CSE 1402 Technical Documentation for Software Engineers

Carlo Kopp, BE(Hons), MSc, PEng,
Monash University, Clayton, Australia
email: carlo@csse.monash.edu.au
Computer Science & Software Engineering
http://www.csse.monash.edu.au/

October 12, 2000

Lecture Notes (7-13)

1 Revision Information

This document is currently at revision level:
$Id: CSE-1402-T-B.tex,v 1.1 2000/08/28 17:29:58 carlo Exp $

2

LaTeX Layout Markup Commands

3 Layout Markup Commands

- Layout markup commands are used to manipulated details of a document layout, rather than changes to global layout.
- Layout markup commands should be used carefully, because they bypass the global setup of the document layout and style.
- When using layout markup commands, it is important to always be consistent in their use.
- An example of consistent use is where every caption to a picture uses the same font type, font weight, font size and is set to italic or bold.
- An example of inconsistent use is where captions to pictures vary widely in the font type, font weight and font size, with some in italic or bold.

4 Manipulating Font Size

- Sometimes we need to use a larger or smaller font to attract the attention of the reader.
- One technique is to change the size of the font. This is very commonly used for more complex header or title pages. Font sizes are measured in 'points' or 'pt', there are 72.27 pt/inch or 2.85 pt/mm.
- LaTeX uses a very simple method - a 'size change' command, which is applied to the following text which is enclosed in curly braces.

- The basic syntax is {\size_change_cmd text_to_be_changed }.
- LaTeX provides ten basic font size commands, which are scaled to match the standard font size in the document, e.g. 12 pt.

5 Font Size Commands

```
{\tiny Font Size}
{\scriptsize Font Size}
{\footnotesize Font Size}
{\small Font Size}
{\normalsize Font Size}
{\large Font Size}
{\Large Font Size}
{\LARGE Font Size}
{\huge Font Size}
{\Huge Font Size}
{\HUGE Font Size}
```

6 Font Size Commands ...

Font Size

Font Size

Font Size

Font Size

Font Size

Font Size

Font Size

Font Size

Font Size

Font Size

7 Manipulating Font Weight

- Most font families provide only one or two weights, typically 'light', 'medium' and 'bold'.
- LaTeX uses 'Computer Modern Roman' as its standard font, this is a 'serifed' font with only 'medium' and 'bold' weights. Some font families will provide 'semi light', 'light', 'extra light' weights.
- LaTeX provides a simple shorthand to change the font to 'bold'.
- The basic syntax is **{\bf text_to_be_changed }**.

Example:

```
This chunk of text is part in medium weight
and {\bf part in bold.}
```

This chunk of text is part in medium weight and **part in bold.**

8 Emphasis with Italics

- *Italics* are used to *emphasise* words or whole paragraphs in a document. This is used to focus the reader's attention to the word or paragraph.
- LaTeX uses the \emph{} environment for emphasis.
- LaTeX provides a simple shorthand to set emphasis.
- The basic syntax is **{\em text_to_be_changed }**.

Example:

```
This chunk of text is part in medium weight
and {\em part in italic.}
```

This chunk of text is part in medium weight and *part in italic.*

9 Changing Font Types

- Most documents will use a single font type throughout. Sometimes it is useful to change the font type to draw attention. An example is when we cite command lines or program source in document.

Example:

```
{\rmfamily This text is in Roman font.}
{\ttfamily This text is in Typewriter font.}
```

This text is in Roman font. `This text is in Typewriter font.`

10 Verbatim and {\tt}

- LaTeX provides two convenient environments for citing source code or commands.
- The 'teletype' or {\tt } environment is shorthand for the `\ttfamily` command.
- The `verbatim` environment ignores all LaTeX commands between `\begin{verbatim}` and `\end{verbatim}`.

Example:
`\begin{verbatim}`.

```
This text is cited verbatim, and includes
{\tt Typewriter Font}.
```

`\end{verbatim}`
This text is cited verbatim, and includes `Typewriter Font`.

11 Footnotes

- Footnotes are frequently used in documents to explain details which might clutter the text, clarify points, or list references.
- LaTeX uses the \footnote{} environment for footnoting. Footnotes are numbered automatically, and should be put at the end of a sentence or paragraph.
- The basic syntax is **\footnote{ footnote_text }**.

Example:

```
This sentence has a footnote\footnote{ Which contains
this trivial comment.}.
```

This sentence has a footnote[1].

[1] Which contains this trivial comment.

12 Quotes

- Quotes are widely used in printed text. In documentation they are most commonly used for titles of papers, texts and other documents.
- In printed documents and books, it is customary to always use paired quotes, whether single or double.
- LaTeX uses a very simple arrangement for single and paired quotes.

Example:

```
This is a ‘single paired quote’.
This is a ‘‘double paired quote’’.
```

This is a ‘single paired quote’.
This is a “double paired quote”.

13 \noindent and \newpage

- Default LaTeX styles typically indent the first line in a paragraph.
- Sometimes this is not convenient, especially if the paragraph comprises less than a line, and is followed by a picture or table.
- The `\noindent{ }` environment stops indenting for the beginning of the first paragraph which is enclosed.
- Frequently we might wish to have a section start on the beginning of a new page.
- LaTeX provides a command to force the following text to start at the beginning of a new page.
- This is done using the `\newpage` command.

14 Example: \noindent

```
\noindent{Example:}\\
\noindent{This is a paragraph without indenting.

This is a paragraph with indenting.}
```

Example:
This is a paragraph without indenting.

This is a paragraph with indenting.

15 Lists

- LaTeX provides some very powerful environments for producing lists.
- A 'bullet' list can be produced using the `itemize` environment.
- A 'numbered' list can be produced using the `enumerate` environment.
- An 'description' list can be produced using the `description` environment.
- Each entry in the list must be preceded by a `\item` command.
- All items between the `\begin` and `\end` delimiters are formatted into the list.

16 Example Bullet List

```
\begin{itemize}
\item Bullet list item one.
\item Bullet list item two.
\item Bullet list item three.
\end{itemize}
```

- Bullet list item one.
- Bullet list item two.
- Bullet list item three.

17 Example Numbered List

```
\begin{enumerate}
\item Numbered list item one.
\item Numbered list item two.
\item Numbered list item three.
\end{enumerate}
```

1. Numbered list item one.
2. Numbered list item two.
3. Numbered list item three.

18 Example Description List #1

```
\begin{description}
\item[1st Entry:] Entry list item one.
\item[2nd Entry:] Entry list item two.
\item[3rd Entry:] Entry list item three.
\end{description}
```

1st Entry: Entry list item one.

2nd Entry: Entry list item two.

3rd Entry: Entry list item three.

19 Example Description List #2

```
\begin{description}
\item[Open:] This command is used to  ...
\item[Close:] This command is used to  ...
\end{description}
```

Open: This command is used to open a file. When it is used, the program will open a window, into which the user types the name of the file. The file then opened.

Close: This command is used to close a file. When it is used, the program will close the currently opened file.

20 Customising Lists

- Sometimes it is convenient, if not necessary, to customise a list environment.
- LaTeX provides very powerful facilities to do this. These are described in any good text.

21 Tables

- LaTeX provides some very powerful environments for producing tables.
- Table entries can be aligned left, right or centred.
- Tables can be positioned at the top or bottom of the page.
- Horizontal separator lines can be incorporated easily using `\hline`.
- The basic syntax is `\begin{tabular}[`*pos*`]{`*cols*`}`, where *pos* is the position specifier and *cols* is the column format specifier, followed by `\end{tabular}`.

- The position specifiers are `t` for the top of the page and `b` for the bottom of the page.
- The column specifiers are `l, r, c` for left, right and centre, respectively, entries are separated by `&`.

22 Table Example #1

```
\begin{tabular}{|l|l|l|l|} \hline
Name   &Title  &Signature  &Date\\ \hline
Jane Doe   &Programmer &   & Today's Date\\ \hline \\
My Tutor   &Project Leader &   & Today's Date\\ \hline  \\
Carlo Kopp &Project Director   &   &28/08/2000\\ \hline \\
\end{tabular}
```

Name	Title	Signature	Date
Jane Doe	Programmer		Today's Date
My Tutor	Project Leader		Today's Date
Carlo Kopp	Project Director		28/08/2000

23 Table Example #2

```
\begin{tabular}{|l|r|l|c|} \hline
Name   &Title  &Signature  &Date\\ \hline
Jane Doe   &Programmer &   & Today's Date\\ \hline
My Tutor   &Project Leader &   & Some Date\\ \hline
Carlo Kopp &Project Director   &   &28/08/2000\\ \hline
\end{tabular}
```

Name	Title	Signature	Date
Jane Doe	Programmer		Today's Date
My Tutor	Project Leader		Some Date
Carlo Kopp	Project Director		28/08/2000

24 Table Example #3

```
\begin{tabular}{llll} \hline
Name   &Title  &Signature  &Date\\
Jane Doe   &Programmer &-   & Today's Date\\
My Tutor   &Project Leader &-   & Some Date\\
Carlo Kopp &Project Director   &-   &28/08/2000\\
\hline
\end{tabular}
```

Name	Title	Signature	Date
Jane Doe	Programmer	-	Today's Date
My Tutor	Project Leader	-	Some Date
Carlo Kopp	Project Director	-	28/08/2000

25 Table Example #4

```
\begin{tabular}{|l|l|l|l|} \hline
Name   &Title  &Signature  &Date\\ \hline \hline
Jane Doe   &Programmer &-   & Today's Date\\ \hline
My Tutor   &Project Leader &-   & Some Date\\ \hline
Carlo Kopp &Project Director   &-   &28/08/2000\\
\hline
\end{tabular}
```

Name	Title	Signature	Date
Jane Doe	Programmer	-	Today's Date
My Tutor	Project Leader	-	Some Date
Carlo Kopp	Project Director	-	28/08/2000

26 Floating Tables

- LaTeX provides facilities for floating tables, which include headers, footers, captions and labels.
- Floating tables can be positioned at the top or bottom of the page.
- The basic syntax is \begin{table}[*pos*]{*header table footer/caption label*}, followed by \end{table}.
- The header appears at the top of the floating table.
- The footer appears at the bottom of the floating table.
- The caption appears at the specified position.

27 Floating Table Example #1

```
\begin{table}[h] {\bf Signoff Table Header}\\
\begin{tabular}{|l|l|l|l|} \hline
Name   &Title  &Signature  &Date\\ \hline \hline
Jane Doe   &Programmer &-   & Today's Date\\ \hline
My Tutor   &Project Leader &-   & Some Date\\ \hline
\end{tabular}\\ {\bf My Project Footer}
\caption{Signoff Table Example} \label{MyTab}
\end{table}
```

```
\noindent{The example table is found at
Table \ref{MyTab}.
```

28 Floating Table Example #1 ...

Signoff Table Header

Name	Title	Signature	Date
Jane Doe	Programmer	-	Today's Date
My Tutor	Project Leader	-	Some Date

My Project Footer

Table 1: Signoff Table Example

The example table is found at Table 1.

29 Labels and References

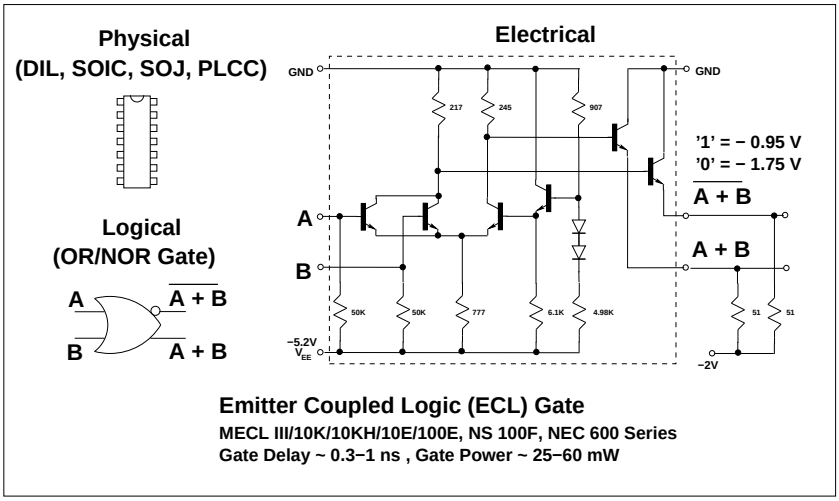
- LaTeX provides a powerful facility for automatic referencing inside documents.
- Automatic referencing is extremely useful since a user need not worry about the specific numbers attached to tables, pictures or sections. LaTeX generates the numbers automatically.
- The label syntax is `\label{mylabel}`.
- The reference syntax is `\ref{mylabel}`.
- Typically the labels and references require two consecutive compiles before they are properly resolved.
- The preceding example of a floating table includes the use of a label and reference.

30 Incorporating Pictures

- LaTeX provides powerful and flexible facilities for incorporating pictures into documents.
- If we are using LaTeX directly, and converting DVI files into PostScript, then we can easily incorporate pictures in EPS format.
- If we are using *PDF*LaTeX, then we can incorporate pictures in JPEG, PNG and PDF formats. N.B. PDF format pictures can be produced from EPS using the `epstopdf myfile.eps` command, which adjusts the bounding box parameters for *PDF*LaTeX.
- Picture environments are usually defined as floating.
- Care should always be taken with pictures since problems with choice of picture format are very quickly noticed by readers.

31 Picture Example #1:

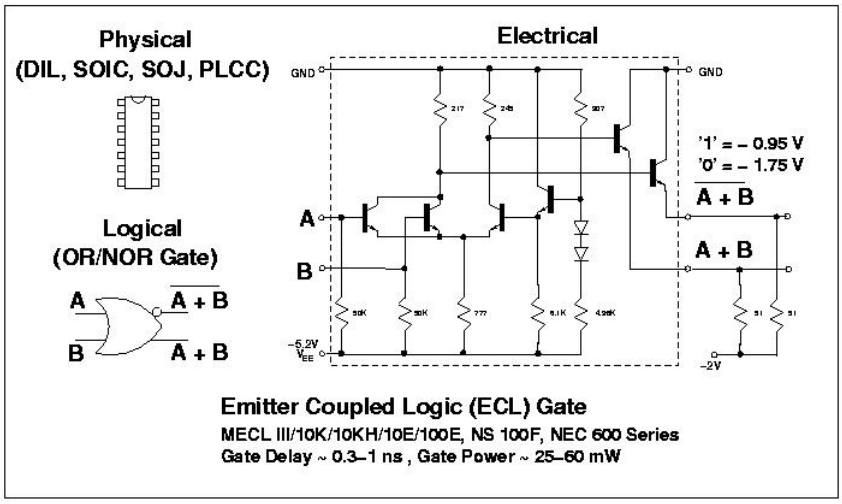
```
\usepackage[pdftex]{graphicx} % in the file header
\includegraphics[scale=0.5]{PDF/mecl-or.pdf}
```



A picture rendered in PDF format.

32 Picture Example #2:

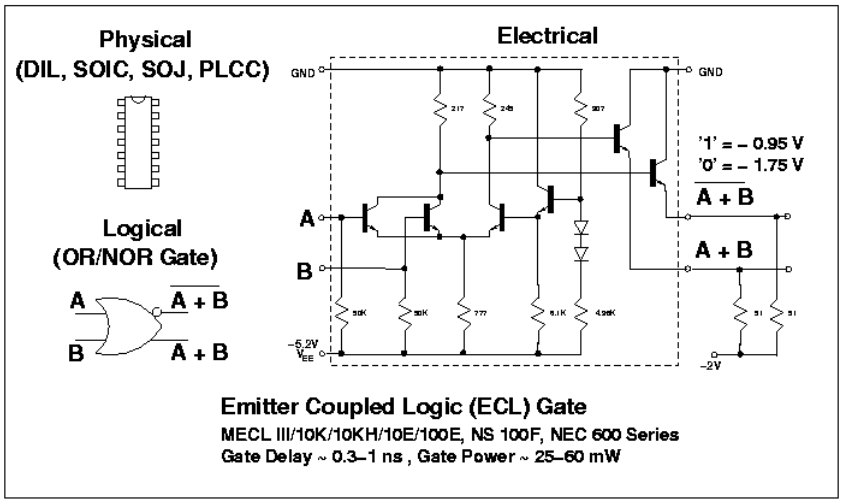
```
\usepackage[pdftex]{graphicx} % in the file header
\includegraphics[scale=0.5]{PDF/mecl-or.jpg}
```



A picture rendered in JPEG format.

33 Picture Example #3:

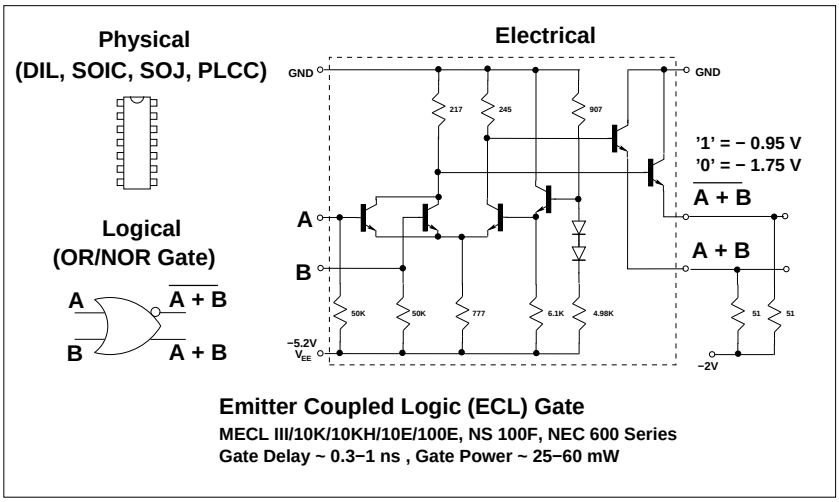
```
\usepackage[pdftex]{graphicx} % in the file header
\includegraphics[scale=0.5]{PDF/mecl-or.png}
```



A picture rendered in PNG format.

34 Picture Example #4:

```
\usepackage {epsfig} % in the file header
\epsfig{file=PDF/mecl-or.eps}
```



A picture rendered in EPS format (fudged with PDF).

35 BibTeX

- The BibTeX tool provides a very powerful and flexible facility for managing references in LaTeX documents.
- BibTeX searches through a `myfile.tex` LaTeX source file and locates citations identified by the `\cite{mykey}` tag. It then searches the bibliographic database file `bib.bib` a produces a bibliography file called `myfile.bbl`.
- When LaTeX is run it uses the bibliography file `myfile.bbl` to produce the bibliography list in the document.
- The user needs to edit the `bib.bib` database to add references, and insert appropriate `\cite{mykey}` tags in the document. LaTeX and BibTeX perform the hard work.

36 The BibTeX Database

- The BibTeX database contains entries which have specific formats for references such as books, journal papers, technical reports and other types of documents.
- The entry type field determines which other fields the entry should contain.
- BibTeX entries in the default format will have their capitalisation changed to fit a default, which is frequently incorrect for many reference types.
- Therefore it is necessary to use a syntactic ploy, and encapsulate each field in the database entry with {{ `double curly braces`}} to force BibTeX to preserve the original capitalisation.
- It is customary for the key field to be in the format `surname:year`.

37 The BibTeX Database Entry

```
@TechReport{   W3C:99-2,
  author = {{Dave Raggett, et al}},
  title = {{HTML 4.01 Specification}},
  year = {{W3C Recommendation, 24 December 1999}},
  institution = {{W3C HTML Working Group.}},
  type = {{Specification,
  http://www.w3.org/TR/1999/REC-html401-19991224}}
}
```

38 The BibTeX Bibliography

```
\begin{thebibliography}{1}

\bibitem{W3C:99-2}
```

```
{Dave Raggett, et al}.
\newblock {HTML 4.01 Specification}.
\newblock {Specification,
  http://www.w3.org/TR/1999/REC-html401-19991224},
  {W3C HTML Working Group.},
  {W3C Recommendation, 24 December 1999}.

\end{thebibliography}
```

39 Example BibTeX Usage

```
... candidates are listed in the references under
\cite{Pepper:2000}, \cite{Mozilla:2000},
\cite{AGOCG:2000}.

\bibliographystyle{plain}
\bibliography{bib}
```

... candidates are listed in the references under [2], [3], [1].

References

[1] Advisory Group on Computer Graphics. The Computer Graphics Metafile (CGM). Web Reference, http://www.agocg.ac.uk/CGM.html, http://www.agocg.ac.uk/train/cgm/ralcgm.htm, Advisory Group on Computer Graphics, 2000.

[2] Steve Pepper. SGML & XML TOOLS - By Vendor Name. Web Reference, http://www.infotek.no/sgmltool/vendors.htm, STEP Infotek, Oslo, Norway, 29 May 2000.

[3] The Mozilla Organization. Scalable Vector Graphics (SVG). Web Reference, http://www.mozilla.org/projects/svg/, The Mozilla Organization, 13th June 2000.

40 Running BibTeX

- To use BibTeX, we must first run LaTeX to produce the auxiliary file `myfile.aux` from `myfile.tex`.
- We then run BibTeX using `bibtex myfile` to produce the bibliography `myfile.bbl`.
- Finally we must run LaTeX again to resolve the references properly.

```
raptor[carlo]1022% pdflatex CSE-1402-T-B.tex
raptor[carlo]1024% bibtex CSE-1402-T-B
This is BibTeX, Version 0.99c (Web2C 7.3.2x)
The top-level auxiliary file: CSE-1402-T-B.aux
```

```
The style file: plain.bst
Database file #1: bib.bib
raptor[carlo]1022% pdflatex CSE-1402-T-B.tex
```

41

Graphics

42 References:

TIFF, Revision 6.0, Final - June 3, 1992, Adobe Systems Inc.

PNG (Portable Network Graphics) Specification, Version 1.0, W3C Recommendation 01-October-1996.

David Cruikshank, et al, **WebCGM Profile**, W3C Recommendation, 21 January 1999.

Jon Ferraiolo et al, Editor, **Scalable Vector Graphics (SVG) 1.0 Specification**, W3C Candidate Recommendation, 2 August 2000.

43 Graphics

- A wide range of file formats are available for including graphics in documents, be they hardcopy or on-line.
- The quality of artwork can have a strong impact on reader perceptions of the quality of a document, therefore it pays to choose file formats wisely.
- Graphics can be in *bitmap* or *vector* formats. Each have particular applications they are best suited to.
- A major issue in considering the flexibility and quality of DTP and typesetting tools is their support for particular graphics file formats. A limited ability to use vector format graphics is a good indicator of a limited tool or package.

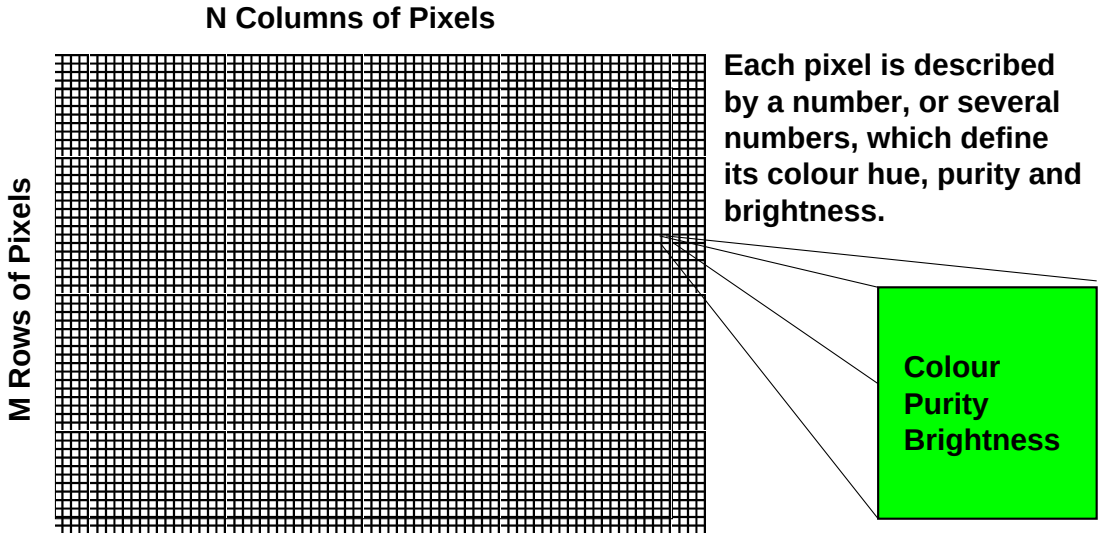
44 Bitmap Formats

- Bitmap formats represent a picture by dividing it into a regular array of rectangles or squares, each of which is uniquely described by colour, saturation and brightness information. Viewed together,

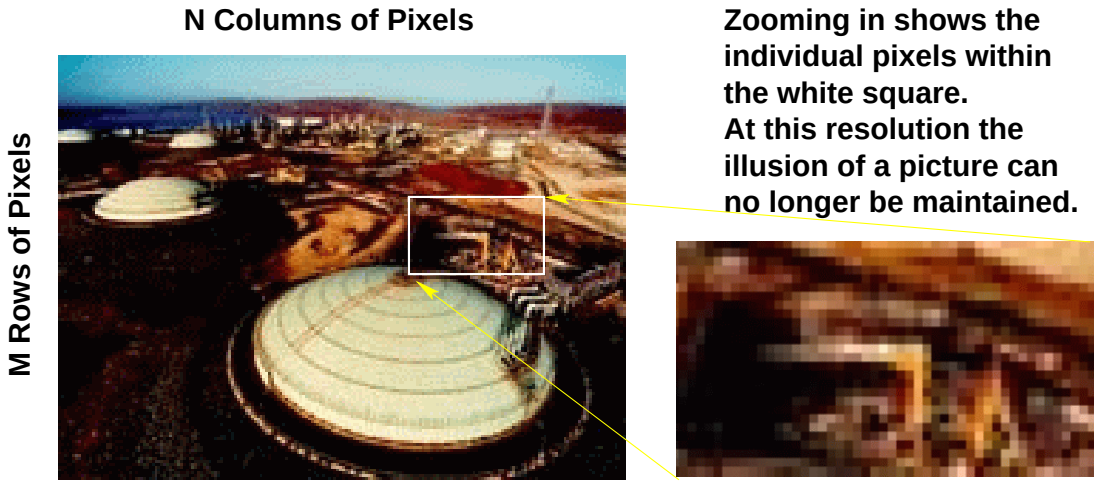
the elements in the array form a picture.

- Each rectangle or square is termed a ‘picture element’ or ‘pixel’.
- Colour, saturation and brightness are usually represented by three or four values, which additively produce the intended result.
- The two most common schemes used are Red-Green-Blue (RGB) or Cyan-Magenta-Yellow-Black (CMYK).
- RGB is used mostly for on-screen rendering, CMYK mostly for printing. PostScript and PDF files can support both schemes.

45 Bitmap Formats ...



46 Bitmap Formats ...



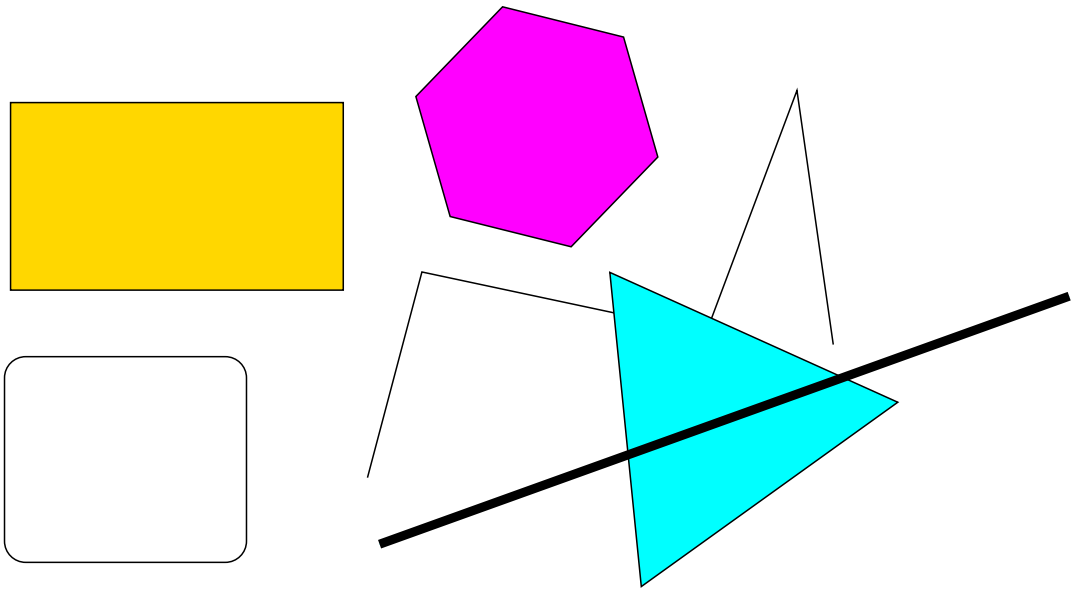
47 Bitmap Image Quality

- The conversion of a real image into a digital representation as an array of numbers describing the properties of pixels involves necessary compromises in quality.
- The image must be 'quantized' in the X and Y axes to break it up into pixels, and each pixel must be 'quantized' in colour, saturation and brightness.
- Quantization amounts to an approximation, and determines the quality of the bitmap image.
- An image which is spatially quantized into 3000 x 2000 pixels has a higher quality than one quantized into 30 x 20 pixels.
- An image in which each pixel uses 24 bits to describe the pixel has a higher quality than one using 16, 8, 4 or 1 bit per pixel.

48 Vector Formats

- Vector formats represent a picture as a collection of 'graphics primitives' or basic shapes, each of which is described by a set of numbers.
- For instance, a line can be described by its two end points, (X_1, Y_1) and (X_2, Y_2), a circle by its centre (X_C, Y_C) and a radius value.
- A pure vector format is limited in that only objects which can be properly described in such a form can be represented.
- Many vector formats include provisions for embedding bitmap images into the vector format image.
- Most vector formats include means of filling areas or shapes with specific colours, some even allow colour gradients.

49 Vector Formats ...



50 Vector Graphics Quality

- A vector representation of a picture is exact, therefore the quality of rendering depends completely on the quality of the rendering device, such as a display or a printer. Therefore, vector graphics alway produce the highest possible quality when displayed or printed.
- Another advantage of vector representation is that it can be very compact, in comparison with a bitmap rendering of the same picture. This is because large areas can be completely described with very few numbers.
- Where the choice exists, vector graphics should always be used in preference to bitmap graphics.

51 Vector Graphics Quality ...

- Most drawing packages can produce both vector and bitmap output files, however, DTP and typesetting packages are frequently limited in what vector formats they can import.
- A nice attribute of well designed vector formats is that pictures in one format can be converted to another with no loss of information or quality.
- For instance, a picture in EPS format can be converted to PDF and vice versa, a picture in SVG into EPS and so on ...

52 Tagged Image File Format 6.0

- TIFF was devised by Aldus Corp in 1986 as a bitmap standard for supporting scanners and photo-retouching tools.
- The current version of TIFF is 6.0, the specification is maintained now by Adobe.
- TIFF is widely used in the graphics industry as a portable format for bitmap imagery.
- While early TIFF compatible tools lacked lossless compression facilities, current implementations usually support them.
- TIFF supports full 24-bit colour, Version 6.0 can represent colour in RGB, CMYK, YCbCr, and CIE formats.
- TIFF files with no compression are frequently very large and cumbersome to handle.

53 Tagged Image File Format 6.0 ...

- TIFF is usually supported by retouching and drawing tools, less frequently by DTP and TS tools, and browsers.
- Therefore it should be converted to another format for use with the latter, either JPEG or PNG.
- TIFF files usually have a `.tif` or `.tiff` suffix.

54 GIF87a, GIF89a

- The GIF format is now considered largely obsolete, but remains very widely used for web publishing.
- GIF is an 'encumbered' defacto-standard, since it incorporates a compression mechanism owned by Unisys. Therefore royalties are supposed to be paid on tools which can process GIF files.
- GIF has the nice attribute of lossless compression, as a result of which GIF files are always compact, and it represents line drawings without artifacts.
- GIF has an important limitation in poor colour handling. This is a consequence of its origins being in a period when most displays used only 8-bit colour representation.
- When producing new web documents, PNG should be used instead.

55 Portable Network Graphics

- PNG is an unencumbered bitmap graphics standard intended to replace GIF, and in many applications also TIFF.
- PNG employs lossless compression, transparency, gamma correction, cyclic redundancy check (CRC) error detection and progressive display.

- Colour handling in PNG is powerful, allowing 48-bit per pixel true colour, 16-bit per pixel grayscale and gamma correction to compensate for display non-linearity.
- *PNG is the best current standard for web publishing, since it provides similar colour handling to TIFF, but employs lossless compression which allows it to render line drawings sharply and without artifacts.*

56 Joint Photographic Experts Group

- JPEG is a bitmap representation standard which employs a lossy compression scheme. It is very widely used on the web.
- JPEG is commonly used by cheaper digital cameras for disk storage.
- The principal limitation of JPEG is its lossy compression mechanism, which can result in a highly variable quality of rendering and the appearance of decoding artifacts such as speckle.
- Lossy compression will 'throw away' some detail in the picture to achieve a higher compression ratio. Most tools allow some control of image quality by manipulating the compression parameters.
- JPEG can provide very good representation of imagery such as scanned pictures, but frequently performs poorly on line drawings.

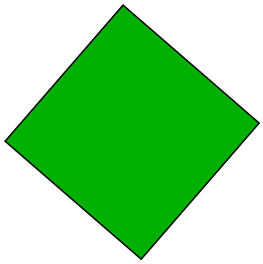
57 Joint Photographic Experts Group ...

- JPEG files can usually be identified by a `.jpg` or `.jpeg` suffix.
- *Since you cannot control the quality of the tool being used to view your JPEG image, you can never guarantee the quality of the rendering seen by a user. JPEG should not be used for line drawings.*

58 Computer Graphics Metafile

- The ISO Computer Graphics Metafile standard (ISO/IEC 8632:1992) is a vector graphics representation scheme, which is widely used in engineering tools.
- CGM is important for two reasons. It is the basis of the new W3C WebCGM standard, and it is the only vector graphics scheme at this time which is compatible with Microsoft tools and widely used Unix tools.
- A CGM file can include embedded and compressed bitmaps using CCITT group 4, JPEG, and LZ77 derivative compression.
- WebCGM employs RGB colour representation and supports transparent backgrounds.

59 Computer Graphics Metafile Example



```
BEGMF 'xfig-fig001442';
mfversion 1;
mfdesc 'Converted from /tmp/xfig-fig001442
using fig2dev -Lcgm';
mfelemlist 'DRAWINGPLUS';
vdctype integer;
fontlist 'Hardware',
```

60

```
  'Times New Roman';
BEGMFDEFAULTS;
  vdcext (0,0) (1284,1284);
  clip off;
  colrmode indexed;
  colrtable 1
    0 0 0
    255 214 0;
  linewidthmode abs;
  edgewidthmode abs;
  backcolr 255 255 255;
  transparency ON;
ENDMFDEFAULTS;
```

61

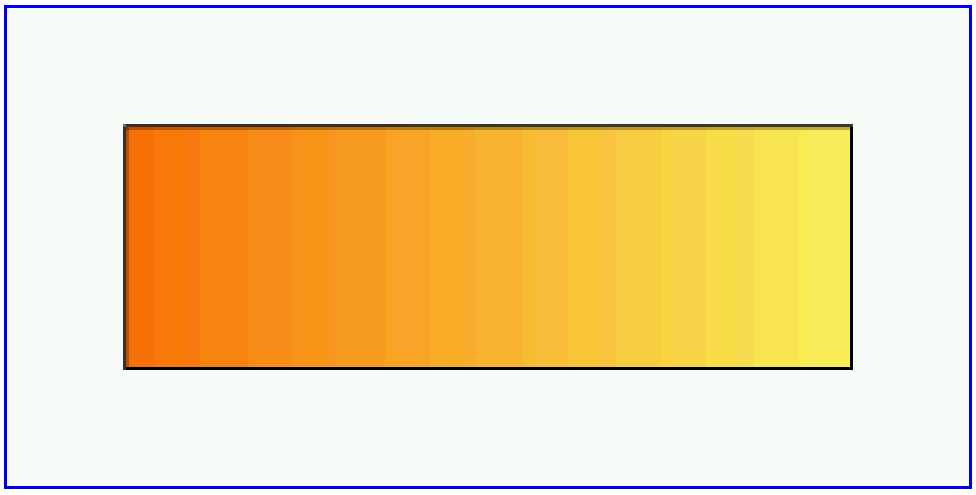
```
BEGPIC 'xfig-fig001442';
BEGPICBODY;
% Polygon %
intstyle SOLID;
fillcolr 14;
```

```
edgevis ON;
edgetype 1;
edgewidth 15;
edgecolr 1;
polygon (1272,687) (597,1272) (12,597) (687,12) (1272,687);
% End of Picture %
ENDPIC;
ENDMF;
```

62 Scalable Vector Graphics (SVG)

- SVG is a vector graphics representation scheme based upon the XML standard, and is intended for use in web publishing.
- SVG may become the most widely used of all vector graphics representation schemes, once it matures.
- SVG supports vector graphics images, embedded bitmap images and text, and also includes two mechanisms for animation.
- Since SVG is based upon XML, it incorporates XML features such as style sheets and the Document Object Model.
- Once SVG and WebCGM become widely adopted, browsers will become capable of rendering line drawings with similar quality to that achieved by PostScript viewers such as `gv` and PDF viewers such as *Acroread*.

63 SVG Example



W3C Example “lingrad01.svg”.

64 SVG Example ...

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
  "http://www.w3.org/TR/2000/CR-SVG-20000802/DTD/
   svg-20000802.dtd">
<svg width="8cm" height="4cm">
  <desc>Example lingrad01 - linear gradient</desc>
  <g>
    <defs>
      <linearGradient id="MyGradient">
      </linearGradient>
    </defs>
```

65 SVG Example ...

```
    <!-- Outline the drawing area in blue -->
    <rect style="fill:none; stroke:blue"
          x=".01cm" y=".01cm" width="7.98cm"
          height="3.98cm"/>
    <!-- The rectangle is filled using a linear gradient
     paint server -->
    <rect style="fill:url(#MyGradient); stroke:black"
          x="1cm" y="1cm" width="6cm" height="2cm"/>
  </g>
</svg>
```

66

Hypertext

67 References:

Goosens M. Rahtz S., *The Latex Web Companion*, Addison-Wesley, 1999.

Charles F. Goldfarb's SGML SOURCE HOME PAGE:
`http://www.sgmlsource.com/`

68 What is Hypertext?

- Hypertext is a mechanism which enables a reader to access another portion of a local or remote document by invoking a **link**.
- One important aim of hypertext is to hide the details of accessing the document or portion of a document which a link points to.
- Hypertext is therefore in concept an automated implementation of the process of finding a reference in a footnote or bibliography, and then using that reference to find another document.
- A hypertext system can be designed to support only links within a document, or only links on a single host system, or links across a large number of systems. The W3 is an example of the latter.

69 Hypertext Implementations

- Two very common hypertext schemes are the HTML/XML model used on the W3, and the simple GNU `info` scheme for online documentation.
- The GNU `info` scheme uses source files which are written in TEX layout markup language, with additional commands. These source files can be compiled into DVI format for conversion to hard-copy, or converted into online text documentation. A good overview can be found by typing `info Texinfo` on a Unix host with GNU `info` installed.
- The most widely used hypertext scheme is the W3C HTML/XML model, which allows links inside documents, links to documents on the same system, and links to documents anywhere on the W3.

70 HTML Limitations

- **Invalid HTML Implementations.** HTML syntax is simple, and is thus frequently abused. Some vendors create proprietary HTML which misbehaves on other browsers. No mechanism exists for 'validating' the syntactic correctness of HTML generators.
- **Handling of Broken Links.** When a web page is moved to another web server, all links to it are broken. No mechanism exists for handling this automatically.
- **HTML Syntax is Fixed.** The syntax of HTML is described by an SGML Document Type Definition (DTD) and cannot be changed. Therefore support for new features requires a new version of HTML.

71 HTML Limitations ...

- **Poor Metadata Facilities.** Metadata tags are important for automated web searches, the HTML <meta> tag is very limited and frequently not used.
- **Layout Markup.** HTML is primarily used for layout markup and does not have the powerful generalised markup facilities of languages such as LATEX. This makes it difficult to structure documents, and requires very disciplined use of layout markup commands by HTML programmers.

- **Lack of Object Oriented Features.** The OO model which is very popular in programming (e.g. C++ and Java) is not supported in basic HTML.

72 HTML Limitations ...

- **Poor Support for Multilingual Text.** Languages which use character sets other than the Latin type, e.g. Cyrillic, Kanji/Chinese, Katakana, Bengali, Hindi, Malayalam, are difficult to represent.

Goosens & Rahtz, Section 3.4, contains an excellent overview of LaTeX schemes for representing non-latin character sets.

The practical consequence of the limitations which exist in the early HTML versions (1-3) is that every vendor will add their own proprietary features to add functionality. In turn this results in serious compatibility problems between authoring tools and browsers.

73 Why Not SGML For Web Pages?

- SGML is a meta-language, which like BNF syntax allows the definition of other languages. Therefore it was considered around 1995 as a direct replacement for HTML.
- Replacing HTML with SGML would provide the full functionality of SGML markup in webpages.
- The penalty for using SGML is that browsers would become significantly more complex, since the more complicated syntax of SGML would result in more complicated parsers.
- Most large software vendors objected to the use of SGML and insisted upon an alternative.
- The W3C standards body therefore defined the **eXtensible Markup Language (XML)** as a subset of SGML.

74 XML Goals (W3C REC-xml)

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.

75 XML Goals (W3C REC-xml)

10. Terseness in XML markup is of minimal importance.

The XML specification reflects these 'ten commandments' very closely. XML is thus designed mainly for machine generated markup, rather than markup written by humans. This fits very closely with the industry trend to generate web pages using publishing tools rather than text editors. XML output filters are now becoming available for a range of tools, e.g. FrameMaker*. Note the following observations:*

1. XML is genuine industry standard and is not proprietary.
2. XML is **not** an variant of HTML, it is a new language.
3. XML web pages will require separate style sheets to define appearance.
4. XML is not limited to uses in presenting or exchanging data.

76 XML Features

- XML is a heavily restricted subset of SGML, the XML standard is about 26 pages long, the SGML standard around 600 pages long.
- An XML file can be viewed by an SGML tool by adding an SGML declaration at the beginning of the file.
- An XML document contains a number of 'entities' (objects), each with some logical 'elements', each element can have a range of 'attributes'.
- The preferred technique for defining attributes is through a 'Document Type Definition' or DTD (a DTD is analogous to a `documentclass` or package in LaTeX).
- Unlike HTML which has a fixed set of tags, XML allows user defined tags.

77 XML Syntax

- XML tag and entity reference syntax is fixed. This is to simply the design of parsers.
- Elements and attributes are declared between matched pairs of angle brackets, i.e. < ... >, with attribute values always placed between single or double quotes i.e. <`ename attr1="value1" attr2='value2' .... >` .
- A reference to an entity always starts with an ampersand and ends with a semicolon, i.e. `&eref;` .
- The names of entities, elements and attributes are case sensitive.
- Comments are delimited by <!-- and -->, i.e. <`!-- This is an XML comment :-D -->`.

78 XML Syntax ...

- Empty elements are denoted by a slash before the closing bracket, i.e. <empty/> .
- A trivial XML example is:

```
<!-- This is a comment in my XML example -->
<myxmlexample>This is my XML example</myxmlexample>
```

Note the use of named tags, and paired opening and closing tags. XML will not tolerate missing closing tags in a statement, this is an intended design feature.

79 XML External DTD Syntax

- An XML DTD can be placed in a file external to the document, or embedded within the document. If the DTD is embedded, the document is said to be a 'standalone' document.

```
<?xml version="1.0"?>
<!DOCTYPE docu SYSTEM "http://www.blah.com/~doe/my.dtd">
<!-- This is an external DTD -->
<docu>
... <element1> ... <empel/> ... </element1>
</docu>
```

80 XML Embedded DTD Syntax

- The syntax for for an embedded DTD is also very simple. Square brackets are used to encapsulate the contents of the DTD.

```
<?xml version="1.0"?>
<!-- This is an embedded DTD -->
<!DOCTYPE myxml [
<!ELEMENT myxml  (#PCDATA)>
 ]>
<myxml> This is my XML example !  </myxml>
```

81 XML DTD Syntax

- An *element declaration* starts with <!ELEMENT and contains the type of the element and the model for its content.
- An *attribute declaration* starts with <!ATTLIST and for the given element contains one or more attributes, with type and default values.

- An *entity declaration* starts with <!ENTITY and contains the name and definition of an entity.
- A *notation declaration* starts with <!NOTATION and contains the name and external or public identifier used with a notation.
- A *processing instruction* is delimited with <? ...?> and contains data which is passed through to an application.
- A *comment* is delimited with <-- ... --> .

82 DTD Element Declarations

- Every element in an XML document must be declared.
- The declaration must always include the *type* and *content model* of the element.
- XML supports four different content models:
  1. An **empty** element, e.g. <ELEMENT emptyelement EMPTY>.
  2. **'Any'** element compatible with XML, e.g. <!ELEMENT blah ANY>.
  3. A list of, or nested list of **child** elements.
  4. A **mixed content** element, which includes both character data and child elements.

83 DTD Attribute Declarations

- The attributes of every element in an XML document must be explicitly declared.
- XML supports three categories of attribute **types**:
  1. **String types** defined by CDATA, e.g. <!ATTLIST bloggs name CDATA #REQUIRED>.
  2. **Tokenised types** which may be:
     (a) ID uniquely identifying a name.
     (b) IDREF, IDREFS referring to elements with an ID.
     (c) ENTITY, ENTITIES referring to entity names elsewhere in the DTD.
     (d) NMTOKEN, NMTOKENS referring to name tokens.

84 DTD Attribute Declarations ...

3. **Enumerated types** take one of a list of values defined in the DTD.
   (a) **notation types** with the keyword NOTATION followed by names of notations.
   (b) **enumerations** which are lists of name tokens associated with an attribute value.

85 DTD Attribute Declarations Example

Consider this example attribute declaration for a picture:

```
<!ATTLIST pict  name         ID      #REQUIRED
                size         CDATA   #IMPLIED
                title        CDATA   'Default Title'
                bordercolour NMTOKEN #FIXED 'blue'>
```

86 Entity Declarations

- Foreign material such as pieces of text, special characters, images and external files can be included in an XML document by using entity references.
- XML supports two types of entity:
  1. **General** entities, with declarations such as <!ENTITY GenEntityName GEDef>. These are referred to with an & ... ; pair, e.g. &GEDef;.
  2. **Parameter** entities, occuring only in the DTD with declarations of the form <!ENTITY ParEntityName PEName>. These are referred to by using a % ... ; pair, e.g. %PEName;.

87 External vs Internal Entities

- An **internal** entity has its value specified inside the DTD itself.
- An bf external entity is any entity not specified inside the DTD.

88

Style Sheets, CSS, XSL

89 References:

Goosens M. Rahtz S., *The Latex Web Companion*, Addison-Wesley, 1999.

Bert Bos et al, Editors, *Cascading Style Sheets, level 2, CSS2 Specification*, W3C Recommendation 12-May-1998.

90 The Style Sheet in HTML4 and XML

- Both HTML version 4. and XML employ a style sheet mechanism, based upon the **Cascading Style Sheets (CSS2)** specification.
- The aim of this mechanism is to provide these markup languages with a genuine capability to function as generic markup languages, in which style information is separated from content information.
- The style sheet mechanism is currently implemented using the CSS2 standard, with ongoing work being done on the **eXtensible Stylesheet Language (XSL)** and the **Document Style Semantics and Specification Language (DSSSL)**.

91 CSS2 versus XSL?

- While the CSS2 and XSL schemes both provide means of manipulating the style information in a document, independently of content, they are somewhat different in their exact aims.
- The CSS2 scheme is optimised for use on the W3, and is designed to accommodate both HTML and XML.
- The XSL scheme is optimised for more complex documents and a print publishing environment.
- If you intend to produce W3 documents using HTML4 and XML, familiarity with the CSS2 specification is very useful.

92 CSS2 Syntax

- The CSS2 scheme uses a very simple model, in which rules are defined to specify the properties used in rendering a document.
- Each rule comprises two parts:
  1. A 'selector' which identifies the feature of the document, e.g. in HTML a header such as <H1>.
- The CSS2 specification has more than 100 properties which can be manipulated.

93 CSS2 HTML Example (W3C)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
  <TITLE>Bach's home page</TITLE>
  </HEAD>
  <BODY>
    <H1>Bach's home page</H1>
```

```
    <P>Johann Sebastian Bach was a prolific composer.
  </BODY>
</HTML>
```

94 CSS2 HTML Example (W3C)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
  <TITLE>Bach's home page</TITLE>
  <STYLE type="text/css">
    BODY {
      font-family: "Gill Sans", sans-serif;
      font-size: 12pt;
      margin: 3em;
    }
  </STYLE>
  </HEAD>
  <BODY>
    <H1>Bach's home page</H1>
    <P>Johann Sebastian Bach was a prolific composer.
  </BODY>
</HTML>
```

95 CSS2 XML Example (W3C)

```
TICLE>
  <HEADLINE>Fredrick the Great meets Bach</HEADLINE>
  <AUTHOR>Johann Nikolaus Forkel</AUTHOR>
  <PARA>
    One evening, just as he was getting his
    <INSTRUMENT>flute</INSTRUMENT> ready and his
    musicians were assembled, an officer brought him
a list of the strangers who had arrived.
  </PARA>
</ARTICLE>
```

96 CSS2 XML Example "bach.css" (W3C)

```
INSTRUMENT { display: inline }
ARTICLE, HEADLINE, AUTHOR, PARA { display: block }
HEADLINE { font-size: 1.3em }
AUTHOR { font-style: italic }
```

```
ARTICLE, HEADLINE, AUTHOR, PARA { margin: 0.5em }
```

97

Literate Programming Tools

98 References:

Literate Programming Website,
`http://www.literateprogramming.com`

Literate Programming FAQ,
`http://shelob.ce.ttu.edu/daves/lpfaq/faq.html`

Norman Ramsey's **noweb** page,
`http://www.cs.virginia.edu/~nr/noweb`

Briggs P., *Nuweb Version 0.91, A Simple Literate Programming Tool*, User Manual.

99 LP Toolsets

- The first toolset for LP was developed by Donald Knuth. It was based upon the use of Pascal and TEX, and called *WEB*.
- Knuth later adapted *WEB* to use the C language and called the new toolset *CWEB*.
- Since then, a wide range of LP toolsets have been developed, and many researchers have continued working in this area.

100 LP Toolsets ...

- The best known LP toolsets are:
  1. Norman Ramsey's *noweb*, which uses any programming language and generates TEX, LATEX and HTML documentation.
  2. Preston Briggs' *nuweb*, which uses any programming language and generates LATEX documentation.

3. Ross Williams' *FunnelWeb*, which uses any programming language and generates TEX or HTML documentation.
4. Werner Lemberg's *c2cweb*, which uses C and C++, and generates TEX.
5. Markus Oellinger's *mCWEB*, which uses C and C++, and generates TEX.

101 Tangling and Weaving

- An LP toolset must perform two basic tasks.
- *Weaving* is the process of creating documentation from the LP source files.
- *Tangling* is the process of creating program source code for compilation from the LP source files.
- Tools for weaving mostly generate TEX or LATEX source files, which can then be compiled into formatted documents.
- Tools for tangling may be specialised, and work only with specific languages, or they may be more general and thus work with any language.

102 Nuweb

- The Nuweb toolset is one of the simpler LP packages, and has the advantage of being very fast to execute.
- Nuweb employs a single tool which performs both tangling and weaving, unlike most LP toolsets.
- Nuweb generates LATEX documentation output, and exploits LATEX features where possible to simplify its own syntax.
- Nuweb permits the use of multiple output files from a single Nuweb source file, to facilitate the use of multiple languages and work by multiple programmers.
- Nuweb does not support the 'pretty printing' function used in WEB / CWEB and also cannot generate an index of identifiers and variables.

103 Nuweb ...

- Most of a Nuweb source file (`filename.w`) will be written in LATEX syntax, and is directly copied through to the documentation file during weaving.
- Fragments of program source in a Nuweb file are termed *scraps*. A scrap is typeset in `verbatim` format in the documentation, or used to generate a program source file.
- Scraps are identified by a beginning and an end delimiter, in this manner: `@{myscrap@}`. Everything in between the delimiters is copied through without change.
- Scraps may also be used to encapsulate math mode expressions in LATEX. The syntax is then `@(mathmodescrap@)`.

104 Nuweb Syntax (Cited from Briggs)

Files and Macros:

`@o` ***file-name flags scrap*** Output a file. The file name is terminated by whitespace.

`@d` ***macro-name scrap*** Define a macro. The macro name is terminated by a return or the beginning of a scrap.

105 Nuweb Syntax (Cited from Briggs)

Scraps:

`@{`*anything*`@}` where the scrap body includes every character in *anything*—all the blanks, all the tabs, all the carriage returns. This scrap will be typeset in verbatim mode.

`@[`*anything*`@ ]` where the scrap body includes every character in *anything*—all the blanks, all the tabs, all the carriage returns. This scrap will be typeset in paragraph mode, allowing sections of TEX documents to be scraps, but still be pretty printed in the document.

`@(`*anything*`@)` where the scrap body includes every character in *anything*—all the blanks, all the tabs, all the carriage returns. This scrap will be typeset in math mode. This allows this scrap to have a formula which will be typeset nicely.

106 Nuweb Syntax (Cited from Briggs)

Macros inside scraps:

`@<`*macro-name*`@>` Causes the macro *macro-name* to be expanded inline as the code is written out to a file. It is an error to specify recursive macro invocations.

Note that macro names may be abbreviated, either during invocation or definition. For example, it would be very tedious to have to repeatedly type the macro name

```
@d Check for terminating at-sequence and return name.
```

Therefore, we provide a mechanism (stolen from Knuth) of indicating abbreviated names.

```
@d Check for terminating...
```

107 Nuweb Syntax (Cited from Briggs)

File output flags:

`-d` Forces the creation of `#line` directives in the output file. These are useful with C (and sometimes C++ and Fortran) on many Unix systems since they cause the compiler's error messages to refer to the web file rather than the output file. Similarly, they allow source debugging in terms of the web file.

`-i` Suppresses the indentation of macros. That is, when a macro is expanded in a scrap, it will *not* be indented to match the indentation of the macro invocation. This flag would seem most useful for Fortran programmers.

`-t` Suppresses expansion of tabs in the output file. This feature seems important when generating `make` files.

108 Nuweb Syntax (Cited from Briggs)

Minor commands:

`@@` Causes a single "at sign" to be copied into the output.

`@_` Causes the text between it and the next `@_` to be made bold (for keywords, etc.)

`@i` *`file-name`* Includes a file. Includes may be nested, though there is currently a limit of 10 levels. The file name should be complete (no extension will be appended) and should be terminated by a carriage return.

109 Nuweb Syntax (Cited from Briggs)

Indexing commands:

`@f` Create an index of file names.

`@m` Create an index of macro name.

`@u` Create an index of user-specified identifiers.

110 Nuweb Example (`nuweb.w`)

Global include file definition:

```
@o global.h
@{@<Include files@>
@<Type declarations@>
@<Global variable declarations@>
@<Function prototypes@>
@}
```

111 Nuweb Example (nuweb.w)

Include file macro definition:

```
@d Include files
@{#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
@| FILE stderr exit fprintf fputs fopen fclose
getc putc strlen toupper isupper islower isgraph
isspace tempnam remove malloc size_t @}
```

112 Nuweb Example (nuweb.w)

Preprocessor constant definition:

```
@d Type dec...
@{#ifndef FALSE
#define FALSE 0
#endif
#ifndef TRUE
#define TRUE 1
#endif
@| FALSE TRUE @}
```

113 Nuweb Example (nuweb.w)

Main file definitions:

```
@o main.c
@{#include "global.h"
@}
```

```
The first pass over the source file is contained in
\verb|pass1.c|.  It handles collection of all the
file names, macros names, and scraps (see
Section~\ref{pass-one}).
@o pass1.c
@{#include "global.h"
@}

The \verb|.tex| file is created during a second
pass over the source file. The file \verb|latex.c|
contains the code controlling the construction of
the \verb|.tex| file (see Section~\ref{latex-file}).
@o latex.c
@{#include "global.h"
@}

The file \verb|html.c| contains the code controlling
the construction of the \verb|.tex| file appropriate
for use with {\LaTeX}2HTML
(see Section~\ref{html-file}).
@o html.c
@{#include "global.h"
@}
```

114

End CSE-1402 Tools Stream Lectures