

MENUKEYS

Tobias Weh

mail@tobiw.de

<https://tobiw.de/en>

Jonathan P. Spratte

jspratte@yahoo.de

<https://github.com/tweh/menukeys>

<https://www.ctan.org/pkg/menukeys>

☞macros▸latex▸contrib▸menukeys

2026/09/23 — v1.6.4

Abstract

This package is build to format menu sequences, paths and keystrokes.

You're welcome to send me feedback, questions, bug reports and feature requests. If you like to support this package – especially improving or proof-reading the manual – send me an e-mail, please.

Many thanks to Ahmed Musa, who provided the original list parsing code at <https://tex.stackexchange.com/a/44989/4918>.

Contents

1	Introduction	3
2	Installation	3
3	Package loading and options	4
4	Usage	4
4.1	Basics	4
4.2	Styles	5
4.2.1	Predefined styles	5
4.2.2	Declaring styles	10
4.2.3	Copying styles	11
4.2.4	Changing styles	11
4.3	Color themes	12
4.3.1	Predefined themes	12
4.3.2	Create a theme	12
4.3.3	Copy a theme	12
4.3.4	Change a theme	13
4.4	Menu macros	13
4.4.1	Predefined menu macros	13
4.4.2	Defining or changing menu macros	13
4.5	Keys	14
5	Known issues and bugs	14
6	Implementation	15
6.1	Required packages	15
6.2	Helper macros	17
6.3	Options	17
6.4	Workarounds	18
6.4.1	<code>hyperref</code> 's <code>colorlinks</code> option	18
6.5	Color themes	18
6.5.1	Internal commands	18
6.5.2	User-level commands	18
6.5.3	Predefined themes	19
6.6	Styles	20
6.6.1	Internal commands	20
6.6.2	User-level commands	21
6.6.3	Copying and changing	23
6.6.4	Predefined styles	23
6.7	Menu macros	29
6.7.1	Internal commands	29
6.7.2	User-level commands	31
6.7.3	Predefined menu macros	31
6.8	Keys	31
7	Change history	39
8	Macro index	40

1 Introduction

The `menukeys` package is mainly designed to parse and print sequences of software menus, folders and files, or keystrokes. Most predefined styles use the power of `TikZ`¹ to format the output.

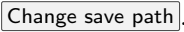
For example if you want to tell the reader of a manual how to set the ruler unit you may type

```
To set the unit of the rulers go to \menu{Extras > Settings > Rulers}
and choose between millimeters, inches and pixels. The shortcut
to view the rulers is \keys{cmd + R}. Pressing these keys again
will hide the rulers.
```

```
The standard path for saving your document is \directory{Macintosh HD/
Users/Your Name/Documents} but you can change it at \menu{Extras >
Settings > Saving} by clicking \menu{Change save path}.
```

and get this:

To set the unit of the rulers go to    and choose between millimeters, inches and pixels. The shortcut to view the rulers is  + . Pressing these keys again will hide the rulers.

The standard path for saving your document is     but you can change it at    by clicking .

The package is loaded as usual via

```
\usepackage{menukeys}
```

2 Installation

To install `menukeys` manually run

```
latex menukeys.ins
```

and copy `menukeys.sty` to a path where `LATEX` can find it.

To typeset this manual run

```
pdflatex menukeys.dtx
makeindex -s gglo.ist -o menukeys.gls menukeys.glo
makeindex -s gind.ist -o menukeys.ind menukeys.idx
pdflatex menukeys.dtx
pdflatex menukeys.dtx
```

¹ See <https://www.ctan.org/pkg/pgf>.

3 Package loading and options

Since `menukeys` used to use `catoptions`, which does some heavy changes on key-value options, it **was** recommended to load `menukeys` as the last package (even after `hyperref`²). **This is no longer the case!**

These are the possible options:

`definenumacros` (opt.) **definenumacros:** Most of `menukeys`' macros should not conflict with other packages³ but the predefined menu macros should be short and easy-to-read commands, which means that `\menu{A,B,C}` is preferred against `\printmenusequence{A,B,C}`. For that it's not unlikely that they conflict with other packages. To prevent this you can tell `menukeys` to not define them by calling the option `definenumacros=false`. The default value is `true`.

If you do so you have to define your own menu macros, see section 4.4 for details.

`definekeys` (opt.) **definekeys:** Equal to `definenumacros` for the key macros. The default value is `true`.

`defineshortnames` (opt.) **defineshortnames:** The key macros are defined with a prefix and without that prefix. You can use `defineshortnames=false` to instruct `menukeys` to omit the shorter variants. The default value is `true`.

`keysprefix` (opt.) **keysprefix:** Since the short key macro names like `\tab` or `\ctrl` might clash with other packages `menukeys` defines them with a common prefix. If a shortname (so without a prefix) clashes `menukeys` will throw a warning, however for the macros with the prefix instead an error will be thrown. If you set an empty prefix the prefix definitions will be omitted and only the shortnames will be used. The default value is `mk` (so in addition to `\tab` you can use `\mktab` to access the same symbol, *etc.*).

`mackeys` (opt.) **mackeys:** This option allows you to decide whether the mac keys are shown as text (`mackeys=text`) or symbols (`mackeys=symbols`). The default value is `symbols`.

`os` (opt.) **os:** You can specify the OS by saying `os=mac` or `os=win`. This will cause some key macros to be rendered differently. The default value is `mac`.

hyperrefcolorlinks: *Obsolete* (see sec. 5 and 6.4.1).

4 Usage

4.1 Basics

`menukeys` comes with three “menu macros” that parse and print lists. We have

`\menu` `\menu{<menu sequence>}`, with `>` as default input list separator, `\directory{<path and files>}` with `/` as default separator and `\keys{<keystrokes>}` with `+` as default

² See <https://tex.stackexchange.com/q/237683/4918> and <https://github.com/tweh/menukeys/issues/41>.

³ If you find a conflict send an e-mail.

separator. You’ve seen examples for all of them in section 1.

These macros have also an optional argument to set the input list separator. E.g. if you want to put in your menus with `,` instead of `>` you can say `\menu[,]{\langle menu sequence \rangle}`.⁴

The possible input separators are `/`, `=`, `*`, `+`, `,`, `;`, `:`, `-`, `>`, `<` and `bslash` (to use `\` as separator). You can hide a separator from the parser by putting a part of the sequence in braces. Spaces around the separator will be ignored, i.e. `\keys{\ctrl+C}` equals `\keys{\ctrl + C}`.

Example `\menu[,]{Extras,Settings,{Units, rulers and origin}}` gives
Extras Settings Units, rulers and origin

4.2 Styles

`menukeys` defines several “styles” that determine the output format of a menu macro. There are some predefined styles and others can be created by the user.

Both pre-defined and custom styles may be applied using the command `\renewmenumacro{\langle macro \rangle}[\langle input sep \rangle]{\langle style \rangle}`. For further information on defining and using macro styles, refer to section 4.4. For example, the “shadowed angular” keys style may be applied using the command `\renewmenumacro{\keys}{\shadowedangularkeys}`.

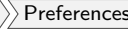
4.2.1 Predefined styles

Name: menus File Extras Preferences Menu	<pre style="color: green;">\newmenumacro\test{menus} \test{File,Extras,Preferences} \test{Menu}</pre>
This is some more or less blind text, to demonstrate how the sequence looks in text. This File Extras Preferences is the result of a style which name is menus. And again some blind text without any sense. <i>This is the default style for \menu.</i>	

Name: roundedmenus File Extras Preferences Menu	<pre style="color: green;">\renewmenumacro\test{roundedmenus} \test{File,Extras,Preferences} \test{Menu}</pre>
This is some more or less blind text, to demonstrate how the sequence looks in text. This File Extras Preferences is the result of a style which name is roundedmenus. And again some blind text without any sense.	

⁴ If you want to change the input separator globally it’s recommended to renew the menu macro as described in section 4.4.

Name: angularmenus	<code>\renewmenumacro\test{angularmenus}</code>
  	<code>\test{File,Extras,Preferences}</code>
	<code>\test{Menu}</code>

This is some more or less blind text, to demonstrate how the sequence looks in text. This    is the result of a style which name is **angularmenus**. And again some blind text without any sense.

Name: roundedkeys	<code>\renewmenumacro\test{roundedkeys}</code>
 +  + 	<code>\test{Ctrl,Alt,Q}</code>
	<code>\test{S}</code>

This is some more or less blind text, to demonstrate how the sequence looks in text. This  +  +  is the result of a style which name is **roundedkeys**. And again some blind text without any sense.

The color of + is taken from optional color B.
This is the default style for \keys.

Name: shadowedroundedkeys	<code>\renewmenumacro\test{shadowedroundedkeys}</code>
 +  + 	<code>\test{Ctrl,Alt,Q}</code>
	<code>\test{S}</code>

This is some more or less blind text, to demonstrate how the sequence looks in text. This  +  +  is the result of a style which name is **shadowedroundedkeys**. And again some blind text without any sense.

The color of + is taken from optional color B.
The shadow color is taken from optional color C.

Name: angularkeys	<code>\renewmenumacro\test{angularkeys}</code>
 +  + 	<code>\test{Ctrl,Alt,Q}</code>
	<code>\test{S}</code>

This is some more or less blind text, to demonstrate how the sequence looks in text. This  +  +  is the result of a style which name is **angularkeys**. And again some blind text without any sense.

The color of + is taken from optional color B.

Name: shadowedangularkeys `\renewmenumacro\test{shadowedangularkeys}`
`\test{Ctrl,Alt,Q}`
`\test{S}`

 +  + 



This is some more or less blind text, to demonstrate how the sequence looks in text. This  +  +  is the result of a style which name is shadowedangularkeys. And again some blind text without any sense.

The color of + is taken from optional color B.
The shadow color is taken from optional color C.

Name: typewriterkeys `\renewmenumacro\test{typewriterkeys}`
`\test{Alt,Q}`
`\test{S}`

 + 



This is some more or less blind text, to demonstrate how the sequence looks in text. This  +  is the result of a style which name is typewriterkeys. And again some blind text without any sense.

The color of + is taken from optional color B.

Name: paths `\renewmenumacro\test{paths}`
`\test{C:,User,Folder,MyFile.tex}`
`\test{MyFile.tex}`

C: ▶ User ▶ Folder ▶ MyFile.tex

MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This C: ▶ User ▶ Folder ▶ MyFile.tex is the result of a style which name is paths. And again some blind text without any sense.

The sep color is taken from optional color C.
This is the default style for \directory.

Name: pathswithfolder `\renewmenumacro\test{pathswithfolder}`
`\test{C:,User,Folder,MyFile.tex}`
`\test{MyFile.tex}`

 C: ▶ User ▶ Folder ▶ MyFile.tex

 MyFile.tex

This is some more or less blind text, to demonstrate how the sequence looks in text. This  C: ▶ User ▶ Folder ▶ MyFile.tex is the result of a style which name is pathswithfolder. And again some blind text without any sense.

The folder draw color is taken from optional color B.
The folder fill color is taken from optional color A.
The sep color is taken from optional color C.

Name: pathswithblackfolder	<code>\renewmenumacro\test{pathswithblackfolder}</code>
	<code>\test{C:,User,Folder,MyFile.tex}</code>
 C: › User › Folder › MyFile.tex	<code>\test{MyFile.tex}</code>
 MyFile.tex	

This is some more or less blind text, to demonstrate how the sequence looks in text. This  C: › User › Folder › MyFile.tex is the result of a style which name is `pathswithblackfolder`. And again some blind text without any sense.

The folder draw color is taken from optional color B.
The folder fill color is taken from optional color C.
The sep color is taken from optional color C.

The following three styles allow paths elements to be hyphenated, but they insert only a line break without a hyphen dash. Note that they only work with T1 and OT1 encoding (at least I tested only these ones) and that this in some cases doesn't work very well.

Name: hyphenatepaths	<code>\renewmenumacro\test{hyphenatepaths}</code>
	<code>\test{C:,Database,User,ALongUserNameHere,</code>
C: › Database › User › ALongUser	<code>ALongerFolderNameAtThisPlace,MyFile.tex}</code>
NameHere › ALongerFolder	<code>\test{MyFile.tex}</code>
NameAtThisPlace › MyFile.tex	
 MyFile.tex	

This is some more or less blind text, to demonstrate how the sequence looks in text. This C: › Database › User › ALongUserNameHere › ALongerFolder NameAtThisPlace › MyFile.tex is the result of a style which name is `hyphenatepaths`. And again some blind text without any sense.

The sep color is taken from optional color C.

Name: hyphenatepathswith	<code>\renewmenumacro\test</code>
folder	<code>{hyphenatepathswithfolder}</code>
	<code>\test{C:,Database,User,ALongUserNameHere,</code>
 C: › Database › User › ALongUser	<code>ALongerFolderNameAtThisPlace,MyFile.tex}</code>
NameHere › ALongerFolder	<code>\test{MyFile.tex}</code>
NameAtThisPlace › MyFile.tex	
 MyFile.tex	

This is some more or less blind text, to demonstrate how the sequence looks in text. This  C: › Database › User › ALongUserNameHere › ALongerFolder NameAtThisPlace › MyFile.tex is the result of a style which name is `hyphenatepathswithfolder`. And again some blind text without any sense.

The folder draw color is taken from optional color B.
The folder fill color is taken from optional color A.
The sep color is taken from optional color C.

```

Name: hyphenatepathswithblack \renewmenumacro\test
folder                        {hyphenatepathswithblackfolder}
                               \test{C:,Database,User,ALongUserNameHere,
C: › Database › User › ALongUser  ALongerFolderNameAtThisPlace,MyFile.tex}
NameHere › ALongerFolder \test{MyFile.tex}
NameAtThisPlace › MyFile.tex
MyFile.tex

```

This is some more or less blind text, to demonstrate how the sequence looks in text. This  C: › Database › User › ALongUserNameHere › ALongerFolderNameAtThisPlace › MyFile.tex is the result of a style which name is hyphenatepathswithblackfolder. And again some blind text without any sense.

*The folder draw color is taken from optional color B.
The folder fill color is taken from optional color C.
The sep color is taken from optional color C.*

`\drawtikzfolder` **Hint** The folder is drawn with the command `\drawtikzfolder` which is part of `menukeys` and has two optional arguments to change the color of the lines and the fill color of the front:

```
\drawtikzfolder[<front fill>][<draw>]
```

4.2.2 Declaring styles

`\newmenustylesimple` The simplest way to define a new style is to use `\newmenustylesimple`. It has six arguments: `\newmenustylesimple<*>{<name>}[<pre>]{<style>}[<sep>][<post>]{<theme>}`

name is the name of the new style. It must follow the specifications of T_EX control sequences, which means it must contain only letters and no numbers.

pre is the code which is executed before a menu macro.

style is the style for the first list element. It has to be a TikZ-style which is applied to a **node**, e.g. `draw,blue`.

sep is the code executed between the lists elements, e.g. some space or a symbol.

post is the code which is executed after a menu macro.

theme is a color theme (see section 4.3).

Example Let us consider we want a list that prints a frame around its elements and separates them by a star. We can use

```
\newmenustylesimple{mystyle}{draw}[$\ast$]{mycolors}
```

`\newmenustyle` The more advanced command is `\newmenustyle`. It has nine arguments: `\newmenustyle<*>{<name>}[<pre>]{<first>}[<sep>]{<mid>}{<last>}{<single>}[<post>]{<theme>}`

name is the name of the new style. It must follow the specifications of T_EX control sequences, which means it must contain only letters and no numbers.

pre is the code which is executed before a menu macro.

first is the style for the first list element. It has to be a TikZ-style which is applied to a **node**, e.g. `draw,blue`.

sep is the code executed between the lists elements, e.g. some space or a symbol.

mid is the style for all elements between the first and the last one. It has to be a TikZ style.

last is the style for the last list element. It has to be a TikZ style.

single this style is used if the list contains only one element. It has to be a TikZ style.

post is the code which is executed after a menu macro.

theme is a color theme (see section 4.3).

Example We can extend the previous example and desire that the first and the last element became red, and a single element should have a dashed frame. Furthermore the menu sequence should be preceded and followed by a bullet point:

```
\newmenustyle{mystyle}[$\bullet$]{draw,red}[$\ast$]%
{draw}{draw,red}{draw,dashed}[$\bullet$]
```

If the TikZ node system doesn't fit your needs there are the **starred versions**: Use them and the arguments $\langle first \rangle$, $\langle mid \rangle$, $\langle last \rangle$, $\langle single \rangle$ can be any L^AT_EX code.

`\CurrentMenuElement` To access the current list element use `\CurrentMenuElement`.

Example consider that we want all menu elements simple be fat and not drawn with a TikZ node. The separator should be the star again:

```
\newmenustylesimple*{mystyle}{\textbf{\CurrentMenuElement}}[$\ast$]
```

If you want to make your own style you must take care of using the color theme. To access a color of the currently applied theme while defining a style use `\usemenucolor{<element>}` (See section 4.3 for details about possible elements).

4.2.3 Copying styles

`\copymenustyle` To copy an existing style to a new style use `\copymenustyle {<copy>}{<original>}`.

Example To copy the definition of `mystyle` to `mycopy` use

```
\copymenustyle{mycopy}{mystyle}
```

4.2.4 Changing styles

The simplest change we can imagine is to change a single element or the color theme of an existing style. For the first case there is `\changemenuelement{<*>}{<name>}{<element>}{<definition>}`, where the starred version works like the one of `\newmenustyle` does.

Example To change the single element of `mystyle` from dashed to solid use the following code. You may save the original style by copying it as described above.

```
\changemenuelement{mystyle}{single}{draw}
```

`\changemenucolortheme` To satisfy the second case use `\changemenucolortheme {<name>}{<color theme>}`.

Example To change the color theme of `mystyle` to `myothercolors` call

```
\changemenucolortheme{mystyle}{myothercolors}
```

`\renewmenustylesimple` The next level is redefining a style. This package provides the following macros the work like their L^AT_EX-paragons and have the same arguments as the `\renewmenustyle` above described macros: `\renewmenustylesimple`, `\providemenustylesimple`, `\providemenustyle` `\renewmenustyle` and `\providemenustyle`.

4.3 Color themes

To make the colors of a style become changeable without touching the style itself, `menukeys` uses “color themes”. Every color theme must contain three color definitions that can be used to draw a `node` background, a `node` frame and a text color, and additionally two optional colors used by some themes.

4.3.1 Predefined themes

There are two predefined color themes

Name: `gray`

Background:  Border:  Text:  (A:  B:  C: )

Name: `blacknwhite`

Background:  Border:  Text:  (A:  B:  C: )

4.3.2 Create a theme

`\newmenucolortheme` To create a new theme use `\newmenucolortheme`. It uses the following arguments:
`\newmenucolortheme{<name>}{<model>}{<bg>}{<br>}{<txt>}[<a>][<b>][<c>]`

name is the name of the theme and must contain only letters.

model is the `xcolor` color model which is used to define a color, e.g. `named`, `rgb`, `cmymk`, ...

bg is the color definition for the `node` background.

br is the color definition for the `node` border.

txt is the color definition for the `node`'s text.

a is an optional additional color (by default same as `bg`).

b is an optional additional color (by default same as `br`).

c is an optional additional color (by default same as `txt`).

Example To create a theme called `mycolors` we can say

```
\newmenucolortheme{mycolors}{named}{red}{green}{blue}
```

4.3.3 Copy a theme

`\copymenucolortheme` To copy the definitions of one theme to another, use `\copymenucolortheme{<copy>}{<original>}`.

Example To copy the colors of `mycolors` to `copycolors` type

```
\copymenucolortheme{copycolors}{mycolors}
```

4.3.4 Change a theme

`\changemenucolor` If you want to change the color of a theme's element use `\changemenucolor{<name>}{<element>}{<model>}{<color definition>}`, where `name` is the theme's name and `<element>` is `bg`, `br`, or `txt`.

Example Let's change the text color of `mycolors`:

```
\changemenucolor{mycolors}{txt}{named}{gray}
```

`\renewmenucolortheme` To redefine a complete theme use `\renewmenucolortheme`. It works with the same arguments as `\newmenucolortheme`.

4.4 Menu macros

The “menu macros” take a list separated by a special symbol to print it with a menu style.

4.4.1 Predefined menu macros

See section 4.1.

4.4.2 Defining or changing menu macros

`\newmenumacro` To define a new menu macro call `\newmenumacro{<macro>}[<input sep>]{<style>}`.

`name` is a L^AT_EX control sequence name.

input sep is the default separator used in the input list (see section 4.1 for a list of valid separators).

If you don't give it the package's default (,) is used.

style is a menu style.

This will give you a macro like `\<macro>[<input sep>]{<list>}`

Example Assuming you need a command to format Windows paths, you can define it with

```
\newmenumacro{\winpath}[bslash]{mystyle}
```

and then use it as e.g. `\winpath{C:\System\Deep\Deeper\YourFile.txt}`. Note that `mystyle` must be defined before you call `\newmenumacro`.

`\providemenumacro` There are also the two commands `\providemenumacro` and `\renewmenumacro`
`\renewmenumacro` which take the same arguments as `\newmenumacro` and work like the L^AT_EX macros `\renewcommand` and `\providecommand`.

Example To change the default input separator of `\menu` you must know the default style (which is `menus`) and then you can say

```
\renewmenumacro{\menu}[,]{menus}
```

4.5 Keys

The `menukeys` package comes with some macros to print special keys in the sequences set with `\keys`. Depending on the given OS (see section 3) some macros behave differently to be able to use a key even if it's undefined via the `os` option macros like `\⟨key⟩mac` and `\⟨key⟩win` that will always give the right symbol.

Each key is defined with two names depending on package options. With `defineshortnames` (opt.) `defineshortnames=true` a macro with only the key name is defined. Additionally `keysprefix` (opt.) every key macro is defined with a prefix declared by the `keysprefix` option. So with `keysprefix=mk` (the default) you can use `\mkshift` to get $\Uparrow$.

The full list of key macros using the shortnames is shown in table 1.

Table 1: Overview of all key macros.

Macro	Mac	Win.	Macro	Mac	Win.
<code>\shift</code>	$\Uparrow$	$\Uparrow$	<code>\winmenu</code>		
<code>\capslock</code>	$\Updownarrow$	$\Downarrow$	<code>\backspace</code>	$\longleftarrow$	$\longleftarrow$
<code>\tab</code>	$\rightarrow$	$\xrightarrow{\leftarrow}$	<code>\del</code>	Del. / $\boxtimes$	Del.
<code>\esc</code>	esc / $\circlearrowleft$	Esc	<code>\backdel</code>	Del. / $\boxtimes$	Del.
<code>\oldesc</code>	esc / $\emptyset$	Esc	<code>\arrowkey{^}</code>	$\uparrow$	$\uparrow$
<code>\ctrl</code>	ctrl	Ctrl	<code>\arrowkeyup</code>	$\uparrow$	$\uparrow$
<code>\Alt</code>	alt / $\neg$	Alt	<code>\arrowkey{v}</code>	$\downarrow$	$\downarrow$
<code>\AltGr</code>		Alt Gr	<code>\arrowkeydown</code>	$\downarrow$	$\downarrow$
<code>\cmd</code>	cmd / $\text{\textasciitilde}$		<code>\arrowkey{>}</code>	$\rightarrow$	$\rightarrow$
<code>\Space</code>	[empty sp.]	[empty sp.]	<code>\arrowkeyright</code>	$\rightarrow$	$\rightarrow$
<code>\SPACE</code>	Space	Space	<code>\arrowkey{<}</code>	$\leftarrow$	$\leftarrow$
<code>\return</code>	$\hookrightarrow$	$\downarrow$	<code>\arrowkeyleft</code>	$\leftarrow$	$\leftarrow$
<code>\enter</code>	$\times$	Enter			

`\arrowkey` The macro `\arrowkey{⟨direction⟩}` is a little special since it takes the direction as a single character $\wedge$, v (lower case v), $>$ or $<$.

`\ctrlname` The texts for `\ctrl`, `\del` and `\SPACE` are saved in `\ctrlname`, `\delname`, `\delname` `\spacename` respectively. So you can change them with `\renewcommand`.

`\spacename` The rendering of some Mac macros depend on the option `mackeys` The different `mackeys` (opt.) versions are shown in the table (left: text, right: symbols).

I apologize that there are no commands for the windows key and the apple logo, but that would be a copyright infringement.

5 Known issues and bugs

- If you use the `inputenc` package `menukeys` must be loaded after it. Otherwise some key macros get corrupted.
- `menukeys` must be loaded after `xcolor`, if you load the latter with options. Otherwise you'll get an option clash. Since `menukeys` loads `xcolor` internally you may pass options as global options via `\documentclass` or directly to it via `\PassOptionsToPackage`.

Example Set xcolor to cmyk model:

```
\documentclass{article}
\PassOptionsToPackage{cmyk}{xcolor}
\usepackage{menukeys}
\begin{document}
  Hello World!
\end{document}
```

If you find something to add to this list please send me an e-mail or report a bug on GitHub (<https://github.com/tweh/menukeys>).

6 Implementation

1 $\langle *pkg \rangle$

6.1 Required packages

Load the required packages

```
2 \RequirePackage{xparse}
3 \RequirePackage{xstring}
4 \RequirePackage{etoolbox}
```

Furthermore we need TikZ and some of its libraries,

```
5 \RequirePackage{tikz}
6 \usetikzlibrary{calc,shapes.symbols,shadows}
```

the color package xcolor and adjustbox for the typewriterkeys style.

```
7 \RequirePackage{xcolor}
8 \RequirePackage{adjustbox}
```

Load relsize to be able to change the font size relative to the surrounding text.

```
9 \RequirePackage{relsize}
```

To define the list parsing commands and allow $\backslash$ as a separator we used to load catoptions. Instead we now use some expl3 functions to replace the macros we required from catoptions.

The first few of these functions are more or less direct equivalents. A bit of attention has to be paid for $\backslash tw@mk@xifinsetTF$ as it requires the arguments to get swapped.

```
10 \ExplSyntaxOn
11 \cs_new_eq:NN \tw@mk@trimspaces \tl_trim_spaces:n
12 \cs_new_eq:NN \tw@mk@exp@Nnno \exp_args:Nnno
13 \cs_new_eq:NN \tw@mk@string \cs_to_str:N
14 \prg_generate_conditional_variant:Nnn \tl_if_in:nn { xx } { TF }
15 \cs_new:Npn \tw@mk@xifinsetTF #1 #2
16 {
17   \tl_if_in:xxTF {#2} {#1}
18 }
```

The replacement for $\backslash indrisloop$ will not set the conditional $\backslash iflastindris$, instead we can check whether the sequence is empty to see whether this is the last element. This test will not use a TeX-like $\backslash iflastindris\dots\else\dots\fi$ construct but instead two branches.

```

19 \cs_new:Npn \tw@mk@iflastindris
20 {
21   \seq_if_empty:NTF \l__twmk_indris_seq
22 }

```

Replacing `\indrisloop` is a bit more work as it requires us to push some values to a stack (to allow nested usage, this may not be necessary for `menukeys`, but it is part of the original `\indrisloop` so we should play nice here). First we'll need a few variables.

```

23 \seq_new:N \l__twmk_indris_seq
24 \int_new:N \l__twmk_indris_int
25 \tl_new:N \l__twmk_indris_tl
26 \cs_new_eq:NN \tw@mk@indrisnr \l__twmk_indris_int
27 \seq_new:N \l__twmk_indris_seqstack_seq
28 \seq_new:N \l__twmk_indris_intstack_seq

```

Our stack will use another sequence in which the definitions of the parent call will be stored for the sequence and the integer. The other variables put on a stack by `\indrisloop` aren't required. The synopsis of `\tw@mk@indrisloop` will be different to the one provided by `catoptions`. The delimiter will be a mandatory argument (not in brackets), and there is no starred version.

```

29 \cs_new_protected:Npn \__twmk_pushseq:
30 {
31   \seq_push:N \l__twmk_indris_seqstack_seq \l__twmk_indris_seq
32 }
33 \cs_new_protected:Npn \__twmk_pushint:
34 {
35   \seq_push:N \l__twmk_indris_intstack_seq \l__twmk_indris_int
36 }
37 \cs_new_protected:Npn \__twmk_popseq:
38 {
39   \seq_if_empty:NTF \l__twmk_indris_seqstack_seq
40   { \seq_clear:N \l__twmk_indris_seq }
41   { \seq_pop:NN \l__twmk_indris_seqstack_seq \l__twmk_indris_seq }
42 }
43 \cs_new_protected:Npn \__twmk_popint:
44 {
45   \seq_if_empty:NTF \l__twmk_indris_intstack_seq
46   { \int_zero:N \l__twmk_indris_int }
47   {
48     \group_begin:
49     \seq_pop:NN \l__twmk_indris_intstack_seq \l__twmk_indris_tl
50     \exp_args:NNNo
51     \group_end:
52     \int_set:Nn \l__twmk_indris_int \l__twmk_indris_tl
53   }
54 }

```

The real loop works by first splitting the input into a sequence according to the delimiter in `#1`. Then this sequence is stepped through, but instead of using `\seq_map:NN` we'll have to pop the sequence into a local variable so that our test for the last element works. The parameter `#2` has to be expanded once as it is handed in as a token storing the real argument in later use.

```

55 \cs_generate_variant:Nn \seq_set_split:Nnn { Noo }

```

```

56 \cs_new_protected:Npn \tw@mk@indrisloop #1 #2 #3
57 {
58   \__twmk_pushseq:
59   \__twmk_pushint:
60   \seq_set_split:Noo \l__twmk_indris_seq {#1} {#2}
61   \int_zero:N \l__twmk_indris_int
62   \bool_do_while:nn { \bool_not_p:n { \seq_if_empty_p:N \l__twmk_indris_seq } }
63   {
64     \int_incr:N \l__twmk_indris_int
65     \seq_pop_left:NN \l__twmk_indris_seq \l__twmk_indris_tl
66     \exp_args:No #3 \l__twmk_indris_tl
67   }
68   \__twmk_popseq:
69   \__twmk_popint:
70 }
71 \ExplSyntaxOff

```

6.2 Helper macros

```

\tw@mk@error Define macros to call \PackageError and warnings
\tw@mk@warning 72 \newcommand*{\tw@mk@error}[2] [Please consult the manual for more information.]{%
\tw@mk@warning@noline 73   \PackageError{menukeys}{#2}{#1}%
74 }
75 \newcommand*{\tw@mk@warning}[1]{%
76   \PackageWarning{menukeys}{#1}%
77 }
78 \newcommand*{\tw@mk@warning@noline}[1]{%
79   \PackageWarningNoLine{menukeys}{#1}%
80 }

\tw@mk@tempa Some commands for temporary use:
\tw@mk@tempb 81 \def\tw@mk@tempa{}
82 \def\tw@mk@tempb{}

\tw@mk@gobble@args Define a command to gobble arguments.
83 \DeclareDocumentCommand{\tw@mk@gobble@args}{m}{%
84   \RenewDocumentCommand{\tw@mk@tempa}{#1}{}%
85   \tw@mk@tempa%
86 }

```

6.3 Options

First we declare and process the package options

```

87 \RequirePackage{kvoptions}
88 \SetupKeyvalOptions{
89   family=tw@mk,
90   prefix=tw@mk@
91 }
92 \DeclareBoolOption[true]{definenummacros}
93 \DeclareBoolOption[true]{definekeys}
94 \DeclareBoolOption[true]{defineshortnames}
95 \DeclareBoolOption[false]{hyperrefcolorlinks}
96 \DeclareStringOption[mk]{keysprefix}

```

```

97 \DeclareStringOption[mac]{os}
98 \DeclareStringOption[symbols]{macks}
99 \ProcessKeyvalOptions{tw@mk}\relax

```

Now we have to do some error treatment:

```

100 \IfSubStr{.mac.win.}{.\tw@mk@os.}{}{%
101   \tw@mk@error{Unknown value for option 'os'\MessageBreak
102   Possible values are 'mac' or 'win'.}%
103 }
104 \IfSubStr{.symbols.text.}{.\tw@mk@macks.}{}{%
105   \tw@mk@error{Unknown value for option 'macks'\MessageBreak
106   Possible values are 'symbols' or 'text'.}%
107 }

```

6.4 Workarounds

Some workarounds to “solve” some incompatibilities:

6.4.1 hyperref’s colorlinks option

There used to be an issue with using the `colorlinks` option of `hyperref` due to `catoptions` being loaded. Since `catoptions` isn’t required any more, this workaround isn’t necessary. For backwards compatibility the `hyperrefcolorlinks` option is still evaluated and just uses `\hypersetup` or `\PassOptionsToPackage` depending on whether `hyperref` is already loaded.

```

108 \iftw@mk@hyperrefcolorlinks
109   \tw@mk@warning{The option 'hyperrefcolorlinks' is obsolete}
110   \ifpackage{hyperref}
111     {\hypersetup{colorlinks}}
112     {\PassOptionsToPackage{colorlinks}{hyperref}}
113 \fi

```

6.5 Color themes

6.5.1 Internal commands

`\tw@make@color@theme` First we define an internal command to make a color theme

```

114 \newcommand*{\tw@make@color@theme}[8]{%
115   \definecolor{tw@color@theme@#1@bg}{#2}{#3}%
116   \definecolor{tw@color@theme@#1@br}{#2}{#4}%
117   \definecolor{tw@color@theme@#1@txt}{#2}{#5}%
118   \definecolor{tw@color@theme@#1@a}{#2}{#6}%
119   \definecolor{tw@color@theme@#1@b}{#2}{#7}%
120   \definecolor{tw@color@theme@#1@c}{#2}{#8}%
121 }

```

6.5.2 User-level commands

`\newmenucolortheme` After that we define the user-level commands:

```

\renewmenucolortheme 122 \NewDocumentCommand{\newmenucolortheme}{ m m m m m O{#3} O{#4} O{#5} }{%
123   \ifundefinedcolor{tw@color@theme@#1@bg}{%
124     \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
125   }{%

```

```

126     \tw@mk@error{Color theme '#1' already defined!\MessageBreak
127     Use \string\renewmenucolortheme\space instead.}%
128 }
129 }
130 \NewDocumentCommand{\renewmenucolortheme}{ m m m m m O{#3} O{#4} O{#5} }{%
131     \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
132 }

```

\changemenucolor Lastly we define the changing and copying commands

```

\copymenucolortheme 133 \newcommand*{\changemenucolor}[4]{%
134     \IfSubStr{ bg br txt }{ #2 }{%
135         \definecolor{tw@color@theme@#1@#2}{#3}{#4}%
136     }{%
137         \tw@mk@error{No such color element ('#2')!\MessageBreak
138         Possible values are bg, br and txt.}
139     }%
140 }
141 \newcommand*{\copymenucolortheme}[2]{%
142     \@ifundefinedcolor{tw@color@theme@#1@bg}{%
143         \colorlet{tw@color@theme@#1@bg}{tw@color@theme@#2@bg}%
144         \colorlet{tw@color@theme@#1@br}{tw@color@theme@#2@br}%
145         \colorlet{tw@color@theme@#1@txt}{tw@color@theme@#2@txt}%
146         \colorlet{tw@color@theme@#1@a}{tw@color@theme@#2@a}%
147         \colorlet{tw@color@theme@#1@b}{tw@color@theme@#2@b}%
148         \colorlet{tw@color@theme@#1@c}{tw@color@theme@#2@c}%
149     }{%
150         \tw@mk@error{Color theme '#1' already defined!\MessageBreak
151         Use \string\renewmenucolortheme\space instead.}
152     }
153 }

```

\changemenucolortheme To be able to change the color theme of a style we must define this:

```

154 \newcommand{\changemenucolortheme}[2]{%
155     \ifcsundef{tw@style@#1@pre}{%
156         \tw@mk@error{Style '#1' undefined!\MessageBreak
157         Maybe you misspelled it?}%
158     }{%
159         \@ifundefinedcolor{tw@color@theme@#2@bg}{%
160             \tw@mk@error{Color theme '#2' is not defined!}%
161         }{%
162             \csdef{tw@style@#1@color@theme}{#2}%
163         }%
164     }%
165 }

```

\usemenucolor To use a color of a theme we define \usemenucolor as following.

```

166 \newcommand{\usemenucolor}[1]{%
167     tw@color@theme@tw@current@color@theme @#1%
168 }

```

6.5.3 Predefined themes

There are two predefined color themes

```

169 \newmenucolortheme{gray}{gray}{0.95}{0.3}{0}{0.95}[0][0]
170 \newmenucolortheme{blacknwhite}{gray}{1}{0}{0}{1}[0][0]

```

6.6 Styles

The style generating commands will set some commands that are named like `\tw@style@<name>@<element>`.

```

\tw@default@sep Before we can define the internal declaring macro to use it later in the user level
\tw@default@pre commands, we have to set some defaults for the optional arguments
\tw@default@post 171 \newcommand{\tw@default@sep}{%
                  172   \hspace{0.2em plus 0.1em minus 0.5em}%
                  173 }
                  174 \newcommand{\tw@default@pre}{%
                  175 \newcommand{\tw@default@post}{%

```

6.6.1 Internal commands

Now we can define the internal commands.

```

\tw@declare@style@simple Our first step is to define the simple command.
176 \DeclareDocumentCommand{\tw@declare@style@simple}{%
177   s m O{\tw@default@pre} m O{\tw@default@sep} O{\tw@default@post} m
178 }{%
179   \csdef{tw@style@#2@color@theme}{#7}%
180   \csdef{tw@style@#2@pre}{#3}%
181   \csdef{tw@style@#2@sep}{#5}%
182   \csdef{tw@style@#2@post}{#6}%
183   \IfBooleanTF{#1}{%
184     \csdef{tw@style@#2@single}{#4}%
185     \csdef{tw@style@#2@first}{#4}%
186     \csdef{tw@style@#2@mid}{#4}%
187     \csdef{tw@style@#2@last}{#4}%
188   }{%
189     \csdef{tw@style@#2@single}{%
190       \tikz[baseline=(tw@node.base)]{%
191         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
192     \csdef{tw@style@#2@first}{%
193       \tikz[baseline=(tw@node.base)]{%
194         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
195     \csdef{tw@style@#2@mid}{%
196       \tikz[baseline=(tw@node.base)]{%
197         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
198     \csdef{tw@style@#2@last}{%
199       \tikz[baseline=(tw@node.base)]{%
200         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
201     }%
202 }

```

```

\tw@declare@style The next step is to create the extended command. This command must have ten
\tw@declare@style@extra@args arguments (including the star) so we have to define a helping macro to grab the
last two macros.

```

```

203 \DeclareDocumentCommand{\tw@declare@style@extra@args}{%

```

```

204   0{\tw@default@post} m
205 }{%
206   \csdef{tw@style@\tw@current@style @post}{#1}%
207   \csdef{tw@style@\tw@current@style @color@theme}{#2}%
208 }

```

Now we can define `\tw@declare@style`:

```

209 \DeclareDocumentCommand{\tw@declare@style}{%
210   s m 0{\tw@default@pre} m 0{\tw@default@sep} m m m
211 }{%
212   \def\tw@current@style{#2}
213   \csdef{tw@style@#2@pre}{#3}%
214   \csdef{tw@style@#2@sep}{#5}%
215   \IfBooleanTF{#1}{%
216     \csdef{tw@style@#2@single}{#8}%
217     \csdef{tw@style@#2@first}{#4}%
218     \csdef{tw@style@#2@mid}{#6}%
219     \csdef{tw@style@#2@last}{#7}%
220   }{%
221     \csdef{tw@style@#2@single}{%
222       \tikz[baseline=(tw@node.base)]{%
223         \node(tw@node)[#8]{\strut\CurrentMenuElement};}%
224     \csdef{tw@style@#2@first}{%
225       \tikz[baseline=(tw@node.base)]{%
226         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
227     \csdef{tw@style@#2@mid}{%
228       \tikz[baseline=(tw@node.base)]{%
229         \node(tw@node)[#6]{\strut\CurrentMenuElement};}%
230     \csdef{tw@style@#2@last}{%
231       \tikz[baseline=(tw@node.base)]{%
232         \node(tw@node)[#7]{\strut\CurrentMenuElement};}%
233   }%
234   \tw@declare@style@extra@args%
235 }

```

6.6.2 User-level commands

`newmenustylesimple` It's time to define the user-level commands now:

```

renewmenustylesimple 236 \NewDocumentCommand{\newmenustylesimple}{s m}{%
providemenustylesimple 237   \ifcsundef{tw@style@#2@pre}{%
newmenustyle 238     \IfBooleanTF{#1}{%
renewmenustyle 239       \tw@declare@style@simple*{#2}%
providemenustyle 240     }{%
241       \tw@declare@style@simple{#2}%
242     }%
243   }{%
244     \tw@mk@error{Style '#2' already defined!\MessageBreak
245       Use \string\renewmenustylesimple\space instead.}%
246     \tw@mk@gobble@args{o m o m}%
247   }%
248 }
249 \NewDocumentCommand{\renewmenustylesimple}{s m}{%
250   \IfBooleanTF{#1}{%
251     \tw@declare@style@simple*{#2}%

```

```

252 }{%
253     \tw@declare@style@simple{#2}%
254 }%
255 }
256 \NewDocumentCommand{\providemenustylesimple}{s m}{%
257     \ifcsundef{tw@style@#2@pre}{%
258         \IfBooleanTF{#1}{%
259             \tw@declare@style@simple*{#2}%
260         }{%
261             \tw@declare@style@simple{#2}%
262         }%
263     }{%
264         \tw@mk@warning{Trying to provide style ' #2' failed,\MessageBreak
265         because it's already defined.\MessageBreak
266         You may use \string\renewmenustylesimple\space instead.}%
267         \tw@mk@gobble@args{o m o m}%
268     }%
269 }
270
271 \NewDocumentCommand{\newmenustyle}{s m}{%
272     \ifcsundef{tw@style@#2@pre}{%
273         \IfBooleanTF{#1}{%
274             \tw@declare@style*{#2}%
275         }{%
276             \tw@declare@style{#2}%
277         }%
278     }{%
279         \tw@mk@error{Style ' #2' already defined!\MessageBreak
280         Use \string\renewmenustyle\space instead.}%
281         \tw@mk@gobble@args{o m o m m m o m}%
282     }%
283 }
284 \NewDocumentCommand{\renewmenustyle}{s m}{%
285     \IfBooleanTF{#1}{%
286         \tw@declare@style*{#2}%
287     }{%
288         \tw@declare@style{#2}%
289     }%
290 }
291 \NewDocumentCommand{\providemenustyle}{s m}{%
292     \ifcsundef{tw@style@#2@pre}{%
293         \IfBooleanTF{#1}{%
294             \tw@declare@style*{#2}%
295         }{%
296             \tw@declare@style{#2}%
297         }%
298     }{%
299         \tw@mk@warning{Trying to provide style #2 failed,\MessageBreak
300         because it's already defined.\MessageBreak
301         You may use \string\renewmenustyle\space instead.}%
302         \tw@mk@gobble@args{o m o m m m o m}%
303     }%
304 }

```

6.6.3 Copying and changing

`\copymenustyle` The last two steps in this part are to define a command to copy styles

```

305 \newcommand*{\copymenustyle}[2]{%
306   \ifcsundef{tw@style@#1@pre}{%
307     \ifcsundef{tw@style@#2@pre}{%
308       \tw@mk@error{Can't copy not existing style ('#2')!}%
309     }{%
310       \csletcs{tw@style@#1@pre}{tw@style@#2@pre}%
311       \csletcs{tw@style@#1@post}{tw@style@#2@post}%
312       \csletcs{tw@style@#1@sep}{tw@style@#2@sep}%
313       \csletcs{tw@style@#1@single}{tw@style@#2@single}%
314       \csletcs{tw@style@#1@first}{tw@style@#2@first}%
315       \csletcs{tw@style@#1@mid}{tw@style@#2@mid}%
316       \csletcs{tw@style@#1@last}{tw@style@#2@last}%
317       \csletcs{tw@style@#1@color@theme}{tw@style@#2@color@theme}%
318     }%
319   }{%
320     \tw@mk@error{Style '#1' already exists!}%
321   }%
322 }
```

`\changemenuelement` and one to change a single element of a style.

```

323 \NewDocumentCommand{\changemenuelement}{s m m m}{%
324   \ifcsundef{tw@style@#2@pre}{%
325     \tw@mk@error{Style '#2' undefined.}%
326   }{%
327     \IfSubStr{ single first middle last pre post sep }{ #3 }{%
328       \IfBooleanTF{#1}{%
329         \csdef{tw@style@#2@#3}{#4}%
330       }{%
331         \IfSubStr{ pre post sep }{ #3 }{%
332           \csdef{tw@style@#2@#3}{#4}%
333         }{%
334           \csdef{tw@style@#2@#3}{%
335             \tikz[baseline=(tw@node.base)]{%
336               \node(tw@node)[#4]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
337           }%
338         }%
339       }{\tw@mk@error{No element '#3'. Possible values are\MessageBreak
340         single, first, middle, last, pre, post or sep.}}%
341     }%
342 }
```

6.6.4 Predefined styles

We define several styles for menu sequences, paths and keystrokes.

`tw@set@tikz@colors` First we define a TikZ-style to apply the color theme to a node easily

```

343 \tikzset{tw@set@tikz@colors/.style={%
344   draw=\usemenucolor{br},
345   fill=\usemenucolor{bg},
346   text=\usemenucolor{txt},
347 }}
```

Now we can define the styles. To keep the most settings of a style together we make additional TikZ-styles instead of setting everything directly to the nodes.

```

348 \tikzset{tw@menus@base/.style={%
349     tw@set@tikz@colors,
350     rounded corners=0.15ex,
351     inner sep=0pt,
352     inner xsep=2pt,
353     text height=1.825ex,
354     text depth=0.7ex,
355     minimum width=1.5em,
356     font=\relsize{-1}\sffamily,
357     signal,
358     signal to=nowhere,
359     signal pointer angle=110,
360 }}
361 \tw@declare@style*{menus}{%
362     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
363         \node(tw@node)[tw@menus@base,signal to=east]%
364             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
365 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]{%
366 {%
367     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
368         \node(tw@node)[tw@menus@base,signal from=west,signal to=east]%
369             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
370 }{%
371     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
372         \node(tw@node)[tw@menus@base,signal from=west,%
373             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
374 }{%
375     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
376         \node(tw@node)[tw@menus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
377 }{gray}
378 }
379 \tikzset{tw@roundedmenus@base/.style={%
380     tw@set@tikz@colors,
381     rounded corners=0.3ex,
382     inner sep=0pt,
383     inner xsep=2pt,
384     text height=1.825ex,
385     text depth=0.7ex,
386     minimum width=1.5em,
387     font=\relsize{-1}\sffamily,
388     signal,
389     signal to=nowhere,
390     signal pointer angle=110,
391 }}
392 \tw@declare@style*{roundedmenus}{%
393     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
394         \node(tw@node)[tw@roundedmenus@base,signal to=east]%
395             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
396 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]{%
397 {%
398     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
399         \node(tw@node)[tw@roundedmenus@base,signal from=west,signal to=east]%

```

```

400     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement}};%
401 }{%
402     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
403         \node(tw@node)[tw@roundedmenus@base,signal from=west,]%
404         {\strut\color{\usemenucolor{txt}}\CurrentMenuElement}};%
405 }{%
406     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
407         \node(tw@node)[tw@roundedmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement}};%
408 }{gray}
409
410 \tikzset{tw@angularmenus@base/.style={%
411     tw@set@tikz@colors,
412     inner sep=0pt,
413     inner xsep=2pt,
414     text height=1.825ex,
415     text depth=0.7ex,
416     minimum width=1.5em,
417     font=\relsize{-1}\sffamily,
418     signal,
419     signal to=nowhere,
420     signal pointer angle=110,
421 }}
422 \tw@declare@style*{angularmenus}{%
423     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
424         \node(tw@node)[tw@angularmenus@base,signal to=east]%
425         {\strut\color{\usemenucolor{txt}}\CurrentMenuElement}};%
426 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
427 {%
428     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
429         \node(tw@node)[tw@angularmenus@base,signal from=west,signal to=east]%
430         {\strut\color{\usemenucolor{txt}}\CurrentMenuElement}};%
431 }{%
432     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
433         \node(tw@node)[tw@angularmenus@base,signal from=west,]%
434         {\strut\color{\usemenucolor{txt}}\CurrentMenuElement}};%
435 }{%
436     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
437         \node(tw@node)[tw@angularmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement}};%
438 }{gray}
439
440 \tikzset{tw@roundedkeys@base/.style={%
441     tw@set@tikz@colors,
442     rounded corners=0.3ex,
443     inner sep=0pt,
444     inner xsep=2pt,
445     text height=1.825ex,
446     text depth=0.7ex,
447     minimum width=1.5em,
448     font=\relsize{-1}\sffamily,
449 }}
450 \tw@declare@style@simple*{roundedkeys}{%
451     \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
452         \node(tw@node)[tw@roundedkeys@base]%
453         {\strut\color{\usemenucolor{txt}}\CurrentMenuElement}};%

```

```

454 }[%
455   \hspace{0.1em plus 0.1em minus 0.05em}%
456   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
457   \hspace{0.1em plus 0.1em minus 0.05em}%
458 ]{gray}
459
460 \tikzset{tw@shadowedroundedkeys@base/.style={%
461   tw@set@tikz@colors,
462   rounded corners=0.3ex,
463   inner sep=0pt,
464   inner xsep=2pt,
465   text height=1.825ex,
466   text depth=0.7ex,
467   minimum width=1.5em,
468   font=\relsize{-1}\sffamily,
469   general shadow={%
470     shadow xshift=.2ex, shadow yshift=-.15ex,
471     fill=\usemenucolor{c},
472   },
473 }}
474 \tw@declare@style@simple*{shadowedroundedkeys}{%
475   \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
476     \node(tw@node)[tw@shadowedroundedkeys@base]%
477       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
478   }%
479 }[%
480   \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
481   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
482   \hspace{0.1em plus 0.1em minus 0.05em}%
483 </pkg>
484 <cur>[\kern0.2ex\relax]{gray}
485 <161>[\hspace{0.2ex}]{gray}
486 <*pkg>
487
488 \tikzset{tw@angularkeys@base/.style={%
489   tw@set@tikz@colors,
490   inner sep=0pt,
491   inner xsep=2pt,
492   text height=1.825ex,
493   text depth=0.7ex,
494   minimum width=1.5em,
495   font=\relsize{-1}\sffamily,
496 }}
497 \tw@declare@style@simple*{angularkeys}{%
498   \tikz[baseline={($(tw@node.base)+(0,-0.2ex)$)}]{%
499     \node(tw@node)[tw@angularkeys@base]%
500       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
501 }[%
502   \hspace{0.1em plus 0.1em minus 0.05em}%
503   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
504   \hspace{0.1em plus 0.1em minus 0.05em}%
505 ]{gray}
506

```

```

507 \tikzset{tw@shadowedangularkeys@base/.style={%
508     tw@set@tikz@colors,
509     inner sep=0pt,
510     inner xsep=2pt,
511     text height=1.825ex,
512     text depth=0.7ex,
513     minimum width=1.5em,
514     font=\relsize{-1}\sffamily,
515     general shadow={%
516         shadow xshift=.2ex, shadow yshift=-.15ex,
517         fill=\usemenucolor{c},
518     },
519 }}
520 \tw@declare@style@simple*{shadowedangularkeys}{%
521     \tikz[baseline={($ (tw@node.base)+(0,-0.2ex)$)]{%
522         \node(tw@node)[tw@shadowedangularkeys@base]%
523             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
524 }[%
525     \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
526     \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
527     \hspace{0.1em plus 0.1em minus 0.05em}%

528 </pkg>
529 <cur>[\kern0.2ex\relax]{gray}
530 {161}[\hspace{0.2ex}]{gray}
531 <*pkg>

532
533 \tikzset{tw@typewriterkeys@base/.style={%
534     tw@set@tikz@colors,
535     shape=circle,
536     minimum size=2ex,
537     inner sep=0.5pt, outer sep=1pt,
538     font=\ttfamily\relsize{-1},
539 }}
540 \tw@declare@style@simple*{typewriterkeys}{%
541     \def\tw@typewriterkeys@curr@elem{%
542         \maxsizebox*{2ex}{2ex}{\CurrentMenuElement}%
543     }%
544     \begin{tikzpicture}[baseline={($ (tw@node.south)+(0,0.8ex)$)]%
545         \node(tw@node)[%
546             tw@typewriterkeys@base, inner sep=1.25pt, line width=0.6pt%
547             ]{\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
548         \node[tw@typewriterkeys@base]%
549             {\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
550     \end{tikzpicture}%
551 }[%
552     \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
553     \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
554     \hspace{0.1em plus 0.1em minus 0.05em}%
555 ]{blacknwhite}

556
557 \tw@declare@style@simple*{paths}{%
558     {\ttfamily\color{\usemenucolor{txt}}\CurrentMenuElement}%
559 }[%

```

```

560 \hspace{0.2em plus 0.1em}%
561 \raisebox{0.08ex}{%
562 \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
563 -- (0,1ex) -- cycle;}%
564 }%
565 \hspace{0.2em plus 0.1em}%
566 ]{blacknwhite}
567
568 \newcounter{tw@hyphen@char@num}
569 \newif\if@tw@hyphen@paths@warnig
570 \@tw@hyphen@paths@warnigtrue
571 \tw@declare@style@simple*{hyphen@paths}{%
572 {\ttfamily
573 \IfStrEq{T1}{\encodingdefault}{%
574 \setcounter{tw@hyphen@char@num}{23}%
575 }{%
576 \IfStrEq{OT1}{\encodingdefault}{%
577 \setcounter{tw@hyphen@char@num}{255}%
578 }{%
579 \if@tw@hyphen@paths@warnig%
580 \tw@mk@warning{The hyphen@paths styles will probably only\MessageBreak
581 work with T1 or OT1 encoding.}%
582 \fi\global\@tw@hyphen@paths@warnigfalse%
583 }%
584 }%
585 \hyphenchar\font=\value{tw@hyphen@char@num}\relax
586 \color{\usemenucolor{txt}}}%
587 \CurrentMenuElement}%
588 }[%
589 \hspace{0.2em plus 0.1em}%
590 \raisebox{0.08ex}{%
591 \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
592 -- (0,1ex) -- cycle;}%
593 }%
594 \hspace{0.2em plus 0.1em}%
595 ]{blacknwhite}
596
597 \NewDocumentCommand{\drawtikzfolder}{O{white} O{black}}{%
598 \begin{tikzpicture}[rounded corners=0.02ex,scale=0.7]
599 \draw [#2] (0,0) -- (1em,0) -- (1em,1.5ex) -- (0.5em,1.5ex) -- %
600 (0.4em,1.7ex) -- (0.1em,1.7ex) -- (0,1.5ex) -- cycle;
601 \draw [#2,fill=#1] (0,0) -- (1em,0) -- (0.85em,1.15ex) -- %
602 ++(-1em,0) -- cycle;
603 \end{tikzpicture}%
604 }
605
606 \copymenustyle{pathswithfolder}{paths}
607 \changemenuelement{pathswithfolder}{pre}{%
608 \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
609 \hspace{0.2em plus 0.1em}%
610 }
611
612 \copymenustyle{pathswithblackfolder}{paths}
613 \changemenuelement{pathswithblackfolder}{pre}{%

```

```

614 \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
615 \hspace{0.2em plus 0.1em}%
616 }
617
618 \copymenustyle{hyphenatepathswithfolder}{hyphenatepaths}
619 \changemenuelement{hyphenatepathswithfolder}{pre}{%
620 \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
621 \hspace{0.2em plus 0.1em}%
622 }
623
624 \copymenustyle{hyphenatepathswithblackfolder}{hyphenatepaths}
625 \changemenuelement{hyphenatepathswithblackfolder}{pre}{%
626 \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
627 \hspace{0.2em plus 0.1em}%
628 }

```

6.7 Menu macros

6.7.1 Internal commands

`\tw@default@input@sep` First we define our default input separator

```
629 \def\tw@default@input@sep{,}
```

`\CurrentMenuElement` and the `\CurrentMenuElement` dummy

```
630 \def\CurrentMenuElement{}
```

`\tw@define@menu@macro` Then we set up the internal command to create new menu macros. The list parsing

`\tw@define@menu@macro@` code was essentially provided by Ahmed Musa at <https://tex.stackexchange.com/a/44989/4918>. Jonathan P. Spratte made some major changes to make `menukeys` work without `catoptions` and reimplemented the parsing code using `LaTeX3`. Thank you both very much!

```

631 \begingroup
632 \lccode'\,=1
633 \lowercase{\endgroup
634 \@ifdefinable\tw@mk@test@input@sep
635   {%
636     \protected\def\tw@mk@test@input@sep#1{%
637       \tw@mk@xifinsetTF
638         {,\tw@mk@trimspaces{#1},}{,bslash,backslash,directory,location,}%
639     }%
640   }%
641 }
642 \newcommand\tw@define@menu@macro[3]
643   {%
644     \IfNoValueTF{#3}
645       {\tw@mk@exp@Nno\tw@define@menu@macro@{#1}{#2}\tw@default@input@sep}
646       {\tw@define@menu@macro@{#1}{#2}{#3}}%
647   }
648 \newcommand\tw@define@menu@macro@[4]
649   {%
650     \ifcsundef{tw@style@#4@sep}
651       {%
652         \tw@mk@error

```

```

653         {%
654         Can't define menu macro \string#2\space,\MessageBreak
655         because the style '#4' is not available!%
656     }%
657 }
658 {%
659 \csdef{tw@parse@menu@list@tw@mk@string#2}##1%
660 {%
661     \def\CurrentMenuElement{##1}%
662     \tw@mk@iflastindris
663     {%
664         \ifnum\tw@mk@indrisnr=\@ne
665             \@nameuse{tw@style@#4@single}%
666         \else
667             \@nameuse{tw@style@#4@sep}%
668             \@nameuse{tw@style@#4@last}%
669         \fi
670     }
671     {%
672         \ifnum\tw@mk@indrisnr=\@ne
673             \@nameuse{tw@style@#4@first}%
674         \else
675             \@nameuse{tw@style@#4@sep}%
676             \@nameuse{tw@style@#4@mid}%
677         \fi
678     }%
679 }%
680 #1#2{+0{#3}+m}%
681 {%
682     \leavevmode
683     \begingroup
684     \def\tw@current@color@theme
685     {\csname tw@style@#4@color@theme\endcsname}%
686     \@nameuse{tw@style@#4@pre}%
687     \tw@mk@test@input@sep{##1}
688     {%
689         \edef\tw@menu@list{\detokenize{##2}}%
690         \edef\tw@mk@tempa{\@backslashchar}%
691     }
692     {%
693         \edef\tw@menu@list{\unexpanded{##2}}%
694         \edef\tw@mk@tempa{\tw@mk@trimspaces{##1}}%
695     }%
696     \begingroup
697     \letcs{tw@mk@tempb}{tw@parse@menu@list@tw@mk@string#2}%
698     \tw@mk@indrisloop\tw@mk@tempa\tw@menu@list\tw@mk@tempb
699     \endgroup
700     \@nameuse{tw@style@#4@post}%
701 \endgroup
702 }%
703 }%
704 }

```

6.7.2 User-level commands

```

\newmenumacro Now it's time to build the user-level commands
\renewmenumacro 705 \NewDocumentCommand{\newmenumacro}{m o m}{%
\providemenumacro 706 \ifcsundef{tw@mk@string#1}{%
707 \tw@define@menu@macro\NewDocumentCommand{#1}{#2}{#3}%
708 }{%
709 \tw@mk@error{Menu macro '\string#1' already defined!\MessageBreak
710 Use \string\renewmenustyle\space instead.}%
711 }%
712 }
713 \NewDocumentCommand{\renewmenumacro}{m o m}{%
714 \tw@define@menu@macro\RenewDocumentCommand{#1}{#2}{#3}%
715 }
716 \NewDocumentCommand{\providemenumacro}{m o m}{%
717 \ifcsundef{tw@mk@string#1}{%
718 \tw@define@menu@macro\ProvideDocumentCommand{#1}{#2}{#3}%
719 }{%
720 \tw@mk@warning{Menu macro '\string#1' already defined!\MessageBreak
721 Use \string\renewmenustyle\space to redefine it.}%
722 }%
723 }

```

6.7.3 Predefined menu macros

Now we got all tools to predefine some menu macros. To be sure that these commands won't conflict with other packages we introduced the option `definemacros`. Here we have to check it:

```

724 \iftw@mk@definemacros

\menu And then we define three basic macros.
\directory 725 \newmenumacro{\menu}[>]{menus}
\keys 726 \newmenumacro{\directory}[/{paths}
727 \newmenumacro{\keys}[+]{roundedkeys}

```

Lastly we close the `definemacros` if statement:

```
728 \fi
```

6.8 Keys

Before we define anything we check if the user allows it:

```
729 \iftw@mk@definekeys
```

Before define the key macros we create some macros that save some typing by condensing the similarities between the key macros.

`\tw@make@key@box` The first of these macros helps us building save boxes to store the `{tikzpicture}`, that will draw the key later. This is necessary because otherwise the picture will inherit the style of the key sequence `node`.

```

730 \NewDocumentCommand{\tw@make@key@box}{m m}{%
731 % \expandafter\newbox\csname tw@mk@box@#1\endcsname
732 % \expandafter\sbox\csname tw@mk@box@#1\endcsname{%
733 % #2%

```

```

734 % }%
735 \csdef{tw@mk@#1}{%
736 % \expandafter\usebox\csname tw@mk@box@#1\endcsname%
737 #2%
738 }%
739 }

```

`\tw@make@key@macro` The next macro defines the user level command by accessing a macro like `\tw@make@simple@key@macro` `tw@mk@⟨key⟩` or `tw@mk@⟨key⟩@⟨os⟩`, if the appearance differs between Mac and Windows. To use this macro we assume that the `tw@mk@⟨key⟩` commands are defined.

```

\tw@make@simple@key@macro@aux 740 \NewDocumentCommand{\tw@make@key@macro}{s m}{%
\tw@mk@symbol 741 \iftw@mk@defineshortnames
\tw@mk@warn@if@defined 742 \ExpandArgs{nee}\tw@make@key@macro@aux{#1}%
743 {\tw@mk@string#2}%
744 {\tw@mk@string#2}%
745 \tw@mk@warn@if@defined
746 \providecommand
747 \fi
748 \ifx\@empty\tw@mk@keysprefix
749 \else
750 \ExpandArgs{nee}\tw@make@key@macro@aux{#1}%
751 {\tw@mk@keysprefix\tw@mk@string#2}%
752 {\tw@mk@string#2}%
753 \@gobble
754 \newcommand
755 \fi
756 }
757 \newcommand\tw@make@key@macro@aux[5]{%
758 #4{#2}%
759 \IfBooleanTF{#1}
760 {%
761 #4{#2mac}%
762 #4{#2win}%
763 \expandafter#5\csname#2\endcsname {\tw@mk@symbol{tw@mk@#3\tw@mk@os}}%
764 \expandafter#5\csname#2mac\endcsname{\tw@mk@symbol{tw@mk@#3mac}}%
765 \expandafter#5\csname#2win\endcsname{\tw@mk@symbol{tw@mk@#3win}}%
766 }%
767 {\expandafter#5\csname#2\endcsname{\tw@mk@symbol{tw@mk@#3}}}%
768 }
769 \newcommand\tw@mk@symbol[1]{\maxsizebox{!}{1.8ex}{\@nameuse{#1}}}
770 \newcommand\tw@mk@warn@if@defined[1]
771 {%
772 \ifcsundef{#1}%
773 {}%
774 {\tw@mk@warning{Macro \tw@mk@string\\#1 already defined!}}%
775 }
776 \newcommand\tw@make@simple@key@macro[2]
777 {%
778 \iftw@mk@defineshortnames
779 \ExpandArgs{e}\tw@make@simple@key@macro@aux
780 {\tw@mk@string#1}
781 {#2}%
782 \tw@mk@warn@if@defined
783 \providecommand

```

```

784 \fi
785 \ifx\@empty\tw@mk@keys@prefix
786 \else
787 \ExpandArgs{e}\tw@make@simple@key@macro@aux
788 {\tw@mk@keys@prefix\tw@mk@string#1}%
789 {#2}%
790 \@gobble
791 \newcommand
792 \fi
793 }
794 \newcommand\tw@make@simple@key@macro@aux[4]
795 {%
796 #3{#1}%
797 \expandafter#4\csname#1\endcsname{#2}%
798 }

```

`\tw@define@mackey` The last helping macro is `\tw@define@mackey`. We use it to execute code depending on the `mackeys` option.

```

799 \newcommand*{\tw@define@mackey}[2]{%
800 \IfStrEq{text}{\tw@mk@mackeys}{#1}{%
801 \IfStrEq{symbols}{\tw@mk@mackeys}{#2}{}%
802 }%
803 }

```

Next thing to do is to set up some TikZ-styles.

```

804 \tikzset{
805 menukeys key symbol/.style={
806 rounded corners=0pt,
807 line width=0.1ex,
808 baseline={ (0,0) },
809 >=To,
810 },
811 menukeys thick/.style={line width=0.25ex},
812 }

```

Now we are prepared to generate the key macros. I will be nearly the same way for all keys. Step one is to build a `\tw@mk@<key>` macro and then we define the user-level command `\<key>`

`\shift`

```

813 \normalsize
814 \tw@make@key@box{shift}{%
815 \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
816 \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
817 (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
818 (0.3ex,1.2ex) -- cycle;
819 \end{tikzpicture}%
820 }
821 \tw@make@key@macro{\shift}

```

It's a little more complicated if the appearance should differ depending on the OS: The first step again is to define `\tw@mk@<key>@mac` and `\tw@mk@<key>@win`. And then use the starred version `\tw@make@key@macro*` which creates `\<key>` that depends on the `os` option, `\<key>@mac` and `\<key>@win`, that are not affected by `os`.

\capslock

```
822 \tw@make@key@box{capslock@mac}{%
823   \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
824     \draw (0.3ex,0.7ex) -- (1.1ex,0.7ex) -- (1.1ex,1.2ex) -- %
825           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
826           (0.3ex,1.2ex) -- cycle;
827     \draw (0.3ex,0) rectangle (1.1ex,0.4ex);
828   \end{tikzpicture}%
829 }
830 \tw@make@key@box{capslock@win}{%
831   \begin{tikzpicture}[yscale=-1,yshift=-1.8ex,menukeys key symbol]
832     \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
833           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
834           (0.3ex,1.2ex) -- cycle;
835   \end{tikzpicture}%
836 }
837 \tw@make@key@macro*{\capslock}
```

Here are the other macros:

\tab

```
838 \tw@make@key@box{tab@mac}{%
839   \begin{tikzpicture}[yshift=0.6ex,menukeys key symbol]
840     \draw [->] (0,0) -- (1em,0);
841     \draw (1em,-0.35ex) -- (1em,0.35ex);
842   \end{tikzpicture}%
843 }
844 \tw@make@key@box{tab@win}{%
845   \begin{tikzpicture}[yshift=0.1ex,menukeys key symbol]
846     \draw [->] (0.2em,0) -- (1.2em,0);
847     \draw (1.2em,-0.35ex) -- (1.2em,0.35ex);
848     \draw [<-] (0,1ex) -- (1em,1ex);
849     \draw (0,0.65ex) -- (0,1.35ex);
850   \end{tikzpicture}%
851 }
852 \tw@make@key@macro*{\tab}
```

\esc

\oldesc

```
853 \def\tw@mk@esc@win{Esc}
854 \tw@define@mackey{%
855   \def\tw@mk@esc@mac{esc}
856 }{%
857   \tw@make@key@box{esc@mac}{%
858     \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
859       \draw [->] (0.5ex,0.5ex) -- ++(135:1.1ex);
860       \draw (0.5ex,0.5ex) ++(105:0.6ex) arc (105:-195:0.6ex);
861     \end{tikzpicture}%
862   }%
863 }
864 \tw@make@key@macro*{\esc}
865 \def\tw@mk@oldesc@win{Esc}
866 \tw@define@mackey{%
867   \def\tw@mk@oldesc@mac{esc}
868 }{%
```

```

869 \tw@make@key@box{oldesc@mac}{%
870 \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
871 \draw [->] (0.5ex,0.5ex) -- ++(45:1.1ex);
872 \draw (0.5ex,0.5ex) ++(15:0.6ex) arc (15:-285:0.6ex);
873 \end{tikzpicture}%
874 }%
875 }
876 \tw@make@key@macro*{\oldesc}

\ctrl
877 \providecommand\ctrlname{Ctrl}
878 \def\tw@mk@ctrl@win{\ctrlname}
879 \def\tw@mk@ctrl@mac{ctrl}
880 \tw@make@key@macro*{\ctrl}

\Alt
\AltGr 881 \def\tw@mk@Alt@win{Alt}
882 \tw@define@mackey{%
883 \def\tw@mk@Alt@mac{alt}%
884 }%
885 \tw@make@key@box{Alt@mac}{%
886 \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
887 \draw (0,1ex) -- (0.5ex,1ex) -- (1ex,0.3ex) -- (1.8ex,0.3ex);
888 \draw (0.8ex,1ex) -- (1.8ex,1ex);
889 \end{tikzpicture}%
890 }%
891 }
892 \tw@make@key@macro*{\Alt}
893 \tw@make@simple@key@macro\AltGr{Alt\,Gr}

\cmd
894 \def\tw@mk@cmd@win{%
895 \tw@mk@warning{'\string\cmd' only for Mac!}%
896 }
897 \tw@define@mackey{%
898 \def\tw@mk@cmd@mac{cmd}%
899 }%
900 \tw@make@key@box{cmd@mac}{%
901 \begin{tikzpicture}[yshift=-0.15ex,menukeys key symbol]
902 \draw (0.5ex,0.7ex) -- (0.5ex,1.25ex) arc (0:270:0.25ex) -- %
903 (1.25ex,1ex) arc (-90:180:0.25ex) -- (1ex,0.25ex) %
904 arc (-180:90:0.25ex) -- (0.25ex,0.5ex) arc (90:360:0.25ex) %
905 -- cycle;
906 \end{tikzpicture}%
907 }%
908 }
909 \tw@make@key@macro*{\cmd}

\Space
\SPACE 910 \tw@make@simple@key@macro\Space{\rule{3em}{0pt}}
911 \newcommand{\spacename}{Space}
912 \tw@make@simple@key@macro\SPACE{\rule{2em}{0pt}\spacename\rule{2em}{0pt}}

```

```

\return
913 \tw@make@key@box{return@mac}{%
914   \begin{tikzpicture}[yshift=0.25ex,menukeys key symbol]
915     \draw [->, rounded corners=0.2ex] (1.25ex,1ex) -| %
916       (2ex,0) -- (0,0);
917   \end{tikzpicture}%
918 }
919 \tw@make@key@box{return@win}{%
920   \begin{tikzpicture}[menukeys key symbol]
921     \draw [->] (1ex,1.25ex) |- (0,0);
922   \end{tikzpicture}%
923 }
924 \tw@make@key@macro*{\return}

\enter
925 \def\tw@mk@enter@win{Enter}
926 \tw@make@key@box{enter@mac}{%
927   \begin{tikzpicture}[menukeys key symbol]
928     \draw (0,0) -- (0.5ex,0.5ex) -- (1ex,0);
929     \draw (0,0.55ex) -- (1ex,0.55ex);
930   \end{tikzpicture}%
931 }
932 \tw@make@key@macro*{\enter}

\winmenu
933 \def\tw@mk@winmenu@mac{%
934   \tw@mk@warning{'\string\winmenu' only for Windows!}%
935 }
936 \tw@make@key@box{winmenu@win}{%
937   \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
938     \draw (0,0) rectangle (1.5ex,1.8ex);
939     \draw (0.25ex,1.4ex) -- ++(1ex,0);
940     \draw (0.25ex,1ex) -- ++(1ex,0);
941     \draw (0.25ex,0.6ex) -- ++(1ex,0);
942   \end{tikzpicture}%
943 }
944 \tw@make@key@macro*{\winmenu}

\backspace
945 \tw@make@key@box{backspace}{%
946   \begin{tikzpicture}[yshift=0.65ex,menukeys key symbol]
947     \draw [<-,menukeys thick] (0,0) -- (1.35em,0);
948   \end{tikzpicture}%
949 }
950 \tw@make@key@macro*{\backspace}

\del
\backdel 951 \providecommand{\delname}{Del.}
952 \def\tw@mk@del@win{\delname}
953 \tw@define@mackey{%
954   \def\tw@mk@del@mac{\delname}%
955 }{%
956   \tw@make@key@box{del@mac}{%

```

```

957     \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
958     \draw (0,0) -- (1.5ex,0) -- (2ex,0.5ex) --%
959           (1.5ex,1ex) -- (0,1ex) -- cycle;
960     \draw (0.5ex,0.2ex) -- (1.1ex,0.8ex);
961     \draw (0.5ex,0.8ex) -- (1.1ex,0.2ex);
962     \end{tikzpicture}%
963 }%
964 }
965 \tw@make@key@macro*{\del}
966 \def\tw@mk@backdel@win{\delname}
967 \tw@define@mackey{%
968   \def\tw@mk@backdel@mac{\delname}%
969 }{%
970   \tw@make@key@box{backdel@mac}{%
971     \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
972     \draw (2ex,0) -- (0.5ex,0) -- (0,0.5ex) --%
973           (0.5ex,1ex) -- (2ex,1ex) -- cycle;
974     \draw (1ex,0.2ex) -- (1.6ex,0.8ex);
975     \draw (1ex,0.8ex) -- (1.6ex,0.2ex);
976     \end{tikzpicture}%
977   }%
978 }
979 \tw@make@key@macro*{\backdel}

\arrowkeyup Lastly we define the arrow macros:
\arrowkeydown 980 \tw@make@key@box{arrowkeyup}{%
\arrowkeyleft 981   \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
\arrowkeyright 982     \draw [->] (0,0) -- (0,0.8em);
983     \end{tikzpicture}%
984   }
985   \tw@make@key@macro{\arrowkeyup}
986
987   \tw@make@key@box{arrowkeydown}{%
988     \begin{tikzpicture}[yshift=0.7em,menukeys key symbol]
989     \draw [->] (0,0) -- (0,-0.8em);
990     \end{tikzpicture}%
991   }
992   \tw@make@key@macro{\arrowkeydown}
993
994   \tw@make@key@box{arrowkeyright}{%
995     \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
996     \draw [->] (0,0) -- (0.8em,0);
997     \end{tikzpicture}%
998   }
999   \tw@make@key@macro{\arrowkeyright}
1000
1001   \tw@make@key@box{arrowkeyleft}{%
1002     \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
1003     \draw [->] (0,0) -- (-0.8em,0);
1004     \end{tikzpicture}%
1005   }
1006   \tw@make@key@macro{\arrowkeyleft}

```

`\arrowkey` And the `\arrowkey` macro that get's it's direction as argument.

```

1007 \ExplSyntaxOn
1008 \cs_new_protected:Npn \__twmk_arrowkey_aux:nn #1#2
1009 {
1010   \cs_if_exist_use:cF
1011   {
1012     #1
1013     arrowkey
1014     \str_case:nnF {#2}
1015     {
1016       { ^ } { up }
1017       { v } { down }
1018       { < } { left }
1019       { > } { right }
1020     }
1021     { UNDEFINED }
1022   }
1023   {
1024     \tw@mk@error
1025     {
1026       Wrong~ value~ '#2'~ for~ \string\arrowkey.\MessageBreak
1027       Possible~ values~ are~ '^',~ 'v',~ '<'~ or~ '>'
1028     }
1029   }
1030 }
1031 \iftw@mk@defineshortnames
1032   \tw@mk@warn@if@defined{arrowkey}
1033   \providecommand\arrowkey{\__twmk_arrowkey_aux:nn{}}
1034 \fi
1035 \ifx\@empty\tw@mk@keysprefix
1036 \else
1037   \exp_args:Nc \newcommand { \tw@mk@keysprefix arrowkey }
1038   { \__twmk_arrowkey_aux:nn\tw@mk@keysprefix }
1039 \fi
1040 \ExplSyntaxOff

Close the \iftw@mk@definekeys
1041 \fi
1042 </pkg>

```

7 Change history

v1.0	theme (fix issue #16).	1
General: Initial version	<code>\backdel</code> : Added <code>\backdel</code>	36
v1.1	<code>\oldesc</code> : Fixed direction of <code>\escmac</code> ; added <code>\oldesc</code>	34
General: Improved manual		
Load <code>xcolor</code> before <code>menukeys</code>	v1.5	
<code>\directory</code> : Renamed <code>\path</code> to <code>\directory</code> because it crashes with <code>biblatex</code>	General: New option <code>hyperrefcolorlinks</code>	18
v1.1a	v1.6	
<code>\newmenumacro</code> : Added a line to make a new macro robust. . . .	General: <code>hyperrefcolorlinks</code> obsolete	18
<code>\tw@define@menu@macro@</code> : Fixed minor bug, that causes a warning about robustifying (issue #23), by deleting the line to make the command robust.	Don't load <code>catoptions</code>	15
v1.2	Load order no longer important	4
General: Added <code>\normalsize</code> before symbol definitions to make the <code>ex</code> unit available	<code>\newmenumacro</code> : use <code>\NewDocumentCommand</code>	31
Added <code>\SPACE</code> and <code>\spacename</code>	<code>\providenumacro</code> : use <code>\ProvideDocumentCommand</code>	31
Fixed GitHub issues #9, #10, #11, #13, #17, #24 and #26	<code>\renewmenumacro</code> : use <code>\RenewDocumentCommand</code>	31
Tidy up version and date	<code>\tw@define@menu@macro@</code> : Don't use <code>\NewDocumentCommand</code>	29
<code>\tw@define@menu@macro@</code> : Added <code>\leavevmode</code>	v1.6.1	
Replaced <code>\edef</code> by <code>\protected@edef</code>	<code>\newmenumacro</code> : default handled by <code>\tw@define@menu@macro</code>	31
v1.2a	<code>\providenumacro</code> : default handled by <code>\tw@define@menu@macro</code>	31
General: Added braces to the <code>\tikz</code> macro since the parser seems to crash with <code>babel</code> 's french option otherwise.	<code>\renewmenumacro</code> : default handled by <code>\tw@define@menu@macro</code>	31
Replaced obsolete <code>\tikzstyle</code>	<code>\tw@define@menu@macro</code> : Handles default input separator.	29
v1.2c	<code>\tw@define@menu@macro@</code> : No x-type expansion on the separator to call the loop	29
<code>\tw@define@menu@macro@</code> : Replaced <code>\protected@edef</code> by <code>\def</code>	Renamed from <code>\tw@define@menu@macro</code>	29
v1.3	v1.6.2	
General: Added TikZ-styles for the key symbols.	General: changed <code>\hspace</code> to <code>\kern</code>	26, 27
Improved key symbols.	v1.6.4	
v1.4	General: Added <code>defineshortnames</code> and <code>keysprefix</code> options	17
General: Extended color theme features.	Added setting arrow style to <code>To</code>	33
The <code>path...</code> styles now use the text color of the selected color	<code>\tw@make@key@macro</code> : Add <code>keysprefix</code> variant definitions	32
	<code>\tw@make@simple@key@macro</code> : Added <code>\tw@make@simple@key@macro</code>	32

8 Macro index

Numbers written in bold face refer to the page where the corresponding entry is described; italic numbers refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols		373, 376, 395, 400, 404, 407, 425, 430, 434, 437, 453, 477, 500, 523, 542, 558, 587, 630, 661
<code>\@ifdefinable</code>		634
<code>\@tw@hyphenatepaths@warnigfalse</code>	.	582
<code>\@tw@hyphenatepaths@warnigtrue</code>	.	570
A		D
<code>\Alt</code>		881
<code>\AltGr</code>		881
<code>angularkeys</code> (style)		6
<code>angularmenus</code> (style)		6
<code>\arrowkey</code>		14, 1007
<code>\arrowkeydown</code>		980
<code>\arrowkeyleft</code>		980
<code>\arrowkeyright</code>		980
<code>\arrowkeyup</code>		980
B		E
<code>\backdel</code>		951
<code>\backspace</code>		945
<code>blacknwhite</code> (theme)		12
<code>\bool</code>		62
		<code>\enter</code> 925
		<code>\esc</code> 853
		<code>\exp</code> 12, 50, 66, 1037
		<code>\ExpandArgs</code> 742, 750, 779, 787
		<code>\ExplSyntaxOff</code> 71, 1040
		<code>\ExplSyntaxOn</code> 10, 1007
C		F
<code>\capslock</code>		822
<code>\changemenucolor</code>		13, 133
<code>\changemenucolortheme</code>		11, 154
<code>\changemenuelement</code>		11, 323, 607, 613, 619, 625
<code>\cmd</code>		894
<code>\color</code>		336, 364, 369, 373, 376, 395, 400, 404, 407, 425, 430, 434, 437, 453, 477, 500, 523, 547, 549, 558, 586
Color themes:		
<code>blacknwhite</code>		12
<code>gray</code>		12
<code>\copymenucolortheme</code>		12, 133
<code>\copymenustyle</code>		11, 305, 606, 612, 618, 624
<code>\cs</code>		11, 12, 13, 15, 19, 26, 29, 33, 37, 43, 55, 56, 1008, 1010
<code>\csletcs</code>		310, 311, 312, 313, 314, 315, 316, 317
<code>\ctrl</code>		877
<code>\ctrlname</code>		14, 877, 878
<code>\CurrentMenuElement</code>		11, 191, 194, 197, 200, 223, 226, 229, 232, 336, 364, 369,
		<code>definekeys</code> (option) 4
		<code>definenumacros</code> (option) 4
		<code>defineshortnames</code> (option) 4, 14
		<code>\del</code> 951
		<code>\delname</code> ... 14, 951, 952, 954, 966, 968
		<code>\directory</code> 4, 725
		<code>\drawtikzfolder</code> 10, 597, 608, 614, 620, 626
		G
		<code>gray</code> (theme) 12
		<code>\group</code> 48, 51
		H
		<code>\hypersetup</code> 111
		<code>hyphenatepaths</code> (style) 8
		<code>hyphenatepathswithblackfolder</code> (style) 9
		<code>hyphenatepathswithfolder</code> (style) .. 8
		I
		<code>\if@tw@hyphenatepaths@warnig</code> 569, 579
		<code>\IfNoValueTF</code> 644
		<code>\iftw@mk@definekeys</code> 729
		<code>\iftw@mk@definenumacros</code> 724
		<code>\iftw@mk@defineshortnames</code> 741, 778, 1031
		<code>\iftw@mk@hyperrefcolorlinks</code> 108
		<code>\int</code> 24, 46, 52, 61, 64
		K
		<code>\keys</code> 4, 725
		<code>keysprefix</code> (option) 4, 14

L	
\l	21, 23, 24, 25, 26, 27, 28, 31, 35, 39, 40, 41, 45, 46, 49, 52, 60, 61, 62, 64, 65, 66
M	
mackeys (option)	4, 14
\menu	4, 725
menus (style)	5
N	
\newmenucolortheme .	12, 122, 169, 170
\newmenumacro ..	13, 705, 725, 726, 727
\newmenustyle	10, 236, 271
\newmenustylesimple	10, 236, 236
O	
\oldesc	853
Options:	
definekeys	4
definenumacros	4
defineshortnames	4, 14
keysprefix	4, 14
mackeys	4, 14
os	4
os (option)	4
P	
\PassOptionsToPackage	112
paths (style)	7
pathswithblackfolder (style)	8
pathswithfolder (style)	7
\prg	14
\protected	636
\providenumacro	13, 705
\providemenustyle	11, 236, 291
\providemenustylesimple ..	11, 236, 256
R	
\relsize	356, 387, 417, 448, 456, 468, 481, 495, 503, 514, 526, 538, 553
\renewmenucolortheme ...	13, 122, 151
\renewmenumacro	13, 705
\renewmenustyle	.. 11, 236, 280, 284, 301, 710, 721
\renewmenustylesimple	.. 11, 236, 245, 249, 266
\return	913
roundedkeys (style)	6
roundedmenus (style)	5
S	
\seq	21, 23, 27, 28, 31, 35, 39, 40, 41, 45, 49, 55, 60, 62, 65
shadowedangularkeys (style)	7
shadowedroundedkeys (style)	6
\shift	813
\SPACE	910
\Space	910
\spacename	14, 911, 912
\str	1014
Styles:	
angularkeys	6
angularmenus	6
hyphenatepathswithblackfolder .	9
hyphenatepathswithfolder	8
hyphenatepaths	8
menus	5
pathswithblackfolder	8
pathswithfolder	7
paths	7
roundedkeys	6
roundedmenus	5
shadowedangularkeys	7
shadowedroundedkeys	6
typewriterkeys	7
T	
\tab	838
\tikzset	343, 348, 379, 410, 440, 460, 488, 507, 533, 804
\tl	11, 14, 17, 25
\tw@current@color@theme ...	167, 684
\tw@current@style	206, 207, 212
\tw@declare@style .	203, 274, 276, 286, 288, 294, 296, 361, 392, 422
\tw@declare@style@extra@args ..	203
\tw@declare@style@simple ...	176, 239, 241, 251, 253, 259, 261, 450, 474, 497, 520, 540, 557, 571
\tw@default@input@sep	629, 645
\tw@default@post	171, 177, 204
\tw@default@pre	171, 177, 210
\tw@default@sep	171, 177, 210
\tw@define@mackey	.. 799, 854, 866, 882, 897, 953, 967
\tw@define@menu@macro	.. 631, 707, 714, 718
\tw@define@menu@macro@	631
\tw@make@color@theme ..	114, 124, 131
\tw@make@key@box	730, 814, 822, 830, 838, 844, 857, 869, 885, 900, 913, 919, 926, 936, 945, 956, 970, 980, 987, 994, 1001
\tw@make@key@macro	.. 740, 821, 837, 852, 864, 876, 880, 892, 909, 924, 932, 944, 950, 965, 979, 985, 992, 999, 1006
\tw@make@key@macro@aux	740

<code>\tw@make@simple@key@macro</code>	<code>\tw@mk@oldesc@win</code>	865
..... 740, 893, 910, 912	<code>\tw@mk@eos</code>	100, 763
<code>\tw@make@simple@key@macro@aux</code> .	<code>\tw@mk@estring</code> 13, 659, 697, 706, 717,	
740	743, 744, 751, 752, 774, 780, 788	
<code>\tw@menu@list</code>	<code>\tw@mk@symbol</code>	740
689, 693, 698	<code>\tw@mk@tempa</code> .	81, 84, 85, 690, 694, 698
<code>\tw@mk@Alt@mac</code>	<code>\tw@mk@tempb</code>	81, 697, 698
883	<code>\tw@mk@test@input@sep</code> .	634, 636, 687
<code>\tw@mk@Alt@win</code>	<code>\tw@mk@trimspaces</code>	11, 638, 694
881	<code>\tw@mk@warn@if@defined</code> ...	740, 1032
<code>\tw@mk@backdel@mac</code>	<code>\tw@mk@warning</code>	72, 109,
968	264, 299, 580, 720, 774, 895, 934	
<code>\tw@mk@backdel@win</code>	<code>\tw@mk@warning@noline</code>	72
966	<code>\tw@mk@winmenu@mac</code>	933
<code>\tw@mk@cmd@mac</code>	<code>\tw@mk@xifinsetTF</code>	15, 637
898	<code>\tw@set@tikz@colors</code>	343
<code>\tw@mk@cmd@win</code>	<code>\tw@typewriterkeys@curr@elem</code> ...	
894	 541, 547, 549	
<code>\tw@mk@ctrl@mac</code>	<code>typewriterkeys (style)</code>	7
879		
<code>\tw@mk@ctrl@win</code>		
878		
<code>\tw@mk@del@mac</code>		
954		
<code>\tw@mk@del@win</code>		
952		
<code>\tw@mk@enter@win</code>		
925		
<code>\tw@mk@error</code> 72, 101, 105, 126,		
137, 150, 156, 160, 244, 279,		
308, 320, 325, 339, 652, 709, 1024		
<code>\tw@mk@esc@mac</code>		
855		
<code>\tw@mk@esc@win</code>		
853		
<code>\tw@mk@exp@Nnno</code>		
12, 645		
<code>\tw@mk@gobble@cargs</code>		
..... 83, 246, 267, 281, 302		
<code>\tw@mk@iflastindris</code>		
19, 662		
<code>\tw@mk@indrisloop</code>		
56, 698		
<code>\tw@mk@indrisnr</code>		
26, 664, 672		
<code>\tw@mk@keys@prefix</code>		
748, 751, 785, 788, 1035, 1037, 1038		
<code>\tw@mk@mackeys</code>		
104, 800, 801		
<code>\tw@mk@oldesc@mac</code>		
867		

U

<code>\usemenucolor</code>	11, 166,
336, 344, 345, 346, 364, 369,	
373, 376, 395, 400, 404, 407,	
425, 430, 434, 437, 453, 456,	
471, 477, 481, 500, 503, 517,	
523, 526, 547, 549, 553, 558,	
562, 586, 591, 608, 614, 620, 626	

W

<code>\winmenu</code>	933
-----------------------------	-----