# Graphics

## Representing Graphical Data

*Chapman & Chapman, chapters 3,4,5*
*Richardson*

1

## Graphics in IT82

*What does "computer graphics" cover?*

**IT82**

- Input, output, and representation of graphical data
- Creation of graphics
  - two-dimensional (flat) images
  - three-dimensional "scenes", with the modelling of objects in virtual worlds
- Manipulation of existing data
- Graphics programming, with (e.g.)
  - OpenGL
  - increasingly, Java

**traditional "computer graphics" courses**

2

## Graphics in IT82

- We take a practically-oriented look at graphics
- Richardson's *"Practical Computer Graphics"* is one of the few books which takes this approach
- The aim is not to make you a "graphic designer", but instead:
  - To give you an understanding of issues concerning graphics input/output and representation
  - To equip you for practical situations where you might need to use graphics, e.g. MultiMedia
    - web pages of holiday snapshots, icon designs, lecture presentations, suitable graphics file formats to use

3

## Lectures Overview

*This week:*

- Basic principles of representing graphical data
- Practical graphics issues (e.g. fonts)

*Next week / later:*

- Representing colour
- Input/Output: Scanners, Cameras and Printers
- Basic principles of representing animation
- Compression of graphical data; storage in appropriate file formats
- Overview of Java and Graphics

4

## Representing Graphical Data

- Logical and Physical Representation
- Use of colour:
  - Pixels
  - Colours
  - Transparency
  - Palettes
- Types of representation:
  - Bitmaps
  - Vector data
  - Other ways


5

## Logical / Physical Representation

⚠ A warning to bear in mind:

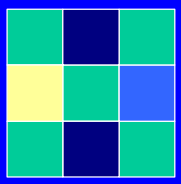
- Physical representation of graphical data is how it *actually* appears on devices
- A virtual/logical representation of graphical data may be in a graphics file, or internally in a program
- These are often not the same!
- The differences vary from slight to very large
- Converting from a virtual representation to an actual display on a device is called *rendering.*

6

# Pixels

- All of “computer graphics” is based on properties of screen or display device
- Displays are divided into lots of small dots called pixels (PICture ELements)
- Pixel is smallest logical unit of display on the screen
- Can be monochrome (black and one colour) or coloured

3 x 3 array of coloured pixels

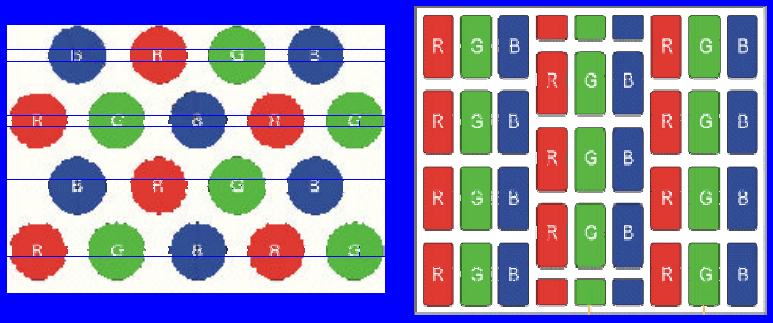


- Arranged (logically!) in a 2D grid

7

# Pixels

- Physical display is different!
- Not necessarily a perfect 2D grid:




8

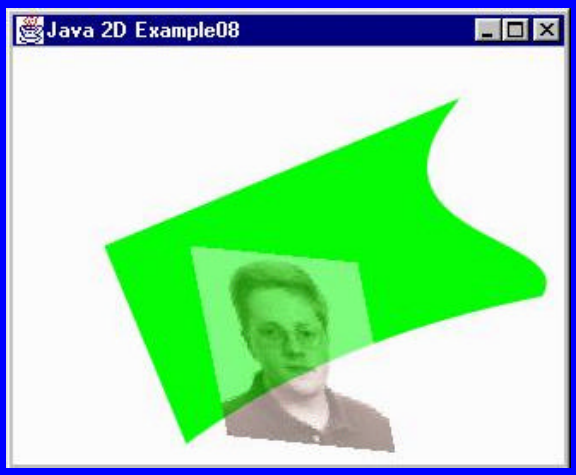
## Colours of Pixels

- Black/white pixels are represented using bits
- Colours are specified (or get converted to) using RGB values in some way (- see Lectures on Colour later)

- A typical format is using 24 bits format
  - (R,G,B) takes up 1 byte for each colour
- A common feature these days is to also have an *alpha channel* , used for transparency. e.g. in Java 2,
  - (R,G,B,α) takes up 4 bytes
  - see *Chapman & Chapman pg 135*

9

## Transparency




10

## Colour Terminology

Lots of confusion:

- "Black and white" not good terminology to use
- Black and white photographs are not just black/white, but really greyscale
- "Greyscale" refers to shades of grey, ie where the RGB values are all the same
- "Monochrome" refers not to one colour, but historically to "one colour with black", so "monochrome" really means two colours, usually black and white
- "Monochromatic" in colour blindness refers to greyscale!

11

## Palettes

- A *palette* is a mapping from a small set of numbers, to specifically chosen colours from a wide range ($2^{24}$ typically)
- "Indexed images" use palettes
- Used in various file formats, monitor displays

Example (*web-safe palette*, reproduced by all web-browsers on any system using 8-bit colour)):

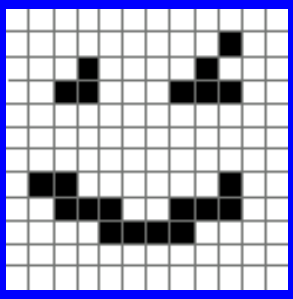
*Chapman & Chapman, pgs 161-165*

12

## One way to represent image data

- Bitmapped formats: image is modelled by an array of pixel values
- Bitmap data is (logically) a 2D array of pixels
- A *bitmap* gives the colours of the picture, pixel-by-pixel (bit-by-bit), in this example:

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



(each pixel is represented by one bit (on or off)

13

## Bitmaps

- Bitmaps were also known (in ye olden days) as a *raster* (the term is still in use in some circumstances, as it defines a set of dots (pixels) arranged in parallel lines)
- When there used to be just monochrome monitors, bitmaps did indeed have bits in them!
- When colours were introduced, the term *pixelmap* was used for coloured images.
- Nowadays, bitmaps can refer to 2D arrays of bits *or* colours.
- Logically, bitmaps are 2D arrays, although in fact they may be stored by other means
  - Java 2 uses a 1D **`int`** array

14

## Graphical Data Representation

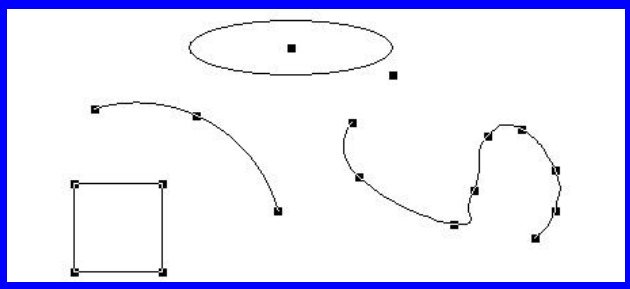
- Bitmaps have a fixed *resolution* (amount of detail in an image)
- There are other ways of representing image data which do not:
  - Some are general purpose
  - Some are program-specific
  - Some are application-specific
- In many state-of-the-art graphics programs, images are represented internally in an application-specific way, then exported to bitmap formats.

*Richardson, Section 1.4*

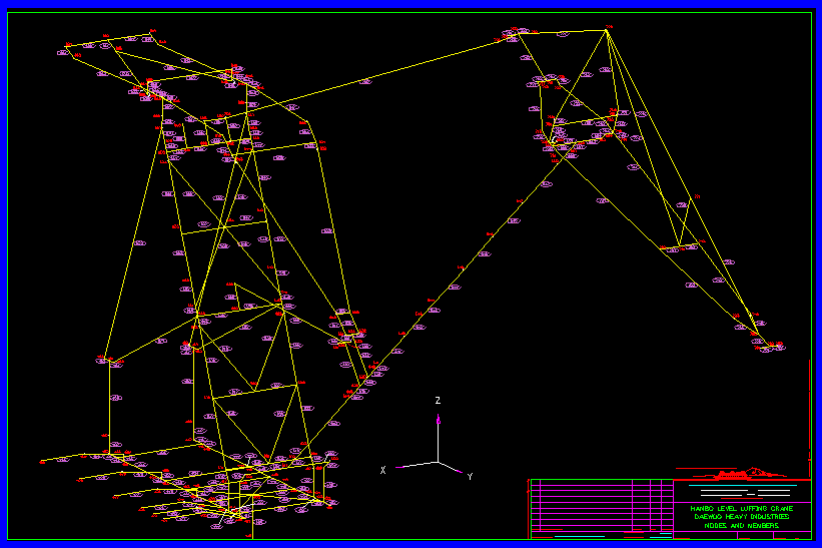
15

## Another way to represent image data

- *Vector-based* formats contain descriptions of one or more objects, rather than pixels
- Uses a “draw-then-edit” method of image creation
- Often the objects are mathematically based
  - eg line segments, polygons, circles, splines



16

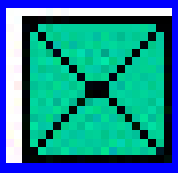


Vector formats are well suited to representing 2-D images such as pencil drawings, graphs and architectural or engineering drawings

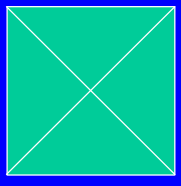
17

## Bitmaps vs Vector Files

Bitmap files are fixed resolution



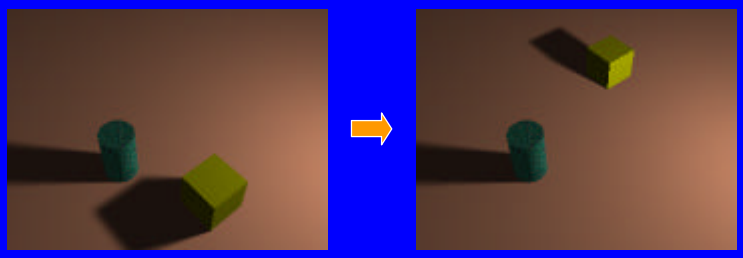
Vector files can be displayed at whatever level of detail is preferred

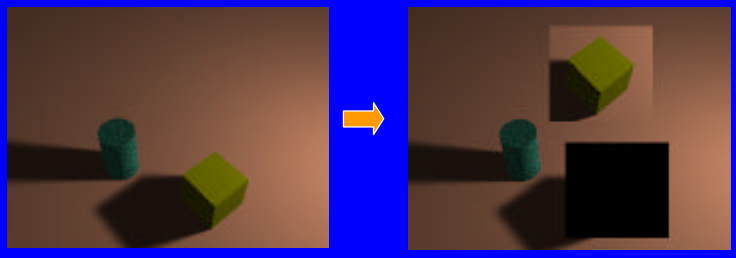


18

## Bitmaps vs Vector Files

Editing a vector file





…and in a bitmap?

19

## Bitmaps vs Vector Files

Further vector advantages:

- Good for storing images composed of line-based or 3D objects (e.g. wire-frame models)
- Easy to convert to bitmap format

Vector file disadvantages:

- Not good for storing complex images (such as photographs)
- Appearance of image can vary widely, depending upon the application
- Rendering of the image may take significantly longer than for bitmaps

20

## From Vectors to Bitmaps...

- Historically, vector data was used a lot.
- Pen plotters used pens to draw on paper (an early form of graphics printer)
- These were cheap and produced line-based drawings.
- Storage of high-volume bitmap files was expensive!

- With the advent of cheap storage, and high-resolution output, now most images are bitmap-based.
- Bitmaps are everywhere!
  - Just look at the WWW, with GIFs, JPEGs everywhere!

21

## …and Back Again

- Trends are shifting towards a greater use of vector data - the bitmap trend may not last!
- Memory size is again an issue
  - Big bitmaps take longer to transport over the internet

- Vector-based formats are better for 3D imaging, and 3D imaging is growing more important (fuelled by such concerns as the entertainment industry)

22

## Other Graphics Representations

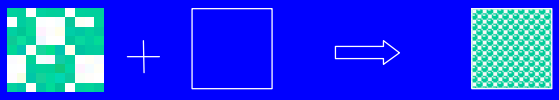
- Hybrid formats
  - e.g. Metafile formats
- Fractal compression techniques

- Animation formats
- Special purpose 3D formats

23

## Metafile Formats

- A metafile can store both vector and bitmap data
- A bitmap is typically regarded as one type of “vector” object
- Typically most elements in the file are vectors, with the occasional bitmap
  - e.g. a bitmap stored as a “fill pattern” for a shape



*Richardson, Section 6.9*

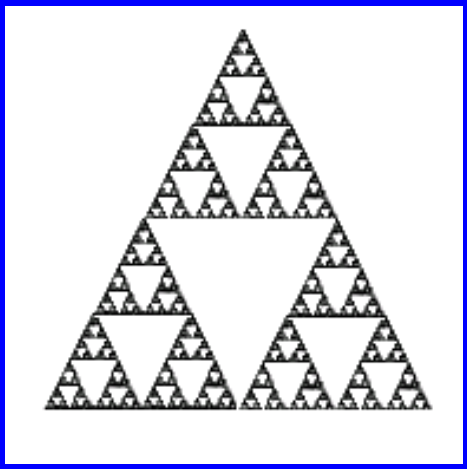
24

## Fractal Image Compression

- A recent development in image representation
- An image is represented by a mathematical formula
- To produce a display of the image on a device, the formula is repeatedly applied to a (maybe) blank "seed" image of the required size
- A resolution-independent way of storing images
- Although the word "compression" is used, really this is just another way of representing an image (encoding/decoding would be better terminology)
- It is compression because the formula takes up less space than a bitmap would.

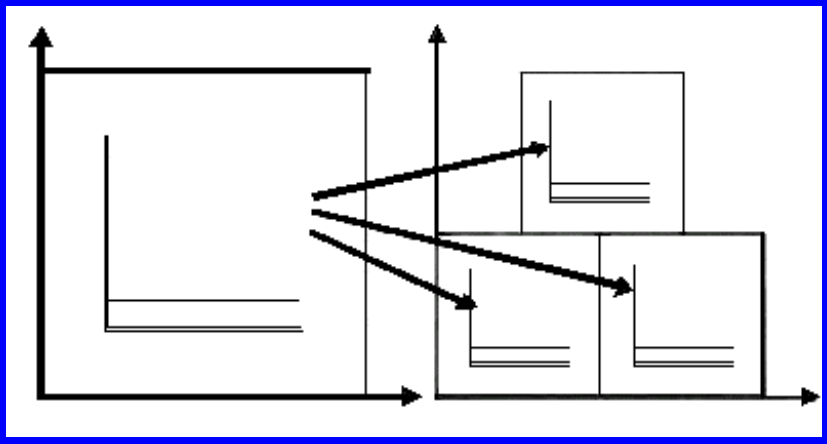
25

## Example: Sierpinski Triangle



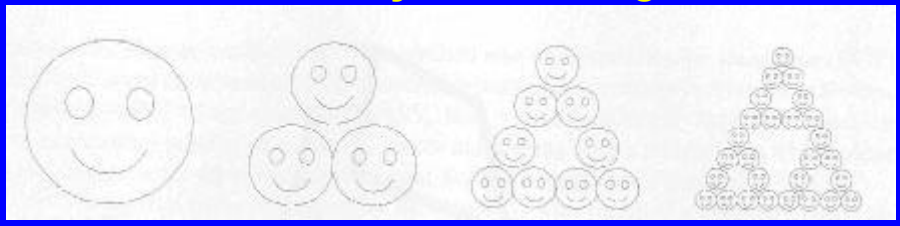
26

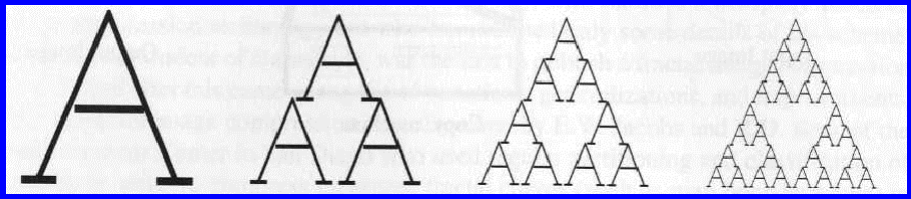
## Transformation



27

## Producing the Sierpinski Triangle from **any** initial image





28

## Use of Fractal Image Compression

- Typically an image would be originally in bitmap form, before this technique is applied.
- The resulting image then has an optimal resolution close to that of the original image.

- Magnification still looks nicer (softer, not pixellated) with a fractally-compressed image.
- Sometimes representing an image in this way can be used for image enhancement.

29

## Magnification Example

Original bitmap x2 Fractal version x2

30

## Fractal Compression

- Pictures that are very *self-similar* compress well using this method
- Examples:
  - Sierpinski triangle **very** self-similar
  - Photograph with leaves in foreground and background would have a lot of self-similarity
- Most pictures (photographs or diagrams) that humans use have a lot of self-similarity
- Finding an example not suitable for fractal representation is difficult.
- Fractal compression/encoding can take time but decompression/decoding process is very fast, hence also used for archived images in a CD-Rom encyclopaedia

*Chapman & Chapman, pgs 109-110*

31

## End of Lecture

- Next lecture is on “Practical Graphics Issues”
  - particularly to do with fonts

32