Graphics 2

Practical Graphics Issues

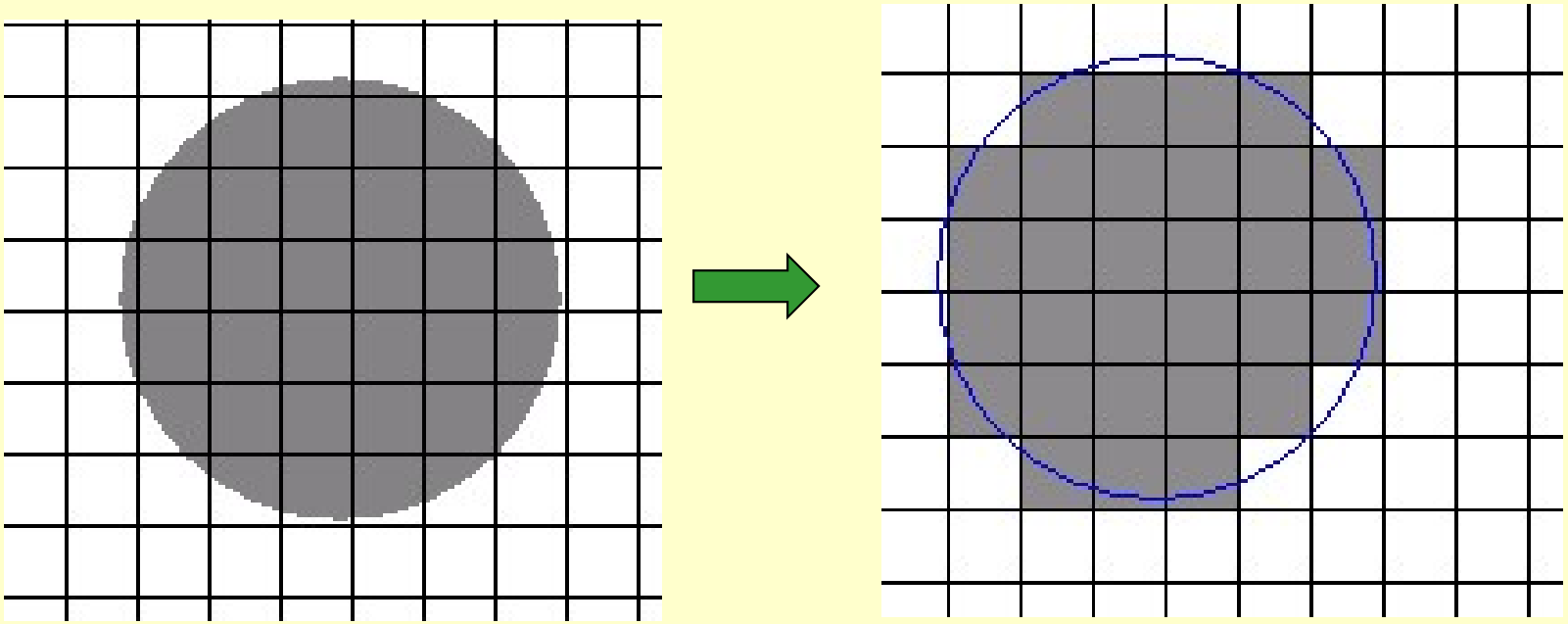
Overview

- Digitizing and Rendering
  - From image to bitmap
- Anti-aliasing
  - What is it and how to do it?
    - Lines
    - Shapes
- Fonts
  - Anatomy and types of a fonts
  - How to make fonts look nice
- Image manipulation
  - Pixel selection
  - Pixel point and group processing

Cunliffe & Elliott, ch 7; Chapman & Chapman, ch 9

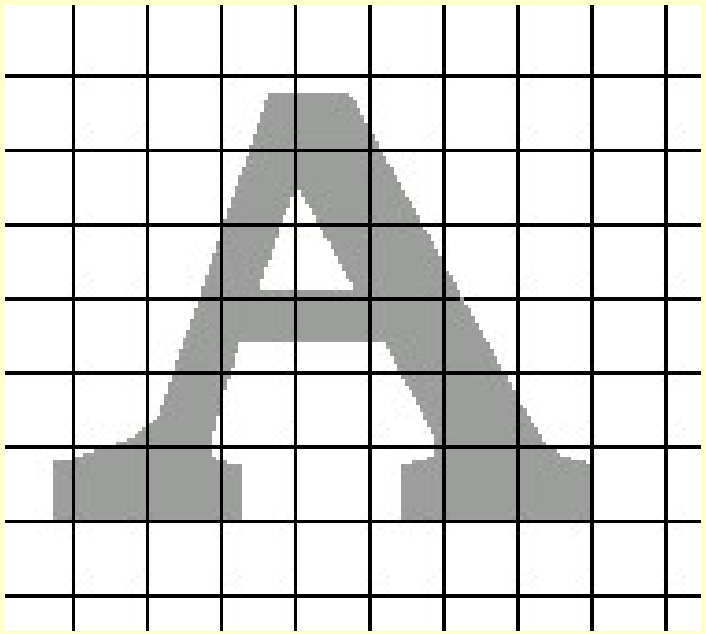
Digitizing Pictures

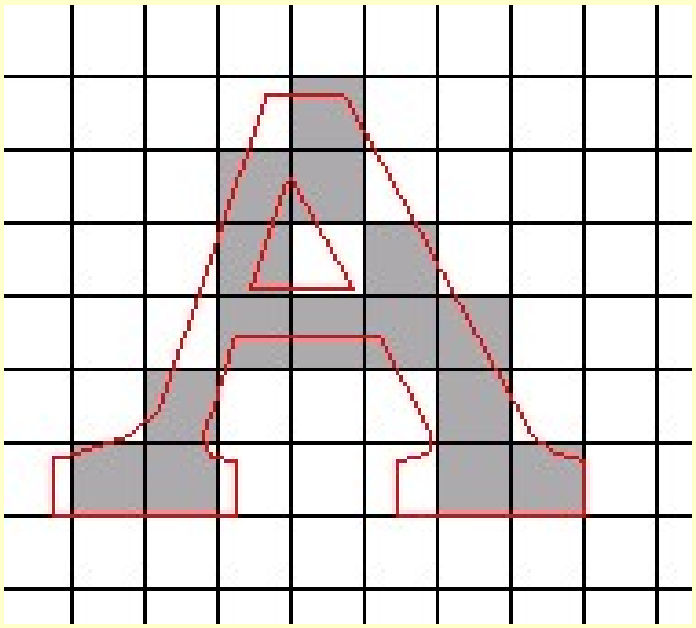
- How do we go from a continuous image to a bitmap?
- A picture is sampled and the bits of the bitmap set according to what colour "wins" for that pixel



Digitizing Pictures

Same principle for text production:



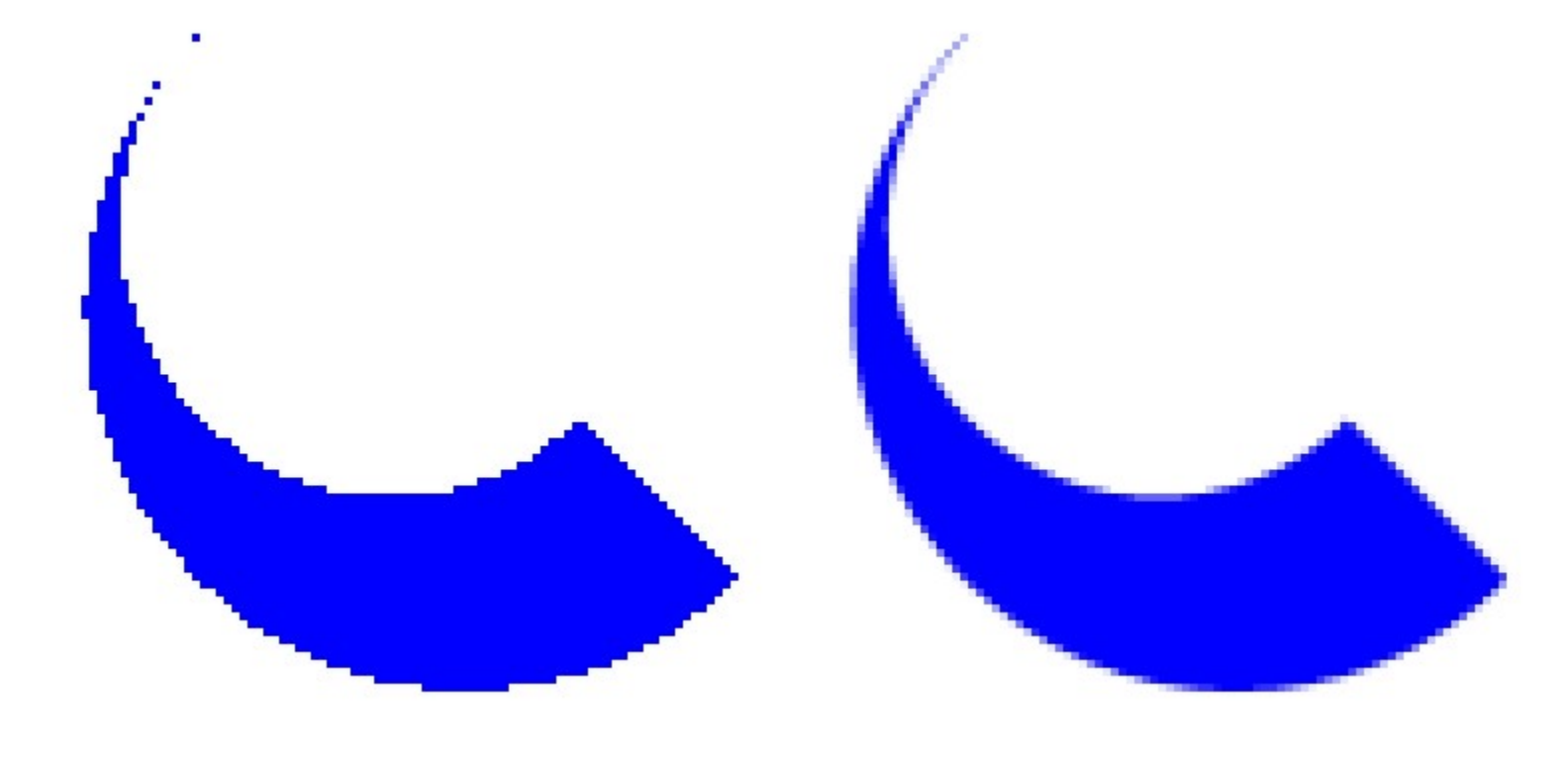


Vector graphics → bitmap conversion

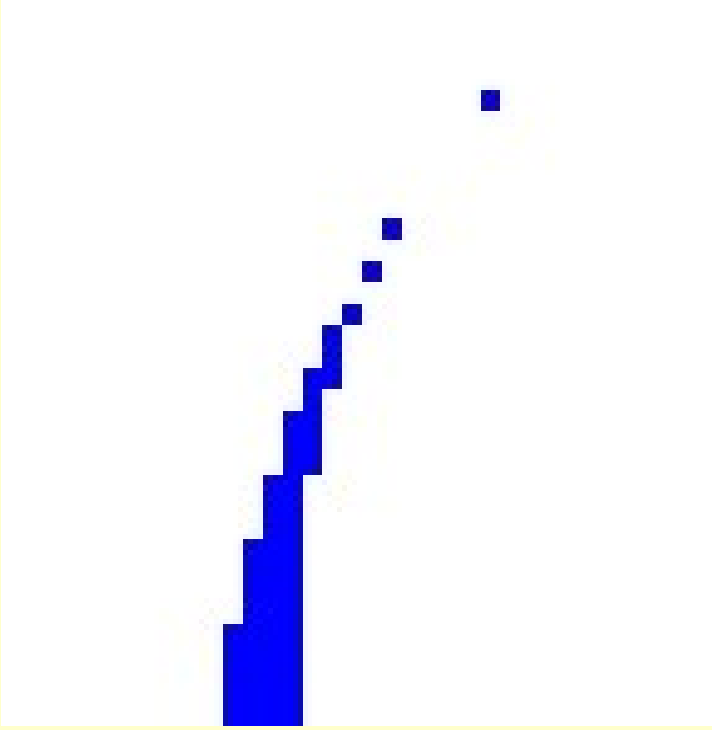
- First step in rendering an image

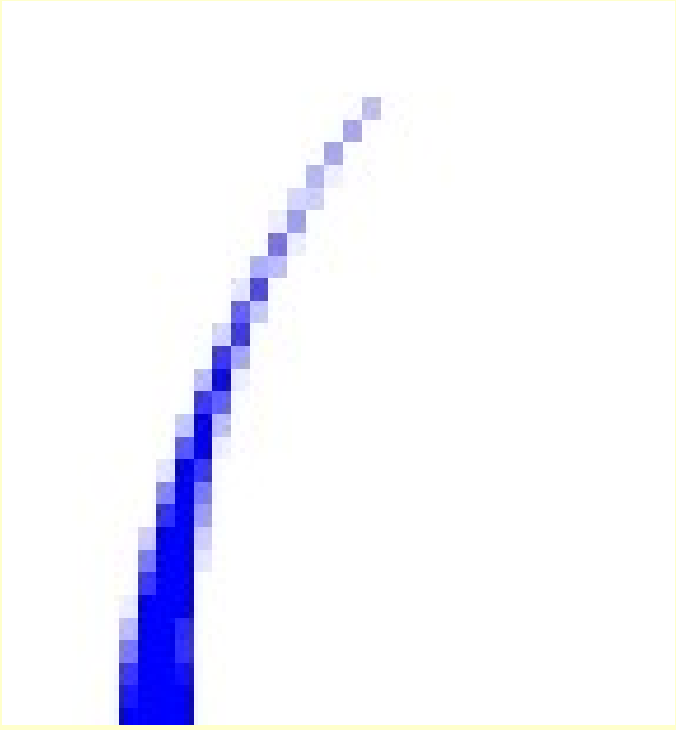
Anti-aliasing

- Previous examples use only two colours
- The results can be more visually appealing if a range of colours is used at the edges of an object

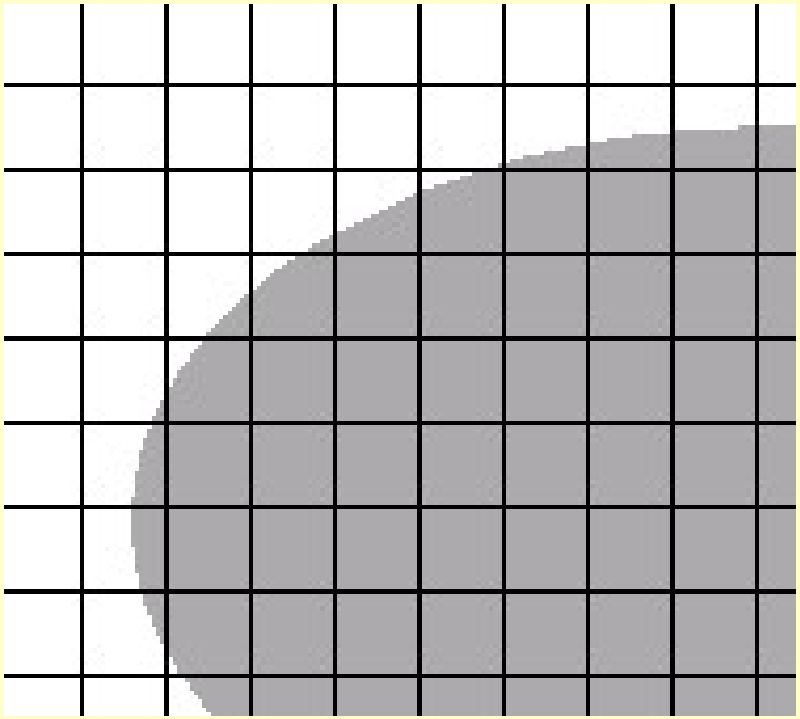


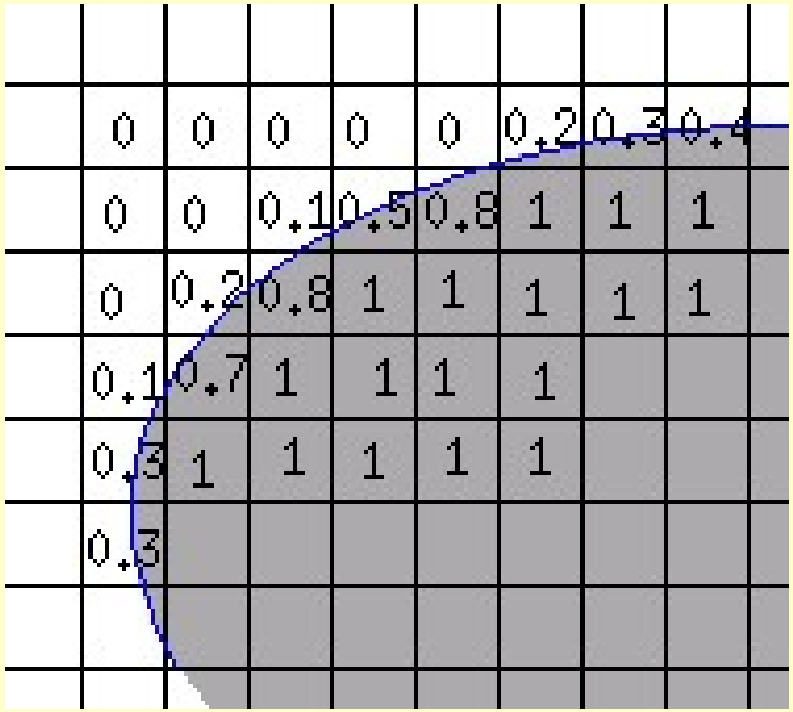
(a close-up)





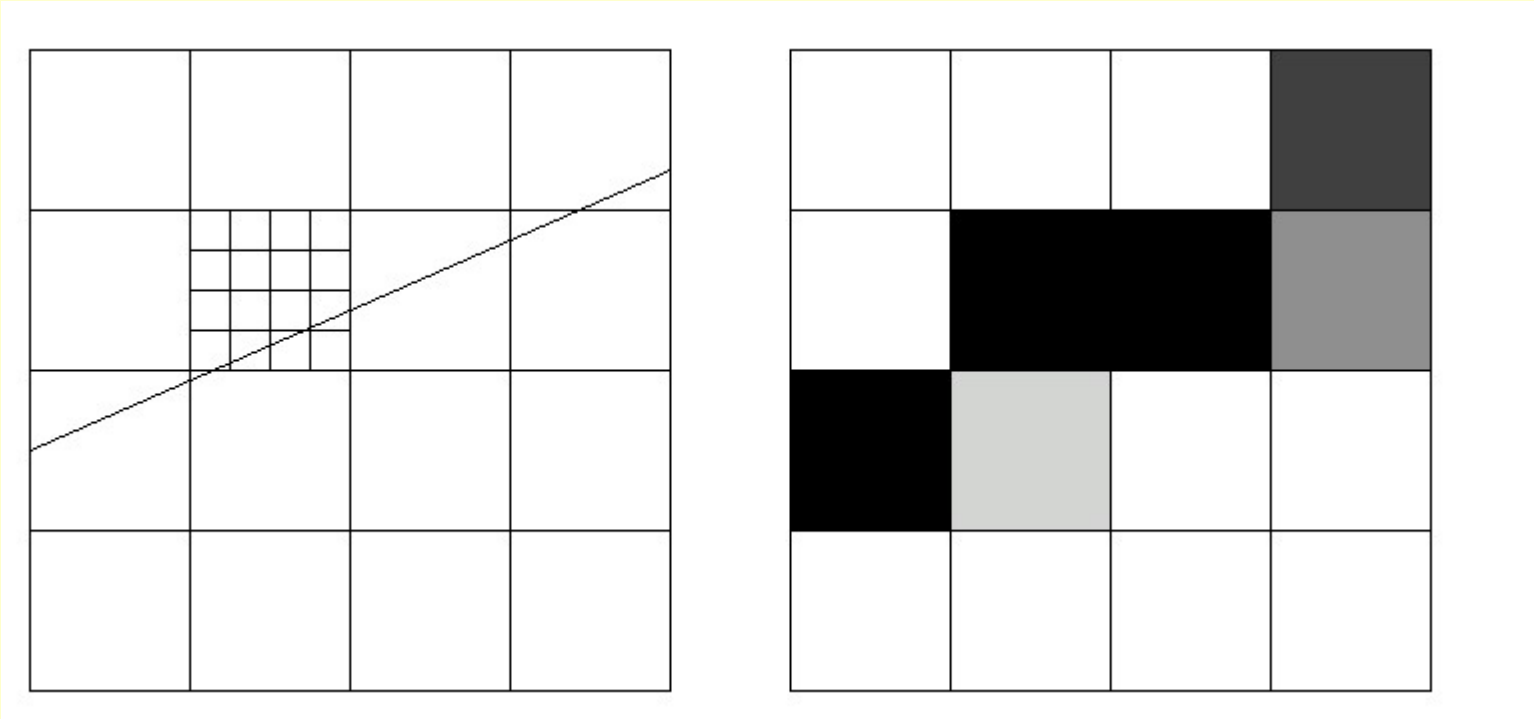
Shapes





Opaqueness in proportion to coverage of shape

Lines

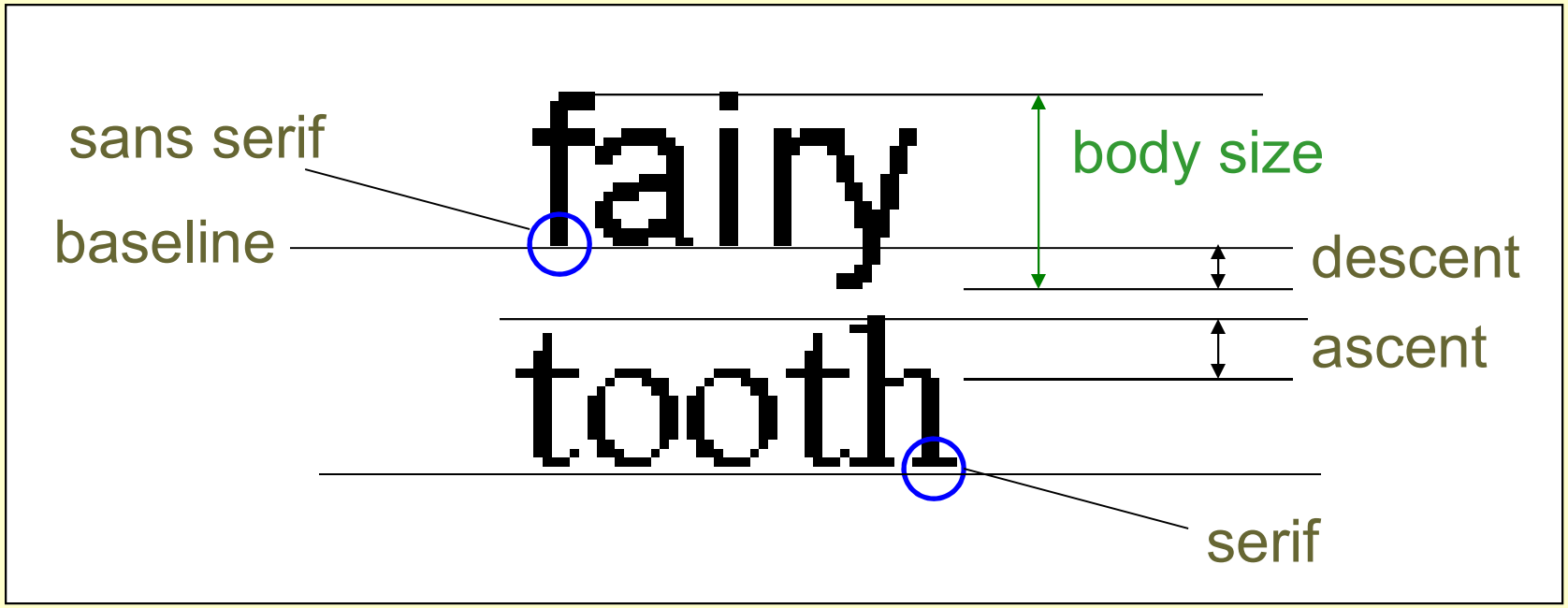


Opaqueness in proportion to how many subpixels the line passes through

Fonts

- Historically there were no *fonts* (styles of text)
- Characters were just characters, displayed on the
  - screen (fixed number of columns and rows)
  - or printer (dot matrix, daisywheel)

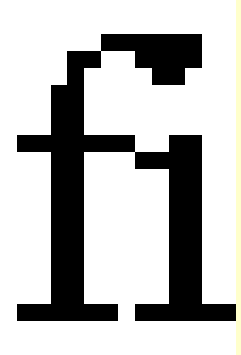
Anatomy of a Font



Point size is a measure of the body size (the distance from the top of the highest character to the bottom of the lowest character) in 72nds of an inch

Fonts

- Not to be confused with character sets!
- Fonts are different ways (styles) of representing characters
- e.g. to get the Greek letter alpha,
  - ‘a’ in “Symbol” font is displayed as ‘α’
  - α can also be obtained as character **`03B1x`** of the Unicode character set
- Not a 1-1 correspondence between characters and font shapes (glyphs)
  - some characters consist of more than one glyph
  - some glyphs represent a combination of characters:

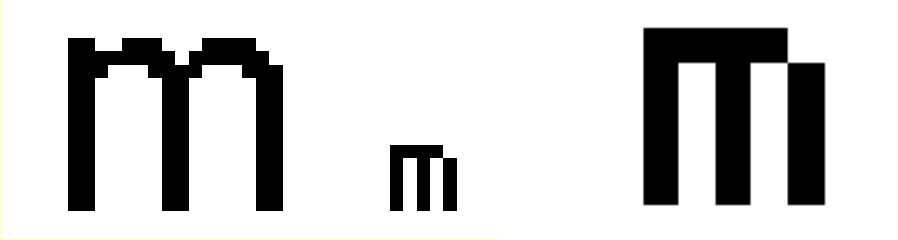


Some Common Fonts

- Arial (sans serif, TrueType)
- Times New Roman (serif, TrueType)
- `Lucida Console (TrueType, fixed-width)`
- **`Courier New (serif, TrueType, fixed-width)`**
- `Courier` (serif)
- **Script (Script, a script font)**
- Σψνβολ (Symbol, a symbol font)
- ♋ ♦♓■♑♎♓■♑♦ (Wingdings, a symbol font)

Bitmapped Fonts

- Each glyph is represented as a bitmap
- Scaling doesn't work!



- Entire font families required (how fonts were used in early word processors)

8 point 10 point 12 point 14 point 16 point 18 point

20 point 24 point 36 point

72 point

Vector-based Fonts

- Nowadays, vector-based fonts have taken over
  - referred to as scalable or outline fonts
- These are defined in terms of:
  - a set of component shapes (straight lines and curves)
  - hints (see later)
- Scalable fonts are converted to bitmapped glyphs as needed
  - for rendering on a screen or a printer

- Aside:
  - Java's fonts in Java2D are scalable fonts
  - These use Unicode as the character set

Displaying Fonts Nicely

- Solutions from printing industry
- Solutions specific to computer-based fonts
- Ways to do this include:
  - Non-fixed-width
  - Anti-aliasing
  - Hinting
  - Kerning (spacing)

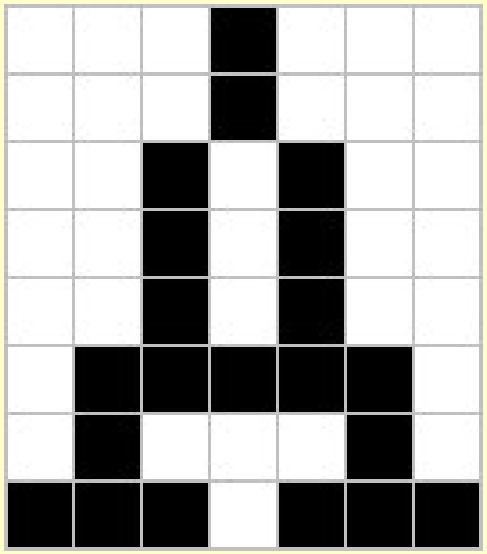
Anti-aliasing Text

Anti-aliasing Text

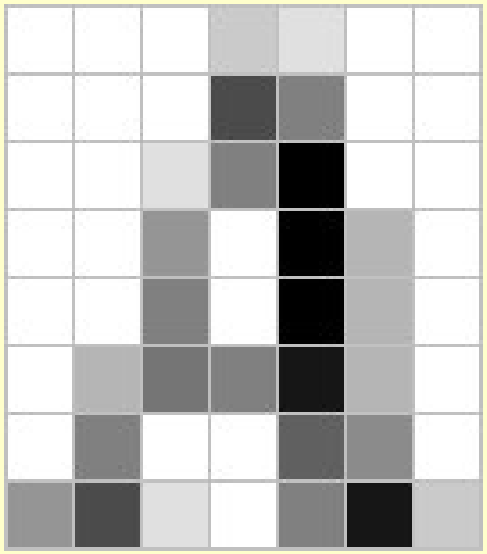
It's *not always better*:

Here's a 8pt plain 'A' in Times New Roman A

enlarged

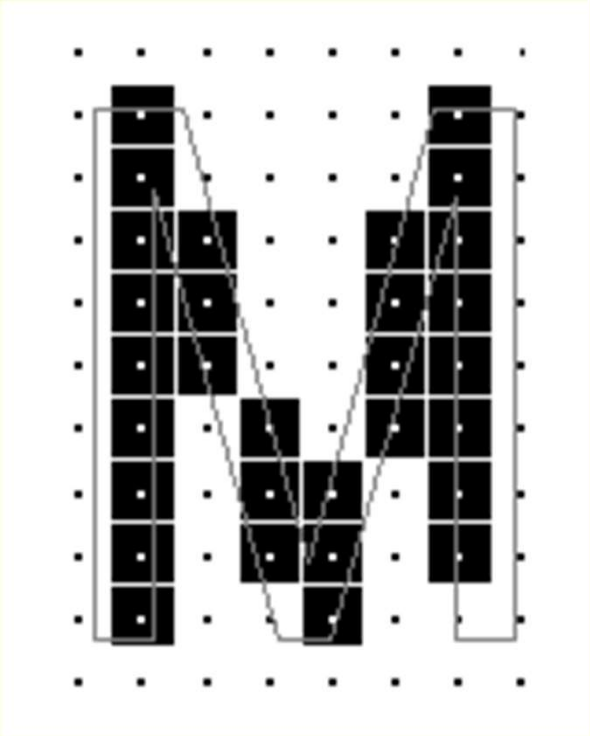


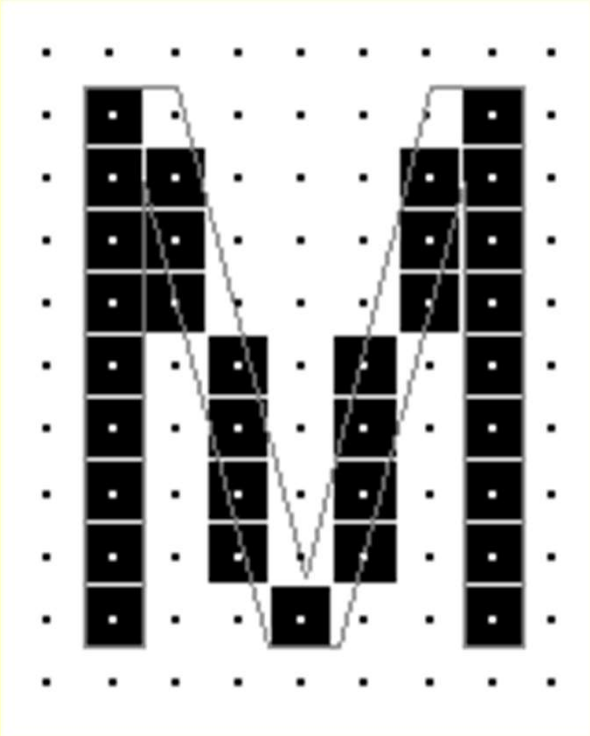
anti-aliased



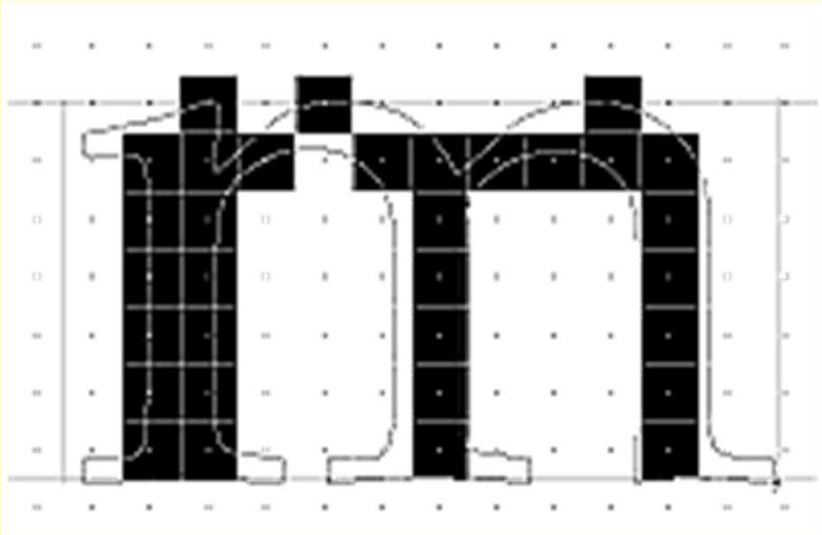
Hinting

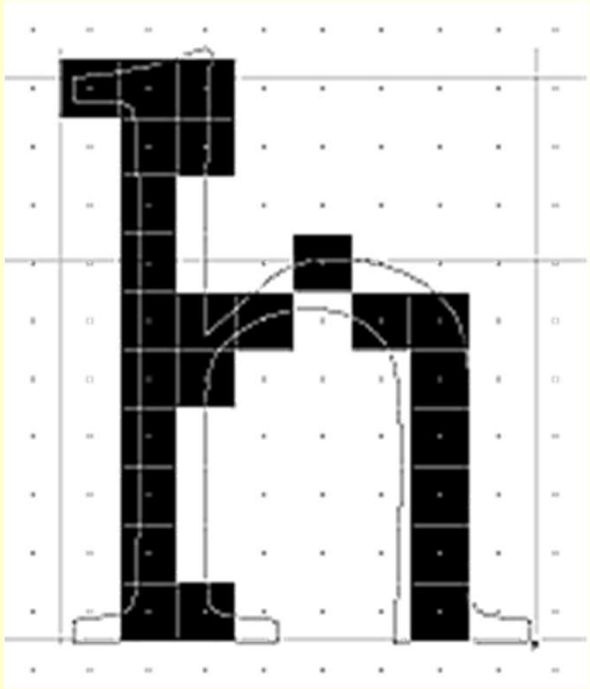
- Problem: display of small-sized text
- Digitizing may result in unevenness

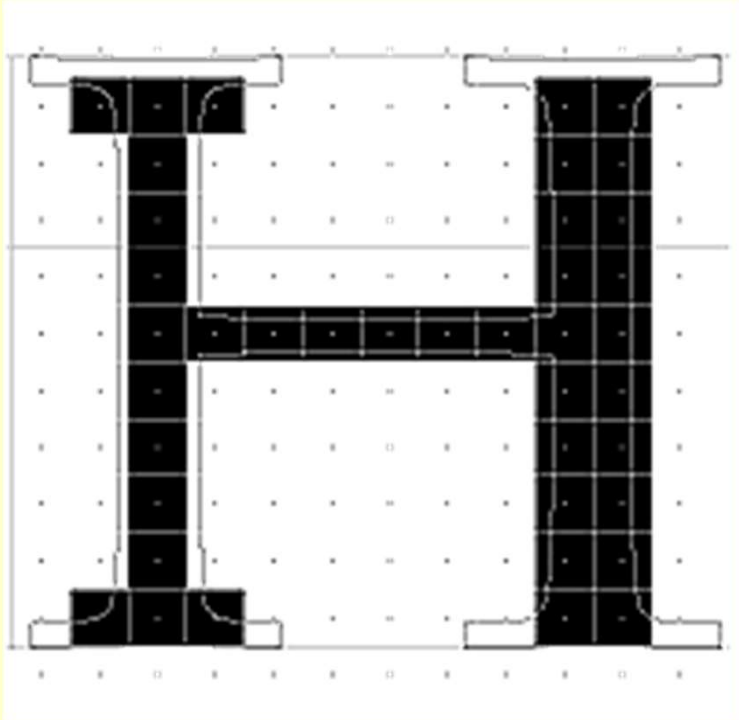




More examples:







Hinting

- *Hinting* is the provision of hints as to how to display fonts in certain circumstances
- A font may be defined without hints, but having hinting results in a better quality font

A Microsoft hint to leave vertical and horizontal strokes solid!

Kerning

- Improves text appearance by changing the spacing of adjacent text (even to the point of overlapping)
- Done by use of *kern pairs*
- A high quality font (such as TrueType fonts) have suitable kern pair settings
- TrueType fonts may have anything from 100 to 500 kern pairs (150 is typical)

Without kerning:

With kerning:

Font Foundries: *the rivals*

Postscript

- A PDL (page description language) for printers
- Often hardware-implemented in a printer
- Provides many fonts (more than 20,000)

TrueType

- A collection of software-implemented, high quality vector-based fonts
- Suitable for non-printing uses
- Converted to bitmaps which are then sent to a printer

Font foundries

- Other fonts required can be purchased!

Bitmap Image Manipulation

- Pixel selection
- Pixel point processing
- Pixel group processing

Chapman & Chapman, chapter 4

Pixel Selection

- Basic marquee tools
  - Rectangle, ellipse, hand-drawn curves
- Photoshop's *magic wand*
  - Clicking on a pixel causes all adjacent pixels of a *similar* colour to be selected
  - Level of similarity can be adjusted
- Photoshop's *magic lasso*
  - Draw around the area to be selected with the mouse
  - Boundary of selection snaps to *edges* within a specified distance
  - Degree of contrast that defines an edge can be adjusted

Pixel Selection (2)

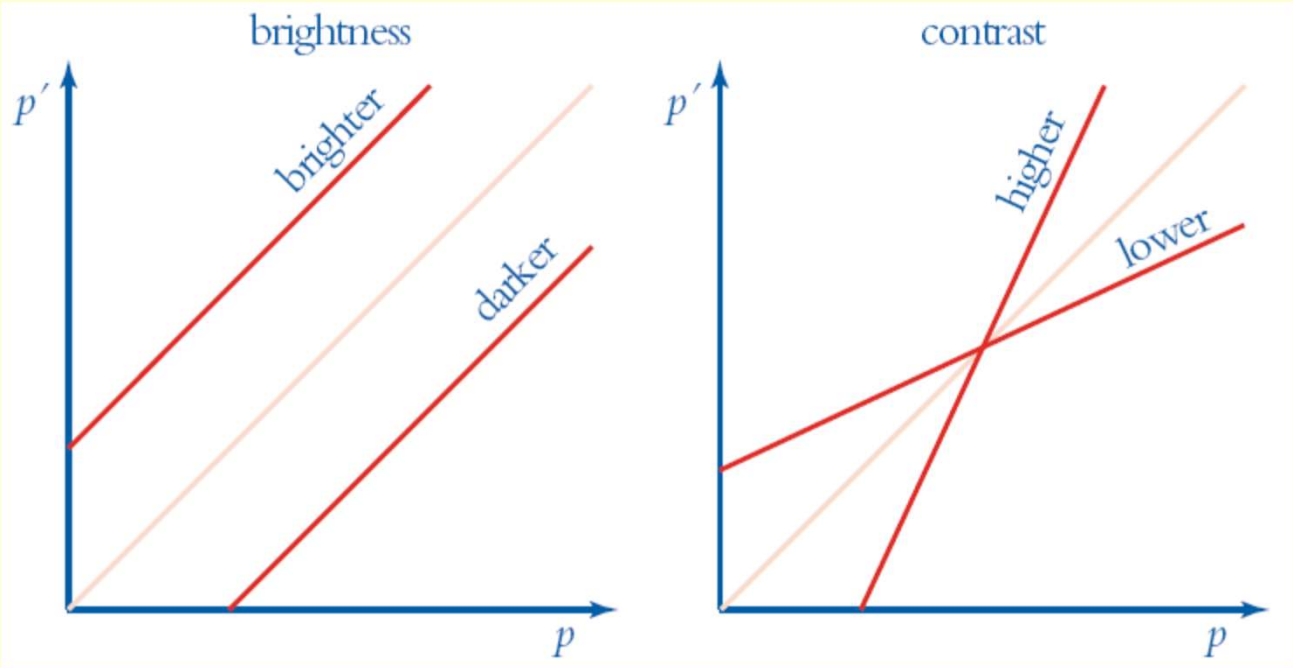
- Magic wand selection

- Magic lasso selection

(Images © MacAvon Media Productions)

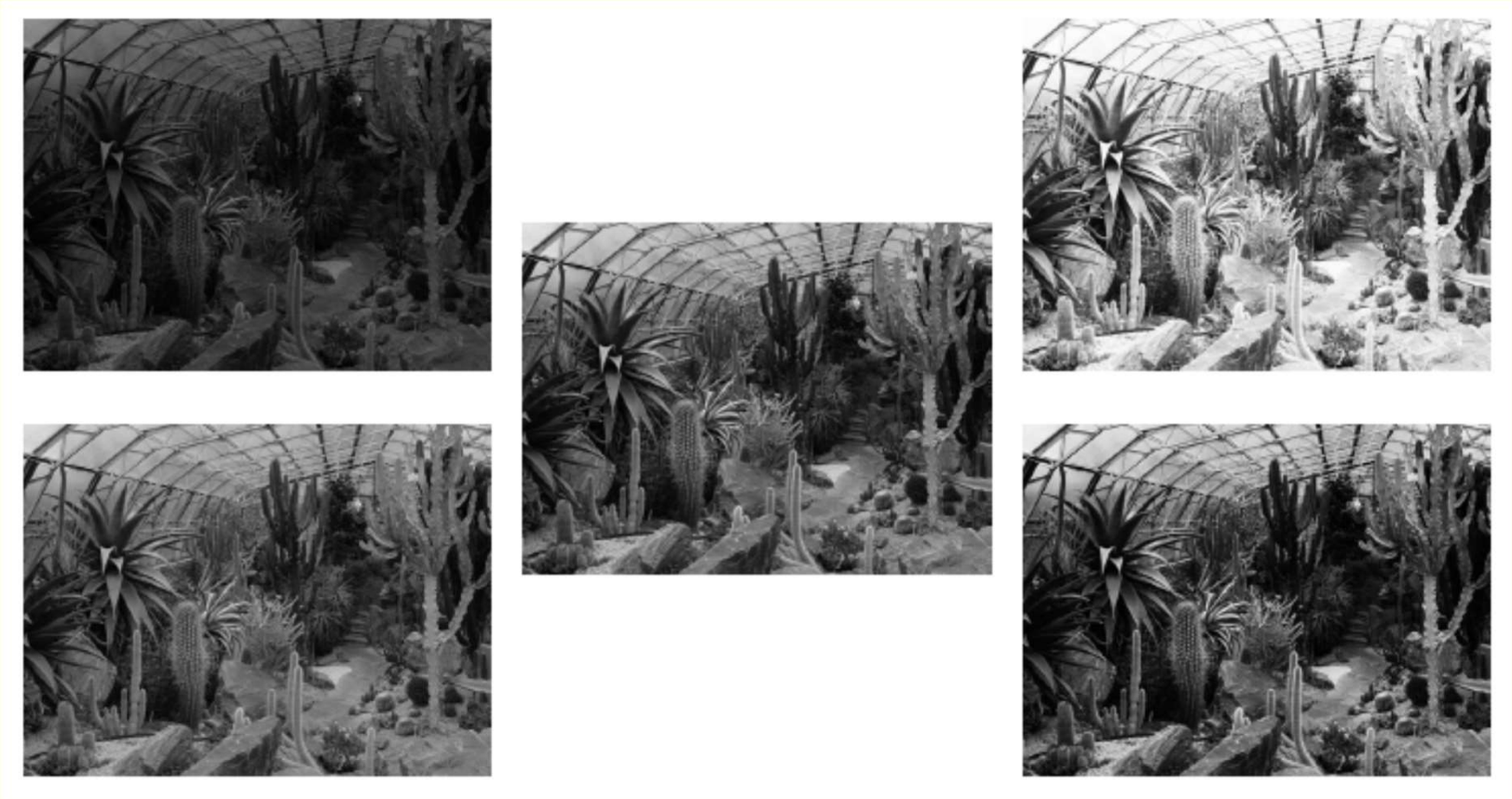
Pixel Point Processing

- Redefining individual pixel colours
  - Independent of the colour of neighbouring pixels
  - Mapping function: p'=f(p)
  - E.g. negative of a grayscale photo: p'=W-p
- Major use is systematically recolouring a photo
  - Changing brightness or contrast level



Brightness and Contrast

- Top: brightness
- Bottom: contrast
- Centre: original image

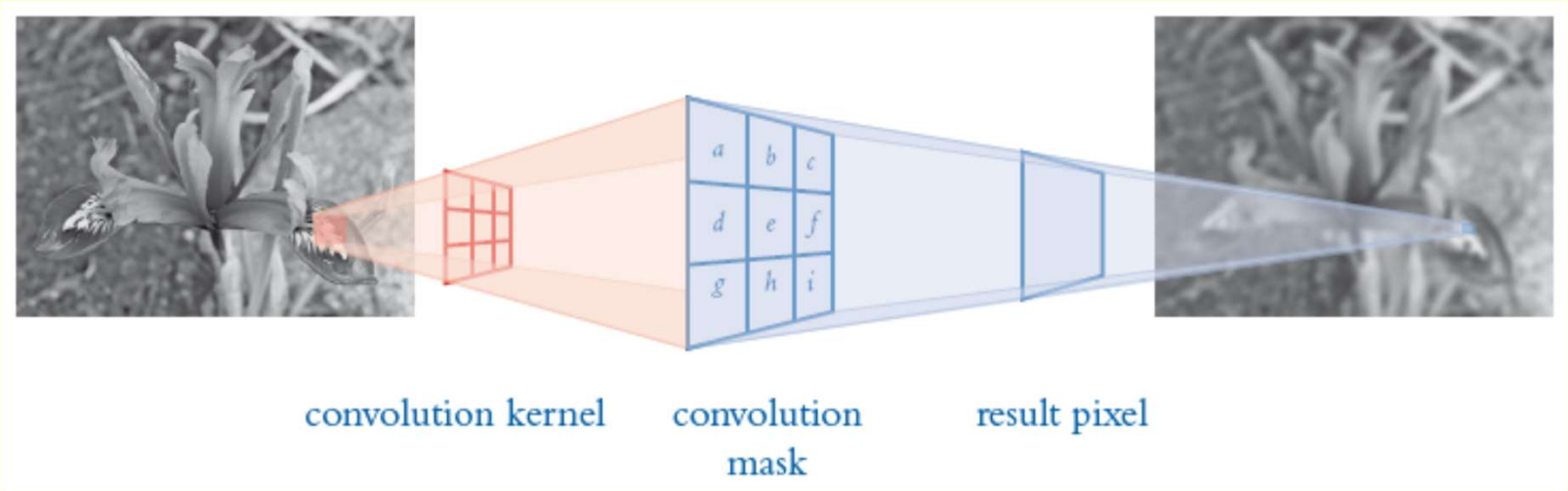


Pixel Group Processing

- Redefining pixel colours in groups
  - colour depends on neighbouring pixels
- Applying a spatial filter
  - Antialiasing, blurring, sharpening
- Applying arbitrary image filters
  - Warping, morphing etc

Blurring with Convolution Mask

- Pixel colour is a function of immediate neighbours
  - $p' = ap_{x-1,y+1} + bp_{x,y+1} + cp_{x+1,y+1} \cdots$
  - If a=b=c=d=e=f=g=h=i=1/9 :



(Images © MacAvon Media Productions)

End of Lecture