

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

2026/05/14 — v1.6.3

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	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	11
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	13
6	Implementation	14
6.1	Required packages	14
6.2	Helper macros	16
6.3	Options	16
6.4	Workarounds	17
6.4.1	<code>hyperref</code> 's <code>colorlinks</code> option	17
6.5	Color themes	17
6.5.1	Internal commands	17
6.5.2	User-level commands	17
6.5.3	Predefined themes	18
6.6	Styles	18
6.6.1	Internal commands	19
6.6.2	User-level commands	20
6.6.3	Copying and changing	21
6.6.4	Predefined styles	22
6.7	Menu macros	28
6.7.1	Internal commands	28
6.7.2	User-level commands	29
6.7.3	Predefined menu macros	30
6.8	Keys	30
7	Change history	37
8	Macro index	38

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: 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.

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 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

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.

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

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{<macro>}[<input sep>]{<style>}`. 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`

```
\newmenumacro\test{menus}  
\test{File,Extras,Preferences}  
\test{Menu}
```

File » Extras » Preferences

Menu

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.

This is the default style for `\menu`.

Name: `roundedmenus`

```
\renewmenumacro\test{roundedmenus}  
\test{File,Extras,Preferences}  
\test{Menu}
```

File » Extras » Preferences

Menu

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.

Name: `angularmenus`

```
\renewmenumacro\test{angularmenus}  
\test{File,Extras,Preferences}  
\test{Menu}
```

File » Extras » Preferences

Menu

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

Name: roundedkeys

Ctrl

Alt

Q

S

```

\renewmenumacro\test{roundedkeys}
\test{Ctrl,Alt,Q}
\test{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.
This is the default style for \keys.

Name: shadowedroundedkeys

Ctrl

Alt

Q

S

```

\renewmenumacro\test{shadowedroundedkeys}
\test{Ctrl,Alt,Q}
\test{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

```

\renewmenumacro\test{angularkeys}
\test{Ctrl,Alt,Q}
\test{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

```

\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




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	<code>\renewmenumacro\test{typewriterkeys}</code>
	<code>\test{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 `typewriterkeys`. And again some blind text without any sense.


The color of + is taken from optional color B.

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



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

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

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>
C: ▶ Database ▶ User ▶ ALongUser	<code>\test{C:,Database,User,ALongUserNameHere,</code>
NameHere ▶ ALongerFolder	<code>ALongerFolderNameAtThisPlace,MyFile.tex}</code>
NameAtThisPlace▶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: ▶ Database ▶ User ▶ ALongUserNameHere ▶ ALongerFolder NameAtThisPlace ▶ MyFile.tex is the result of a style which name is hyphen atepaths. And again some blind text without any sense.

The sep color is taken from optional color C.

Name: hyphenatepathswith folder	<code>\renewmenumacro\test {hyphenatepathswithfolder}</code>
☐ C: ▶ Database ▶ User ▶ ALongUser	<code>\test{C:,Database,User,ALongUserNameHere,</code>
NameHere ▶ ALongerFolder	<code>ALongerFolderNameAtThisPlace,MyFile.tex}</code>
NameAtThisPlace▶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: ▶ Database ▶ User ▶ ALongUserNameHere ▶ ALongerFolder NameAtThisPlace ▶ MyFile.tex is the result of a style which name is hyphen atepathswithfolder. 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 folder	<code>\renewmenumacro\test {hyphenatepathswithblackfolder}</code>
▣ C: ▶ Database ▶ User ▶ ALongUser	<code>\test{C:,Database,User,ALongUserNameHere,</code>
NameHere ▶ ALongerFolder	<code>ALongerFolderNameAtThisPlace,MyFile.tex}</code>
NameAtThisPlace▶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: ▶ Database ▶ User ▶ ALongUserNameHere ▶ ALongerFolder NameAtThisPlace ▶ MyFile.tex is the result of a style which name is hyphen atepathswithblackfolder. 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 above described macros: `\renewmenustylesimple`, `\providemenustylesimple`, `\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>}{
}{<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`, `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.

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{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 (*pkg)

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 \ 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.

```

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 \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.

```

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]{definenumacros}
93 \DeclareBoolOption[true]{definekeys}
94 \DeclareBoolOption[false]{hyperrefcolorlinks}
95 \DeclareStringOption[mac]{os}
96 \DeclareStringOption[symbols]{mackeys}
97 \ProcessKeyvalOptions{tw@mk}\relax

```

Now we have to do some error treatment:

```

98 \IfSubStr{.mac.win.}{.\tw@mk@os.}{}{%
99 \tw@mk@error{Unknown value for option 'os'\MessageBreak

```



```

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

```

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.

```

106 \iftw@mk@hyperrefcolorlinks
107   \tw@mk@warning{The option 'hyperrefcolorlinks' is obsolete}
108   \ifpackageloaded{hyperref}
109     {\hypersetup{colorlinks}}
110     {\PassOptionsToPackage{colorlinks}{hyperref}}
111 \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

```

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

```

6.5.2 User-level commands

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

```

\renewmenucolortheme 120 \NewDocumentCommand{\newmenucolortheme}{ m m m m m 0{#3} 0{#4} 0{#5} }{%
121   \@ifundefinedcolor{tw@color@theme@#1@bg}{%
122     \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
123   }{%
124     \tw@mk@error{Color theme '#1' already defined!\MessageBreak
125     Use \string\renewmenucolortheme\space instead.}%
126   }
127 }
128 \NewDocumentCommand{\renewmenucolortheme}{ m m m m m 0{#3} 0{#4} 0{#5} }{%
129   \tw@make@color@theme{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}%
130 }

```

```

\changemenucolor Lastly we define the changing and copying commands
\copymenucolortheme 131 \newcommand*\changemenucolor}[4]{%
132   \IfSubStr{ bg br txt }{ #2 }{%
133     \definecolor{tw@color@theme@#1@#2}{#3}{#4}%
134   }{%
135     \tw@mk@error{No such color element ('#2')!\MessageBreak
136       Possible values are bg, br and txt.}
137   }%
138 }
139 \newcommand*\copymenucolortheme}[2]{%
140   \@ifundefinedcolor{tw@color@theme@#1@bg}{%
141     \colorlet{tw@color@theme@#1@bg}{tw@color@theme@#2@bg}%
142     \colorlet{tw@color@theme@#1@br}{tw@color@theme@#2@br}%
143     \colorlet{tw@color@theme@#1@txt}{tw@color@theme@#2@txt}%
144     \colorlet{tw@color@theme@#1@a}{tw@color@theme@#2@a}%
145     \colorlet{tw@color@theme@#1@b}{tw@color@theme@#2@b}%
146     \colorlet{tw@color@theme@#1@c}{tw@color@theme@#2@c}%
147   }{%
148     \tw@mk@error{Color theme '#1' already defined!\MessageBreak
149       Use \string\renewmenucolortheme\space instead.}
150   }
151 }

\changemenucolortheme To be able to change the color theme of a style we must define this:
152 \newcommand*\changemenucolortheme}[2]{%
153   \ifcsundef{tw@style@#1@pre}{%
154     \tw@mk@error{Style '#1' undefined!\MessageBreak
155       Maybe you misspelled it?}%
156   }{%
157     \@ifundefinedcolor{tw@color@theme@#2@bg}{%
158       \tw@mk@error{Color theme '#2' is not defined!}%
159     }{%
160       \csdef{tw@style@#1@color@theme}{#2}%
161     }%
162   }%
163 }

\usemenucolor To use a color of a theme we define \usemenucolor as following.
164 \newcommand*\usemenucolor}[1]{%
165   tw@color@theme@\tw@current@color@theme @#1%
166 }

```

6.5.3 Predefined themes

There are two predefined color themes

```

167 \newmenucolortheme{gray}{gray}{0.95}{0.3}{0}{0.95}[0][0]
168 \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 169 \newcommand{\tw@default@sep}{%
170 \hspace{0.2em plus 0.1em minus 0.5em}%
171 }
172 \newcommand{\tw@default@pre}{}
173 \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.
174 \DeclareDocumentCommand{\tw@declare@style@simple}{%
175 s m O{\tw@default@pre} m O{\tw@default@sep} O{\tw@default@post} m
176 }{%
177 \csdef{tw@style@#2@color@theme}{#7}%
178 \csdef{tw@style@#2@pre}{#3}%
179 \csdef{tw@style@#2@sep}{#5}%
180 \csdef{tw@style@#2@post}{#6}%
181 \IfBooleanTF{#1}{%
182 \csdef{tw@style@#2@single}{#4}%
183 \csdef{tw@style@#2@first}{#4}%
184 \csdef{tw@style@#2@mid}{#4}%
185 \csdef{tw@style@#2@last}{#4}%
186 }{%
187 \csdef{tw@style@#2@single}{%
188 \tikz[baseline=(tw@node.base)]{%
189 \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
190 \csdef{tw@style@#2@first}{%
191 \tikz[baseline=(tw@node.base)]{%
192 \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
193 \csdef{tw@style@#2@mid}{%
194 \tikz[baseline=(tw@node.base)]{%
195 \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
196 \csdef{tw@style@#2@last}{%
197 \tikz[baseline=(tw@node.base)]{%
198 \node(tw@node)[#4]{\strut\CurrentMenuElement};}%
199 }%
200 }

```

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

```

201 \DeclareDocumentCommand{\tw@declare@style@extra@args}{%
202 O{\tw@default@post} m
203 }{%
204 \csdef{tw@style@\tw@current@style @post}{#1}%
205 \csdef{tw@style@\tw@current@style @color@theme}{#2}%
206 }

```

Now we can define \tw@declare@style:

```

207 \DeclareDocumentCommand{\tw@declare@style}{%
208 s m O{\tw@default@pre} m O{\tw@default@sep} m m m

```

```

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

```

6.6.2 User-level commands

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

```

renewmenustylesimple 234 \NewDocumentCommand{\newmenustylesimple}{s m}{%
providemenustylesimple 235   \ifcsundef{tw@style@#2@pre}{%
  newmenustyle 236     \IfBooleanTF{#1}{%
    renewmenustyle 237       \tw@declare@style@simple*{#2}%
    providemenustyle 238     }{%
      239       \tw@declare@style@simple{#2}%
      240     }%
    241   }{%
    242     \tw@mk@error{Style '#2' already defined!\MessageBreak
    243       Use \string\renewmenustylesimple\space instead.}%
    244     \tw@mk@gobble@args{o m o o m}%
    245   }%
  246 }
  247 \NewDocumentCommand{\renewmenustylesimple}{s m}{%
  248   \IfBooleanTF{#1}{%
  249     \tw@declare@style@simple*{#2}%
  250   }{%
  251     \tw@declare@style@simple{#2}%
  252   }%
  253 }
  254 \NewDocumentCommand{\providemenustylesimple}{s m}{%
  255   \ifcsundef{tw@style@#2@pre}{%
  256     \IfBooleanTF{#1}{%
  257       \tw@declare@style@simple*{#2}%
  258     }{%

```

```

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

```

6.6.3 Copying and changing

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

```

303 \newcommand*{\copymenustyle}[2]{%
304     \ifcsundef{tw@style@#1@pre}{%
305         \ifcsundef{tw@style@#2@pre}{%
306             \tw@mk@error{Can't copy not existing style ('#2')!}%
307         }{%
308             \csletcs{tw@style@#1@pre}{tw@style@#2@pre}%

```

```

309         \csletcs{tw@style@#1@post}{tw@style@#2@post}%
310         \csletcs{tw@style@#1@sep}{tw@style@#2@sep}%
311         \csletcs{tw@style@#1@single}{tw@style@#2@single}%
312         \csletcs{tw@style@#1@first}{tw@style@#2@first}%
313         \csletcs{tw@style@#1@mid}{tw@style@#2@mid}%
314         \csletcs{tw@style@#1@last}{tw@style@#2@last}%
315         \csletcs{tw@style@#1@color@theme}{tw@style@#2@color@theme}
316     }%
317 }{%
318     \tw@mk@error{Style '#1' already exists!}%
319 }%
320 }

```

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

```

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

```

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

```

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

```

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`.

```

346 \tikzset{tw@menus@base/.style={%
347     tw@set@tikz@colors,
348     rounded corners=0.15ex,
349     inner sep=0pt,
350     inner xsep=2pt,
351     text height=1.825ex,

```

```

352 text depth=0.7ex,
353 minimum width=1.5em,
354 font=\relsize{-1}\sffamily,
355 signal,
356 signal to=nowhere,
357 signal pointer angle=110,
358 }}
359 \tw@declare@style*{menus}{%
360 \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
361 \node(tw@node)[tw@menus@base,signal to=east]%
362 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
363 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
364 {%
365 \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
366 \node(tw@node)[tw@menus@base,signal from=west,signal to=east]%
367 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
368 }{%
369 \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
370 \node(tw@node)[tw@menus@base,signal from=west,]%
371 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
372 }{%
373 \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
374 \node(tw@node)[tw@menus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
375 }{gray}
376
377 \tikzset{tw@roundedmenus@base/.style={%
378 tw@set@tikz@colors,
379 rounded corners=0.3ex,
380 inner sep=0pt,
381 inner xsep=2pt,
382 text height=1.825ex,
383 text depth=0.7ex,
384 minimum width=1.5em,
385 font=\relsize{-1}\sffamily,
386 signal,
387 signal to=nowhere,
388 signal pointer angle=110,
389 }}
390 \tw@declare@style*{roundedmenus}{%
391 \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
392 \node(tw@node)[tw@roundedmenus@base,signal to=east]%
393 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
394 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
395 {%
396 \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
397 \node(tw@node)[tw@roundedmenus@base,signal from=west,signal to=east]%
398 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
399 }{%
400 \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
401 \node(tw@node)[tw@roundedmenus@base,signal from=west,]%
402 {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
403 }{%
404 \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
405 \node(tw@node)[tw@roundedmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%

```

```

406 }{gray}
407
408 \tikzset{tw@angularmenus@base/.style={%
409     tw@set@tikz@colors,
410     inner sep=0pt,
411     inner xsep=2pt,
412     text height=1.825ex,
413     text depth=0.7ex,
414     minimum width=1.5em,
415     font=\relsize{-1}\sffamily,
416     signal,
417     signal to=nowhere,
418     signal pointer angle=110,
419 }}
420 \tw@declare@style*{angularmenus}{%
421     \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
422         \node(tw@node)[tw@angularmenus@base,signal to=east]%
423             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
424 }[\hspace{-0.2em}\hspace{0em plus 0.1em minus 0.05em}]%
425 {%
426     \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
427         \node(tw@node)[tw@angularmenus@base,signal from=west,signal to=east]%
428             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
429 }{%
430     \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
431         \node(tw@node)[tw@angularmenus@base,signal from=west,]%
432             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
433 }{%
434     \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
435         \node(tw@node)[tw@angularmenus@base]{\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
436 }{gray}
437
438 \tikzset{tw@roundedkeys@base/.style={%
439     tw@set@tikz@colors,
440     rounded corners=0.3ex,
441     inner sep=0pt,
442     inner xsep=2pt,
443     text height=1.825ex,
444     text depth=0.7ex,
445     minimum width=1.5em,
446     font=\relsize{-1}\sffamily,
447 }}
448 \tw@declare@style*{simple*{roundedkeys}}{%
449     \tikz[baseline={$(tw@node.base)+(0,-0.2ex)$}]{%
450         \node(tw@node)[tw@roundedkeys@base]%
451             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
452 }[%
453     \hspace{0.1em plus 0.1em minus 0.05em}%
454     \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
455     \hspace{0.1em plus 0.1em minus 0.05em}%
456 ]{gray}
457
458 \tikzset{tw@shadowedroundedkeys@base/.style={%
459     tw@set@tikz@colors,

```



```

460 rounded corners=0.3ex,
461 inner sep=0pt,
462 inner xsep=2pt,
463 text height=1.825ex,
464 text depth=0.7ex,
465 minimum width=1.5em,
466 font=\relsize{-1}\sffamily,
467 general shadow={%
468     shadow xshift=.2ex, shadow yshift=-.15ex,
469     fill=\usemenucolor{c},
470 },
471 }}
472 \tw@declare@style@simple*{shadowedroundedkeys}{%
473     \tikz[baseline={($\tw@node.base)+(0,-0.2ex)$}]{%
474         \node(tw@node)[tw@shadowedroundedkeys@base]%
475             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
476     }%
477 }[%
478     \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
479     \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
480     \hspace{0.1em plus 0.1em minus 0.05em}%

481 \end{package}
482 \cur\relax\kern0.2ex\relax\gray
483 \l161\hspace{0.2ex}\gray
484 \end{package}

485
486 \tikzset{tw@angularkeys@base/.style={%
487     tw@set@tikz@colors,
488     inner sep=0pt,
489     inner xsep=2pt,
490     text height=1.825ex,
491     text depth=0.7ex,
492     minimum width=1.5em,
493     font=\relsize{-1}\sffamily,
494 }}
495 \tw@declare@style@simple*{angularkeys}{%
496     \tikz[baseline={($\tw@node.base)+(0,-0.2ex)$}]{%
497         \node(tw@node)[tw@angularkeys@base]%
498             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};%
499 }[%
500     \hspace{0.1em plus 0.1em minus 0.05em}%
501     \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
502     \hspace{0.1em plus 0.1em minus 0.05em}%
503 ]\gray
504
505 \tikzset{tw@shadowedangularkeys@base/.style={%
506     tw@set@tikz@colors,
507     inner sep=0pt,
508     inner xsep=2pt,
509     text height=1.825ex,
510     text depth=0.7ex,
511     minimum width=1.5em,
512     font=\relsize{-1}\sffamily,

```

```

513   general shadow={%
514       shadow xshift=.2ex, shadow yshift=-.15ex,
515       fill=\usemenucolor{c},
516   },
517 }}
518 \tw@declare@style@simple*{shadowedangularkeys}{%
519     \tikz[baseline={($(\tw@node.base)+(0,-0.2ex)$)}]{%
520         \node(tw@node)[tw@shadowedangularkeys@base]%
521             {\strut\color{\usemenucolor{txt}}\CurrentMenuElement};}%
522 }[%
523     \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
524     \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
525     \hspace{0.1em plus 0.1em minus 0.05em}%

526 \end{pgf}
527 \cur\kern0.2ex\relax\gray}
528 \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
529 \end{pgf}

530
531 \tikzset{tw@typewriterkeys@base/.style={%
532     tw@set@tikz@colors,
533     shape=circle,
534     minimum size=2ex,
535     inner sep=0.5pt, outer sep=1pt,
536     font=\ttfamily\relsize{-1},
537 }}
538 \tw@declare@style@simple*{typewriterkeys}{%
539     \def\tw@typewriterkeys@curr@elem{%
540         \maxsizebox*{2ex}{2ex}{\CurrentMenuElement}%
541     }%
542     \begin{tikzpicture}[baseline={($(\tw@node.south)+(0,0.8ex)$)}]{%
543         \node(tw@node)[%
544             tw@typewriterkeys@base, inner sep=1.25pt, line width=0.6pt%
545             ]{\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
546         \node[tw@typewriterkeys@base]%
547             {\color{\usemenucolor{txt}}\tw@typewriterkeys@curr@elem};
548     \end{tikzpicture}%
549 }[%
550     \hspace{0.2ex}\hspace{0.1em plus 0.1em minus 0.05em}%
551     \textcolor{\usemenucolor{b}}{\raisebox{0.25ex}{\sffamily\relsize{-2}+}}%
552     \hspace{0.1em plus 0.1em minus 0.05em}%
553 ]{blacknwhite}

554
555 \tw@declare@style@simple*{paths}{%
556     {\ttfamily\color{\usemenucolor{txt}}\CurrentMenuElement}%
557 }[%
558     \hspace{0.2em plus 0.1em}%
559     \raisebox{0.08ex}{%
560         \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
561             -- (0,1ex) -- cycle;}%
562     }%
563     \hspace{0.2em plus 0.1em}%
564 ]{blacknwhite}
565

```

```

566 \newcounter{tw@hyphen@char@num}
567 \newif\if@tw@hyphenatepaths@warnig
568 \@tw@hyphenatepaths@warnigtrue
569 \tw@declare@style@simple*{hyphenatepaths}{%
570   {\ttfamily
571     \IfStrEq{T1}{\encodingdefault}{%
572       \setcounter{tw@hyphen@char@num}{23}%
573     }{%
574       \IfStrEq{OT1}{\encodingdefault}{%
575         \setcounter{tw@hyphen@char@num}{255}%
576       }{%
577         \if@tw@hyphenatepaths@warnig%
578         \tw@mk@warning{The hyphenatepaths styles will probably only\MessageBreak
579           work with T1 or OT1 encoding.}%
580         \fi\global\@tw@hyphenatepaths@warnigfalse%
581       }%
582     }%
583     \hyphenchar\font=\value{tw@hyphen@char@num}\relax
584     \color{\usemenucolor{txt}}}%
585     \CurrentMenuElement}%
586 }[%
587   \hspace{0.2em plus 0.1em}%
588   \raisebox{0.08ex}{%
589     \tikz{\fill[\usemenucolor{c}] (0,0) -- (0.5ex,0.5ex)%
590       -- (0,1ex) -- cycle;}%
591   }%
592   \hspace{0.2em plus 0.1em}%
593 }{blacknwhite}
594
595 \NewDocumentCommand{\drawtikzfolder}{0{white} 0{black}}{%
596   \begin{tikzpicture}[rounded corners=0.02ex,scale=0.7]
597     \draw [#2] (0,0) -- (1em,0) -- (1em,1.5ex) -- (0.5em,1.5ex) -- %
598       (0.4em,1.7ex) -- (0.1em,1.7ex) -- (0,1.5ex) -- cycle;
599     \draw [#2,fill=#1] (0,0) -- (1em,0) -- (0.85em,1.15ex) -- %
600       ++(-1em,0) -- cycle;
601   \end{tikzpicture}%
602 }
603
604 \copymenustyle{pathswithfolder}{paths}
605 \changemenuelement{pathswithfolder}{pre}{%
606   \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
607   \hspace{0.2em plus 0.1em}%
608 }
609
610 \copymenustyle{pathswithblackfolder}{paths}
611 \changemenuelement{pathswithblackfolder}{pre}{%
612   \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
613   \hspace{0.2em plus 0.1em}%
614 }
615
616 \copymenustyle{hyphenatepathswithfolder}{hyphenatepaths}
617 \changemenuelement{hyphenatepathswithfolder}{pre}{%
618   \drawtikzfolder[\usemenucolor{a}][\usemenucolor{b}]%
619   \hspace{0.2em plus 0.1em}%

```

```

620 }
621
622 \copymenustyle{hyphenatepathswithblackfolder}{hyphenatepaths}
623 \changemenuelement{hyphenatepathswithblackfolder}{pre}{%
624   \drawtikzfolder[\usemenucolor{c}][\usemenucolor{b}]%
625   \hspace{0.2em plus 0.1em}%
626 }

```

6.7 Menu macros

6.7.1 Internal commands

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

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

`\CurrentMenuElement` and the `\CurrentMenuElement` dummy

```
628 \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 <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 L^AT_EX3. Thank you both very much!

```

629 \begingroup
630 \lccode'\,=1
631 \lowercase{\endgroup
632   \@ifdefinable\tw@mk@test@input@sep
633   {%
634     \protected\def\tw@mk@test@input@sep#1{%
635       \tw@mk@xifinsetTF
636       {,\tw@mk@trimspaces{#1},}{,bslash,backslash,directory,location,}%
637     }%
638   }%
639 }
640 \newcommand\tw@define@menu@macro[3]
641 {%
642   \IfNoValueTF{#3}
643   {\tw@mk@exp@Nnno\tw@define@menu@macro{#1}{#2}\tw@default@input@sep}
644   {\tw@define@menu@macro{#1}{#2}{#3}}%
645 }
646 \newcommand\tw@define@menu@macro@[4]
647 {%
648   \ifcsundef{tw@style@#4@sep}
649   {%
650     \tw@mk@error
651     {%
652       Can't define menu macro \string#2\space,\MessageBreak
653       because the style '#4' is not available!%
654     }%
655   }
656   {%
657     \csdef{tw@parse@menu@list@tw@mk@string#2}##1%
658     {%

```

```

659         \def\CurrentMenuElement{##1}%
660         \tw@mk@iflastindris
661         {%
662             \ifnum\tw@mk@indrisnr=\@ne
663                 \@nameuse{tw@style@#4@single}%
664             \else
665                 \@nameuse{tw@style@#4@sep}%
666                 \@nameuse{tw@style@#4@last}%
667             \fi
668         }
669         {%
670             \ifnum\tw@mk@indrisnr=\@ne
671                 \@nameuse{tw@style@#4@first}%
672             \else
673                 \@nameuse{tw@style@#4@sep}%
674                 \@nameuse{tw@style@#4@mid}%
675             \fi
676         }%
677     }%
678     #1#2{+0{#3}+m}%
679     {%
680         \leavevmode
681         \begingroup
682         \def\tw@current@color@theme
683             {\csname tw@style@#4@color@theme\endcsname}%
684         \@nameuse{tw@style@#4@pre}%
685         \tw@mk@test@input@sep{##1}
686         {%
687             \edef\tw@menu@list{\detokenize{##2}}%
688             \edef\tw@mk@tempa{\@backslashchar}%
689         }
690         {%
691             \edef\tw@menu@list{\unexpanded{##2}}%
692             \edef\tw@mk@tempa{\tw@mk@trimspaces{##1}}%
693         }%
694         \begingroup
695         \letcs{\tw@mk@tempb}{tw@parse@menu@list@tw@mk@string#2}%
696         \tw@mk@indrisloop\tw@mk@tempa\tw@menu@list\tw@mk@tempb
697         \endgroup
698         \@nameuse{tw@style@#4@post}%
699     \endgroup
700 }%
701 }%
702 }

```

6.7.2 User-level commands

`\newmenumacro` Now it's time to build the user-level commands

```

\renewmenumacro 703 \NewDocumentCommand{\newmenumacro}{m o m}{%
\providemenumacro 704 \ifcsundef{\tw@mk@string#1}{%
705     \tw@define@menu@macro\NewDocumentCommand{#1}{#2}{#3}%
706     }{%
707     \tw@mk@error{Menu macro '\string#1' already defined!\MessageBreak
708     Use \string\renewmenustyle\space instead.}%

```

```

709     }%
710 }
711 \NewDocumentCommand{\renewmenumacro}{m o m}{%
712     \tw@define@menu@macro\RenewDocumentCommand{#1}{#2}{#3}%
713 }
714 \NewDocumentCommand{\providemenumacro}{m o m}{%
715     \ifcsundef{tw@mk@string#1}{%
716         \tw@define@menu@macro\ProvideDocumentCommand{#1}{#2}{#3}%
717     }{%
718         \tw@mk@warning{Menu macro '\string#1' already defined!\MessageBreak
719             Use \string\renewmenustyle\space to redefine it.}%
720     }%
721 }

```

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:

```
722 \iftw@mk@definemenumacros
```

```

\menu And then we define three basic macros.
\directory 723 \newmenumacro{\menu}[>]{menus}
\keys 724 \newmenumacro{\directory}[/{]{paths}
725 \newmenumacro{\keys}[+]{roundedkeys}

```

Lastly we close the `definemacros` if statement:

```
726 \fi
```

6.8 Keys

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

```
727 \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`.

```

728 \NewDocumentCommand{\tw@make@key@box}{m m}{%
729 % \expandafter\newbox\csname tw@mk@box@#1\endcsname
730 % \expandafter\sbox\csname tw@mk@box@#1\endcsname{%
731 %     #2%
732 % }%
733 \csdef{tw@mk@#1}{%
734 %     \expandafter\usebox\csname tw@mk@box@#1\endcsname%
735 %     #2%
736 % }%
737 }

```

```
\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.

```

738 \NewDocumentCommand{\tw@make@key@macro}{s m}{%
739   \IfBooleanTF{#1}{%
740     \expandafter\providecommand\csname\tw@mk@string#2\endcsname{%
741       \expandonce{\maxsizebox{!}}{1.8ex}{%
742         \@nameuse{tw@mk@\tw@mk@string#2@\tw@mk@os}}}%
743     }%
744   }%
745   \expandafter\providecommand\csname\tw@mk@string#2mac\endcsname{%
746     \expandonce{\maxsizebox{!}}{1.8ex}{%
747       \@nameuse{tw@mk@\tw@mk@string#2@mac}}}%
748   }%
749 }%
750 \expandafter\providecommand\csname\tw@mk@string#2win\endcsname{%
751   \expandonce{\maxsizebox{!}}{1.8ex}{%
752     \@nameuse{tw@mk@\tw@mk@string#2@win}}}%
753   }%
754 }%
755 }{%
756   \expandafter\providecommand\csname\tw@mk@string#2\endcsname{%
757     \expandonce{\maxsizebox{!}}{1.8ex}{%
758       \@nameuse{tw@mk@\tw@mk@string#2}}}%
759   }%
760 }%
761 }%
762 }

```

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

```

763 \newcommand*{\tw@define@mackey}[2]{%
764   \IfStrEq{text}{\tw@mk@mackeys}{#1}{%
765     \IfStrEq{symbols}{\tw@mk@mackeys}{#2}{}%
766   }%
767 }

```

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

```

768 \tikzset{
769   menukeys key symbol/.style={
770     rounded corners=0pt,
771     line width=0.1ex,
772     baseline={(0,0)},
773   },
774   menukeys thick/.style={line width=0.25ex},
775 }

```

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

```

776 \normalsize
777 \tw@make@key@box{shift}{%
778   \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
779     \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
780           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %

```

```

781             (0.3ex,1.2ex) -- cycle;
782     \end{tikzpicture}%
783 }
784 \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`

```

785 \tw@make@key@box{capslock@mac}{%
786   \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
787     \draw (0.3ex,0.7ex) -- (1.1ex,0.7ex) -- (1.1ex,1.2ex) -- %
788           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
789           (0.3ex,1.2ex) -- cycle;
790     \draw (0.3ex,0) rectangle (1.1ex,0.4ex);
791   \end{tikzpicture}%
792 }
793 \tw@make@key@box{capslock@win}{%
794   \begin{tikzpicture}[yscale=-1,yshift=-1.8ex,menukeys key symbol]
795     \draw (0.3ex,0) -- (1.1ex,0) -- (1.1ex,1.2ex) -- %
796           (1.5ex,1.2ex) -- (0.7ex,1.9ex) -- (-0.1ex,1.2ex) -- %
797           (0.3ex,1.2ex) -- cycle;
798   \end{tikzpicture}%
799 }
800 \tw@make@key@macro*{\capslock}

```

Here are the other macros:

`\tab`

```

801 \tw@make@key@box{tab@mac}{%
802   \begin{tikzpicture}[yshift=0.6ex,menukeys key symbol]
803     \draw [->] (0,0) -- (1em,0);
804     \draw (1em,-0.35ex) -- (1em,0.35ex);
805   \end{tikzpicture}%
806 }
807 \tw@make@key@box{tab@win}{%
808   \begin{tikzpicture}[yshift=0.1ex,menukeys key symbol]
809     \draw [->] (0.2em,0) -- (1.2em,0);
810     \draw (1.2em,-0.35ex) -- (1.2em,0.35ex);
811     \draw [<-] (0,1ex) -- (1em,1ex);
812     \draw (0,0.65ex) -- (0,1.35ex);
813   \end{tikzpicture}%
814 }
815 \tw@make@key@macro*{\tab}

```

`\esc`

`\oldesc`

```

816 \def\tw@mk@esc@win{Esc}
817 \tw@define@mackey{%
818   \def\tw@mk@esc@mac{esc}
819 }{%
820   \tw@make@key@box{esc@mac}{%
821     \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]

```



```

822         \draw [->] (0.5ex,0.5ex) -- ++(135:1.1ex);
823         \draw (0.5ex,0.5ex) ++(105:0.6ex) arc (105:-195:0.6ex);
824     \end{tikzpicture}%
825 }%
826 }
827 \tw@make@key@macro*{\esc}
828 \def\tw@mk@oldesc@win{Esc}
829 \tw@define@mackey{%
830     \def\tw@mk@oldesc@mac{esc}
831 }{%
832     \tw@make@key@box{oldesc@mac}{%
833         \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
834             \draw [->] (0.5ex,0.5ex) -- ++(45:1.1ex);
835             \draw (0.5ex,0.5ex) ++(15:0.6ex) arc (15:-285:0.6ex);
836         \end{tikzpicture}%
837     }%
838 }
839 \tw@make@key@macro*{\oldesc}

\ctrl
840 \providecommand\ctrlname{Ctrl}
841 \def\tw@mk@ctrl@win{\ctrlname}
842 \def\tw@mk@ctrl@mac{ctrl}
843 \tw@make@key@macro*{\ctrl}

\Alt
\AltGr 844 \def\tw@mk@Alt@win{Alt}
845 \tw@define@mackey{%
846     \def\tw@mk@Alt@mac{alt}%
847 }{%
848     \tw@make@key@box{Alt@mac}{%
849         \begin{tikzpicture}[yshift=-0.1ex,menukeys key symbol]
850             \draw (0,1ex) -- (0.5ex,1ex) -- (1ex,0.3ex) -- (1.8ex,0.3ex);
851             \draw (0.8ex,1ex) -- (1.8ex,1ex);
852         \end{tikzpicture}%
853     }%
854 }
855 \tw@make@key@macro*{\Alt}
856 \providecommand*{\AltGr}{Alt\,Gr}

\cmd
857 \def\tw@mk@cmd@win{%
858     \tw@mk@warning{'\string\cmd' only for Mac!}%
859 }
860 \tw@define@mackey{%
861     \def\tw@mk@cmd@mac{cmd}%
862 }{%
863     \tw@make@key@box{cmd@mac}{%
864         \begin{tikzpicture}[yshift=-0.15ex,menukeys key symbol]
865             \draw (0.5ex,0.7ex) -- (0.5ex,1.25ex) arc (0:270:0.25ex) -- %
866                 (1.25ex,1ex) arc (-90:180:0.25ex) -- (1ex,0.25ex) %
867                 arc (-180:90:0.25ex) -- (0.25ex,0.5ex) arc (90:360:0.25ex) %
868                 -- cycle;
869         \end{tikzpicture}%

```

```

870 }%
871 }
872 \tw@make@key@macro*{\cmd}

\Space
\SPACE 873 \providecommand*{\Space}{\expandonce{\rule{3em}{0pt}}}
874 \newcommand{\spacename}{Space}
875 \providecommand*{\SPACE}{\expandonce{\rule{2em}{0pt}}\spacename\rule{2em}{0pt}}}

\return
876 \tw@make@key@box{return@mac}{%
877 \begin{tikzpicture}[yshift=0.25ex,menukeys key symbol]
878 \draw [->, rounded corners=0.2ex] (1.25ex,1ex) -| %
879 (2ex,0) -- (0,0);
880 \end{tikzpicture}%
881 }
882 \tw@make@key@box{return@win}{%
883 \begin{tikzpicture}[menukeys key symbol]
884 \draw [->] (1ex,1.25ex) |- (0,0);
885 \end{tikzpicture}%
886 }
887 \tw@make@key@macro*{\return}

\enter
888 \def\tw@mk@enter@win{Enter}
889 \tw@make@key@box{enter@mac}{%
890 \begin{tikzpicture}[menukeys key symbol]
891 \draw (0,0) -- (0.5ex,0.5ex) -- (1ex,0);
892 \draw (0,0.55ex) -- (1ex,0.55ex);
893 \end{tikzpicture}%
894 }
895 \tw@make@key@macro*{\enter}

\winmenu
896 \def\tw@mk@winmenu@mac{%
897 \tw@mk@warning{'\string\winmenu' only for Windows!}%
898 }
899 \tw@make@key@box{winmenu@win}{%
900 \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
901 \draw (0,0) rectangle (1.5ex,1.8ex);
902 \draw (0.25ex,1.4ex) -- ++(1ex,0);
903 \draw (0.25ex,1ex) -- ++(1ex,0);
904 \draw (0.25ex,0.6ex) -- ++(1ex,0);
905 \end{tikzpicture}%
906 }
907 \tw@make@key@macro*{\winmenu}

\backspace
908 \tw@make@key@box{backspace}{%
909 \begin{tikzpicture}[yshift=0.65ex,menukeys key symbol]
910 \draw [<-,menukeys thick] (0,0) -- (1.35em,0);
911 \end{tikzpicture}%
912 }
913 \tw@make@key@macro*{\backspace}

```

```

\del
\backdel 914 \providecommand{\delname}{Del.}
          915 \def\tw@mk@del@win{\delname}
          916 \tw@define@macrokey{%
          917   \def\tw@mk@del@mac{\delname}%
          918 }{%
          919   \tw@make@key@box{del@mac}{%
          920     \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
          921       \draw (0,0) -- (1.5ex,0) -- (2ex,0.5ex) --%
          922         (1.5ex,1ex) -- (0,1ex) -- cycle;
          923       \draw (0.5ex,0.2ex) -- (1.1ex,0.8ex);
          924       \draw (0.5ex,0.8ex) -- (1.1ex,0.2ex);
          925     \end{tikzpicture}%
          926   }%
          927 }
          928 \tw@make@key@macro*{\del}
          929 \def\tw@mk@backdel@win{\delname}
          930 \tw@define@macrokey{%
          931   \def\tw@mk@backdel@mac{\delname}%
          932 }{%
          933   \tw@make@key@box{backdel@mac}{%
          934     \begin{tikzpicture}[yshift=0.2ex,menukeys key symbol]
          935       \draw (2ex,0) -- (0.5ex,0) -- (0,0.5ex) --%
          936         (0.5ex,1ex) -- (2ex,1ex) -- cycle;
          937       \draw (1ex,0.2ex) -- (1.6ex,0.8ex);
          938       \draw (1ex,0.8ex) -- (1.6ex,0.2ex);
          939     \end{tikzpicture}%
          940   }%
          941 }
          942 \tw@make@key@macro*{\backdel}

\arrowkeyup Lastly we define the arrow macros:
\arrowkeydown 943 \tw@make@key@box{arrowkeyup}{%
\arrowkeyleft 944   \begin{tikzpicture}[yshift=-0.2ex,menukeys key symbol]
\arrowkeyright 945     \draw [->] (0,0) -- (0,0.8em);
                946   \end{tikzpicture}%
                947 }
                948 \tw@make@key@macro{\arrowkeyup}
                949
                950 \tw@make@key@box{arrowkeydown}{%
                951   \begin{tikzpicture}[yshift=0.7em,menukeys key symbol]
                952     \draw [->] (0,0) -- (0,-0.8em);
                953   \end{tikzpicture}%
                954 }
                955 \tw@make@key@macro{\arrowkeydown}
                956
                957 \tw@make@key@box{arrowkeyright}{%
                958   \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
                959     \draw [->] (0,0) -- (0.8em,0);
                960   \end{tikzpicture}%
                961 }
                962 \tw@make@key@macro{\arrowkeyright}
                963
                964 \tw@make@key@box{arrowkeyleft}{%

```

```

965 \begin{tikzpicture}[yshift=0.5ex,menukeys key symbol]
966 \draw [->] (0,0) -- (-0.8em,0);
967 \end{tikzpicture}%
968 }
969 \tw@make@key@macro{\arrowkeyleft}

```

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

```

970 \newcommand{\arrowkey}[1]{%
971 \IfStrEq{^}{#1}{\arrowkeyup}{%
972 \IfStrEq{v}{#1}{\arrowkeydown}{%
973 \IfStrEq{<}{#1}{\arrowkeyleft}{%
974 \IfStrEq{>}{#1}{\arrowkeyright}{%
975 \tw@mk@error{Wrong value '#1' for \string\arrowkey\MessageBreak
976 Possible values are '^', 'v', '<' or '>'}%
977 }%
978 }%
979 }%
980 }%
981 }

```

Close the `\iftw@mk@definekeys`

```

982 \fi
983 </pkg>

```

7 Change history

v1.0	General: Initial version 1	v1.4	General: Extended color theme features. 1
v1.1	General: Improved manual 1		The <code>path...</code> styles now use the text color of the selected color theme (fix issue #16). 1
	Load <code>xcolor</code> before <code>menukeys</code> . 13		<code>\backdel</code> : Added <code>\backdel</code> 35
	<code>\directory</code> : Renamed <code>\path</code> to <code>\directory</code> because it crashes with <code>biblatex</code> 30		<code>\oldesc</code> : Fixed direction of <code>\escmac</code> ; added <code>\oldesc</code> 32
v1.1a	<code>\newmenumacro</code> : Added a line to make a new macro robust. . . . 29	v1.5	General: New option <code>hyperrefcolorlinks</code> 17
	<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. 28	v1.6	General: <code>hyperrefcolorlinks</code> obsolete 17
v1.2	General: Added <code>\normalsize</code> before symbol definitions to make the <code>ex</code> unit available . . . 1		Don't load <code>catoptions</code> 14
	Added <code>\SPACE</code> and <code>\spacename</code> . 1		Load order no longer important 4
	Fixed GitHub issues #9, #10, #11, #13, #17, #24 and #26 . 1		<code>\newmenumacro</code> : use <code>\NewDocumentCommand</code> 29
	Tidy up version and date 1		<code>\providenumacro</code> : use <code>\ProvideDocumentCommand</code> . . 29
	<code>\tw@define@menu@macro@</code> : Added <code>\leavevmode</code> 28		<code>\renewmenumacro</code> : use <code>\RenewDocumentCommand</code> 29
	Replaced <code>\edef</code> by <code>\protected@edef</code> 28		<code>\tw@define@menu@macro@</code> : Don't use <code>\NewDocumentCommand</code> . . . 28
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	v1.6.1	<code>\newmenumacro</code> : default handled by <code>\tw@define@menu@macro</code> 29
	Replaced obsolete <code>\tikzstyle</code> . 1		<code>\providenumacro</code> : default handled by <code>\tw@define@menu@macro</code> 29
v1.2c	<code>\tw@define@menu@macro@</code> : Replaced <code>\protected@edef</code> by <code>\def</code> 28		<code>\renewmenumacro</code> : default handled by <code>\tw@define@menu@macro</code> . . 29
v1.3	General: Added TikZ-styles for the key symbols. 1		<code>\tw@define@menu@macro</code> : Handles default input separator. 28
	Improved key symbols. 1		<code>\tw@define@menu@macro@</code> : No x-type expansion on the separator to call the loop 28
			Renamed from <code>\tw@define@menu@macro</code> 28
		v1.6.2	General: changed <code>\hspace</code> to <code>\kern</code> 25, 26

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		371, 374, 393, 398, 402, 405, 423, 428, 432, 435, 451, 475, 498, 521, 540, 556, 585, 628, 659	
<code>\@ifdefinable</code>	632		
<code>\@tw@hyphenatepaths@warnigfalse</code> .	580		
<code>\@tw@hyphenatepaths@warnigtrue</code> .	568		
A		D	
<code>\Alt</code>	844	<code>definekeys</code> (option)	4
<code>\AltGr</code>	844	<code>definemenumacros</code> (option)	4
<code>angularkeys</code> (style)	6	<code>\del</code>	914
<code>angularmenus</code> (style)	5	<code>\delname</code> ...	13, 914, 915, 917, 929, 931
<code>\arrowkey</code>	13, 970	<code>\directory</code>	4, 723
<code>\arrowkeydown</code>	943, 972	<code>\drawtikzfolder</code>	
<code>\arrowkeyleft</code>	943, 973	9, 595, 606, 612, 618, 624
<code>\arrowkeyright</code>	943, 974	E	
<code>\arrowkeyup</code>	943, 971	<code>\enter</code>	888
B		<code>\esc</code>	816
<code>\backdel</code>	914	<code>\exp</code>	12, 50, 66
<code>\backspace</code>	908	<code>\ExplSyntaxOff</code>	71
<code>blacknwhite</code> (theme)	11	<code>\ExplSyntaxOn</code>	10
<code>\bool</code>	62	F	
C		<code>\font</code>	583
<code>\capslock</code>	785	G	
<code>\changemenucolor</code>	12, 131	<code>gray</code> (theme)	11
<code>\changemenucolortheme</code>	10, 152	<code>\group</code>	48, 51
<code>\changemenuelement</code>		H	
.....	10, 321, 605, 611, 617, 623	<code>\hypersetup</code>	109
<code>\cmd</code>	857	<code>hyphenatepaths</code> (style)	8
<code>\color</code>	334, 362, 367, 371, 374, 393, 398, 402, 405, 423, 428, 432, 435, 451, 475, 498, 521, 545, 547, 556, 584	<code>hyphenatepathswithblackfolder</code> (style)	8
Color themes:		<code>hyphenatepathswithfolder</code> (style) ..	8
<code>blacknwhite</code>	11	I	
<code>gray</code>	11	<code>\if@tw@hyphenatepaths@warnig</code>	567, 577
<code>\copymenucolortheme</code>	11, 131	<code>\IfNoValueTF</code>	642
<code>\copymenustyle</code>		<code>\iftw@mk@definekeys</code>	727
.....	10, 303, 604, 610, 616, 622	<code>\iftw@mk@definemenumacros</code>	722
<code>\cs</code>	11, 12, 13, 15, 19, 26, 29, 33, 37, 43, 55, 56	<code>\iftw@mk@hyperrefcolorlinks</code>	106
<code>\csletcs</code>	308, 309, 310, 311, 312, 313, 314, 315	<code>\int</code>	24, 46, 52, 61, 64
<code>\ctrl</code>	840	K	
<code>\ctrlname</code>	13, 840, 841	<code>\keys</code>	4, 723
<code>\CurrentMenuElement</code>		L	
..	10, 189, 192, 195, 198, 221, 224, 227, 230, 334, 362, 367,	<code>\l</code>	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, 13
\menu	4, 723
menus (style)	5
N	
\newmenucolortheme	11, 120, 167, 168
\newmenumacro	12, 703, 723, 724, 725
\newmenustyle	9, 234, 269
\newmenustylesimple	9, 234, 234
O	
\oldesc	816
Options:	
definekeys	4
definenumacros	4
mackeys	4, 13
os	4
os (option)	4
P	
\PassOptionsToPackage	110
paths (style)	7
pathswithblackfolder (style)	7
pathswithfolder (style)	7
\prg	14
\protected	634
\providenumacro	12, 703
\providemenustyle	10, 234, 289
\providemenustylesimple	10, 234, 254
R	
\relsize	354, 385, 415, 446, 454, 466, 479, 493, 501, 512, 524, 536, 551
\renewmenucolortheme	12, 120, 149
\renewmenumacro	12, 703
\renewmenustyle	10, 234, 278, 282, 299, 708, 719
\renewmenustylesimple	10, 234, 243, 247, 264
\return	876
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)	6
shadowedroundedkeys (style)	6
\shift	776
\SPACE	873
\Space	873
\spacename	13, 874, 875
Styles:	
angularkeys	6
angularmenus	5
hyphenatepathswithblackfolder	8
hyphenatepathswithfolder	8
hyphenatepaths	8
menus	5
pathswithblackfolder	7
pathswithfolder	7
paths	7
roundedkeys	6
roundedmenus	5
shadowedangularkeys	6
shadowedroundedkeys	6
typewriterkeys	7
T	
\tab	801
\tikzset	341, 346, 377, 408, 438, 458, 486, 505, 531, 768
\tl	11, 14, 17, 25
\tw@current@color@theme	165, 682
\tw@current@style	204, 205, 210
\tw@declare@style	201, 272, 274, 284, 286, 292, 294, 359, 390, 420
\tw@declare@style@extra@cargs	201
\tw@declare@style@simple	174, 237, 239, 249, 251, 257, 259, 448, 472, 495, 518, 538, 555, 569
\tw@default@input@sep	627, 643
\tw@default@post	169, 175, 202
\tw@default@pre	169, 175, 208
\tw@default@sep	169, 175, 208
\tw@define@macro	763, 817, 829, 845, 860, 916, 930
\tw@define@menu@macro	629, 705, 712, 716
\tw@define@menu@macro@	629
\tw@make@color@theme	112, 122, 129
\tw@make@key@box	728, 777, 785, 793, 801, 807, 820, 832, 848, 863, 876, 882, 889, 899, 908, 919, 933, 943, 950, 957, 964
\tw@make@key@macro	738, 784, 800, 815, 827, 839, 843, 855, 872, 887, 895, 907, 913, 928, 942, 948, 955, 962, 969
\tw@menu@list	687, 691, 696
\tw@mk@Alt@mac	846
\tw@mk@Alt@win	844
\tw@mk@backdel@mac	931
\tw@mk@backdel@win	929
\tw@mk@cmd@mac	861
\tw@mk@cmd@win	857
\tw@mk@ctrl@mac	842
\tw@mk@ctrl@win	841

<code>\tw@mk@del@mac</code>	917	<code>\tw@mk@test@input@sep</code> .	632, 634, 685
<code>\tw@mk@del@win</code>	915	<code>\tw@mk@trimspaces</code>	11, 636, 692
<code>\tw@mk@center@win</code>	888	<code>\tw@mk@warning</code>	72,
<code>\tw@mk@error</code>	72, 99, 103, 124,		107, 262, 297, 578, 718, 858, 897
	135, 148, 154, 158, 242, 277,	<code>\tw@mk@warning@noline</code>	72
	306, 318, 323, 337, 650, 707, 975	<code>\tw@mk@winmenu@mac</code>	896
<code>\tw@mk@esc@mac</code>	818	<code>\tw@mk@xifinsetTF</code>	15, 635
<code>\tw@mk@esc@win</code>	816	<code>\tw@set@tikz@colors</code>	341
<code>\tw@mk@exp@Nnno</code>	12, 643	<code>\tw@typewriterkeys@curr@elem</code> ...	
<code>\tw@mk@gobble@cargs</code>			539, 545, 547
	83, 244, 265, 279, 300	<code>typewriterkeys (style)</code>	7
<code>\tw@mk@iflastindris</code>	19, 660		
<code>\tw@mk@indrisloop</code>	56, 696		
<code>\tw@mk@indrisnr</code>	26, 662, 670		
<code>\tw@mk@mackeys</code>	102, 764, 765		
<code>\tw@mk@oldesc@mac</code>	830		
<code>\tw@mk@oldesc@win</code>	828		
<code>\tw@mk@os</code>	98, 742		
<code>\tw@mk@string</code>			
	13, 657, 695, 704, 715, 740,		
	742, 745, 747, 750, 752, 756, 758		
<code>\tw@mk@tempa</code> .	81, 84, 85, 688, 692, 696		
<code>\tw@mk@tempb</code>	81, 695, 696		

U

<code>\usemenucolor</code>	10, 164,
	334, 342, 343, 344, 362, 367,
	371, 374, 393, 398, 402, 405,
	423, 428, 432, 435, 451, 454,
	469, 475, 479, 498, 501, 515,
	521, 524, 545, 547, 551, 556,
	560, 584, 589, 606, 612, 618, 624

W

<code>\winmenu</code>	896
-----------------------------	-----