The tikzsymbols package*

Ben Vitecek
b.vitecek@gmx.at

May 14, 2017

Abstract

Some symbols created using tikz.
For differences between the releases see section 2.
English is (still) not my native language so there (still) might be some errors[1]. .

Contents

1 Introduction **2**

2 Important changes **3**

3 Options **3**

3.1 Load-time Options . . . 3
3.1.1 marvosym (true/false) . . . 3
3.1.2 usebox (true/false) . . . 4
3.1.3 prefix (<string>) . . . 4
3.2 Further (Change!) Options . . . 5
3.2.1 final (true/false) . . . 5
3.2.2 draft (true/false) . . . 5
3.2.3 tree (true/false/on/off) . . . 6
3.2.4 after-symbol (<string or command>) . . . 6
3.2.5 global-scale (<number>)
symbol-scale (<symbol=number,...>) . . . 6
3.2.6 baseline (true/false) . . . 6

*This document corresponds to tikzsymbols v4.04, dated 2018/08/08.
[1]They are – of course – on purpose.

4 Symbols **7**

4.1 cooking-symbols 9

4.2 Emoticons 11

4.2.1 “normal” Emoticons 11

4.2.2 “3D” Emoticons 15

4.3 other Symbols 17

4.4 Trees 19

5 Known errors & Problems **20**

6 Nobody is perfect **21**

7 Danksagung **21**

8 Changes **21**

1 Introduction

As far as I can remember this package is the result of me writing a cooking book[2]. Back then I wasn't able to find the cooking symbols I wanted and using time, tikz, lot's of magic (also known as “programming”, but only if the respective person knows what's going on) and a documentation in bad grammar[3] I somehow ended up with this package.

During time LaTeX3 became known to me and I started experimenting and programming in this (I would say due to its simplicity compared to LaTeX 2_ε far superior) language. Well, long story short: I was impressed. And so the idea of writing my package in LaTeX3 was born.

I finally took my time and started rewriting my code using LaTeX3. This process can be summarized as: “What *does* this command?”, “Why did I define *this* command?” and more generally “ *What* have I done?!” Well, let's hope my code (and grammar) is better this time[4].

Well . . . thats it, have fun!

[2] Well, it's one result, the other one is a cooking book.
[3] Not that it' now any better.
[4] Looking at own risk. You have been warned.

2 Important changes

The packages should behave the same way as the "old" (LaTeX 2_ε) release.

The option `draft=absolute` is now obsolete and replaced by the much simpler option `draft=true`.

Furthermore the horribly named command `\tikzsymbolsaftersymbolinput` is not defined anymore by this package. Please use the new option `after-symbol`, in combination with the new command `\tikzsymbolsset`, see section 3 for more information.

3 Options

Options can either be set as package options or using `\tikzsymbolsset`. Some options can only be set as package options, those are described in section 3.1.

It is recommended to use the option `draft=true` while working on the document.

`\tikzsymbolsset`

`\tikzsymbolsset {⟨keys = values⟩}`

Most keys, except for the load-time options (section 3.1), can be set using this command.

3.1 Load-time Options

The following options *cannot* be set using `\tikzsymbolsset`.

3.1.1 marvosym (true/false)

`marvosym = true / false`

Please load tikzsymbols *after* marvosym.

marvosym also defines `\Smiley` and `\Coffeecup`. If you prefer those symbols (☺, ☕) over the tikzsymbols ones (□, □) you can use this option. If set to true tikzsymbols cancels the definition of its `\Smiley` and `\Coffeecup`:

Without option "marvosym": □ □	With option "marvosym": ☺ ☕
`\usepackage{marvosym}` `\usepackage{tikzsymbols}`	`\usepackage{marvosym}` `\usepackage[marvosym]{tikzsymbols}`

This option raises an error if set `true` without loading package marvosym.

Can only be set as load-time option.

You may also use the option `prefix` (section 3.1.3).

3.1.2 usebox (true/false)

In tikzsymbols all symbols are stored inside boxes (`\sbox`) and while I still have no idea what exactly happens, it shortens the compilation time of the document. By default this option is `true`.

The drawback is that LaTeX has only a limited number of box registers. If you come across an error message regarding boxes try setting `usebox=false`.

Can only be set as load-time option.

3.1.3 prefix (<string>)

This option takes a string as value: `prefix=<string>` and adds this prefix to every command defined by this package. So setting `prefix=<prefix>` adds `<prefix>` to all commands of this package: `\<prefix>command`.

`<prefix>` should neither contain any special characters (e.g., ä, ü, ß, etc.) nor spaces.

By default it is empty, so no prefix is given, if this option is given without an argument `<prefix>` is set to `tikzsymbols`.

Can only be set as a load-time option.

For example:

```
\usepackage[prefix=tikzsym]{tikzsymbols}
```

defines `\Smiley` as `\tikzsymSmiley`, `\Kochtopf` as `\tikzsymKochtopf`, `\pot` as `\tikzsympot`, etc.

If you use this option or think about using this option the following command may be handy:

`\tikzsymbolsuse` `\tikzsymbolsuse{⟨Symbolname⟩}`

This command takes the name of the symbol *without* backslash and prints the symbol (or raises an error if the symbol is not defined). Using this command you don't have to worry about a `<prefix>`, just write the command name and this command adds automatically the given prefix to the command name.

For example: `\tikzsymbolsuse{Smiley}[2]`

`\tikzsymbolsuse{BasicTree}[1.2]{black}{red!50!black}{red}{leaf}`

`\tikzsymbolsuse{Ofen}`

`\tikzsymbolsuse{Fire}[-1.3]`

etc.

3.2 Further (Change!) Options

Most of these commands can be set either as package option or with `\tikzsymbolsset`.

3.2.1 final (true/false)

`final` `final= <true/false>`

This key has the opposite behavior of the option `draft`.

It is a boolean key and therefore accepts only `true` or `false` and is set to `true` by default. Setting it to `true` prints all symbols normally. Setting it to `false` prints plain vanilla draft-boxes instead which speeds up the compile-process.

3.2.2 draft (true/false)

`draft` `draft = <true/false>`

While working on the document it is recommended to set this option to `true` because creating many symbols may takes some time to compile and by setting this option to `true` the symbols are replaced by plain vanilla rectangles which are faster to create.

The old option `draft=absolute` is obsolete and should therefore not be used.

3.2.3 tree (true/false/on/off)

tree `tree= <true/on/false/off>`

This key accepts `true`, `false` and furthermore `on` and `off`. The latter do exactly the same as the first ones.

This option has only an effect on the command `\BasicTree` and his derivates (`\Springtree`, `\Summertree`, `\Autumntree` and `\Wintertree`) and substitutes them with tikz drawn boxes.

So while `draft=true` replaces the output of *all* commands with simple black boxes, `tree=true/on` only replaces the output of "tree"-commands with boxes.

It is recommended to use `draft=true`, but if you want you can use this option.

3.2.4 after-symbol (`<string or command>`)

after-symbol `after-symbol = {⟨string or command⟩}`

Is more stable if set using `\tikzsymbolsset`. The value of this key is inserted after every command of this package. By default it is set to `\xspace`.

3.2.5 global-scale (`<number>`) symbol-scale (`<symbol=number,...>`)

global-scale `global-scale = {⟨number⟩}`
symbol-scale `symbol-scale = {⟨symbol=number,...⟩}`

`global-scale` can be used to scale *all* commands by `{⟨number⟩}`.

If only a specific symbol should be scaled globally you can use the second option.

Examples: If only `\Schneebesen` and `\Smiley` should be scaled you may use `\tikzsymbolsset{symbol-scale={ Schneebesen=2, Smiley= 1.5 }}`.

If a local scale is given (e.g. `\Smiley[2]`) with `global-scale=3` the resulting scaling will be $2\cdot = 6$.

3.2.6 baseline (true/false)

baseline `baseline = {⟨true/false⟩}`

This option mainly exists to let the commands of this package work inside todonotes `\todo` command. If `true` adds to each symbol of this package the tikz option `baseline=default`. If you do not want this, set this option to `false`. It is set to `true` by default.

4 Symbols

In this section the symbols are introduced. They $\square$ $_{\text{all}}$ $\square$ change $\square$ automatically $\square$ with $\square$ text-size $\square$.

4.1 cooking-symbols

The following table shows all available cooking-symbols and their respective commands. The first column shows the command-names (german & english), the second the optional parameter(s). The optional parameter(s) are for both the german and the english commands the same.

⟨*scale*⟩ can be a number between (not exactly) −1400 and (also not exactly) 1400, default is 1.

Da Umlaute nicht in Befehlsnamen vorkommen dürfen, werden die Umlaute ö, ä, ü durch o, a, u ersetzt.

German & English Commands		Optional parameter(s)	Output
`\Kochtopf`	`\pot`	[⟨*scale*⟩]	
`\Bratpfanne`	`\fryingpan`	[⟨*scale*⟩]	
`\Schneebesen`	`\eggbeater`	[⟨*scale*⟩]	
`\Sieb`	`\sieve`	[⟨*scale*⟩]	
`\Purierstab`	`\blender`	9 [⟨*scale*⟩]	
`\Dreizack`	`\trident`	[⟨*scale*⟩]	
`\Backblech`	`\bakingplate`	[⟨*scale*⟩]	

`\Kochtopf`
`\pot`
`\Bratpfanne`
`\fryingpan`
`\Schneebesen`
`\eggbeater`
`\Sieb`
`\sieve`
`\Purierstab`
`\blender`
`\Dreizack`
`\trident`
`\Backblech`
`\bakingplate`
`\Ofen`
`\oven`
`\Pfanne`
`\pan`
`\Herd`
`\cooker`
`\Saftpresse`
`\squeezer`
`\Schussel`
`\bowl`
`\Schaler`
`\peeler`
`\Reibe`
`\grater`
`\Flasche`
`\bottle`
`\Nudelholz`
`\rollingpin`

4.2 Emoticons

4.2.1 "normal" Emoticons

```
\Smiley
\Sadey
\Neutrey
\Changey
\cChangey
\Annoey
\Laughey
\Winkey
\oldWinkey
\Sey
\Xey
\Innocey
\wInnocey
\Cooley
\Tongey
\Nursey
\Vomey
\Walley
\rWalley
\Cat
\SchrodingersCat
\Ninja
\Sleepey
\NiceReapey
```

First column shows the commands, the second the (optional) parameter(s), the third the default-output (the only command with a mandatory argument is `\Changey`).

⟨*scale*⟩ can be a number between (not exactly) −2000 and (not exactly) 2000, default is 1.

⟨*color*⟩ can be every defined color. Note: The color names shouldn't contain special characters like ß, ä, ö, ...

`\Changey`'s ⟨*mood*⟩ has to be between −2 and 2 (1 equals `\Smiley`, −1 `\Sadey` and 0 `\Neutrey`).

Commands	(Optional) parameter(s)	Output
`\Smiley`	[⟨*scale*⟩] [⟨*color*⟩]	
`\Sadey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\Neutrey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\Changey`	[⟨*scale*⟩] [⟨*color*⟩] {⟨*mood*⟩}	
`\cChangey`	[⟨*scale*⟩] [⟨*color1*⟩] [⟨*color2*⟩] [⟨*color3*⟩] {⟨*mood*⟩}	
`\Annoey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\Laughey`	[⟨*scale*⟩] [⟨*color*⟩] [⟨*mouth color*⟩]	
`\Winkey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\oldWinkey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\Sey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\Xey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\Innocey`	[⟨*scale*⟩] [⟨*color*⟩] [⟨*halo color*⟩]	

Examples: `\Sadey[][red]`

`\Cooley[-3][cyan]`

`\Vomey[1.5][green!80!black][olive]`

`\Nursey[][yellow][blue][red]`.

`\Ninja[1.3][][violet][red]`.

`\colorbox{yellow}{\Winkey \Annoey[-1]\Neutrey}`

`\textcolor{blue}{\Sey}`

`\Sleepey[1][white][blue][yellow!95!black]`

`\SchrodingersCat{1}` `\SchrodingersCat{0}` `\SchrodingersCat{-1}`

`\Changey{-2}` `\Changey{-1.367}` `\Changey{-1}`

`\Changey{0}` `\Changey{1}` `\Changey{1.41}` `\Changey{2}`

`\cChangey{2}` `\cChangey{1}` `\cChangey{0.5}` `\cChangey{0.1}` `\cChangey{0}` `\cChangey{-0.5}` `\cChangey{-1}` `\cChangey{-2}`

`\cChangey[][][blue]{-1}` `\cChangey[][][blue]{0.5}`

If you intent to change the color of `\cChangey` you may define a new command so that you do not have to write those brackets each time.

4.2.2 "3D" Emoticons

\dSmiley
\dSadey
\dNeutrey
\dChangey
\dcChangey
\dAnnoey
\dLaughey
\dWinkey
\dSey
\dXey
\dInnocey
\dCooley
\dNinja
\drWalley
\dWalley
\dVomey
\dNursey
\dTongey
\dSleepey
\olddWinkey

First column shows the commands (note: the "3D" Emoticons begin with `\d...`), the second shows the (optional) parameter(s), the third shows the default-output (the only command with a mandatory argument is `\dChangey`).

⟨*scale*⟩ can be a number between a small number (under −500 for sure) and a large number (over 500 for sure), default is 1.

⟨*color*⟩ can be every defined color (see examples below). Note: The color names shouldn't contain special characters like ß, ä, ö, ...

`\Changey`'s ⟨*mood*⟩ has to be between −2 and 2 (1 equals `\dSmiley`, −1 `\dSadey` and 0 `\dNeutrey`).

Commands	Optional parameter(s)	Output
`\dSmiley`	[⟨*scale*⟩] [⟨*color*⟩]	
`\dSadey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\dNeutrey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\dChangey`	[⟨*scale*⟩] [⟨*color*⟩] {⟨*mood*⟩}	
`\dcChangey`	[⟨*scale*⟩] [⟨*color1*⟩] [⟨*color2*⟩] [⟨*color3*⟩] {⟨*mood*⟩}	
`\dLaughey`	[⟨*scale*⟩] [⟨*color*⟩] [⟨*mouth color*⟩]	
`\dAnnoey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\dWinkey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\olddWinkey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\dSey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\dXey`	[⟨*scale*⟩] [⟨*color*⟩]	
`\dInnocey`	[⟨*scale*⟩] [⟨*color*⟩] [⟨*halo color*⟩]	
`\dCooley`	[⟨*scale*⟩] [⟨*color*⟩]	
`\dTongey`	[⟨*scale*⟩] [⟨*color*⟩] [⟨*tongue color*⟩]	

Examples: `\dSadey[][red]`

`\dCooley[-3][cyan]`

`\dVomey[1.5][green!70!black][olive]`

`\dNursey[][yellow][blue][red]`.

`\dNinja[1.3][][violet][red]`.

`\dChangey{-2}` `\dChangey{-1.367}` `\dChangey{-1}` `\dChangey{0}` `\dChangey{1}` `\dChangey{1.41}` `\dChangey{2}`

`\dcChangey{2}` `\dcChangey{1}` `\dcChangey{0.5}` `\dcChangey{0.1}` `\dcChangey{0}` `\dcChangey{-0.5}` `\dcChangey{-1}` `\dcChangey{-2}`

`\dcChangey[][][blue]{-1}` `\dcChangey[][][blue]{0.5}`

If you intent to change the color of `\dcChangey` you may define a new command so that you do not have to write those brackets each time.

4.3 other Symbols

\Strichmaxerl
\Candle
\Fire
\Coffeecup
\Chair
\Bed
\Tribar
\Moai
\Snowman

`\Strichmaxerl`'s optional parameters 2–5 (⟨*left arm*⟩ to ⟨*right leg*⟩) can be a number between −360 and 360 (of course the number can be even greater or even smaller.). The parameters are the angles between the body and the separate parts of `\Strichmaxerl` (see examples).

⟨*scale*⟩ can be a very great and a very small negative number (but I don't think, that you need so large symbols).

⟨*color*⟩ can be every defined color. Note: The color names shouldn't contain special characters like ß, ä, ö, ...

Commands	Optional parameter(s)	Output
`\Strichmaxerl`	[⟨*scale*⟩] [⟨*left arm*⟩] [⟨*right arm*⟩] [⟨*left leg*⟩] [⟨*right leg*⟩]	
`\Candle`	[⟨*scale*⟩]	
`\Fire`	[⟨*scale*⟩]	
`\Coffeecup`	[⟨*scale*⟩]	
`\Chair`	[⟨*scale*⟩]	
`\Bed`	[⟨*scale*⟩]	
`\Moai`	[⟨*scale*⟩]	
`\Tribar`	[⟨*scale*⟩] [⟨*color 1*⟩] [⟨*color 2*⟩] [⟨*color 3*⟩]	
`\Snowman`	[⟨*scale*⟩]	

`\Tribar[-10][blue][red][green]`

`\Tribar[2.1][blue][blue!50][blue!20]`

`\Strichmaxerl[1][10][30][40][4]`,

`\Strichmaxerl[1.4][210][310][10][90]`,

`\Strichmaxerl[2][510][110][190][990]`,

`\Strichmaxerl[0.9][54][28][95][16]`

`\Strichmaxerl[][54][28]`

\BasicTree
\Springtree
\Summertree
\Wintertree
\WorstTree

4.4 Trees

⟨*scale*⟩ can be a number between (not exactly) −900 and (again not exactly) 900, default is 1.

⟨*color*⟩ can be every defined color (see examples below). Note: The color names shouldn't contain special characters like ß, ä, ö, ...

`{`⟨*leaf*⟩`}` uses the colors of `{`⟨*leaf color a*⟩`}` and `{`⟨*leaf color b*⟩`}`, you can leave this one empty if you don't want leaves (`\Wintertree` is without *leaf*, see examples below).

If you are using those trees, LaTeX needs longer to produce the output. So you may use the package option `tree=off`, or (better) `draft=true` (see section section 3.2.2 and section 3.2.3) to make LaTeX faster.

Furthermore this trees are pretty much stolen from the tikz manual.

Commands	Optional/Needed parameter(s)	Output
`\BasicTree`	`[`⟨*scale*⟩`]{`⟨*trunk color*⟩`}{`⟨*leaf color a*⟩`}{`⟨*leaf color b*⟩`}{`⟨*leaf*⟩`}`	see below
`\Springtree`	`[`⟨*scale*⟩`]`	
`\Summertree`	`[`⟨*scale*⟩`]`	
`\Autumntree`	`[`⟨*scale*⟩`]`	
`\Wintertree`	`[`⟨*scale*⟩`]`	
`\WorstTree`	`[`⟨*scale*⟩`]`	

`\BasicTree` examples Some "normal" trees:

```
\colorbox{green}{\BasicTree{red}{orange}{yellow}{leaf}}
\BasicTree[5]{orange!95!black}{orange!80!black}{orange!70!black}{leaf}
```

```
\BasicTree[2]{blue!65!white}{cyan!50!white}{cyan!50!white}{}
\BasicTree[-1.54]{green!20!black}{green!50!black}{green!70!black}{leaf}
```

```
\colorbox{black}{\BasicTree[3.75]{gray!80}{gray!50}{gray!40}{leaf}}
```

draftbox `\BasicTree` examples Some "draftbox" trees (using `tree=false`):

. . . and using the same trees with `tree=off/false` or `draft`(`=true`):

```
\colorbox{green}{\BasicTree{red}{orange}{yellow}{leaf}}
```

```
\BasicTree[5]{orange!95!black}{orange!80!black}{orange!70!black}{leaf}
```



```
\BasicTree[2]{blue!65!white}{cyan!50!white}{cyan!50!white}{}
\BasicTree[-1.54]{green!20!black}{green!50!black}{green!70!black}{leaf}
```

```
\colorbox{black}{\BasicTree[3.75]{gray!80}{gray!50}{gray!40}{leaf}}
```

I think it's better if you define your own trees using `\newcommand` and `\BasicTree`:

```
\newcommand{\Myicetree}[1][1]{%
  \BasicTree[#1]{blue!65!white}{cyan!50!white}{cyan!50!white}{}}
```

5 Known errors & Problems

marvosym

Make sure you load marvosym *before* tikzsymbols because both packages define `\Smiley`, marvosym via `\newcommand` tikzsymbols via `\DeclareDocumentCommand`.

If you load marvosym *after* tikzsymbols, LaTeX generates an error-message because `\Smiley` has already been defined.

If you load marvosym *before* tikzsymbols, tikzsymbols will overwrite marvosym's Smiley (and Coffeecup) and no error-message is generated (if you like the `\Smiley` from marvosym more, use the tikzsymbols option `marvosym` or `prefix`).

babel

If you encounter an error message like

```
Argument of \pgffor@next has an extra }
```

while using babel with e.g. language "francais" and for example `\Cooley` you may add

```
\usetikzlibrary{babel}
```

to your preamble. This should (hopefully) fix the problem.

6 Nobody is perfect

If you find a bug please send me a mail involving a *minimal example* showing the bug and a short description. Please mention "tikzsymbols" in the header, "gmx" has a habit of putting mails into the spam-folder and it helps me to recognize those mails faster. This can also be the reason why I may need some time to answer the mail.

Suggestions are also welcome.

7 Danksagung

I would like to thank all users for providing bug reports and helping to improve this package.

Furthermore many thanks to my brother helping me improving the symbols.

8 Changes

See the README.md file.