

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

errors¹.



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)	6
3.2.6	baseline (true/false)	6

*This document corresponds to `tikzsymbols` v4.04, dated 2018/08/08.

¹They 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 book². 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 grammar³ I somehow ended up with this package.

During time L^AT_EX3 became known to me and I started experimenting and programming in this (I would say due to its simplicity compared to L^AT_EX 2_ε far superior) language. Well, long story short: I was impressed. And so the idea of writing my package in L^AT_EX3 was born.

I finally took my time and started rewriting my code using L^AT_EX3. 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 time⁴.

Well ... thats it, have fun!

²Well, it’s one result, the other one is a cooking book.

³Not that it’ now any better.

⁴Looking at own risk. You have been warned.

2 Important changes

The packages should behave the same way as the “old” (L^AT_EX 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”: ☺ ☕
<code>\usepackage{tikzsymbols}</code>	<code>\usepackage{marvosym}</code> <code>\usepackage{tikzsymbols}</code>	<code>\usepackage{marvosym}</code> <code>\usepackage[marvosym]{tikzsymbols}</code>

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}
```

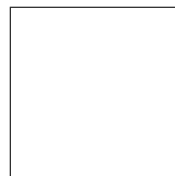
defines `\Smiley` as `\tikzsymSmiley`, `\Kochtopf` as `\tikzsymKochtopf`, `\pot` as `\tikzsympot`, etc.

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.

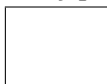
For example: `\tikzsymbolsuse{Smiley}[2]`



`\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 derivatives (`\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 `\tikzsymbolset`. 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)

global-scale global-scale = {\<number>}



This option can be used to scale *all* commands by {\<number>}. If a local scale is given (e.g. `\Smiley[2]`) and **global-scale=3** the resulting scaling will be $2 \cdot 3 = 6$.





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.

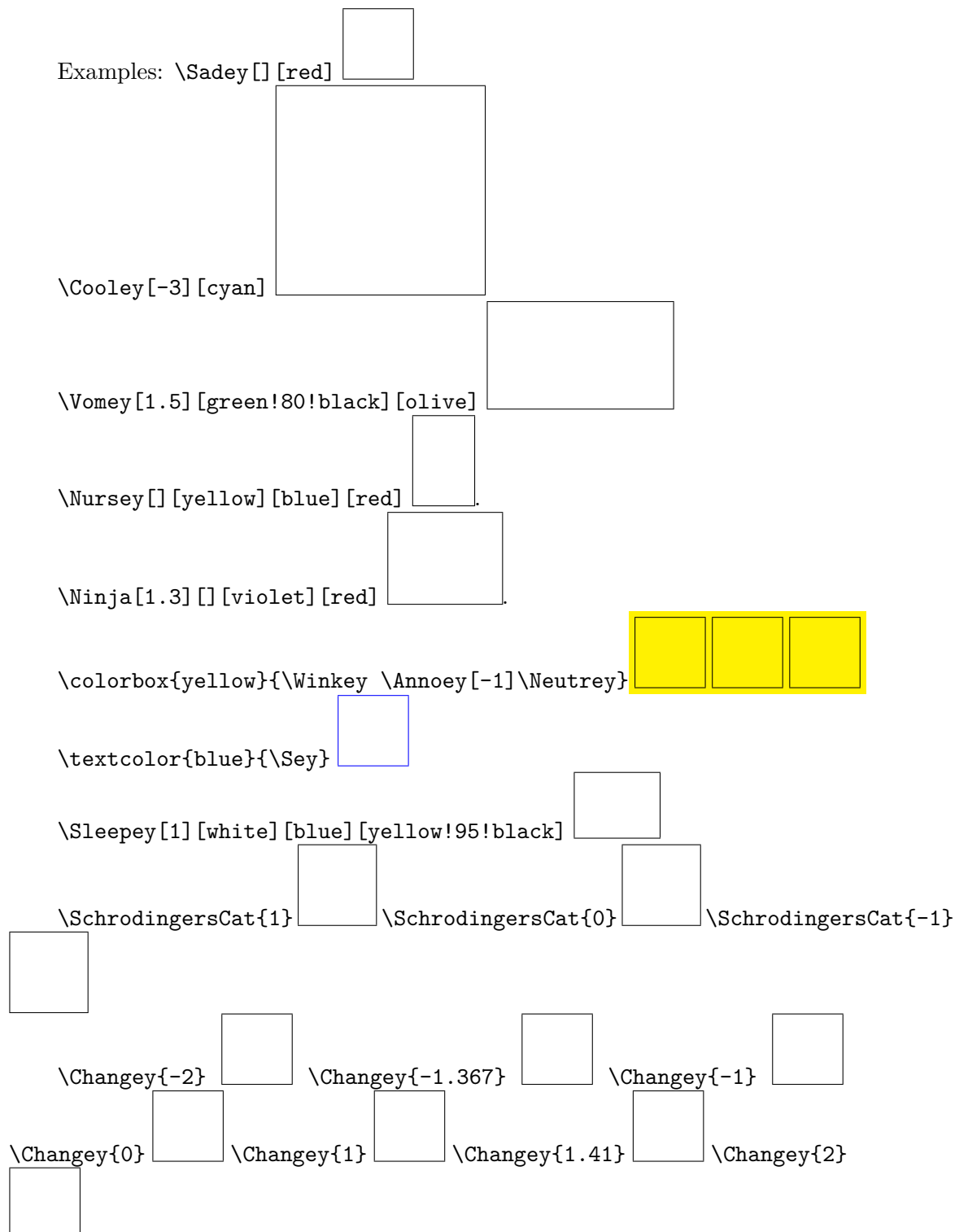
4 Symbols

In this section the symbols are introduced. They  all  change

 automatically  with  text-size .

<u>\Kochtopf</u>	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.		
<u>\pot</u>	$\langle scale \rangle$ can be a number between (not exactly) −1400 and (also not exactly) 1400, default is 1.		
<u>\Bratpfanne</u>	Da Umlaute nicht in Befehlsnamen vorkommen dürfen, werden die Umlaute ö, ä, ü durch o, a, u ersetzt.		
<u>\fryingpan</u>			
<u>\Schneebesen</u>	German & English Commands	Optional parameter(s)	Output
<u>\eggbeater</u>			
<u>\Sieb</u>			
<u>\sieve</u>			
<u>\Purierstab</u>			
<u>\blender</u>			
<u>\Dreizack</u>			
<u>\trident</u>			
<u>\Backblech</u>			
<u>\bakingplate</u>	\Kochtopf \pot	[$\langle scale \rangle$]	
<u>\Ofen</u>			
<u>\oven</u>			
<u>\Pfanne</u>	\Bratpfanne \fryingpan	[$\langle scale \rangle$]	
<u>\pan</u>			
<u>\Herd</u>			
<u>\cooker</u>			
<u>\Saftpresse</u>			
<u>\squeezer</u>			
<u>\Schussel</u>			
<u>\bowl</u>			
<u>\Schaler</u>			
<u>\peeler</u>			
<u>\Reibe</u>			
<u>\grater</u>			
<u>\Flasche</u>			
<u>\bottle</u>			
<u>\Nudelholz</u>			
<u>\rollingpin</u>			
	\Schneebesen \eggbeater	[$\langle scale \rangle$]	
	\Sieb \sieve	[$\langle scale \rangle$]	
	\Purierstab \blender	9 [$\langle scale \rangle$]	
	\Dreizack \trident	[$\langle scale \rangle$]	
	\Backblech \bakingplate	[$\langle scale \rangle$]	

[illegible]



$\backslash\text{cChangey}\{2\}$ $\backslash\text{cChangey}\{1\}$ $\backslash\text{cChangey}\{0.5\}$ $\backslash\text{cChangey}\{0.1\}$
 $\backslash\text{cChangey}\{0\}$ $\backslash\text{cChangey}\{-0.5\}$ $\backslash\text{cChangey}\{-1\}$
 $\backslash\text{cChangey}\{-2\}$
 $\backslash\text{cChangey}[] [] [\text{blue}]\{-1\}$ $\backslash\text{cChangey}[] [] [\text{blue}]\{0.5\}$

If you intent to change the color of $\backslash\text{cChangey}$ 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`
`\dSleepy`
`\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`).

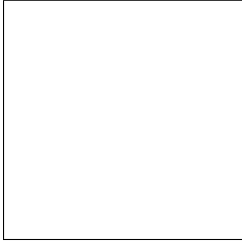
`<scale>` can be a number between a small number (under -500 for sure) and a large number (over 500 for sure), default is 1 .


`<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
<code>\dSmiley</code>	<code>[<scale>] [<color>]</code>	
<code>\dSadey</code>	<code>[<scale>] [<color>]</code>	
<code>\dNeutrey</code>	<code>[<scale>] [<color>]</code>	
<code>\dChangey</code>	<code>[<scale>] [<color>] {<mood>}</code>	
<code>\dcChangey</code>	<code>[<scale>] [<color1>] [<color2>] [<color3>] {<mood>}</code>	
<code>\dLaughey</code>	<code>[<scale>] [<color>] [<mouth color>]</code>	
<code>\dAnnoey</code>	<code>[<scale>] [<color>]</code>	
<code>\dWinkey</code>	<code>[<scale>] [<color>]</code>	
<code>\olddWinkey</code>	<code>[<scale>] [<color>]</code>	
<code>\dSey</code>	<code>[<scale>] [<color>]</code>	
<code>\dXey</code>	<code>[<scale>] [<color>]</code>	
<code>\dInnocey</code>	<code>[<scale>] [<color>] [<half color>]</code>	
<code>\dCooley</code>	<code>[<scale>] [<color>]</code>	
<code>\dTongey</code>	<code>[<scale>] [<color>] [<tongue color>]</code>	




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
<code>\Strichmaxerl</code>	<code>[<scale>] [<left arm>] [<right arm>] [<left leg>] [<right leg>]</code>	
<code>\Candle</code>	<code>[<scale>]</code>	
<code>\Fire</code>	<code>[<scale>]</code>	
<code>\Coffeecup</code>	<code>[<scale>]</code>	
<code>\Chair</code>	<code>[<scale>]</code>	
<code>\Bed</code>	<code>[<scale>]</code>	
<code>\Moai</code>	<code>[<scale>]</code>	
<code>\Tribar</code>	<code>[<scale>] [<color 1>] [<color 2>] [<color 3>]</code>	
<code>\Snowman</code>	<code>[<scale>]</code>	

\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








$\langle scale \rangle$ can be a number between (not exactly) -900 and (again not exactly) 900 , default is 1 .

$\langle color \rangle$ can be every defined color (see examples below). Note: The color names shouldn't contain special characters like β , \ddot{a} , \ddot{o} , \dots

$\langle leaf \rangle$ uses the colors of $\langle leaf\ color\ a \rangle$ and $\langle leaf\ color\ b \rangle$, 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 [3.2.2](#) and section [3.2.3](#)) to make \LaTeX faster.

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

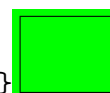
Commands	Optional/Needed parameter(s)	Output
<code>\BasicTree</code>	$[\langle scale \rangle] \{ \langle trunk\ color \rangle \} \{ \langle leaf\ color\ a \rangle \} \{ \langle leaf\ color\ b \rangle \} \{ \langle leaf \rangle \}$	see below
<code>\Springtree</code>	$[\langle scale \rangle]$	
<code>\Summertree</code>	$[\langle scale \rangle]$	
<code>\Autumntree</code>	$[\langle scale \rangle]$	
<code>\Wintertree</code>	$[\langle scale \rangle]$	
<code>\WorstTree</code>	$[\langle scale \rangle]$	

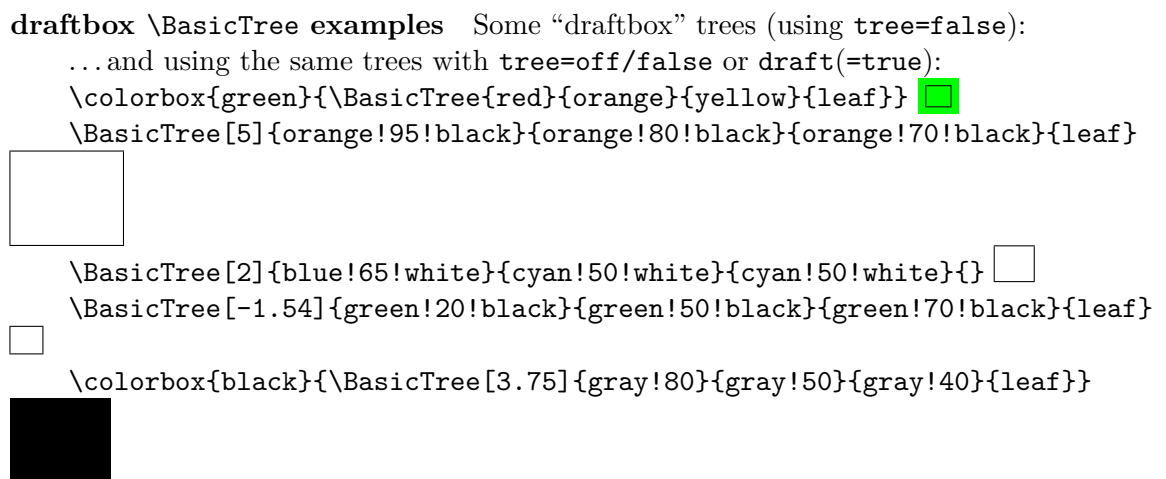
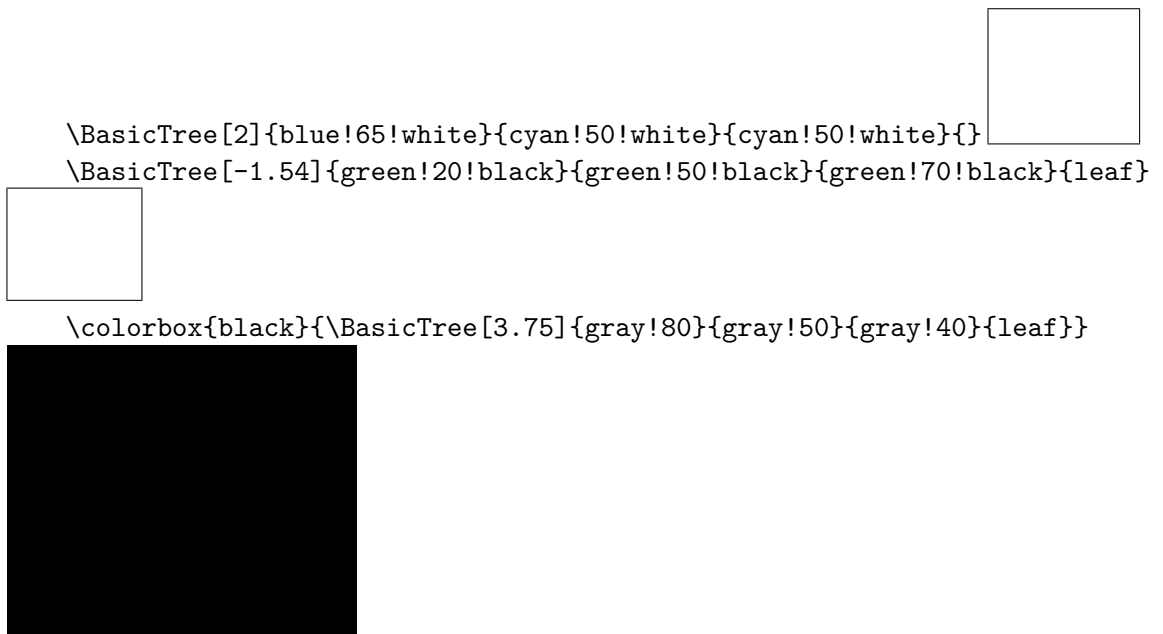
`\BasicTree` **examples** Some “normal” trees:

```

\colorbox{green}{\BasicTree{red}{orange}{yellow}{leaf}}
\BasicTree[5]{orange!95!black}{orange!80!black}{orange!70!black}{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`, L^AT_EX 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 “`français`” 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.