BLUe's Format

— The best of both worlds —

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rc.service.rug.nl

Abstract

An independent format—blue.fmt—is proposed to assist authors with creating, formatting, exchanging and maintaining compuscripts during the lifephases of publications. The format builds upon manmac.tex and the functionalities provided by tugboat.sty. Experience gained by publishers has been picked up too, because of my in-depth study of the activities of AMS with respect to TEX formatting. More recent work of Knuth and co-authors has been borrowed from gkpmac.tex.
The design goal was to provide a format which suits me, which is easy to customize—to the world outside, and in general to changing circumstances—and which complies with the adages of software engineering. Another aim of blue.fmt is that it can be used throughout the lifecycle of publications on modest equipment to format articles, transparencies and you name it. The hoped for lifety is a lifetime. En-passant the design process is accounted for.
New is the handling of a database of references—with cross-referencing—or pictures all in one-pass job.

Keywords: Active documents, design, documentation, education, error handling, floats, format, inner versus outer world, inserts, lifephases formats, macro writing, markup language, plain TEX, scripts, software engineering, (reusable) software parts, style, transparencies.

1 Introduction

After having studied manmac,[1] the TUGboat and AMS styles, I decided to combine the best of both worlds. The goodies given by Knuth to the world are merged with the achievements of the members of the various TEX users groups.

I decided to separate outer level markup from inner level markup. By splitting off as much as possible to the inner level the outer level will shrink. The outer level markup is suitable to account for articles, transparencies, and you name it. The amount of work to adapt a script for other environments is lessened by this approach.

We have to deal with the specifications for
- look-and-feel in print—typographical design
- markup language—syntax and semantics
- coding style—data types, control sequences.

The paper is an abridged version—let us say an appetizer—of the one available from the CTAN and NTG's fileserver.

1.1 Why?

At GUST '94 when presenting Manmac's BLUes, I was urged to provide a user's guide for manmac. BLUe's Format is my answer to that request. Whatever the value of blue.fmt it definitely serves an educational purpose, with a wink to active documents.

2 Design

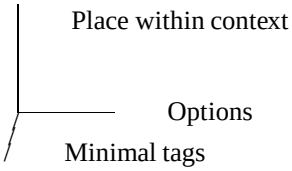
This comprises three aspects: typography, markup language, and coding conventions.

2.1 Typography

For the look-and-feel of BLUe's two-column in print I was inspired by TUGboat's layout, and started from their OTR. No switching from two-column into 1-column on a page. In one-column format pseudo two-column document parts[2] can be integrated.

2.2 Markup language

I like the orthogonal approach, for example to treat



[1] Knuth's macros for formatting his Computers and Typesetting series.

[2] See Knuth's mathdemos. This is a different approach to $1 \leftrightarrow 2$-column mixing, and an efficient one. The 1-column format as basis and markup for two-column document parts now and then. Another example is his formatting of the index. The latter output routine is similar to tugproc.sty, which starts with a 1-column opening, with the rest in two-column. Footnotes, and inserts in general, are not supported by the index OTR.

independent from each other. This approach has been on my mind throughout blue.fmt, but not applied unduly. Tags can be distinguished in

- `\begin<tag>`, `\end<tag>` pairs, which take global options via `\every<tag>` and local options via `\this<tag>`
- `\<tag>`, minimal markup
- token variables `\this-<tag>` and `\every-<tag>` for user guidance
- `\pre<tag>` and `\post<tag>` token variables
- token variables to convey information, like the title
- `\pasteup<tag>`-s, which position the formatted script element within context.

In a nutshell this comes down to the outer markup structures

```
\<tag>{...}     or     \begin<tag>
                        ...
                       \end<tag>
```

optionally preceded by

`\every<tag>{...}` and/or `\this<tag>{...}`

When the place within context should differ from the default supply[3]

`\pre<tag>{...}` and/or `\post<tag>{...}`

I did not bother about the size of the tags, because with nowadays (LA)TEX intelligent editors these tags don't have to be typed in anymore. Moreover, for TEX each tag represents one token independent of its length.

The bad news
is that minimal markup for `\head` and the like can be felt as an anachronism. With (LA)TEX intelligent editors we don't need minimal markup anymore. The reason to process headtitles and the like on the fly is that catcodes are assigned in phase while processing, meaning for example we can use verbatims as part of the title.

The good news
is that via this mechanism we can abstract at the *user level* from token variables and definitions. The user can supply the information as if it was a token variable.

2.3 What to start with?

The script should start with what I call information elements, preliminary matter such as title, keywords and the like. Provide the information via assignments to the token variables `\title`, `\subtitle`, `\issue`, next to providing the information for keywords, abstract and references, all in arbitrary order, as long as done before `\beginscript`.

Token variables have defaults. For example your blue.fmt has the token variables `\author`, `\address`, and `\netaddress` already filled in with defaults. If you wish to assign values to these token variables at the start of your compuscript, feel free to do so. That is the way of substituting actual values for the defaults. Handy isn't it?

The contents proper should start with `\beginscript`—to handle the paste up of the title matter—and end with `\endscript`—to finish the script. —Optionally preceded by `\everyscript` or `\thisscript`. When a script is formatted on its own the use of `\thisscript`, and especially `\everyscript` can be omitted.

For processing more than one script provide `\notlastscript` in `\everyscript`, and provide the last script with `\lastscript` in `\thisscript`.

2.4 Active documents

The revolution of computer-assisted typography has introduced the concept of active documents. For me this activity is all about

> the discrepancy of the ordering of elements as marked up in the compuscript, and the positioning and representation of the document elements in the result.

The hierarchy in which the document elements have been supplied does no longer imply a similar sequence in print. The ordering can be altered due to writing to and reading from the computer's memory, be it RAM (random access memory) or be it external memory with sequential access.

Actually, this happens with the title and the like. The keywords, abstract and contents are set in boxes and pasted up in the replacement text of `\beginscript`. Similarly, references have to be specified in the preliminary part.[4]

The references can be paste up by
`\pasteupreferences`.

New is that pictures can be selectively loaded in the preliminary part from the picture base. In order not to lose ground I will start from an informal template and formalise from there.

2.5 Template

The outer level markup is as follows.

```
%Template script---outer level markup
%Info part: elements in arbitrary order
\issue{MAPS 94.2}
\title{BLUe's Cross-Referencing}
\subtitle{---A one-pass approach}
\keywords{...}
\abstract{...}
\contents{Introduction
  \quad Why?
  ...
  Conclusion, References}

\references{\<name_1>...\<name_n>}
\pictures{\<name_1>pic...\<name_n>pic}

\beginscript            %Copy proper
\head{Introduction}
%various parts with inner markup
```

[3] This does not hold for the title part elements. Too much nitty-gritty. Not only the positioning within context is relevant but also the ordering.

[4] Numbers are assigned to the names to serve cross-referencing.

```
%Back matter
\pasteupreferences
\endscript
```

2.6 Outer markup

Generally I use an empty article as fill-in form.

The information part

consists of title,[5] author (address information), keywords, abstract, contents, references, pictures and the like, independent from whether these issues appear at the beginning in print or not.[6] The idea is to supply this information early in the script—and in fact author and address information are suited for defaults—such that it can be used anywhere.[7] By supplying the references also at the beginning we don't need a multi-pass job for cross-referencing.[8]

The `\beginscript` control sequence marks the beginning of the compuscript proper. Whether it concerns an article, report, book or a set of transparencies, `\beginscript` is a generic control sequence which has been defined appropriately within the context. It starts formatting the title part and processing the script elements which follow. The compuscript is ended by `\endscript` to abstract from `\end`.[9]

Note that the contents requires the table of 'contents' to be supplied in a *near natural* way: line by line. For indentation simply supply `\quad` or `\qquad`.[10]

Chapters and the like

are marked up by `\head` and subelements, with the title as argument.

Running headers and footers

make use of the token variables `\headline` and `\footline`, with the information automatically distilled from the title token variables, like `\title` and `\issue`.

Bibliography

handling is a much overstressed aspect IMHO. I like to keep it simple. Firts of all I like that the database of formatted entries to be already with the publisher, and that we only have to refer to the entries. But that seems wishful thinking still, although my approach comes close.

I use the macros and database which emerged from 'BLUe's Bibliography,' because it facilitates the selection from the database and the formatting all in one-pass job. The typograhical design has been inspired by ams.ppt style. It is customizable. Since 'BLUe's Bibliography' has been distributed in MAPS, I extended the macros with a simpler user interface.

Supply the names for the references as argument to `\references`, and paste up via `\pasteupreferences`.

`\references` not only formats the references but also redefines the names by their sequence numbers for cross-referencing.

Index

preparation can be done via manmac's writing of index reminders (IRs) to a file, and formatting the sorted, compressed and in general enriched IRs via manmac's doublecolumn environment.

2.7 Inner markup

Examples are borrowed from special papers on the issue.

(Special) Paragraphs

are for example provided by `\item`, and derivatives.

Example *(Bulleted items)*

I love Knuth's `\item`. It is so general and can easily be adapted to automatically format bulleted items—and others as well—via the use of[11]

```
\def\bitem{\item{$\bullet$}}
```

Example *(AN items)*

A lettered list can be obtained via[12]

```
\aitem first\\next line
\itemitem{} nested\\next line
\aitem second\\next line
\smallbreak
```

with results

a. first
   next line
      nested
      next line
b. second
   next line

[5] I favour a short title, optionally extended by a subtitle. Because of this I can use the title in the running head too, and there is no need for a separate specification of a running title.

[6] In regular issues of TUGboat the address information is printed at the end, while in the proceedings issues the address information is printed as part of the front matter.

[7] This is reminiscent of providing it at the beginning and typesetting it immediately. Nowadays the provision of the information is no longer tied up with formatting it immediately.

[8] The references are set and the names get their numbers as replacement text. Later in the script the markup command `\pasteupreferences` pastes up the earlier set references.

[9] Via this idea of scripts an editor can happily run `\onecol` and `\twocol` submissions in one job.

[10] No automatisms have been provided for automatically collecting the table of contents information while formatting. No multi-pass job!

[11] *The TEXbook*, exercise 14.20

[12] The numbered list is also part of blue.fmt.

For footnotes with automatic numbering I use `\ftn`, based upon *The TEXbook* exercise 15.12, which processes the footnote on the fly.

Quotations

start with `\beginquote` and end with `\endquote`. The effect is that the left and right margin are indented with `\smallskip` before and after.

Exercise and answer

markup macros are part of manmac. As known the answers are set at the end in Appendix A. Actice documents avant là lettre.

Math

from plain can be enriched by automatic numbering and cross-referencing.

Cross-referencing is a compatible extension. Provide the `\ref`, optionally followed by `\<name>`, at the place after the `\eqno` instead of an explicit number.

Example *(From Math into BLUes)*

- Labeled 1-line

$$\sin 2x = 2 \sin x \cos x \qquad (1)$$

- Two lines aligned, with labeling per line

$$\begin{aligned} \cosh 2x &= 2\cosh^2 x - 1 \qquad (2)\\ &= \cosh^2 x + \sinh^2 x \end{aligned}$$

and citations (1), (2).

via

```
\bitem Labeled 1-line
$$\sin2x=2\sin x\, \cos x
        \eqno\ref\cgla$$
\bitem Two lines aligned, with
       labeling per line
$$\eqalignno{
\cosh2x&=2\cosh^2x-1&\ref\cgld\cr
       &=\cosh^2x+\sinh^2x\cr}$$

and citations \crsref\cgla,
\crsref\cgld.
```

When I worked on 'Math into BLUes,' I considered it handy to extend `\eqalign` with a repetitive template such that more than one alignment point is accounted for. This extension has been incorporated in blue.fmt.

Tables

can be marked up by `\halign`,[13] or the alignment display. In manmac the alignment display is used within `\begindisplay`.

Example *(Table of markup tags)*

```
\begindisplay\displayindent\parindent
Tags  &Reg script\quad&Tansparencies\cr
\noalign{\vskip.5ex\hrule\vskip1ex}
\multispan3{Token variables\hfil}\cr
\cs{title}         &+      &+\cr
%et cetera
\enddisplay
```

At a higher level I use a bordered table model, where the first row and column are marked up separately from the data proper. Moreover, I abstracted from row and column separators, with the effect that rules or a frame can be obtained via the attributes `\ruled`, respectively `\framed`.

Example *(From 'What is TEX etc.')*

- just framed data

11	12
21	22

- add header and rowstubs

	Header	
1^{st} row	11	12
2^{nd} row	21	22

- vary with ruled and framed, add header and footer

Caption

	Header	
1^{st} row	11	12
2^{nd} row	21	22

Footer

via

```
\def\data{11\cs12\rs21\cs22}
\bitem just framed data
   $$\vcenter{\framed
     \beginbtable\data\endbtable}$$
\bitem add header and rowstubs
  \def\header{\logms2\hfill
                  Header\hfill}
  \def\rowstblst{{$1^{st}$ row}%
              {{$2^{nd}$ row}}}
  $$\vcenter{\beginbtable\data\
     endbtable}$$
\bitem vary with ruled and framed
  $$\def\btablecaption{Caption}
    \def\footer{Footer}
    \vcenter{\ruled\framed
    \beginbtable\data\endbtable}$$
```

[13] Or its transpose `\valign`.

Pictures

gkpmac provides LaTeX's picture environment in essence for plain. My added value to this collection is that I provided options via `\thispicture` and `\everypicture`, and created a database of pictures.

The new `\beginpicture` with uncoupled coordinates and offsets allows an interesting possibility to store a picture. An invocation comes down to

```
\<name>pic
%or with user guidance
\thispicture{\unitlength<dimen value>
  \xoffset{<number>}\yoffset{<number>}
}
\<name>pic
```

The database should contain the picture

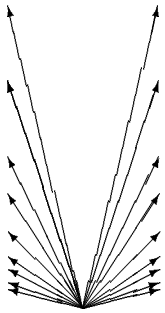
```
\def\<name>pic
{\bgroup\unitlength<dimen value>
 \xdim{<number>}\ydim{<number>}
\beginpicture
....%picture descriptions
\endpicture
\egroup}
```

with `\beginpicture` defined in the format to handle the `\thispicture` and the `\everypicture` among others.

Example *(Vectorbundle via gkpmac)*

```
\vectorbundlepic
```

yields



I assumed that the picture was loaded from the pic.dat file via

```
\pictures{\vectorbundlepic}
```

For an elaborate discussion consult 'BLUe's Graphs.'

Verbatim

macros are provided for in-line and display verbatim.

Example *(From BLUe's Verbatim)*

Important functionalities are shown below

```
%1. To handle via
%   default ! escape char
%   -- other fonts
%   -- footnotes
%   -- emc (enable metacode)
\thisverbatim={\emc
   \def\footnotetxt{{\rm Footnote text
    typed in on more than one line.}}}
\beginverbatim
Some <meta code> and
blah, blah, ...    !it
Now text in italics!tt
and back again in tt
footnote!ftn!ftntxt
!endverbatim
%
%2. To handle numbering and
%   verbatim file inclusion
\everyverbatim={\numvrb}
\thisverbatim={%
   \catcode`\!=12
   \catcode`\|=12
   \input vrb.tex
   \catcode`\*=0 }
\beginverbatim
Extras after file
*endverbatim
%
%3. To restart (line)numbers
\thisverbatim={\vrblin0 }
\beginverbatim
Just some text with
ligatures such as ?` switched
off (TB 381).
line numbers restarted via
\vrblin0 (in general  \numvrb).


After two blank lines.
!endverbatim
%
%4. Inline verbatim
%Minimal | tag
\makeactive\|
\def|{\bgroup\setupverbatim
   \the\everyverbatim
   \the\thisverbatim
   \def|{\egroup\thisverbatim{}}}
\thisverbatim{\emc}
Before \verb|<inline verbatim>| after.
```

Remark. The `|`-definition for the minimal tag has also been included in blue.fmt. Beware, don't use `|` as such for the symbol, but mark up with `\vrt` when just the symbol is needed.

The escape character in verbatim is very convenient. I use this for subscripting in verbatims, and for selective numbering of the macros in verbatim listings, meaning that code parts can start with a nice number, like 100, 200 and so on. This can be done by including

```
%<escape char>vrblin= 100
%<escape char>vrblin= 200.
```

Special texts

are for example programs. Manmac's syntax macros can be used to format the syntax rules.

Example *(The TeXbook p. 268)*

```
\beginsyntax
<unit of measure>\is%
  <optional spaces><internal unit>
\alt<optional {\tt true}>%
  <physical unit>
\endsyntax
```

The user must be aware that the end of line has been made active and apart from starting a new line looks for an opening <. Then it assumes that a new syntax rule has started. To continue the line provide a % at the end of the line in the script. Continuation via alternatives on the next line goes via `\alt`. See the syntax diagram later.

Handy

is manmac's possibility for shipping out selected pages. All what has to be done is to supply the page numbers in the file pages.tex, each number on a separate line, and in increasing order. Also handy are tugboat.cmn's abbreviations and macros for logos, to enhance consistency.

Syntax for outer markup

The template given earlier can more formally be described via the following syntax diagrams,[14] in the spirit of chapter 24 of the TEXbook. IM copy denotes script parts marked up with lower level TEX markup.

⟨blue script⟩ ⟶ `\input` blue.fmt ⟨scripts⟩

⟨scripts⟩ ⟶ ⟨script⟩ | ⟨script⟩ ⟨scripts⟩

⟨script⟩ ⟶ `\transparencies`⟨reg script⟩
| `\`⟨`yours`⟩ ⟨reg script⟩
| ⟨reg script⟩

⟨reg script⟩ ⟶ ⟨preliminaries⟩
⟨script options⟩
`\beginscript`
⟨sections⟩
`\endscript`

with refinements

⟨preliminaries⟩ ⟶ `\title`
| ⟨preliminaries⟩ ⟨preliminary⟩

⟨preliminary⟩ ⟶ `\subtitle`
| `\input` | ⟨TeX code⟩ | `\issue`
| `\author\address\netaddress`[15]
| `\keywords` | `\abstract`
| `\contents` | `\references`
| `\pictures`

⟨script options⟩ ⟶ ⟨empty⟩
| `\everyscript` | `\thisscript`
| `\everyscript\thisscript`

The sections consist of the tree of head structures with IM copy, inner marked up copy proper.

⟨sections⟩ ⟶ ⟨empty⟩ | ⟨section⟩ ⟨sections⟩

⟨section⟩ ⟶ `\head`⟨paragraphs⟩

⟨paragraphs⟩ ⟶ ⟨empty⟩
| ⟨paragraph⟩ ⟨paragraphs⟩

⟨paragraph⟩ ⟶ ⟨ssections⟩ | ⟨IM copy⟩

The above is repeated for sub(sub)sections.

The markup for the top level reads

```
\beginsyntax
<blue script>\is|\input|\thinspace%
         blue.fmt\thinspace<scripts>
<scripts>\is<script>\alt<script>%
                          <scripts>
<script>\is|\transparencies|%
                     <reg script>%
 \alt|\<yours>|<reg script>
 \alt<reg script>
<reg script>\is<preliminaries>
 \quad<script options>
 \quad|\beginscript|
  \qquad<sections>
 \quad|\endscript|
\endsyntax
```

3 Coding

I assumed as little TEX knowledge as possible for the outer level markup.

3.1 Conventions

Names are important. In pursuit of Knuth I favour `\begin<tag>` and `\end<tag>` pairs.

I like to program in paradigms and code these systematically given the language at hand.

For language dependent names I introduced token variables with names `\<tag>name`, for example for use within the environments keywords, abstract and references.

When I built upon available macros I added an extra level to comply with my markup conventions. When the original name of the macro had to be maintained at the outer level I changed the name with a prefix denoting the source.

Below I provide, and explain in detail, code for an outer level suite of macros—head—and code for an inner level suite of macros—AN-items. Because of the relevancy I also explained `\beginscript` with its `\pasteup<name>`-s and the one-pass handling of references.

3.2 Head and the like

I will design `\beginhead` and `\endhead`, next to the minimal markup variant `\head{...}`.[16]

[14] I refrained from more elaborated notational schemes, for example those used in the SGML world. The reason is simply that they are not needed to convey the idea. The basics of Backus-Nauer notation with some comments will do, similar to the way the syntax diagrams are used in the TEXbook.

[15] Defaults.

[16] The subhead variants are similar but different in detail. Especially in undoing glue when heads follow-up each other without intervening text. Beware!

Design goals

are

- discourage to set the title alone at the end of a page
- typeset (flexible) vertical space before (big) and after (med) the title
- gobble spaces at the beginning of the title
- set the title in bold face (and the current size) unindented
- don't indent the first line of the text after.

What are the problems?

The language constructs to be adopted are a problem. In macro expansion the two-part macro TEXnique should be the basis. In blue's format I adopted therefore the pairs `\begin<tag>` and `\end<tag>`.

In order to set the title loose from the context we have to determine values for the amount of glue, and be sure that it disappears at the top of a page. Another problem is to prevent the head text alone to be printed at the bottom of the page. The latter is related to the 'orphan-widow' phenomenon.

The coding solution below has different values for the parameters as supplied in `\beginsection`. From experiments it turned out that those supplied by Knuth did not give nice results together with 'BLUe's Format.' Too much glue was inserted for my taste.

The idea of a `\pre<tag>` and `\post<tag>` is general. It parameterizes the placement of a document element within context. Important! Whether you talk about titles, displays, tables, graphs, or you name it.

The code below is inspired by `\beginsection`, *The TEXbook*, p355.

```
\def\beginhead{\prehead\bgroup\headfont}
\def\endhead{\egroup\posthead}
%and auxiliaries
\def\prehead{\vskip0pt plus5ex
   \penalty-250\vskip0pt plus1ex
   \bigskip\noindent}
\def\posthead{\medskip\nobreak
   \noindent}
```

When deprived from a (LA)TEX intelligent editor use the following minimal variant, which has the same functionalities, especially the processing on the fly of the 'argument.'[17]

```
\def\head#{\beginhead\bgroup
   \aftergroup\endhead
   \afterassignment\ignorespaces
   \let\dummy=}
```

Explanation. The # as last character of the parameter text makes that the opening brace and what follows is placed after the replacement text of `\head`, *The TEXbook*, p204. The replacement text starts a group via `\bgroup` and reads away the opening brace. The latter TEXnique has been borrowed from plain's `\footnote`, *The TEXbook*, p.363, as used at the end of the replacement text of `\fo@t`. Spaces which precede the title are ignored via the invoke of `\ignorespaces` at the right time. Sigh, quite something isn't it?

Paradigm

From the above variant the following template can be distilled, to be used with *similar* two-part macros for a minimal markup variant

```
\def\<tag>#{\begin<tag>\bgroup
   \aftergroup\end<tag>
   \afterassignment\ignorespaces
   \let\dummy=}
```

Although nice it has its limitations.

Some remarks

Note that I did not gobble spaces at the end of the title argument. In the headtitle they won't harm because the title is set on a line of its own.[18]

In the unabridged version the template is automated too via partial expansion.

3.3 Items with AN

With automatic numbering and lettering we have to stop the automatism of increasing the counter and to reset the counter for subsequent use at the end of the list.

In `\item` nothing special had to be done. Ending the paragraph via a blank line (or `\par`) was enough. We cannot modify `\par` locally because `\par` starts the replacement text of `\item` (and `\itemitem` as well). So the natural markup to let a blank line (or `\par`) end the AN-list is not possible.

I could not work along the remark made in *The TEXbook* exercise 14.29 either, because this goes wrong with nesting, read with the use of `\itemitem`.

My solution is a mixture of coding borrowed from `\item` p355, `\beginchapter` p418, `\d@nger` p419 of *The TEXbook*.[19]

```
\newcount\itemno
%item with automatic numbering
\def\nitem{\bgroup
   \def\nitem{\advance\itemno1
      \item{\number\itemno.}}
   \def\smallbreak{\endgraf\egroup
      \smallbreak}
   \let\smallskip\smallbreak
   \nitem}
%item with automatic lettering,
%that is with a. b. etc
\def\aitem{\bgroup\itemno96
   \def\aitem{\advance\itemno1
      \item{\char\itemno.}}
   \def\smallbreak{\endgraf\egroup
      \smallbreak}
   \let\smallskip\smallbreak
   \aitem}
%end with \smallbreak or
%with \endlist
\def\endlist{\endgraf\egroup}
```

[17] And if you don't use it as such peruse it, because of the macro writing TEXniques.

[18] In the other heads the space at the end of the argument can be there on purpose. Watch out!

[19] The more I come to think of it the more ways of doing boil up. How to decide which one is best?

We not only have to end the paragraph(s) but also end the scope.

The paradigm
is the redefinition of a control sequence when this control sequence is used repetitively but with a different function the first time. The undoing of the redefinitions via closing the group is also a general macro writing feature.

3.4 Beginning of the script

This is included because it shows the paste-up of the various title elements.

```
\def\beginscript{\lastscript
   \the\everyscript\begingroup
   \the\thisscript
   \hrule\kern2ex\noindent
   {\titlefont\the\title
   {\subtitlefont\the\subtitle}}
   \medskip\the\author
   \medskip\pasteupkeywords
   \medskip\pasteupabstract
   \medbreak\pasteupcontents
   \smallbreak}
\def\endscript{\makesignature
   \xcol=\maxcols
   \vfil\eject\endgroup\tracingstats1
   \stop\thisscript{}}
%with auxiliaries
\def\pasteupkeywords{\box\keywordsbox}
\def\pasteupabstract{\box\abstractbox}
\def\pasteupcontents{\unvbox
               \contentsbox}
\def\notlastscript{\global
    \let\stop\relax}
\def\lastscript{\global\let\stop\end}
```

3.5 One-pass handling of references

Various mental steps are integrated.

- maintain an independent database of references
- provide a references macro to perform the tasks of selecting and formatting
- the list of names has to be supplied as argument to the macro, and is used twice, for selecting and formatting
- load selectively from the database
- format the selected entries
- redefine the names of the selected entries by their number
- provide a paste-up tag.

In the code I made use of the fifo paradigm.

```
\def\beginreferences#1\endreferences{%
  \bgroup\def\process##1{\gdef##1{%
Reference {\tt\string##1}
not in database (Sorry!)
  \loaderror{Reference}}}\fifo#1\ofif
  \loadselectivefrom{lit.dat}
  %formatting
  \ifstore\global\setbox\referencesbox=
    \vbox\bgroup\fi
  \bcnt0 \lsams%Default ls
  \the\thisreferences
  \def\process##1{\ls{##1}
    \xdef##1{\the\bcnt}}
  \fifo#1\ofif
  \endgraf\endreferences
}
\def\endreferences{\egroup
  \ifstore\egroup\fi
  \thisreferences{}}

\def\pasteupreferences{%
  \the\prereferences
  \unvbox\referencesbox
  \the\postreferences}

\prereferences{\head{\the
  \referencesname}}
\postreferences{}

\def\references#1{\beginreferences#1
   \endreferences}
```

with the auxiliaries

```
\def\loadselective{#1}
{\let\x\def%or any name for x
 \x\def##1{\ifx##1\undefined\ea\gobble
         \else \ea\x\ea##1\fi}
 \input #1
 \let\def\x}
%and
\def\gobble#1{}
```

Explanation. The selective loading is hinted at in *The TEXbook*, p382, Selective loading of macros. The idea is to consider `\def` as the name of a macro with as argument the name of the entry to be considered. Remind that lit.dat contains entries of the form

```
\def\<name>{...}
```

Then if the name is not yet known the replacement text is gobbled otherwise it is stored under its name. Neat isn't it?

3.6 Selective loading of pictures

This is done similarly.

```
\def\beginpictures#1\endpictures{%
  \bgroup\let\commonforpic=x
  \def\process##1{\gdef##1{Picture
  {\tt\string##1} not in database,
  Sorry!\loaderror{Picture}}}%
  \fifo#1\ofif
  \loadselectivefrom{pic.dat}
  \endpictures\commonforpic}
%
\def\endpictures{\egroup}
\def\pictures#1{\beginpictures#1%
   \endpictures}
%
\def\loadallpictures{\input pic.dat
   \commonforpic}
%
\def\loaderror#1{\write16{#1 not in
      database}}
```

Explanation. `\commonforpic` is a defintion which contains common elements for all pictures. I did not want the user to bother about this, and also invoked it at the end. The rest is similar to the handling of references.

4 Software Engineering

Not much of this of yet, except for the explicit design goals and the nice error message with `\references` and `\pictures` when a name contains typos.

5 Customization

As example I treat the adaptation of a blue script into a MAPS script, which is LaTeX biased.

```
%Template for MAPS submissions
\documentstyle[twocolumn]{maps}
\begin{document}
\title{...}    \subtitle{...}
\author{...}   \date{}
\maketitle

\begin{abstract}...\end{abstract}
\begin{keywords}...\end{keywords}
%Contents table
%Copy proper
\section{...}%et cetera
%back matter, LaTeX's
\begin{thebibliography}{abcde}
...
\end{thebibliography}
\end{document}
```

5.1 Conversion

The title block is filled in or simply reused from maps.tem. Globals and running-in heads need special attention. I fell back upon LaTeX's bibliography handling. Laborous is to 'translate' the various `\this<tag>` and `\every<tag>` directives. Some transformations are shown in the accompanying table.

BLUe fmt		MAPS sty
`\begin` abstract `\end`	⟶	`\begin{abstract}` `\end{abstract}`
`\beginscript`	⟶	`\begin{bijlage}`
`\head`	⟶	`\let\head\section`
`\bitem`		`\begin{itemize}`
...	⟶	`\item...`
`\bitem`		`\item...` `\end{itemize}`
`\begin` `<esc char>end` verbatim	⟶	`\begin{verbatim}` `\end{verbatim}`
`\endscript`	⟶	`\end{bijlage}` `\end{document}`

6 Related work

First a disclaimer. It is not possible to really compare blue.fmt with other formats because of different goals. What follows below has to be read as a rough indication of how blue.fmt relates to the works mentioned.

In a sense LaTeX provides the functionalities I'm after. However, LaTeX is complex and customization is too time-consuming. More important is however that LaTeX lacks the two-part macro TeXnique as basis. The simple and innovative approach of Knuth via two-part macros has not been appreciated appropriately.

Doob's macros for typesetting his 'Gentle introduction to TeX' is too limited, especially when we consider the life-cycle of documents.

Berry builds upon plain alone and neglects other achievements.

Spivak provides much of LaTeX's functionalities in LaMaSTeX but he refrains from plain's math markup.

The TeXinfo.tex macros look as if designed bottom-up.[20] No attempt for simplicity, nor a set of common markup tags as a foundation to build upon.

TUGboat's OTR functionality lies at the heart of the page makeup of blue.fmt. However blue.fmt goes much further than tugboat.sty in for example the markup for pictures, references, (bordered) tables, verbatims, and math cross-referencing, all designed for use with (LA)TeX, that is it is not tied up with LaTeX or plain, exclusively. Add to that the way how references and pictures are selected from a database and you will agree that blue.fmt is a leap forward. The coding of 'options' is less monolithic, and the setup via modules with thin interfaces is more flexible. blue.fmt is not a goal per se. It has an open eye for change in general, and adaptation towards other contexts in particular.

7 Transparencies

Start with `\transparencies` after `\input` blue.fmt.

7.1 Typography

Only recently I realized that transparencies are completely different in look-and-feel from pages in a book. Each transparency is more or less on its own, and has a strong centered flavour.

My running heads bear contextual information, a functionality similar to the running heads of a book. From each transparency the audience can see at a glance what the context is. The running foot takes the date, a number and the copyright.

In 'BLUe's Transparencies' I have included pictures showing the layout of the title transparency and the regular transparencies.

7.2 Markup

As extras I needed token variables for the head titles, because I like to suppress the headers on the transparency proper now and then. Also needed are things like 'continue on the next transparency,' or just ship out the current transparency.

Template for transparencies

The accompanying template shows that the outer markup is nearly the same as for articles. It is a blue script.

```
\transparencies %Template script
\title{Manmac BLUes}
\subtitle{---how to ...---}
\contents{User's guide
Coding
```

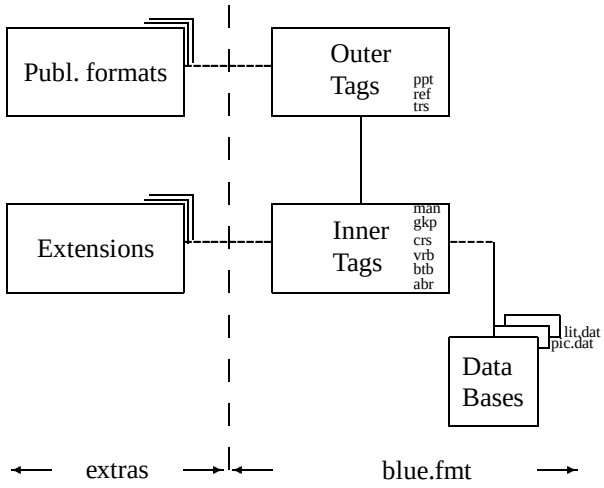
[20] What is the use of a meta-parsing macro—very clever there is no question about that—if we can do without this functionality in ordinary formats?

```
Modifications
Conclusions}

\beginscript
\head{Why?}
To return to the roots...

\head{User's Guide}
%Relative page numbering
\pagenumber{U\folio}\pageno=1
...
\nxttrs %Next transparency
\subsubhead{Markup paragraphs}
...
\continue%Continuation on next slide
...
\endscript
```

Remarks. I allowed for the possibility to number transparencies by parts. For example User's guide and Coding can take their own page numbers. I find that convenient when I modify transparencies.[21]

8 What?

The file blue.fmt starts with an enumeration of the parts included. Each part starts and ends with a %-line denoting which part started or ended. At the end of each part a table of contents is supplied.



8.1 Where?

blue.fmt, lit.dat, pic.dat and the article fmt.art are available at the CTAN sites in directory . . . , and
NTG's fileserver in directory . . .

9 Acknowledgements

There are too many people to be named whom I owe much while working on both the BLUes and BLUe's series. They all have helped me a lot. One exception though is for NTG, in allowing me to bring out early versions of the papers in MAPS. Thank you!

10 Conclusions

Macro writing with respect to format or style development is a special way of programming and to embed this within the realm of software engineering is a real challenge, and much needed.

blue.fmt is beyond manmac because it

- distinguishes two markup levels
- allows for tranparencies and your format to be added
- automatically numbers or 'letters' items
- has awareness of databases to be coupled
- automatically numbers and cross-references
- allows two and one column layout
- contains generic verbatims, bordered tables and gkp's picture environment, and
- automatically selects and formats references supplied in the literature database.

My added value to manmac and gkppic is

- integration of these within a two-column and one-column environment, especially articles and transparencies
- separation of specification and paste up
- `\this<tag>{...}` and `\every<tag>{...}` extensions
- `\pre<tag>{...}` and `\post<tag>{...}` parameterizations
- literature and (new) graphics database
- one-pass bibliography handling
- formatting references in the AMS spirit
- cross-referencing
- outer and inner markup separation
- a verbatim suite
- a bordered table macro
- transparency macros
- customization outer markup.

With respect to coding I added the paradigm—a real pearl to paraphrase Bentley—to add systematically and automatically a one-part macro with the same functionlities on top of a two-part macro.

My thesis is that users can benefit from blue.fmt in typesetting like a craftsman, to achieve the quality of Knuth.

The extra bonus is stability, and simplicity as well. The disadavantage is investment in learning, which by trial-and-error takes a substantial amount of time. It has all to do with your attitude, whether you believe that TEX proper will serve a lifetime, and whether you like to invest in learning plain and manmac as basis. The reward is freedom. You are no longer dependent upon the gurus for what-and-when.[22]

[21] In 'BLUe's Transparencies' I have paid attention to the maintenance aspects. How to modify just a few transparencies, or how to add a couple to the collection, without the need to rerun the total collection.

[22] Don't believe the argument that plain and manmac don't support structured markup. That is not true. Peruse the TEXbook script and you will know better.

Remind De Vinne's adage

'The last thing to learn is simplicity'

Look at the results as provided in the TEXbook and peruse the markup in the compuscript,[23] and then simply practise Knuth's way of markup.

11 References

The unabridged version of the paper contains the references to the works I have built upon.

12 Appendix: Control sequences

The header denotes the minimal one-part macro, the prefixes pre-, post-, this-, every-, and the postfixes -name and -box.

Regular scripts

	min	pre	post	this	every	name	box
abstract	+	¬	¬	¬	¬	+	+
btable	▯	+	+	+	+	¬	+
center	+	¬	¬	¬	¬	¬	¬
contents	+	¬	¬	¬	¬	+	+
head	+	+	+	¬	¬	¬	¬
keywords	+	¬	¬	¬	¬	+	+
picture	¬	¬	¬	+	+	¬	¬
pictures	▯	¬	¬	¬	¬	¬	¬
quote	+	¬	¬	¬	¬	¬	¬
references	▯	+	+	+	+	+	+
script	+	¬	¬	+	+	¬	¬
subhead	+	+	+	¬	¬	¬	¬
subsubhead	+	+	+	¬	¬	¬	¬
subtitle	▯	¬	¬	¬	¬	¬	¬
syntax	+	¬	¬	¬	¬	¬	¬
title	▯	¬	¬	¬	¬	¬	¬
verbatim	¬	+	+	+	+	¬	¬

¬ not implemented
\+ implemented
▯ implemented without processing on the fly.

[23] Available in the public domain, also on NTG's 4AllTEX CD-ROM, and undoubtedly on others.