The tikzmake package*

Christoph Reller
creller@ee.ethz.ch

July 4, 2013

Abstract

Make use of the `list and make` mode of the Ti*k*Z externalization.

1 Introduction

The Ti*k*Z externalization library (see `pgfmanual.pdf`) provides automatic pre-compilation of `tikzpicture` environments. When the LaTeX document is compiled, the externalization library replaces a `tikzpicture` environment by an `\includegraphics` command that inserts the pre-compiled picture provided it exists and otherwise re-compiles it. This can result in much quicker compilation when doing small changes to the LaTeX document.

There is, however, a problem with detecting when a `tikzpicture` environment has changed and needs to be rebuilt. The idea is to use a separate source file for each Ti*k*Z picture and to use the `list and make` mode of the Ti*k*Z externalization library. In this mode, Ti*k*Z produces a makefile describing dependencies that have been declared with `\tikzpicturedependsonfile`. The purpose of the `tikzmake` package is to automatically make use of this makefile and to simplify the externalization in `list and make` mode.

2 Usage

The `tikzmake` package consists of a makefile and a LaTeX package, both of which cooperate, but can be used independently.

2.1 Automatic Build with a GNU Makefile

The provided GNU makefile (`tikzmake`) is able to compile "any" LaTeX document[1]. The procedure to use it is simple:

*This document corresponds to tikzmake v0.2?, dated 2012/12/19.

[1]Note that there may be more elaborate and general makefiles for LaTeX around, e.g. "The" LaTeX Makefile http://code.google.com/p/latex-makefile/

1. Copy `tikzmake` into the directory where the main LaTeX source file is located and rename `tikzmake` to `Makefile`.
2. In a shell, navigate to this directory and type `make`.

The makefile has some basic support for:

- multiple LaTeX source files (no recursive checks for inclusions)
- detecting if `pdflatex` or `latex` should be used
- running `bibtex` for bibliography generation if needed
- running `makeindex` for index generation and nomenclatures if needed
- detecting additional needed `(pdf)latex` runs for missing labels and missing citations
- the Ti*k*Z externalization library in `list and make` mode

The makefile defines the following targets:

`all:` builds everything, deletes nothing

`clean_intermediate:` deletes all output files except the final pdf file(s) and all pre-compiled Ti*k*Z picture environments.

`clean_dep:` deletes all additionally generated makefiles

`clean:` deletes all output files except the resulting pdf file(s)

`clean_all:` deletes all output files

2.2 Easy Externalization

2.2.1 Preliminaries

In the following I explain by example, how to use the externalization mechanism. Note that various directory and file names can be customized via options (see Section 3). Also note that for the externalization to work you have to call `(pdf)latex` with the option `-shell-escape`.

Let us assume that you are working on a document that contains `tikzpicture` environments, some of which you would like to pre-compile, i.e., externalize. All pre-compiled pdf files will reside in a subdirectory `tikzext`. You have to create this directory, unless you are using the `tikzmake` makefile. You load the package in the preamble with:

```
\usepackage{tikzmake}
```

This in itself has no effect, since the `tikzmake` package does not enable pre-compilation per default for all pictures.

2.2.2 Pictures

`\inputtikz` Assume that in the text body, you have:

⟨*before*⟩
`\begin{tikzpicture}`
  ⟨*tikzpicture-contents*⟩
`\end{tikzpicture}`
⟨*after*⟩

To enable externalization for this `tikzpicture`, you substitute the above by:

⟨*before*⟩
`\inputtikz{`⟨*pict-filename*⟩`}`
⟨*after*⟩

and in the subdirectory `picts` (create this directory if it does not yet exist) you create a new file ⟨*pict-filename*⟩`.tex` that contains:

`\begin{tikzpicture}`
  ⟨*tikzpicture-contents*⟩
`\end{tikzpicture}`

In principle, the above works for any `tikzpicture` environment, also for plots done with `pgfplots`. For the case where plots depend on an external data file, we provide an additional macro.

2.2.3 Plots

`\inputplot` Assume that in the text body, you have the following plot:

⟨*before*⟩
`\pgfplotstableread{../plotdat/`⟨*plot-filename*⟩`.dat}\plotdata`
`\begin{tikzpicture}`
  ⟨*plot-contents*⟩
`\end{tikzpicture}`
⟨*after*⟩

Note that the above uses some datafile in a subdirectory `plotdat` and loads this data into a macro `\plotdata`.[2] To enable externalization for this plot, you substitute the above by:

⟨*before*⟩
`\inputplot{`⟨*plot-filename*⟩`}`
⟨*after*⟩

and in the subdirectory `plots` (create this directory if it does not yet exist) you create a new file ⟨*plot-filename*⟩`.tex` that contains:

`\begin{tikzpicture}`
  ⟨*plot-contents*⟩
`\end{tikzpicture}`

[2] See `pgfplots.pdf` for how to construct plots in this mode.

2.2.4 Work Cycle Without `tikzmake` Makefile

With the externalization functionality provided by `\inputtikz` and `\inputplot`, pre-compiled Ti*k*Z pictures are not updated automatically when the source file has changed. In `list and make` mode, the Ti*k*Z externalization library produces, however, a makefile ⟨*maintex*⟩`.makefile`, where ⟨*maintex*⟩ is the name of the main LaTeX file without the `.tex` extension. With the help of this makefile, typical work cycle scenarios are:

Externalize an existing Ti*k*Z picture: Proceed as in Section 2.2.2 or 2.2.3 and LaTeX as:

```
pdflatex -shell-escape ⟨maintex⟩.tex
```

or:

```
latex -shell-escape ⟨maintex⟩.tex
```

Change an externalized Ti*k*Z picture: Use the makefile ⟨*maintex*⟩`.makefile` to recompile the changed source files:

```
make -f ⟨maintex⟩.makefile
```

Remove an externalized Ti*k*Z picture: Remove the source file ⟨*filename*⟩`.tex` and the pre-compilation output, i.e., all the files `tikzext/`⟨*filename*⟩`.*`. Make a normal `(pdf)latex` run.

2.3 Externalization and Makefile

The `tikzmake` makefile as described in Section 2.1 and the `tikzmake` LaTeX package as described in Section 2.2 work hand in hand. Specifically, all the work cycle scenarios in Section 2.2.4 are substituted by typing `make` on the shell. This includes the tracking of changes in externalized Ti*k*Z pictures and automatic creation of directories.

3 Package Options

The package uses the ⟨*key*⟩=⟨*value*⟩ paradigm for options. All options are string options.

⟨*key*⟩	*default*	*meaning*
`extdir`	`tikzext`	Directory containing the pre-compiled output files for Ti*k*Z pictures externalized both with `\inputtikz` and `\inputplot`.
`pictdir`	`picts`	Directory containing the source files for Ti*k*Z picture environments externalized with `\inputtikz`.
`pictprefix`		Prefix prepended to filenames given to `\inputtikz` to construct the final LaTeX source filename.
`plotdir`	`plots`	Directory containing the source files for plots externalized with `\inputplot`.
`plotprefix`		Prefix prepended to filenames given to `\inputplot` to construct the final LaTeX source filename.
`datadir`	`plotdat`	Directory containing the data files for plots externalized with `\inputplot`.
`dataprefix`		Prefix prepended to filenames given to `\inputplot` to construct the final filename (without filename extension) of data files.
`dataext`	`dat`	Filename extension appended to filenames given to `\inputplot` to construct the final filename of data files.

Summary

`\inputtikz{`⟨*name*⟩`}` implies the following file names:

		default:
Source file	⟨*pictdir*⟩`/`⟨*pictprefix*⟩⟨*name*⟩`.tex`	`picts/`⟨*name*⟩`.tex`
Output file	⟨*extdir*⟩`/`⟨*pictprefix*⟩⟨*name*⟩`.pdf`	`tikzext/`⟨*name*⟩`.pdf`

`\inputplot{`⟨*name*⟩`}` implies the following file names:

		default:
Source file	⟨*plotdir*⟩`/`⟨*plotprefix*⟩⟨*name*⟩`.tex`	`plots/`⟨*name*⟩`.tex`
Data file	⟨*datadir*⟩`/`⟨*dataprefix*⟩⟨*name*⟩`.`⟨*dataext*⟩	`plotdat/`⟨*name*⟩`.dat`
Output file	⟨*extdir*⟩`/`⟨*plotprefix*⟩⟨*name*⟩`.pdf`	`tikzext/`⟨*name*⟩`.pdf`

4 Implementation

Declare the options:

```
\DeclareStringOption[tikzext]{extdir}
\DeclareStringOption[picts]{pictdir}
\DeclareStringOption[]{pictprefix}
\DeclareStringOption[plots]{plotdir}
\DeclareStringOption[]{plotprefix}
\DeclareStringOption[plotdat]{datadir}
\DeclareStringOption[]{dataprefix}
\DeclareStringOption[dat]{dataext}
\ProcessKeyvalOptions*\relax
```

Load the Ti*k*Z externalization library, choose the `list and make` mode and disable externalization per default:

```
\usetikzlibrary{external}
\tikzexternalize[prefix=\tikzmake@extdir/,mode=list and make,export=false]
```

Use different external system calls in pdf and dvi mode as suggested in the `pgfmanual.pdf`:

```
\ifpdf
\tikzset{external/system call={pdflatex \tikzexternalcheckshellescape
    -halt-on-error -interaction=batchmode -jobname '\image' '\texsource'}}
\else
\tikzset{external/system call={latex \tikzexternalcheckshellescape
    -halt-on-error -interaction=batchmode -jobname '\image' '\texsource';
    dvips -o '\image'.ps '\image.dvi'}}
\fi
```

Provide the `\inputtikz` command:

```
\providecommand{\inputtikz}[1]{%
\tikzset{/tikz/external/export=true}%
\tikzsetnextfilename{\tikzmake@pictprefix#1}%
\tikzpicturedependsonfile{\tikzmake@pictdir/\tikzmake@pictprefix#1.tex}%
\tikzsetnextfilename{\tikzmake@pictprefix#1}%
\input{\tikzmake@pictdir/\tikzmake@pictprefix#1.tex}%
\tikzset{/tikz/external/export=false}}
```

Provide the `\inputplot` command:

```
\providecommand{\inputplot}[1]{%
\tikzset{/tikz/external/export=true}%
\tikzsetnextfilename{\tikzmake@plotprefix#1}%
\tikzpicturedependsonfile{\tikzmake@plotdir/\tikzmake@plotprefix#1.tex
  \tikzmake@datadir/\tikzmake@dataprefix#1.\tikzmake@dataext}%
\tikzsetnextfilename{\tikzmake@plotprefix#1}%
\tikzifexternalizing{%
  \pgfplotstableread{\tikzmake@datadir/\tikzmake@dataprefix#1.\tikzmake@dataext}%
  \plotdata}{}
\input{\tikzmake@plotdir/\tikzmake@plotprefix#1.tex}%
\tikzset{/tikz/external/export=false}}
```

Change History

Index