

# The `tikzsymbols` package\*

Ben Vitecek  
b.vitecek@gmx.at  
GitHub

March 16, 2018

## 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<sup>1</sup>  $\square$ .








## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Important changes</b>	<b>3</b>
<b>3</b>	<b>Options</b>	<b>3</b>
3.1	Load-time Options . . . . .	3
3.1.1	marvosym (true/false) . . . . .	3
3.1.2	prefix ( $\langle string \rangle$ ) . . . . .	4
3.2	Normal Options . . . . .	4
3.2.1	draft (true/false) . . . . .	5
3.2.2	final (true/false) . . . . .	5
3.2.3	tree (true/false/on/off) . . . . .	5
3.2.4	after-symbol ( $\langle string \text{ or } command \rangle$ ) . . . . .	5
3.2.5	global-scale ( $\langle number \rangle$ ) symbol-scale ( $\langle key-value \text{ list} \rangle$ ) . . . . .	6
3.2.6	append-style ( $\langle tikz' \text{ keyval} \rangle$ ) . . . . .	6
3.2.7	baseline (true/false) . . . . .	7
3.2.8	remember-picture (true/false) . . . . .	7
3.2.9	usebox (true/false) . . . . .	7

---

\*This document corresponds to `tikzsymbols` v4.07, dated 2017/09/05.

<sup>1</sup>They are – of course – on purpose (expect for “avaiable” (sic!)).

<b>4</b>	<b>Symbols</b>	<b>7</b>
4.1	Cooking-symbols 	8
4.2	Emoticons 	9
4.2.1	“Normal” Emoticons 	9
4.2.2	“3D” Emoticons  	11
4.3	Other Symbols 	12
4.4	Trees 	13
<b>5</b>	<b>Known errors &amp; Problems</b>	<b>14</b>
<b>6</b>	<b>FAQ (Known errors and problems)</b>	<b>14</b>
6.1	How to get rid of the space after each symbol? . . . . .	15
6.2	Using the symbols causes unwanted <code>&lt;problem&gt;</code> . How could I get rid of it? . . . . .	15
6.3	I am getting the error-message <code>Argument of \pgffor@next has an extra }</code> . . . . .	15
6.4	Another package I load already defines <code>&lt;symbol&gt;</code> . . . . .	15
6.5	Does this package store symbols in boxes and reuses them instead of creating a new picture every time? . . . . .	15
6.6	Are the symbols created with the environment <code>tikzpicture</code> ? . .	16
<b>7</b>	<b>Nobody is perfect</b>	<b>16</b>
<b>8</b>	<b>Danksagung</b>	<b>16</b>
<b>9</b>	<b>Changes</b>	<b>16</b>

## 1 Introduction

As far as I can remember this package is the result of me writing a cooking book<sup>2</sup>. 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<sup>3</sup> I somehow ended up with this package.

During time L<sup>A</sup>T<sub>E</sub>X3 became known to me and I started experimenting and programming in this (I would say due to its simplicity compared to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> far superior) language. Well, long story short: I was impressed. And so the idea of writing my package in L<sup>A</sup>T<sub>E</sub>X3 was born.

I finally took my time and started rewriting my code using L<sup>A</sup>T<sub>E</sub>X3. This process can be summarized as: “What *does* this command?”, “Why did I define

<sup>2</sup>Well, it’s one result, the other one is a cooking book.

<sup>3</sup>Not that it’s now any better.

*this* command?” and more generally “*What* have I done?!” Well, let’s hope my code (and grammar) is better this time<sup>4</sup>.

Well . . . that’s it, have fun!

## 2 Important changes

The package should behave the same way as the “old” L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> release.

**2017** Option `usebox` can be used during the document.

**old** 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.

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

## 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.

---

<code>\tikzsymbolsset</code>	<code>\tikzsymbolsset {⟨keys = values⟩}</code>
------------------------------	--

---

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`:

---

<sup>4</sup>Looking at own risk. You have been warned.

Without option “marvosym”: <input type="checkbox"/> <input type="checkbox"/>	With option “marvosym”: ☺ ☹
<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 3.1.2).

### 3.1.2 `prefix` ( $\langle string \rangle$ )

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

$\langle prefix \rangle$  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  $\langle prefix \rangle$  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  $\langle prefix \rangle$ , just write the command name and this command adds automatically the given prefix to the command name.

**Examples:** `\tikzsymbolsuse{Smiley}[2]` ☐

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

`\tikzsymbolsuse{Ofen}` ☐

`\tikzsymbolsuse{Fire}[-1.3]` ☐

etc.

## 3.2 Normal Options

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

### 3.2.1 draft (true/false)

---

**draft** draft =  $\langle true/false \rangle$ 

---

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 (with the same height and width as the symbols<sup>5</sup>) which are faster to create.

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

### 3.2.2 final (true/false)

---

**final** final=  $\langle true/false \rangle$ 

---

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.

### 3.2.3 tree (true/false/on/off)

---

**tree** tree=  $\langle true/on/false/off \rangle$ 

---

This key accepts **true**, **false** and furthermore **on** and **off** (for historical reasons). The latter do exactly the same as the first ones.

This option has only an effect on the command `\BasicTree` and its 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 ( $\langle string \text{ or } command \rangle$ )

---

**after-symbol** after-symbol =  $\{\langle string \text{ or } command \rangle\}$ 

---

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 ( $\langle number \rangle$ ) symbol-scale ( $\langle key-value list \rangle$ )

---

global-scale	global-scale = { $\langle number \rangle$ }
symbol-scale	symbol-scale = { $\langle symbol-1=number-1, symbol-2=number-2, \dots \rangle$ }

---


global-scale can be used to scale *all* commands by given  $\langle number \rangle$ .

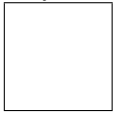
If only some specific symbols should be scaled, you may use the second option and specify which symbol or symbols (name of the symbol without backslash) should be scaled. Using the german name (if available) has the same effect as using the english one.



**Note:** You can scale the symbols in this package in three different ways: The first is to scale *all* symbols using **global-scale**, the second is scaling specific symbols using **symbol-scale** and the third is by using the optional argument provided by the symbols (which I call **local-scale**; e.g. `\Smiley[2]`).

The important thing is that those scaling methods *do not cancel* each other, but behave multiplicative.

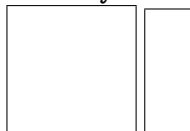
If a local scale is given (e.g. `\Smiley[2]`) with **global-scale**=3 the resulting scaling will be  $3 \cdot 2 = 6$ . Is furthermore this specific symbol is also scaled (e.g. by 1.1), the resulting scaling (for this symbol) will be  $3 \cdot 1.1 \cdot 2 = 6.6$ .

**Examples:** `\tikzsymbolsset{symbol-scale={ Smiley= 1.5 }}` 

`\tikzsymbolsset{symbol-scale={ Smiley= 5 }}` 

`\tikzsymbolsset{symbol-scale={ Smiley= 2, Schneebesen=2.1 }}`  

`\tikzsymbolsset{global-scale=3,symbol-scale={ Smiley= 2, Schneebesen=2.1 }}`



**Note:** Using “eggbeater” instead of “Schneebesen” does the same thing.

### 3.2.6 append-style ( $\langle tikz' keyval \rangle$ )

---

append-style	append-style = { $\langle tikz' keyval \rangle$ }
--------------	---

---

With this option you can append `tikz'  $\langle keyval \rangle$`  to `tikzsymbols` internal style.

**Note:** The style is called `__tikzsymbols` and while the name will probably not change, you are discouraged to use it directly unless it is *really* necessary).

### 3.2.7 baseline (true/false)

---

---

**baseline**    `baseline = {\true/false}`

This option mainly exists to let the commands of this package work inside `todonotes' \todo` command. If set to `true` it 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.

### 3.2.8 remember-picture (true/false)

---

---

**remember-picture**    `remember-picture = {\true/false}`

Adds to each symbol created by this package the tikz option `remember picture=\true/false`. It is set to `false` by default.

### 3.2.9 usebox (true/false)

---

---

**usebox**    `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`.

## 4 Symbols

In this section the symbols are introduced. They ☐ all ☐ change ☐ automatically ☐ with ☐ text-size ☐.

















## 4.1 Cooking-symbols

\Kochtopf  
 \pot  
 \Bratpfanne  
 \fryingpan  
 \Schneebeesen  
 \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

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.

$\langle scale \rangle$  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	$[\langle scale \rangle]$	
\Bratpfanne    \fryingpan	$[\langle scale \rangle]$	
\Schneebeesen    \eggbeater	$[\langle scale \rangle]$	
\Sieb    \sieve	$[\langle scale \rangle]$	
\Purierstab    \blender	$[\langle scale \rangle]$	
\Dreizack    \trident	$[\langle scale \rangle]$	
\Backblech    \bakingplate	$[\langle scale \rangle]$	
\Ofen    \oven	$[\langle scale \rangle]$	
\Pfanne    \pan	$[\langle scale \rangle]$	
\Herd    \cooker	$[\langle scale \rangle]$	
\Saftpresse    \squeezer	$[\langle scale \rangle]$	
\Schussel    \bowl	$[\langle scale \rangle]$	
\Schaler    \peeler	$[\langle scale \rangle]$	
\Reibe    \grater	$[\langle scale \rangle]$	
\Flasche    \bottle	$[\langle scale \rangle]$	
\Nudelholz    \rollingpin	$[\langle scale \rangle]$	



## 4.2 Emoticons □

### 4.2.1 “Normal” Emoticons □

---

\Smiley  
 \Sadey  
 \Neutrey  
 \Changey  
 \cChangey  
 \Annoey  
 \Laughy  
 \Winkey  
 \oldWinkey  
 \Sey  
 \Xey  
 \Innocey  
 \wInnocey  
 \Cooley  
 \Tongey  
 \Nursey  
 \Vomey  
 \Walley  
 \rWalley  
 \Cat  
 \SchrodingersCat  
 \Ninja  
 \Sleepy  
 \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).

\SchrodingersCat's *<case>* can either be  $1$  (alive),  $0$  (unknown) or  $-1$  (dead).

	Commands	(Optional) parameter(s)	Output
	\Smiley	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ]	□
	\Sadey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ]	□
	\Neutrey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ]	□
	\Changey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ] { <i>&lt;mood&gt;</i> }	□
	\cChangey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color1&gt;</i> ] [ <i>&lt;color2&gt;</i> ] [ <i>&lt;color3&gt;</i> ] { <i>&lt;mood&gt;</i> }	□
	\Annoey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ]	□
	\Laughy	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ] [ <i>&lt;mouth color&gt;</i> ]	□
	\Winkey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ]	□
	\oldWinkey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ]	□
	\Sey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ]	□
	\Xey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ]	□
	\Innocey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ] [ <i>&lt;halo color&gt;</i> ]	□
	\wInnocey	[ <i>&lt;scale&gt;</i> ]	□
	\Cooley	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ]	□
	\Tongey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ] [ <i>&lt;tongue color&gt;</i> ]	□
	\Nursey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ] [ <i>&lt;cap color&gt;</i> ] [ <i>&lt;cross color&gt;</i> ]	□
	\Vomey	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ] [ <i>&lt;vomit color&gt;</i> ]	□
	\Walley	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ] [ <i>&lt;wall color&gt;</i> ]	□
	\rWalley	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ] [ <i>&lt;wall color&gt;</i> ]	□
	\Cat	[ <i>&lt;scale&gt;</i> ]	□
	\SchrodingersCat	[ <i>&lt;scale&gt;</i> ] { <i>&lt;case&gt;</i> }	□
	\Ninja	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ] [ <i>&lt;headband color&gt;</i> ] [ <i>&lt;eye color&gt;</i> ]	□
	\Sleepy	[ <i>&lt;scale&gt;</i> ] [ <i>&lt;color&gt;</i> ] [ <i>&lt;cap color&gt;</i> ] [ <i>&lt;star color&gt;</i> ]	□
	\NiceReapey	[ <i>&lt;scale&gt;</i> ]	□

“r” for “random generated cracks”.

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}`   
`\Sleepy[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`  
`\dLaughy`  
`\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
<code>\dSmiley</code>	<code>[&lt;scale&gt;] [&lt;color&gt;]</code>	□
<code>\dSadey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;]</code>	□
<code>\dNeutrey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;]</code>	□
<code>\dChangey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;] {&lt;mood&gt;}</code>	□
<code>\dcChangey</code>	<code>[&lt;scale&gt;] [&lt;color1&gt;] [&lt;color2&gt;] [&lt;color3&gt;] {&lt;mood&gt;}</code>	□
<code>\dLaughy</code>	<code>[&lt;scale&gt;] [&lt;color&gt;] [&lt;mouth color&gt;]</code>	□
<code>\dAnnoey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;]</code>	□
<code>\dWinkey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;]</code>	□
<code>\olddWinkey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;]</code>	□
<code>\dSey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;]</code>	□
<code>\dXey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;]</code>	□
<code>\dInnocey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;] [&lt;halo color&gt;]</code>	□
<code>\dCooley</code>	<code>[&lt;scale&gt;] [&lt;color&gt;]</code>	□
<code>\dTongey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;] [&lt;tongue color&gt;]</code>	□
<code>\dNursey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;] [&lt;cap color&gt;] [&lt;cross color&gt;]</code>	□
<code>\dVomey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;] [&lt;vomit color&gt;]</code>	□
<code>\dWalley</code>	<code>[&lt;scale&gt;] [&lt;color&gt;] [&lt;wall color&gt;]</code>	□
<code>\drWalley</code>	<code>[&lt;scale&gt;] [&lt;color&gt;] [&lt;wall color&gt;]</code>	□
<code>\dNinja</code>	<code>[&lt;scale&gt;] [&lt;color&gt;] [&lt;headband color&gt;] [&lt;eye color&gt;]</code>	□
<code>\dSleepey</code>	<code>[&lt;scale&gt;] [&lt;color&gt;] [&lt;cap color&gt;] [&lt;star color&gt;]</code>	□

“r” for “random generated cracks”.

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`

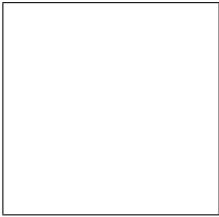
---


`\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






`<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, L<sup>A</sup>T<sub>E</sub>X needs longer to produce the output. So you may use the package option `tree=off`, or (better) `draft=true` (see section 3.2.1 and section 3.2.3) to make L<sup>A</sup>T<sub>E</sub>X faster.

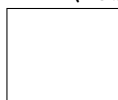
Furthermore those trees are pretty much stolen from the tikz manual.


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

`\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}}`




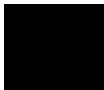


**draftbox \BasicTree examples** Some “draftbox” trees (using `tree=false`):

...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`, L<sup>A</sup>T<sub>E</sub>X 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 FAQ (Known errors and problems)

Or “Questions I assume would be frequently asked, if people would frequently ask questions”.

### 6.1 How to get rid of the space after each symbol?

By default the package adds `\xspace` after each command. To remove it use the option `after-symbol`. Using

```
\tikzsymbolsset{after-symbol={}}
```

removes the `\xspace` command and thus the unwanted space.

### 6.2 Using the symbols causes unwanted *<problem>*. How could I get rid of it?

This could have something to do with question 6.5 (after you made sure that the symbols cause the problem). Try using setting the option `usebox=false` and recompile a few times. If the problem persists, please send a bug report (section 7).

### 6.3 I am getting the error-message Argument of \pgffor@next has an extra }

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.

### 6.4 Another package I load already defines *<symbol>*.

You can override pretty much every symbol simply by loading `tikzsymbols` last as it defines the symbols via `\DeclareDocumentCommand` (see `xparse`).

If you want to use the symbols of both packages you may have a look at option `prefix`.

### 6.5 Does this package store symbols in boxes and reuses them instead of creating a new picture every time?

Yes, it does. It can become a problem if `LATEX` runs out of boxes. If this happens, use `usebox=false`.

Another problem is that if a label is added to a symbol (for example by the `tikz` option `remember picture`) then this label is repeated every time the symbol is used. If you have already used the symbol *before* the label is added, nothing happens as a copy of the symbol without the label is used. If it is used the first time, then the label is also stored and repeated every time the symbol is used later. This behavior can also be fixed by `usebox=false` or adding a `tikz` style by `append-style`.

## 6.6 Are the symbols created with the environment `tikzpicture`?

Yes, they are.

## 7 Nobody is perfect

If you find a bug please send me a mail (or report it on GitHub) involving a *minimal example* showing the bug and a short description (english or german). Please mention (if you are writing a mail) “`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.

## 8 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.

## 9 Changes

See the “`README.md`” file.