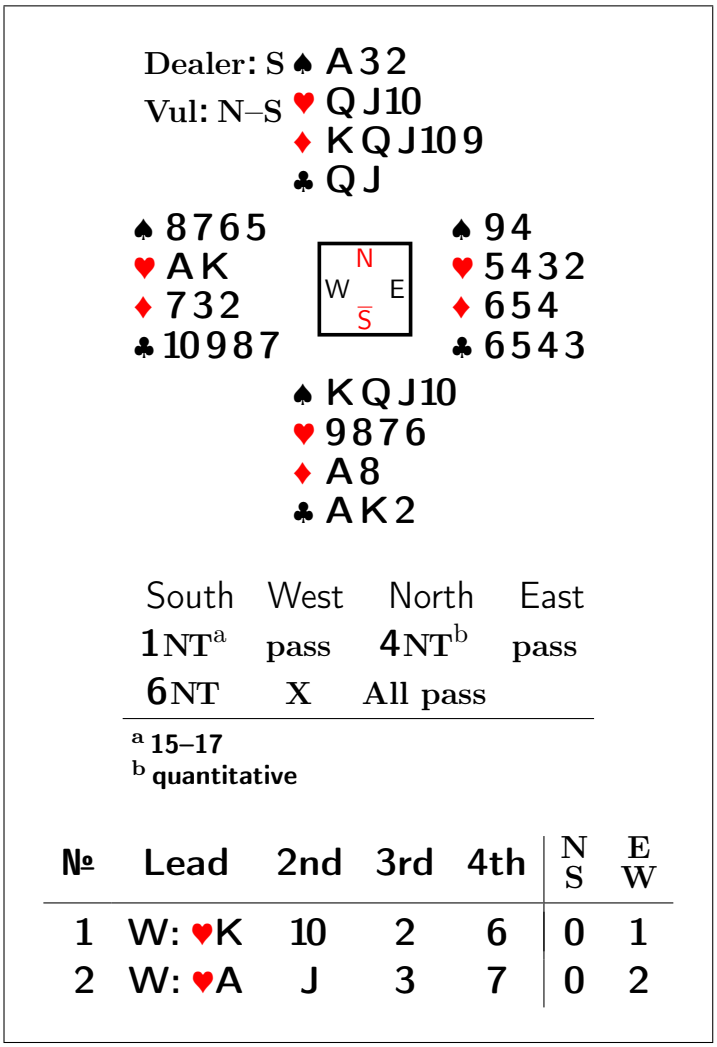
The OneDown package

www.ctan.org/pkg/onedown

Jacob Wiersma
jack46@online.de

v1.0 from 2018/05/23



South	West	North	East
1NT[a]	pass	4NT[b]	pass
6NT	X	All pass	

[a] 15–17
[b] quantitative

№	Lead	2nd	3rd	4th	N S	E W
1	W: ♥K	10	2	6	0	1
2	W: ♥A	J	3	7	0	2

Abstract

This package implements commands and environments to typeset various bridge diagrams, with or without a `bidding` sequence. It offers following features:

- It is possible to use an own font and/or a font-size with which the diagrams will be typeset independently from the main font used in the document. This also allows an easy production of overhead slides and for digital projection. Different fonts can be used for e.g. the bidding diagram, its header, the compass, the hands etc. Most diagrams can be centered both horizontally and vertically.
- A special feature is the automated check on consistency of suits and hands. If a hand holds more than 13 cards an error is printed, if there are less then 13 cards a warning. If a suit over the 4 hands has more than 13 cards or if a card appears more than once an error like Error: Card ♠8 occurs 2 times is printed. These warnings and error messages are controlled by the `err` and `warn` options, e.g. when loading the package.
- The output of the implemented bridge terms like **Double** (which you get by calling the command `\double`) are multilingual and get translated automatically. When the language `german` is active the command `\double` would produce **Kontra**. Also the basic symbols like A(ce), K(ing), Q(ueen) and J(ack) are multilingual. So ♦ A K Q J would automatically become ♦ A K D B in a German text.
- It is possible to add annotations to a card diagram, like the board number, the dealer or the vulnerability etc. on several positions in the diagram (if a board number is given, the dealer and vulnerability are computed automatically). One can also add explanations to the `bidding` diagram, as well as the real world names of the bidders.
- There are two specials: a command to typeset a quiz with answers, and an environment to typeset the sequence of playing tricks, where also the total number of tricks won by each side is calculated and displayed.

Contents

1 Preface

I am neither a good bridge player nor a good package writer. But I like to read about bridge and when I write about bridge (sometimes a funny short story, sometimes some training exercises for beginners) I do feel the need for an appropriate tool to support that. Surprisingly enough, there exists no comprehensive package on CTAN for typesetting bridge diagrams. For all those people who feel more or less the same as I do, there is this package called OneDown. As some say: *one down is good bridge*[1], I hope that OneDown is a good bridge package.

You can generate this documentation by running

```
pdflatex --shell-escape onedown.dtx
makeindex -s gind.ist onedown.idx
makeindex -s gglo.ist -o onedown.gls onedown.glo
pdflatex --shell-escape onedown.dtx
pdflatex --shell-escape onedown.dtx
```

Use:

```
pdflatex --shell-escape '\AtBeginDocument{\NoColortrue}\input{onedown.dtx}'
```

as last run to get a pdf for printing on a monochrome printer. The `--shell-escape` flag is necesary to generate the list of user commands. If you think this is too dangerous, then run pdflatex without this flag and you will get the docmentation without the list of user command (of about 1 page). In any case you'll find a multi-page reference overview of all commands in `onedown-ref.pdf`.

2 Introduction

There must be a lot of bridge players who also use LaTeX to typeset their documents. And it is almost incredible that on CTAN there exist no modern package with a decent documentation that supports this.

[1] please, don't discuss the truth-value of this statement with me

In 1990 Kees van der Laan [1] published an article in TUGBoat[2] in which he describes how well the TEX-machinery is able to produce beautiful bridge diagrams. Based on this article and examples, Johannes Braams put these commands together in a style file[3] and added more. Some time later René Steiner and Thomas Hof produced the `bridge-i` and the `kibitzer`[4] style files, in which they made a lot of enhancements. Also others made some efforts in this direction.[5]

Around 2005 I used these style files for some tiny projects. The quality of the output was splendid. Putting the text and diagrams together was not always easy and the documentation was poor. In 2015, after a long pause, I had to produce bridge texts again. I enhanced some of the existing stuff and made ad hoc changes in the code, which led to smaller and greater catastrophes. Summer 2016 I decided to write a new package, based on the work of the previously mentioned persons. I called it OneDown. The central goal was to offer a user friendly package with detailed documentation. For example you don't need to say `\setlength{\handskip}{5mm}` or `\def\handskip{5mm}` but rather call the command `\handskip{1em}`. Not only the call is somewhat friendlier, more important, setting the width in terms of the font used, it will automatically adapt its size accordingly to font and font-size changes. OneDown features:

- Sizing of diagrams relative to font and font-size.
- The font-sizes of the diagrams and text are independent.
- Automated translation of all important bridge terms.
- Diagrams can optionally contain information about the dealer, who is vulnerable etc.

The OneDown package is designed to be used for typesetting texts that have to do with the game of bridge. It provides not only simple commands like `\Sp` which produces the spade symbol ♠. Also complete card diagrams with the hands of the **North**, **East**, etc. player

[2] http://tug.org/TUGboat/Articles/tb11-2/tb28laan.pdf

[3] `bridge.sty`, last version v1.7c, 1994/12/20

[4] both v1.0, 1995/04/06

[5] Antony Lee released his package `bridge` in 2012 and Gordon Bower his package `grbbridge` in 2013. Both are very interesting but offer only limited features and are not on CTAN at the time of writing. See http://www.bridgebase.com/forums/topic/51967-latex-package-for-typesetting-bridge-related-stuff/

can be defined in several ways. One can select which hands are to be shown. Bidding diagrams can be shown stand-alone or in connection with one or more hands. In `bidding` diagrams annotations are possible.

3 Usage

3.1 Initialization

3.1.1 Requirements

The package OneDown depends on several other packages, such as `ifthen`, `translations` or `xspace`. All these packages get loaded automatically if not already used in your document. For a complete overview of all required packages, refer to page 31. All the packages are loaded without any option, so the risk of an option clash should be low: Just load your package with options *before* OneDown.

Furthermore, for the several languages that OneDown supports, there are the dictionary files with translations of the specific bridge terms. These dictionaries follow the naming convention: `ODw-`⟨*language*⟩`.trsl` and are included in the bundle. The name of the ⟨*language*⟩ is generally the same as the name that you use as option for `babel`. There is one exception: the Norwegian language uses `norsk` for `babel` and `norwegian` for `translations`. But all the same, the `norwegian` dictionary is automatically loaded when `norsk` is used.

Should you make a dictionary for a language that is not provided yet, or have corrections for an existing one, please send it to the maintainer, so it can be added to the bundle.

3.1.2 Loading the package

Simply say `\usepackage{onedown}` in the preamble of your document if you want to load OneDown with its default settings.

Warning: OneDown loads all necessary `ODw`-dictionaries automatically at the begin of the document. In order to know which languages must be loaded, these must be specified *before* package `onedown` is loaded. In general this means that if you use `babel` (or `polyglossia`) you must load it before package OneDown. If for some reason you cannot or do not want to do that, you can load any `ODw`-dictionary if you put the command: `\LoadDictionaryFor{{`⟨*language*⟩`}}{ODw}` in your preamble, provided

that the dictionary is in the TEX-path. For a discussion about the caveats of using e.g. babel, refer to section 3.1.4.

3.1.3 Options

To change the behaviour of OneDown one can load the package with certain options: `\usepackage[`⟨*options*⟩`]{onedown}`. Of course this 'option loading' takes place in the preamble. But it is also possible to set (or change) options within the document by calling the macro `\setdefaults`. This macro uses the same ⟨*key*⟩=⟨*val*⟩ syntax as is used for the options and offers some more keys that cannot be used when loading the package. Refer to page 26 for details.

As said before, the package loads its options using the ⟨*key*⟩=⟨*val*⟩ syntax. These options deal with:

colors `colors=0|1|2|4A|4B`

The color in which the card symbols will be printed. The color options are **0** (black only), **1** (black and white) **2** (black and red), **4A** (green, orange, red and blue), **4B** (black, orange, red and green). We also defined some synonyms, as shown in the table below.

Thus loading the package with `\usepackage[colors=X]{onedown}` will print

x=0: ♣, ♦, ♥ and ♠ (synonyms: `mono`, `black`)

x=1: ♣, ♢, ♡ and ♠ (synonyms: `b+w`)

x=2: ♣, ♦, ♥ and ♠ (synonyms: `b+r`)

x=4A: ♣, ♦, ♥ and ♠ (synonyms: `4a`, `fourA`)

x=4B: ♣, ♦, ♥ and ♠ (synonyms: `4b`, `fourB`)

The default is `colors=2` for printing in black and red.

err, warn `err=on|off` `warn=|on|off`

These options regulate which messages are to be output. These messages have to do with the consistency of cards in a suit, in a hand or combined hands. It is an error when a hand has more that 13 cards, or when the same card occurs twice or more in a hand or a deal. With the option `err=on` (which is the default) these error messages appear as output. With `err=off` you can suppress that. On the other hand when a suit has less than 13 cards, this must not necessarily be wrong. Maybe only some

cards are to be shown, e.g. in an example concerning a finesse. Or when only e.g. `E-W` hands are concerned, not all cards of the deck will be specified. These situations will be caught by setting `warn=on`. To suppress these spurious warnings use `warn=off`, which is the default. Synonyms for `on` are `1` and `true`. Synonyms for `off` are `0` and `false`. This also applies for other keys that do not control a package option.

3.1.4 Active Characters: a Warning

OneDown uses the `tranlation` package to automatically translate oft appearing brigde terms like e.g. *declarer*. It does so by looking up these terms in the special `ODw`-dictionary for the active language. The current OneDown version supports English, German, Dutch, French, most Scadinavian languages and Turkish. Some dictionaries may not be complete or may contain errors, please send corrections/additions to the maintainer.

Warning for people using active characters.

Some language packages fiddle around making characters active. This can have unexpected influence on OneDown. The `=` sign is used when loading the package `onedown` to specify options. It appears also in calls like `\setdefaults{warn=on}`. We also use the following characters as tokens for optional arguments with the meaning as shown in this list:

- `*` to center diagrams or print a long or capitalized text
- `!` special action like short names of vertical layout
- `-` to hide what would normally been shown
- `+` to show what would normally be hidden

To give you an idea what e.g. babel can cause we cite from *The Turkish style for babel*:

> Turkish typographic rules specify that a little 'white space' should be added before the characters ':', '!' and '='. In order to insert this white space automatically these characters are made `\active`, so they have to be treated in a special way.

So babel-Turkish makes the equal sign and exlamation mark active. This leads to errors when you call e.g. `\setdefaults{warn=on}` or `\hand!`.

If you do not need any character to be active, then load this language with `\usepackage[turkish,shorthands=]{babel}`. If you do need the shorthand then you must disable it every time you have to use e.g. the = character as a normal character by:

```
\shorthandoff{=}% Make '=' not active any more
  \setdefaults{warn=on}
\shorthandon{=}% Restore '=' to active again
```

4 User Commands

4.1 Overview

In the next sections we give a short description of all the user commands and environments that are defined in OneDown. The commands marked with ML are multilingual. I.e. the text they typeset gets translated automatically into the active language.

In order to make sure that the example diagrams do not disturb the page layout of this document too much, we scaled them down to `footnotesize`. Sometimes the output of a command is shown as an example. This output is framed in this document like this, just to recognize it easily as an output example. In an accompanying file[6] with examples one can find in more detail how these commands are used and what they produce.

We have loaded the package OneDown with the default option for colors, giving us black and red. Furthermore, when we descibe macros, we use a colored frame that also shows the output of the commend. Some commands have optional tokens that produce an output that differs from the naked version. The output of tokenized calls in the command overview is shown in parenthesis.

4.1.1 The Compass

The compass N W E S is not available as a user command itself, but it is used in all user commands that draw a card diagram. It has some special features.

[6] onedown-examples.pdf

- It can mark the dealer (**North**) 

- It can write the vulnerable side (**North–South**) in red 

- it can put something (a board number) in the middle 

The machinery is intelligent enough to calculate the dealer and vulnerability from the board number. When only `black` or `b+w` is selected as option for colors, then the vulnerable side is written in italics rather colorized. Because underlining the **South**-hand would interfere with the compass frame, we overline it. With the command `\setdefaults` one can customize the look of the compass. In particular, if you want to print the actual board number, specified by calling `\boardnr{Nr}`, you can achieve that by calling `\setdefaults{compmid=\boardtext}`. In the accompanying file with examples you'll find more examples about `\setdefaults`.

4.1.2 Hooks

We use kind of hooks to

1. change the font or the font-size. These are discussed in chapter *Sizing and Fonts* on page 19.
2. add commentary information to card diagrams. These are discussed in chapter *Conditions in Diagrams* on page 23
3. There is one other hook to enable the user to add something to the compass. This is done by calling `\setdefaults{compmid=<Text>}`.

4.2 Basic Symbols

In this section we show the predefined commands that produce terms that occur often in bridge text. On page 21 an easy way is shown to redefine them as to use a different variant of the term in question.

The next 5 macros are shorthands for the suit symbols and **NT**:

\Cl \Cl ♣

\Di \Di ♦

\He \He ♥

\Sp \Sp ♠

\NT ML \NT **NT**

The next 4 macros typeset the non-bid calls

\pass ML \pass **Pass**

\allpass ML \allpass **All pass**

\double ML \double **Double**

\redouble ML \redouble **ReDouble**

The next 6 macros typeset the directions that identify dealer (or player) and the axis that is vulnerable.

\north ML \north **North**

\east ML \east **East**

\south ML \south **South**

\west ML \west **West**

\northsouth ML \northsouth **North–South**

\eastwest ML \eastwest **East–West**

The next 4 macros typeset the *unit* of the points when valuing a bridge hand. If one or more of these items do not appear in your bridge world, don't argue them, Just don't use them!

\HCP ML \HCP **HCP**

\LP ML \LP **LP**

\DP ML \DP **DP**

\TP ML \TP **TP**

The next 5 commands produce the abbreviations for several forcing expressions.

\GF ML \GF **GF**

\SF ML \SF **SF**

\NMF ML \NMF **NMF**

\TSF ML \TSF **3rd SF**

\FSF ML \FSF **4th SF**

4.2.1 Header of the Bidding Table

\namesNS `\namesNS{⟨N-name⟩}{⟨S-name⟩}`

Defines the real world names for the **North** and the **South** player. If they are defined, they appear in the **N**- and the **S**-column of the `bidding` diagram.

\namesEW `\namesEW{⟨E-name⟩}{⟨W-name⟩}`

Same as `\namesNS` but now for the **East** and the **West** player.

4.3 The Card Diagrams

4.3.1 The hands of the players

Before a card diagram can be shown, one must specify the cards that each player holds. With the various ⟨*player*⟩`hand`-commands one can do this. The suits they define are only shown when a `show`-command is issued after they have been defined. The `show`-commands are discussed on page 16.

`\northhand` `\northhand[`⟨*v-offset*⟩`]{`⟨*Sp*⟩`}{`⟨*He*⟩`}{`⟨*Di*⟩`}{`⟨*Cl*⟩`}`

`\easthand` `\easthand[`⟨*h-offset*⟩`]{`⟨*Sp*⟩`}{`⟨*He*⟩`}{`⟨*Di*⟩`}{`⟨*Cl*⟩`}`

`\southhand` `\southhand[`⟨*v-offset*⟩`]{`⟨*Sp*⟩`}{`⟨*He*⟩`}{`⟨*Di*⟩`}{`⟨*Cl*⟩`}`

`\westhand` `\westhand[`⟨*h-offset*⟩`]{`⟨*Sp*⟩`}{`⟨*He*⟩`}{`⟨*Di*⟩`}{`⟨*Cl*⟩`}`

The command `\northhand` defines the cards (all 4 suits) for the **N**-player. `\easthand`, `\southhand` and `\westhand` do that for the **E**-, **S**- or **W**-player. These commands have 4 mandatory arguments in which the cards of the 4 suits are specified. In all suit commands where card ranks are issued, one must use **T** to denote the value **10**. On output, some kerning takes care that the output looks like 10 and not like a **1** followed by a **0**. So `\suit{AKJT8}` produces A K J 10 8. These commands also have an optional argument, an offset which by default is 0pt. This offset is meant to finetune the layout of the hands in the card diagrams. They change the distance between a hand and the compass. `\northhand` and `\southhand` have a *vertical* offset, whereas `\easthand` and `\westhand` have a *horizontal* one. A positive value moves away from the compass.

4.3.2 The single hand

`\hand` `\hand[*!-][`⟨*pos*⟩`]{`⟨*Sp*⟩`}{`⟨*He*⟩`}{`⟨*Di*⟩`}{`⟨*Cl*⟩`}`

This macro typesets the cards of one single hand, either vertically or horizontally. There are 4 mandatory arguments defining the 4 suits. With 2 optional tokens '`*`' resp. '`!`' one can typeset the hand with some special features:

- `\hand*` typesets a hand horizontally, centered

- `\hand!` typesets a hand vertically, left aligned
- `\hand*!` typesets a hand vertically, centered

For vertical hands the optional argument `pos` (default= **c**) controls the horizontal alignment. Without a token, the hand is typeset horizontally, left aligned: The call `\hand{AK2}{T85}{AQT6}{A42}` typesets the hand horizontally like:

♠ A K 2 ♥ 10 8 5 ♦ A Q 10 6 ♣ A 4 2 ,

whereas the `\hand!` version produces

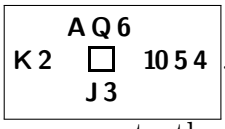
.

The third optional token, a '-' suppresses all output and saves the stuff for later use. This is used e.g. in `\expertquiz`.

4.3.3 Only one suit

`\onesuitAll` `\onesuitAll[*!]{⟨N⟩}{⟨S⟩}{⟨E⟩}{⟨W⟩}`

Typesets the cards of 1 suit for all players. This command has 4 mandatory arguments defining the cards of the 4 players. There are 2 optional tokens. With `\onesuitAll*` the output is centered, with `onesuitAll!` the cards are placed around a `NESW` compass. Without the '`!`'-token a small box (□) is used instead. Thus the macro call `\onesuitAll{AQ6}{J3}{T54}{K2}` produces



.

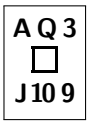
Please note the order of the players in the arguments: the first 2 denote the **North** and **South** hand. The last 2 denote the **East** and **West** hand. We choose it this way so you can easily cut and paste one pair from the `\onesuitAll`, or extend `\onesuitNS` to showing all hands.

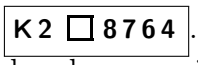
`\onesuitNS` `\onesuitNS[*!]{⟨N⟩}{⟨S⟩}`

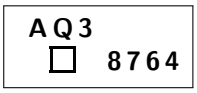
`\onesuitEW` `\onesuitEW[*!]{⟨E⟩}{⟨W⟩}`

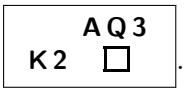
`\onesuitNE` `\onesuitNE[*!]{⟨N⟩}{⟨E⟩}`

`\onesuitNW` `\onesuitNW[*!]{⟨N⟩}{⟨W⟩}`

These commands are similar to `\onesuitAll` but have only 2 mandatory arguments. The command `\onesuitNS{AQ3}{JT9}` typesets 

and `\onesuitEW{8764}{K2}` will produce 

. Please note that at the latter the cards for the **East** hand appear in the first argument. Finally `\onesuitNE{AQ3}{8764}` produces 

and `\onesuitNW{AQ3}{K2}` 

.

`\suit` `\suit[⟨suit symbol⟩]{⟨cards⟩}`

This command has 1 optional argument denoting a suit symbol and 1 mandatory argument, defining the cards of the suit. `\suit{AQJ7}` by default produces A Q J 7. When the German language is active it would produce A D B 7. Using the optional argument like in `\suit[\Di]{AQJ7}` will produce: ♦ A Q J 7.

4.3.4 Showing Card Diagrams

`\showAll` `\showAll[*+][⟨pos⟩]`

`\showNS` `\showNS[*+][⟨pos⟩]`

`\showEW` `\showEW[*+][⟨pos⟩]`

`\showNE` `\showNE[*+][⟨pos⟩]`

`\showNW` `\showNW[*+][⟨pos⟩]`

All `show`-commands have two optional tokens, a '`*`' which centers the output and a '`+`' which also displays a `bidding` diagram next to the card diagram. This `bidding` diagram must have been defined before, see page 28. They also have one optional argument that defines the aligning. Its default is **c**. `\showAll` typesets a card diagram with the `NESW` compass with **N** in top and the hands of the 4 players surrounding it. These hands must have been defined before by calling `\northhand`

etc. Hands that are not defined are left empty. Optionally some conditions (like the dealer or who is vulnerable etc.) can be added to the diagram by using the commands described in section *Diagram Conditions*. Please note that when the **North** or **South** hand contains a long suit that extends beyond the NESW compass, this might collide with these extra texts. You can correct that with the optional offset parameter of the condition commands (see page 17).

The other commands are similar to `\showAll` but typeset only the hands of the players that are represented in the name of the command: **N–S**, **E–W**, **N–E** and **N–W**.

4.3.5 Showing Card Diagrams with Bidding

4.3.6 Diagram Conditions

`\headlinetext` `\headlinetext{⟨text⟩}`

`\footlinetext` `\footlinetext{⟨text⟩}`

These commands have 1 mandatory argument: the text that defines the annotation that is to be added to a card diagram. The text can be on more than one line, just separate them with a `\par` or `\newline`[7]. `\headlinetext` places the annotation above the diagram, `\footlinetext` below it.

`\leftupper` `\leftupper[⟨h-offset⟩]{⟨line1⟩}{⟨line2⟩}{⟨line3⟩}`

`\leftlower` `\leftlower[⟨h-offset⟩]{⟨line1⟩}{⟨line2⟩}{⟨line3⟩}`

`\rightupper` `\rightupper[⟨h-offset⟩]{⟨line1⟩}{⟨line2⟩}{⟨line3⟩}`

`\rightlower` `\rightlower[⟨h-offset⟩]{⟨line1⟩}{⟨line2⟩}{⟨line3⟩}`

These commands have 1 optional argument (default 0pt) with which you can add some extra horizontal space if hand and legend collide, and 3 mandatory arguments: the lines of text that are added as conditions to the card diagram. Both `\leftupper` (`\rightupper`) place their text in the left- (right-) upper corner of the diagram. The top line will be aligned with the (inner) top of the diagram. `\leftlower`

[7] Using '\\' instead produces a misleading error: **! Missing } inserted...**

(\rightlower) are similar, but place their text at the lower corner of the diagram. The last line is aligned with the (inner) bottom of the diagram. For an empty line you must issue an empty argument. With a positive offset, \leftupper and \leftlower shift to the left whereas \rightupper and \rightlower shift to the right. I.e. they shift away from their neighbouring hand.

\dealer `\dealer[⟨text⟩]`

\vulner `\vulner[⟨text⟩]`

Both commands have 1 optional argument. If present it sets (and prints) the internal corresponding variable to this value, otherwise it only outputs the value of this internal variable.

\dealertext `\dealertext[⟨text⟩]`

\vulnertext `\vulnertext[⟨text⟩]`

These commands have also 1 optional argument. If present e.g. \dealertext[\North*] this text is output in the form **Dealer: North**. If the German language is active then the call \dealertext[\South*] produces the text **Teiler: Süd**. Calling \dealertext without an argument outputs the predefined text, which can be set with \dealer.
Example: \dealer[Jacob]\dealertext produces **Dealer**: Jacob.

\boardnr `\boardnr{⟨Nr⟩}`

The macro \boardnr has 1 mandatory argument. If it is a number, it is considered to be the board number. The dealer and which side is vulnerable is then calculated from it and stored in the appropriate variables. If it is not a positive integer, it is considered user-defined text and will be stored and used *as is*.

\boardtext ML `\boardtext[*]`

The macro \boardtext has 1 token and no arguments. \boardtext retrieves only the board number (stored by calling \boardnr). \boardtext* outputs the board number with some additional (multilingual) text. \boardnr{23}\boardtext produces **23** whereas \boardtext* would produce Board: **23**. Note that \boardnr can have a non-integer argument. \boardnr{Fun}\boardtext produces Fun and with \boardtext* it would

produce Board: Fun.

4.3.7 Sizing and Fonts

\handskip `\handskip{⟨length description⟩}`

This command has 1 mandatory argument: a **text** describing a length. `\handskip` enlarges the distance (default `1em`) between the rightmost hand and the `bidding` diagram. A negative value diminishes the distance.

\bidderfont `\bidderfont{⟨font description⟩}`

\compassfont `\compassfont{⟨font description⟩}`

\gamefont `\gamefont{⟨font description⟩}`

\legendfont `\legendfont{⟨font description⟩}`

\namefont `\namefont{⟨font description⟩}`

\otherfont `\otherfont{⟨font description⟩}`

These commands all have 1 mandatory argument: a **description** of the font to be used. In the list below the command names are typeset in their default font.

- bidderfont: Used for the player-names in the `bidding` diagram. The default is `\mdseries\sffamily`.
- compassfont: Used for the directions and the "midvalue" in the compass. The default is `\mdseries\sffamily`.
- **gamefont**: Used for card diagrams, hands and suits. The default is `\bfseries\sffamily`.
- legendfont: Used for the conditions in card diagrams. The default is `\mdseries\rmfamily`.
- *namefont*: Used for the real world names in `bidding` diagrams. The default is `\mdseries\slshape`.
- **otherfont**: Used for the other bridge expressions, also outside diagrams. The default is `\bfseries\sffamily`.

If a new font is defined, all relevant dimensions of the card diagrams (including the `NESW` compass, the `bidding` diagram etc.) will be recalculated. Some examples for setting the gamefont to a new value are:

- `\gamefont{\sffamily\bfseries\HUGE}` to get *HUGE* diagrams. Refer to the documentation of package `moresize` for details.
- `\legendfont{\smaller}` to diminish the text in the card diagram conditions a little. Refer to the documentation of package `relsize` for details.
- `\gamefont{\sffamily\scalefont{3}}`[8] to typeset real big diagrams for overhead sheets

4.4 Misc

Many of the text producing macros have in common that they can produce 4 different versions of the text they represent. Normally, without any token, they produce the lowercase text. With the token `*` they produce the capitalised text. With the token `!` they produce some abbreviation of the text (if available). Finally with both tokens `*!` the produce the capitalised abbreviation of the text. What exactly is produced, is shown in the macro descriptions. In some cases it seems rather strange to have the code for an abbreviated form, i.e. `\Lead[*!]`, because it produces only the variants **lead** and **Lead**. But remember that we also support automatic translations into other languages and that in another language an abreviationt might be feasible: With the german language active `\Lead*` and `\Lead*!` produce **Ausspiel** and **Aussp.** respectively.

At the other hand it seems peculiar to let `\Ace!` produce a for an Ace. But we do not foresee which modern novelist might want to produce this. That's why they are defined, but 'normal' writers probably will never used it.

The short versions are primarily meant to be used within diagrams, allthough it is possible to get the long forms there too. Refer to page 26 for details. We show the output of such a macro `\Macro` (note the capital `M`!) in the form:

```
\Macro[*!]                    \Macro (\Macro*, \Macro!, \Macro*!)
```

[8] needs package scalefnt

In addition to each macro `\Macro` with its 4 variants, we also create a macro `\macro` which is defined to output the most used variant of `\Macro`:

`\macro`	most used variant of `\Macro`

It is very easy to redefine `\macro`. As an example we take the macro `\ace`. Its definition is:

```
\def\ace{\Ace*!}
```

so calling `\ace` will produce A. If somewhere in your document you redefine `\ace` to be

```
\def\ace{\Ace*}
```

then `\ace` will produce Ace rather than A.

4.4.1 Honor Cards

`\Ace`	ML `\Ace[*!]`	ace (Ace, a, A)
`\ace`	ML `\ace`	A
`\King`	ML `\King[*!]`	king (King, k, K)
`\king`	ML `\king`	K
`\Queen`	ML `\Queen[*!]`	queen (Queen, q, Q)
`\queen`	ML `\queen`	Q
`\Jack`	ML `\Jack[*!]`	jack (Jack, j, J)
`\jack`	ML `\jack`	J

These commands produce the language dependent names for the honor cards. To be used primarily when adding a lead to a card diagram.

4.4.2 Variants of Basic Symbols

In section 4.2 we already described the macros for the main variant. Here we introduce the macros that handle 4 variants with a combination of the tokens `*` and `!`.
We start with the variants for `\NT`

`\nt` ML `\nt[*!]` **no trump (No Trump, nt, NT)**

Next we show the 4 macros for the non-bid calls:

`\Pass` ML `\Pass[*!]` **pass (Pass, pass, Pass)**

`\Allpass` ML `\Allpass[*!]` **all pass (All pass, ap, AP)**

`\Double` ML `\Double[*!]` **double (Double, X, Dbl)**

`\Redouble` ML `\Redouble[*!]` **redouble (ReDouble, XX, ReDbl)**

Next come the 6 macros for the symbolic names of players and axes.

`\North` ML `\North[*!]` **north (North, n, N)**

`\East` ML `\East[*!]` **east (East, e, E)**

`\South` ML `\South[*!]` **south (South, s, S)**

`\West` ML `\West[*!]` **west (West, w, W)**

`\NorthSouth` ML `\NorthSouth[*!]` **north–south (North–South, n–s, N–S)**

`\EastWest` ML `\EastWest[*!]` **east–west (East–West, e–w, E–W)**

The 4 commands that typeset the *point-unit*

`\hpts` ML `\hpts[*!]` **high card points (High Card Points, hcp, HCP)**

\tpts ML \tpts[*!] **total points (Total Points, tp, TP)**

\lpts ML \lpts[*!] **length points (Length Points, lp, LP)**

\dpts ML \dpts[*!] **distribution points (Distribution Points, dp, DP)**

The forcing terms

\gforce ML \gforce[*!] **game forcing (Game Forcing, gf, GF)**

\sforce ML \sforce[*!] **semi forcing (Semi Forcing, sf, SF)**

\nmforce ML \nmforce[*!] **new minor forcing (New Minor Forcing, nmf, NMF)**

\tsforce ML \tsforce[*!] **third suit forcing (Third Suit Forcing, 3rd sf, 3rd SF)**

\fsforce ML \fsforce[*!] **fourth suit forcing (Fourth Suit Forcing, 4th sf, 4th SF)**

4.4.3 Conditions in Diagrams

These commands produce the langugage dependent expressions for ***All***, ***None***, ***by***, ***Board*** etc. To be used primarily in card diagrams.

\All ML \All[*!] **all (All, all, All)**

\all ML \all **All**

\None ML \None[*!] **none (None, none, None)**

\none ML \none **None**

\by ML \by **by**

\Board ML \Board[*!] **board (Board, brd, Brd)**

\board	ML \board	**Board**
\Contract	ML \Contract[*!]	**contract (Contract, contr, Contr)**
\contract	ML \contract	**Contract**
\Declarer	ML \Declarer[*!]	**declarer (Declarer, decl, Decl)**
\declarer	ML \declarer	**Declarer**
\Deal	ML \Deal[*!]	**deal (Deal, deal, Deal)**
\deal	ML \deal	**Deal**
\Lead	ML \Lead[*!]	**lead (Lead, lead, Lead)**
\lead	ML \lead	**Lead**

4.4.4 Annotations in Bidding Diagrams

\alert	\alert	*
\announce	\announce	A

These macros have no argument. With `\alert` one can mark a call that must be alerted with an asterisk (*) e.g. a weak 2**NT** opening with `2\NT\alert`. It produces $\boxed{2\mathbf{NT}^{*}}$. With `\announce` one can mark a bid with an 'A' ($^{\mathbf{A}}$) where an announcement is obligatory, e.g `2\He\announce` produces $\boxed{2\heartsuit^{\mathbf{A}}}$.

\markit `\markit`

\explainit `\explainit{⟨explanation⟩}`

These commands are to be used to mark a call in the `bidding` diagram and explain it with a kind of footnote-like mechanism, directly below the `bidding` diagram. Both `\markit` and `\explainit` step a counter for associating the explanation with the mark. `\markit` has no arguments; `\explainit` has one mandatory argument: the text to be displayed as explanation. `\explainit` should be called in the description part of `bidding` diagrams (or expertquiz). The text of the explanation is then typeset under the `bidding` diagram and has the same width. You can use newline (`\\`, or `\newline`) in your text to force an new line in the explanation[9].

4.4.5 Specials

\expertquiz `\expertquiz[*!][⟨comment⟩]{⟨award⟩}`

This command displays a hand, a `bidding` sequence and some additional stuff. It is designed after the *Expert Quiz* column in the *Bridge Magazin* of the DBV[10], the German Bridge Union.
It has 2 optional tokens: a '`*`' for centering and a '`!`' which 1) forces the `bidding` diagram to appear on a new line, and 2) shifts the hand a bit to the right. Next there is one optional argument with wich some commentary information can be added. And finally there is 1 mandatory argument that describes the awards for certain solutions. Both the hand and the bidding must be defined before calling `\expertquiz`. One can do that by calling e.g. `\hand-` which suppresses the output of the hand.

4.4.6 Re-Initialization

\newgame `\newgame`

This command resets most bridge diagram data and can be used to start a new series of bridge diagrams. It is however not necessary to use `\newgame` if one enters new cards for the **North** etc. hands. The list below shows which items are reset by calling `\newgame`

[9] Using \par produces the error: **Runaway argument?...**
[10] Deutsche Bridge Verband

- resets `\boardnr{0}`.
- resets `headlinetext` and `footlinetext`.
- resets `LeftUpperText`, `LeftLowerText`, `RightUpperText` and `RightLowerText`.
- resets `northhand`, `easthand`, `southhand` and `westhand`.
- resets `namesNS` and `namesEW`.
- resets the checks for Spades, Hearts, Diamonds and Clubs.

`\setdefaults` `\setdefaults[*]{⟨key1=val1⟩,⟨key2=val2⟩,...}`

key	font
bidder	bidderfont
compass	compassfont
game	gamefont
legend	legendfont
name	namefont
other	otherfont

key	value
colors	0|1|<u>2</u>|4A|4B
warn	<u>off</u>|on
err	off|<u>on</u>

key	value
compline	⟨*factor*⟩
compshow	off|<u>on</u>
compsize	⟨*factor*⟩
compturn	<u>off</u>|on

key	value
bidders	off|<u>on</u>
bidfirst	N|E|S|<u>W</u>
bidline	<u>off</u>|on
bidlong	off|<u>on</u>

The macro `\setdefaults` uses a ⟨*key*⟩=⟨*val*⟩ mechanism. The tables above show which keys are available. The underlined key values are defaults. The available keys with respect to fonts are: `bidder`, `compass`, `game`, `legend`, `name` and `other`. In upper the table to the left the association between key and font is shown. A call e.g. `\setdefaults{name=\bfseries\scriptsize}` will set the default for the namefont to the value specified. The starred form `\setdefaults*` will also call `\resetfonts`, which effectuates any change in a new default font immediately.

In the lower table to the left you'll find the three keys that are also possible as package options. They have been described already in section 3.1.3.

The keys with respect to the compass are: `compline`, `compshow`, `compsize` and `compturn`. They are shown in the upper table to the right. With the first key one can set the linethickness of the compass

frame, the default is `0.1em`. The second key controls whether the compass is shown or not. The third key controls the size of the compass, which per default is `2.5em`. With the fourth key one can rotate the letters for the **E–W** direction over 90°. The multiplication-⟨*factor*⟩, which defaults to the value `1`, can have any non-negative real value.

The keys with respect to the `bidding` diagram are: `bidders`, `bidfirst`, `bidline` and `bidlong`. They are shown in the lower table to the right. With the first key one can suppress the bidders in the bidding header. With the second key one can set which bidder appears in the first column of the diagram. The default is `W`. The third key controls whether an `\hline` is printed below the header. The fourth key switches between the long or short form of the non-bid calls, like **Pass** or **pass**.

For the key-value `on` we have the synonyms `true` and `1`, for the key-value `off` we have the synonyms `false` and `0`.

`\resetfonts`

```
\resetfonts
```

When calling this macro, all fonts are set back to their default value. This is the value that was explicitly set by a previous call of `\setdefaults`, or to the intrinsic default value if `\setdefaults` has not been called before.

4.5 Environments

In the first place we must warn the user for a peculiarity of the package `collcell`, which is used to do some special processing in these 3 environments: *The last row must either end with a newline* (`\\`) *or with an* empty *cell.*

The advantage of the `collcell` processing is that within the `bidding` and `play` diagrams we can use shorthands for the suit symbols: rather than writing `3\Sp` in a `bidding` diagram we can also write `3S` to obtain 3♠. In the `play` diagram we could write `HA` instead of `\He\,\Ace` and get ♥A as output. In `bidding` diagrams some non-bid calls may appear in short or in long form, controlled by calling `\setdfefaults{bidlong=on}`, which switches to the lang form, or `\setdfefaults{bidlong=off}` which switches to the short form. These non-bid calls are coded as follows: A small `p` yields short form **pass** or long form **Pass**. A capital `P` yields short form **AP** or long form **All pass**. A capital `X` yields short form **X** and long form **Double**. A capital `R` yields short form **XX** and long

form **ReDouble**.[11]

bidding ML `\begin{bidding}[*!-][⟨pos⟩](⟨description⟩)...\end{bidding}`

This environment has 3 tokens. The '`*`' centers the `bidding` diagram. The '`!`' outputs the short form: e.g. **N** rather than the long form **North** in the table heading. The '-' completely suppresses the output. The data is stored in a savebox and can be used in other macros, e.g. in all `\showXX` macros. Next come 2 optional parameters. The first one controls the alignment (default **c**) and the second one adds the list of annotations below the `bidding` diagram. With `\setdefaults` one can fine tune the look of the `bidding` diagram. Refer to page 26 for details. For example

```
\begin{bidding}(\explainit{WeakTwo}\explainit{preemptive})
  2H\markit & p & 3H\markit & p \\
\end{bidding}
\quad
\setdefaults{bidlong=off}
\begin{bidding}!
  2H & p & 3H & p\\
\end{bidding}\\
```

produces

South	West	North	East
2♥[a]	pass	3♥[b]	pass

[a] WeakTwo
[b] preemptive

S	W	N	E
2♥	pass	3♥	pass

In addition to the shorthands we mentioned before, one can also use macros in the bidding and play diagrams. As long as these macros appear as 1 single token in the diagram, no special care has to be taken. But if a macro appears as multiple tokens, e.g. like the call `\Pass*!`, which consists of the 3 tokens `\Pass`, `*` and `!`, then it **must** be enclosed in braces `{...}` to make it act as 1 token. Without the braces `\Pass*!` will produce **pass***!, with them `{\Pass*!}` will produce **Pass**. Note that although `\pass` expands to `\Pass*`, it wil produce the correct **Pass**.

biddingpair ML `\begin{biddingpair}[*!-][⟨pos⟩](⟨descr⟩)...\end{biddingpair}`

This environment is essentially the same as the `bidding` environment, but here the `bidding` diagram has only two columns rather than four.

[11] due to the fact that `x` denotes a spotcard, we had to implement it this way

play ML `\begin{play}[*]{⟨Lead⟩}[⟨Trump⟩]...\end{play}`

This environment has 1 token, a '*', which controls the centering of the table; 1 mandatory argument, denoting the opening lead and 1 optional argument which specifies the trump suit, with default: **NT**. It typesets a diagram with the sequence of playing tricks, by producing a table with 8 columns: the running number; the player who had had the lead; the 4 cards played in this trick and finally the 2 columns that show how many tricks **N–S** and **E–W** have won so far. The user has to specify only the columns 3-6, i.e. the cards that were played. Columns 1–2 and 7–8 are constructed by OneDown. Just as in `bidding` diagrams, also here one can denote the suit with `S`, `H`, `D`, or `C`. The winning card is automatically detected, taking into account if a suit contract or **NT** is played. The winning tricks counters are then updated automatically. Concerning consistency it is just as with hands and deals: If a card occurs more than one time, an error is raised and for missing cards a warning is issued.

```
\begin{play}{W}  \\
  D3 & 2 & Q & K \\
  HA & 5 & 4 & J \\
  C3 & 5 & K & A \\
\end{play}
```

№	Lead	2nd	3rd	4th	N S	E W
1	W: ♦3	2	Q	K	1	0
2	S: ♥A	5	4	J	2	0
3	S: ♣3	5	K	A	2	1

```
\begin{play}*{W} \\
  D3 & 2 & Q & K \\
  HA & 5 & 4 & J \\
  C3 & 5 & K & A \\
\end{play}
```

produces the same diagram, but this time horizontally centered.

№	Lead	2nd	3rd	4th	N S	E W
1	W: ♦3	2	Q	K	1	0
2	S: ♥A	5	4	J	2	0
3	S: ♣3	5	K	A	2	1

5 Final Remarks

5.1 Known Bugs

- Some dictionaries have questionable translations.

- The boxes in `\northhand` and `\southhand` lead to problems with shifting the **West** hand. See file Legends in the example bundle.

5.2 ToDo

- Check if `\def\xspace{}` is also needed in bidding...?
- Make a template for showing 16 hands on 1 DIN A4 (3x5+1 or 4x4 landscape) e.g as handout for the hands on slides.
- Read source files in PBN format. Example: http://new.bridgekosice.sk/bridzove-diagramy-vykrelene-pomocou-tex/

5.3 Acknowledgements

This package is based on (ideas from) the style files:

- `bridge-i v1.0 (1995/04/16)` by René Steiner and Thomas Hof.
- `kibitzer v1.0 (1995/04/16)` by René Steiner and Thomas Hof.
- `bridge v0.1 (2012/03/18)` by Antony Lee.
- `grbbridge v2.2 (2013/12/24)` by Gordon Bower.

The style file `bridge-i` is based on the style file `bridge` v1.7c (1994/12/20) by J.L. Braams, which on itself was based on an article by Kees van der Laan in TUGboat (Vol 11 (1990), No 2: p265ff.

Last but not least I want to thank all those wonderful people down there in the Internet who spent their time in answering silly questions and solving difficult problems. If I had imagined the difficulties I would encounter, then I would not have started this work. And without the help of all these, to me unknown, people, this package would not exist.

6 Implementation

6.1 Preliminaries

6.1.1 Packages we depend upon

```
\RequirePackage{%
  xcolor,%        colorizing symbols \Sp etc.
  textcomp,%      for the numbersign in environment play.
  moresize,%      add \HUGE and \ssmall to font-sizes
  relsize,%       relative font-sizes. (e.g. \smaller)
  makecmds,%      needed for provideenvironment
  expl3,%         needed for LaTeX3 packages (xparse)
  xparse,%        optional params and starred commands
  xspace,%        handling of spacing behind a command
  calc,%          makes calculations and lengths easier
```

```
  ifthen,%        easy booleans, tests and loops
  adjustbox,%     stacked boxes in L-/R-Lower captions
  translations,%  auto-translate terms (e.g. East->Ost)
  array,%         actions for tabular column cells
  collcell,%      macro calls for tabular column cells
  pgfopts,%       for keyval opts, loads also pgfkeys
  environ,%       for handling bidding environments
  xstring,%       for easy string processing.
  tracklang,%     for iterating over loaded languages
  pict2e,%        for drawing the compass
}
```

Add exceptions for xspace

```
\xspaceaddexceptions{%
  = \markit \, \suit \GetTranslation
  2 3 4 5 6 7 8 9 T J Q K A
}
```

Load the fallback dictionary. This is the default one and equals the English dictionary. All other dictionaries are loaded on the fly, when needed, provided they are in the TEX-path.

```
\LoadDictionaryFor{fallback}{ODw}
```

6.1.2 Options

We use the `pgf` ⟨*key*⟩=⟨*val*⟩ system for our options: `colors`, `warn` and `err`.

```
\pgfkeys{/ODw/.is family}
\def\ODw@set#1{\pgfkeys{/ODw,#1}}
\ODw@set{colors/.is choice,}
\ODw@set{warn/.is choice,}
\ODw@set{err/.is choice,}
```

The details for option `colors` are on page 39ff and those for option `warn` and `err` on page 100.

6.1.3 Misc

`\,` `\,`

`\thinspace` `\thinspace`

We redefine `\thinspace` (originally $\frac{1}{6}$em) to a smaller amount. That makes denominations like `3\Sp` (3♠) look better. The code is from: https://tex.stackexchange.com/questions/181003/multiply-fine-

tuning-with-a-thinspace
This code however doesn't work when coded within an own package, unless we use `\AtBeginDocument`.

```
\AtBeginDocument{%
  \renewcommand{\,}[1][1]{%
    \ifmmode\mskip#1\thinmuskip%
     \else\thinspace[#1]\fi%
  }%
  \renewcommand{\thinspace}[1][1]{%
   \kern#1\dimexpr0.16667em\relax%
  }%
}% AtBeginDocument
```

`\ODw@gsetlength` `\ODw@gsetlength{⟨\len-name⟩}{⟨len-value⟩}`

We need to store the length of certain objects that are within a group (the group is needed to keep the font-size changes local). Therefore we define macro `\ODw@gsetlength` that works globally. The code is based upon a solution on LaTeX StackExchange: (https://tex.stackexchange.com/questions/406015/defining-macro-gsetlength-as-global-setlength-reliable)

```
\gdef\ODw@gsetlength#1#2{%
  \begingroup
    \setlength\skip@{#2}% local assign to scratch reg.
    \global#1=\skip@%     global assignment to #1;
  \endgroup%              restore \skip@ by endgroup.
}% ODw@gsetlength
```

`\ODw@append` `\ODw@append{⟨tokens⟩}`

In the environment `play` we need to calculate the winning tricks for **N–S** and **E–W**[12]. We store this information as a string in `\ODw@Scratch` and use `\ODw@append` to accumulate them.

```
\gdef\ODw@append#1{%
  \bgroup%
    \edef\tmp{\the\ODw@Scratch #1}%
    \global\ODw@Scratch=\expandafter{\tmp}%
  \egroup%
}% ODw@append
```

[12] For details see page 90

6.1.4 Variables

In this package we do use font relative sizing. That means that widths and skips are defined in terms of `em`, `ex` and `baselineskip`. On the other hand there are e.g. the real world names of the bidders, that must be recorded. Some of these the user should be able to control. Rather than forcing the user to do that directly with `\def` or `\renewcommand` we store all this information in internal variables, by defining a constant command. These variables can be set by calling a user command, that is associated with the variable. E.g. the variable `\ODw@Skipwidth` gets set by the command `\handskip`. The variables we use are:

`\ODw@BidderFont` Controlled by `\bidderfont`: the font used for the players (`bidding`).

`\ODw@CompassFont` Controlled by `\compassfont`: the font used for the compass.

`\ODw@GameFont` Controlled by `\gamefont`: the font used for all card diagrams

`\ODw@LegendFont` Controlled by `\legendfont`: the font used for the annotations.

`\ODw@NameFont` Controlled by `\namefont`: the font used for the real world names.

`\ODw@OtherFont` Controlled by `\otherfont`: the font used for other bridge expressions, also outside diagrams.

`\ODw@BidderDefault` `\ODw@CompassDefault` `\ODw@GameDefault` `\ODw@LegendDefault` `\ODw@NameDefault` `\ODw@OtherDefault` These contain the default values for the fonts

`\ODw@North@Name` `\ODw@South@Name` Controlled by `\namesNS`: hold the real world names of the **North** and the **South** player.

`\ODw@East@Name` `\ODw@West@Name` Controlled by `\namesEW`: hold the real world names of the **East** and the **West** player. These names are typeset using the font specified with `\namefont`.

`ODw@BoardText` Controlled by `\boardtext`: holds the board number and extra text.

`ODw@HeaderText` Controlled by `\headlinetext`: holds header information for card diagrams.

`ODw@FooterText` Controlled by `\footlinetext`: holds footer information for card diagrams.

`ODw@Last` Used in command `\ODw@Tricks`: to store the player that had the lead[13].

`\ODw@Skipwidth` Controlled by `\handskip`: holds the distance between hand and bidding diagram.

[13] see page 89

```
\def\ODw@CompSize{1}% factor to enlarge the compass
\def\ODw@CompLine{}%  thickness of compass frame
\def\ODw@Skipwidth{1em}%
```

6.1.5 Booleans, Saveboxes, Lengths, Counters and Registers

6.1.5.1 Booleans

\ifODw@description `\ifODw@description` is used in the `bidding` environments and the command `\expertquiz` to test if there is an annotation that should be written.

\ifODw@short `\ifODw@short` is used in the `bidding` environments and the command `\expertquiz` to denote if a short form of the diagram header is to be used.

\ifODw@monochrome `\ifODw@monochrome` flags the case that the user specified `colors=0` or `colors=1`, i.e. just black and white. In this situation we will not print the vulnerable side in red, but use italics instead.

\ifOdW@CardSkip `\ifOdW@CardSkip` determines whether we need some extra space between card ranks (i.e. in suit descriptions) or not (i.e. in `bidding` or `play` diagrams).

\ifOdW@Bidders `\ifODw@Bidders` suppresses the bidders in the bidding header.

\ifOdW@BidLine `\ifODw@BidLine` draw a `\hline` below the bidding header.

\ifOdW@LongCalls `\ifODw@LongCalls` determines whether to use the short (like **N**) or the long form (like **North**) for calls in the `bidding` diagram.

\ifOdW@CompShow With `\ifODw@CompShow` one can suppress drawing a compass within card diagramns completely.

\ifOdW@CompTurn With `\ifODw@CompTurn` one rotates letters `E` and `W` in the compass 90°.

```
\newif\ifODw@description% must typeset an annotation
\newif\ifODw@short%       short form in bidding header
\newif\ifODw@monochrome%  no colors wanted
\newif\ifOdW@CardSkip%    skip between ranks needed
```

The next booleans are directly controlled by `\setdefaults`.

```
\newif\ifODw@Bidders%    suppress bidders in bidding header
\newif\ifODw@BidLine%    draw \hline below bidding header
\newif\ifODw@LongCalls%  switch between long/short calls
\newif\ifODw@CompShow%   show compass or not
\newif\ifODw@CompTurn%   turn E-W letters 90°
```

\ODw@EmptyHeader Since there seems to be a problem in using `\ifthenelse`[14] in particular

\ODw@EmptyFooter places, these booleans are set by calling `ODw@TestIfEmpty` (which uses

[14] **ifthenelse** bites **multicolumn**!

an ordinary \ifthenelse) *outside* the dangerous places, and then use e.g. \ifODw@EmptyHeader as a test whether the header is empty or not.

```
\newboolean{ODw@EmptyHeader}% = 'header is empty'
\newboolean{ODw@EmptyFooter}% = 'footer is empty'
```

6.1.5.2 Saveboxes

\ODw@Diagram@Box \ODw@Diagram@Box is to store the actual card diagram (the compass with the hands) in order to calculate its width.

\ODw@BidBox \ODw@BidBox stores a bidding diagram.

\ODw@Hand@Box \ODw@Hand@Box stores a hand with the 4 suits.

```
\newsavebox\ODw@Diagram@Box
\newsavebox\ODw@Hand@Box
\newsavebox\ODw@BidBox
```

6.1.5.3 Lengths

\ODw@Bid@Width Is used to store the actual width of the bidding diagram.

\ODw@Card@Skip Defines space between adjacent cards in suits.

\ODw@Diagram@Width Defines the width of compass plus (**E**–**W**) hands.

\ODw@Skip@Width Defines the distance between the card diagram and the bidding diagram.

\ODw@Tmp@Len

\ODw@Tmp@Width Auxiliary lengths, used in several calculations.

```
\newlength\ODw@Compasssize% the size of the compass.
\newlength\ODw@Diagram@Width
\newlength\ODw@Card@Skip
\setlength\ODw@Card@Skip{.15em}% space between cards
\newlength\ODw@Bid@Width
\newlength\ODw@Skip@Width
  \setlength\ODw@Skip@Width{\ODw@Skipwidth}
\newlength\ODw@Tmp@Len%   temp var for computations
\newlength\ODw@Tmp@Width% temp var for computations
```

6.1.5.4 Counters

ODw@Nr Counts lines (in play diagrams) and explanations (in bidding diagrams).

```
\newcounter{ODw@Nr}
```

ODw@Cnt Auxiliary counter, used in several calculations.

```
\newcounter{ODw@Cnt}
```

ODw@PlayerNr Set to the player that won the trick in environment play.

```
\newcounter{ODw@PlayerNr}
```

ODw@NSCnt Holds the number of N-S tricks in environment play.

```
\newcounter{ODw@NSCnt}
```

ODw@EWCnt Holds the number of E-W tricks in environment play.

```
\newcounter{ODw@EWCnt}
```

6.1.5.5 Registers

ODw@Scratch Tempory store for winning tricks in environment play.

```
\newtoks{\ODw@Scratch}
```

6.1.6 Fonts

6.1.6.1 Text Fonts

Here we merely define the commands to set the default fonts. At the end of this .sty file they are set to their value. Refer to section 6.10 for details.

\bidderfont
\ODw@BidderFont
The font used to indicate the symbolic player (N, E, S, W) in bidding diagrams. The default is \mdseries\sffamily.

```
\newcommand\bidderfont[1]{\gdef\ODw@BidderFont{#1}}
```

\compassfont
\ODw@CompassFont
The font used to indicate the directions (N, E, S, W) in the compass. The default is \mdseries\sffamily.

```
\newcommand\compassfont[1]{\gdef\ODw@CompassFont{#1}}
```

\namefont
\ODw@NameFont
The font used for the real world names of the players in bidding diagrams. The default id \mdseries\slshape.

```
\newcommand\namefont[1]{\gdef\ODw@NameFont{#1}}
```

\legendfont
\ODw@LegendFont
The font used for the conditions in card diagrams. The default is \mdseries\rmfamily.

```
\newcommand\legendfont[1]{\gdef\ODw@LegendFont{#1}}
```

\otherfont
\ODw@OtherFont
The font used for the other bridge expressions like \north, \pass or \double. The default is \bfseries\sffamily.

```
\newcommand\otherfont[1]{\gdef\ODw@OtherFont{#1}}
```

\gamefont The font for the hands and calls. It sets the general font-size/widths for the game. The default is \bfseries\sffamily.

\ODw@GameFont We use widths that are dynamically adjusted at font changes and store the 'value' as text in `\ODw@GameFont`.

\ODw@GameSize `\ODw@GameSize` recalculates these sizes and is called in all show- and bid-diagrams.

```
\newcommand\gamefont[1]{%
  \gdef\ODw@GameFont{#1}%
  \gdef\ODw@GameSize{% recalculate dimens for the new font
    \ODw@GameFont%
    \setlength\ODw@Skip@Width{\ODw@Skipwidth}%
  }%
}% gamefont
```

6.1.6.2 Symbol Font

We need special symbols to get solid colored ♥ and ♦, rather than ♡ and ♢. We use those from `stix`. As the shape of the 'normal' black suits differ from the red ones we also take the black suits from the font `stix`. First we define the symbols and font. As we do not want to load the complete package, we only use the relevant piece of code found in `txfont.sty`:

```
\fontencoding{T1}\fontfamily{stix}
\fontseries{m}\fontshape{n}\selectfont
%
% Code stolen from txfonts.sty.
% It works smoothly: thank you guys!
%
\DeclareSymbolFont{symbols}{OMS}{txsy}{m}{n}
\SetSymbolFont{symbols}{bold}{OMS}{txsy}{bx}{n}
\DeclareFontSubstitution{OMS}{txsy}{m}{n}
\DeclareSymbolFont{symbolsC}{U}{txsyc}{m}{n}
\SetSymbolFont{symbolsC}{bold}{U}{txsyc}{bx}{n}
\DeclareFontSubstitution{U}{txsyc}{m}{n}
```

\ODw@spadesuit

```
\DeclareMathSymbol{\ODw@spadesuit}{\mathord}{symbols}{127}
```

\ODw@varheart

```
\DeclareMathSymbol{\ODw@varheart}{\mathord}{symbolsC}{114}
```

\ODw@vardiamond

```
\DeclareMathSymbol{\ODw@vardiamond}{\mathord}{symbolsC}{113}
```

\ODw@clubsuit

```
\DeclareMathSymbol{\ODw@clubsuit}{\mathord}{symbols}{124}
```

6.2 Bridge Basic Terms

6.2.1 Suit Symbols

First we supply shorthands for the 'five' suits (♣, ♦, ♥, ♠ and **NT**) that are used in the game of bridge. We define the international version with the English shortcuts. We use the `xcolor` package to colorize the suit symbols. The color can be set as an ⟨*key*⟩=⟨*val*⟩ option when loading the package. The option `colors=0` means mono-color (black only), synonyms of key 0 are `mono` and `black`. `colors=1` means black and white, a synonym is `b+w`. `colors=2` means bi-color (black and red), with synonym `b+r`. `colors=4A` gives qua-color (green, orange, red and blue); synonyms are `fourA` and `4a`. Finally `colors=4B` defines the second qua-color (black, orange, red and green) with synonyms `fourB` and `4b`.

We precede all the suit symbols with a 'very-thin-space' (`\,[0.3]`) which is 0.3 the size of a normal `\thinspace`.

In order to test which suit (`\Cl`,...) was encountered in `\ODw@translate`[15] we *must* define the suits with a `renewrobustcommand`. So we must `\def` them first.

```
% \def them first and then use renewrobustcmd!
\def\Cl{}\def\Di{}\def\He{}\def\Sp{}%
\ODw@set{%
 colors/0/.code={%
  \ODw@monochrometrue%
  \renewrobustcmd\Cl{\textcolor{black}%
   {\,[0.3]\ensuremath{\ODw@clubsuit}}\xspace}%
  \renewrobustcmd\Di{\textcolor{black}%
   {\,[0.3]\ensuremath{\ODw@vardiamond}}\xspace}%
  \renewrobustcmd\He{\textcolor{black}%
   {\,[0.3]\ensuremath{\ODw@varheart}}\xspace}%
  \renewrobustcmd\Sp{\textcolor{black}%
   {\,[0.3]\ensuremath{\ODw@spadesuit}}\xspace}%
 }%
}
\ODw@set{colors/mono/.code={\pgfkeys{/ODw/colors=0}}}
\ODw@set{colors/black/.code={\pgfkeys{/ODw/colors=0}}}
%
```

[15] see page 45

```
\ODw@set{%
 colors/1/.code={%
  \ODw@monochrometrue%
  \renewrobustcmd\Cl{\textcolor{black}%
   {\,[0.3]\ensuremath{\ODw@clubsuit}}\xspace}%
  \renewrobustcmd\Di{\,[0.3]\ensuremath{\diamondsuit}\xspace}%
  \renewrobustcmd\He{\,[0.3]\ensuremath{\heartsuit}\xspace}%
  \renewrobustcmd\Sp{\textcolor{black}%
   {\,[0.3]\ensuremath{\ODw@spadesuit}}\xspace}%
 }%
}
\ODw@set{colors/b+w/.code={\pgfkeys{/ODw/colors=1}}}
%
\ODw@set{%
 colors/2/.code={%
  \renewrobustcmd\Cl{\textcolor{black}%
   {\,[0.3]\ensuremath{\ODw@clubsuit}}\xspace}%
  \renewrobustcmd\Di{\textcolor{red}%
   {\,[0.3]\ensuremath{\ODw@vardiamond}}\xspace}%
  \renewrobustcmd\He{\textcolor{red}%
   {\,[0.3]\ensuremath{\ODw@varheart}}\xspace}%
  \renewrobustcmd\Sp{\textcolor{black}%
   {\,[0.3]\ensuremath{\ODw@spadesuit}}\xspace}%
 }%
}
\ODw@set{colors/b+r/.code={\pgfkeys{/ODw/colors=2}}}
%
\ODw@set{%
 colors/4A/.code={%
  \renewrobustcmd\Cl{\textcolor{green}%
   {\,[0.3]\ensuremath{\ODw@clubsuit}}\xspace}%
  \renewrobustcmd\Di{\textcolor{orange}%
   {\,[0.3]\ensuremath{\ODw@vardiamond}}\xspace}%
  \renewrobustcmd\He{\textcolor{red}%
   {\,[0.3]\ensuremath{\ODw@varheart}}\xspace}%
  \renewrobustcmd\Sp{\textcolor{blue}%
   {\,[0.3]\ensuremath{\ODw@spadesuit}}\xspace}%
 }%
}
\ODw@set{colors/fourA/.code={\pgfkeys{/ODw/colors=4A}}}
\ODw@set{colors/4a/.code={\pgfkeys{/ODw/colors=4A}}}
%
\ODw@set{%
```

```
colors/4B/.code={%
 \renewrobustcmd\Cl{\textcolor{black}%
  {\,[0.3]\ensuremath{\ODw@clubsuit}}\xspace}%
 \renewrobustcmd\Di{\textcolor{orange}%
  {\,[0.3]\ensuremath{\ODw@vardiamond}}\xspace}%
 \renewrobustcmd\He{\textcolor{red}%
  {\,[0.3]\ensuremath{\ODw@varheart}}\xspace}%
 \renewrobustcmd\Sp{\textcolor{green}%
  {\,[0.3]\ensuremath{\ODw@spadesuit}}\xspace}%
}%
}
\ODw@set{colors/fourB/.code={\pgfkeys{/ODw/colors=4B}}}
\ODw@set{colors/4b/.code={\pgfkeys{/ODw/colors=4B}}}
```

Set default coloring to black and red

```
\ODw@set{colors=2}
```

`\nt` `\NT` Because some languages use a different symbol for **NT** (**No Trump**) we must look it up in the dictionary to find e.g. **SA**(**Sans Atout**) for German.

```
\NewDocumentCommand\nt{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF#1{%
      \IfBooleanTF#2
        {\,[0.3]\GetTranslation{NT-(ODw)}}%
        {\GetTranslation{No Trump-(ODw)}}%
      }{%
      \IfBooleanTF#2
        {\,[0.3]\GetTranslation{nt-(ODw)}}%
        {\GetTranslation{no trump-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% nt
```

Define a practical shorthand to produce **NT** without the need to add a token.

```
\def\NT{\nt*!}
```

`\ODw@SetRank` `\ODw@SetRank{⟨card rank⟩}`

`\ODw@SetRank` stores the rank of the card played in `\ODw@Rank`. This is essentially the intrinsic rank of the card (*2* for a 2, *14* for an Ace),

but there are special cases:

- Spot cards (denoted with x) always get rank 0
- Discards always get rank 0
- Trump cards get 15 (*15* to ensure that a spot trump card defeats an Ace) added to the intrinsic rank, to make sure that:
  - A trump card will defeat all other cards
  - The highest trump card will win the trick

We first define three variables, one to store the suit of the actual card, the second one to store which suit was led and the last variable to store which suit is the trump suit, all initialized with the '**NT**-suit'.

```
\gdef\ODw@SuitPlayed{N}
\gdef\ODw@SuitLead{N}
\gdef\ODw@TrumpSuit{N}
\newcounter{ODw@Rank}

\def\ODw@SetRank#1{%
  \ifthenelse{\equal{\ODw@SuitPlayed}{\ODw@SuitLead}}%
  % if a suit is followed, store the intrinsic rank
    {\setcounter{ODw@Rank}{#1}}%
    {% else, if a suit is not followed then ...
  % (at NoTrump, the trumpsuit is coded 'N' and will
  % never match a real suit (coded C, D, H and S))
  % thus avoiding that trump cards are detected
       \ifthenelse{\equal{\ODw@SuitPlayed}{\ODw@TrumpSuit}}%
  % if it is a trump card, increase the rank
         {\setcounter{ODw@Rank}{#1}%
          \addtocounter{ODw@Rank}{15}}%
  % if it is a discard, set the rank to 0 to make
  % sure it will never win
         {\setcounter{ODw@Rank}{0}}%
    }% ifthen
  % If the card was of another suit,
  %    then ODw@SuitPlayed was changed.
  % If we encounter 'unsuited' cards,
  %    then we must reestablish the
  %    original ODw@SuitPlayed.
    \global\edef\ODw@SuitPlayed{\ODw@SuitLead}% org. suit
}% ODw@SetRank
```

`\ODw@Xfer` `\ODw@Xfer{⟨tokens⟩}`

This macro gets called by `ODw@Tfer` which is automatically called in the environments `play`, `bidding` and `biddingpair` by means of `columntype P`

and `columntype B` to convert at one hand the shorthand suit code in suit symbols and at the other hand to translate card honors into the active language. It also converts the card value `T` into 10 and a hyphen into an en-dash. It calls `\ODw@translate` to do the work.

The following code was contributed on StackExchange by egreg, see https://tex.stackexchange.com/questions/417731/problem-with-xstring-ifeqcase-case-falls-thru/417788?noredirect=1#comment1045001_417788

```
\ExplSyntaxOn
 % NB: now all spaces are ignored, use '~' if needed.
\NewDocumentCommand{\ODw@Xfer}{m}{
  \bgroup
  % we do not want spaces here
    \def\xspace{}
    \tl_map_function:nN {#1} \ODw@translate:n
  \egroup
}% ODw@Xfer
```

`\ODw@translate` `\ODw@translate{⟨tokens⟩}`

`\ODw@translate` processes a (relatively short) string of tokens that determine an entry in `bidding` or `play` diagrams, and also in all situations where suits are defined. It gets called by `\ODw@Xfer`.

It expects bridge stuff describing strings like `AKT54` to produce the suit A K 10 5 2, `2H\alert` to produce the call 2♥* in the `bidding` diagram, or `DA` to produce ♦A as entry in the `play` diagram to show that the ace of diamonds was played. Please note that constructs like `\textit{DA}` or `\frame{2H}` are not allowed and will produce rather misleading errors like:

```
! Argument of \ODw@translate:n has an extra }.
```
or
```
! Missing number,treated as zero.
```

Even clever people who use `{\frame{2H}}` will get disoppointed, because they'll get 2H rather than the wanted 2♥. But the very clever people can reach their goal by using `{\frame{2\He}}` or `{\textit{\Di A}}`.

```
\cs_new_protected:Nn \ODw@translate:n
 {
  \setcounter{ODw@Rank}{0}
  \str_case:nnTF {#1}
   { % Store the suit of the card played
     % needed to determine the winner
```

```
    % and for checking for multiple cards
    {C}{\Cl\gdef\ODw@SuitPlayed{C}}
    {D}{\Di\gdef\ODw@SuitPlayed{D}}
    {H}{\He\gdef\ODw@SuitPlayed{H}}
    {S}{\Sp\gdef\ODw@SuitPlayed{S}}
    {N}{\NT\gdef\ODw@SuitPlayed{N}}
% Translate a hyphen into an en-dash
    {-}{--}
%
% 1: translate the honour cards,
% 2: store the played cards for checking
% 3: and set their rank. This must be done last, because
% \ODw@SetRank resets \ODw@SuitPlayed to \ODw@SuitLead
% Honour Cards
%        1                        2
    {A}{\Ace*!\ODw@AppendCard{\ODw@SuitPlayed}{A}
         \ODw@SetRank{14}}% 3
    {K}{\King*!\ODw@AppendCard{\ODw@SuitPlayed}{K}
         \ODw@SetRank{13}}
    {Q}{\Queen*!\ODw@AppendCard{\ODw@SuitPlayed}{Q}
         \ODw@SetRank{12}}
    {J}{\Jack*!\ODw@AppendCard{\ODw@SuitPlayed}{J}
         \ODw@SetRank{11}}
    {T}{\kern-0.1em1\kern-0.1em0% massage 1 and 0 a bit
        \ODw@AppendCard{\ODw@SuitPlayed}{T}\ODw@SetRank{10}}
% Numeral Cards
    {9}{9\ODw@AppendCard{\ODw@SuitPlayed}{9}\ODw@SetRank{9}}
    {8}{8\ODw@AppendCard{\ODw@SuitPlayed}{8}\ODw@SetRank{8}}
    {7}{7\ODw@AppendCard{\ODw@SuitPlayed}{7}\ODw@SetRank{7}}
    {6}{6\ODw@AppendCard{\ODw@SuitPlayed}{6}\ODw@SetRank{6}}
    {5}{5\ODw@AppendCard{\ODw@SuitPlayed}{5}\ODw@SetRank{5}}
    {4}{4\ODw@AppendCard{\ODw@SuitPlayed}{4}\ODw@SetRank{4}}
    {3}{3\ODw@AppendCard{\ODw@SuitPlayed}{3}\ODw@SetRank{3}}
    {2}{2\ODw@AppendCard{\ODw@SuitPlayed}{2}\ODw@SetRank{2}}
% A spot card has rank 0
    {x}{x\ODw@SetRank{0}}
%
% Non cards (bidding only)
    {1}{1}% this enables e.g. 1\He in biddings
    {p}{\ifODw@LongCalls\Pass*\else\Pass!\fi}
    {P}{\ifODw@LongCalls\Allpass*\else\Allpass*!\fi}
    {X}{\ifODw@LongCalls\Double*\else\Double!\fi}
    {R}{\ifODw@LongCalls\Redouble*\else\Redouble!\fi}
```

```
  }% case
  {% if matched (case T(rue))
   \ifOdW@CardSkip\hspace{\ODw@Card@Skip}\fi
 % suit of 1st card (ODw@SuitLead) is ODw@SuitPlayed
  \if\theODw@PlayerNr1
    \global\edef\ODw@SuitLead{\ODw@SuitPlayed}
  \fi
  }
```

We offer the possibility that one can use also `\He` in `bidding` and `play` diagrams rather than just the abbreviation `H`. Therefore we must test which suit was given and set `\ODw@SuitPlayed` accordingly. To make this test work, we had to redefine the suit macros with an `\renewrobustcmd`. Here we also issue `\expandafter{#1}` rather than just `#1`. Otherwise, among others, the coloring of the suit symbol would extend behind it. Curiously enough the phenomena does not occur anymore. I leave the expandafter in, until this is cleared.

```
  {% if not matched (case F(alse))
    \ifx#1\Cl\gdef\ODw@SuitPlayed{C}\fi
    \ifx#1\Di\gdef\ODw@SuitPlayed{D}\fi
    \ifx#1\He\gdef\ODw@SuitPlayed{H}\fi
    \ifx#1\Sp\gdef\ODw@SuitPlayed{S}\fi
    \expandafter{#1}% enables e.g. 1\He
  % suit of 1st card (ODw@SuitLead) is ODw@SuitPlayed
    \if\theODw@PlayerNr1
      \global\edef\ODw@SuitLead{\ODw@SuitPlayed}
    \fi
  }
}% ODw@translate
\ExplSyntaxOff
```

`\ODw@AppendCard` `\ODw@AppendCard{⟨suit⟩}{⟨card⟩}`

In order to do a simple consistency check in `play` diagrams, we need to store the cards that were played. We do that for each suit in the variable `\ODw@<suit>`. This macro is called in `\ODw@translate`, i.e. for all situations where cards are to be manipulated. But the result of `\ODw@AppendCard` is used only within `play` diagrams. The macro `\ODw@appendcard` appends 1 character to a string.

```
%
\newcommand{\ODw@appendcard}[2]{\xdef#1{#1#2}}

```

```
\newcommand\ODw@AppendCard[2]{%
  \IfEqCase{#1}{%
    {C}{\ODw@appendcard{\ODw@Clubs}{#2}}%
    {D}{\ODw@appendcard{\ODw@Diamonds}{#2}}%
    {H}{\ODw@appendcard{\ODw@Hearts}{#2}}%
    {S}{\ODw@appendcard{\ODw@Spades}{#2}}%
  }%
}% ODw@AppendCard
```

\ODw@PTfer

\ODw@PTfer{⟨*tokens*⟩}

This macro is called within play diagrams where we can write HA and get ♥A. Also all relevant symbols get translated into the active language. We use the counter ODw@PlayerNr to determine the column in the play diagrams with the winning card, and from this we can compute which player won the trick. \ODw@PTfer is essentially called for each entry in all columns of the play diagram through the column definition:

```
\newcolumntype{P}{>{\collectcell\ODw@PTfer}c<{\endcollectcell}}
```

We first define two counters, both initially set to zero.

```
\newcounter{ODw@Highest}%    the highest rank until now
\setcounter{ODw@Highest}{0}
\newcounter{ODw@WinningNr}%  player with the highest rank
\setcounter{ODw@WinningNr}{0}

\def\ODw@PTfer#1{%
  \stepcounter{ODw@PlayerNr}%
  \ODw@Xfer{#1}% ODw@Rank = the rank for this card
  \ifthenelse{\value{ODw@Rank} > \value{ODw@Highest}}%
    {% This rank is higher than previous highest one
      \setcounter{ODw@WinningNr}{\theODw@PlayerNr}%
      \setcounter{ODw@Highest}{\theODw@Rank}%
    }%
    {}%
  \ifthenelse{\value{ODw@PlayerNr} = 4}%
    {% last player: Process the winning trick:
      \stepcounter{ODw@Nr}% Start new row with new player
      \ODw@AccTricks%      Accumulate tricks for N-S/E-W
    }%
    {}%
}
```

\ODw@FTfer `\ODw@FTfer{⟨tokens⟩}`

`\ODw@FTfer` is called for the first column of the `play` diagram TableII. In `\ODw@Tricks` it just resets `ODw@PlayerNr` and `\ODw@Last` and writes the player who leads. Finally it processes the entry of the first column by calling `\ODw@PTfer`. `\ODw@FTfer` is essentially called for the entries in the first column of TableII in the `play` diagram through the column definition:

```
\newcolumntype{F}{>{\collectcell\ODw@FTfer}c<{\endcollectcell}}
```

```
\def\ODw@FTfer#1{%
  \ODw@Tricks%
  \ODw@PTfer{#1}%
}% ODw@FTfer
```

\ODw@BTfer `\ODw@BTfer{⟨tokens⟩}`

This macro is called within `bidding` diagrams and enables us to type `1C\announce` and get 1♣[A]. The symbols get translated into the active language. `\ODw@BTfer` is essentially called for each entry in the `bidding` diagrams through:

```
\newcolumntype{B}{>{\collectcell\ODw@BTfer}c<{\endcollectcell}}
```

As there is no special processing for the biding entries, we call `\ODw@Xfer` right away to do the job.

```
\def\ODw@BTfer#1{%
  \ODw@Xfer{#1}%
}
```

6.2.2 Names of Directions and Axes

\North `\North[*!]`

\north `\north`

```
\NewDocumentCommand\North{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF#1{%
      \IfBooleanTF{#2}{\ODw@N*}{\ODw@North*}%
    }{%
      \IfBooleanTF{#2}{\ODw@N}{\ODw@North}%
    }% TF#1
  \egroup%
```

```
  \xspace%
}% North
%
\def\north{\North*}
```

\East `\East[*!]`

\east `\east`

```
\NewDocumentCommand\East{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF#1{%
      \IfBooleanTF{#2}{\ODw@E*}{\ODw@East*}%
    }{%
      \IfBooleanTF{#2}{\ODw@E}{\ODw@East}%
    }% TF#1
  \egroup%
  \xspace%
}% East
%
\def\east{\East*}
```

\South `\South[*!]`

\south `\south`

```
\NewDocumentCommand\South{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF#1{%
      \IfBooleanTF{#2}{\ODw@S*}{\ODw@South*}%
    }{%
      \IfBooleanTF{#2}{\ODw@S}{\ODw@South}%
    }% TF#1
  \egroup%
  \xspace%
}% South
%
\def\south{\South*}
```

\West `\West[*!]`

\west \west

```
\NewDocumentCommand\West{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF#1{%
      \IfBooleanTF{#2}{\ODw@W*}{\ODw@West*}%
    }{%
      \IfBooleanTF{#2}{\ODw@W}{\ODw@West}%
    }% TF#1
  \egroup%
  \xspace%
}% West
%
\def\west{\West*}
```

\NorthSouth \NorthSouth[*!]

\northsouth \northsouth

```
\NewDocumentCommand\NorthSouth{s t!}{%
  \bgroup%
   \ODw@OtherFont%
   \IfBooleanTF#1{%
     \IfBooleanTF{#2}{\North*!--\South*!}{\North*--\South*}%
   }{%
     \IfBooleanTF{#2}{\North!--\South!}{\North--\South}%
   }%
  \egroup%
  \xspace%
}% NorthSouth
%
\def\northsouth{\NorthSouth*}
```

\EastWest \EastWest[*!]

\eastwest \eastwest

```
\NewDocumentCommand\EastWest{s t!}{%
  \bgroup%
    \ODw@OtherFont%
   \IfBooleanTF#1{%
     \IfBooleanTF{#2}{\East*!--\West*!}{\East*--\West*}%
```

```
    }{%
      \IfBooleanTF{#2}{\East!--\West!}{\East--\West}%
    }%
  \egroup%
  \xspace%
}% EastWest
%
\def\eastwest{\EastWest*}
```

Next we define macros that translate the short form of the directions into the active language.

\ODw@N
\ODw@E
\ODw@S
\ODw@W

```
\def\ODw@N{%
  \@ifstar{\GetTranslation{N-(ODw)}}%
          {\GetTranslation{n-(ODw)}}%
}
\def\ODw@E{%
  \@ifstar{\GetTranslation{E-(ODw)}}%
          {\GetTranslation{e-(ODw)}}%
}
\def\ODw@S{%
  \@ifstar{\GetTranslation{S-(ODw)}}%
          {\GetTranslation{s-(ODw)}}%
}
\def\ODw@W{%
  \@ifstar{\GetTranslation{W-(ODw)}}%
          {\GetTranslation{w-(ODw)}}%
}
```

\ODw@NS
\ODw@EW

```
\def\ODw@NS{\ODw@N--\ODw@S}
\def\ODw@EW{\ODw@E--\ODw@W}
```

\ODw@North
\ODw@East
\ODw@South
\ODw@West

```
\def\ODw@North{%
  \@ifstar{\GetTranslation{North-(ODw)}}%
          {\GetTranslation{north-(ODw)}}%
}
\def\ODw@East{%
  \@ifstar{\GetTranslation{East-(ODw)}}%
          {\GetTranslation{east-(ODw)}}%
}
\def\ODw@South{%
```

```
  \@ifstar{\GetTranslation{South-(ODw)}}%
          {\GetTranslation{south-(ODw)}}%
}
\def\ODw@West{%
  \@ifstar{\GetTranslation{West-(ODw)}}%
          {\GetTranslation{west-(ODw)}}%
}
```

6.2.3 Non-Bid Calls

\Pass `\Pass[*!]`

\pass `\pass`

```
\NewDocumentCommand\Pass{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{Pass!-(ODw)}}%
        {\GetTranslation{Pass-(ODw)}}%
    }{%
     \IfBooleanTF{#2}%
       {\GetTranslation{pass!-(ODw)}}%
       {\GetTranslation{pass-(ODw)}}%
    }% TF#1
  \egroup%
  \xspace%
}% Pass
%
\def\pass{\Pass*}
```

\Allpass `\Allpass[*!]`

\allpass `\allpass`

```
\NewDocumentCommand\Allpass{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF#1{%
      \IfBooleanTF{#2}%
        {\GetTranslation{AP-(ODw)}}%
```

```
        {\GetTranslation{All pass-(ODw)}}%
    }{%
     \IfBooleanTF{#2}%
       {\GetTranslation{ap-(ODw)}}%
       {\GetTranslation{all pass-(ODw)}}%
    }% TF#1
  \egroup%
  \xspace%
}% Allpass
%
\def\allpass{\Allpass*}
```

\Double `\Double[*!]`

\double `\double`

```
\NewDocumentCommand\Double{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF#1{%
      \IfBooleanTF{#2}%
       {\GetTranslation{Dbl-(ODw)}}%
       {\GetTranslation{Double-(ODw)}}%
    }{%
     \IfBooleanTF{#2}%
       {X}%
       {\GetTranslation{double-(ODw)}}%
    }% TF#1
  \egroup%
  \xspace%
}% Double
%
\def\double{\Double*}
```

\Redouble `\Redouble[*!]`

\redouble `\redouble`

```
\NewDocumentCommand\Redouble{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF#1{%
      \IfBooleanTF{#2}%
```

```
        {\GetTranslation{ReDbl-(ODw)}}%
        {\GetTranslation{ReDouble-(ODw)}}%
   }{%
     \IfBooleanTF{#2}%
       {\mbox{X\kern-0.1em X}}%
       {\GetTranslation{redouble-(ODw)}}%
    }% TF#1
  \egroup%
  \xspace%
}% Redouble
%
\def\redouble{\Redouble*}
```

6.2.4 Bidding Diagrams

\ODw@FirstBidCol `\ODw@FirstBidCol{⟨N|S|E|W⟩}`

`\ODw@FirstBidCol` determines which player starts in the first bidding column. We also take care that the real world name of the players are kept associated with the columns. For `\ODw@BidderX` (X=I,II,III,IV) we define two macros: `\ODw@BidderX` and `\ODw@BidderX*` in one. When these macros are called in the `bidding` diagram, the starred version writes the short notation whereas the unstarred one writes the long version.

```
\newcommand\ODw@FirstBidCol[1]{%
 \IfEqCase{#1}{%
  {N}{%
   \gdef\ODw@BidderI{\@ifstar{\ODw@North*}{\ODw@N*}}%
   \gdef\ODw@BidderII{\@ifstar{\ODw@East*}{\ODw@E*}}%
   \gdef\ODw@BidderIII{\@ifstar{\ODw@South*}{\ODw@S*}}%
   \gdef\ODw@BidderIV{\@ifstar{\ODw@West*}{\ODw@W*}}%
   \gdef\ODw@NameI{\ODw@North@Name}%
   \gdef\ODw@NameII{\ODw@East@Name}%
   \gdef\ODw@NameIII{\ODw@South@Name}%
   \gdef\ODw@NameIV{\ODw@West@Name}%
  }%
  {E}{%
   \gdef\ODw@BidderI{\@ifstar{\ODw@East*}{\ODw@E*}}%
   \gdef\ODw@BidderII{\@ifstar{\ODw@South*}{\ODw@S*}}%
   \gdef\ODw@BidderIII{\@ifstar{\ODw@West*}{\ODw@W*}}%
   \gdef\ODw@BidderIV{\@ifstar{\ODw@North*}{\ODw@N*}}%
   \gdef\ODw@NameI{\ODw@East@Name}%
```

```
    \gdef\ODw@NameII{\ODw@South@Name}%
    \gdef\ODw@NameIII{\ODw@West@Name}%
    \gdef\ODw@NameIV{\ODw@North@Name}%
   }%
   {S}{%
    \gdef\ODw@BidderI{\@ifstar{\ODw@South*}{\ODw@S*}}%
    \gdef\ODw@BidderII{\@ifstar{\ODw@West*}{\ODw@W*}}%
    \gdef\ODw@BidderIII{\@ifstar{\ODw@North*}{\ODw@N*}}%
    \gdef\ODw@BidderIV{\@ifstar{\ODw@East*}{\ODw@E*}}%
    \gdef\ODw@NameI{\ODw@South@Name}%
    \gdef\ODw@NameII{\ODw@West@Name}%
    \gdef\ODw@NameIII{\ODw@North@Name}%
    \gdef\ODw@NameIV{\ODw@East@Name}%
   }%
   {W}{%
    \gdef\ODw@BidderI{\@ifstar{\ODw@West*}{\ODw@W*}}%
    \gdef\ODw@BidderII{\@ifstar{\ODw@North*}{\ODw@N*}}%
    \gdef\ODw@BidderIII{\@ifstar{\ODw@East*}{\ODw@E*}}%
    \gdef\ODw@BidderIV{\@ifstar{\ODw@South*}{\ODw@S*}}%
    \gdef\ODw@NameI{\ODw@West@Name}%
    \gdef\ODw@NameII{\ODw@North@Name}%
    \gdef\ODw@NameIII{\ODw@East@Name}%
    \gdef\ODw@NameIV{\ODw@South@Name}%
   }%
  }% IfEqCase
}% ODw@FirstBidCol
```

Next we define the real world names for the **N–S** and the **E–W** bidders. We use `\ODw@All@Names` as variable to test if we have names for bidders at all: If it is empty, then no names were defined.

`\namesNS` `\namesNS{⟨N-name⟩}{⟨S-name⟩}`

```
\newcommand\namesNS[2]{%
  \gdef\ODw@North@Name{#1}%
  \gdef\ODw@South@Name{#2}%
  \gdef\ODw@All@Names{#1#2\ODw@East@Name\ODw@West@Name}%
}% namesNS
```

`\namesEW` `\namesEW{⟨E-name⟩}{⟨W-name⟩}`

```
\newcommand\namesEW[2]{%
  \gdef\ODw@East@Name{#1}%
  \gdef\ODw@West@Name{#2}%
```

```
  \gdef\ODw@All@Names{#1#2\ODw@North@Name\ODw@South@Name}%
}% namesEW
```

6.2.5 Diagram Hands

Here we implement commands that store the cards that each player holds. E.g. `\northhand` defines the complete hand, i.e. all 4 suits for the **North**-player. They have an optional parameter (an offset, default `0pt`) to finetune the distance to the compass. In `\ODw@Nhand` and `\ODw@Shand` we use a makebox to prevent that its width goes beyond the `NESW` compass (but this can interfere with the Right-U/L-Legend). Within these macros, we check the consistency of the cards in the hand and also store card-information to check a complete deal later.

`\northhand` `\northhand[⟨v-offset⟩]{⟨Sp⟩}{⟨He⟩}{⟨Di⟩}{⟨Cl⟩}`

```
\northhand[v-offset]{Sp}{He}{Di}{Cl}
              1      2   3   4   5
\newcommand\northhand[5][0pt]{%
% check that north has 13 cards
  \ODw@ChkNrOfCards{#2#3#4#5}{\north}%
  \gdef\ODw@NSpades{#2}%   Save norths cards
  \gdef\ODw@NHearts{#3}%   of all suits
  \gdef\ODw@NDiamonds{#4}% for later
  \gdef\ODw@NClubs{#5}%    checking
```

`\ODw@Nhand` We fit the **North** hand in a box to avoid that a very long suit shifts the **East** hand to the right

```
  \gdef\ODw@Nhand{%
    \makebox[\ODw@Compasssize + 2ex][l]{%
      \ODw@hand{t}{#2}{#3}{#4}{#5}%
    }%
    \vspace{#1}%
  }%

}% northhand
```

`\easthand` `\easthand[⟨h-offset⟩]{⟨Sp⟩}{⟨He⟩}{⟨Di⟩}{⟨Cl⟩}`

```
\easthand[h-offset]{Sp}{He}{Di}{Cl}
              1      2   3   4   5
\newcommand\easthand[5][0pt]{%
% check that east has 13 cards
  \ODw@ChkNrOfCards{#2#3#4#5}{\east}%
```

```
  \gdef\ODw@ESpades{#2}%   Save easts cards
  \gdef\ODw@EHearts{#3}%   of all suits
  \gdef\ODw@EDiamonds{#4}% for later
  \gdef\ODw@EClubs{#5}%    checking
```

\ODw@Ehand

```
  \gdef\ODw@Ehand{%
    \hspace{#1}%
    \ODw@hand{c}{#2}{#3}{#4}{#5}%
  }%
}% easthand
```

\southhand `\southhand[⟨v-offset⟩]{⟨Sp⟩}{⟨He⟩}{⟨Di⟩}{⟨Cl⟩}`

```
\southhand[v-offset]{Sp}{He}{Di}{Cl}
             1      2   3   4   5
```

```
\newcommand\southhand[5][0pt]{%
% check that south has 13 cards
  \ODw@ChkNrOfCards{#2#3#4#5}{\south}%
  \gdef\ODw@SSpades{#2}%   Save souths cards
  \gdef\ODw@SHearts{#3}%   of all suits
  \gdef\ODw@SDiamonds{#4}% for later
  \gdef\ODw@SClubs{#5}%    checking
```

\ODw@Shand We fit the **South** hand in a box to avoid that a very long suit shifts the **East** hand to the right

```
  \gdef\ODw@Shand{%
    \parbox[b]{\ODw@Compasssize + 2ex}{%
      \vspace*{#1}\par%
      \makebox[0pt][l]{%
        \ODw@hand{b}{#2}{#3}{#4}{#5}%
      }%
    }%
  }%
}% southhand
```

\westhand `\westhand[⟨h-offset⟩]{⟨Sp⟩}{⟨He⟩}{⟨Di⟩}{⟨Cl⟩}`

```
\westhand[h-offset]{Sp}{He}{Di}{Cl}
             1      2   3   4   5
```

```
\newcommand\westhand[5][0pt]{%
% check that west has 13 cards
  \ODw@ChkNrOfCards{#2#3#4#5}{\west}%
```

```
628   \gdef\ODw@WSpades{#2}%   Save wests cards
629   \gdef\ODw@WHearts{#3}%   of all suits
630   \gdef\ODw@WDiamonds{#4}% for later
631   \gdef\ODw@WClubs{#5}%    checking
```

\ODw@Whand

```
632   \gdef\ODw@Whand{%
633     \ODw@hand{c}{#2}{#3}{#4}{#5}%
634     \hspace*{#1}%
635   }%
636 }% westhand
```

6.2.6 A Single Hand

Sometimes we want to show only (the cards of) one single hand.

\hand `\hand[*!-][⟨pos⟩]{⟨Sp⟩}{⟨He⟩}{⟨Di⟩}{⟨Cl⟩}`

```
637 \NewDocumentCommand\hand{s t! t- O{c}mmmm}{%
```

```
\hand* ! -[pos]{Sp}{He}{Di}{Cl}
     1 2 3  4   5   6   7   8
```

This macro has 3 optional tokens that act as follows:
Naked mode: Displays the hand horizontally, left aligned.
`*` mode: Displays the hand horizontally, centered.
`!` mode: Displays the hand vertically, left aligned.
`*!` mode: Displays the hand vertically, centered.
`-` mode: Output is suppressed, it is stored in a savebox for later use.

The 4th argument is the aligning (used only in case of a vertical hand): default **c**. The rest of the arguments denote the cards of the suits.

We first check that this hand has 13 cards. Then we check for each suit that there are no multiples. Finally we store the hand in `\ODw@Hand@Box`. If output is not suppressed, we write the hand with `\usebox\ODw@Hand@Box`.

```
638   \ODw@ChkNrOfCards{#5#6#7#8}{Hand}%
639   \ODw@ChkSameCards{#5}{\Sp}%
640   \ODw@ChkSameCards{#6}{\He}%
641   \ODw@ChkSameCards{#7}{\Di}%
642   \ODw@ChkSameCards{#8}{\Cl}%
643   \global\sbox{\ODw@Hand@Box}{%
644     \bgroup%
645       \ODw@GameSize%
```

```
      \IfBooleanTF{#2}%
        {\ODw@vhand[#4]{#5}{#6}{#7}{#8}}%
        {\ODw@hhand{#5}{#6}{#7}{#8}}%
    \egroup%
  }% sbox
  \IfBooleanTF{#3}{}{%
   \IfBooleanTF{#1}%
      {{\centering \usebox{\ODw@Hand@Box}\par}}%
      {\usebox{\ODw@Hand@Box}}%
  }%
}% hand
```

\ODw@hhand `\ODw@hhand{⟨Sp⟩}{⟨He⟩}{⟨Di⟩}{⟨Cl⟩}`

Displays a hand horizontally (e.g. ♠ xxxx ♥ xxx ♦ xxx ♣ xxx). We use a `tabular` with 4 columns in 1 row to print the 4 suits.

```
\newcommand\ODw@hhand[4]{%
  \bgroup%
  \def\xspace{}% undo xspace locally
                % it screws the distance between suit and cards
  \setlength\tabcolsep{1\ODw@Card@Skip}%
  \begin{tabular}{llll}
  % we can't use \suit here: it would cause double checks!
    \Sp\hspace{0.3em}\ODw@Cards{#1} &%
    \He\hspace{0.3em}\ODw@Cards{#2} &%
    \Di\hspace{0.3em}\ODw@Cards{#3} &%
    \Cl\hspace{0.3em}\ODw@Cards{#4} \\
  \end{tabular}%
  \egroup%
}% ODw@hhand
```

\ODw@vhand `\ODw@vhand[⟨pos⟩]{⟨Sp⟩}{⟨He⟩}{⟨Di⟩}{⟨Cl⟩}`

Display a hand vertically (e.g. ♠ xxxx ♥ xxx ♦ xxx ♣ xxx). We call `\ODw@hand` to do the job.

```
\newcommand\ODw@vhand[5][c]{%
  \ODw@hand{#1}{#2}{#3}{#4}{#5}%
}% ODw@vhand
```

\ODw@hand `\ODw@hand[⟨pos⟩]{⟨Sp⟩}{⟨He⟩}{⟨Di⟩}{⟨Cl⟩}`

We put a hand and some spacing in a `tabular` by reading the cards

for each suit, making the lines more tense with the `\\[-0.5ex]`. Empty hands are discarded completely.

```
\newcommand\ODw@hand[5]{%
```

```
ODw@hand{pos}{spades}{hearts}{diamonds}{clubs}
          1     2       3        4        5
```

First we test if the hand is completely empty; only if not, we output something.

```
  \ifthenelse{\equal{#2#3#4#5}{}}{}{%
    \setlength\tabcolsep{\ODw@Card@Skip}%
%JW    \ODw@GameSize% XYX JW 30.04.2018
    \begin{tabular}[#1]{ll}%
  % we can't use \suit here: it would cause double checks!
      \Sp & \ODw@Cards{#2}\\[-0.5ex]
      \He & \ODw@Cards{#3}\\[-0.5ex]
      \Di & \ODw@Cards{#4}\\[-0.5ex]
      \Cl & \ODw@Cards{#5}\\
    \end{tabular}%
  }% ifthenelse
}% ODw@hand
```

6.2.7 Suits

In some cases, we need only a collection of cards, without a suit symbol.

`\onesuitAll` `\onesuitAll[*!]{⟨N⟩}{⟨S⟩}{⟨E⟩}{⟨W⟩}`

Display the cards of one suit in a **NS**–**EW** diagram, with the **N**-, **E**-, **S**- and **W**-hand.

```
\NewDocumentCommand\onesuitAll{s t! mmmm}{%
```

```
onesuitAll* !{N-hand}{S-hand}{E-hand}{W-hand}
          1 2   3       4       5       6
Naked version:  Use a □
* version: Display the diagram centered
! version: use the NESW compass
```

First we test that we have no multiple cards in the suit. Then we use a `tabular` to place the cards around a compass or around a box.

```
  \ODw@ChkSameCards{#3#4#5#6}{}%
  \IfBooleanTF#1{\begin{center}}{}%
  \bgroup%
    \ODw@GameSize%
```

```
    \setlength\tabcolsep{0em}%
\begin{tabular}{@{}r@{ }c@{ }l@{}}%
%\begin{tabular}{@{}rcl@{}}%
       & \ODw@Cards{#3} \IfBooleanTF#2{\\[-0.2em]}{\\}%
 \ODw@Cards{#6} & \IfBooleanTF#2{\ODw@Compass}{\ODw@Box}%
         & \ODw@Cards{#5} \IfBooleanTF#2{\\[-0.2em]}{\\}%
                & \ODw@Cards{#4}\\
    \end{tabular}%
  \egroup%
  \IfBooleanTF#1{\end{center}}{}%
}% onesuitAll
```

\onesuitNS `\onesuitNS[*!]{⟨N⟩}{⟨S⟩}`

Display a suit as NS-diagram. Similar to `\onesuitAll` but with only N- and S-hand.

```
\NewDocumentCommand\onesuitNS{s t! mm}{%
```

```
onesuitNS* !{N-hand}{S-hand}
         1 2   3       4
Naked version:  Use a □
* version:  Display the diagram centered
! version: use the NESW compass
```

```
  \ODw@ChkSameCards{#3#4}{}%
  \IfBooleanTF#1{\begin{center}}{}%
  \bgroup%
    \ODw@GameSize%
    \begin{tabular}{@{}c@{}}%
      \ODw@Cards{#3}\IfBooleanTF#2{\\[-0.2em]}{\\}%
        \IfBooleanTF#2{\ODw@Compass\\[-0.2em]}{\ODw@Box\\}%
      \ODw@Cards{#4}%
    \end{tabular}%
  \egroup%
  \IfBooleanTF#1{\end{center}}{}%
}% onesuitNS
```

\onesuitEW `\onesuitEW[*!]{⟨E⟩}{⟨W⟩}`

Display a suit as EW diagram. Similar to `\onesuitAll` but with only E- and W-hand.

```
\NewDocumentCommand\onesuitEW{s t! mm}{%
```

```
onesuitEW* !{E-hand}{W-hand}
         1 2   3       4
Naked version:  Use a □
```

```
* version: Display the diagram centered
! version: use the NESW compass
```

```
  \ODw@ChkSameCards{#3#4}{}%
  \IfBooleanTF#1{\begin{center}}{}%
  \bgroup%
    \ODw@GameSize%
    \begin{tabular}{@{}r@{ }c@{ }l@{}}%
      \ODw@Cards{#4} &%
      \IfBooleanTF#2{\ODw@Compass}{\ODw@Box} &%
         \ODw@Cards{#3} \\%
    \end{tabular}%
  \egroup%
   \IfBooleanTF#1{\end{center}}{}%
}% onesuitEW
```

\onesuitNE `\onesuitNE[*!]{⟨N⟩}{⟨E⟩}`

Display a suit as NE diagram. Similar to `\onesuitAll` but with only N- and E-hand.

```
\NewDocumentCommand\onesuitNE{s t! mm}{%
```

```
onesuitNE* !{N-hand}{E-hand}
         1 2   3       4
Naked version:  Use a □
* version: Display the diagram centered
! version: use the NESW compass
```

```
  \ODw@ChkSameCards{#3#4}{}%
  \IfBooleanTF#1{\begin{center}}{}%
  \bgroup%
    \ODw@GameSize%
    \begin{tabular}[b]{c@{ }l@{}}%
      \ODw@Cards{#3} \\%
      \IfBooleanTF#2{\ODw@Compass}{\ODw@Box} &%
         \ODw@Cards{#4} \\%
    \end{tabular}%
  \egroup%
   \IfBooleanTF#1{\end{center}}{}%
}% onesuitEW
```

\onesuitNW `\onesuitNW[*!]{⟨N⟩}{⟨W⟩}`

Display a suit as NW diagram. Similar to `\onesuitAll` but with only N- and W-hand.

```
\NewDocumentCommand\onesuitNW{s t! mm}{%
```

```
onesuitNW* !{N-hand}{W-hand}
         1 2   3       4
Naked version:  Use a □
* version: Display the diagram centered
! version: use the NESW compass
```

```
743   \ODw@ChkSameCards{#3#4}{}%
744   \IfBooleanTF#1{\begin{center}}{}%
745   \bgroup%
746     \ODw@GameSize%
747     \begin{tabular}[b]{r@{ }c}%
748                         & \ODw@Cards{#3} \\%
749       \ODw@Cards{#4} & \IfBooleanTF#2{\ODw@Compass}{\ODw@Box} \\%
750     \end{tabular}%
751   \egroup%
752    \IfBooleanTF#1{\end{center}}{}%
753 }% onesuitEW
```

\suit `\suit[⟨suit symbol⟩]{⟨cards⟩}`

Command for displaying the cards of a suit. With the optional argument one can add a suit symbol to the suit.

```
754 \newcommand\suit[2][]{%
```

```
\suit[suit]{cards}
       1      2
```

```
755   \ODw@ChkNrOfCards{#2}{suit}%
756   \ODw@ChkSameCards{#2}{#1}%
757   \bgroup% keep font change local
758 %JW    \ODw@GameSize%
759     #1\ODw@Cards{#2}%
760   \egroup%
761   \xspace%
762 }% suit
```

\ODw@Cards `\ODw@Cards{⟨cards⟩}`

This macro gets called by `\suit` and all commands that process hands. It processes the ranks of the cards. Between ranks some space is typeset. Care is taken that `T` becomes 10 and honor cards are translated into the active language.

```
763 \newcommand{\ODw@Cards}[1]{%
```

We enable the cardskip and call `\ODw@Xfer` to do the job

```
  \OdW@CardSkiptrue%
  \ODw@Xfer{#1}%
}% ODw@Cards
```

6.2.8 Card Diagrams

Next we define several diagrams with hands around the compass. `\ODw@GameSize` sets the size of the compass, the directions and the hands, according to the actual font or font-size. `\ODw@LeftUpperText` etc. displays extra text, that appear in the left upper, etc. corner of the diagram. If `\headlinetext` (`\footlinetext`) is the empty string, we set boolean `ODw@EmptyHeader` (`ODw@EmptyFooter`) to true[16]. This value is used in `\ODw@ProcessHeader` to conditionally span the 3 columns with the headline- (footline)text. We first store the diagram in a box, so we can calculate its width and use that as a size to limit the header/footer texts.

`\showAll` `\showAll[*+][⟨pos⟩]`

Define the diagram, showing the cards for **All** hands.

```
\NewDocumentCommand\showAll{s t+ O{c}}{%
showAll* + [pos]
       1 2   3
Display the NS--EW diagram, defined by \northhand, etc
* Version: Display the diagram centered
+ Version: Also display a bidding diagram
pos: aligning, default= c
```

First, for all suits we store all cards of all sides together. Next we check the consistency of all complete suits. The individual suits of each player have already been checked as we defined the hands. Finally we print the hands around the compass using a `tabular`, taking care of the additions above, below and in the corners of the diagram. Before we write the diagram, we store it in an sbox to calulate its width, so we can use that in other places.

```
  \gdef\ODw@Spades{% store all Spades together
    \ODw@NSpades\ODw@ESpades\ODw@SSpades\ODw@WSpades%
  }%
  \gdef\ODw@Hearts{% store all Hearts together
    \ODw@NHearts\ODw@EHearts\ODw@SHearts\ODw@WHearts%
```

[16] see page 35, last paragraph why this is necessary

```
  }%
  \gdef\ODw@Diamonds{% store all Diamonds together
    \ODw@NDiamonds\ODw@EDiamonds\ODw@SDiamonds\ODw@WDiamonds%
  }%
  \gdef\ODw@Clubs{% store all Clubs together
    \ODw@NClubs\ODw@EClubs\ODw@SClubs\ODw@WClubs%
  }%
  % check for multiple and nr. of cards
  \ODw@ChkNrOfCards{\ODw@Spades}{\Sp}%
  \ODw@ChkSameCards{\ODw@Spades}{\Sp}%
  \ODw@ChkNrOfCards{\ODw@Hearts}{\He}%
  \ODw@ChkSameCards{\ODw@Hearts}{\He}%
  \ODw@ChkNrOfCards{\ODw@Diamonds}{\Di}%
  \ODw@ChkSameCards{\ODw@Diamonds}{\Di}%
  \ODw@ChkNrOfCards{\ODw@Clubs}{\Cl}%
  \ODw@ChkSameCards{\ODw@Clubs}{\Cl}%
  \IfBooleanTF#1{\begin{center}}{}%
  \bgroup%
    \setlength\tabcolsep{0em}%
    \ODw@GameSize%
    \ODw@TestIfEmpty{\ODw@HeaderText}{ODw@EmptyHeader}%
    \ODw@TestIfEmpty{\ODw@FooterText}{ODw@EmptyFooter}%
  % sbox1 necessary to calc. |Compasssize| for |Nhand|
    \sbox1{\ODw@Compass}%
    \sbox0{%
\begin{tabular}[#3]{@{}r@{}c@{}l@{}}%
  \ODw@LeftUpperText & \ODw@Nhand & \ODw@RightUpperText\\
  \ODw@Whand         & \usebox{1} & \ODw@Ehand\\
  \ODw@LeftLowerText & \ODw@Shand & \ODw@RightLowerText\\
\end{tabular}%
    }% sbox
    \ODw@gsetlength{\ODw@Diagram@Width}{\wd0}%
    \begin{tabular}[#3]{@{}r@{}c@{}l@{}}%
      \ODw@ProcessHeader{3}% span 3 columns
      \ODw@LeftUpperText & \ODw@Nhand & \ODw@RightUpperText\\
      \ODw@Whand         & \usebox{1} & \ODw@Ehand\\
      \ODw@LeftLowerText & \ODw@Shand & \ODw@RightLowerText\\
      \ODw@ProcessFooter{3}% span 3 columns
    \end{tabular}%
    \IfBooleanTF#2{%
 %  needed for \ODw@CondNewLine
      \setlength{\ODw@Bid@Width}{\wd\ODw@BidBox}%
      \ODw@CondNewLine%
```

```
      \usebox{\ODw@BidBox}%
    }{}%
  \egroup%
  \IfBooleanTF#1{\end{center}}{}%
}% showAll
```

\showNS `\showNS[*+][⟨pos⟩]`

Define the diagram, showing the cards for the **N–S** hands.

```
\NewDocumentCommand\showNS{s t+ O{c}}{%
```

```
showNS* +[pos]
      1 2  3
Display the NS diagram, defined by \northhand, etc
* Version: Display the diagram centered
+ Version: Also display a bidding diagram
pos: aligning, default c
```

Description: similar to `\showAll`

```
% For all suits store all cards of north and south together
  \gdef\ODw@Spades{\ODw@NSpades\ODw@SSpades}%
  \gdef\ODw@Hearts{\ODw@NHearts\ODw@SHearts}%
  \gdef\ODw@Diamonds{\ODw@NDiamonds\ODw@SDiamonds}%
  \gdef\ODw@Clubs{\ODw@NClubs\ODw@SClubs}%
  \ODw@ChkSameCards{\ODw@Spades}{\Sp}%
  \ODw@ChkSameCards{\ODw@Hearts}{\He}%
  \ODw@ChkSameCards{\ODw@Diamonds}{\Di}%
  \ODw@ChkSameCards{\ODw@Clubs}{\Cl}%
  \IfBooleanTF#1{\begin{center}}{}%
  \bgroup%
    \setlength\tabcolsep{0em}%
    \ODw@GameSize%
    \ODw@TestIfEmpty{\ODw@HeaderText}{ODw@EmptyHeader}%
    \ODw@TestIfEmpty{\ODw@FooterText}{ODw@EmptyFooter}%
```

Here we store the width of the diagram **without** the header and footer. So we can limit their width to the diagramwidth.

```
  % sbox1 necessary to calc. |Compasssize| for |Nhand|
    \sbox1{\ODw@Compass}%
    \sbox0{%
  \begin{tabular}[#3]{@{}r@{}c@{}l@{}}%
      & \ODw@Nhand & \\
      & \usebox{1} & \\
      & \ODw@Shand & \\
```

```
  \end{tabular}%
    }% sbox0
  \ODw@gsetlength\ODw@Diagram@Width{\wd0}%
  \begin{tabular}[#3]{@{}r@{}c@{}l@{}}%
    \ODw@ProcessHeader{3}% span 3 columns
     & \ODw@Nhand & \\
     & \usebox{1} & \\
     & \ODw@Shand & \\
    \ODw@ProcessFooter{3}% span 3 columns
  \end{tabular}%
   \IfBooleanTF#2{%
 % necessary for \ODw@CondNewLine
     \setlength{\ODw@Bid@Width}{\wd\ODw@BidBox}%
     \ODw@CondNewLine%
     \usebox{\ODw@BidBox}%
   }{}%
 \egroup%
 \IfBooleanTF#1{\end{center}}{}%
}% showNS
```

\showEW `\showEW[*+][⟨pos⟩]`

Define the diagram, showing the cards for the **E–W** hands.

```
\NewDocumentCommand\showEW{s t+ O{c}}{%
```

```
showEW* + [pos]
      1 2   3
Display the EW diagram, defined by \easthand, etc
* Version: Display the diagram centered
+ Version: Also display a bidding diagram
pos: aligning, default c
```

Description: similar to `\showAll`

```
% For all suits put all cards of east and west together
  \gdef\ODw@Spades{\ODw@ESpades\ODw@WSpades}%
  \gdef\ODw@Hearts{\ODw@EHearts\ODw@WHearts}%
  \gdef\ODw@Diamonds{\ODw@EDiamonds\ODw@WDiamonds}%
  \gdef\ODw@Clubs{\ODw@EClubs\ODw@WClubs}%
  \ODw@ChkSameCards{\ODw@Spades}{\Sp}%
  \ODw@ChkSameCards{\ODw@Hearts}{\He}%
  \ODw@ChkSameCards{\ODw@Diamonds}{\Di}%
  \ODw@ChkSameCards{\ODw@Clubs}{\Cl}%
  \IfBooleanTF#1{\begin{center}}{}%
  \bgroup%
```

```
    \setlength\tabcolsep{0em}%
    \ODw@GameSize%
    \ODw@TestIfEmpty{\ODw@HeaderText}{ODw@EmptyHeader}%
    \ODw@TestIfEmpty{\ODw@FooterText}{ODw@EmptyFooter}%
    \sbox0{%
      \begin{tabular}[#3]{@{}r@{}c@{}l@{}}%
        \ODw@Whand       & \ODw@Compass & \ODw@Ehand \\
      \end{tabular}%
    }% sbox
    \ODw@gsetlength{\ODw@Diagram@Width}{\wd0}%
    \begin{tabular}[#3]{@{}r@{}c@{}l@{}}%
      \ODw@ProcessHeader{3}% span 3 columns
                          &              & \\
      \ODw@Whand       & \ODw@Compass & \ODw@Ehand \\
      \ODw@ProcessFooter{3}% span 3 columns
    \end{tabular}%
   \IfBooleanTF#2{%
 % necessary for \ODw@CondNewLine
     \setlength{\ODw@Bid@Width}{\wd\ODw@BidBox}%
     \ODw@CondNewLine%
     \usebox{\ODw@BidBox}%
   }{}%
  \egroup%
  \IfBooleanTF#1{\end{center}}{}%
}% showEW
```

\showNE `\showNE[*+][⟨pos⟩]`

Define the diagram, showing the cards for the **N**–**E** hands.

```
\NewDocumentCommand\showNE{s t+ O{c}}{%
```

```
showNE* +[pos]
      1 2  3
Display the NE diagram, defined by \northhand, etc
* Version: Display the diagram centered
+ Version: Also display a bidding diagram
pos: aligning, default c
```

Description: similar to `\showAll`

```
% For all suits put all cards of north and east together
  \gdef\ODw@Spades{\ODw@NSpades\ODw@ESpades}%
  \gdef\ODw@Hearts{\ODw@NHearts\ODw@EHearts}%
  \gdef\ODw@Diamonds{\ODw@NDiamonds\ODw@EDiamonds}%
  \gdef\ODw@Clubs{\ODw@NClubs\ODw@EClubs}%
  \ODw@ChkSameCards{\ODw@Spades}{\Sp}%
```

```
  \ODw@ChkSameCards{\ODw@Hearts}{\He}%
  \ODw@ChkSameCards{\ODw@Diamonds}{\Di}%
  \ODw@ChkSameCards{\ODw@Clubs}{\Cl}%
  \IfBooleanTF#1{\begin{center}}{}%
  \bgroup%
    \setlength\tabcolsep{0em}%
    \ODw@GameSize%
    \ODw@TestIfEmpty{\ODw@HeaderText}{ODw@EmptyHeader}%
    \ODw@TestIfEmpty{\ODw@FooterText}{ODw@EmptyFooter}%
  % sbox1 necessary to calc. |Compasssize| for |Nhand|
    \sbox1{\ODw@Compass}%
    \sbox0{%
      \begin{tabular}[#3]{@{}c@{}l@{}}%
        \ODw@Nhand & \ODw@RightUpperText\\
        \usebox{1} & \ODw@Ehand\\
      \end{tabular}%
    }% sbox
    \ODw@gsetlength{\ODw@Diagram@Width}{\wd0}%
    \begin{tabular}[#3]{@{}c@{}l@{}}%
      \ODw@ProcessHeader{2}% span 2 columns
      \ODw@Nhand & \ODw@RightUpperText\\
      \usebox{1} & \ODw@Ehand\\
      \ODw@ProcessFooter{2}% span 2 columns
    \end{tabular}%
   \IfBooleanTF#2{%
 % necessary for \ODw@CondNewLine
     \setlength{\ODw@Bid@Width}{\wd\ODw@BidBox}%
     \ODw@CondNewLine%
     \usebox{\ODw@BidBox}%
   }{}%
  \egroup%
  \IfBooleanTF#1{\end{center}}{}%
}% showNE
```

\showNW `\showNW[*+][⟨pos⟩]`

Define the diagram, showing the cards for the **N**–**W** hands.

```
\NewDocumentCommand\showNW{s t+ O{c}}{%
```

```
showNW* +[pos]
      1 2  3
Display the NW diagram, defined by \northhand, etc
* Version: Display the diagram centered
+ Version: Also display a bidding diagram
```

pos: aligning, default c

Description: similar to `\showAll`

```
% For all suits put all cards of north and west together
  \gdef\ODw@Spades{\ODw@NSpades\ODw@WSpades}%
  \gdef\ODw@Hearts{\ODw@NHearts\ODw@WHearts}%
  \gdef\ODw@Diamonds{\ODw@NDiamonds\ODw@WDiamonds}%
  \gdef\ODw@Clubs{\ODw@NClubs\ODw@WClubs}%
  \ODw@ChkSameCards{\ODw@Spades}{\Sp}%
  \ODw@ChkSameCards{\ODw@Hearts}{\He}%
  \ODw@ChkSameCards{\ODw@Diamonds}{\Di}%
  \ODw@ChkSameCards{\ODw@Clubs}{\Cl}%
  \IfBooleanTF#1{\begin{center}}{}%
  \bgroup%
    \setlength\tabcolsep{0em}%
    \ODw@GameSize%
    \ODw@TestIfEmpty{\ODw@HeaderText}{ODw@EmptyHeader}%
    \ODw@TestIfEmpty{\ODw@FooterText}{ODw@EmptyFooter}%
  % sbox1 necessary to calc. |Compasssize| for |Nhand|
    \sbox1{\ODw@Compass}%
    \sbox0{%
      \begin{tabular}[#3]{@{}c@{}l@{}}%
        \ODw@LeftUpperText & \ODw@Nhand\\
        \ODw@Whand         & \usebox{1}\\
      \end{tabular}%
    }% sbox
    \ODw@gsetlength{\ODw@Diagram@Width}{\wd0}%
    \begin{tabular}[#3]{@{}r@{}c@{}}%
      \ODw@ProcessHeader{2}% span 2 columns
      \ODw@LeftUpperText & \ODw@Nhand\\
      \ODw@Whand         & \usebox{1}\\
      \ODw@ProcessFooter{2}% span 2 columns
    \end{tabular}%
   \IfBooleanTF#2{%
 % necessary for \ODw@CondNewLine
     \setlength{\ODw@Bid@Width}{\wd\ODw@BidBox}%
     \ODw@CondNewLine%
     \usebox{\ODw@BidBox}%
   }{}%
  \egroup%
  \IfBooleanTF#1{\end{center}}{}%
}% showNW
```

6.2.9 The Compass

When displaying the compass, the square with `N-S` and `E-W` axes, we try to achieve several things:

1. Making the size font-size dependent
2. Put both `N` and `S` horizontally centered
3. Put both `E` and `W` vertically centered
4. Print the vulnerable side in red if in colored mode
5. Underline the dealer (we *over*line `S` for better clarity)

We use the mapping as shown in the tables below. The `U` stands for undefined. This reflects the situation where neither `\vulner` nor `\dealer` have been called, and also there is no board number known.

Player	
N	0
E	1
S	2
W	3

Vulner	
none	0
all	1
N-S	2
E-W	3
U	-1

Dealer	
N	0
E	1
S	2
W	3
U	-1

`\ODw@Compass` `\ODw@Compass`

```
\newcommand{\ODw@Compass}{%
```

The codes for dealership (`\ODw@D`) and vulnerability (`\ODw@V`) are used in `\ODw@Print`. We initialize them with the value `-1` to denote the undefined state. `\@ODw` acts as a local temp variable in oder to make a smooth comparison.

```
 \begingroup
  \def\ODw@V{-1}\def\ODw@D{-1}%
%
% Set the code for vulnerability
%
  \def\@ODw{\none}\ifx\ODw@Vulner\@ODw\def\ODw@V{0}\fi%
  \def\@ODw{\none*}\ifx\ODw@Vulner\@ODw\def\ODw@V{0}\fi%
  \def\@ODw{\none!}\ifx\ODw@Vulner\@ODw\def\ODw@V{0}\fi%
  \def\@ODw{\none*!}\ifx\ODw@Vulner\@ODw\def\ODw@V{0}\fi%
  \def\@ODw{\all}\ifx\ODw@Vulner\@ODw\def\ODw@V{1}\fi%
  \def\@ODw{\all*}\ifx\ODw@Vulner\@ODw\def\ODw@V{1}\fi%
  \def\@ODw{\all!}\ifx\ODw@Vulner\@ODw\def\ODw@V{1}\fi%
```

```
  \def\@ODw{\all*!}\ifx\ODw@Vulner\@ODw\def\ODw@V{1}\fi%
  \def\@ODw{\NorthSouth}\ifx\ODw@Vulner\@ODw\def\ODw@V{2}\fi%
  \def\@ODw{\NorthSouth*}\ifx\ODw@Vulner\@ODw\def\ODw@V{2}\fi%
  \def\@ODw{\NorthSouth!}\ifx\ODw@Vulner\@ODw\def\ODw@V{2}\fi%
  \def\@ODw{\NorthSouth*!}\ifx\ODw@Vulner\@ODw\def\ODw@V{2}\fi%
  \def\@ODw{\EastWest}\ifx\ODw@Vulner\@ODw\def\ODw@V{3}\fi%
  \def\@ODw{\EastWest*}\ifx\ODw@Vulner\@ODw\def\ODw@V{3}\fi%
  \def\@ODw{\EastWest!}\ifx\ODw@Vulner\@ODw\def\ODw@V{3}\fi%
  \def\@ODw{\EastWest*!}\ifx\ODw@Vulner\@ODw\def\ODw@V{3}\fi%
%
% Set the code for dealership
%
  \def\@ODw{\North}\ifx\ODw@Dealer\@ODw\def\ODw@D{0}\fi%
  \def\@ODw{\North*}\ifx\ODw@Dealer\@ODw\def\ODw@D{0}\fi%
  \def\@ODw{\North!}\ifx\ODw@Dealer\@ODw\def\ODw@D{0}\fi%
  \def\@ODw{\North*!}\ifx\ODw@Dealer\@ODw\def\ODw@D{0}\fi%
  \def\@ODw{\East}\ifx\ODw@Dealer\@ODw\def\ODw@D{1}\fi%
  \def\@ODw{\East*}\ifx\ODw@Dealer\@ODw\def\ODw@D{1}\fi%
  \def\@ODw{\East!}\ifx\ODw@Dealer\@ODw\def\ODw@D{1}\fi%
  \def\@ODw{\East*!}\ifx\ODw@Dealer\@ODw\def\ODw@D{1}\fi%
  \def\@ODw{\South}\ifx\ODw@Dealer\@ODw\def\ODw@D{2}\fi%
  \def\@ODw{\South*}\ifx\ODw@Dealer\@ODw\def\ODw@D{2}\fi%
  \def\@ODw{\South!}\ifx\ODw@Dealer\@ODw\def\ODw@D{2}\fi%
  \def\@ODw{\South*!}\ifx\ODw@Dealer\@ODw\def\ODw@D{2}\fi%
  \def\@ODw{\West}\ifx\ODw@Dealer\@ODw\def\ODw@D{3}\fi%
  \def\@ODw{\West*}\ifx\ODw@Dealer\@ODw\def\ODw@D{3}\fi%
  \def\@ODw{\West!}\ifx\ODw@Dealer\@ODw\def\ODw@D{3}\fi%
  \def\@ODw{\West*!}\ifx\ODw@Dealer\@ODw\def\ODw@D{3}\fi%
%
```

We use a `picture` environment and set its size to `2.5em` × `2.5em` by setting the `\PicSize` to 500 and the unitlength to `0.005em`. Doing this enables us the avoid floating point arithmetic in the calculations of positions. Both `\PicSize` and `\MidSize` are local to `\ODw@Compass` and skipped from indexing. The same goes for `\Hoffset` and `\Voffset`.

```
  \ODw@CompassDefault% use the compass font
  \def\PicSize{500}%
  \def\MidSize{250}%
  % Multiply unitlength=0.005em with CompSize (default= 1)
  \setlength\unitlength{0.005em * \real{\ODw@CompSize}}%
  \ODw@gsetlength\ODw@Compasssize{\unitlength * \PicSize + 2ex}%
  \def\Hoffset{30}% distance between W (E) and frame
  \def\Voffset{30}% distance between N (S) and frame
  \setlength\ODw@Tmp@Width{0.1em * \real{\ODw@CompLine}}%
```

```
\linethickness{\ODw@Tmp@Width}%
% leave 1ex space on all sides
\parbox[c][\ODw@Compasssize]{\ODw@Compasssize}{%
  \centering%
  \begin{picture}(\PicSize,\PicSize)%
    \ifODw@CompShow%
      % the frame
      \moveto(0,0)
     \if\ODw@CompLine0% must do it this way, because
     \else% linethickness zero does not suppress the line
       \lineto(0,\PicSize)\lineto(\PicSize,\PicSize)
       \lineto(\PicSize,0)\closepath\strokepath
     \fi%
      % the cardinal points
     \put(\MidSize,\the\numexpr \PicSize - \Voffset)%
       {\makebox[0pt]{\raisebox{-\height}{\ODw@Print{0}}}}% N
     \put(\MidSize,\Voffset){\makebox[0pt]{\ODw@Print{2}}}% S
     \put(\Hoffset,\MidSize){%
       \makebox[0pt][l]{%
        \ifODw@CompTurn%
          \raisebox{-0.5\height}{%
            \rotatebox[origin=t]{90}{\ODw@Print{3}}%
          }%
        \else%
           \raisebox{-0.5\height}{\ODw@Print{3}}%
        \fi%
       }% makebox
      }% W
      \put(\the\numexpr \PicSize - \Hoffset,\MidSize)%
       {\makebox[0pt][r]{%
          \raisebox{-0.5\height}{%
          \ifODw@CompTurn%
              \rotatebox[origin=c]{90}{\ODw@Print{1}}%
            \else%
              \ODw@Print{1}%
            \fi%
          }%
         }% makebox
       }% E
      % the center
        \put(\MidSize,\MidSize){\makebox(0,0){\ODw@mid}}
    \fi%
  \end{picture}%
```

```
1074    }% parbox
1075  \endgroup
1076 }% ODw@Compass
```

\ODw@Print `\ODw@Print{⟨player-code⟩}`

`\ODw@Print` prints `N`, `E`, `S` and `W` in the compass. The side that is vulnerable is printed in red (or italics if we are monochrome), otherwise in black. The dealer is under- or overlined.
The `\ifcase` distinguishes between the players. Then dealership and vulnerability are tested. `\ODw@PrintColor` is called to actually print the player.

```
1077 \newcommand\ODw@Print[1]{%
\ODw@Print{player-code}
           1 (player-code=0-3)
1078 \bgroup
1079   \smaller\smaller%
1080   \ifcase#1% #1=0: print N
1081     \ifboolexpr{ test {\ifnumcomp{\ODw@D}{=}{0}}}%
1082     {% dealer = N
1083       \ifboolexpr{ test {\ifnumcomp{\ODw@V}{=}{1}} or%
1084                    test {\ifnumcomp{\ODw@V}{=}{2}} }%
1085         {\ODw@PrintColor{\underline{\ODw@N*}}}% Vul
1086         {\underline{\ODw@N*}}% not Vul
1087     }{% dealer <> N
1088       \ifboolexpr{ test {\ifnumcomp{\ODw@V}{=}{1}} or%
1089                    test {\ifnumcomp{\ODw@V}{=}{2}} }%
1090         {\ODw@PrintColor{\ODw@N*}}% Vul
1091         {\ODw@N*}% not Vul
1092     }%
1093   \or% #1=1: print E
1094     \ifboolexpr{ test {\ifnumcomp{\ODw@D}{=}{1}}}%
1095     {% dealer E
1096       \ifboolexpr{ test {\ifnumcomp{\ODw@V}{=}{1}} or%
1097                    test {\ifnumcomp{\ODw@V}{=}{3}} }%
1098         {\ODw@PrintColor{\underline{\ODw@E*}}}% Vul
1099         {\underline{\ODw@E*}}% not Vul
1100     }{% dealer <> E
1101       \ifboolexpr{ test {\ifnumcomp{\ODw@V}{=}{1}} or%
1102                    test {\ifnumcomp{\ODw@V}{=}{3}} }%
1103         {\ODw@PrintColor{\ODw@E*}}% Vul
1104         {\ODw@E*}% not Vul
1105     }%
```

```
  \or% #1=2: print S
    \ifboolexpr{ test {\ifnumcomp{\ODw@D}{=}{2}}}%
    {% dealer S
      \ifboolexpr{ test {\ifnumcomp{\ODw@V}{=}{1}} or%
                   test {\ifnumcomp{\ODw@V}{=}{2}} }%
        {\ODw@PrintColor{%
           \ensuremath{\overline{\mbox{\ODw@S*}}}}}%
        }% Vul
        {\ensuremath{\overline{\mbox{\ODw@S*}}}}% not Vul
    }{% dealer <> S
      \ifboolexpr{ test {\ifnumcomp{\ODw@V}{=}{1}} or%
                   test {\ifnumcomp{\ODw@V}{=}{2}} }%
        {\ODw@PrintColor{\ODw@S*}}% Vul
        {\ODw@S*}% not Vul
    }%
  \or% #1=3: print W
    \ifboolexpr{ test {\ifnumcomp{\ODw@D}{=}{3}}}%
    {% dealer W
      \ifboolexpr{ test {\ifnumcomp{\ODw@V}{=}{1}} or%
                   test {\ifnumcomp{\ODw@V}{=}{3}} }%
        {\ODw@PrintColor{\underline{\ODw@W*}}}% Vul
        {\underline{\ODw@W*}}% not Vul
    }{% dealer <> W
      \ifboolexpr{ test {\ifnumcomp{\ODw@V}{=}{1}} or%
                   test {\ifnumcomp{\ODw@V}{=}{3}} }%
        {\ODw@PrintColor{\ODw@W*}}% Vul
        {\ODw@W*}% not Vul
    }%
  \fi% (ifcase#1)
\egroup%
}% ODw@Print
```

\ODw@PrintColor `\ODw@PrintColor{|N|E|S|W}`

`\ODw@PrintColor` checks if we are in monochrome mode. If that is the case we print `N`, `E`, `S` or `W` in italics, otherwise in color.

```
\newcommand\ODw@PrintColor[1]{%
  \ifODw@monochrome\textit{#1}\else\textcolor{red}{#1}\fi%
}% ODw@PrintColor
```

\ODw@mid Hook to write something in the middle of the compass. We write what is stored in `\ODw@CompMid` a bit smaller than the `N-S` and `E-W` letters. `\ODw@mid` writes the contents of `\ODw@CompMid` in the middle of the compass. As there is only very limited room, this text should be

very short. It is primarily meant to write just a board number in the compass.

\ODw@CompMid

```
\def\ODw@CompMid{}
\def\ODw@mid{{\smaller\smaller\smaller\ODw@CompMid}}
```

\ODw@Box Displays a □

```
\newcommand{\ODw@Box}{%
  \bgroup
    \setlength{\fboxsep}{0pt}%
    \setlength{\fboxrule}{0.1em}%
    \fbox{\rule{0mm}{0.7em}\rule{0.7em}{0mm}}%
  \egroup
}% ODw@Box
```

6.2.10 Diagram Conditions

Here we implement several macros that add some board information to the card diagram. `\ODw@BoardText` serves as a variable to store the user-defined or (by means of the board number) auto-generated text, concerning the board. The macro `\boardnr` has 1 mandatory argument. If it is a number, it is considered to be the board number. The dealer and which side is vulnerable is then calculated from it and stored by calling `\dealer` resp. `\vulner`. If it is not a positive integer, it is considered user-defined text which is stored 'as is' in `\ODw@BoardText`. The contents can be retrieved by the user by calling `\boardtext` to actually print the board information.

\ODw@BoardText

```
\def\ODw@BoardText{}
```

\boardtext `\boardtext[*]`

`\boardtext` has only 1 token and no arguments. The unstarred form outputs only the text stored in `\ODw@BoardText`, this is normally a board number. `\boardtext*` outputs something like 'Board: 23'.

```
\NewDocumentCommand\boardtext{s}{%
  \IfBooleanTF#1{\GetTranslation{Board-(ODw)}:\,\ODw@BoardText}%
                {\ODw@BoardText}%
}% boardtext
```

\boardnr `\boardnr{⟨Nr⟩}`

`\boardnr{Nr}` sets the dealership and vulnerability according to `Nr`. As the system repeats itself after the 16th board, we canonize `Nr` to the range of 1–16. We also set `\ODw@BoardText` accordingly. For the association between boardnumber and dealer/vulner, see file *Compass* of the `onedown-example` collection.

```
\newcommand{\boardnr}[1]{%
  \IfInteger{#1}{%
    \gdef\ODw@BoardText{%
      \bgroup%
        \ODw@OtherFont%
%         \GetTranslation{Board-(ODw)} #1%
      #1%
      \egroup%
    }%
    \setcounter{ODw@Cnt}{#1}%
    \whiledo{\theODw@Cnt > 16}{%
      \addtocounter{ODw@Cnt}{-16}%
    }% whiledo, now 1 <= Cnt <= 16
    \IfEqCase{\theODw@Cnt}{% set dealer/vulner
% Board 0 = no board: mark dealer and vulner undefined
      {0}{\gdef\ODw@BoardText{}\vulner[-1]\dealer[-1]}
      {1}{\vulner[\none]\dealer[\North*!]}
      {2}{\vulner[\NorthSouth*!]\dealer[\East*!]}
      {3}{\vulner[\EastWest*!]\dealer[\South*!]}
      {4}{\vulner[\all]\dealer[\West*!]}
      {5}{\vulner[\NorthSouth*!]\dealer[\North*!]}
      {6}{\vulner[\EastWest*!]\dealer[\East*!]}
      {7}{\vulner[\all]\dealer[\South*!]}
      {8}{\vulner[\none]\dealer[\West*!]}
      {9}{\vulner[\EastWest*!]\dealer[\North*!]}
      {10}{\vulner[\all]\dealer[\East*!]}
      {11}{\vulner[\none]\dealer[\South*!]}
      {12}{\vulner[\NorthSouth*!]\dealer[\West*!]}
      {13}{\vulner[\all]\dealer[\North*!]}
      {14}{\vulner[\none]\dealer[\East*!]}
      {15}{\vulner[\NorthSouth*!]\dealer[\South*!]}
      {16}{\vulner[\EastWest*!]\dealer[\West*!]}
    }% IfEqCase
  }{\gdef\ODw@BoardText{#1}}% otherwise take #1
}% boardnr
```

The next macros are used to add some game information above resp. below the card diagram. `\ODw@HeaderText` and `\ODw@FooterText`

are used as variables to store the user-defined text.

\headlinetext `\headlinetext{⟨text⟩}`

\ODw@HeaderText

```
\newcommand\headlinetext[1]{\gdef\ODw@HeaderText{#1}}
\headlinetext{}
```

\footlinetext `\footlinetext{⟨text⟩}`

\ODw@FooterText

```
\newcommand\footlinetext[1]{\gdef\ODw@FooterText{#1}}
\footlinetext{}
```

The next macros are used to add some game information in the corners of the card diagram. We use a `tabular` with 1 column and 3 lines to do so.

We redefine the (originally empty) `Left[Upper/Lower]-` and `Right[Upper/Lower]Text`, and set it to the wanted value. The first (optional) parameter defines some horizontal extra space if the hand and a condition text collide. The commands have 3 mandatory arguments, each for 1 of the 3 condition lines, which may be empty.

\leftupper `\leftupper[⟨h-offset⟩]{⟨line1⟩}{⟨line2⟩}{⟨line3⟩}`

\ODw@LeftUpperText

```
\def\ODw@LeftUpperText{}
\newcommand\leftupper[4][0pt]{%
  \gdef\ODw@LeftUpperText{%
    \hspace{-#1}%
     \begin{tabular}[t]{l}#2\\#3\\#4\\\end{tabular}
  }%
}% leftupper
```

\leftlower `\leftlower[⟨h-offset⟩]{⟨line1⟩}{⟨line2⟩}{⟨line3⟩}`

\ODw@LeftLowerText

```
\def\ODw@LeftLowerText{}
\newcommand\leftlower[4][0pt]{%
  \gdef\ODw@LeftLowerText{%
    \hspace{-#1}%
     \begin{tabular}[b]{l}#2\\#3\\#4\\\end{tabular}
```

```
  }%
}% leftlower
```

\rightupper `\rightupper[⟨h-offset⟩]{⟨line1⟩}{⟨line2⟩}{⟨line3⟩}`

\ODw@RightUpperText

```
\def\ODw@RightUpperText{}
\newcommand\rightupper[4][0pt]{%
  \gdef\ODw@RightUpperText{%
    \hspace{#1}%
     \begin{tabular}[t]{l}#2\\#3\\#4\\\end{tabular}
  }%
}% rightupper
```

\rightlower `\rightlower[⟨h-offset⟩]{⟨line1⟩}{⟨line2⟩}{⟨line3⟩}`

\ODw@RightLowerText

```
\def\ODw@RightLowerText{}
\newcommand\rightlower[4][0pt]{%
  \gdef\ODw@RightLowerText{%
    \hspace{#1}%
     \begin{tabular}[b]{l}#2\\#3\\#4\\\end{tabular}
  }%
}% rightlower
```

\ODw@ProcessHeader `\ODw@ProcessHeader{⟨N⟩}`

`\ODw@ProcessHeader[N]` puts `HeaderText` in a multicolumn which spans N columns.

```
\newcommand{\ODw@ProcessHeader}[1]{%
```

Programmers note:
`\ODw@TestIfEmpty` cannot be called in here. The `\ifthenelse` called within the tabular environment leads to the error:

```
! Misplaced \omit. \multispan ->\omit \@multispan
```

So the test of the emptiness of `Header-` and `FooterText` is done out of the tabular. Why is (La)TeX always causing unexpected problems?:-(We set the headline/footline text to the width of the diagram with a solution found at:
https://tex.stackexchange.com/questions/125005/how-to-create-a-table-where-one-cell-spans-all-the-columns-and-the-text-wraps-pr

```
  \ifODw@EmptyHeader% Must be this way (StackExchange)
   \else%             |\ifthenelse| bites |\multicolumn|!
    \multicolumn{#1}{%
      p{\dimexpr\ODw@Diagram@Width-%
          2\tabcolsep-2\arrayrulewidth}%
    }{{\ODw@LegendFont\ODw@HeaderText}}\\
  \fi%
}% ODw@ProcessHeader
```

\ODw@TestIfEmpty `\ODw@TestIfEmpty{⟨Str⟩}{⟨Bool⟩}`

Tests the emptiness of a string.

```
\newcommand{\ODw@TestIfEmpty}[2]{%
```

```
\ODw@TestIfEmpty{Str}{Bool}
 sets boolean Bool to true if string Str is empty
```

```
  \ifthenelse{\equal{#1}{}}{%
    \setboolean{#2}{true}}{%
    \setboolean{#2}{false}%
  }%
}% ODw@TestIfEmpty
```

\ODw@ProcessFooter `\ODw@ProcessFooter{⟨N⟩}`

`\ODw@ProcessFooter[N]` puts `FooterText` in a multicolumn which spans `N` columns.

```
\newcommand{\ODw@ProcessFooter}[1]{%
  \ifODw@EmptyFooter% Must be this way (StackExchange)
   \else%             |\ifthenelse| bites |\multicolumn|!
    \multicolumn{#1}{%
      p{\dimexpr\ODw@Diagram@Width%
        -2\tabcolsep-2\arrayrulewidth}%
    }{{\ODw@LegendFont\ODw@FooterText}}\\
  \fi%
}% ODw@ProcessFooter
```

\handskip `\handskip{⟨length⟩}`

`\handskip` adds ⟨*length*⟩ to `SkipWidth`, i.e. the distance between the card diagram (with or without the east hand) and the `bidding` diagram.

```
\newcommand\handskip[1]{%
  \def\ODw@Skipwidth{1em + #1}% recalculate the new Skipwidth
  \setlength\ODw@Skip@Width{\ODw@Skipwidth}%
}% handskip
```

\ODw@DealerText Typesets the string *Dealer*.

```
\def\ODw@DealerText{%
  \bgroup%
    \ODw@OtherFont\GetTranslation{Dealer-(ODw)}%
  \egroup%
}% ODw@DealerText
```

\ODw@VulnerText

\ODw@VulnerText[*!] **vulnerable** (**Vulnerable**, **vul**, **Vul**)

Typesets the string *vulnerable* or *Vul*.

```
\NewDocumentCommand\ODw@VulnerText{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF#1{%
      \IfBooleanTF#2{\GetTranslation{Vul-(ODw)}}%
        {\GetTranslation{Vulnerable-(ODw)}}%
    }{%
      \IfBooleanTF#2{\GetTranslation{vul-(ODw)}}%
        {\GetTranslation{vulnerable-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% ODw@VulnerText
```

\ODw@AwardText Typesets the text to be put as 'title' in the *award* part of a expert quiz.

```
\def\ODw@AwardText{\textsf{\GetTranslation{Award-(ODw)}}}
```

\dealer

\dealer[⟨*dealer*⟩]

If `#1 = empty` then set `\ODw@Dealer` to `#1` else output `\ODw@Dealer`

```
\newcommand\dealer[1][]{%
  \ifthenelse{\equal{#1}{}}%
    {\ODw@Dealer}%
    {\gdef\ODw@Dealer{#1}}%
}% dealer
```

\ODw@Dealer set **North** as default

```
\def\ODw@Dealer{\North*}
```

\vulner

\vulner[⟨*vulner*⟩]

If `#1 = empty` then set `\ODw@Vulner` to `#1` else output `\ODw@Vulner`

```
\newcommand\vulner[1][]{%
  \ifthenelse{\equal{#1}{}}%
    {\ODw@Vulner}%
    {\gdef\ODw@Vulner{#1}}%
}% vulner
```

`\ODw@Vulner` set **north–south** as default

```
\def\ODw@Vulner{\NorthSouth}
```

`\dealertext`

```
\newcommand\dealertext[1][\ODw@Dealer]{\ODw@DealerText:\,#1}
```

`\vulnertext`

```
\newcommand\vulnertext[1][\ODw@Vulner]{%
  \ifODw@LongCalls%
    \ODw@VulnerText*%
  \else%
    \ODw@VulnerText*!%
  \fi%
  :\,#1%
}
```

`\alert`

```
\newcommand{\alert}{{}\ensuremath{^\textbf{*}}}
```

`\announce`

```
\newcommand{\announce}{{}\ensuremath{^\textbf{\smaller A}}}
```

`\markit` `\markit`

```
\newcommand\markit{%
```

Sets markers *a*, *b* etc. To be used only in `bidding` diagrams.

```
  \stepcounter{ODw@Nr}%
  \footnotemark[\theODw@Nr]%
}% markit
```

`\explainit` `\explainit{⟨text⟩}`

```
\newcommand\explainit[1]{%
```

Explains the marked items. To be used only in `bidding` diagrams. The counter `ODw@Nr` associates the marker with the explanation.

```
  \stepcounter{ODw@Nr}%
  \ensuremath{{}^\textrm{\smaller\alph{ODw@Nr}}}\,#1%
}% explainit
```

6.3 The Bidding Environments

6.3.1 Special Columntypes

\newcolumntype

In order to automatically apply a macro call on all cell contents of a column (translate/convert; step a counter) in `bidding` and `play` diagrams, we define newcolumntypes, made possible by loading package `collectcell`. We define next columntypes:

B: Transfers suits and cards (`bidding` and `play`)
F: sets First column in `play` diagrams
P: Transfers suits and cards, accumulates won tricks (`play`)

```
\newcolumntype{B}{% for Biddings
  >{\collectcell\ODw@BTfer}c<{\endcollectcell}}
\newcolumntype{F}{% for First column in |play| diagrams
  >{\collectcell\ODw@FTfer}c<{\endcollectcell}}
\newcolumntype{P}{% for |Play| diagrams (2nd--4th column)
  >{\collectcell\ODw@PTfer}c<{\endcollectcell}}
```

6.3.2 The Hidden Implementation

As explained before, in the `bidding` diagrams we convert `S` to the spade symbol ♠, etc. We do need packages `array` and `collectcell` for this and define the columntype `B`. In order to avoid these convertions in the top row, where names are displayed, we use the `\cci`-trick to suppress expansion of the cell macro. Curiously the command `\cci` (from `collcell`, v0.5, 2011/02/27) sometimes produces unwanted characters. The reason is unknown to me. We use a space (`" "`) as first character in `\cci` to avoid this.
We store the `bidding` diagram without the explanations in a box, so we can calculate the width of the `bidding` diagram and make our explanation part exactly as wide. In the first row we write the bidders: `North` etc. (or `N`, if it has to be short). In the second row we write the real world names of the bidders, if given. If any description is given, we make a multicolumn over the 4 rows with the previously stored width to write the explanations. We also use this width to calculate whether the bidding diagram will fit on the actual line. If not, we put it on a new line.

ODw@Bidding

`\ODw@Bidding[`⟨*pos*⟩`](`⟨*description*⟩`)...\endODw@Bidding`

```
\NewEnviron{ODw@Bidding}[2][t]{%
  \def\xspace{}%
```

```
  \setlength\tabcolsep{0.2em}%
  \sbox{0}{%
    \begin{tabular}[#1]{BBBB}% 1st column
     \ifODw@Bidders%
      \cci{ % there MUST be a ' ' (space)
        \ODw@BidderFont%
          \ifODw@short\ODw@BidderI%
            \else\ODw@BidderI*%
          \fi%
      } &%                      2nd column
      \cci{ % there MUST be a ' ' (space)
        \ODw@BidderFont%
          \ifODw@short\ODw@BidderII%
            \else\ODw@BidderII*%
          \fi%
      } &%                      3rd column
      \cci{ % there MUST be a ' ' (space)
        \ODw@BidderFont%
          \ifODw@short\ODw@BidderIII%
            \else\ODw@BidderIII*%
          \fi%
      } &%                      4th column
      \cci{ % there MUST be a ' ' (space)
        \ODw@BidderFont%
          \ifODw@short\ODw@BidderIV%
            \else\ODw@BidderIV*%
          \fi%
      } \\% end of 1st row
      \if\ODw@All@Names\empty%
      \else% 2nd row
        \cci{ \ODw@NameFont\ODw@NameI}   &%  please
        \cci{ \ODw@NameFont\ODw@NameII}  &%  mind
        \cci{ \ODw@NameFont\ODw@NameIII} &%  the
        \cci{ \ODw@NameFont\ODw@NameIV}  \\% spaces!
      \fi%
      \ifODw@BidLine\hline\fi%
     \fi%
     \BODY%
%%%%%%
    \end{tabular}%
  }% sbox0
  \setcounter{ODw@Nr}{0}%
  \setlength{\ODw@Bid@Width}{\wd0}%
```

```
1350  \global\sbox\ODw@BidBox{%
1351    \begin{tabular}[#1]{BBBB}%   1st column
1352     \ifODw@Bidders%
1353      \cci{ % there MUST be a ' ' (space)
1354        \ODw@BidderFont%
1355          \ifODw@short\ODw@BidderI%
1356            \else\ODw@BidderI*%
1357          \fi%
1358      } &%                        2nd column
1359      \cci{ % there MUST be a ' ' (space)
1360        \ODw@BidderFont%
1361          \ifODw@short\ODw@BidderII%
1362            \else\ODw@BidderII*%
1363          \fi%
1364      } &%                        3rd column
1365      \cci{ % there MUST be a ' ' (space)
1366        \ODw@BidderFont%
1367          \ifODw@short\ODw@BidderIII%
1368            \else\ODw@BidderIII*%
1369          \fi%
1370      } &%                        4th column
1371      \cci{ % there MUST be a ' ' (space)
1372        \ODw@BidderFont%
1373          \ifODw@short\ODw@BidderIV%
1374            \else\ODw@BidderIV*%
1375          \fi%
1376      } \\% end of 1st row
1377      \if\ODw@All@Names\empty%
1378      \else% 2nd row
1379        \cci{ \ODw@NameFont\ODw@NameI}   &%  please
1380        \cci{ \ODw@NameFont\ODw@NameII}  &%  mind
1381        \cci{ \ODw@NameFont\ODw@NameIII} &%  the
1382        \cci{ \ODw@NameFont\ODw@NameIV}  \\% spaces!
1383      \fi%
1384      \ifODw@BidLine\hline\fi%
1385     \fi
1386     \BODY%
1387 %%%%%  Until here the same code as in the sbox!
1388      \ifODw@description%
1389      % Add the description, if not empty
1390        \hline%
1391        \multicolumn{4}{% span explanations over 4 cols...
1392          p{\dimexpr\ODw@Bid@Width-2\tabcolsep}%
```

```
        }{% ...with the right width
          \setcounter{ODw@Nr}{0}%
          \raggedright%
          \smaller\smaller#2%
        }\\%
      \fi%
    \end{tabular}%
  }% sbox ODw@BidBox
}% ODw@Bidding
```

ODw@Biddingpair `\ODw@Biddingpair[⟨pos⟩](⟨description⟩)...\endODw@Biddingpair`

Decription: Similar to `ODw@Bidding`

```
\NewEnviron{ODw@Biddingpair}[2][t]{%
  \def\xspace{}%
  \setlength\tabcolsep{0.2em}%
  \sbox{0}{%
   \begin{tabular}[#1]{BB}% 1st column
    \ifODw@Bidders%
     \cci{ % there MUST be a ' ' (space)
      \ODw@BidderFont%
       \ifODw@short\ODw@BidderI%
         \else\ODw@BidderI*%
       \fi%
     } &% 2nd column
     \cci{ % there MUST be a ' ' (space)
      \ODw@BidderFont%
       \ifODw@short\ODw@BidderIII%
        \else\ODw@BidderIII*%
       \fi%
      } \\% end of 1st row
      \if\ODw@All@Names\empty%
       \else% 2nd row
        \cci{ \ODw@NameFont\ODw@NameI} &%  please mind
        \cci{ \ODw@NameFont\ODw@NameIII} \\% the spaces!
      \fi%
      \ifODw@BidLine\hline\fi%
     \fi%
     \BODY%
%%%%%%%
    \end{tabular}%
  }% sbox0
  \setcounter{ODw@Nr}{0}%
  \setlength{\ODw@Bid@Width}{\wd0}%
```

```
  \global\sbox\ODw@BidBox{%
    \begin{tabular}[#1]{BB}% 1st column
     \ifODw@Bidders%
      \cci{ % there MUST be a ' ' (space)
       \ODw@BidderFont%
        \ifODw@short\ODw@BidderI%
         \else\ODw@BidderI*%
        \fi%
      } &% 2nd column
      \cci{ % there MUST be a ' ' (space)
        \ODw@BidderFont%
          \ifODw@short\ODw@BidderIII%
            \else\ODw@BidderIII*%
          \fi%
      } \\% end of 1st row
      \if\ODw@All@Names\empty%
      \else% 2nd row
       \cci{ \ODw@NameFont\ODw@NameI}   &%  please mind
       \cci{ \ODw@NameFont\ODw@NameIII} \\% the spaces!
      \fi%
      \ifODw@BidLine\hline\fi%
     \fi%
     \BODY%
  %%%%%% Until here the same code as in the sbox!
      \ifODw@description%
      % Add the description, if not empty
        \hline%
        \multicolumn{2}{%
          p{\dimexpr\ODw@Bid@Width-2\tabcolsep}%
        }{%
          \setcounter{ODw@Nr}{0}%
          \raggedright%
          \smaller\smaller#2%
        }\\%
      \fi%
    \end{tabular}%
  }% sbox ODw@BidBox
}% ODw@Biddingpair
```

6.4 The User Environments

6.4.1 Bidding

The `bidding` environments have 2 optional arguments: an alignment [⟨*pos*⟩] and an annotation (⟨*description*⟩). There are also 3 tokens: the `*` centers the `bidding` diagram, the `+` forces the short notation, i.e. **N** rather than **North** and the `-` suppresses all output.
`\ODw@GameSize` takes care of the font dependent sizing of the diagram. We locally redefine `\thefootnote` and reset the (general) counter `ODw@Nr`, which is stepped in `\markit` and `\explainit` to make the annotations correspond. In the end code we define a multicolumn over all 4 (2) columns and write the annotation given in argument `#2`. With `p{...\ODw@Bid@Width...}` care is taken to limit this text to the width of the diagram.

bidding `\begin{bidding}[*!-][⟨pos⟩](⟨description⟩)...\end{bidding}`

```
1471 \NewDocumentEnvironment{bidding}{s t! t- O{c}d()}{%
```

```
\begin{bidding}* ! -[pos](description)
              1 2 3  4        5
```

The 1st token (`*`) centers the environment; the 2nd token (`!`) switches to the short notation in the table header; the 3rd token (`-`) suppresses the output. Argument 4 regulates the alignment of the table (default is **c** and the 5th argument contains the annotations of the bidding.

```
1472   \ODw@GameSize%
1473   \renewcommand{\thefootnote}{\alph{footnote}}%
1474   \setcounter{ODw@Nr}{0}%
1475   \IfBooleanTF#1{\center}{}%           "*" detected
1476   \IfBooleanTF{#2}{\ODw@shorttrue}{}%  "!" detected
1477   \IfValueTF{#5}{\ODw@descriptiontrue}{\ODw@descriptionfalse}%
1478   \ODw@Bidding[#4]{#5}%
1479 }{%
1480   \endODw@Bidding%
1481   \IfBooleanTF{#3}%
1482     {\rule{0pt}{0pt}}%
1483   %           +---without this, pdflatex aborts compilation!
1484     {\usebox{\ODw@BidBox}}%  "-" detected
1485   \IfBooleanTF#1{\endcenter}{}%
1486 }% bidding
```

biddingpair `\begin{biddingpair}[*!-][⟨pos⟩](⟨description⟩)...\end{biddingpair}`

```
\NewDocumentEnvironment{biddingpair}{s t! t- O{c}d()}{%
\begin{biddingpair}* ! -[pos](description)
                  1 2 3  4        5
```

The same as with environment `bidding`, only with 2 columns instead of 4.

```
  \def\xspace{}%
  \ODw@GameSize%
  \renewcommand{\thefootnote}{\alph{footnote}}%
  \setcounter{ODw@Nr}{0}%
  \IfBooleanTF#1{\center}{}%           "*" detected
  \IfBooleanTF{#2}{\ODw@shorttrue}{}%  "+" detected
  \IfValueTF{#5}{\ODw@descriptiontrue}{\ODw@descriptionfalse}%
  \ODw@Biddingpair[#4]{#5}%
}{%
  \endODw@Biddingpair%
  \IfBooleanTF{#3}%
    {\rule{0pt}{0pt}}%
  %           +---without this, pdflatex aborts compilation!
    {\usebox{\ODw@BidBox}}%  "-" detected
  \IfBooleanTF#1{\endcenter}{}%
}% biddingpair
```

6.4.2 Play

Environment `play` displays the sequence of playing tricks. It uses 2 `newcolumntype`s:

- `F` to increment and display the current row.
- `P` to translate an convert suits/ranks.

`\ODw@AccTricks` `\ODw@AccTricks` calculates and shows the accumulated tricks in `play` that `N-S` and `E-W` has won. The winning card is detected automatically and `\ODw@LastTrick` is called to process the winning trick for whichever side won it (`N-S` or `E-W`) by stepping the counter for the winning side. The counter `ODw@Player` denotes the player who won the trick. From the player who leads and the position that wins we calculate the winning player and step the counter for his side.

```
\def\ODw@AccTricks{%
    \ODw@LastTrick{\ODw@Last}{\theODw@WinningNr}%
    \ODw@append{\theODw@NSCnt,\theODw@EWCnt,}% store counters
    \setcounter{ODw@Highest}{0}% reset for next trick
}% ODw@AccTricks
```

\ODw@Tricks `\ODw@Tricks`

\ODw@Last This macro is automatically called in TableII for column 1. This column displays the player who had the lead. It essentially 1) resets the `ODw@PlayerNr` which will be stepped for each next column in search for the winning card 2) remembers in `\ODw@Last` who had the lead. From these two values we can later calculate who won this trick. The stepping of `ODw@PlayerNr` occurs in `\ODw@Tfer`.

```
\def\ODw@Tricks{%
  \setcounter{ODw@PlayerNr}{0}%
  \gdef\ODw@Last{\ODw@NextLead}%
    \expandafter\GetTranslation%
    \expandafter{\ODw@NextLead-(ODw)}:\,%
% write a colon and a thin space in the table,
% as separator between lead player and lead card.
}% ODw@Tricks
```

\ODw@LastTrick `\ODw@LastTrick{⟨Player⟩}{⟨Pos⟩}`

This macro is called by `\ODw@AccTricks`. It computes who won the last trick and steps the corresponding counter.

```
\newcommand\ODw@LastTrick[2]{%
```

```
\ODw@LastTrick{Player}{Pos}
                1      2--- Seat Nr that won the trick
                +---------- Player (N,E,S,W) who has led
```

Consider the following table, where the seats are in horizontal direction, starting with the player who leads in seat 1. Vertically, in the first column, we have an initial value, stored in `ODw@Cnt` which is associated with the leading player.

	1	2	3	4
0	W^1	N^2	E^3	S^4
1	N^2	E^3	S^4	W^5
2	E^3	S^4	W^5	N^6
3	S^4	W^5	N^6	E^7

If we add this initial value to the *seat* where the trick is won, then the result gives us the *player* who won the trick. Suppose that e.g. `S` had the lead, so `ODw@Cnt` = 3. Suppose also that seat number 3 wins the trick. The sum equals 6 and this is the seat of `N`. For clarity these sums are displayed in the table as superscripts to the players

First we set the counter `ODw@Cnt` to the player who has the lead and

add the seat number (#2) to it. We then store who has the next lead in \ODw@NextLead and increment the counter of the winning side.

```
  \IfEqCase{#1}{%
    {W}{\setcounter{ODw@Cnt}{0}}%
    {N}{\setcounter{ODw@Cnt}{1}}%
    {E}{\setcounter{ODw@Cnt}{2}}%
    {S}{\setcounter{ODw@Cnt}{3}}%
  }% IfEqCase
  \addtocounter{ODw@Cnt}{#2}%
  \IfEqCase{\theODw@Cnt}{%
    {1}{\gdef\ODw@NextLead{W}\stepcounter{ODw@EWCnt}}
    {2}{\gdef\ODw@NextLead{N}\stepcounter{ODw@NSCnt}}
    {3}{\gdef\ODw@NextLead{E}\stepcounter{ODw@EWCnt}}
    {4}{\gdef\ODw@NextLead{S}\stepcounter{ODw@NSCnt}}
    {5}{\gdef\ODw@NextLead{W}\stepcounter{ODw@EWCnt}}
    {6}{\gdef\ODw@NextLead{N}\stepcounter{ODw@NSCnt}}
    {7}{\gdef\ODw@NextLead{E}\stepcounter{ODw@EWCnt}}
  }% IfEqCase
}% ODw@LastTrick
```

play `\begin{play}*{⟨Lead⟩}[⟨Trump⟩]...\end{play}`

Finally we define environment play. It consists primarily of these 3 tables, the middle one with the special newcolumntype **P**. We reset the counter for the running line (=trick) ODw@Nr and the winning trick counters for N–S and E–W: ODw@NSCnt and ODw@EWCnt. As usual \ODw@GameSize takes care for the correct sizing. The first row (the title row) is displayed using the \cci method. The 3th table gets a stacked N/S and E/W title.

```
\NewDocumentEnvironment{play}{s mO{N}}{%
% #1 --> s center
% #2 --> m lead
% #3 --> O trumpsuit (default NoTrump)
  \def\ODw@TrumpSuit{#3}
  \gdef\ODw@NextLead{#2}%
  \setcounter{ODw@Nr}{0}%
  \setcounter{ODw@NSCnt}{0}
  \setcounter{ODw@EWCnt}{0}%
  \ODw@GameSize%
  \ODw@Scratch{}% make empty
  \let\ODw@Clubs\empty%
  \let\ODw@Diamonds\empty%
```

```
  \let\ODw@Hearts\empty%
  \let\ODw@Spades\empty%
%
% We need some data which is calculated in TableII
% to create TableI and TableIII. So we put TableII in a
% box and display it later at due time
%
\def\ODw@EXtra{0.9em}% white space in title
%
\provideenvironment{TableII}{%
    \begin{tabular}[b]{FPPP}%
      \multicolumn{1}{c}{\GetTranslation{Lead!-(ODw)}} &%
      \cci{\GetTranslation{2nd-(ODw)}}                 &%
      \cci{\GetTranslation{3rd-(ODw)}}                 &%
      \cci{\GetTranslation{4th-(ODw)}}   \\[0.3em]\hline%
      \multicolumn{4}{c}{}\\[-\ODw@EXtra]%
}{%
  \end{tabular}%
}% TableII
%
  \begin{lrbox}{0}% save TableII for later
    \begin{TableII}
}{%
    \end{TableII}
  \end{lrbox}%
  % Check consistency of the played cards
  \ODw@ChkSameCards{\ODw@Spades}{\Sp}%
  \ODw@ChkSameCards{\ODw@Hearts}{\He}%
  \ODw@ChkSameCards{\ODw@Diamonds}{\Di}%
  \ODw@ChkSameCards{\ODw@Clubs}{\Cl}%
%
  \setcounter{ODw@Cnt}{0}%
  \IfBooleanTF#1{\begin{center}}{}%
%
```

We need TableI to show the running trick number. When constructing Table II, the total number of tricks that were actually displayed is available in counter `ODw@Nr`. So we just need to loop `\theODw@Nr` times and write the local counter value `\theODw@Cnt`. To avoid *the extra line problem* we use the solution from:
https://tex.stackexchange.com/questions/50296/problem-with-using-loop-inside-the-tabular-environment/142562#142562

```
%
```

```
  \begin{tabular}[b]{r}% TableI
    \cci{\GetTranslation{Nr-(ODw)}}\\[0.3em]
    \hline\\[-\ODw@EXtra]%
    \setcounter{ODw@Cnt}{1}%
    \whiledo{\theODw@Cnt<\theODw@Nr}{%
      \theODw@Cnt\\
      \stepcounter{ODw@Cnt}%
    }%
    \theODw@Cnt\\% MUST be outside the loop
  %  (the extra line problem]!
  \end{tabular}%
%
% TableII showing the cards played in the tricks
%
  \usebox{0}%
%
```

We use TableIII to show the winning trick counts. These are already stored in a CSV-list `\ODw@Scratch`, implemented as a token register. To process this list we use `\docsvlist` and must only define our `\do`. As this table has 2 columns, we check with `ODw@Nr` that after an item is read, we put an & and after the next item a `\\`.

```
%
  \setcounter{ODw@Nr}{0}%
  \renewcommand*{\do}[1]{%
    \ifnumequal{\value{ODw@Nr}}{2}{\\\setcounter{ODw@Nr}{0}}{}%
    \stepcounter{ODw@Nr}%
    ##1
    \ifnumequal{\value{ODw@Nr}}{2}{}{&}%
  }%
  \begin{tabular}[b]{|cc}% TableIII
    \multicolumn{1}{|c}{%
      \cci{\scriptsize\shortstack[c]{\North*!\\\South*!}}} &%
    \multicolumn{1}{c}{%
      \cci{\scriptsize\shortstack[c]{\East*!\\\West*!}}%
    }\\\hline\\[-\ODw@EXtra]%
    \expandafter\docsvlist\expandafter{\the\ODw@Scratch}%
  \end{tabular}%
  \IfBooleanTF#1{\end{center}}{}%
%
}% play
```

6.5 Card Diagrams with Bidding

\ODw@CondNewLine `\ODw@CondNewLine[⟨offset⟩]`

```
\NewDocumentCommand\ODw@CondNewLine{O{0em}}{%
```

`\ODw@CondNewLine` forces a newline if the `bidding` diagram does not fit on the line, taking into account the width of the card diagram and the width of the `bidding` diagram. Otherwise the `bidding` diagram appears to the right of the card diagram at distance `\ODw@Skip@Width`. We call the global macro `\ODw@Diagram@Width` that contains the width of the card diagram. The optional parameter of `\ODw@CondNewLine` is used to add some extra offset if needed.

```
 {\ODw@GameFont% needed to relate skips to the font-size
%  :\the\ODw@Skip@Width:% JW XXX
  \setlength{\ODw@Tmp@Len}{\ODw@Bid@Width}%
  \addtolength{\ODw@Tmp@Len}{\ODw@Diagram@Width}%
  \addtolength{\ODw@Tmp@Len}{#1}%
  \addtolength{\ODw@Tmp@Len}{\ODw@Skip@Width}%
  \ifthenelse{\lengthtest{\ODw@Tmp@Len > \textwidth}}{%
    \\[1em]}{%
      \hspace{\ODw@Skip@Width}%
    }%
 }%
}% ODw@CondNewLine
```

6.6 The Expert Quiz

\expertquiz `\expertquiz[*!][⟨comment⟩]{⟨award⟩}`

The macro `\expertquiz` displays a hand, a `bidding` diagram and the award for the answers. Optionally a description can be added. The hand and the bidding have to be defined before. This is done to avoid having 4 more arguments, needed for specifying the hand. The token '`*`' centers the whole and the token '`!`' forces that the `bidding` diagram appears on a new line and that the hand shifts a bit to the right. The last parameter defines the award. In order to limit the width of the award we use the known widths of the `bidding` diagram and the hand and set the parbox accordingly to display the award.

```
\NewDocumentCommand\expertquiz{st! O{}m}{%
%                                12  3  4
```

```
\expertquiz* ![comment]{award}
          1 2    3       4
```

```
  \noindent%
  \IfBooleanTF#1{\begin{center}}%      "*" detected
    {\par\vspace{0.5\baselineskip}}%
  \bgroup% keep font changes local (e.g. "\smaller").
    \ODw@LegendFont%
    \ifx#3\empty\else#3\par\fi%
  \egroup%
  \IfBooleanTF{#2}{~\hspace*{2em}}{}% "!" detected
  \usebox{\ODw@Hand@Box}%   display the saved hand
  \IfBooleanTF{#2}{}{\quad}%       no "!" detected
  \setlength\ODw@Tmp@Width{\wd\ODw@BidBox + 1em}%
  \IfBooleanTF{#2}%
    {\\}%
    {\addtolength\ODw@Tmp@Width{\wd\ODw@Hand@Box}}%
  \usebox{\ODw@BidBox}%     display the saved bidding
  \par\vspace{0.3em}%\noindent%
   {% keep legendfont and "smaller" local
    \ODw@LegendFont%
    \smaller%
    \IfBooleanTF#1{\bgroup\centering}{}%
    \parbox[t]{\ODw@Tmp@Width}{%
      \textbf{\ODw@AwardText: }%
      \raggedright#4%
    }% parbox
    \IfBooleanTF#1{\egroup}{}%
   }%
%  \fi%
    \IfBooleanTF#1{\end{center}}{}%
}% expertquiz
```

6.7 Resetting the Game

We use `pgfkeys` with its ⟨*key*⟩=⟨*val*⟩ system to specify the fonts and other things that we want to have as defaults, rather than the intitial OneDown values. Therefore we first define the keys and the store for it.

```
\pgfkeys{%
  /ODw/.is family, /ODw,
% fonts
```

```
  bidder/.store in = \ODw@BidderDefault,
  compass/.store in = \ODw@CompassDefault,
  game/.store in = \ODw@GameDefault,
  legend/.store in = \ODw@LegendDefault,
  name/.store in = \ODw@NameDefault,
  other/.store in = \ODw@OtherDefault,
% compass
  compline/.store in = \ODw@CompLine,
  compmid/.store in = \ODw@CompMid,
  compsize/.store in = \ODw@CompSize,
}
% compass
  \ODw@set{compshow/.is choice}
  \ODw@set{compshow/off/.code={\global\ODw@CompShowfalse}}
  \ODw@set{compshow/on/.code={\global\ODw@CompShowtrue}}
  \ODw@set{compturn/.is choice}
  \ODw@set{compturn/off/.code={\global\ODw@CompTurnfalse}}
  \ODw@set{compturn/on/.code={\global\ODw@CompTurntrue}}
% bídding
  \ODw@set{bidders/.is choice}
  \ODw@set{bidders/off/.code={\global\ODw@Biddersfalse}}
  \ODw@set{bidders/on/.code={\global\ODw@Bidderstrue}}
  \ODw@set{bidfirst/.is choice}
  \ODw@set{bidfirst/N/.code=\ODw@FirstBidCol{N}}
  \ODw@set{bidfirst/E/.code=\ODw@FirstBidCol{E}}
  \ODw@set{bidfirst/S/.code=\ODw@FirstBidCol{S}}
  \ODw@set{bidfirst/W/.code=\ODw@FirstBidCol{W}}
  \ODw@set{bidline/.is choice}
  \ODw@set{bidline/off/.code={\global\ODw@BidLinefalse}}
  \ODw@set{bidline/on/.code={\global\ODw@BidLinetrue}}
  \ODw@set{bidlong/.is choice}
  \ODw@set{bidlong/off/.code={\global\ODw@LongCallsfalse}}
  \ODw@set{bidlong/on/.code={\global\ODw@LongCallstrue}}
% synonyms
  \ODw@set{compshow/1/.code={\pgfkeys{/ODw/compshow=on}}}
  \ODw@set{compshow/true/.code={\pgfkeys{/ODw/compshow=on}}}
  \ODw@set{compturn/1/.code={\pgfkeys{/ODw/compturn=on}}}
  \ODw@set{compturn/true/.code={\pgfkeys{/ODw/compturn=on}}}
  \ODw@set{bidline/1/.code={\pgfkeys{/ODw/bidline=on}}}
  \ODw@set{bidders/true/.code={\pgfkeys{/ODw/bidders=on}}}
  \ODw@set{bidders/1/.code={\pgfkeys{/ODw/bidders=on}}}
  \ODw@set{bidline/true/.code={\pgfkeys{/ODw/bidline=on}}}
  \ODw@set{bidlong/1/.code={\pgfkeys{/ODw/bidlong=on}}}
```

```
  \ODw@set{bidlong/true/.code={\pgfkeys{/ODw/bidlong=on}}}
%
  \ODw@set{compshow/0/.code={\pgfkeys{/ODw/compshow=off}}}
  \ODw@set{compshow/false/.code={\pgfkeys{/ODw/compshow=off}}}
  \ODw@set{compturn/0/.code={\pgfkeys{/ODw/compturn=off}}}
  \ODw@set{compturn/false/.code={\pgfkeys{/ODw/compturn=off}}}
  \ODw@set{bidders/0/.code={\pgfkeys{/ODw/bidders=off}}}
  \ODw@set{bidders/false/.code={\pgfkeys{/ODw/bidders=off}}}
  \ODw@set{bidline/0/.code={\pgfkeys{/ODw/bidline=off}}}
  \ODw@set{bidline/false/.code={\pgfkeys{/ODw/bidline=off}}}
  \ODw@set{bidlong/0/.code={\pgfkeys{/ODw/bidlong=off}}}
  \ODw@set{bidlong/false/.code={\pgfkeys{/ODw/bidlong=off}}}
```

\resetfonts

```
\resetfonts
```

```
\newcommand\resetfonts{%
  \bidderfont{\ODw@BidderDefault}%
  \compassfont{\ODw@CompassDefault}%
  \gamefont{\ODw@GameDefault}%
  \legendfont{\ODw@LegendDefault}%
  \namefont{\ODw@NameDefault}%
  \otherfont{\ODw@OtherDefault}%
}% resetfonts
```

\setdefaults

```
\setdefaults*[⟨key1=val1⟩,⟨key2=val2⟩...]
```

The available keys are those defined in `\pgfkeys` some lines up from here. For the fonts they are: `bidder`, `compass`, `game`, `legend`, `name` and `other`. They store the new default value in the corresponding variable. In order to make the new default active, we must use `\setdefaults*` which will also call `\resetfonts`.

The keys for the compass are: `compline`, `compmid`, `compshow`, `compsize` and `compturn`. They control the thickness of the frame, the mid-text, the visibility of the compass, its size and the angle of the compass E–W letters.

For the `bidding` diagram we have: `bidders`, `bidfirst`, `bidline` and `bidlong`. They control if bidders are to be displayed at all, which bidder appears in the first column, draw a `\hline` below the header and showing long calls.

```
\NewDocumentCommand\setdefaults{s m}{%
  \pgfkeys{/ODw,#2}%
  \IfBooleanTF{#1}{\resetfonts}{}%
}% setdefaults
```

```
%
```

\newgame

\newgame resets and clears the stored game information to be ready for a new game. We do not reset the option for warn- and err-messages, nor any selected font. Setting \boardnr{0} executes also: \ODw@BoardText{}\vulner[-1]\dealer[-1]}.

```
\newcommand\newgame{%
  \boardnr{0}%
  \headlinetext{}%
  \footlinetext{}%
% clear the left/right upper/lower stuff
  \gdef\ODw@LeftUpperText{}%
  \gdef\ODw@LeftLowerText{}%
  \gdef\ODw@RightUpperText{}%
  \gdef\ODw@RightLowerText{}%
% clear the hands
  \gdef\ODw@Nhand{\ODw@hand{t}{}{}{}{}}%
  \gdef\ODw@Ehand{\ODw@hand{c}{}{}{}{}}%
  \gdef\ODw@Shand{\ODw@hand{b}{}{}{}{}}%
  \gdef\ODw@Whand{\ODw@hand{c}{}{}{}{}}%
%
% set default for real bidders names: no names
% we print only the symbolic names North, East, etc.
%
   \namesNS{}{}\namesEW{}{}%
%
% reset consistency check stuff
%
  \gdef\ODw@Spades{}%
  \gdef\ODw@Hearts{}%
  \gdef\ODw@Diamonds{}%
  \gdef\ODw@Clubs{}%
%
  \gdef\ODw@NSpades{}\gdef\ODw@ESpades{}%
  \gdef\ODw@SSpades{}\gdef\ODw@WSpades{}%
  \gdef\ODw@NHearts{}\gdef\ODw@EHearts{}%
  \gdef\ODw@SHearts{}\gdef\ODw@WHearts{}%
  \gdef\ODw@NDiamonds{}\gdef\ODw@EDiamonds{}%
  \gdef\ODw@SDiamonds{}\gdef\ODw@WDiamonds{}%
  \gdef\ODw@NClubs{}\gdef\ODw@EClubs{}%
  \gdef\ODw@SClubs{}\gdef\ODw@WClubs{}%
```

```
%
}% newgame
```

6.8 Error Handling

6.8.1 Consistency Checks

We perform different checks on consistency of the cards entered:

1. Check that a hand not has more than 13 cards (E)
2. Check that a hand doesn't contain multiple cards (E)
3. Check that a deal doesn't contain multiple cards (E)
4. Check that a hand has less than 13 cards (W)
5. check that a suit of a deal has more than 13 cards (E)
6. check that a suit of a deal has less than 13 cards (W)
7. In `play` diagrams: check that a card is played only once (E)

The checks marked with (E) raise an error, those marked with (W) raise a warning. They can be controlled with the package options `err` and `warn`.

\ODw@ChkNrOfCards `\ODw@ChkNrOfCards{⟨cards⟩}{⟨hand⟩}`

```
\newcommand\ODw@ChkNrOfCards[2]{%
\ODw@ChkNrOfCards{cards}{hand}
                    1      2
#1 = a string with all cards of all suits of the hand denoted by #2
step 1: remove all "-" (that denotes an empty suit)
step 2: warn if StrLen < 13 ; Err if StrLen > 13
  \StrDel{#1}{-}[\ODw@CardStr]%     remove voids
  \StrLen{\ODw@CardStr}[\ODw@CardLen]%
  \ifthenelse{\ODw@CardLen > 13}{%
    \ODw@Error{#2 has \ODw@CardLen{} cards}%
  }{%
    \ifthenelse{\ODw@CardLen < 13}{%
      \ODw@Warning{#2 has \ODw@CardLen{} cards}%
    }{}%
  }%
}% ODw@ChkNrOfCards
```

\ODw@PrErr `\ODw@PrErr{⟨rank⟩}{⟨count⟩}{⟨suit⟩}`

```
\newcommand\ODw@PrErr[3]{%
```

`\ODw@PrErr{rank}{count}{suit}`

1 2 3

This macro only outputs the warning/error if the card specified by rank (#1) and suit (#3) does not occur (denoted by #2) exactly 1×. An exception for spotcards must not be made, because in `\ODw@ChkSameCards` they are not taken into account. (In fact they are already filtered out by `ODw@translate`).

```
\bgroup%
  \if#1T\def\ODw@T{10}\else\def\ODw@T{#1}\fi%
  \ifthenelse{#2 > 1}{%
    \ODw@Error{Card #3\,\ODw@T{} occurs #2 times}%
  }{%
    \ifthenelse{#2 = 0}{%
      \ODw@Warning{Card #3\,\ODw@T{} fails}}{}%
  }%
\egroup%
}% ODw@PrErr
```

`\ODw@ChkSameCards` `\ODw@ChkSameCards{⟨cards⟩}{⟨suit⟩}`

```
\newcommand\ODw@ChkSameCards[2]{%
#1 = a string with all cards of 1 suit (denoted by #2) of all hands
step 1: remove all "-"
step 2: we count the frequency of all ranks 2--9,T,J,Q,K,A (ODw@CCnt)
step 3: Warn if Freq(card) = 0 ; Err if Freq(card) > 1 (ODw@PrErr)

  \StrDel{#1}{-}[\ODw@CardStr]%
  \StrCount{\ODw@CardStr}{2}[\ODw@CCnt]\ODw@PrErr{2}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{3}[\ODw@CCnt]\ODw@PrErr{3}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{4}[\ODw@CCnt]\ODw@PrErr{4}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{5}[\ODw@CCnt]\ODw@PrErr{5}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{6}[\ODw@CCnt]\ODw@PrErr{6}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{7}[\ODw@CCnt]\ODw@PrErr{7}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{8}[\ODw@CCnt]\ODw@PrErr{8}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{9}[\ODw@CCnt]\ODw@PrErr{9}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{T}[\ODw@CCnt]\ODw@PrErr{T}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{J}[\ODw@CCnt]\ODw@PrErr{J}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{Q}[\ODw@CCnt]\ODw@PrErr{Q}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{K}[\ODw@CCnt]\ODw@PrErr{K}{\ODw@CCnt}{#2}%
  \StrCount{\ODw@CardStr}{A}[\ODw@CCnt]\ODw@PrErr{A}{\ODw@CCnt}{#2}%
}% ODw@ChkSameCards
```

6.8.2 Controlling Messages

```
\newbool{ODw@Warnings}
\newbool{ODw@Errors}
%
\ODw@set{warn/off/.code={%
  \global\setbool{ODw@Warnings}{false}}}
\ODw@set{warn/on/.code={%
  \global\setbool{ODw@Warnings}{true}}}
\ODw@set{err/off/.code={%
  \global\setbool{ODw@Errors}{false}}}
\ODw@set{err/on/.code={%
  \global\setbool{ODw@Errors}{true}}}
\ODw@set{warn=off}
\ODw@set{err=on}

\ProcessPgfOptions{/ODw}
```

\ODw@Error

```
\newcommand\ODw@Error[1]{%
  \ifbool{ODw@Errors}{%
    \par\textcolor{red}{Error: #1}\par}{}%
}% ODw@Error
```

\ODw@Warning

```
\newcommand\ODw@Warning[1]{%
  \ifbool{ODw@Warnings}{%
    \par\textcolor{blue}{Warning: #1}\par}{}%
}% ODw@Warning
```

6.9 Misc Bridge Terms

6.9.1 Honour Cards

These macros retrieve the translations of the 4 alternative forms of the honour cards from the ODw-dictionary of the active language.

\Ace
\ace

```
\NewDocumentCommand{\Ace}{s t!}{%
  \bgroup%
%JW    \ODw@GameFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{A-(ODw)}}%
```

```
        {\GetTranslation{Ace-(ODw)}}%
    }{%
       \IfBooleanTF{#2}%
         {\GetTranslation{a-(ODw)}}%
         {\GetTranslation{ace-(ODw)}}%
   }%
   \egroup%
   \xspace%
}% Ace
%
\def\ace{\Ace*!}
```

\King
\king

```
\NewDocumentCommand{\King}{s t!}{%
   \bgroup%
%JW    \ODw@GameFont%
     \IfBooleanTF{#1}{%
       \IfBooleanTF{#2}%
         {\GetTranslation{K-(ODw)}}%
         {\GetTranslation{King-(ODw)}}%
    }{%
       \IfBooleanTF{#2}%
         {\GetTranslation{k-(ODw)}}%
         {\GetTranslation{king-(ODw)}}%
     }%
   \egroup%
   \xspace%
}% King
%
\def\king{\King*!}
```

\Queen
\queen

```
\NewDocumentCommand{\Queen}{s t!}{%
   \bgroup%
%JW    \ODw@GameFont%
     \IfBooleanTF{#1}{%
       \IfBooleanTF{#2}%
         {\GetTranslation{Q-(ODw)}}%
         {\GetTranslation{Queen-(ODw)}}%
    }{%
       \IfBooleanTF{#2}%
         {\GetTranslation{q-(ODw)}}%
         {\GetTranslation{queen-(ODw)}}%
```

```
    }%
  \egroup%
  \xspace%
}% Queen
%
\def\queen{\Queen*!}

\Jack
\jack \NewDocumentCommand{\Jack}{s t!}{%
  \bgroup%
%JW    \ODw@GameFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{J-(ODw)}}%
        {\GetTranslation{Jack-(ODw)}}%
   }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{j-(ODw)}}%
        {\GetTranslation{jack-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% Jack
%
\def\jack{\Jack*!}
```

6.9.2 Vulnerability

These macros retrieve the translations of the 4 alternative forms of the commands `\All` and `\None` from the `ODw`-dictionary of the active language. As there is no short form for them in the English language, we just code these entries in the `ODw`-dictionaries with an exclamation mark '`!`'.

```
\All
\all \NewDocumentCommand{\All}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{All!-(ODw)}}%
        {\GetTranslation{All-(ODw)}}%
   }{%
```

```
      \IfBooleanTF{#2}%
        {\GetTranslation{all!-(ODw)}}%
        {\GetTranslation{all-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% All
%
\def\all{\All*}
```

\None
\none

```
\NewDocumentCommand{\None}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{None!-(ODw)}}%
        {\GetTranslation{None-(ODw)}}%
   }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{none!-(ODw)}}%
        {\GetTranslation{none-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% None
%
\def\none{\None*}
%
```

6.9.3 Diagram Annotations

These macros retrieve the translations of the 4 alternative forms of the commands `\Contract`, `\Lead`, `\Declarer`, `\Board` and `\Deal` from the `ODw`-dictionary of the active language.

\Contract
\contract

```
\NewDocumentCommand{\Contract}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{Contr-(ODw)}}%
```

```
        {\GetTranslation{Contract-(ODw)}}%
    }{%
       \IfBooleanTF{#2}%
         {\GetTranslation{contr-(ODw)}}%
         {\GetTranslation{contract-(ODw)}}%
     }%
  \egroup%
  \xspace%
}% Contract
%
\def\contract{\Contract*}
%
```

\Lead
\lead

```
\NewDocumentCommand{\Lead}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{Lead!-(ODw)}}%
        {\GetTranslation{Lead-(ODw)}}%
    }{% else #1
      \IfBooleanTF{#2}%
        {\GetTranslation{lead!-(ODw)}}%
        {\GetTranslation{lead-(ODw)}}%
    }% #1
  \egroup%
  \xspace%
}% Lead
%
\def\lead{\Lead*}
%
```

\Declarer
\declarer

```
\NewDocumentCommand{\Declarer}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{Decl-(ODw)}}%
        {\GetTranslation{Declarer-(ODw)}}%
    }{%
       \IfBooleanTF{#2}%
         {\GetTranslation{decl-(ODw)}}%
```

```
        {\GetTranslation{declarer-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% Declarer
%
\def\declarer{\Declarer*}
%
```

\by

```
\newcommand\by{%
  \bgroup%
    \ODw@OtherFont%
    \GetTranslation{by-(ODw)}%
  \egroup%
  \xspace%
}% by
```

\Board
\board

```
\NewDocumentCommand{\Board}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{Brd-(ODw)}}%
        {\GetTranslation{Board-(ODw)}}%
   }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{brd-(ODw)}}%
        {\GetTranslation{board-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% Board
%
\def\board{\Board*}
%
```

\Deal
\deal

```
\NewDocumentCommand{\Deal}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
```

```
        {\GetTranslation{Deal!-(ODw)}}%
        {\GetTranslation{Deal-(ODw)}}%
    }{%
       \IfBooleanTF{#2}%
        {\GetTranslation{deal!-(ODw)}}%
        {\GetTranslation{deal-(ODw)}}%
     }%
  \egroup%
  \xspace%
}% Deal
%
\def\deal{\Deal*}
%
```

\doubled The commands \doubled and \redoubled do not have a short form.

```
\NewDocumentCommand{\doubled}{s}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \GetTranslation{Doubled-(ODw)}}{%
      \GetTranslation{doubled-(ODw)}%
    }%
  \egroup%
  \xspace%
}% doubled
```

\redoubled

```
\NewDocumentCommand{\redoubled}{s}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \GetTranslation{Redoubled-(ODw)}}{%
      \GetTranslation{redoubled-(ODw)}%
    }%
  \egroup%
  \xspace%
}% redoubled
```

6.9.4 Point Units

These macros retrieve the translations of the commands \hpts, \lpts, \dpts and \tpts from the ODw-dictionary of the active language.

\hpts

\HCP

```
\NewDocumentCommand{\hpts}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{HCP-(ODw)}}%
        {\GetTranslation{High Card Points-(ODw)}}%
    }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{hcp-(ODw)}}%
        {\GetTranslation{high card points-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% High Card Points
%
\def\HCP{\hpts*!}
```

\lpts
\LP

```
\NewDocumentCommand{\lpts}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{LP-(ODw)}}%
        {\GetTranslation{Length Points-(ODw)}}%
    }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{lp-(ODw)}}%
        {\GetTranslation{length points-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% Length Points
%
\def\LP{\lpts*!}
```

\dpts
\DP

```
\NewDocumentCommand{\dpts}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
```

```
        {\GetTranslation{DP-(ODw)}}%
        {\GetTranslation{Distribution Points-(ODw)}}%
    }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{dp-(ODw)}}%
        {\GetTranslation{distribution points-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% Distribution Points
%
\def\DP{\dpts*!}
```

\tpts
\TP
```
\NewDocumentCommand{\tpts}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{TP-(ODw)}}%
        {\GetTranslation{Total Points-(ODw)}}%
    }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{tp-(ODw)}}%
        {\GetTranslation{total points-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% Total Points
%
\def\TP{\tpts*!}
```

6.9.5 Forcings

These macros retrieve the translations of the commands `\gforce`, `\sforce`, `\nmforce`, `\tsforce` and `\fsforce` from the `ODw`-dictionary of the active language.

\gforce
\GF
```
\NewDocumentCommand{\gforce}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
```

```
      \IfBooleanTF{#2}%
        {\GetTranslation{GF-(ODw)}}%
        {\GetTranslation{Game Forcing-(ODw)}}%
    }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{gf-(ODw)}}%
        {\GetTranslation{game forcing-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% Game Forcing
%
\def\GF{\gforce*!}
```

\sforce
\SF

```
\NewDocumentCommand{\sforce}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{SF-(ODw)}}%
        {\GetTranslation{Semi Forcing-(ODw)}}%
    }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{sf-(ODw)}}%
        {\GetTranslation{semi forcing-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% Semi Forcing
%
\def\SF{\sforce*!}
```

\nmforce
\NMF

```
\NewDocumentCommand{\nmforce}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{NMF-(ODw)}}%
        {\GetTranslation{New Minor Forcing-(ODw)}}%
    }{%
      \IfBooleanTF{#2}%
```

```
        {\GetTranslation{nmf-(ODw)}}%
        {\GetTranslation{new minor forcing-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% New Minor Forcing
%
\def\NMF{\nmforce*!}
```

\tsforce

\TSF

```
\NewDocumentCommand{\tsforce}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{TSF-(ODw)}}%
        {\GetTranslation{Third Suit Forcing-(ODw)}}%
    }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{tsf-(ODw)}}%
        {\GetTranslation{third suit forcing-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% Third Suit Forcing
%
\def\TSF{\tsforce*!}
```

\fsforce

\FSF

```
\NewDocumentCommand{\fsforce}{s t!}{%
  \bgroup%
    \ODw@OtherFont%
    \IfBooleanTF{#1}{%
      \IfBooleanTF{#2}%
        {\GetTranslation{FSF-(ODw)}}%
        {\GetTranslation{Fourth Suit Forcing-(ODw)}}%
    }{%
      \IfBooleanTF{#2}%
        {\GetTranslation{fsf-(ODw)}}%
        {\GetTranslation{fourth suit forcing-(ODw)}}%
    }%
  \egroup%
  \xspace%
}% Fourth Suit Forcing
```

```
%
\def\FSF{\fsforce*!}
```

6.10 Initialization

It's time to prepare everything. We clear the game and set the defaults.

```
\newgame
```

Set the default fonts

```
\setdefaults{bidder=\mdseries\sffamily}
\setdefaults{compass=\mdseries\sffamily}
\setdefaults{game=\bfseries\sffamily}
\setdefaults{legend=\mdseries\rmfamily}
\setdefaults{name=\mdseries\slshape}
\setdefaults*{other=\bfseries\sffamily}
```

Set the compass

```
\setdefaults{compshow=on,compturn=off}
```

set the start column for bidding (West is recommended) and the long form.

```
\setdefaults{bidfirst=W,bidders=on,bidlong=on}
```

Now we load the dictionaries for the languages that are to be used in the document. We use `tracklang` to iterate over all the document languages and load the corresponding `ODw`-dictionaries. Due to an inconsistency between `babel` and `translations` with respect to the Norwegian language (`babel` calls this language *norsk* whereas `translations` insist on using *norwegian*, we redefine `\thislang` to the latter if it happens to be *norsk*.

```
\AtBeginDocument{%
\ForEachTrackedLanguage{\thislang}{%
  \ifthenelse{\equal{\thislang}{norsk}}%
    {\def\thislang{norwegian}}{}%
  \IfFileExists{ODw-\thislang.trsl}%
  {%
    \LoadDictionaryFor{\thislang}{ODw}%
    \PackageInfo{ODw}{%
      Translation dictionary ODw-\thislang.trsl loaded%
    }%
  }{%
    \PackageWarning{ODw}{%
      Translation dictionary ODw-\thislang.trsl not found%
```

```
    }%
  }%
}% ForEach
}% AtBeginDocument
```

That's it folks, happy TEXing!

```
\endinput% onedown.sty
```

7 References

[1] Kees van der Laan: *Typsetting Bridge via TEX*, TUGboat Vol. 11, No. 2 (1990), p265ff

[2] Richard Pavlicek: *Bridge Writing Style Guide*, http://www.rpbridge.net/7z69.htm

8 Change History

documentation accordingly. 1

v1.0

General: Adapted the urls, the directories and some filenames to conform to the CTAN-standard and made the bundle ready for upload. Corrected a small bug in `\ODw@Compass` that was introduced in v0.9. Made all relevant text writing macros in 4 versions with/without tokens * and !. Corrected a sizing/font bug. Added 4 variants of many text macros, onesuit-NE/NW/. Adapted `\handskip`. Added code to work around a `babel-translations` inconsistency w.r.t. norsk/norwegian. 1

9 Index

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

P

Q

R

S

T

V

W

10 List of User Commands