The altTEX package

Arno L. Trautmann*

Version 0.a.3 January 21, 2009

This is the package `alttex` which will try to give an experimental new way to write X∃LATEX[1] code. So far it is mostly done with very dirty code and actually it's a collection of things that come into my mind during boring lectures. Maybe someone will have fun with the following code fragments.

Contents

---

*arno.trautmann@gmx.de
[1]If you don't know about X∃LATEX, see the appendix.5.2

1 Introduction

The problem I have with LaTeX[2] is the antique way of typing. Because most people still use a hopelessly outdated keyboard layout (»qwerty« or slightly adapted versions of that), LaTeX doesn't make use of some cool features. I'm not talking about writing chinese or arabic text! Maybe this example will make the idea clear:

In standard LaTeX, one has to write

```
This is the normal text, then comes the itemization:
\begin{itemize}
  \item text for first item
  \item text for second item
    \begin{itemize}
      \item this is an item inside an item…
      \item[$\Rightarrow$] Here an item with a formula: $\int_a^b x^2 dx$
    \end{itemize}
  \item and the outer itemize goes on…
\end{itemize}
And your normal Text goes on…
```

Using this package and having a superior keyboard layout[3], you can simply write:

```
This is the normal text, then comes the itemization:

• text for first item
• text for second item
  ‣ this is an item inside an item
  ‣[ ⇒] Here an item with a formula: $∫_a^b x² dx$

• and the outer itemize goes on…

And your normal text goes on…
```

Compare this with the output, which is the same in both cases:

This is the normal text, then comes the itemization:

- text for first item

- text for second item

  − this is an item inside an item…

  ⇒ Here an item with a formula: $\int_a^b x^2 dx$

---

[2] I'll write LaTeX instead of X_ƎLaTeX—saves me two keystrokes. Most of the code below *only* works with X_ƎLaTeX. If you need support for [utf8]inputenc or LuaLaTeX, please contact the author.

[3] E.g. the ergonomic layout Neo: http://neo-layout.org/

- and the outer itemize goes on...

And your normal Text goes on...

The aim of this package is to offer a more intuitive, kind of „wysiwyg" way, without loosing anything of logical markup. One still can re\define the • if he doesn't like the way his items look. Also, I try to make some things easier that are annoying in everyday TEXing. I have just started to write the package, there will be much more stuff here in the future.

2 Implementation

Ok, enough blahblah, now comes the code. We begin with the uninteresting preamble stuff:

```
1 \ProvidesPackage{alttex}
2
3 \RequirePackage{amsmath}
```

\usepackage Now, this is the first highlight. It is an extremely simple and stupid approach to load missing packages on-the-fly, just like MikTEX does. We re\define the \usepackage and hope, it works. This only working with texlive! If you're using MikTEX, put a

```
\let\usepackage\oldpackage
```

into your preamble, *directly* after loading alttex. If this does not work, delete the following lines from your alttex.sty.

```
 4 \let\oldpackage\usepackage
 5 \def\usepackage#1{
 6   \IfFileExists{#1.sty}{
 7     \oldpackage{#1}
 8   }{
 9     \immediate\write18{tlmgr install #1}
10   }
11 }
```

So far, this code seems to be a bit buggy, but it should work anyhow.

Now load some nice packages and testing wether you're running X⅁LATEX or not. We need exscale to write really big formulae, and ifxetex to check wether one uses the correct engine. hhline is used for the tabular experiments.

```
12 \RequirePackage{exscale}
13 \RequirePackage{ifxetex}
14 \RequirePackage{hhline}
15 \ifxetex
16 \else
17   \typeout{^^J%
18   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
19   ! This package can only be compiled with XeLaTeX. ^^J%
20   ! pdfLaTeX cannot handle unicode the way it is used here. ^^J%
21   ! If you want to have support for [utf8]inputenc, please contact the author. ^^J%
22   ! If you want to use LuaLaTeX, give it a try: ^^J%
23   ! comment out the lines 32,33,35–43. ^^J%
24   ! Please e-mail me the result of your experiences!^^J%
25   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
26   }
27   \errmessage{The package alttex is only working with XeLaTeX. \\ See the log message for more details.}
28   \endinput
29 \fi
30
```

3 Textmode

3.1 no escape

\noescape — You want to write plain text. Maybe you're annoyed by always escaping characters like _ # & { } $ ~ and so on. \noescape allows you to never escape anything—except the \, which still might be used for \textit{} or so. Or maybe not... because the { } are not escaped. Have to think about this one. Maybe the \ will be redefined to define { } by itself.

```
31 \def\noescape{
32   \catcode`\_= 11%
33   \catcode`\^= 11%
34   \catcode`\#= 11%
35   \catcode`\&= 11%
36   %\catcode`\{= 11%
37   %\catcode`\}= 11%
38   \catcode`\$= 11%
39   \catcode`\~= 11%
40   \makeatletter%
41   \catcode`\%= 11
42 }
```

The \makeatletter is not necessary. But it fitted into this line, so I will leave it here.

\oldescape — Of course this has to be reset when doing anything like formula, tabular etc. Maybe I will be able to change the behaviour automatically. This idea has been inspired by a discussion on the ConTeXt mailinglist.

```
43 \def\oldescape{
44   \catcode`\%= 14%
45   \catcode`\_= 8%
46   \catcode`\^= 7%
47   \catcode`\#= 6%
48   \catcode`\&= 4%
49   %\catcode`\{= 1%
50   %\catcode`\}= 2%
51   \catcode`\$= 3%
52   \catcode`\~= 13%
53   \makeatother%
54 }
```

3.2 tabular

The way one has to type extensive tabulars is quite complex – and the resulting code is often not easy to read. I don't have good ideas how to change this, but I'm thinking about it. Mail me any suggestions for this!

This will be the first attempt to make tabulars easier: Mostly you want an \hline after an \\. So let's try something like:

> I will try to implement cool stuff from the hhline-package.

Type `\–` (an en-dash) at the end of a line, and you get an `\hhline`. Type `\=` to get a double line

```
55 \def\–{\hhline}
56 \def\={\hhline}
```

This is shurely not a good symbol for this purpose, but I don't have a better idea so far. At least it's a "bar", so one can guess what it should do.

3.3   excel tabulars

`\exceltabular` Often one usese a program to calculate tabulars of numbers. To insert it into LaTeX, one has to do some work. Here we try to copy-paste the tabular from excel, Calc or any other program to a file mytabular.txt (or any other ending). Then you say `\exceltabular{mytabular}` (you do not need the ending, therefor it doesn't matter) and you get the tabular in a standard format. I will extend this to enable caption, variable number of columns, kind of rule used etc. This is just a very first test.

This is the definition of the command:

```
57 \def\exceltabular#1{
58   \catcode`\^^I=4\relax
59   \eolintabular%
60   \begin{tabular}{|c|c|c|}\hline%
61   \input{#1}%
62   \end{tabular}%
63   \catcode`\^^M=5\relax
64 }
```

And a little helper function to make the <enter> `\active`. Again, thanks to the people on the mailinglists.

```
65 \def\mybreak{\\\hline}
66 \begingroup
67   \lccode`\~=`\^^M%
68 \lowercase{%
69   \endgroup
70   \def\eolintabular{%
71     \catcode`\^^M=\active
72     \let~\mybreak
73   }%
74 }
```

3.4   tabbing

`\alttabbing` This will be analog to the `\exceltabular`. You write your tabbing using tabs and <enter>. That's it :)

```
75 \iffalse not yet implemented!\fi
```

4 Math stuff

4.1 braces

\newbraces
\oldbraces
Now this is something most LATEX-beginners don't recognize and wonder why the formula looks so ugly: The braces () do not fit to the hight of the formula. This can be achieved by putting `\left` and `\right` in front of the braces. But actually, this is annoying! In almost any case you want this behaviour, so this should be the standard. So we redefine the way braces are handled. With `\newbraces` the ( ) always fit. If you prefer the normal LATEX way, use `\oldbraces` to reset everything. This new behaviour should be extended to other characters like | [{ < and so on. Maybe in some later version.

There is another nice benefit on the second view: If you forget a) in your formula, no-one will notice until you have the printed output. With the definition given here, you will get an LATEX-error so you cannot compile when) are missing.

I would have never been able to implement this without the help of the mail-inglist members of tex-d-l@listserv.dfn.de!

The redefinition of `\mathstrut` is necessary when using amsmath (you will use amsmath when typesetting formulae, won't you?), because the hight of formulae is determinated by the hight of a brace. But using ( ) as `\active` characters, we need another brace here. So we take [. This will probably also change. But the code is working fine for ( ).

> **The newbraces does *not* work at the moment!**

> **Maybe one could "temporarily hardcode" the hight of [and then use this...**

```
76 \makeatletter
77 \def\resetMathstrut@{%
78     \setbox\z@\hbox{%
79         \mathchardef\@tempa\mathcode`\[ \relax
80         \def\@tempb##1"##2##3{\the\textfont"##3\char"}%
81         \expandafter\@tempb\meaning\@tempa \relax
82     }%
83   \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
84 }
85 \makeatother
86
87 {\catcode`\(\active \xdef({\left\string(}}
88 {\catcode`\)\active \xdef){\right\string)}}
89
90 \def\newbraces{
91   \mathcode`("8000
92   \mathcode`)"8000
93 }
94
95 \edef\oldbraces{
96   \mathcode`(\the\mathcode`(
97   \mathcode`)\the\mathcode`)
98 }
```

4.2 huge display math

Sometimes, especially in presentations, you might need an really big formula. Imagine two hours of struggle with transformations—and finally there is the beautiful formula. Now you can say

```
\begin{hugedisplaymath} E = mc^2 \end{hugedisplaymath}
```

There should be several steps of size, maybe.

```
 99 \def\hugedisplaymath{
100   \makeatletter
101     \makeatother
102   \Huge
103     \begin{equation*}
104 }
105 \def\endhugedisplaymath{
106   \end{equation*}
107 }
```

4.3 unicode math

Typing math in TeX is no great fun – you have to write things like \int instead of ∫ and so on. Have a look at the following formula:

```
\int_\infty^\infty \sum_a
```

The code again is stolen and I don't understand, why it does what it does, but it does it: The first argument is the character you want to use for "unicode math", the second one is the TeX-command.

```
108 \makeatletter
109 \def\altmath#1#2{%
110   \expandafter\ifx\csname cc\string#1\endcsname\relax
111     \add@special{#1}%
112     \expandafter
113     \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
114     \begingroup
115       \catcode`\~\active  \lccode`\~`#1%
116       \lowercase{%
117       \global\expandafter\let
118         \csname ac\string#1\endcsname~%
119       \expandafter\gdef\expandafter~\expandafter{#2}}%
120     \endgroup
121     \global\catcode`#1\active
122   \else
123   \fi
124 }
125 \makeatother
```

We will make a switch to turn this stuff on or off, so it does not interfere with the unicode-math package. This list will increase by time. If you are missing a

symbol, just send me the `\altmath{X}{\Xcode}`-line. I would be very thankful if anybody could send me a whole list of symbols!

```
126 \def\makealtmath{
127 \altmath{α}\alpha
128 \altmath{β}\beta
129 \altmath{γ}\gamma
130 \altmath{δ}\delta
131
132 \altmath{⇒}\Rightarrow
133 \altmath{⇐}\Leftarrow
134 \altmath{⇔}\Leftrightarrow
135
136 \altmath{∫}\int
137 \altmath{∀}\forall
138 }
```

There will be an `\makenormalmath`-switch as well.

4.4 Lazy underscript and superscript

Sometimes one has to make extensive use of subscripts and superscripts, e. g. when typing long formulae including tensors. Then it is a bit annoying to always write the `{}`, especially when there are only two letters in the sub/superscript. So let's try to implement the possibility to type `$F_µν F^µν $`.

First, store the actual meaning of `_` and `^` in `\oldunderscore` and `\oldhat`.

```
139 \let\oldunderscore\_relax
140 \let\oldhat^\relax
```

Now set `_` as `\active` char and define it the way we want it to behave. For this, we need the space char and end-of-line char to be an egroup char. So the underscript group is ended by space or eol and we don't need to close it explicitly.

```
141 \catcode`\_=13
142 \def_{%
143   \ifmmode
144     \catcode`\ =2\relax%
145     \catcode`\^^M=2\relax%
146     \expandafter\oldunderscore\bgroup%
147   \else%
148     \textunderscore%
149   \fi%
150 }
151
152 \iffalse
153 This does not work so far…
154 \catcode`\^=13
155 \def^{%
156   \ifmmode
157     \catcode`\ =2\relax%
158     \catcode`\^^M=2\relax%
```

An underscore at the end of an inline-formula has to be ended with } or egroup. That is not nice…

The redefinition of hat does not work because TeX uses it for definition of catcodes. There has to be a really tricky way to get around that.

```
159    \expandafter\oldhat\bgroup%
160  \else%
161    \oldhat%
162  \fi%
163 }
164 \fi
```

To give the possibility to swith between normal and `alttex` behaviour, store the new underscore.

```
165 \let\advancedunderscore_
```

And the switches. By default, _ is active. Type `\oldUnder` to get the normal _.

```
166 \def\oldUnder{
167   \global\catcode`\_=8\relax
168 }
169 \def\newUnder{
170   \global\let\_\advancedunderscore
171 }
```

5 Lists and such things

5.1 itemize with a single character

• instead of \item   We use an active character (here it is the unicode bullat character •) for the whole `itemize`-construct, and another one for nested itemizations (like a triangular bullet ▸). So far, only two-level nesting is possible, but that is okay for most cases. Deeper nesting is still possible with another `\begin{itemize}`.[4]

\newitemi   First, we define some little helpers:

\newitemii
```
172 \def\outside{o}
173 \def\inside{i}
174 \let\insideitemizei\outside
175 \let\insideitemizeii\outside
```

The following code defines the conditional insertion of the `\begin{itemize}`. This will be assigned to an active character using `\makeitemi` and `\makeitemii`, respectively. To end an `itemize`, just hit <enter> two times (to create a blank line). This will then insert a `\end{itemize}`. You can *not* have an empty line inside an `item`; this will always end the `itemize`.

```
176 \def\altenditemize{
177   \if\altlastitem 1%
178     \let\altlastitem0%
179   \else%
180     \end{itemize}%
181     \let\insideitemizei\outside%
```

---

[4]Take care of the ending of your `itemize`! The `itemize` *must* be followed by an empty line. (Do not use

or

! Use two <enter>.)

```
182   \fi%
183 }
184
185 \begingroup
186   \lccode`\~=`\^^M%
187 \lowercase{%
188   \endgroup
189   \def\makeenteractive{%
190     \catcode`\^^M=\active
191     \let~\altenditemize
192 }%
193 }
194
195 \def\newitemi{%
196   \ifx\insideitemizei\inside%
197     \let\altlastitem1%
198     \expandafter\item%
199   \else%
200     \begin{itemize}%
201     \let\insideitemizei\inside%
202     \let\altlastitem1%
203     \makeenteractive%
204     \expandafter\item%
205   \fi
206 }
207
208 \def\newitemii{
209   \ifx\insideitemizeii\inside
210     \expandafter\item%
211   \else
212     \begin{itemize}
213       \let\insideitemizeii\inside
214       \expandafter\item%
215   \fi
216 }
```

Ok, the following code is stolen from the shortvrb package, and I don't understand anything of it. But I keep on trying... nevertheless, it's working fine, as far as I can see.

\makeitemi With this macro, you can define the character you want to use for first-level
\makeitemii itemize. (Guess the sense of \makeitemii...) Default ist • for first-level and ‣ for second-level. Maybe this will be extended till fourth level. More doesn't seem to make any sense.

```
217 %
218 \makeatletter
219 \def\makeitemi#1{%
220   \expandafter\ifx\csname cc\string#1\endcsname\relax
221     \add@special{#1}%
222     \expandafter
```

11

```
223    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
224    \begingroup
225      \catcode`\~\active  \lccode`\~`#1%
226      \lowercase{%
227      \global\expandafter\let
228        \csname ac\string#1\endcsname~%
229      \expandafter\gdef\expandafter~\expandafter{\newitemi}}%
230    \endgroup
231    \global\catcode`#1\active
232  \else
233  \fi
234 }
235
236 \def\makeitemii#1{%
237   \expandafter\ifx\csname cc\string#1\endcsname\relax
238     \add@special{#1}%
239     \expandafter
240     \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
241     \begingroup
242       \catcode`\~\active  \lccode`\~`#1%
243       \lowercase{%
244       \global\expandafter\let
245         \csname ac\string#1\endcsname~%
246       \expandafter\gdef\expandafter~\expandafter{\newitemii}}%
247     \endgroup
248     \global\catcode`#1\active
249   \else
250   \fi
251 }
```

Now there are the two helperfunctions – no guess what they are really doing.

```
252 \def\add@special#1{%
253   \rem@special{#1}%
254   \expandafter\gdef\expandafter\dospecials\expandafter
255 {\dospecials \do #1}%
256   \expandafter\gdef\expandafter\@sanitize\expandafter
257 {\@sanitize \@makeother #1}}
258 \def\rem@special#1{%
259   \def\do##1{%
260     \ifnum`#1=`##1 \else \noexpand\do\noexpand##1\fi}%
261   \xdef\dospecials{\dospecials}%
262   \begingroup
263     \def\@makeother##1{%
264       \ifnum`#1=`##1 \else \noexpand\@makeother\noexpand##1\fi}%
265     \xdef\@sanitize{\@sanitize}%
266   \endgroup}
267 \makeatother
```

5.2 enumerate with a single character

¹, ² And we do just the same stuff with `\enumerate`. But here we take the character ¹ as first level item, the ²[5] as second level etc. This may be confusing some way, but just try it.

For the implementation: copy-pasted the code above, nothing interesting so far.

```
268 \def\altendenum{
269   \if\altlastitem 1%
270     \let\altlastitem0%
271   \else%
272     \end{enumerate}%
273     \let\insideenumi\outside%
274   \fi%
275 }
276
277 \begingroup
278   \lccode`\~=`\^^M%
279 \lowercase{%
280   \endgroup
281   \def\makeenteractiveenum{%
282     \catcode`\^^M=\active
283     \let~\altendenum
284 }%
285 }
286
287 \def\newenumi{%
288   \ifx\insideenumi\inside%
289     \let\altlastitem1%
290     \expandafter\item%
291   \else%
292     \begin{enumerate}%
293     \let\insideenumi\inside%
294     \let\altlastitem1%
295     \makeenteractiveenum%
296     \expandafter\item%
297   \fi
298 }
299
300 \def\newenumii{
301   \ifx\insideenumii\inside
302     \expandafter\item%
303   \else
304     \begin{enumerate}
305       \let\insideenumii\inside
306       \expandafter\item%
```

---

[5]Maybe this is a very stupid idea, because now the ² cannot be used as a square in mathmode. Of course there could be a test `ifmmode`, but I rather would like to find a better character for `enumerate`.

```
307   \fi
308 }
```

We use the same methods as above, still not understanding, what they are doing.
Just changing two lines of code and hoping, everything will be fine.

```
309 \makeatletter
310 \def\makeenumi#1{%
311   \expandafter\ifx\csname cc\string#1\endcsname\relax
312     \add@special{#1}%
313     \expandafter
314     \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
315     \begingroup
316       \catcode`\~\active  \lccode`\~`#1%
317       \lowercase{%
318       \global\expandafter\let
319         \csname ac\string#1\endcsname~%
320       \expandafter\gdef\expandafter~\expandafter{\newenumi}}%
321     \endgroup
322     \global\catcode`#1\active
323   \else
324   \fi
325 }
326
327 \def\makeenumii#1{%
328   \expandafter\ifx\csname cc\string#1\endcsname\relax
329     \add@special{#1}%
330     \expandafter
331     \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
332     \begingroup
333       \catcode`\~\active  \lccode`\~`#1%
334       \lowercase{%
335       \global\expandafter\let
336         \csname ac\string#1\endcsname~%
337       \expandafter\gdef\expandafter~\expandafter{\newenumii}}%
338     \endgroup
339     \global\catcode`#1\active
340   \else
341   \fi
342 }
343 \makeatother
344
```

Finally, we set the default characters for the items and enumerations:

```
345 \makeitemi•
346 \makeitemii▸
347 \makeenumi¹
348 \makeenumii²
```

And that's it.

<div align="center">

Happy altTEXing!

</div>

A very short introduction to XƎLATEX

Everything you have to know about XƎLATEX to use this package: Write your LATEX file just as you are used to. But save it as utf8-encoded, and say

`\usepackage{xltxra}`

instead of

`\usepackage[latin1]{inputenc} and \usepackage[T1]{fontenc}`

This loads some files that provide all the cool stuff XƎLATEX offers. You don't have to take care of letters TEX would not understand – XƎTEX understands every character you type. But sometimes the font may not have the symbol for this – then you can use `\fontspec{fontname}`, where `fontname` is the name of a font on your system, e.g. `Arno Pro`, `Linux Libertine`, `LT Zapfino One` etc.

Then, you compile your document with the command `xelatex file.tex`, instead of `xelatex file.tex` and you get a pdf as output. Nevertheless, XƎTEX is not an pdfTEX successor, so you cannot use microtypographic extensions.

If you have any trouble using XƎLATEX, just e-mail me!

todo

Here a section with some ideas that could be implemented.

- Use ² as square in mathmode and possibly ¹ as `\footnote`?

- Do something to enable easy tabular

- If there is only one char after an _, there should no space be needed.

- Maybe there could be a ConTeXt-version of this file.