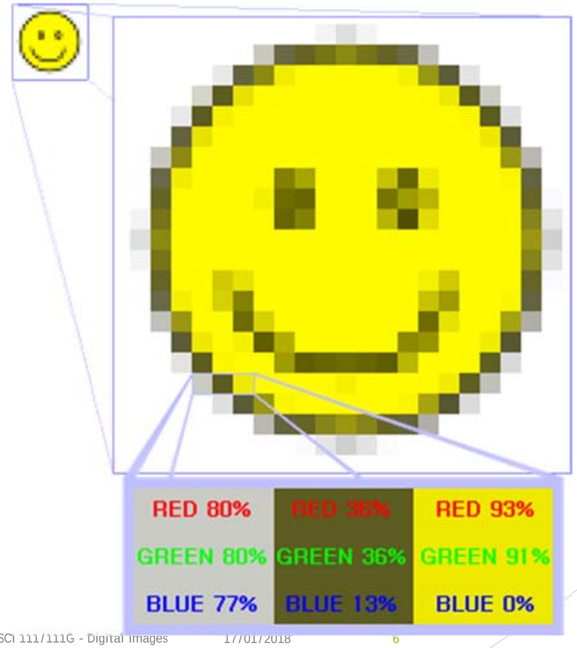
Displaying images

Images are displayed on an output device

- Screen / Printer
- Physical devices have limitations



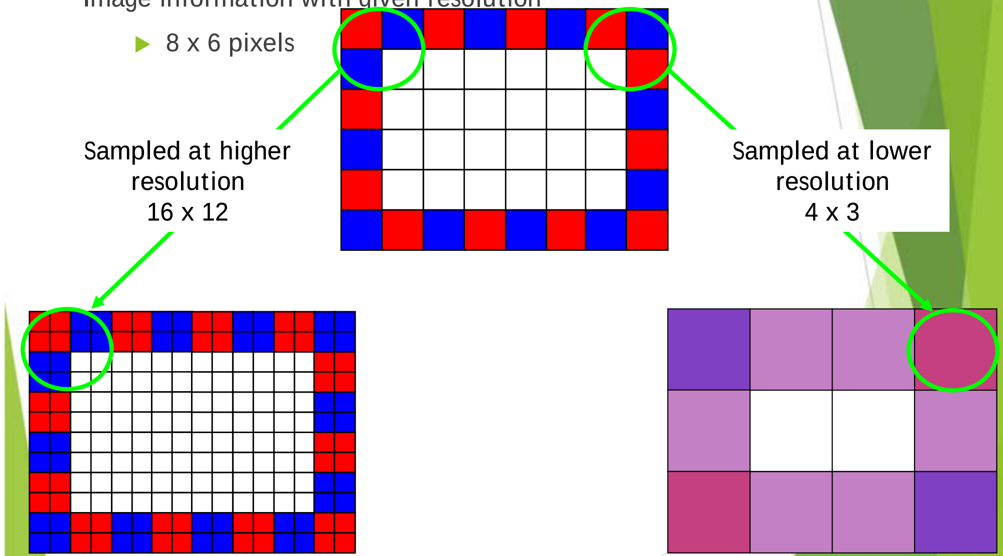
Resizing bitmap images



Resizing images

Image information with given resolution

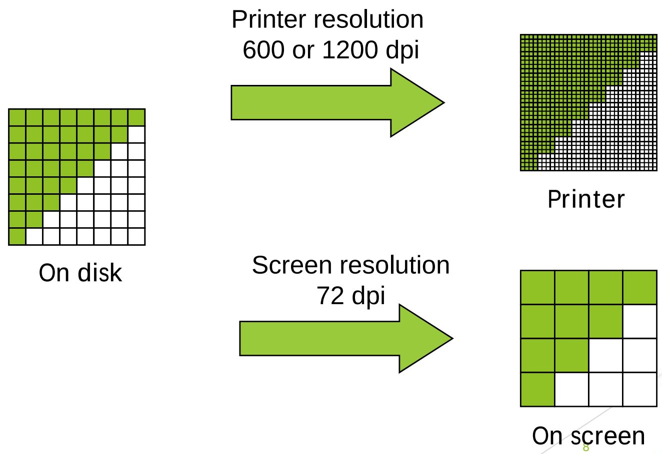
- 8 x 6 pixels



Printing Bitmaps

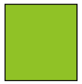
Printer and Screen have different sized dots

- Scale (resample) the bitmap to ensure it looks good on both



Exercises

Imagine you have taken a picture with a 4 megapixel digital camera. For ease of calculation, assume that the picture is square, not rectangular.



4 million pixels

Assume that you are printing this picture out on a printer that has approximately 4000 dots per inch. How many inches across would the picture be when it was printed?

If you viewed this image on a screen that had 1000 dots across, what portion of the image would be visible?

Colour Bitmaps

Colours

- Use more than 1 bit per pixel
- Map the binary number to a colour

1100	0010	1111	1111
1010	0101	0010	1111
1000	0111	0000	1101
0110	1111	1110	1010

Each pixel uses 4 bits

Bits	**Colour**
0000	**Black**
0001	**Red**
0010	**Green**
0011	**Blue**
0100	**Yellow**
…	**…**

Colour table used for display

How much memory is required?

One binary number used for each pixel

- 1 bit 2 colours
- 2 bits 4 colours
- 4 bits 16 colour
- 8 bits 256 colours
- 16 bits 65536 colours
- 24 bits 16,777,216 colours

How many bits are required for a 16 colour image 100 pixels wide x 8 pixels high?

- 100x8x4 = 3200 bits = 400 bytes

An image using 24 bit colour, 1000 wide x 1000 high (1 Megapixel)?

- 3 MB

Exercises

- How many colours can be represented by 3 bits?
- How many bits are required to represent 128 different colours?
- How much memory would be required to store a black and white image that is 10 pixels high and 5 pixels wide? Show your working.

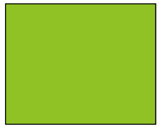
Exercises

- How much memory (in bytes) would be required to store an image that has 256 different colours and is 3 pixels high and 5 pixels wide? Show your working.

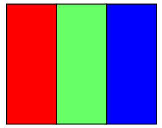
Displays

Screens use a combination of Red, Green and Blue lights

- RGB colour



A single pixel at distance



A single pixel close up

Use one byte (8 bits) for each colour

- 256 different levels of red brightness
- 256 different levels of green brightness
- 256 different levels of blue brightness

Compressing Images

Simply reducing number of colours

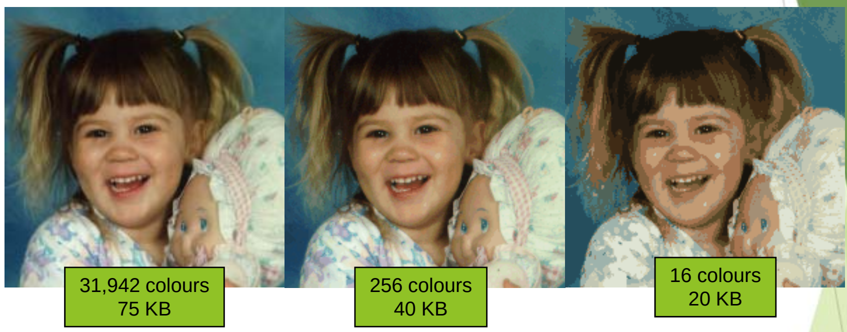


Image is 200 pixels wide, 200 pixels high

= 40,000 pixels

Compression Algorithms

Graphics Interchange Format (GIF)

- Lossless method
- 256 colours
- Good for graphics, poor for photos
- Uses an algorithm that was patented

Image Size: 200x100
Original (256 colours): 20KB
GIF (256 colours): 3KB

Image Size: 200x200
Original (256 colours): 40KB
GIF (256 colours): 32KB

http://en.wikipedia.org/wiki/Gif

Compression Algorithms

Portable Network Graphics (PNG)

- Replacement to GIF
- Lossless method
- 16 million colours (24 bit)
- Good for graphics, poor for photos

Image Size: 200x100
Original (256 colours): 20KB
PNG (16M colours): 4KB

Image Size: 200x200
Original (16M colours): 120KB
PNG (16M colours): 68KB

http://en.wikipedia.org/wiki/Png

Compression Algorithms - JPEG

Joint Photographic Experts Group (JPEG)

- Lossy method
- 16 Million colours (24 bit)
- Averages nearby colours
- Different degrees of compression
- Good for photos, poor for graphics

Image Size: 200x100
Original: 60KB
JPEG (50%): 5KB

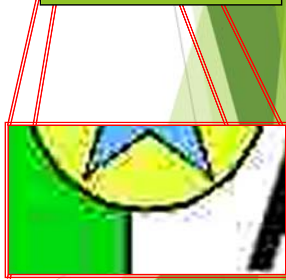


Image Size: 200x200
Original: 120KB
JPEG (50%): 6KB

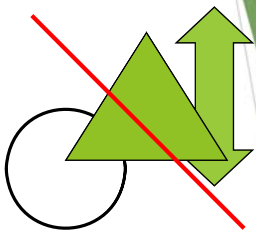
Image Size: 200x200
Original: 120KB
JPEG (99%): 2KB

http://en.wikipedia.org/wiki/jpeg

Vector Graphics

Object-oriented graphics

- Objects created independently
- Defined by mathematical formulae



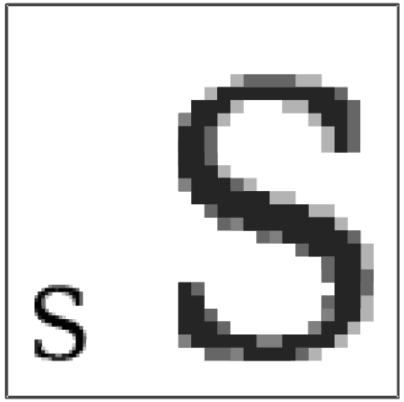
Advantages

- Very small memory requirements
- Memory independent of the image size
- Scale to any size without loss of quality

Object Type: Square
Height: 100
Width: 100
Position_X: 354
Position_Y: 289
Fill Colour: Light Blue

http://en.wikipedia.org/wiki/Vector_graphics

Bitmap and Vector Graphics



Bitmap
.gif, .jpg, .png

Vector Graphics
.svg

Scalable Vector Graphics

Format for representing vector graphics images

- Open standard created by W3C
- New, gaining popularity
- XML, text file similar to HTML

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
 <svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1"
width="520" height="520"> <style type="text/css"> <![CDATA[ text{font-size:362px;font-
weight:bold;font-family:"Times New Roman", serif} #P0 {fill:#d4a000;stroke:#000;stroke-width:9} #P1
{fill:url(#tl)} #P2 {fill:url(#bl)} #P3 {fill:url(#br)} #P4 {fill:url(#tr)} ]]> </style> <defs> <linearGradient
id="dk"> <stop/> <stop style="stop-opacity:0" offset="1"/> </linearGradient> <linearGradient id="lt">
<stop style="stop-color:#ffe681"/> <stop style="stop-color:#ffe681;stop-opacity:0" offset="1"/>
</linearGradient> <linearGradient x1="136.4" y1="136.4" x2="167.5" y2="167.5" id="tl" xlink:href="#lt"
gradientUnits="userSpaceOnUse"/> <linearGradient x1="136.4" y1="383.6" x2="167.5" y2="352.5"
id="bl" xlink:href="#lt" gradientUnits="userSpaceOnUse"/> <linearGradient x1="383.6" y1="383.6"
x2="352.5" y2="352.5" id="br" xlink:href="#dk" gradientUnits="userSpaceOnUse"/> <linearGradient
x1="383.6" y1="136.4" x2="352.5" y2="167.5" id="tr" xlink:href="#dk"
gradientUnits="userSpaceOnUse"/> </defs> <path id="P0" d="M260,6.3L 6.3,260L 260,513.7L
513.7,260L 260,6.3z"/> <text y="380" x="200">!</text> <path id="P1" d="M260,12.7L 260,75L 75,260L
12.7,260L 260,12.7z"/> <path id="P2" d="M260,507.3L 260,445L 75,260L 12.7,260L 260,507.3z"/>
<path id="P3" d="M260,507.3L 260,445L 445,260L 507.3,260L 260,507.3z"/> <path id="P4"
d="M260,12.7L 260,75L 445,260L 507.3,260L 260,12.7z"/>
</svg>
```

http://en.wikipedia.org/wiki/Svg

Summary

Bitmap Images

- Pixel width x pixel height = resolution
- Use numbers to encode colour of each pixel (more colours = more bits per pixel)
- Look jagged when enlarged too much
- Take a lot of memory but can be compressed (e.g. JPG)

Vector Images

- Defined by mathematical formulae
- Can be enlarged and still look nice
- Small compared to bitmap images