

MENUKEYS

Tobias Weh

`mail@tobiw.de`

`http://tobiw.de/en`

`http://github.com/tweh/menukeys`

`http://www.ctan.org/pkg/menukeys`

`└─ macros ─ latex ─ contrib ─ menukeys`

2020/10/31 — v1.6

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 <http://tex.stackexchange.com/a/44989/4918>.

Special thanks to Jonathan P. Spratte, who made the changes in v1.6 to remove the `catoptions` dependency so that `menukeys` works with `LaTeX` releases starting from 2020/10/01.

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	9
4.2.3	Copying styles	10
4.2.4	Changing styles	10
4.3	Color themes	11
4.3.1	Predefined themes	11
4.3.2	Create a theme	11
4.3.3	Copy a theme	12
4.3.4	Change a theme	12
4.4	Menu macros	12
4.4.1	Predefined menu macros	12
4.4.2	Defining or changing menu macros	12
4.5	Keys	13
5	Known issues and bugs	14
6	Implementation	14
6.1	Required packages	14
6.2	Helper macros	16
6.3	Options	17
6.4	Workarounds	17
6.4.1	hyperref's colorlinks option	17
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	19
6.6.1	Internal commands	20
6.6.2	User-level commands	21
6.6.3	Copying and changing	22
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	39

1 Introduction

The `menukeys` package is mainly designed to parse and print sequences of software menus, folders and files or keystrokes. The 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 Extras >> Settings >> Rulers and choose between
millimeters, inches and pixels. The shortcut to view the rulers is cmd + R.
Pressing these keys again will hide the rulers.
```

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

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 <http://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: 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`.

`definenumacros` (opt.)

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

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

`definekeys` (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`.

`mackeys` (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`.

`os` (opt.)

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 sequence>}`, with `>` as default input list separator, `\directory{<path and files>}` with `/` as default separator and `\keys{<keystrokes>}` with `+` as default separator. You've seen examples for all of them in section 1.

`\menu`
`\directory`
`\keys`

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[,]{<menu sequence>}`.⁴


The possible input separators are `/`, `=`, `*`, `+`, `,`, `;`, `:`, `-`, `>`, `<` and `bslash` (to use `\` as separator). You can hide a separator from the parser by putting a

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

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

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

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



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.

4.2.1 Predefined styles


Name: `menu`



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 `menu`. And again some blind text without any sense.


Name: `roundedmenu`



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 `roundedmenu`. And again some blind text without any sense.

Name: `angularmenu`



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 `angularmenu`. And again some blind text without any sense.

Name: **roundedkeys**

Ctrl + Alt + Q

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This Ctrl + Alt + Q 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.

Name: **shadowedroundedkeys**

Ctrl + Alt + Q

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This Ctrl + Alt + Q 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**

Ctrl + Alt + Q

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This Ctrl + Alt + Q 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**

Ctrl + Alt + Q

S

This is some more or less blind text, to demonstrate how the sequence looks in text. This Ctrl + Alt + Q 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`

Ⓢ + Ⓚ

Ⓢ

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`

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.

Name: `pathswithfolder`

☐ 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`

▣ 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 `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

C: › Database › User › ALongUserNameHere › ALongerFolderNameAtThisPlace › 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 hyphenatepaths. And again some blind text without any sense.

The sep color is taken from optional color C.

Name: hyphenatepathswithfolder

☐ C: › Database › User › ALongUserNameHere › ALongerFolderNameAtThisPlace › 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 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: hyphenatepathswithblackfolder

▣ C: › Database › User › ALongUserNameHere › ALongerFolderNameAtThisPlace › 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 *(first)*, *(mid)*, *(last)*, *(single)* 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}}[$\ast$]
```

`\usemenucolor` 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

`\changemenuelement` 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
`\providemenustylesimple` 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: `blackwhite`

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>}{
}{<txt>}[<a>][][<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`, `cmyk`, ...

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` $\{\langle copy \rangle\}\{\langle original \rangle\}$.

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` $\{\langle name \rangle\}\{\langle element \rangle\}\{\langle model \rangle\}\{\langle color\ definition \rangle\}$, 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` $\{\langle macro \rangle\} [\langle input\ sep \rangle] \{\langle style \rangle\}$.

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 `\langle macro \rangle [\langle input\ sep \rangle] \{\langle list \rangle\}`

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.

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

Table 1: Overview of all key macros.

Macro	Mac	Win.	Macro	Mac	Win.
<code>\shift</code>	⇧	⇧	<code>\winmenu</code>		☰
<code>\capslock</code>	⇨	⇩	<code>\backspace</code>	←	←
<code>\tab</code>	→	↔	<code>\del</code>	Del. / ☒	Del.
<code>\esc</code>	esc / ⌘	Esc	<code>\backdel</code>	Del. / ☓	Del.
<code>\oldesc</code>	esc / ⌘	Esc	<code>\arrowkey{^}</code>	↑	↑
<code>\ctrl</code>	ctrl	Ctrl	<code>\arrowkeyup</code>	↑	↑
<code>\Alt</code>	alt / ⌥	Alt	<code>\arrowkey{v}</code>	↓	↓
<code>\AltGr</code>		Alt Gr	<code>\arrowkeydown</code>	↓	↓
<code>\cmd</code>	cmd / ⌘		<code>\arrowkey{>}</code>	→	→
<code>\Space</code>	[empty sp.]	[empty sp.]	<code>\arrowkeyright</code>	→	→
<code>\SPACE</code>	Space	Space	<code>\arrowkey{<}</code>	←	←
<code>\return</code>	↵	↵	<code>\arrowkeyleft</code>	←	←
<code>\enter</code>	↵	Enter			

`\arrowkey` The macro `\arrowkey{<direction>}` is a little special since it takes the direction as a single character `^`, `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 `cmym` model:

```
\documentclass{article}
\PassOptionsToPackage{cmym}{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

6.1 Required packages

Load the required packages

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

Furthermore we need `TikZ` and some of its libraries,

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

the color package `xcolor` and `adjustbox` for the `typewriterkeys` style.

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

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

```
8 \RequirePackage{relsize}
```

To define the list parsing commands and allow `\` 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 `\tw@mk@xifinsetTF` as it requires the arguments to get swapped.

```

9 \ExplSyntaxOn
10 \cs_new_eq:NN \tw@mk@trimspaces \tl_trim_spaces:n
11 \cs_new_eq:NN \tw@mk@expanded \use:x
12 \prg_generate_conditional_variant:Nnn \tl_if_in:nn { xx } { TF }
13 \cs_new:Npn \tw@mk@xifinsetTF #1 #2
14 {
15   \tl_if_in:xxTF {#2} {#1}
16 }

```

The replacement for `\indrisloop` will not set the conditional `\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 `\iflastindris...\else...\fi` construct but instead two branches.

```

17 \cs_new:Npn \tw@mk@iflastindris
18 {
19   \seq_if_empty:NTF \l__twmk_indris_seq
20 }

```

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.

```

21 \seq_new:N \l__twmk_indris_seq
22 \int_new:N \l__twmk_indris_int
23 \tl_new:N \l__twmk_indris_tl
24 \cs_new_eq:NN \tw@mk@indrisnr \l__twmk_indris_int
25 \seq_new:N \l__twmk_indris_seqstack_seq
26 \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.

```

27 \cs_new_protected:Npn \__twmk_pushseq:
28 {
29   \seq_push:No \l__twmk_indris_seqstack_seq \l__twmk_indris_seq
30 }
31 \cs_new_protected:Npn \__twmk_pushint:
32 {
33   \seq_push:NV \l__twmk_indris_intstack_seq \l__twmk_indris_int
34 }
35 \cs_new_protected:Npn \__twmk_popseq:

```



```

36 {
37   \seq_if_empty:NTF \l__twmk_indris_seqstack_seq
38     { \seq_clear:N \l__twmk_indris_seq }
39     { \seq_pop:NN \l__twmk_indris_seqstack_seq \l__twmk_indris_seq }
40 }
41 \cs_new_protected:Npn \__twmk_popint:
42 {
43   \seq_if_empty:NTF \l__twmk_indris_intstack_seq
44     { \int_zero:N \l__twmk_indris_int }
45     {
46       \group_begin:
47         \seq_pop:NN \l__twmk_indris_intstack_seq \l__twmk_indris_tl
48         \exp_args:NNNo
49         \group_end:
50         \int_set:Nn \l__twmk_indris_int \l__twmk_indris_tl
51     }
52 }

```

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.

```

53 \cs_generate_variant:Nn \seq_set_split:Nnn { Nno }
54 \cs_new_protected:Npn \tw@mk@indrisloop #1 #2 #3
55 {
56   \__twmk_pushseq:
57   \__twmk_pushint:
58   \seq_set_split:Nno \l__twmk_indris_seq {#1} {#2}
59   \int_zero:N \l__twmk_indris_int
60   \bool_do_while:nn { \bool_not_p:n { \seq_if_empty_p:N \l__twmk_indris_seq } }
61   {
62     \int_incr:N \l__twmk_indris_int
63     \seq_pop_left:NN \l__twmk_indris_seq \l__twmk_indris_tl
64     \exp_args:No #3 \l__twmk_indris_tl
65   }
66   \__twmk_popseq:
67   \__twmk_popint:
68 }
69 \ExplSyntaxOff

```

6.2 Helper macros

```

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

```

```

75 }
76 \newcommand*{\tw@mk@warning@noline}[1]{%
77   \PackageWarningNoLine{menukeys}{#1}%
78 }

```

`\tw@mk@tempa` Some commands for temporary use:

```

\tw@mk@tempb 79 \def\tw@mk@tempa{}
              80 \def\tw@mk@tempb{}

```

`\tw@mk@gobble@args` Define a command to gobble arguments.

```

81 \DeclareDocumentCommand{\tw@mk@gobble@args}{m}{%
82   \RenewDocumentCommand{\tw@mk@tempa}{#1}{}%
83   \tw@mk@tempa%
84 }

```

6.3 Options

First we declare and process the package options

```

85 \RequirePackage{kvoptions}
86 \SetupKeyvalOptions{
87   family=tw@mk,
88   prefix=tw@mk@
89 }
90 \DeclareBoolOption[true]{definenumacros}
91 \DeclareBoolOption[true]{definekeys}
92 \DeclareBoolOption[false]{hyperrefcolorlinks}
93 \DeclareStringOption[mac]{os}
94 \DeclareStringOption[symbols]{mackeys}
95 \ProcessKeyvalOptions{tw@mk}\relax

```

Now we have to do some error treatment:

```

96 \IfSubStr{.mac.win.}{.\tw@mk@os.}{}{%
97   \tw@mk@error{Unknown value for option 'os'\MessageBreak
98   Possible values are 'mac' or 'win'.}%
99 }
100 \IfSubStr{.symbols.text.}{.\tw@mk@mackeys.}{}{%
101   \tw@mk@error{Unknown value for option 'mackeys'\MessageBreak
102   Possible values are 'symbols' or 'text'.}%
103 }

```

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.

```
104 \iftw@mk@hyperrefcolorlinks
105   \tw@mk@warning{The option 'hyperrefcolorlinks' is obsolete}
106   \@ifpackageloaded{hyperref}
107     {\hypersetup{colorlinks}}
108     {\PassOptionsToPackage{colorlinks}{hyperref}}
109 \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

```
110 \newcommand*{\tw@make@color@theme}[8]{%
111   \definecolor{tw@color@theme@#1@bg}{#2}{#3}%
112   \definecolor{tw@color@theme@#1@br}{#2}{#4}%
113   \definecolor{tw@color@theme@#1@txt}{#2}{#5}%
114   \definecolor{tw@color@theme@#1@a}{#2}{#6}%
115   \definecolor{tw@color@theme@#1@b}{#2}{#7}%
116   \definecolor{tw@color@theme@#1@c}{#2}{#8}%
117 }
```

6.5.2 User-level commands

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

```
\renewmenucolortheme 118 \NewDocumentCommand{\newmenucolortheme}{ m m m m m O{#3} O{#4} O{#5} }{%
119   \@ifundefinedcolor{tw@color@theme@#1@bg}{%
120     \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
121   }{%
122     \tw@mk@error{Color theme '#1' already defined!\MessageBreak
123       Use \string\renewmenucolortheme\space instead.}%
124   }
125 }
126 \NewDocumentCommand{\renewmenucolortheme}{ m m m m m O{#3} O{#4} O{#5} }{%
127   \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
128 }
```

`\changemenucolor` Lastly we define the changing and copying commands

```
\copymenucolortheme 129 \newcommand*{\changemenucolor}[4]{%
130   \IfSubStr{ bg br txt }{ #2 }{%
131     \definecolor{tw@color@theme@#1@#2}{#3}{#4}%
132   }{%
133     \tw@mk@error{No such color element ('#2')!\MessageBreak
134       Possible values are bg, br and txt.}
135   }%
136 }
137 \newcommand*{\copymenucolortheme}[2]{%
```

```

138 \@ifundefinedcolor{tw@color@theme@#1@bg}{%
139   \colorlet{tw@color@theme@#1@bg}{tw@color@theme@#2@bg}%
140   \colorlet{tw@color@theme@#1@br}{tw@color@theme@#2@br}%
141   \colorlet{tw@color@theme@#1@txt}{tw@color@theme@#2@txt}%
142   \colorlet{tw@color@theme@#1@a}{tw@color@theme@#2@a}%
143   \colorlet{tw@color@theme@#1@b}{tw@color@theme@#2@b}%
144   \colorlet{tw@color@theme@#1@c}{tw@color@theme@#2@c}%
145 }{%
146   \tw@mk@error{Color theme '#1' already defined!\MessageBreak
147   Use \string\renewmenucolortheme\space instead.}
148 }
149 }

```

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

```

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

```

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

```

162 \newcommand{\usemenucolor}[1]{%
163   tw@color@theme@\tw@current@color@theme @#1%
164 }

```

6.5.3 Predefined themes

There are two predefined color themes

```

165 \newmenucolortheme{gray}{gray}{0.95}{0.3}{0}{0.95}[0][0]
166 \newmenucolortheme{blackwhite}{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 commands, we have to set some defaults for the optional arguments

```

167 \newcommand{\tw@default@sep}{%
168   \hspace{0.2em plus 0.1em minus 0.5em}%
169 }
170 \newcommand{\tw@default@pre}{}
171 \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.

```
172 \DeclareDocumentCommand{\tw@declare@style@simple}{%
173   s m 0{\tw@default@pre} m 0{\tw@default@sep} 0{\tw@default@post} m
174 }{%
175   \csdef{tw@style@#2@color@theme}{#7}%
176   \csdef{tw@style@#2@pre}{#3}%
177   \csdef{tw@style@#2@sep}{#5}%
178   \csdef{tw@style@#2@post}{#6}%
179   \IfBooleanTF{#1}{%
180     \csdef{tw@style@#2@single}{#4}%
181     \csdef{tw@style@#2@first}{#4}%
182     \csdef{tw@style@#2@mid}{#4}%
183     \csdef{tw@style@#2@last}{#4}%
184   }{%
185     \csdef{tw@style@#2@single}{%
186       \tikz[baseline=(tw@node.base)]{%
187         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
188     \csdef{tw@style@#2@first}{%
189       \tikz[baseline=(tw@node.base)]{%
190         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
191     \csdef{tw@style@#2@mid}{%
192       \tikz[baseline=(tw@node.base)]{%
193         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
194     \csdef{tw@style@#2@last}{%
195       \tikz[baseline=(tw@node.base)]{%
196         \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
197     }%
198 }
```

`\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.

```
199 \DeclareDocumentCommand{\tw@declare@style@extra@args}{%
200   0{\tw@default@post} m
201 }{%
202   \csdef{tw@style@\tw@current@style @post}{#1}%
203   \csdef{tw@style@\tw@current@style @color@theme}{#2}%
204 }
```

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

```
205 \DeclareDocumentCommand{\tw@declare@style}{%
206   s m 0{\tw@default@pre} m 0{\tw@default@sep} m m m
207 }{%
208   \def\tw@current@style{#2}
209   \csdef{tw@style@#2@pre}{#3}%
210   \csdef{tw@style@#2@sep}{#5}%

```

```

211 \IfBooleanTF{#1}{%
212   \csdef{tw@style@#2@single}{#8}%
213   \csdef{tw@style@#2@first}{#4}%
214   \csdef{tw@style@#2@mid}{#6}%
215   \csdef{tw@style@#2@last}{#7}%
216 }{%
217   \csdef{tw@style@#2@single}{%
218     \tikz[baseline=(tw@node.base)]{%
219       \node(tw@node)[#8]{\strut\CurrentMenuElement};}%
220   \csdef{tw@style@#2@first}{%
221     \tikz[baseline=(tw@node.base)]{%
222       \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
223   \csdef{tw@style@#2@mid}{%
224     \tikz[baseline=(tw@node.base)]{%
225       \node(tw@node)[#6]{\strut\CurrentMenuElement};}%
226   \csdef{tw@style@#2@last}{%
227     \tikz[baseline=(tw@node.base)]{%
228       \node(tw@node)[#7]{\strut\CurrentMenuElement};}%
229   }%
230 \tw@declare@style@extra@args%
231 }

```

6.6.2 User-level commands

```

newmenustylesimple It's time to define the user-level commands now:
renewmenustylesimple 232 \NewDocumentCommand{\newmenustylesimple}{s m}{%
providemenustylesimple 233   \ifcsundef{tw@style@#2@pre}{%
newmenustyle 234     \IfBooleanTF{#1}{%
renewmenustyle 235       \tw@declare@style@simple*{#2}%
providemenustyle 236     }{%
237       \tw@declare@style@simple{#2}%
238     }%
239   }{%
240     \tw@mk@error{Style '#2' already defined!\MessageBreak
241       Use \string\renewmenustylesimple\space instead.}%
242     \tw@mk@gobble@args{o m o m}%
243   }%
244 }
245 \NewDocumentCommand{\renewmenustylesimple}{s m}{%
246   \IfBooleanTF{#1}{%
247     \tw@declare@style@simple*{#2}%
248   }{%
249     \tw@declare@style@simple{#2}%
250   }%
251 }
252 \NewDocumentCommand{\providemenustylesimple}{s m}{%
253   \ifcsundef{tw@style@#2@pre}{%
254     \IfBooleanTF{#1}{%
255       \tw@declare@style@simple*{#2}%

```

```

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

```

6.6.3 Copying and changing

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

```

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

```

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

```

319 \NewDocumentCommand{\changemenuelement}{s m m m}{%
320   \ifcsundef{tw@style@#2@pre}{%
321     \tw@mk@error{Style '#2' undefined.}%
322   }{%
323     \IfSubStr{ single first middle last pre post sep }{ #3 }{%
324       \IfBooleanTF{#1}{%
325         \csdef{tw@style@#2@#3}{#4}%
326       }{%
327         \IfSubStr{ pre post sep }{ #3 }{%
328           \csdef{tw@style@#2@#3}{#4}%
329         }{%
330           \csdef{tw@style@#2@#3}{%
331             \tikz[baseline=(tw@node.base)]{%
332               \node(tw@node)[#4]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
333           }%
334         }%
335       }{\tw@mk@error{No element '#3'. Possible values are\MessageBreak
336         single, first, middle, last, pre, post or sep.}}%
337     }%
338 }

```

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

```

339 \tikzset{tw@set@tikz@colors/.style={%
340   draw=\usemenucolor{br},
341   fill=\usemenucolor{bg},
342   text=\usemenucolor{txt},

```


343 }}

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.

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

```

390     \node(tw@node)[tw@roundedmenus@base,signal to=east]%
391     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
392 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
393 {%
394   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
395     \node(tw@node)[tw@roundedmenus@base,signal from=west,signal to=east]%
396     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
397 }{%
398   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
399     \node(tw@node)[tw@roundedmenus@base,signal from=west,]%
400     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
401 }{%
402   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
403     \node(tw@node)[tw@roundedmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement.
404 }{gray}
405
406 \tikzset{tw@angularmenus@base/.style={%
407   tw@set@tikz@colors,
408   inner sep=0pt,
409   inner xsep=2pt,
410   text height=1.825ex,
411   text depth=0.7ex,
412   minimum width=1.5em,
413   font=\relsize{-1}\sffamily,
414   signal,
415   signal to=nowhere,
416   signal pointer angle=110,
417 }}
418 \tw@declare@style*{angularmenus}{%
419   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
420     \node(tw@node)[tw@angularmenus@base,signal to=east]%
421     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
422 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
423 {%
424   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
425     \node(tw@node)[tw@angularmenus@base,signal from=west,signal to=east]%
426     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
427 }{%
428   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
429     \node(tw@node)[tw@angularmenus@base,signal from=west,]%
430     {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
431 }{%
432   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
433     \node(tw@node)[tw@angularmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement.
434 }{gray}
435
436 \tikzset{tw@roundedkeys@base/.style={%
437   tw@set@tikz@colors,
438   rounded corners=0.3ex,
439   inner sep=0pt,

```

```

440   inner xsep=2pt,
441   text height=1.825ex,
442   text depth=0.7ex,
443   minimum width=1.5em,
444   font=\relsize{-1}\sffamily,
445 }}
446 \tw@declare@style@simple*{roundedkeys}{%
447   \tikz[baseline={($(\tw@node.base)+(0,-0.2ex)$)}]{%
448     \node(\tw@node)[tw@roundedkeys@base]%
449       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
450 }[%
451   \hspace{0.1em plus 0.1em minus 0.05em}%
452   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
453   \hspace{0.1em plus 0.1em minus 0.05em}%
454 ]{gray}
455
456 \tikzset{tw@shadowedroundedkeys@base/.style={%
457   tw@set@tikz@colors,
458   rounded corners=0.3ex,
459   inner sep=0pt,
460   inner xsep=2pt,
461   text height=1.825ex,
462   text depth=0.7ex,
463   minimum width=1.5em,
464   font=\relsize{-1}\sffamily,
465   general shadow={%
466     shadow xshift=.2ex, shadow yshift=-.15ex,
467     fill=\usemenucolor{c},
468   },
469 }}
470 \tw@declare@style@simple*{shadowedroundedkeys}{%
471   \tikz[baseline={($(\tw@node.base)+(0,-0.2ex)$)}]{%
472     \node(\tw@node)[tw@shadowedroundedkeys@base]%
473       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
474   }%
475 }[%
476   \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
477   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
478   \hspace{0.1em plus 0.1em minus 0.05em}%
479 ][\hspace{0.2ex}]{gray}
480
481 \tikzset{tw@angularkeys@base/.style={%
482   tw@set@tikz@colors,
483   inner sep=0pt,
484   inner xsep=2pt,
485   text height=1.825ex,
486   text depth=0.7ex,
487   minimum width=1.5em,
488   font=\relsize{-1}\sffamily,
489 }}

```

```

490 \tw@declare@style@simple*{angularkeys}{%
491   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
492     \node(tw@node)[tw@angularkeys@base]%
493       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
494 }[%
495   \hspace{0.1em plus 0.1em minus 0.05em}%
496   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
497   \hspace{0.1em plus 0.1em minus 0.05em}%
498 ]{gray}
499
500 \tikzset{tw@shadowedangularkeys@base/.style={%
501   tw@set@tikz@colors,
502   inner sep=0pt,
503   inner xsep=2pt,
504   text height=1.825ex,
505   text depth=0.7ex,
506   minimum width=1.5em,
507   font=\relsize{-1}\sffamily,
508   general shadow={%
509     shadow xshift=.2ex, shadow yshift=-.15ex,
510     fill=\usemenucolor{c},
511   },
512 }}
513 \tw@declare@style@simple*{shadowedangularkeys}{%
514   \tikz[baseline={{$(tw@node.base)+(0,-0.2ex)$}}]{%
515     \node(tw@node)[tw@shadowedangularkeys@base]%
516       {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
517 }[%
518   \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
519   \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
520   \hspace{0.1em plus 0.1em minus 0.05em}%
521 ][\hspace{0.2ex}]{gray}
522
523 \tikzset{tw@typewriterkeys@base/.style={%
524   tw@set@tikz@colors,
525   shape=circle,
526   minimum size=2ex,
527   inner sep=0.5pt, outer sep=1pt,
528   font=\ttfamily\relsize{-1},
529 }}
530 \tw@declare@style@simple*{typewriterkeys}{%
531   \def\tw@typewriterkeys@curr@elem{%
532     \maxsizebox*{2ex}{2ex}{\CurrentMenuElement}%
533   }%
534   \begin{tikzpicture}[baseline={{$(tw@node.south)+(0,0.8ex)$}}]{%
535     \node(tw@node)[%
536       tw@typewriterkeys@base, inner sep=1.25pt, line width=0.6pt%
537     ]{\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
538     \node[tw@typewriterkeys@base]%
539       {\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};

```

```

540 \end{tikzpicture}%
541 }[%
542 \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
543 \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}}}%
544 \hspace{0.1em plus 0.1em minus 0.05em}%
545 ]{blacknwhite}
546
547 \tw@declare@style@simple*{paths}{%
548 {\ttfamily\color{\usemenucolor{txt}}\CurrentMenuElement}%
549 }[%
550 \hspace{0.2em plus 0.1em}%
551 \raisebox{0.08ex}{%
552 \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
553 -- (0,1ex) -- cycle;}%
554 }%
555 \hspace{0.2em plus 0.1em}%
556 ]{blacknwhite}
557
558 \newcounter{tw@hyphen@char@num}
559 \newif\if@tw@hyphenatepaths@warnig
560 \@tw@hyphenatepaths@warnigtrue
561 \tw@declare@style@simple*{hyphenatepaths}{%
562 {\ttfamily
563 \IfStrEq{T1}{\encodingdefault}{%
564 \setcounter{tw@hyphen@char@num}{23}%
565 }{%
566 \IfStrEq{OT1}{\encodingdefault}{%
567 \setcounter{tw@hyphen@char@num}{255}%
568 }{%
569 \if@tw@hyphenatepaths@warnig%
570 \tw@mk@warning{The hyphenatepaths styles will probably only\MessageBreak
571 work with T1 or OT1 encoding.}%
572 \fi\global\@tw@hyphenatepaths@warnigfalse%
573 }%
574 }%
575 \hyphenchar\font=\value{tw@hyphen@char@num}\relax
576 \color{\usemenucolor{txt}}}%
577 \CurrentMenuElement}%
578 }[%
579 \hspace{0.2em plus 0.1em}%
580 \raisebox{0.08ex}{%
581 \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
582 -- (0,1ex) -- cycle;}%
583 }%
584 \hspace{0.2em plus 0.1em}%
585 ]{blacknwhite}
586
587 \NewDocumentCommand{\drawtikzfolder}{0{white} 0{black}}{%
588 \begin{tikzpicture}[rounded corners=0.02ex,scale=0.7]
589 \draw [#2] (0,0) -- (1em,0) -- (1em,1.5ex) -- (0.5em,1.5ex) -- %

```

```

590         (0.4em,1.7ex) -- (0.1em,1.7ex) -- (0,1.5ex) -- cycle;
591     \draw [#2,fill=#1] (0,0) -- (1em,0) -- (0.85em,1.15ex) -- %
592         ++(-1em,0) -- cycle;
593 \end{tikzpicture}%
594 }
595
596 \copymenustyle{pathswithfolder}{paths}
597 \changemenuelement{pathswithfolder}{pre}{%
598     \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
599     \hspace{0.2em plus 0.1em}%
600 }
601
602 \copymenustyle{pathswithblackfolder}{paths}
603 \changemenuelement{pathswithblackfolder}{pre}{%
604     \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
605     \hspace{0.2em plus 0.1em}%
606 }
607
608 \copymenustyle{hyphenatepathswithfolder}{hyphenatepaths}
609 \changemenuelement{hyphenatepathswithfolder}{pre}{%
610     \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
611     \hspace{0.2em plus 0.1em}%
612 }
613
614 \copymenustyle{hyphenatepathswithblackfolder}{hyphenatepaths}
615 \changemenuelement{hyphenatepathswithblackfolder}{pre}{%
616     \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
617     \hspace{0.2em plus 0.1em}%
618 }

```

6.7 Menu macros

6.7.1 Internal commands

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

```
619 \edef\tw@default@input@sep{,}
```

`\CurrentMenuElement` and the `\CurrentMenuElement` dummy

```
620 \def\CurrentMenuElement{}
```

`\tw@define@menu@macro` Then we set up the internal command to create new menu macros. The list parsing code was essentially provided by Ahmed Musa at <http://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!

```

621 \begingroup
622 \lccode'\,=1
623 \lowercase{\endgroup
624     \@ifdefinable\tw@mk@test@input@sep

```

```

625  {%
626    \protected\def\tw@mk@test@input@sep#1{%
627      \tw@mk@xifinsetTF
628      {,\tw@mk@trimspaces{#1},}{,bslash,backslash,directory,location,}%
629    }%
630  }%
631 }
632 \newcommand\tw@define@menu@macro[4]{%
633   \ifcsundef{tw@style@#4@sep}{%
634     \tw@mk@error{Can't define menu macro \string#2\space,\MessageBreak
635     because the style '#4' is not available!}
636   }{%
637     \csdef{tw@parse@menu@list@\expandafter\@gobble\string#2}##1{%
638       \tw@mk@iflastindris
639       {%
640         \ifnum\tw@mk@indrisnr=\@ne
641           \def\CurrentMenuElement{##1}%
642           \@nameuse{tw@style@#4@single}%
643         \else
644           \def\CurrentMenuElement{##1}%
645           \@nameuse{tw@style@#4@sep}\@nameuse{tw@style@#4@last}%
646         \fi
647       }
648     {%
649       \ifnum\tw@mk@indrisnr=\@ne
650         \def\CurrentMenuElement{##1}%
651         \@nameuse{tw@style@#4@first}%
652       \else
653         \def\CurrentMenuElement{##1}%
654         \@nameuse{tw@style@#4@sep}\@nameuse{tw@style@#4@mid}%
655       \fi
656     }%
657   }%
658   #1 #2 { +0{#3} +m }{%
659     \leavevmode%
660     {\def\tw@current@color@theme{\csname tw@style@#4@color@theme\endcsname}%
661     \@nameuse{tw@style@#4@pre}%
662     \tw@mk@test@input@sep{##1}{%
663       \edef\tw@menu@list{\detokenize{##2}}\edef\tw@mk@tempa{\@backslashchar}%
664     }{%
665       \edef\tw@menu@list{\unexpanded{##2}}\edef\tw@mk@tempa{\tw@mk@trimspaces{##1}}%
666     }%
667     {\letcs{tw@mk@tempb}{tw@parse@menu@list@\expandafter\@gobble\string#2}%
668     \tw@mk@expanded{tw@mk@indrisloop{tw@mk@tempa}}\tw@menu@list\tw@mk@tempb}%
669     \@nameuse{tw@style@#4@post}}%
670   }%
671 }%
672 }

```

6.7.2 User-level commands

```
\newmenumacro Now it's time to build the user-level commands
\renewmenumacro 673 \NewDocumentCommand{\newmenumacro}{m O{\tw@default@input@sep} m}{%
\providemenumacro 674 \ifcsundef{\expandafter\@gobble\string#1}{%
675 \tw@define@menu@macro\NewDocumentCommand{#1}{#2}{#3}%
676 }{%
677 \tw@mk@error{Menu macro '\string#1' already defined!\MessageBreak
678 Use \string\renewmenustyle\space instead.}%
679 }%
680 }
681 \NewDocumentCommand{\renewmenumacro}{m O{\tw@default@input@sep} m}{%
682 \tw@define@menu@macro\RenewDocumentCommand{#1}{#2}{#3}%
683 }
684 \NewDocumentCommand{\providemenumacro}{m O{\tw@default@input@sep} m}{%
685 \ifcsundef{\expandafter\@gobble\string#1}{%
686 \tw@define@menu@macro\ProvideDocumentCommand{#1}{#2}{#3}%
687 }{%
688 \tw@mk@warning{Menu macro '\string#1' already defined!\MessageBreak
689 Use \string\renewmenustyle\space to redefine it.}%
690 }%
691 }
```

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:

```
692 \iftw@mk@definemenumacros
```

```
\menu And then we define three basic macros.
\directory 693 \newmenumacro{\menu}[>]{menus}
\keys 694 \newmenumacro{\directory}[/{paths}
695 \newmenumacro{\keys}[+]{roundedkeys}
```

Lastly we close the `definemacros` if statement:

```
696 \fi
```

6.8 Keys

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

```
697 \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.
```



```

698 \NewDocumentCommand{\tw@make@key@box}{m m}{%
699 %   \expandafter\newbox\csname tw@mk@box@#1\endcsname
700 %   \expandafter\setbox\csname tw@mk@box@#1\endcsname{%
701 %     #2%
702 %   }%
703   \csdef{tw@mk@#1}{%
704 %     \expandafter\usebox\csname tw@mk@box@#1\endcsname%
705 %     #2%
706 %   }%
707 }

```

`\tw@make@key@macro` The next macro defines the user level command by accessing a macro like `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.

```

708 \NewDocumentCommand{\tw@make@key@macro}{s m}{%
709   \IfBooleanTF{#1}{%
710     \expandafter\providecommand\csname\expandafter@gobble\string#2\endcsname{%
711       \expandonce{\maxsizebox{!}{1.8ex}}{%
712         \@nameuse{tw@mk@\expandafter@gobble\string#2@tw@mk@os}}%
713       }%
714     }%
715     \expandafter\providecommand\csname\expandafter@gobble\string#2mac\endcsname{%
716       \expandonce{\maxsizebox{!}{1.8ex}}{%
717         \@nameuse{tw@mk@\expandafter@gobble\string#2@mac}}%
718       }%
719     }%
720     \expandafter\providecommand\csname\expandafter@gobble\string#2win\endcsname{%
721       \expandonce{\maxsizebox{!}{1.8ex}}{%
722         \@nameuse{tw@mk@\expandafter@gobble\string#2@win}}%
723       }%
724     }%
725   }{%
726     \expandafter\providecommand\csname\expandafter@gobble\string#2\endcsname{%
727       \expandonce{\maxsizebox{!}{1.8ex}}{%
728         \@nameuse{tw@mk@\expandafter@gobble\string#2}}%
729       }%
730     }%
731   }%
732 }

```

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

```

733 \newcommand*{\tw@define@mackey}[2]{%
734   \IfStrEq{text}{\tw@mk@mackeys}{#1}{%
735     \IfStrEq{symbols}{\tw@mk@mackeys}{#2}{%
736     }%
737 }

```

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

```

738 \tikzset{
739   menukeys key symbol/.style={
740     rounded corners=0pt,
741     line width=0.1ex,
742     baseline={(0,0)},
743   },
744   menukeys thick/.style={line width=0.25ex},
745 }

```

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`

```

746 \normalsize
747 \tw@make@key@box{shift}{%
748   \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
749     \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
750           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
751           (0.3ex,1.2ex) -- cycle;
752   \end{tikzpicture}%
753 }
754 \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`

```

755 \tw@make@key@box{capslock@mac}{%
756   \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
757     \draw (0.3ex,0.7ex) -- (1.1ex,0.7ex) -- (1.1ex,1.2ex) -- %
758           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
759           (0.3ex,1.2ex) -- cycle;
760     \draw (0.3ex,0) rectangle (1.1ex,0.4ex);
761   \end{tikzpicture}%
762 }
763 \tw@make@key@box{capslock@win}{%
764   \begin{tikzpicture}[yscale=-1,yshift=-1.8ex,menukeys key symbol]
765     \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
766           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
767           (0.3ex,1.2ex) -- cycle;
768   \end{tikzpicture}%
769 }
770 \tw@make@key@macro*{\capslock}

```

Here are the other macros:

`\tab`

```

771 \tw@make@key@box{tab@mac}{%

```

```

772 \begin{tikzpicture}[yshift=0.6ex,menukeys key symbol]
773 \draw [->] (0,0) -- (1em,0);
774 \draw (1em,-0.35ex) -- (1em,0.35ex);
775 \end{tikzpicture}%
776 }
777 \tw@make@key@box{tab@win}{%
778 \begin{tikzpicture}[yshift=0.1ex,menukeys key symbol]
779 \draw [->] (0.2em,0) -- (1.2em,0);
780 \draw (1.2em,-0.35ex) -- (1.2em,0.35ex);
781 \draw [<-] (0,1ex) -- (1em,1ex);
782 \draw (0,0.65ex) -- (0,1.35ex);
783 \end{tikzpicture}%
784 }
785 \tw@make@key@macro*{\tab}

\esc
\oldesc 786 \def\tw@mk@esc@win{Esc}
787 \tw@define@mackey{%
788 \def\tw@mk@esc@mac{esc}
789 }{%
790 \tw@make@key@box{esc@mac}{%
791 \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
792 \draw [->] (0.5ex,0.5ex) -- ++(135:1.1ex);
793 \draw (0.5ex,0.5ex) ++(105:0.6ex) arc (105:-195:0.6ex);
794 \end{tikzpicture}%
795 }%
796 }
797 \tw@make@key@macro*{\esc}
798 \def\tw@mk@oldesc@win{Esc}
799 \tw@define@mackey{%
800 \def\tw@mk@oldesc@mac{esc}
801 }{%
802 \tw@make@key@box{oldesc@mac}{%
803 \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
804 \draw [->] (0.5ex,0.5ex) -- ++(45:1.1ex);
805 \draw (0.5ex,0.5ex) ++(15:0.6ex) arc (15:-285:0.6ex);
806 \end{tikzpicture}%
807 }%
808 }
809 \tw@make@key@macro*{\oldesc}

\ctrl
810 \providecommand\ctrlname{Ctrl}
811 \def\tw@mk@ctrl@win{\ctrlname}
812 \def\tw@mk@ctrl@mac{ctrl}
813 \tw@make@key@macro*{\ctrl}

\Alt
\AltGr 814 \def\tw@mk@Alt@win{Alt}

```

```

815 \tw@define@mackey{%
816   \def\tw@mk@Alt@mac{alt}%
817 }{%
818   \tw@make@key@box{Alt@mac}{%
819     \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
820       \draw (0,1ex) -- (0.5ex,1ex) -- (1ex,0.3ex) -- (1.8ex,0.3ex);
821       \draw (0.8ex,1ex) -- (1.8ex,1ex);
822     \end{tikzpicture}%
823   }%
824 }
825 \tw@make@key@macro*{\Alt}
826 \providecommand*{\AltGr}{Alt\,Gr}

\cmd
827 \def\tw@mk@cmd@win{%
828   \tw@mk@warning{'\string\cmd' only for Mac!}%
829 }
830 \tw@define@mackey{%
831   \def\tw@mk@cmd@mac{cmd}%
832 }{%
833   \tw@make@key@box{cmd@mac}{%
834     \begin{tikzpicture}[yshift=-0.15ex,menukeys key symbol]
835       \draw (0.5ex,0.7ex) -- (0.5ex,1.25ex) arc (0:270:0.25ex) -- %
836         (1.25ex,1ex) arc (-90:180:0.25ex) -- (1ex,0.25ex) %
837         arc (-180:90:0.25ex) -- (0.25ex,0.5ex) arc (90:360:0.25ex) %
838         -- cycle;
839     \end{tikzpicture}%
840   }%
841 }
842 \tw@make@key@macro*{\cmd}

\Space
\SPACE 843 \providecommand*{\Space}{\expandonce{\rule{3em}{0pt}}}
844 \newcommand{\spacename}{Space}
845 \providecommand*{\SPACE}{\expandonce{\rule{2em}{0pt}\spacename\rule{2em}{0pt}}}

\return
846 \tw@make@key@box{return@mac}{%
847   \begin{tikzpicture}[yshift=0.25ex,menukeys key symbol]
848     \draw [->, rounded corners=0.2ex] (1.25ex,1ex) -| %
849       (2ex,0) -- (0,0);
850   \end{tikzpicture}%
851 }
852 \tw@make@key@box{return@win}{%
853   \begin{tikzpicture}[menukeys key symbol]
854     \draw [->] (1ex,1.25ex) |- (0,0);
855   \end{tikzpicture}%
856 }
857 \tw@make@key@macro*{\return}

```

```

\enter
858 \def\tw@mk@enter@win{Enter}
859 \tw@make@key@box{enter@mac}{%
860   \begin{tikzpicture}[menukeys key symbol]
861     \draw (0,0) -- (0.5ex,0.5ex) -- (1ex,0);
862     \draw (0,0.55ex) -- (1ex,0.55ex);
863   \end{tikzpicture}%
864 }
865 \tw@make@key@macro*{\enter}

\winmenu
866 \def\tw@mk@winmenu@mac{%
867   \tw@mk@warning{'\string\winmenu' only for Windows!}%
868 }
869 \tw@make@key@box{winmenu@win}{%
870   \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
871     \draw (0,0) rectangle (1.5ex,1.8ex);
872     \draw (0.25ex,1.4ex) -- ++(1ex,0);
873     \draw (0.25ex,1ex) -- ++(1ex,0);
874     \draw (0.25ex,0.6ex) -- ++(1ex,0);
875   \end{tikzpicture}%
876 }
877 \tw@make@key@macro*{\winmenu}

\backspace
878 \tw@make@key@box{backspace}{%
879   \begin{tikzpicture}[yshift=0.65ex,menukeys key symbol]
880     \draw [lt-,menukeys thick] (0,0) -- (1.35em,0);
881   \end{tikzpicture}%
882 }
883 \tw@make@key@macro*{\backspace}

\del
\backdel 884 \providecommand{\delname}{Del.}
885 \def\tw@mk@del@win{\delname}
886 \tw@define@mackey{%
887   \def\tw@mk@del@mac{\delname}%
888 }{%
889   \tw@make@key@box{del@mac}{%
890     \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
891       \draw (0,0) -- (1.5ex,0) -- (2ex,0.5ex) --%
892         (1.5ex,1ex) -- (0,1ex) -- cycle;
893       \draw (0.5ex,0.2ex) -- (1.1ex,0.8ex);
894       \draw (0.5ex,0.8ex) -- (1.1ex,0.2ex);
895     \end{tikzpicture}%
896   }%
897 }
898 \tw@make@key@macro*{\del}
899 \def\tw@mk@backdel@win{\delname}

```

```

900 \tw@define@mackey{%
901   \def\tw@mk@backdel@mac{\delname}%
902 }{%
903   \tw@make@key@box{backdel@mac}{%
904     \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
905       \draw (2ex,0) -- (0.5ex,0) -- (0,0.5ex) --%
906         (0.5ex,1ex) -- (2ex,1ex) -- cycle;
907       \draw (1ex,0.2ex) -- (1.6ex,0.8ex);
908       \draw (1ex,0.8ex) -- (1.6ex,0.2ex);
909     \end{tikzpicture}%
910   }%
911 }
912 \tw@make@key@macro*{\backdel}

\arrowkeyup Lastly we define the arrow macros:
\arrowkeydown 913 \tw@make@key@box{arrowkeyup}{%
\arrowkeyleft 914   \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
\arrowkeyright 915     \draw [->] (0,0) -- (0,0.8em);
916   \end{tikzpicture}%
917 }
918 \tw@make@key@macro{\arrowkeyup}
919
920 \tw@make@key@box{arrowkeydown}{%
921   \begin{tikzpicture}[yshift=0.7em,menukeys key symbol]
922     \draw [->] (0,0) -- (0,-0.8em);
923   \end{tikzpicture}%
924 }
925 \tw@make@key@macro{\arrowkeydown}
926
927 \tw@make@key@box{arrowkeyright}{%
928   \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
929     \draw [->] (0,0) -- (0.8em,0);
930   \end{tikzpicture}%
931 }
932 \tw@make@key@macro{\arrowkeyright}
933
934 \tw@make@key@box{arrowkeyleft}{%
935   \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
936     \draw [->] (0,0) -- (-0.8em,0);
937   \end{tikzpicture}%
938 }
939 \tw@make@key@macro{\arrowkeyleft}

\arrowkey And the \arrowkey macro that get's it's direction as argument.
940 \newcommand{\arrowkey}[1]{%
941   \IfStrEq{~}{#1}{\arrowkeyup}{%
942     \IfStrEq{v}{#1}{\arrowkeydown}{%
943       \IfStrEq{<}{#1}{\arrowkeyleft}{%
944         \IfStrEq{>}{#1}{\arrowkeyright}{%
945           \tw@mk@error{Wrong value '#1' for \string\arrowkey}\MessageBreak

```

```
946             Possible values are '^', 'v', '<' or '>'}%
947             }%
948         }%
949     }%
950 }%
951 }
```

Close the \iftw@mk@definekeys

```
952 \fi
```

7 Change history

v1.0	Replaced obsolete <code>\tikzstyle</code> . . .	1
General: Initial version		1
v1.1	<code>\tw@define@menu@macro</code> : Replaced	
<code>\directory</code> : Renamed <code>\path</code> to	<code>\protected@edef</code> by <code>\def</code> . . .	29
<code>\directory</code> because it crashes		
with <code>biblatex</code>		31
General: Improved manual		1
Load <code>xcolor</code> before <code>menukeys</code>		14
v1.1a		
<code>\newenumacro</code> : Added a line to		
make a new macro robust.		31
<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.		29
v1.2		
<code>\tw@define@menu@macro</code> : Added		
<code>\leavevmode</code>		29
Replaced <code>\edef</code> by		
<code>\protected@edef</code>		29
General: Added <code>\normalsize</code>		
before symbol definitions to		
make the <code>ex</code> unit available		1
Added <code>\SPACE</code> and <code>\spacename</code>		1
Fixed GitHub issues #9, #10,		
#11, #13, #17, #24 and #26		1
Tidy up version and date		1
v1.2a		
General: Added braces to the		
<code>\tikz</code> macro since the parser		
seems to crash with <code>babel</code> 's		
french option otherwise.		1
	Replaced obsolete <code>\tikzstyle</code> . . .	1
	v1.2c	
	<code>\tw@define@menu@macro</code> : Replaced	
	<code>\protected@edef</code> by <code>\def</code> . . .	29
	v1.3	
	General: Added TikZ-styles for the	
	key symbols.	1
	Improved key symbols.	1
	v1.4	
	<code>\backdel</code> : Added <code>\backdel</code>	36
	<code>\oldesc</code> : Fixed direction of	
	<code>\escmac</code> ; added <code>\oldesc</code>	34
	General: Extended color theme	
	features.	1
	The <code>path...</code> styles now use the	
	text color of the selected color	
	theme (fix issue #16).	1
	v1.5	
	General: New option	
	<code>hyperrefcolorlinks</code>	17
	v1.6	
	<code>\newenumacro</code> : use	
	<code>\NewDocumentCommand</code>	31
	<code>\providenumacro</code> : use	
	<code>\ProvideDocumentCommand</code>	31
	<code>\renewenumacro</code> : use	
	<code>\RenewDocumentCommand</code>	31
	<code>\tw@define@menu@macro</code> : Don't	
	use <code>\NewDocumentCommand</code>	29
	General: <code>hyperrefcolorlinks</code>	
	obsolete	17
	Don't load <code>catoptions</code>	14
	Load order no longer important	4

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	<code>\@tw@hyphenatepaths@warnigtrue</code> . . .	560
<code>\@ifdefinable</code>		624
<code>\@tw@hyphenatepaths@warnigfalse</code>	<code>_</code>	27, 31, 35, 41, 56, 57, 66, 67

A		<code>\drawtikzfolder</code>	
<code>\Alt</code>	814 9, 587, 598, 604, 610, 616	
<code>\AltGr</code>	814		
E			
<code>angularkeys</code> (style)	6	<code>\enter</code>	
<code>angularmenus</code> (style)	5	858	
<code>\arrowkey</code>	13, 940	<code>\esc</code>	
<code>\arrowkeydown</code>	913, 942	786	
<code>\arrowkeyleft</code>	913, 943	<code>\exp</code>	
<code>\arrowkeyright</code>	913, 944	48, 64	
<code>\arrowkeyup</code>	913, 941	<code>\ExplSyntaxOff</code>	
		69	
		<code>\ExplSyntaxOn</code>	
		9	
B		F	
<code>\backdel</code>	884	<code>\font</code>	
<code>\backspace</code>	878	575	
<code>blacknwhite</code> (theme)	11	G	
<code>\bool</code>	60	<code>gray</code> (theme)	
		11	
		<code>\group</code>	
		46, 49	
C		H	
<code>\capslock</code>	755	<code>\hypersetup</code>	
<code>\changemenucolor</code>	12, 129	107	
<code>\changemenucolortheme</code>	10, 150	<code>hyphenatepaths</code> (style)	
<code>\changemenuelement</code>	10, 319, 597, 603, 609, 615	8	
<code>\cmd</code>	827	<code>hyphenatepathswithblackfolder</code> (style)	
<code>\color</code>	332, 360, 365, 369, 372, 391, 396, 400, 403, 421, 426, 430, 433, 449, 473, 493, 516, 537, 539, 548, 576	8	
Color themes:> <code>blacknwhite</code>	11	<code>hyphenatepathswithfolder</code> (style) ..	
Color themes:> <code>gray</code>	11	8	
<code>\copymenucolortheme</code>	12, 129	I	
<code>\copymenustyle</code>	10, 301, 596, 602, 608, 614	<code>\if@tw@hyphenatepaths@warnig</code> 559, 569	
<code>\cs</code>	10, 11, 13, 17, 24, 27, 31, 35, 41, 53, 54	<code>\iftw@mk@definekeys</code>	
<code>\ctrl</code>	810	697	
<code>\ctrlname</code>	13, 810, 811	<code>\iftw@mk@definemenumacros</code>	
<code>\CurrentMenuElement</code> .	10, 187, 190, 193, 196, 219, 222, 225, 228, 332, 360, 365, 369, 372, 391, 396, 400, 403, 421, 426, 430, 433, 449, 473, 493, 516, 532, 548, 577, 620, 641, 644, 650, 653	104	
		<code>\int</code>	
		22, 44, 50, 59, 62	
D		K	
<code>definekeys</code> (option)	4	<code>\keys</code>	
<code>definemenumacros</code> (option)	4	4, 693	
<code>\del</code>	884	L	
<code>\delname</code> . . .	13, 884, 885, 887, 899, 901	<code>\l</code>	
<code>\directory</code>	4, 693	19, 21, 22, 23, 24, 25, 26, 29, 33, 37, 38, 39, 43, 44, 47, 50, 58, 59, 60, 62, 63, 64	
		M	
		<code>mackeys</code> (option)	
		4, 13	
		<code>\menu</code>	
		4, 693	
		<code>menus</code> (style)	
		5	
		N	
		<code>\newmenucolortheme</code> .	
		11, 118, 165, 166	
		<code>\newmenumacro</code> ..	
		12, 673, 693, 694, 695	
		<code>\newmenustyle</code>	
		9, 232, 267	
		<code>\newmenustylesimple</code>	
		9, 232, 232	
		O	
		<code>\oldesc</code>	
		786	
		Options:> <code>definekeys</code>	
		4	

Options:>definenumacros	4	Styles:>shadowedangularkeys	6
Options:>mackeys	4, 13	Styles:>shadowedroundedkeys	6
Options:>os	4	Styles:>typewriterkeys	7
os (option)	4		

P

<code>\PassOptionsToPackage</code>	108
paths (style)	7
pathswithblackfolder (style)	7
pathswithfolder (style)	7
<code>\prg</code>	12
<code>\protected</code>	626
<code>\providenumacro</code>	13, 673
<code>\providemenustyle</code>	11, 232, 287
<code>\providemenustylesimple</code>	11, 232, 252

R

<code>\relsize</code>	352, 383, 413, 444, 452, 464, 477, 488, 496, 507, 519, 528, 543
<code>\renewmenucolortheme</code>	12, 118, 147
<code>\renewnummacro</code>	13, 673
<code>\renewmenustyle</code>	11, 232, 276, 280, 297, 678, 689
<code>\renewmenustylesimple</code>	11, 232, 241, 245, 262
<code>\return</code>	846
roundedkeys (style)	6
roundedmenus (style)	5

S

<code>\seq</code>	19, 21, 25, 26, 29, 33, 37, 38, 39, 43, 47, 53, 58, 60, 63
shadowedangularkeys (style)	6
shadowedroundedkeys (style)	6
<code>\shift</code>	746
<code>\SPACE</code>	843
<code>\Space</code>	843
<code>\spacename</code>	13, 844, 845
Styles:>angularkeys	6
Styles:>angularmenus	5
Styles:>hyphenatepathswithblackfolder	8
Styles:>hyphenatepathswithfolder	8
Styles:>hyphenatepaths	8
Styles:>menus	5
Styles:>pathswithblackfolder	7
Styles:>pathswithfolder	7
Styles:>paths	7
Styles:>roundedkeys	6
Styles:>roundedmenus	5

T

<code>\tab</code>	771
<code>\tikzset</code>	339, 344, 375, 406, 436, 456, 481, 500, 523, 738
<code>\tl</code>	10, 12, 15, 23
<code>\tw@current@color@theme</code>	163, 660
<code>\tw@current@style</code>	202, 203, 208
<code>\tw@declare@style</code>	205, 270, 272, 282, 284, 290, 292, 357, 388, 418
<code>\tw@declare@style@simple</code>	172, 235, 237, 247, 249, 255, 257, 446, 470, 490, 513, 530, 547, 561
<code>\tw@declare@style</code>	199
<code>\tw@declare@style@extra@args</code>	199
<code>\tw@default@input@sep</code>	619, 673, 681, 684
<code>\tw@default@post</code>	167, 173, 200
<code>\tw@default@pre</code>	167, 173, 206
<code>\tw@default@sep</code>	167, 173, 206
<code>\tw@define@macro</code>	733, 787, 799, 815, 830, 886, 900
<code>\tw@define@menu@macro</code>	621, 675, 682, 686
<code>\tw@make@color@theme</code>	110, 120, 127
<code>\tw@make@key@box</code>	698, 747, 755, 763, 771, 777, 790, 802, 818, 833, 846, 852, 859, 869, 878, 889, 903, 913, 920, 927, 934
<code>\tw@make@key@macro</code>	708, 754, 770, 785, 797, 809, 813, 825, 842, 857, 865, 877, 883, 898, 912, 918, 925, 932, 939
<code>\tw@menu@list</code>	663, 665, 668
<code>\tw@mk@Alt@mac</code>	816
<code>\tw@mk@Alt@win</code>	814
<code>\tw@mk@backdel@mac</code>	901
<code>\tw@mk@backdel@win</code>	899
<code>\tw@mk@cmd@mac</code>	831
<code>\tw@mk@cmd@win</code>	827
<code>\tw@mk@ctrl@mac</code>	812
<code>\tw@mk@ctrl@win</code>	811
<code>\tw@mk@del@mac</code>	887
<code>\tw@mk@del@win</code>	885
<code>\tw@mk@center@win</code>	858

<code>\tw@mk@error</code>	70, 97, 101, 122, 133, 146, 152, 156, 240, 275, 304, 316, 321, 335, 634, 677, 945	<code>\tw@mk@warning@noline</code>	70
<code>\tw@mk@esc@mac</code>	788	<code>\tw@mk@winmenu@mac</code>	866
<code>\tw@mk@esc@win</code>	786	<code>\tw@mk@xifinsetTF</code>	13, 627
<code>\tw@mk@expanded</code>	11, 668	<code>\tw@set@tikz@colors</code>	339
<code>\tw@mk@gobble@args</code>	81, 242, 263, 277, 298	<code>\tw@typewriterkeys@curr@elem</code>	531, 537, 539
<code>\tw@mk@iflastindris</code>	17, 638	<code>typewriterkeys (style)</code>	7
<code>\tw@mk@indrisloop</code>	54, 668	U	
<code>\tw@mk@indrisnr</code>	24, 640, 649	<code>\use</code>	11
<code>\tw@mk@mackeys</code>	100, 734, 735	<code>\usemenucolor</code>	10 , 162, 332, 340, 341, 342, 360, 365, 369, 372, 391, 396, 400, 403, 421, 426, 430, 433, 449, 452, 467, 473, 477, 493, 496, 510, 516, 519, 537, 539, 543, 548, 552, 576, 581, 598, 604, 610, 616
<code>\tw@mk@oldesc@mac</code>	800	W	
<code>\tw@mk@oldesc@win</code>	798	<code>\winmenu</code>	866
<code>\tw@mk@os</code>	96, 712		
<code>\tw@mk@tempa</code>	79, 82, 83, 663, 665, 668		
<code>\tw@mk@tempb</code>	79, 667, 668		
<code>\tw@mk@test@input@sep</code>	624, 626, 662		
<code>\tw@mk@trimspaces</code>	10, 628, 665		
<code>\tw@mk@warning</code>	70, 105, 260, 295, 570, 688, 828, 867		