

SciTE

Introduction

The SciTE editor is now some 15 years old, and still one of the nicest around. This editor is a wrapper around the scintilla editor framework. It is available for free for Windows and Linux, and there is a relatively cheap version for MacOSX.

Here are a few reasons why I prefer using this editor:

- The footprint is small and (at least on Windows and Linux) installation and updating is easy.
- The editor starts up fast, performs well and the fonts have always been rendered very well.
- Configuration is easy and flexible, and changes to configuration files are reflected immediately (no restart required).
- Lots of file formats are supported by syntax highlighting.
- Processing of files is logged in a fast and efficient log pane.
- There is a Lua interface built-in that permits you to write extensions.

Syntax highlighting

One of the first things that I did when I started using SciTE is to (re)write the \TeX and MetaPost lexers so that they were more suitable for Con \TeX t. For quite some years Con \TeX t has shipped with the relevant files for editing and processing \TeX .

Recently the external lexer feature has been extended by an Lpeg based lexer. As a consequence, I wrote a couple of lexers that go beyond the older ones. These are tuned for Con \TeX t and are shipped with the Con \TeX t distribution.

- The \TeX lexer can not only distinguish between tex primitives (including conditionals), low level Con \TeX t commands, special registers, Con \TeX t user interface commands and special symbols, but can also deal with nested Lua and MetaPost code.
- The Lua lexer has a variant that can recognize some of the Con \TeX t MkIV features.
- The xml lexer can recognize some errors in the syntax.
- The Pdf lexer can recognize the relevant objects to some extent.
- The MetaPost lexer can distinguish primitives, MetaFun and user defined commands.

The Con \TeX t distribution ships with all the files needed for getting this up and running. In addition to initializing lexers, we also tune some of the menus for use with \TeX based workflows. The background of the text areas is set to a light shade of gray and the font defaults to DejaVu Mono; which happens to cover lots of Unicode characters.

The \TeX , xml and the yet unmentioned text lexers can do realtime spell checking. As with the lexers, spell checking is more advanced in the Lpeg lexers than in the traditional ones: we recognize rightly spelled words, mark unknown words and also mark words that need case checking. The nice thing is that the regular command highlighting works in parallel. This is shown in figure 1. Normally I use the full (high-res and wide) screen which gives enough room for regular documents as well as the (real-time) log pane. Menus and configured tools adapt themselves automatically to the current file type.

```

1 % language=uk
2 \defineformattext
3 [entry]
4 \starttext
5
6 \startchapter[title=Some fancy title]
7
8 \startluacode
9
10 local entries = { -- there can be more
11 { text = "The third entry!" },
12 { text = "The fourth entry!" },
13 }
14
15 for i=1,#entries do
16 context.startentry()
17 context.entries[i].text
18 context.stopentry()
19 end
20 \stopluacode
21
22 This is just some text to demonstrate the realtime spellchecker
23 in combination with the embedded lua and metapost lexers and
24 inline as well as display \ctlua{context("lua code")}.
25
26 \startlinecorrection
27 \startMPcode
28 for i=1 upto 100 :
29 draw fullcircle scaled i mm ;
30 endfor ;
31 \stopMPcode
32 \stoplinecorrection
33
34 \iftrue
35 \def\crap{some text} % who cares
36 \else
37 \def\crap{some crap} % about this
38 \fi
39
40 \blank[2*big]
41
42 \crap
43
44 \stopchapter
45
46 \stoptext
47

```

```

mtx-run --autogenerate --script context --autopdf scite-context-visual.tex
mtx-context | run 1: luatex --fmt=C:/data/develop/tex-context/tex/texmf-
This is LuaTeX, Version beta=0.71.0-20110801 (rev 4015)
\write18 enabled.
[scite-context-visual.tex
scite-context-visual.tex
ConteKt ver: 2011.11.08 19:35 MKIV fmt: 2011.11.8 int: english/english
system > cont-new.mkiv loaded
C:/data/develop/context/sources/cont-new.mkiv
system > beware: some patches loaded from cont-new.mkiv
system > cont-loc.mkiv loaded
C:/data/develop/context/sources/cont-loc.mkiv
system > beware: some patches loaded from cont-loc.mkiv
system > cont-exp.mkiv loaded
C:/data/develop/context/sources/cont-exp.mkiv
system > beware: some patches loaded from cont-exp.mkiv
system > scite-context-visual.top loaded
[scite-context-visual.top]
fonts > latin modern fonts are not preloaded
[language] > language en is active
C:/data/develop/tex-context/texmf-context/fonts/map/pdf/tex-context/texmf-
fonts > preloading latin modern fonts (second stage)
C:/data/develop/context/sources/type/size.mkiv C:/data/develop/context/sour
fonts > virtual math > unable to resolve name mapsfrownchar
fonts > fallback modern rm ligat is loaded
structure > sectioning > chapter @ level 2 : 0.1 -> Some fancy title
metapost > initializing instance 'metafun' using format 'metafun'
metapost > loading 'metafun': C:/data/develop/context/metapost/context
backend > xmp > using file 'C:/data/develop/context/sources/lpdf.pdx'.
> flushing resplage lu, uspage 1, subpage 1
C:/data/develop/tex-context/tex/texmf/fonts/opentype/public/ln/loroman12.r
Mkiv lua stats > used config file C:/data/develop/tex-context/tex
Mkiv lua stats > used cache path C:/data/develop/tex-context/tex
Mkiv lua stats > resource resolver loadtime: 0.016 seconds, 1 scame
Mkiv lua stats > stored bytecode data 392 modules, 63 tables, 365 chu
Mkiv lua stats > cleaned up reserved nodes - 39 nodes, 9 lists of 427
Mkiv lua stats > mode memory usage 2 blue, 2 penalty, 10 attribute
Mkiv lua stats > mode list callback tasks 6 unique task lists, 5 instance
Mkiv lua stats > used backends pdf (backend for directly gener
Mkiv lua stats > loaded patterns emi:2
Mkiv lua stats > callbacks 2880 direct, 3573 indirect, 638
Mkiv lua stats > randomizer resumed with value 0.6023320340
Mkiv lua stats > link preparation time 0.909 seconds, 0 nodes, 15 lpat
Mkiv lua stats > result saved in file scite-context-visual.pdf
Mkiv lua stats > loaded fonts 25 files: steary10.afm lmonno12
Mkiv lua stats > fonts load time 0.361 seconds
Mkiv lua stats > metapost processing time 0.016 seconds, loading: 0.078 s
Mkiv lua stats > fonts load time 0.361 seconds
Mkiv lua stats > luatex banner this is luatex, version beta=0-
Mkiv lua stats > control sequences 31366 of 65536 = 100000
Mkiv lua stats > current memory usage 24 MB (ctex: 35 MB)
Mkiv lua stats > runtime 1.188 seconds, 1 processed page
mtx-context | pdfview methods: acrobat default okular, current method: ac
system > total runtime: 2.264Exit code: 0

```

Figure 1. Nested lexers in action.

A side note: currently the MacOSX version does not ship with the library needed for the lpeg lexer. Hopefully this will change one day.

Functionality

Of course, SciTE provides all the regular editing functionality; although in practice one will use only a small subset of features. A real nice feature is the rectangular selection, cut and paste. It's really easy to move columns around. Spacing can be visualized. Structures can be folded.

As an example of a Lua extensions I wrote a word wrapper, simply because I want document sources to look nice as well.

Many files can be open at the same time and are accessed using tabs. The state is remembered and restored at a next startup. Quite handy is the fact that one can collapse all start-ups into one instance.

Word completion is supported (which is quite handy) as is expansion of abbreviation (although I never used that). Also nice is the ability to comment a line or a selected block of text with one key.

Of course, once you are fluent with an editor it stays your favourite no matter what others tell you, but for me it's the only editor that I want to work with.

Side note: a nice alternative is textadept, an editor written in Lua using the same scintilla editing component. It shares the Lpeg lexing code but it lacks a realtime log pane and has no tabs.

Processing

Users can easily tweak the .properties files to define commands that can be applied to a file. One can add runners to menus and associate them with keys. For T_EX compiling, linking et cetera this makes no sense, but checking, processing and

previewing does. For ConT_EXt we use (mtx-)check for checking, (mtx-)context for processing T_EX and xml files, mtxrun for running Lua, et cetera.

Of course, you can use SciTE for any macro package you like. If you don't use ConT_EXt, it provides support for T_EX anyway. It's an easy to install editor, so if you're looking for something new it's worth a try.

Hans Hagen