The `wheelchart` package

Diagrams with circular or other shapes using Ti*k*Z and LaTeX3

Matthias Floré

Version 4.0 (2024/07/28)

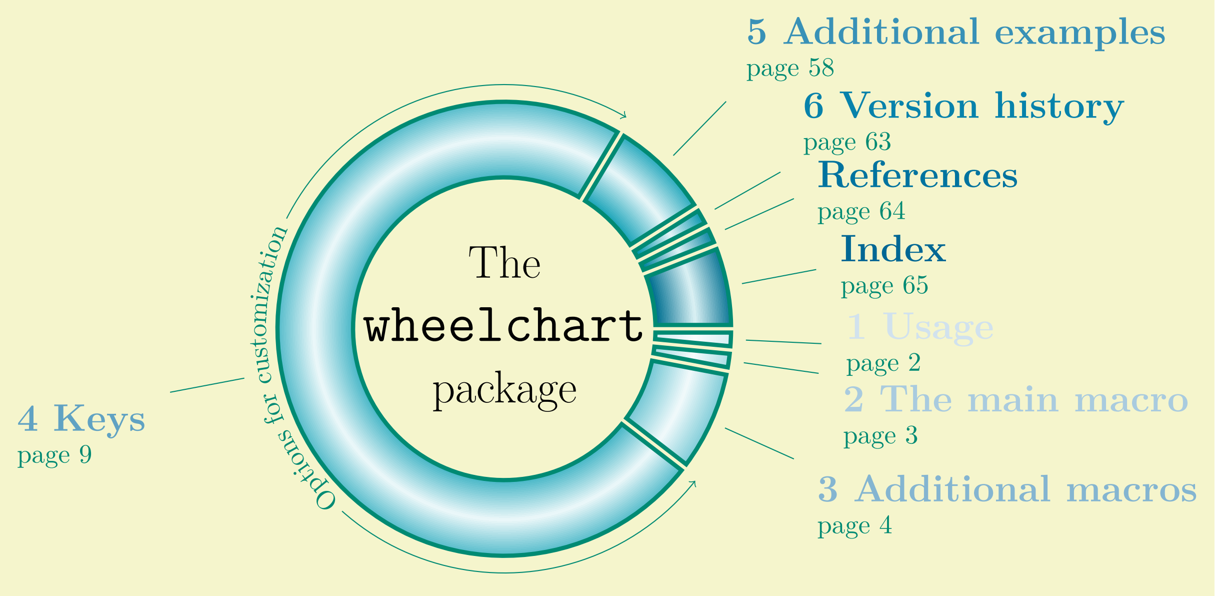
Abstract

This package is based on the package `tikz` (see [5]) and can be used to draw various kinds of diagrams such as bar charts, doughnut charts, infographics, pie charts, ring charts, square charts, sunburst charts, waffle charts and wheel charts.

It provides several options to customize the diagrams. It is also possible to specify a plot for the shape of the chart. Furthermore a legend can be added and the table of contents can be displayed as one of these diagrams.

Other tools for creating wheel charts or pie charts can be found in [2], [1], [4], [6] and [3].

Contents



```
\usepackage {etoolbox} \usetikzlibrary {decorations.text} \usepackage {etoc} \etocsettocstyle {\hypersetup {hidelinks}}{}
\etocglobaldefs \usepackage [linktoc=all]{hyperref}
\begin{tikzpicture}
\pgfkeys{
  /wheelchart,
  for loop start={\colorlet{WCcolor}{MidnightBlue!\fpeval{(\WCcount/\WCtotalcount)*100}}},
  gap,
  start angle=0,
  value=\WCetocthenumberofpages
}
\wheelchart[
  after slices={
    \pgfdeclareradialshading{WCshading}{\pgfpoint{0cm}{0cm}}{
      color(0bp)=(WCcolor);
      color(16.66666bp)=(WCcolor);%2/3 * 25bp
      color(20.83333bp)=(WCcolor!10);%2.5/3 * 25bp
      color(25bp)=(WCcolor);
      color(50bp)=(WCcolor)
    }
    \shade[even odd rule,shading=WCshading] (0,0) circle[radius=3] circle[radius=2];
  },
  data=,
  etoc count total pages=\getpagerefnumber{Thesourcecode}-1,%\totalpages
  etoc level=section,
  etoc name=wheelchart table of contents,
  slices style={
    fill=none,
    clip
  }
]{}
\hypersetup{linkcolor=.}
\wheelchart[
  anchor ysep{7,8}=30,
  data={%
    \textcolor{WCcolor}{%
    \textbf{\Large\ifdefempty{\WCetocthenumber}{}{\WCetocthelinkednumber{} }\WCetocthelinkedname}}\\%
    \textcolor{PineGreen}{page \WCetocthelinkedpage}%
  },
  etoc use name=wheelchart table of contents,
  lines,
  lines style=PineGreen,
  middle={\LARGE The\\[10pt]\huge\texttt{wheelchart}\\[10pt]\LARGE package},
  slice{\getrefnumber{Keys}}={
    arc={
      draw=PineGreen,
      ->
    },
    arc around text,
    arc data=~Options for customization~,
    arc data style={text color=PineGreen},
    lines sep=0.5
  },
  slices style={
    fill=none,
    draw=PineGreen,
    ultra thick
  }
]{}
\end{tikzpicture}
```

1 Usage

The package **wheelchart** can be used by putting the following in the preamble.

```
\usepackage{wheelchart}
```

The package **wheelchart** loads the package **tikz** and the Ti*k*Z library **calc**.

Many examples in this manual use colors which can be defined by giving **dvipsnames** as an option to `\documentclass`.

2 The main macro

\wheelchart[⟨*options*⟩]{⟨*wheelchart data*⟩}

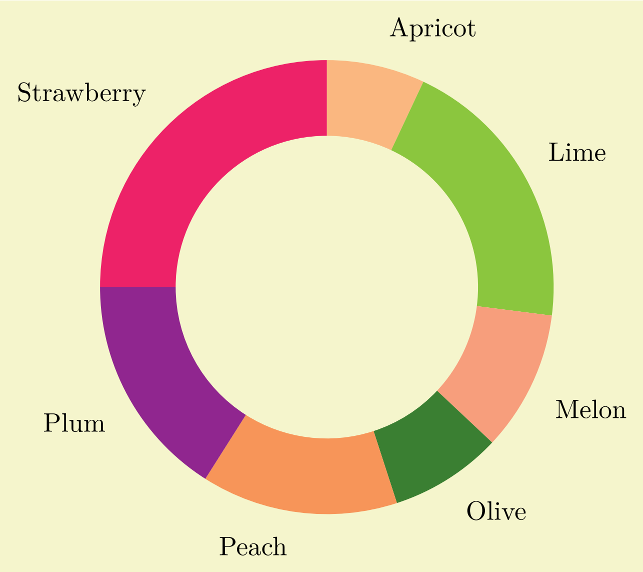
This command can be placed inside a `tikzpicture` environment. It draws a wheelchart with ⟨*wheelchart data*⟩. With the initial settings, the ⟨*wheelchart data*⟩ is a comma separated list in which each item corresponds to one slice of the wheelchart and consists of data separated by a `/`. The precise syntax of the ⟨*wheelchart data*⟩ will be explained below. The ⟨*options*⟩ can be given with the keys described in Section 4.

\exampleforthismanual

To simplify the creation of examples in this manual, we define the ⟨*wheelchart data*⟩ below.

```
\gdef\exampleforthismanual{%
14/Apricot/Apricot/{A, B, C, E, K}/north east lines/0/0/Gray,
40/LimeGreen/Lime/{B, C}/grid/0/15/Black,
20/Melon/Melon/{A, C}//0.5/0/none,
16/OliveGreen/Olive/{A, B, E, K}/dots/0/0/none,
28/Peach/Peach/{A, B, C, E, K}/fivepointed stars/0/0/Lavender,
32/Plum/Plum/{A, B, C, E, K}/bricks/0/-15/none,
50/WildStrawberry/Strawberry/{B, C, E, K}//1/0/DarkOrchid%
}
```

The default wheelchart with these data is shown below.



```
\begin{tikzpicture}
\wheelchart{\exampleforthismanual}
\end{tikzpicture}
```

3 Additional macros

`\WCangle[⟨number⟩]{⟨angle pos⟩}{⟨angle shift⟩}{⟨pos⟩}{⟨sep⟩}`

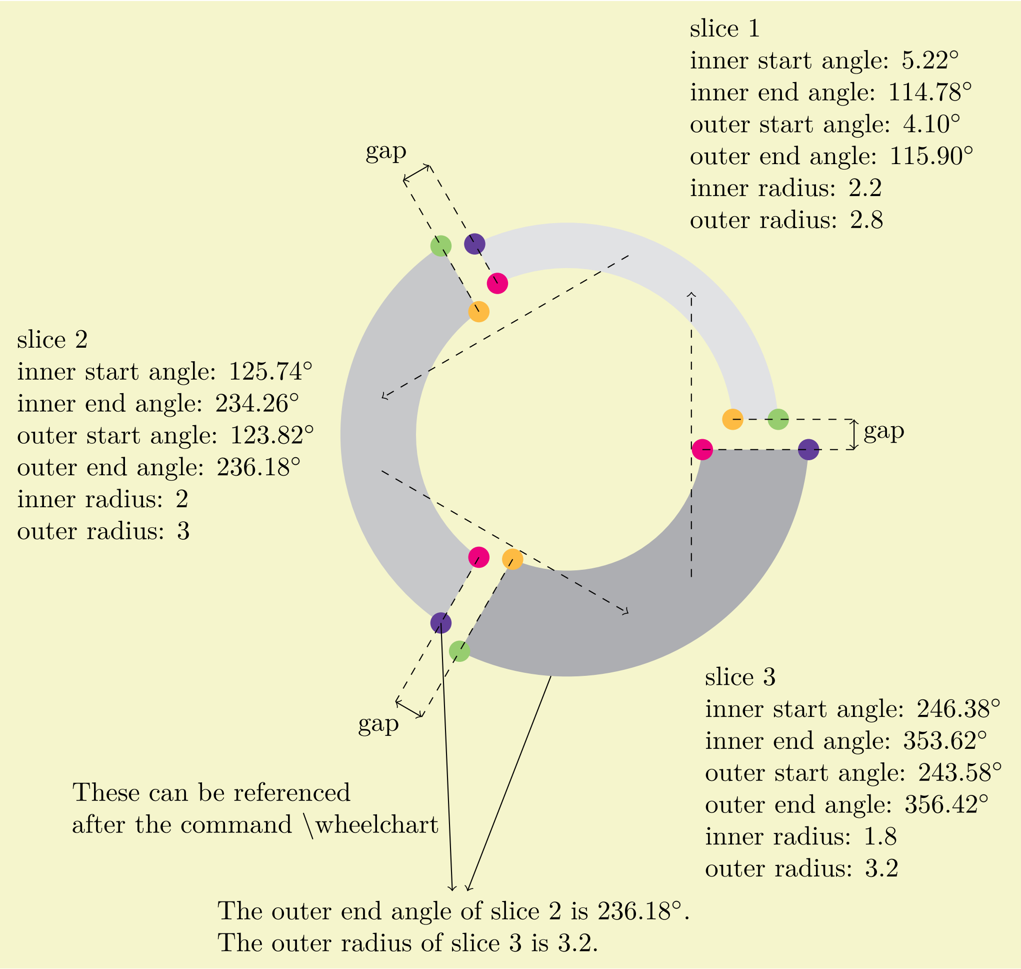
This command gives the angle in degrees of the point which is constructed as follows.

1. Consider the inner start angle and the inner end angle of slice ⟨*number*⟩. This ⟨*number*⟩ is computed modulo the total number of slices. Form the convex combination of these two angles with parameter ⟨*angle pos*⟩. Then add ⟨*angle shift*⟩. Then consider the point with this angle and as radius the inner radius.
2. Consider the similar point constructed with the outer start angle, the outer end angle and the outer radius of slice ⟨*number*⟩. Then construct the straight line between those two points.
3. Consider the radius given by the command `\WCradius` with arguments ⟨*number*⟩, ⟨*pos*⟩ and ⟨*sep*⟩.
4. Consider the intersection of the previous line and the arc with the previous radius. The command `\WCangle` gives the angle in degrees of this point.

The default value for ⟨*number*⟩ is `\WCcount`.

The command `\WCangle` can be used in the ⟨*options*⟩ of the command `\wheelchart`. It can also be used after the command `\wheelchart`. In that case, the computed angles will correspond to the last `\wheelchart`.

The command `\WCangle` should not be used with a plot.



These can be referenced after the command \wheelchart

The outer end angle of slice 2 is 236.18°.
The outer radius of slice 3 is 3.2.

```
\usepackage {siunitx}
\begin{tikzpicture}
\sisetup{round-mode=places,round-precision=2}
\wheelchart[
  counterclockwise,
  data=slice \WCcount\\
    inner start angle: \ang{\WCangle{0}{0}{0}{0}}\\
    inner end angle: \ang{\WCangle{1}{0}{0}{0}}\\
    outer start angle: \ang{\WCangle{0}{0}{1}{0}}\\
    outer end angle: \ang{\WCangle{1}{0}{1}{0}}\\
    inner radius: \WCradius{0}{0}\\
    outer radius: \WCradius{1}{0},
  gap=0.2,
  inner radius{list}={2.2,2,1.8},
  legend entry={
    \fill[Dandelion] (\WCcoordinate[\WCcount +1]{inner start}) circle[radius=4pt];
    \fill[RubineRed] (\WCcoordinate{inner end}) circle[radius=4pt];
    \fill[YellowGreen] (\WCcoordinate[\WCcount +1]{outer start}) circle[radius=4pt];
    \fill[RoyalPurple] (\WCcoordinate{outer end}) circle[radius=4pt];
    \draw[->,dashed] (\WCpoint[\WCcount -1]{0.6}{0}{0.5}{0})--(\WCpoint{0.4}{0}{0.5}{0});
    \draw[dashed] (\WCcoordinate{inner end})
      --(\WCpoint{1}{0}{1}{\WClistsep}) coordinate (A);
    \draw[dashed] (\WCcoordinate[\WCcount +1]{inner start})
      --(\WCpoint[\WCcount +1]{0}{0}{1}{1}) coordinate (B);
    \draw[<->] (A)--(B) node[\WClistpos,midway] {gap};
  },
  outer radius{list}={2.8,3,3.2},
  slices style{list}={Gray!25,Gray!50,Gray!75},
  start angle=0,
  total count=3,
  WClistpos={above left,below left,right},
  WClistsep={1.2,1.2,0.6}
]{}
\node[align=left] (N) at (-1.5,-6.5) {%
  The outer end angle of slice 2 is \ang{\WCangle[2]{1}{0}{1}{0}}.\\
  The outer radius of slice 3 is \WCradius[3]{1}{0}.%
};
\draw[->] (\WCcoordinate[2]{outer end})--(N) node[left,pos=0.7,align=left]
  {These can be referenced\\after the command \textbackslash wheelchart};
\draw[->] (\WCpoint[3]{0.2}{0}{1}{0})--(N);
\end{tikzpicture}
```

`\WCcoordinate[⟨number⟩]{⟨name⟩}`

- If the key `discrete` is false then this command gives the coordinate positioned at ⟨*name*⟩ of slice ⟨*number*⟩. The ⟨*name*⟩ can be `inner end`, `inner start`, `outer end` or `outer start`.
- If the key `discrete` is true then this command gives the coordinate positioned at point ⟨*name*⟩ of slice ⟨*number*⟩. The ⟨*name*⟩ can be an integer from 1 till the number of points of slice ⟨*number*⟩.

The ⟨*number*⟩ is computed modulo the total number of slices.

The default value for ⟨*number*⟩ is `\WCcount`.

The command `\WCcoordinate` can be used in the ⟨*options*⟩ of the command `\wheelchart`. It can also be used after the command `\wheelchart`. In that case, the coordinate will correspond to the last `\wheelchart`.

`\WCcount`

This macro gives the current number of the slice in the ⟨*wheelchart data*⟩.

`\WCcountdiscrete`

If the key `discrete` is true then this macro gives the current number of the Ti*k*Z pic from the key `discrete pic`.

`\WCdataangle`

This macro is similar to `\WCmidangle` but also takes into account the keys `data angle pos`, `data angle shift`, `data pos` and `data sep` (with respect to the key `counterclockwise`).

`\WCetocthelinkedname`

`\WCetocthelinkednumber`

\WCetocthelinkedpage

\WCetocthename

\WCetocthenumber

\WCetocthenumberofpages

\WCetocthepage

These macros are defined when the key `etoc level` is used.

\WClegend

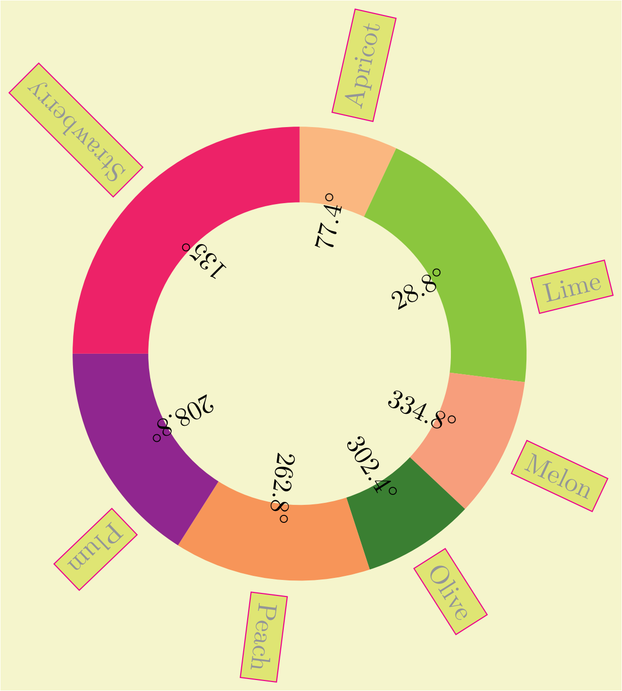
If the key `legend row` is used then the resulting legend is stored in the macro `\WClegend`.

\WClist⟨*name*⟩

This macro is defined when the key `WClist`⟨*name*⟩ is used and gives the element in the ⟨*list*⟩ given to the key `WClist`⟨*name*⟩ with as index `\WCcount` modulo the length of this ⟨*list*⟩. The ⟨*name*⟩ is the one given to the key `WClist`⟨*name*⟩.

\WCmidangle

This macro gives the angle in degrees modulo 360 of the middle of the current slice.



```
\usepackage {siunitx}
\begin{tikzpicture}
\wheelchart[
  data angle shift=\WCvarG,
  data style={
    rotate=\WCdataangle,
    draw=Magenta,
    fill=GreenYellow,
    anchor=west,
    text=Gray
  },
  inner data=\ang{\WCmidangle},
  inner data style={
    rotate=\WCmidangle,
    font=\ttfamily
  }
]{\exampleforthismanual}
\end{tikzpicture}
```

\WCperc

This macro displays `\WCpercentagerounded` followed by a % symbol.

If the package `siunitx` is loaded then the following code is used outside the key `arc data`. The package `siunitx` can be loaded before or after the package `wheelchart`.

```
\qty{\WCpercentagerounded}{\percent}
```

If the package `siunitx` is not loaded then the following code is used outside the key `arc data`.

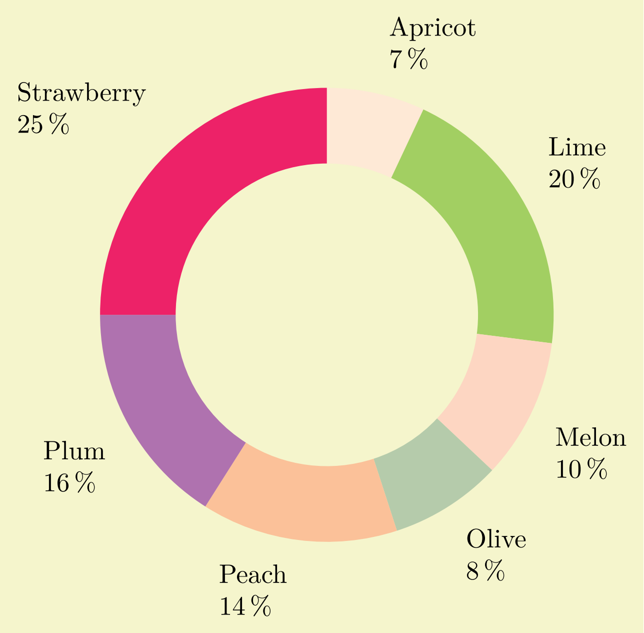
```
\WCpercentagerounded\,\%
```

Inside the key `arc data`, the following code is used.

```
\WCpercentagerounded{\,}{\%}
```

\WCpercentage

This macro gives the percentage of the current slice where the total is computed with the values of the key `value`. Note that rounding errors can occur.



```
\usepackage {siunitx}
\begin{tikzpicture}
\wheelchart[
  data=\WCvarC\\\WCperc,
  slices style={
    \WCvarB!\fpeval{4*\WCpercentage}
  }
]{\exampleforthismanual}
\end{tikzpicture}
```

\WCpercentagerounded

This macro displays `\WCpercentage` rounded up to the number of decimals determined by the key `perc precision`.

\WCpoint[⟨*number*⟩]{⟨*angle pos*⟩}{⟨*angle shift*⟩}{⟨*pos*⟩}{⟨*sep*⟩}

This command gives the point where the angle is determined by `\WCangle` and the radius by `\WCradius` computed with the given arguments.

The ⟨*number*⟩ is computed modulo the total number of slices.

The default value for ⟨*number*⟩ is `\WCcount`.

The command `\WCpoint` can be used in the ⟨*options*⟩ of the command `\wheelchart`. It can also be used after the command `\wheelchart`. In that case, the point will correspond to the last `\wheelchart`.

The command `\WCpoint` should not be used with a plot.

\WCradius[⟨*number*⟩]{⟨*pos*⟩}{⟨*sep*⟩}

This command gives the convex combination with parameter ⟨*pos*⟩ of the inner radius of slice ⟨*number*⟩ minus ⟨*sep*⟩ and the outer radius of slice ⟨*number*⟩ plus ⟨*sep*⟩.

The ⟨*number*⟩ is computed modulo the total number of slices.

The default value for ⟨*number*⟩ is `\WCcount`.

The command `\WCradius` can be used in the ⟨*options*⟩ of the command `\wheelchart`. It can also be used after the command `\wheelchart`. In that case, the computed radius will correspond to the last `\wheelchart`.

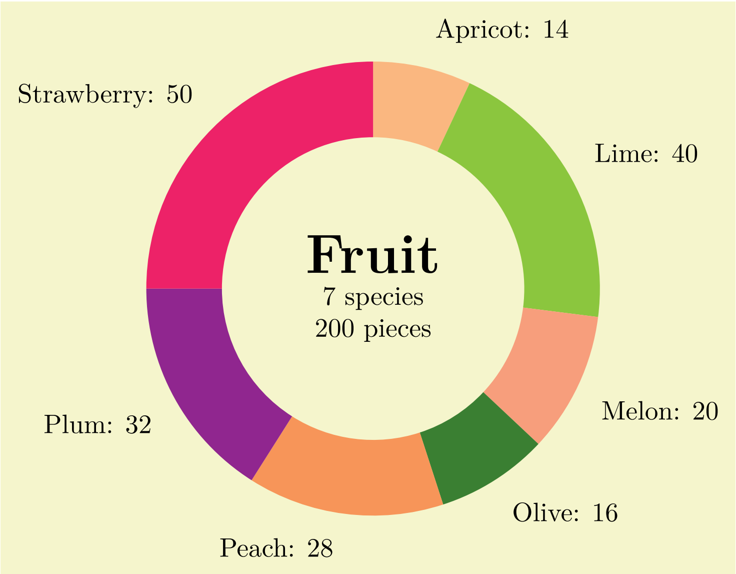
The command `\WCradius` should not be used with a plot.

\WCtotalcount

This macro gives the total number of slices.

\WCtotalnum

This macro gives the sum of all values of the key `value`.



```
\begin{tikzpicture}
\wheelchart[
  data=\WCvarC: \WCvarA,
  middle={%
    \textbf{\huge Fruit}\\%
    \WCtotalcount{} species\\%
    \WCtotalnum{} pieces%
  }
]{\exampleforthismanual}
\end{tikzpicture}
```

`\WCvarA`

`\WCvarB`

`\WCvarC`

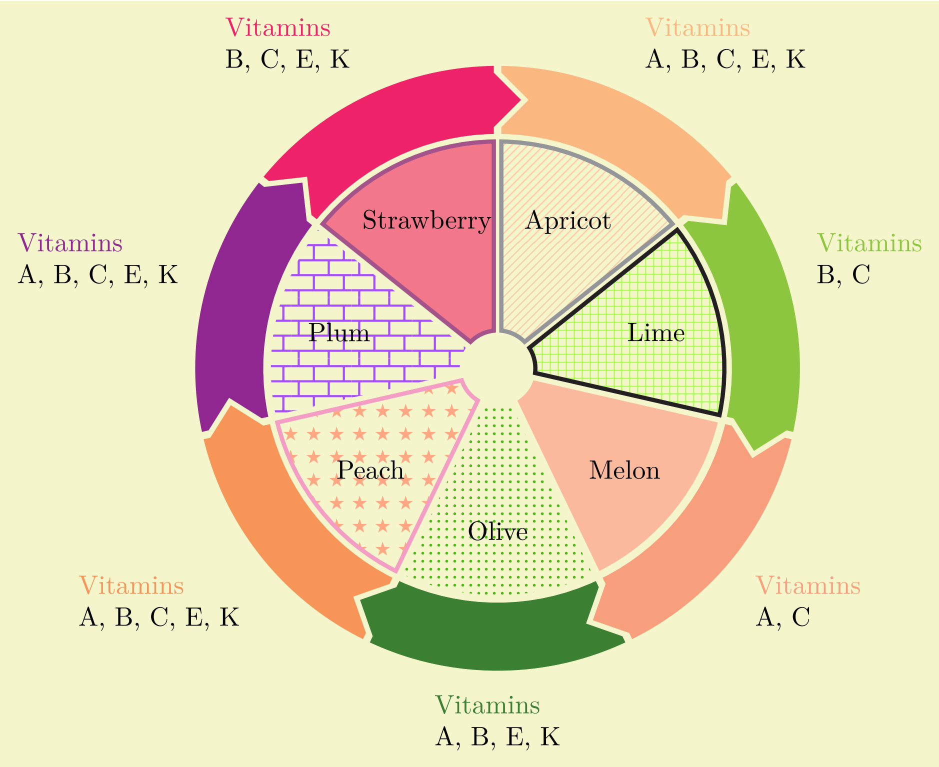
`\`⟨*prefix*⟩⟨*name*⟩

The ⟨*wheelchart data*⟩ in the command `\wheelchart` is a list in which the items are separated by the value of the key `separator rows`. Each item in this list corresponds to one slice of the wheelchart and consists of data separated by the value of the key `separator columns`. With the initial settings, these individual data are interpreted as the macros `\WCvarA`, `\WCvarB`, `\WCvarC`, …, `\WCvarZ`, `\WCvarAA` and so on and can be accessed within the ⟨*options*⟩ of the command `\wheelchart` if applicable.

The names of these macros can be specified with ⟨*prefix*⟩ and ⟨*name*⟩ which are determined by respectively the keys `header prefix` and `header`.

Initially, only `\WCvarA`, `\WCvarB` and `\WCvarC` are used for `value=\WCvarA`, `slices style=\WCvarB` and `data=\WCvarC`.

Other ways to specify data are by using for example a list such as an array with the package `tikz`, a list with the package `listofitems` or with the key `WClist`⟨*name*⟩.



```
\usetikzlibrary {patterns}
\begin{tikzpicture}
\pgfkeys{
  /wheelchart,
  gap,
  header={value,color,text,vitamins,pattern,explode,data angle shift,border},
  header prefix=my,
  value=1
}
\wheelchart[
  data=,
  radius={0.5}{3},
  slices style={\mycolor!70,draw=\myborder,ultra thick,pattern=\mypattern,pattern color=\mycolor!70},
  wheel data=\mytext,
  %wheel data style={shift={(\WCmidangle:0.5)}},
  %wheel data pos=0.5
]{\exampleforthismanual}
\wheelchart[
  data=\textcolor{\mycolor}{Vitamins}\\\myvitamins,
  radius={3.1}{4},
  slices arrow={1}{0.2},
  slices style=\mycolor
]{\exampleforthismanual}
\end{tikzpicture}
```

4 Keys

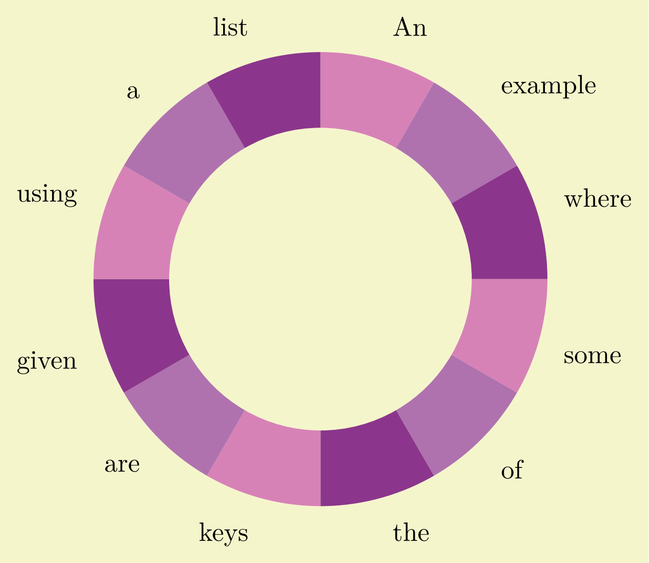
The keys in this Section can be given as ⟨*options*⟩ to the command `\wheelchart`.

If applicable, an optional non-empty ⟨*range*⟩ between braces can be given to a key after the ⟨*key name*⟩ except for the key `slice` where the ⟨*range*⟩ is mandatory. This ⟨*range*⟩ is processed with `\foreach` with the option `parse=true`. Hereafter the elements are processed with `\fp_eval:n`. If such a ⟨*range*⟩ is given to a key then the options given to this key will only be applied to a slice if the number of the slice is in the ⟨*range*⟩. The ⟨*range*⟩ only makes sense for a key which is processed for each slice. For example, the ⟨*range*⟩ does not make sense for the key `middle`.

Furthermore, it is possible to add `{list}` after the ⟨*key name*⟩. Then a list can be given to the key. This list is processed analogously as how the key `WClist`⟨*name*⟩ works. Then the result is given to the key.

Below are some examples for the options ⟨*range*⟩ and `{list}`.

- The following wheelchart can be obtained with the 3 possibilities below.

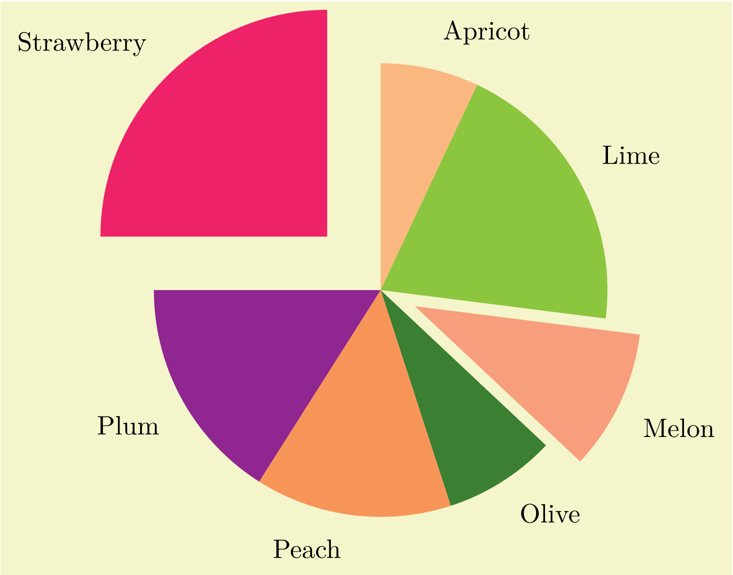


```
\begin{tikzpicture}
\wheelchart[
  data{list}={
    An,example,where,some,of,the,
    keys,are,given,using,a,list
  },
  slices style{list}={
    Thistle,Orchid,Fuchsia
  },
  total count=12
]{}
\end{tikzpicture}
```

```
\usepackage {listofitems}
\readlist\WCcolors{Thistle,Orchid,Fuchsia}

\setsepchar{ }
\readlist\WCdata{An example where some of the keys are given using a list}

data={\WCdata[\WCcount]},

slices style={
  /utils/exec={\pgfmathsetmacro{\WCcolornumber}{int(Mod({\WCcount-1},\WCcolorslen)+1)}},
  \WCcolors[\WCcolornumber]
},

total count=\WCdatalen,
```

```
slices style{1,4,...,\WCdatalen}=Thistle,
slices style{2,5,...,\WCdatalen}=Orchid,
slices style{3,6,...,\WCdatalen}=Fuchsia,
```

- The following wheelchart can be obtained with the 3 possibilities below.



```
\begin{tikzpicture}
\wheelchart[
  explode=\WCvarF,
  pie
]{\exampleforthismanual}
\end{tikzpicture}
```

```
explode={\WCcount==3?0.5:(\WCcount==7?1:0)},
```

```
explode{3}=0.5,
explode{7}=1,
```

`/wheelchart/after slices={⟨code⟩}` (no default, initially empty)

The ⟨*code*⟩ given to this key will be executed after each slice of the wheelchart.

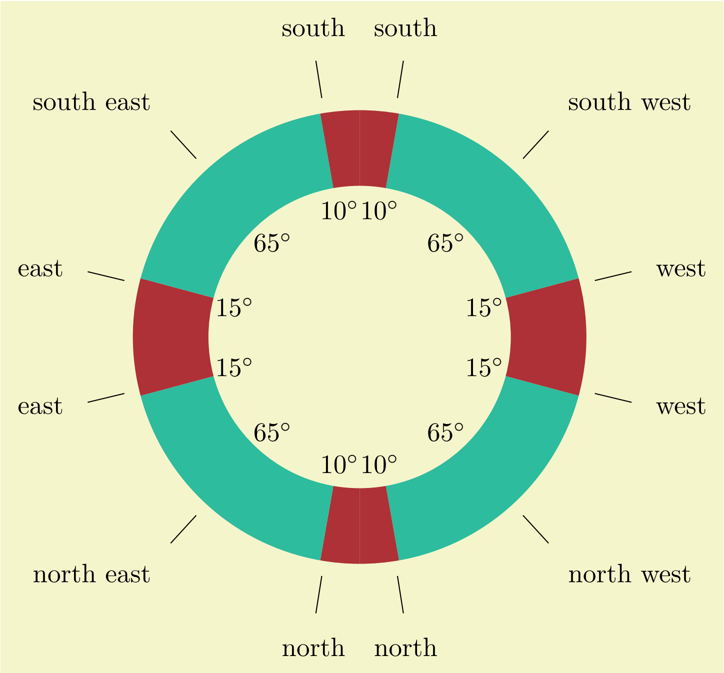
`/wheelchart/anchor xsep={⟨angle⟩}` (no default, initially 5)

`/wheelchart/anchor ysep={⟨angle⟩}` (no default, initially 5)

These keys determine the default anchor of the key `data` in the case that `lines ext=0`. Note that rounding errors can occur in the computation of the angle which is used to determine the default anchor according to Table 1.

Angle (up to rounding errors)	Anchor of the key `data` in the case that `lines ext=0`
0	west
90	south
180	east
270	north
For other angles not in $\{0, 90, 180, 270\}$:	
$[0, \texttt{anchor ysep}]$	west
$]\texttt{anchor ysep}, 90 - \texttt{anchor xsep}[$	south west
$[90 - \texttt{anchor xsep}, 90 + \texttt{anchor xsep}]$	south
$]90 + \texttt{anchor xsep}, 180 - \texttt{anchor ysep}[$	south east
$[180 - \texttt{anchor ysep}, 180 + \texttt{anchor ysep}]$	east
$]180 + \texttt{anchor ysep}, 270 - \texttt{anchor xsep}[$	north east
$[270 - \texttt{anchor xsep}, 270 + \texttt{anchor xsep}]$	north
$]270 + \texttt{anchor xsep}, 360 - \texttt{anchor ysep}[$	north west
$[360 - \texttt{anchor ysep}, 360]$	west

Table 1: Anchor of the key `data` in the case that `lines ext=0`.



```
\usepackage {siunitx}
\begin{tikzpicture}
\wheelchart[
  anchor xsep=10,
  anchor ysep=15,
  data=\WCvarA,
  data angle pos=\WClistdap,
  inner data=\ang{\WClistvalue},
  inner data angle pos=\WClistdap,
  inner data sep=0.3,
  lines=0.5,
  lines angle pos=\WClistdap,
  slices style{list}={
    Maroon,SeaGreen,Maroon
  },
  value=\WClistvalue,
  WClistdap={0.9,0.5,0.1},
  WClistvalue={10,65,15,15,65,10}
]{%
  south,
  south west,
  west,
  west,
  north west,
  north,
  north,
  north east,
  east,
  east,
  south east,
  south%
}
\end{tikzpicture}
```

The anchor of the key `data` can also be specified manually by using `data style={anchor=`⟨*anchor*⟩`}`.

`/wheelchart/arc={`⟨*options*⟩`}` (style, no default, initially empty)

If this key is set then an arc with the style determined by this key will be drawn following the arc or plot for a slice of the wheelchart.

`/wheelchart/arc around line={`⟨*number*⟩`}` (no default, initially `1`)

The contents of the key `arc data` can consist of multiple lines separated by `\\`. If the key `arc around text` is true then the corresponding line is determined by ⟨*number*⟩.

`/wheelchart/arc around text=`⟨*boolean*⟩ (default `true`, initially `false`)

If true then the arc with the style determined by the key `arc` will be split in two parts such that the gap between these two parts leaves space for the contents of line ⟨*number*⟩ of the key `arc data` where ⟨*number*⟩ is determined by the key `arc around line`. The space between the arc and the contents of the key `arc data` can be increased with for example `~` in `arc data=~text~`.

`/wheelchart/arc data={`⟨*text*⟩`}` (no default, initially empty)

This key contains the ⟨*text*⟩ which will be placed following the arc or plot for a slice of the wheelchart using the decoration `text along path`. This requires the Ti*k*Z library `decorations.text`. The style of this decoration is given as follows. First, the option `raise=-0.5ex` is given. Then `text align` is determined by the key `arc data align`. Thereafter, the style of the key `arc data style` is added.

The ⟨*text*⟩ can consist of multiple lines separated by `\\`.

Braces or multiple pairs of braces are required around some macros.

`/wheelchart/arc data align=center|left|right` (no default, initially `center`)

This key determines the alignment of the contents of the key `arc data`.

`/wheelchart/arc data angle pos={`⟨*value*⟩`}` (no default, initially `0.5`)

`/wheelchart/arc data angle shift={`⟨*angle*⟩`}` (no default, initially `0`)

These keys determine the position of the contents of the key `arc data` similar as the corresponding keys for the key `data`.

`/wheelchart/arc data dir={⟨value⟩}` (no default, initially `1`)

This key determines the direction of the contents of the key `arc data`. If the ⟨*value*⟩ is positive then the direction is the same as the direction of the slice. If the ⟨*value*⟩ is negative then the direction is reversed. The values `1` and `-1` are recommended.

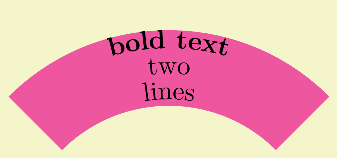
When the contents of the key `arc data` is placed, the corresponding domain for the arc or plot is estimated. A warning is given when the contents of the key `arc data` did (possibly) not fit. In this case, the absolute value of the key `arc data dir` should be increased.

If an error `Dimension too large` occurs then the absolute value of the key `arc data dir` should be increased or decreased depending on the situation.

`/wheelchart/arc data expand={⟨expansion type⟩}` (no default, initially `n`)

The contents of the key `arc data` can consist of multiple lines separated by `\\`. This splitting is done with `\seq_set_split:Nnn` or a variant thereof depending on the ⟨*expansion type*⟩ which determines the last letter in the signature. For most use cases, this ⟨*expansion type*⟩ is `n`, `e` or `f`.

In the example below, it is necessary to use `arc data expand=e` and to place `\noexpand` before `\bfseries`.



```
\usetikzlibrary {decorations.text}
\begin{tikzpicture}
\wheelchart[
  arc data=|\noexpand\bfseries|\WCvarC\\
    \WCvarD,
  arc data expand=e,
  arc data pos=0.5,
  data=,
  start angle=135,
  total angle=90
]{1/VioletRed/bold text/two\\lines}
\end{tikzpicture}
```

`/wheelchart/arc data line sep factor={⟨factor⟩}` (no default, initially `1`)

The contents of the key `arc data` can consist of multiple lines separated by `\\`. The ⟨*factor*⟩ determines the spacing between these lines.

`/wheelchart/arc data lines pos={⟨factor⟩}` (no default, initially `0.5`)

`/wheelchart/arc data lines shift={⟨value⟩}` (no default, initially `0`)

The positioning of the lines of the key `arc data` is determined by the index $k-1-$`arc data lines pos`$\cdot (N-1)+$`arc data lines shift` where N is the number of lines and $k \in \{1,\ldots,N\}$.

`/wheelchart/arc data pos={⟨value⟩}` (no default, initially `1`)

`/wheelchart/arc data sep={⟨value⟩}` (no default, initially `1ex/1cm`)

These keys determine the position of the contents of the key `arc data` similar as the corresponding keys for the key `data`.

`/wheelchart/arc data style={⟨options⟩}` (style, no default, initially empty)

This key accepts a list of keys which will be applied to the decoration for the key `arc data`.

`/wheelchart/arc first half={⟨options⟩}` (style, no default, initially empty)

If `arc around text` is true then the arc with the style determined by the key `arc` will be split in two parts. The style determined by the key `arc first half` will be appended to the first half of the arc.

`/wheelchart/arc pos={⟨value⟩}` (no default, initially `1`)

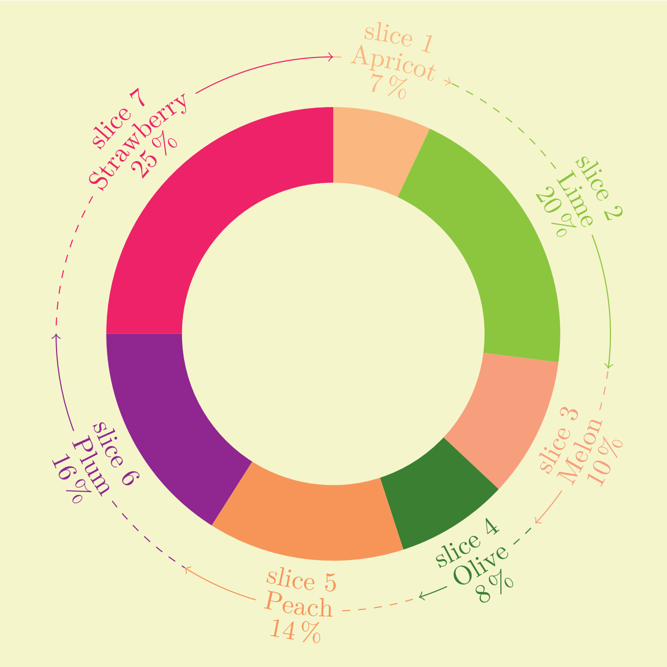
This key determines the position of the arc similar as the corresponding key for the key `data`.

`/wheelchart/arc second half={⟨options⟩}` (style, no default, initially empty)

This key is similar to the key `arc first half` but will be appended to the second half of the arc.

`/wheelchart/arc sep={⟨value⟩}` (no default, initially `1ex/1cm`)

This key determines the position of the arc similar as the corresponding key for the key `data`. Note that the actual distance is given by `0.5ex/1cm` plus `arc sep` to match the option `raise=-0.5ex` given to the decoration for the key `arc data`.



```
\usetikzlibrary {decorations.text}
\begin{tikzpicture}
\wheelchart[
  arc=\WCvarB,
  arc around line=2,
  arc around text,
  arc data=slice \WCcount\\
    {~}\WCvarC{~}\\
    \WCperc,
  arc data dir={\WCmidangle<180?1:-1},
  arc data expand=f,
  arc data pos=1.3,
  arc data style={text color=\WCvarB},
  arc first half=dashed,
  arc pos=1.3,
  arc second half=->,
  data={}
]{\exampleforthismanual}
\useasboundingbox (0,0)
  circle[radius=4.3];
\end{tikzpicture}
```

`/wheelchart/at={⟨point⟩}` (no default, initially `(0,0)`)

This key defines the center of the wheelchart.

`/wheelchart/before slices={⟨code⟩}` (no default, initially empty)

The ⟨*code*⟩ given to this key will be executed before each slice of the wheelchart.

`/wheelchart/caption={⟨text⟩}` (no default, initially empty)

This key contains the ⟨*text*⟩ which will be placed below the wheelchart. The ⟨*text*⟩ is placed in a node. The x coordinate of this node is the x coordinate of the center of the wheelchart, which is defined by the key `at`. In general, this is *not* the same as the x coordinate of the center of the `local bounding box` around the wheelchart. The y coordinate of this node is at a value determined by the key `caption sep` below the south of the `local bounding box` around the wheelchart. The style of this node is given as follows. First, the options `anchor=north,align=center` are given. Thereafter, the style of the key `caption style` is added.

`/wheelchart/caption left={⟨text⟩}` (no default, initially empty)

This key contains the ⟨*text*⟩ which will be placed below left of the wheelchart. The ⟨*text*⟩ is placed in a node. This node is placed at a value determined by the key `caption left sep` below the south west of the `local bounding box` around the wheelchart. The style of this node is given as follows. First, the options `anchor=north west,align=left` are given. Thereafter, the style of the key `caption left style` is added.

`/wheelchart/caption left sep={⟨value⟩}` (no default, initially `0.5`)

The node where the contents of the key `caption left` is placed is at ⟨*value*⟩ below the south west of the `local bounding box` around the wheelchart.

`/wheelchart/caption left style={⟨options⟩}` (style, no default, initially empty)

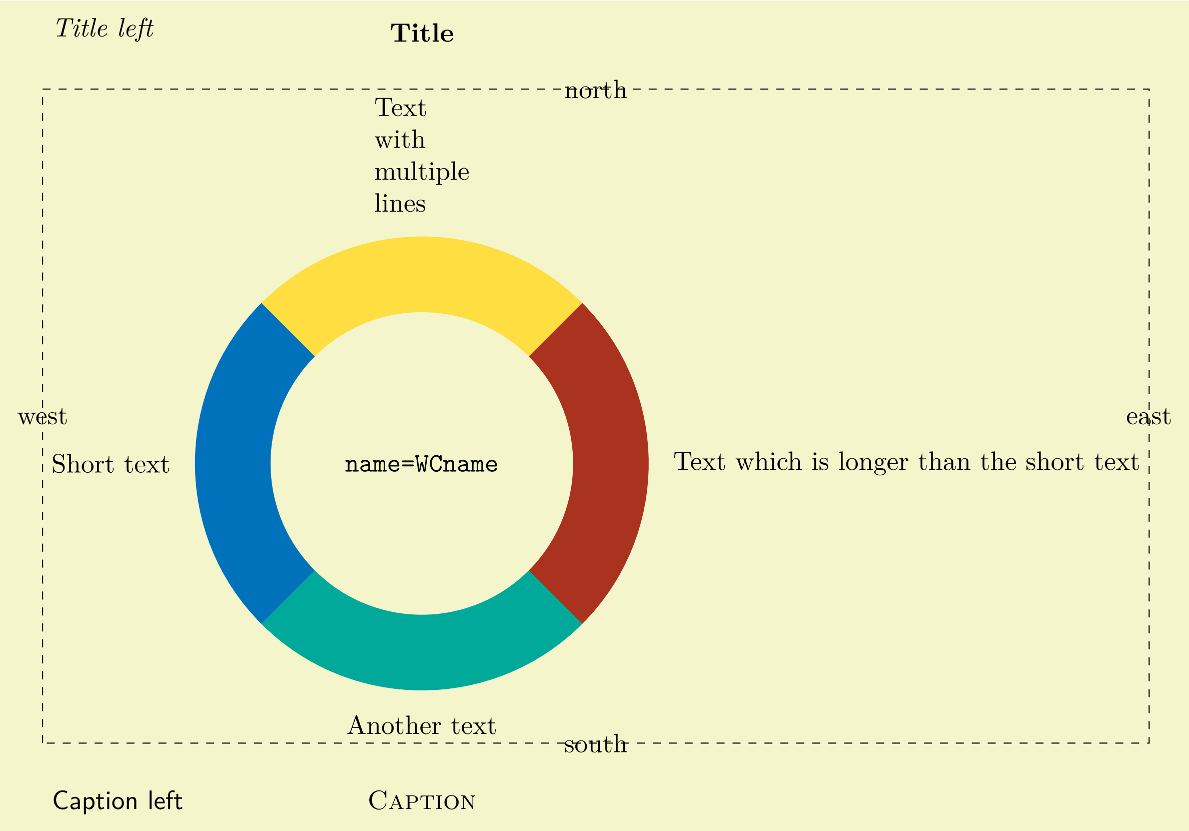
This key accepts a list of keys which will be applied to the node where the contents of the key `caption left` is placed.

`/wheelchart/caption sep={⟨value⟩}` (no default, initially `0.5`)

The y coordinate of the node where the contents of the key `caption` is placed is at ⟨*value*⟩ below the south of the `local bounding box` around the wheelchart.

`/wheelchart/caption style={⟨options⟩}` (style, no default, initially empty)

This key accepts a list of keys which will be applied to the node where the contents of the key `caption` is placed.



```
\begin{tikzpicture}
\wheelchart[
  caption=Caption,
  caption style={font=\scshape},
  caption left=Caption left,
  caption left style={font=\sffamily},
  middle=\texttt{name=WCname},
  name=WCname,
  start half,
  title=Title,
  title style={font=\bfseries},
  title left=Title left,
  title left style={font=\em}
]{%
  1/Goldenrod/Text\\with\\multiple\\lines,
  1/Mahogany/Text which is longer than the short text,
  1/JungleGreen/Another text,
  1/RoyalBlue/Short text%
}
\draw[dashed] (WCname.south west) rectangle (WCname.north east);
\foreach\pos in {north,east,south,west}{
  \node at (WCname.\pos) {\pos};
}
\end{tikzpicture}
```

`/wheelchart/contour={⟨options⟩}` (style, no default, initially empty)

If this key is set then a contour with the style determined by this key will be drawn around the wheelchart. This requires a fixed inner and outer radius for all slices. This key does *not* apply if a plot is used.

`/wheelchart/counterclockwise=⟨boolean⟩` (default `true`, initially `false`)

If true, the wheelchart will be drawn counterclockwise instead of clockwise.

`/wheelchart/data={⟨text⟩}` (no default, initially `\WCvarC`)

This key contains the ⟨*text*⟩ which will be placed at each slice of the wheelchart. This can be suppressed by using `data={}`. The ⟨*text*⟩ is placed in a node. The style of this node is given as follows. First, the anchor is set following Table 1 and Table 3. Then the option `align=left` is added. Thereafter, the style of the key `data style` is added.

`/wheelchart/data angle pos={⟨value⟩}` (no default, initially `0.5`)

`/wheelchart/data angle shift={⟨angle⟩}` (no default, initially `0`)

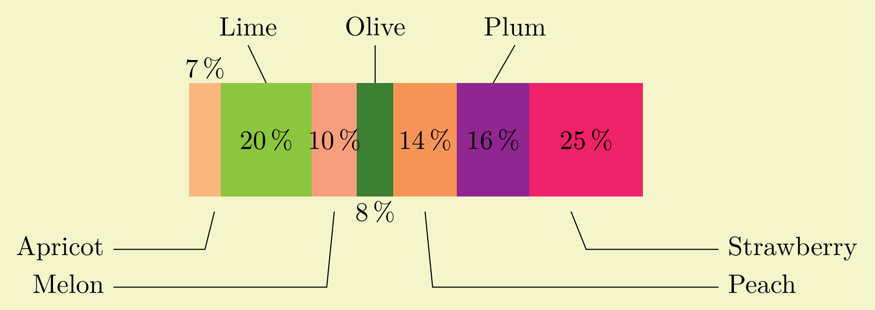
`/wheelchart/data pos={⟨value⟩}` (no default, initially `1`)

`/wheelchart/data sep={⟨value⟩}` (no default, initially `0.2`)

If no plot is used then the position of the contents of the key `data` is determined as described for the commands `\WCangle` and `\WCradius`.

If a plot is used then the position of the contents of the key `data` is determined as follows.

1. The inner plot is evaluated in the point with as angle the convex combination with as parameter the key `data angle pos` of the inner start angle and the inner end angle, added with the key `data angle shift` in degrees (taking into account the key `counterclockwise`) and as radius the inner radius minus the key `data sep`.
2. The outer plot is evaluated in the similar point but using the outer start angle, the outer end angle and the outer radius plus the key `data sep`.
3. If `lines` $\neq 0$ then the values of the keys `lines sep` and `lines` are added to the radii above, in addition to the key `data sep`.
4. The contents of the key `data` is placed at the convex combination with as parameter the key `data pos` of the previous two points.



```
\begin{tikzpicture}
\wheelchart[
  data angle pos{2}=0.3,
  data angle pos{6}=0.8,
  data angle shift{3}=-0.1,
  data angle shift{5}=0.1,
  data pos=\WClistB,
  data sep=0,
  lines{1,2,4,6,7}=0.5,
  lines{3,5}=1,
  lines angle pos{1}=0.8,
  lines angle shift{7}=-0.2,
  lines ext=\WClistA,
  lines ext dir{1,...,3}=left,
  lines ext dir{4,...,7}=right,
  lines ext fixed,
  lines ext fixed left=-1,
  lines ext fixed right=7,
  lines pos=\WClistB,
  lines sep=0.2*\WClistA,
  xbar={6}{1.5},
  WClistA={1,0},
  WClistB={0,1},
  wheel data=\WCperc,
  wheel data pos=0.5,
  wheel data pos{1}=1,
  wheel data pos{4}=0,
  wheel data sep=0.2
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/data style={⟨options⟩}` (style, no default, initially empty)

This key accepts a list of keys which will be applied to the node where the contents of the key `data` is placed.

`/wheelchart/discrete=⟨boolean⟩` (default `true`, initially `false`)

If true then Ti*k*Z pics are placed with the ⟨*code*⟩ determined by the key `discrete pic`. The number of pics is determined by the key `value`. It is required to set the key `discrete space at borders`.

`/wheelchart/discrete factor={⟨value⟩}` (no default, initially `1`)

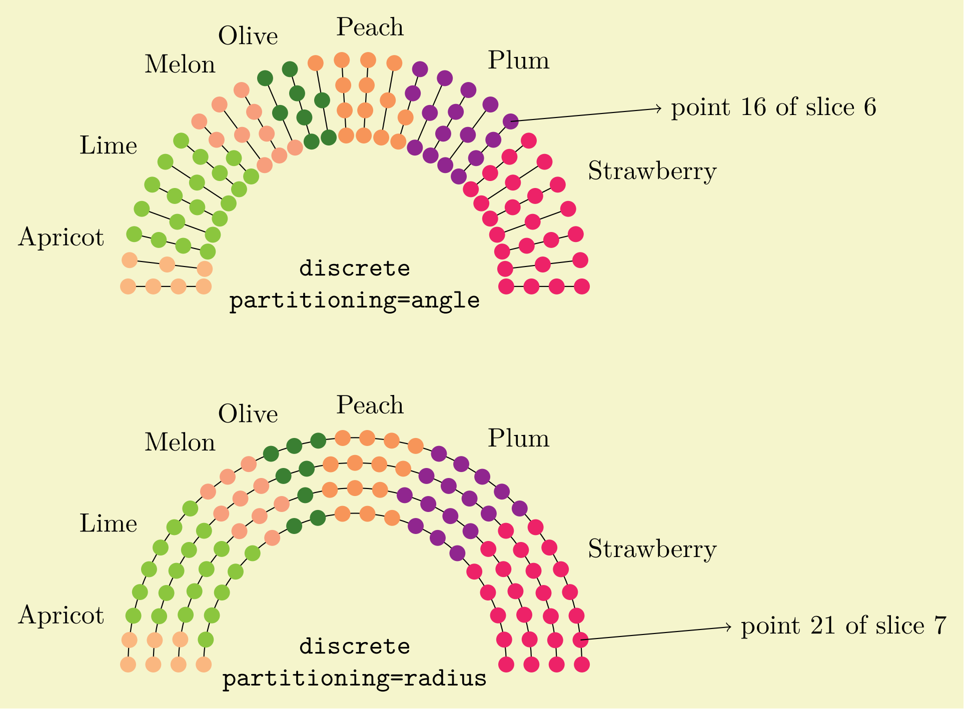
The algorithm to place the Ti*k*Z pics depends on the ⟨*value*⟩. The value `1` is recommended.

`/wheelchart/discrete partitioning=angle|radius` (no default, initially `radius`)

`angle` In this case, the Ti*k*Z pics are placed uniformly with respect to the angle.

`radius` In this case, the Ti*k*Z pics are placed uniformly with respect to the radius.

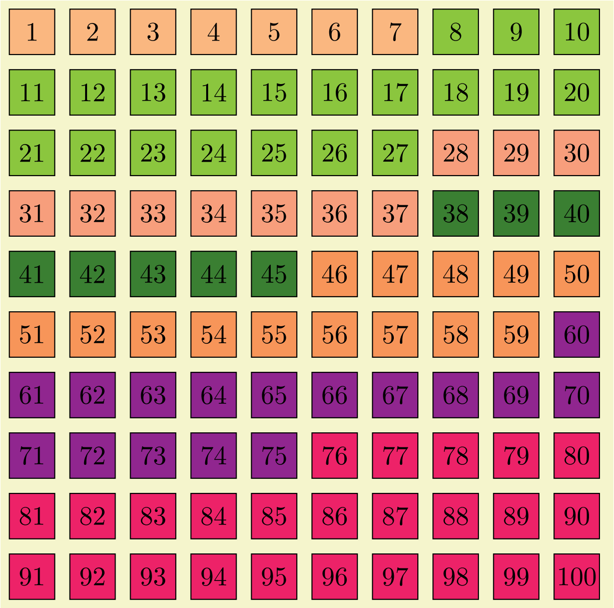
These options are illustrated in the examples below.



```
\begin{tikzpicture}
\pgfkeys{
  /wheelchart,
  discrete,
  discrete pic={\fill (0,0) circle[radius=3pt];},
  discrete space at borders=false,
  middle style={font=\ttfamily},
  start angle=180,
  total angle=180,
  value=\WCvarA/2
}
\foreach\angle in {0,...,27}{
  \draw ({180*(\angle/27)}:2)--({180*(\angle/27)}:3);
}
\wheelchart[
  discrete partitioning=angle,
  middle={discrete\\partitioning=angle}
]{\exampleforthismanual}
\draw[->] (\WCcoordinate[6]{16})--++(5:2) node[right] {point 16 of slice 6};
\foreach\radius in {0,...,3}{
  \draw ({2+\radius/3},-5) arc[start angle=0,end angle=180,radius={2+\radius/3}];
}
\wheelchart[
  at={(0,-5)},
  middle={discrete\\partitioning=radius}
]{\exampleforthismanual}
\draw[->] (\WCcoordinate[7]{21})--++(5:2) node[right] {point 21 of slice 7};
\end{tikzpicture}
```

`/wheelchart/discrete pic={⟨code⟩}` (no default, initially empty)

The ⟨*code*⟩ determines the Ti*k*Z pics.



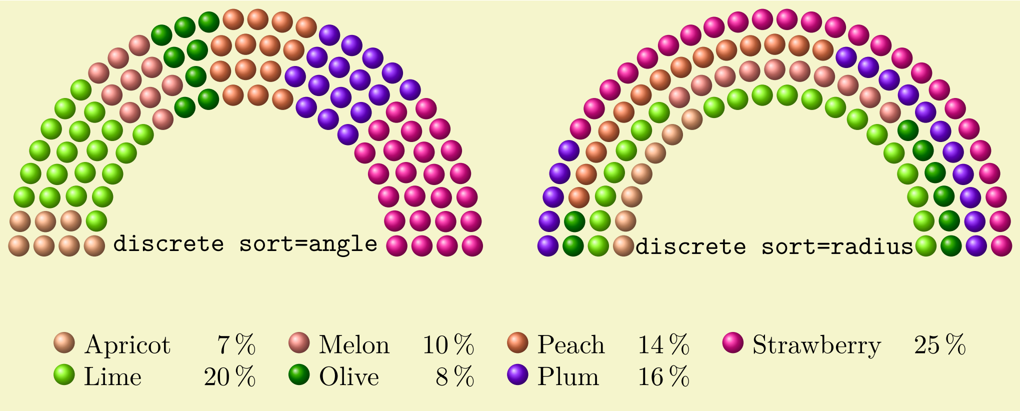
```
\begin{tikzpicture}[yscale=-1]
\wheelchart[
  data=,
  discrete,
  discrete pic={
    \fill[draw=black] (-0.3,-0.3)
      rectangle +(0.6,0.6);
    \node[black] at (0,0)
      {\WCcountdiscrete};
  },
  discrete space at borders,
  value=\WCvarA/2,
  ybar={8}{8}
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/discrete sort=angle|radius` (no default, initially `angle`)

`angle` In this case, the Ti*k*Z pics are ordered with respect to the angle.

`radius` In this case, the Ti*k*Z pics are ordered with respect to the radius.

These options are illustrated in the examples below.

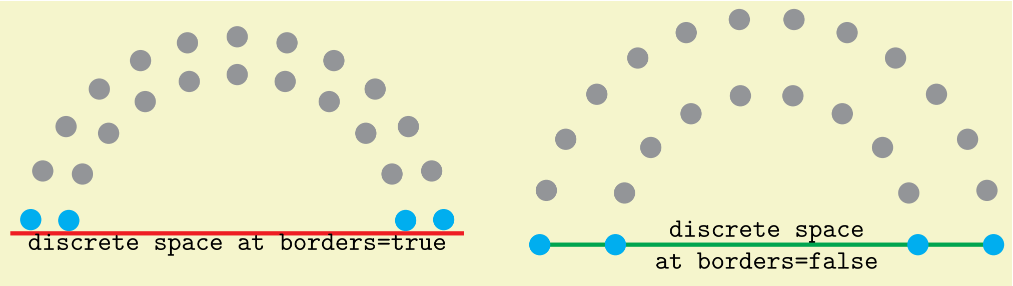


```
\begin{tikzpicture}
\pgfkeys{
  /wheelchart,
  data=,
  discrete,
  discrete pic={\shade[ball color=\WCvarB] (0,0) circle[radius=4pt];},
  discrete space at borders=false,
  middle style={font=\ttfamily},
  start angle=180,
  total angle=180,
  value=\WCvarA/2
}
\wheelchart[
  legend columns=4,
  legend row={\tikz\shade[ball color=\WCvarB] (0,0) circle[radius=4pt]; & \WCvarC & \WCperc},
  legend={\node[anchor=north] at (3.5,-1) {\begin{tabular}{*{4}{l@{ }lr}}\WClegend\end{tabular}};},
  middle={discrete sort=angle}
]{\exampleforthismanual}
\wheelchart[
  at={(7,0)},
  discrete sort=radius,
  middle={discrete sort=radius}
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/discrete space at borders=`⟨*boolean*⟩ (default `true`)

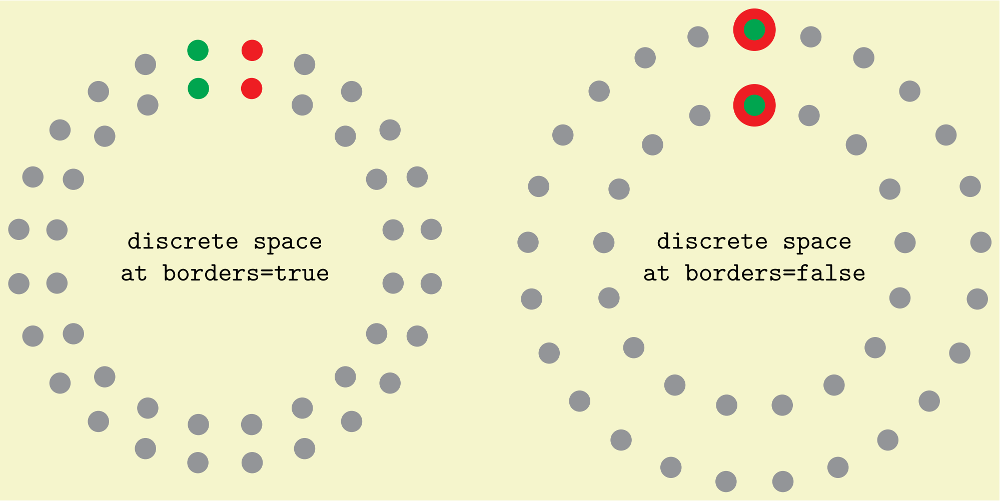
This key determines whether space is left at the begin and end where the Ti*k*Z pics are placed. For example, suppose that 3 Ti*k*Z pics are placed at positions between 0 and 1. If `discrete space at borders` is true then these are placed at the positions $\frac{1}{6}$, $\frac{3}{6}$ and $\frac{5}{6}$. If `discrete space at borders` is false then these are placed at the positions 0, $\frac{1}{2}$ and 1.

This key deliberately has no initial value in order to force awareness of the consequences of the settings of this key. In the example below, the cyan Ti*k*Z pics are aligned if `discrete space at borders` is false while this is *not* the case if `discrete space at borders` is true.



```
\begin{tikzpicture}
\pgfkeys{
  /wheelchart,
  discrete,
  discrete pic={\fill (0,0) circle[radius=4pt];},
  middle style={font=\ttfamily},
  start angle=180,
  total angle=180
}
\draw[Red,ultra thick] (-3,0.15)--+(6,0);
\wheelchart[
  discrete space at borders,
  middle={discrete space at borders=true}
]{2/Cyan/,20/Gray/,2/Cyan/}
\draw[Green,ultra thick] (4,0)--+(6,0);
\wheelchart[
  at={(7,0)},
  discrete space at borders=false,
  middle={discrete space\\at borders=false}
]{2/Cyan/,20/Gray/,2/Cyan/}
\end{tikzpicture}
```

In the example below, the red and green Ti*k*Z pics overlap if `discrete space at borders` is false while this is *not* the case if `discrete space at borders` is true.



```
\begin{tikzpicture}
\pgfkeys{
  /wheelchart,
  discrete,
  discrete pic={\fill (0,0) circle[radius=\WClistradius pt];},
  middle style={font=\ttfamily}
}
\wheelchart[
  discrete space at borders,
  middle={discrete space\\at borders=true},
  WClistradius=4
]{2/Red/,40/Gray/,2/Green/}
\wheelchart[
  at={(7,0)},
  discrete space at borders=false,
  middle={discrete space\\at borders=false},
  WClistradius={8,4,4}
]{2/Red/,40/Gray/,2/Green/}
\end{tikzpicture}
```

`/wheelchart/domain={`⟨*start*⟩`}:{`⟨*end*⟩`}` (no default)

This key sets `counterclockwise`, `start angle` to ⟨*start*⟩ and `total angle` to ⟨*end*⟩ − ⟨*start*⟩.

`/wheelchart/etoc code={`⟨*code*⟩`}` (no default, initially `\tableofcontents`)

The ⟨*code*⟩ will be executed to build the ⟨*wheelchart data*⟩ if the key `etoc level` is used.

`/wheelchart/etoc count total pages={⟨number⟩}` (no default, initially `0`)

If the key `etoc level` is used then the number of pages of the last section depends on ⟨*number*⟩ which can for example represent the total number of pages in the document or the number of pages before the start of the Appendix or the Index. For example, `etoc count total pages=\totalpages` can be used. To provide the command `\totalpages`, this requires `\usepackage[page]{totalcount}`, which should normally be loaded *before* the package `wheelchart` to give a correct result.

`/wheelchart/etoc level={⟨level⟩}` (no default)

If this key is used then the ⟨*wheelchart data*⟩ of the command `\wheelchart` can be left empty and is defined to match the sections of the level defined by ⟨*level*⟩. Here, `\WCetocthelinkedname` corresponds to `\etocthelinkedname`, `\WCetocthelinkednumber` to `\etocthelinkednumber`, `\WCetocthelinkedpage` to `\etocthelinkedpage`, `\WCetocthename` to `\etocthename`, `\WCetocthenumber` to `\etocthenumber` and `\WCetocthepage` to `\etocthepage`. The package `etoc` is required to provide these commands. Furthermore, `\WCetocthenumberofpages` corresponds to the number of pages of the current section. For the last section, this depends on the value of the key `etoc count total pages`.

`/wheelchart/etoc name={⟨name⟩}` (no default, initially empty)

The resulting ⟨*wheelchart data*⟩ from the key `etoc level` is stored globally and can be reused later with the key `etoc use name`.

`/wheelchart/etoc use name={⟨name⟩}` (no default)

If this key is used then the ⟨*wheelchart data*⟩ is reused from where `etoc name` has the same ⟨*name*⟩.

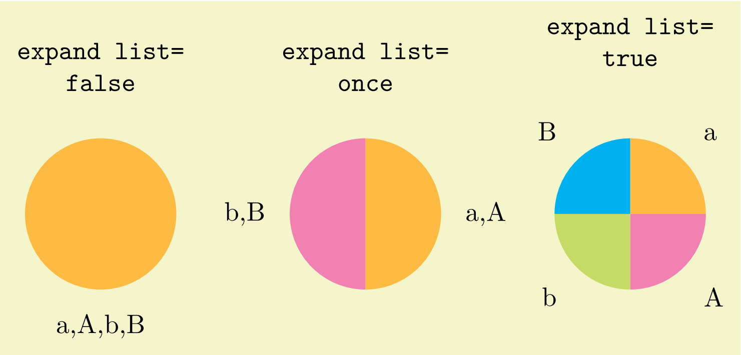
`/wheelchart/expand list=false|once|true` (no default, initially `once`)

`false` In this case, the ⟨*wheelchart data*⟩ of the command `\wheelchart` will not be expanded.

`once` In this case, the ⟨*wheelchart data*⟩ of the command `\wheelchart` will be expanded once.

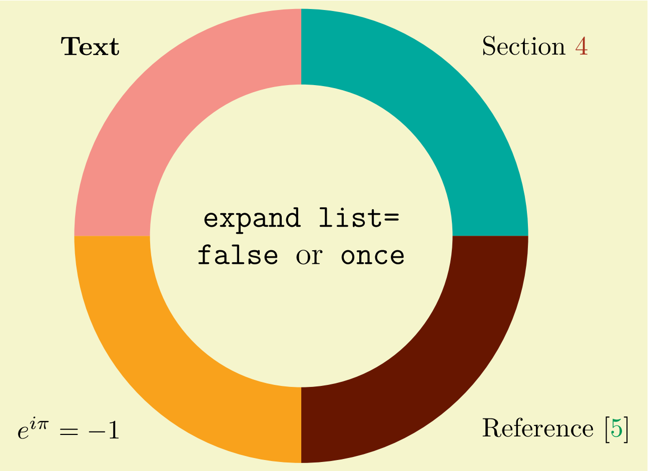
`true` In this case, the ⟨*wheelchart data*⟩ of the command `\wheelchart` will be fully expanded.

The following example illustrates the difference between the possible values of the key `expand list`.



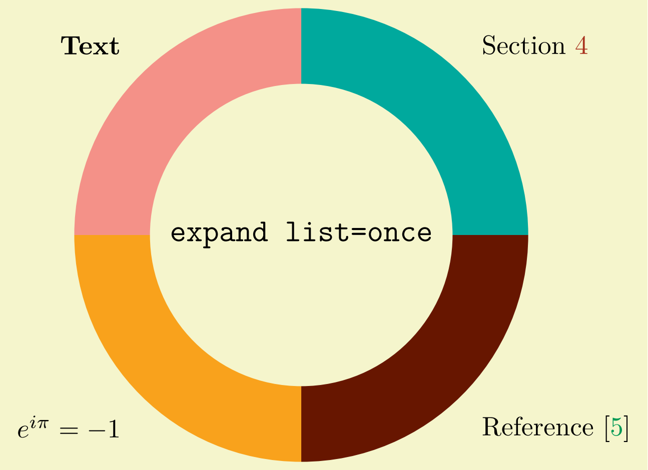
```
\begin{tikzpicture}
\def\WClistA{a,A}
\def\WClistB{b,B}
\def\WCdata{\WClistA,\WClistB}
\foreach\expandlist [count=\n] in
  {false,once,true}{
  \wheelchart[
    at={({3.5*\n},0)},
    data=\WCvarA,
    expand list=\expandlist,
    radius={0}{1},
    slices style{list}={
      Dandelion,CarnationPink,
      SpringGreen,ProcessBlue
    },
    title={expand list=\\\expandlist},
    title style={font=\ttfamily},
    value=1
  ]{\WCdata}
}
\end{tikzpicture}
```

The initial setting `expand list=once` works in most situations, even when commands such as `\ref`, `\cite` and `\textbf` are used such as in the example below.



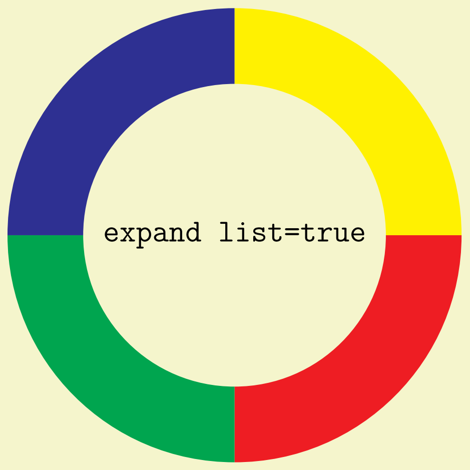
```
\begin{tikzpicture}
\wheelchart[
  %expand list=false,%false also works
  %expand list=true,%true doesn't work
  middle={expand list=\\false %
    {\normalfont or} once},
  middle style={font=\large\ttfamily}
]{%
  1/Emerald/Section \ref{Keys},
  1/Sepia/Reference \cite{TtTaPGFp},
  1/YellowOrange/{$e^{i\pi}=-1$},
  1/Salmon/\textbf{Text}%
}
\end{tikzpicture}
```

In the following example, the ⟨*wheelchart data*⟩ from the previous example is stored in a macro. In this case, we have to use the initial setting `expand list=once`.



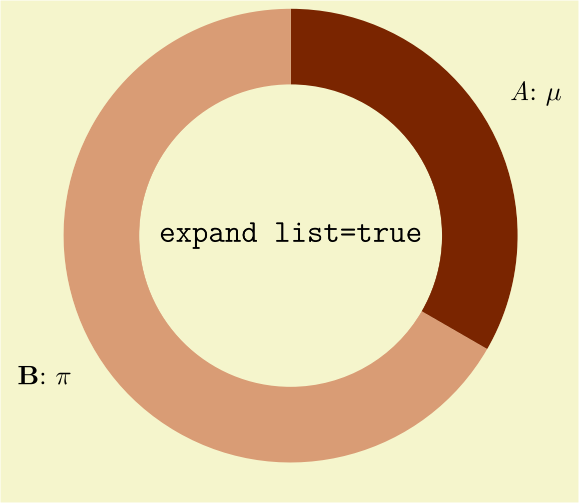
```
\begin{tikzpicture}
\def\WClist{%
  1/Emerald/Section \ref{Keys},
  1/Sepia/Reference \cite{TtTaPGFp},
  1/YellowOrange/{$e^{i\pi}=-1$},
  1/Salmon/\textbf{Text}%
}
\wheelchart[
  %expand list=false,
  %expand list=true,
  %false and true do not work
  middle={expand list=once},
  middle style={font=\large\ttfamily}
]{\WClist}
\end{tikzpicture}
```

In the example below, we have to use `expand list=true`.



```
\begin{tikzpicture}
\def\WCcolorsA{Yellow,Red}
\def\WCcolorsB{Green,Blue}
\wheelchart[
  data=,
  expand list=true,%false and once
                   %do not work
  middle={expand list=true},
  middle style={font=\large\ttfamily},
  slices style=\WCvarA,
  value=1
]{\WCcolorsA,\WCcolorsB}
\end{tikzpicture}
```

In the example below, we have to use `expand list=true` and the command `\expandonce` from the package `etoolbox`.



```
\usepackage {etoolbox}
\begin{tikzpicture}
\def\WCsliceA{1/Brown/{\emph{A}: $\mu$}}
\def\WCsliceAfinal{\expandonce\WCsliceA}
\def\WCsliceB{2/Tan/{\textbf{B}: $\pi$}}
\def\WCsliceBfinal{\expandonce\WCsliceB}
\wheelchart[
  expand list=true,%false and once
                   %do not work
  middle={expand list=true},
  middle style={font=\large\ttfamily}
]{\WCsliceAfinal,\WCsliceBfinal}
%\WCsliceA and \WCsliceB do not work
\end{tikzpicture}
```

`/wheelchart/expand list items=false|once|true` (no default, initially `false`)

This key is similar to the key `expand list` but applies to the items in the ⟨*wheelchart data*⟩ of the command `\wheelchart` which correspond to a slice of the wheelchart.

`expand list items` ; `false`: a/b/c/d ; `once`: a/b ; `true`: a

```
\def\WClistA{a/b}%
\def\WClistB{c/d}%
\def\WCdata{\WClistA/\WClistB}%
\texttt{expand list items}%
\foreach\expandlistitems in
    {false,once,true}{%
  \wheelchart[
    expand list=false,
    expand list items=\expandlistitems,
    legend={; \texttt{\expandlistitems}:
        \WCvarA},
    legend only,
    value=1
  ]{\WCdata}%
}
```

`/wheelchart/explode={`⟨*value*⟩`}` (default `0.2, initially 0`)

This key will shift the slices of the wheelchart with ⟨*value*⟩ with respect to the center of the wheelchart.

`/wheelchart/for loop end={`⟨*code*⟩`}` (no default, initially empty)

The slices of the wheelchart, the wheel lines determined by the key `wheel lines` and the different kinds of data are placed in for loops. If the key `for loop end` is set then the ⟨*code*⟩ given to this key will be executed at the end of the body of these for loops.

`/wheelchart/for loop start={`⟨*code*⟩`}` (no default, initially empty)

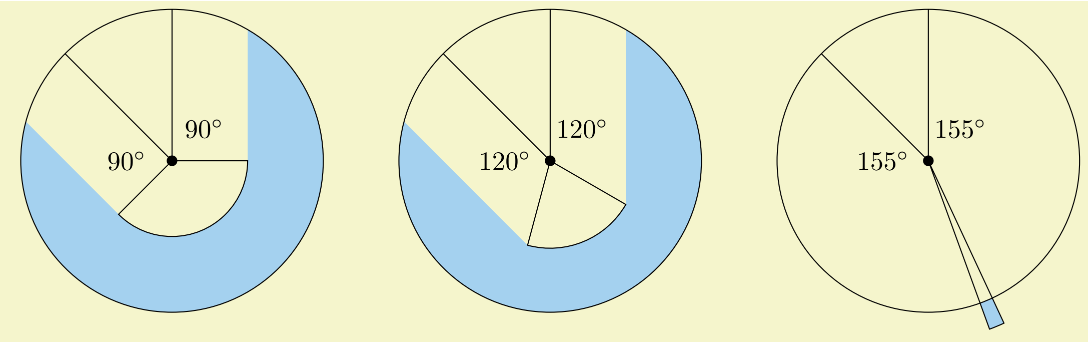
This key is similar to the key `for loop end` but the ⟨*code*⟩ given to this key will be executed at the start of the body of the for loops.

`/wheelchart/gap={`⟨*value*⟩`}` (default `0.05, initially 0`)

The ⟨*value*⟩ of this key defines half the distance between two slices of the wheelchart. This key does *not* apply if a plot is used.

`/wheelchart/gap max angle={`⟨*angle*⟩`}` (no default, initially `180`)

If the value of the key `gap` is too large then a slice can partly disappear such as for example below when `gap max angle` is 155°. The ⟨*angle*⟩ of the key `gap max angle` determines the inner arc of the slice as illustrated in the examples below.

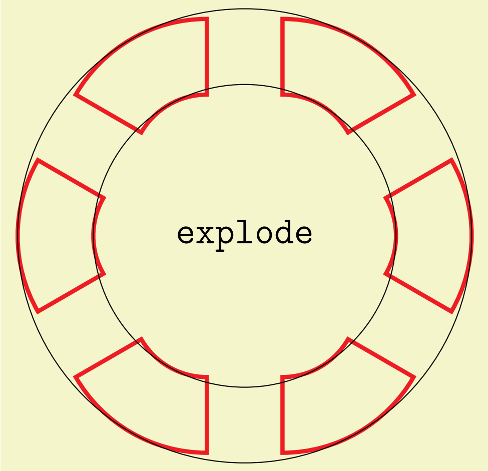


```
\usepackage {siunitx}
\begin{tikzpicture}
\foreach\gapmaxangle [count=\n] in {90,120,155}{
  \begin{scope}[shift={({5*\n},0)}]
    \wheelchart[
      gap=1,
      gap max angle=\gapmaxangle,
      radius={0}{2},
      total angle=315
    ]{1/CornflowerBlue!50/}
    \fill (0,0) circle[radius=2pt];
    \draw (0,0) circle[radius=2];
    \draw (135:2)--(0:0)--(90:2);
    \draw (0:0)--({135+\gapmaxangle}:{1/sin(\gapmaxangle)}) arc[start angle={135+\gapmaxangle},
      end angle={450-\gapmaxangle},radius={1/sin(\gapmaxangle)}]--cycle;
    \node at (45:0.6) {\ang{\gapmaxangle}};
    \node at (180:0.6) {\ang{\gapmaxangle}};
  \end{scope}
}
\end{tikzpicture}
```

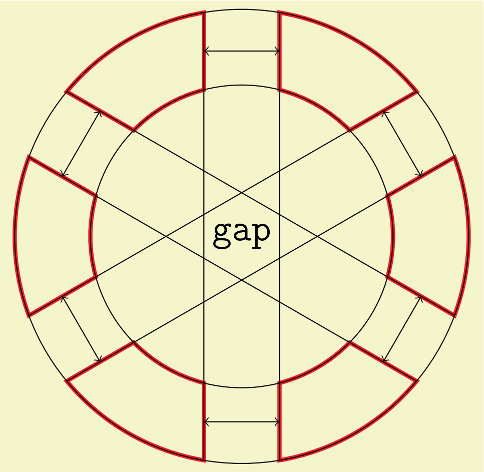
/wheelchart/gap polar={⟨*value*⟩} (default 1, initially 0)

The ⟨*value*⟩ of this key defines half the polar gap in degrees between two slices of the wheelchart.

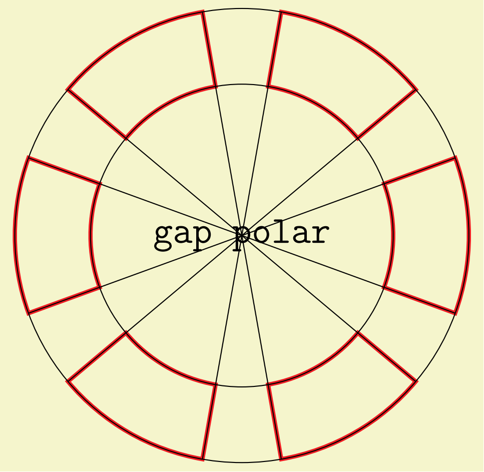
Note the difference between the keys explode, gap and gap polar. This is illustrated in the examples below.



```
\begin{tikzpicture}
\wheelchart[
  explode=1,
  middle={\Large\texttt{explode}},
  radius={1}{2},
  slices style={
    draw=Red,
    fill=none,
    ultra thick
  },
  total count=6
]{}
\draw (0,0) circle[radius=2];
\draw (0,0) circle[radius=3];
\end{tikzpicture}
```



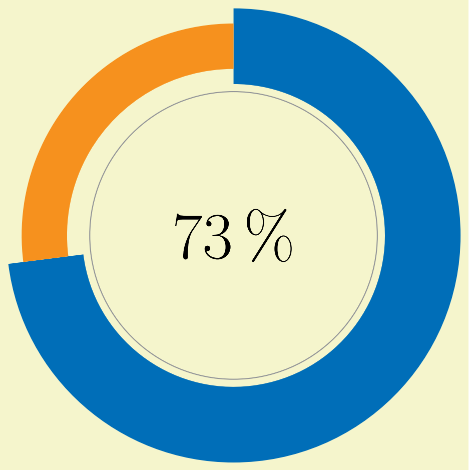
```
\begin{tikzpicture}
\wheelchart[
  gap=0.5,
  legend entry={
    \draw (\WCcoordinate{outer end})
      --(\WCcoordinate[\WCcount -2]
        {outer start});
    \draw[<->] (\WCpoint{1}{0}{0.5}{0})
      --(\WCpoint[\WCcount +1]
        {0}{0}{0.5}{0});
  },
  middle={\Large\texttt{gap}},
  slices style={
    draw=Red,
    fill=none,
    ultra thick
  },
  total count=6
]{}
\draw (0,0) circle[radius=2];
\draw (0,0) circle[radius=3];
\end{tikzpicture}
```



```
\begin{tikzpicture}
\wheelchart[
  gap polar=10,
  legend entry{1,2,3}={
    \draw (\WCcoordinate{outer start})
      --(\WCcoordinate[\WCcount +3]
        {outer start});
    \draw (\WCcoordinate{outer end})
      --(\WCcoordinate[\WCcount +3]
        {outer end});
  },
  middle={\Large\texttt{gap polar}},
  slices style={
    draw=Red,
    fill=none,
    ultra thick
  },
  total count=6
]{}
\draw (0,0) circle[radius=2];
\draw (0,0) circle[radius=3];
\end{tikzpicture}
```

/wheelchart/gap radius={⟨*value*⟩} (default 0.05, initially 0)

The ⟨*value*⟩ of this key will be added to inner radius and substracted from outer radius.



```
\usepackage {siunitx}
\begin{tikzpicture}
\def\n{73}
\wheelchart[
  data=,
  gap radius=\WCvarC,
  middle={\Huge\qty{\n}{\percent}}
]{%
  \n/NavyBlue/0,
  {100-\n}/BurntOrange/0.2%
}
\draw[Gray] (0,0) circle[radius=1.9];
\end{tikzpicture}
```

/wheelchart/header={⟨*list*⟩} (no default)

The items in the ⟨*list*⟩ determine the names in the macros `\`⟨*prefix*⟩⟨*name*⟩.

`/wheelchart/header prefix={`⟨*prefix*⟩`}` (no default, initially `WC`)

The ⟨*prefix*⟩ is used in the macros `\`⟨*prefix*⟩⟨*name*⟩.

`/wheelchart/inner data={`⟨*text*⟩`}` (no default, initially empty)

This key contains the ⟨*text*⟩ which will be placed at each slice of the wheelchart. The ⟨*text*⟩ is placed in a node. The style of this node is given as follows. First, the option `align=left` is given. Thereafter, the style of the key `inner data style` is added.

`/wheelchart/inner data angle pos={`⟨*value*⟩`}` (no default, initially `0.5`)

`/wheelchart/inner data angle shift={`⟨*angle*⟩`}` (no default, initially `0`)

`/wheelchart/inner data pos={`⟨*value*⟩`}` (no default, initially `0`)

`/wheelchart/inner data sep={`⟨*value*⟩`}` (no default, initially `0.2`)

These keys determine the position of the contents of the key `inner data` similar as the corresponding keys for the key `data`. No lines are drawn for the inner data.

`/wheelchart/inner data style={`⟨*options*⟩`}` (style, no default, initially empty)

This key accepts a list of keys which will be applied to the node where the contents of the key `inner data` is placed.

`/wheelchart/inner plot={`⟨*code*⟩`}` (no default)

The ⟨*code*⟩ is a coordinate definition which will be used for the inner parts of the slices of the wheelchart. In the ⟨*code*⟩, `#1` and `#2` can be used where `#1` corresponds to the angle and `#2` corresponds to the radius. For example, a circle can be obtained with `inner plot={{#1}:{#2}}`.

`/wheelchart/inner plot style={`⟨*options*⟩`}` (style, no default, initially empty)

This key accepts a list of keys which will be applied to the plot determined by the key `inner plot`.

`/wheelchart/inner radius={`⟨*value*⟩`}` (no default, initially `2`)

The ⟨*value*⟩ of this key defines the inner radius of the wheelchart.

`/wheelchart/legend={`⟨*code*⟩`}` (no default, initially empty)

The ⟨*code*⟩ given to this key will be executed at the end of the command `\wheelchart`.

`/wheelchart/legend columns={`⟨*number*⟩`}` (no default, initially `1`)

If the key `legend row` is used then the maximum number of times that the ⟨*code*⟩ given to the key `legend row` appears on one row is determined by ⟨*number*⟩. The environment (for example `tabular`, `tabularx` from the package `tabularx`, `tabulary` from the package `tabulary` or `tblr` from the package `tabularray`) which contains the macro `\WClegend` needs to have a suitable column specification according with ⟨*number*⟩ and the key `legend row`.

`/wheelchart/legend entry={`⟨*code*⟩`}` (no default, initially empty)

The ⟨*code*⟩ given to this key will be executed for each slice of the wheelchart.

`/wheelchart/legend only=`⟨*boolean*⟩ (default `true, initially false`)

If true then only the legend is constructed. This does *not* apply to the key `legend entry`.

In this case it is *not* necessary to place the command `\wheelchart` in a `tikzpicture` environment.

11 animals from the package **tikzlings**

	bear		elephant		penguin
	bee		koala		snowman
	bug		owl		squirrel
	cat		panda		

```
\usepackage {tikzlings}
\wheelchart[
  header={animal,accessory},
  legend columns=3,
  legend only,
  legend row={\tikz[scale=0.3]{
    \csname \WCanimal\endcsname[
      signpost=\WCcount,
      \WCaccessory
    ]} & \WCanimal},
  legend={%
    \begin{tabular}{*{3}{cl}}
    \multicolumn{6}{c}{%
      \WCtotalcount{} animals%
      from the package%
      \texttt{tikzlings}}\\\hline
    \WClegend\hline
    \end{tabular}%
  },
  separator columns={{ }},
  separator rows=;,
  value=1
]{%
  bear basket;
  bee book;
  bug chef;
  cat crown;
  elephant football;
  koala handbag;
  owl hat;
  panda icecream;
  penguin milkshake;
  snowman santa;
  squirrel shovel%
}
```

`/wheelchart/legend row={⟨code⟩}` (no default)

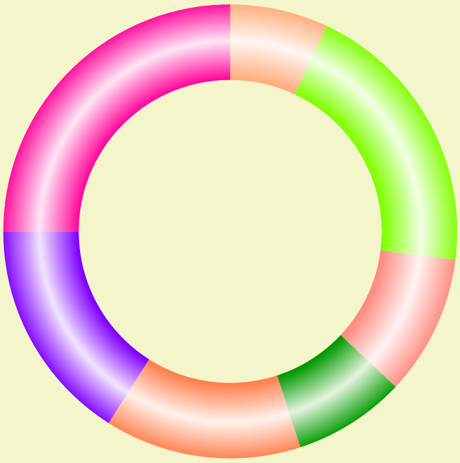
If this key is set then a legend consisting of rows for an environment such as `tabular`, `tabularx` from the package `tabularx`, `tabulary` from the package `tabulary` or `tblr` from the package `tabularray` is constructed using the ⟨*code*⟩ for each slice of the wheelchart.

If a `tblr` environment from the package `tabularray` is used then the option `expand=\WClegend` needs to be given to this `tblr` environment and `\UseTblrLibrary{counter}` is required.

The maximum number of times that the ⟨*code*⟩ appears on one row is determined by the key `legend columns`.

The code automatically inserts `&` and `\\` after the ⟨*code*⟩ if necessary.

The result is stored in the macro `\WClegend`.

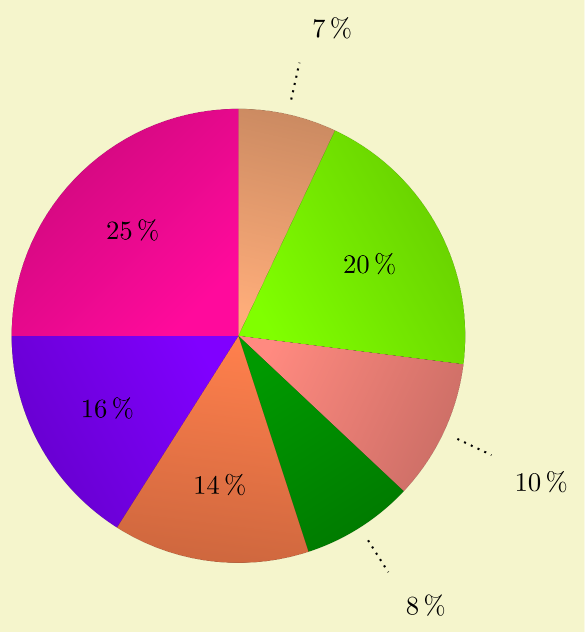


	Fruit	Value	Percentage	Vitamins
🟧	Apricot	14	7 %	A, B, C, E, K
🟩	Lime	40	20 %	B, C
🟧	Melon	20	10 %	A, C
🟩	Olive	16	8 %	A, B, E, K
🟧	Peach	28	14 %	A, B, C, E, K
🟪	Plum	32	16 %	A, B, C, E, K
🟥	Strawberry	50	25 %	B, C, E, K
	Total	200		

```
\usepackage {tabularray} \UseTblrLibrary {counter,siunitx}
\begin{tikzpicture}
\wheelchart[
  after slices={
    \pgfdeclareradialshading{WCshading}{\pgfpoint{0cm}{0cm}}{
      color(0bp)=(\WCvarB);
      color(16.66666bp)=(\WCvarB);%2/3 * 25bp
      color(20.83333bp)=(\WCvarB!10);%2.5/3 * 25bp
      color(25bp)=(\WCvarB);
      color(50bp)=(\WCvarB)
    }
    \shade[even odd rule,shading=WCshading] (0,0) circle[radius=3] circle[radius=2];
  },
  data=,
  legend row={\tikz\fill[\WCvarB] (0,0) rectangle (0.3,0.3);%
     & \WCvarC & \WCvarA & \WCpercentagerounded & \WCvarD},
  legend={
    \node[anchor=west] at (3.5,0) {%
      \begin{tblr}[expand=\WClegend]{
        colspec={llS[table-format=3.0]S[table-format=2.0{\,\unit{\percent}}]l},
        column{1}={rightsep=0pt,appto={\ }},
        column{2}={leftsep=0pt},
        cell{2-Y}{4}={appto={\,\unit{\percent}}},
        row{1}=guard
      }
       & Fruit & Value & Percentage & Vitamins\\\hline
      \WClegend\hline
       & \textbf{Total} & \WCtotalnum & & \\
      \end{tblr}%
    };
  },
  slices style={
    fill=none,
    clip
  }
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/lines={⟨value⟩}` (default `1`, initially `0`)

The ⟨*value*⟩ is used in the positioning of the contents of the key `data`. The end point of the lines is determined similarly but without the key `data sep`.



```
\usepackage {siunitx}
\begin{tikzpicture}
\def\WCtest#1#2{%
  \pgfmathparse{
    \WCpercentage>10?"#1":"#2"
  }%
  \pgfmathresult%
}
\wheelchart[
  data=\WCtest{}{\WCperc},
  lines={\WCpercentage>10?0:0.5},
  lines style={dotted,thick},
  pie,
  slices style={
    bottom color=\WCvarB,
    top color=\WCvarB!80!black,
    shading angle=\WCmidangle-90
  },
  wheel data=\WCtest{\WCperc}{}
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/lines angle pos={⟨value⟩}` (no default, initially `0.5`)

/wheelchart/lines angle shift={⟨*angle*⟩} (no default, initially 0)

These keys are similar to the corresponding keys for data but determine the start point of the lines.

/wheelchart/lines ext={⟨*value*⟩} (default 0.5, initially 0)

If the ⟨*value*⟩ of this key is nonzero and lines ext fixed is false then the lines between the wheelchart and the contents of the key data will be extended horizontally with a length defined by ⟨*value*⟩.

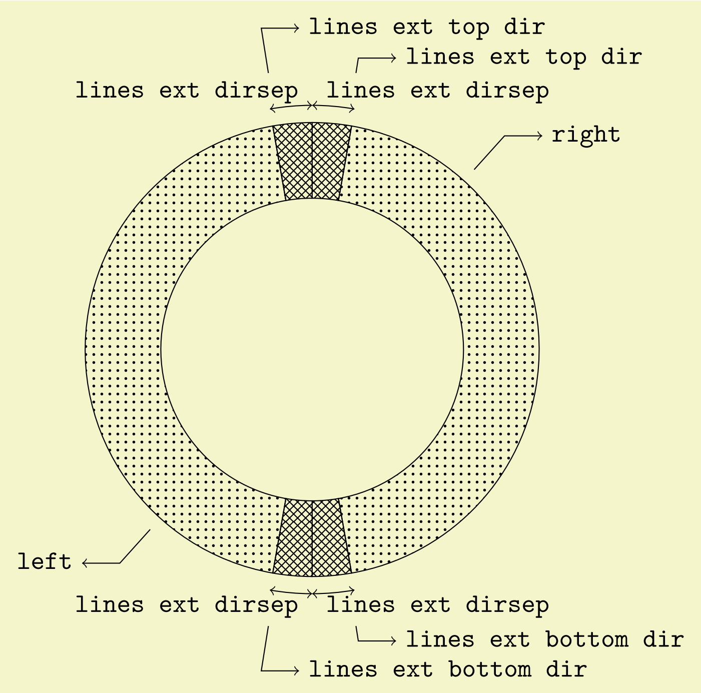
/wheelchart/lines ext bottom dir=left|right (no default, initially right)

/wheelchart/lines ext dir=left|right (no default)

The default direction in which the lines between the wheelchart and the contents of the key data will be extended horizontally if lines ext $\neq 0$ is determined by Table 2 and illustrated in the following example. This can be overruled by giving an explicit value to this key. Note that rounding errors can occur in the computation of the angle which is used to determine the default direction according to Table 2.

Angle (up to rounding errors)	The direction in which the lines between the wheelchart and the contents of the key data will be extended horizontally if lines ext $\neq 0$ and if the key lines ext dir is not used
$[0, 90 -$ lines ext dirsep$[$	right
$[90 -$ lines ext dirsep$, 90 +$ lines ext dirsep$]$	value of the key lines ext top dir
$]90 +$ lines ext dirsep$, 270 -$ lines ext dirsep$[$	left
$[270 -$ lines ext dirsep$, 270 +$ lines ext dirsep$]$	value of the key lines ext bottom dir
$]270 +$ lines ext dirsep$, 360[$	right

Table 2: The direction in which the lines between the wheelchart and the contents of the key data will be extended horizontally if lines ext $\neq 0$ and if the key lines ext dir is not used.



```
\usetikzlibrary {patterns}
\begin{tikzpicture}[font=\ttfamily]
\def\WClinesextdirsep{10}
\wheelchart[
  data{1,6}=lines ext top dir,
  data{2}=right,
  data{3,4}=lines ext bottom dir,
  data{5}=left,
  data angle pos=\WClistdap,
  data sep=0,
  inner data angle pos{1,4}=0.1,
  inner data angle pos{3,6}=0.9,
  inner data pos=1,
  inner data sep=0.4,
  inner data style={anchor=\WClistia},
  lines=0.6,
  lines{1,3}=0.2,
  lines angle pos=\WClistdap,
  lines ext,
  lines ext dirsep=\WClinesextdirsep,
  lines sep{list}={0.7,0.2,0.7},
  lines style=->,
  slice{1,3,4,6}={
    arc=<->,
    inner data=lines ext dirsep,
    value=\WClinesextdirsep
  },
  slices style={
    draw,
    pattern=\WClistpattern
  },
  total count=6,
  value{2,5}=180-2*\WClinesextdirsep,
  WClistdap={0.9,0.2,0.1},
  WClistia={west,east},
  WClistpattern={
    crosshatch,dots,crosshatch
  }
]{}
\end{tikzpicture}
```

`/wheelchart/lines ext dirsep={⟨angle⟩}` (no default, initially `0`)

This key determines half the angle in degrees of the segment to which the keys `lines ext bottom dir` and `lines ext top dir` apply.

`/wheelchart/lines ext fixed=⟨boolean⟩` (default `true`, initially `false`)

If true, the line between the wheelchart and the contents of the key `data` will be extended horizontally till the x coordinate determined by the keys `lines ext fixed left` and `lines ext fixed right`.

`/wheelchart/lines ext fixed left={⟨value⟩}` (no default)

`/wheelchart/lines ext fixed right={⟨value⟩}` (no default)

If `lines ext fixed` is true, the lines are extended horizontally initially to the right till the x coordinate `outer radius` + `lines sep` + `lines` + `lines ext` and to the left till the opposite of this x coordinate. This can be overruled by giving an explicit value to the key `lines ext fixed left` and/or `lines ext fixed right`.

`/wheelchart/lines ext left anchor={⟨anchor⟩}` (no default, initially `mid east`)

`/wheelchart/lines ext right anchor={⟨anchor⟩}` (no default, initially `mid west`)

The direction in which the lines between the wheelchart and the contents of the key `data` will be extended horizontally if `lines ext` $\neq 0$	Anchor of the key `data`
left	value of the key `lines ext left anchor`
right	value of the key `lines ext right anchor`

Table 3: Anchor of the key `data` in the case that `lines ext` $\neq 0$.

`/wheelchart/lines ext top dir=left|right` (no default, initially `right`)

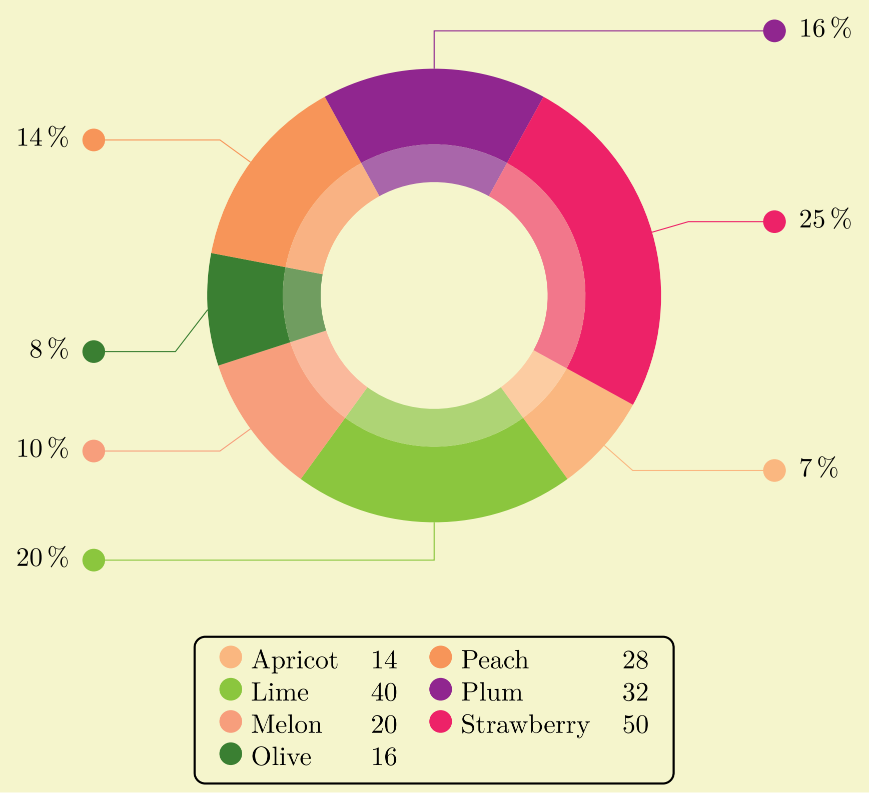
`/wheelchart/lines pos={⟨value⟩}` (no default, initially `1`)

`/wheelchart/lines sep={⟨value⟩}` (no default, initially `0.2`)

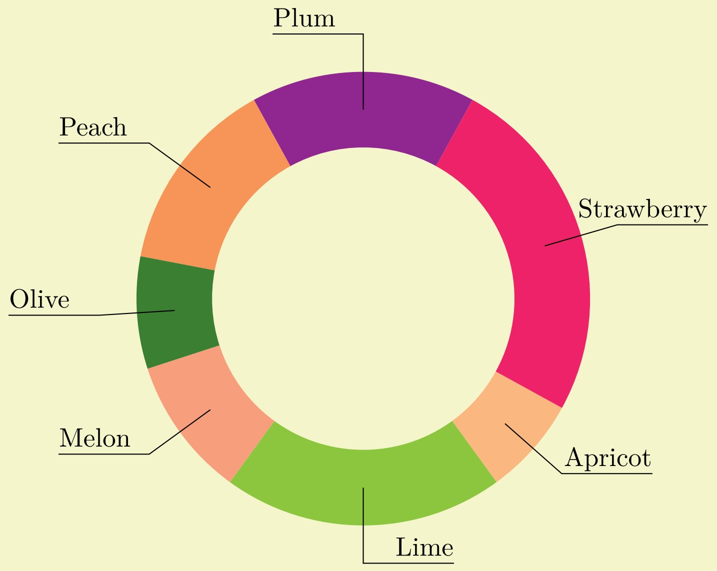
These keys are similar to the corresponding keys for `data` but determine the start point of the lines.

`/wheelchart/lines style={⟨options⟩}` (style, no default, initially empty)

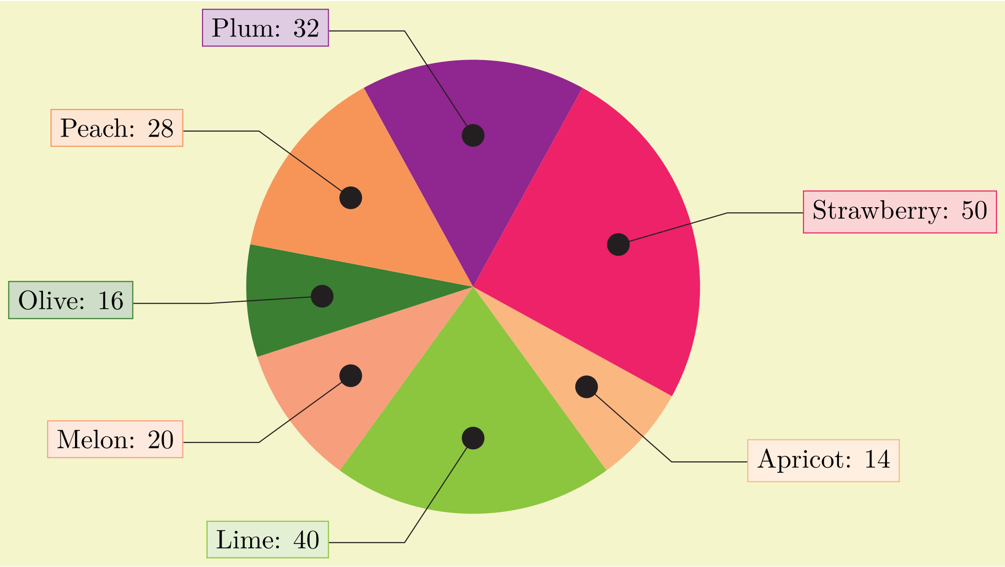
This key accepts a list of keys which will be applied to the lines drawn by the key `lines`.



```
\usepackage {siunitx} \usetikzlibrary {decorations.markings}
\begin{tikzpicture}
\wheelchart[
  data=\WCperc,
  data angle pos{4}=0.2,
%  data style={outer xsep=4pt},
  legend columns=2,
  legend row={\tikz\fill[\WCvarB] (0,0) circle[radius=0.15]; & \WCvarC & $\WCvarA$},
  legend={
    \node[anchor=north,draw,rounded corners,thick] at (0,-4.5) {%
      \begin{tabular}{*{2}{l@{ }lr}}%
      \WClegend%
      \end{tabular}%
    };
  },
  lines=0.5,
  lines ext=1,
  lines ext bottom dir=left,
  lines ext dirsep=1,
  lines ext fixed,
  lines ext top dir=right,
  lines sep=0,
  lines style={
    \WCvarB,
    postaction=decorate,
    decoration={
      markings,
      mark=at position 1 with {\fill[\WCvarB] (0,0) circle[radius=0.15];}
    }
  },
  start angle=331.2
]{\exampleforthismanual}
\wheelchart[
  data=,
  radius={1.5}{2},
  slices style=\WCvarB!70,
  start angle=331.2
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\wheelchart[
  data sep=0,
  data style={
    inner sep=0pt,
    shift={(0,0.1)}
  },
  lines=0.5,
  lines ext=1.2,
  lines ext bottom dir=right,
  lines ext dirsep=1,
  %lines ext fixed,
  lines ext left anchor=base west,
  lines ext right anchor=base east,
  lines ext top dir=left,
  lines pos=0.5,
  lines sep=0,
  %lines style=\WCvarB,
  start angle=331.2
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\usetikzlibrary {decorations.markings}
\begin{tikzpicture}
\wheelchart[
  data=\WCvarC: \WCvarA,
  data angle shift=\WCvarG,
  data sep=0,
  data style={draw=\WCvarB,fill=\WCvarB!20},
  lines=1.5,
  lines ext=1,
  lines sep=-1,
  lines style={
    Black,
    postaction=decorate,
    decoration={
      markings,
      mark=at position 0 with {\fill[Black] (0,0) circle[radius=0.15];}
    }
  },
  pie,
  start angle=331.2
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/middle={⟨text⟩}` (no default, initially empty)

This key contains the ⟨*text*⟩ which will be placed at the center of the wheelchart. The ⟨*text*⟩ is placed in a node. The style of this node is given as follows. First, the option `align=center` is given. Thereafter, the style of the key `middle style` is added.

`/wheelchart/middle fill={⟨options⟩}` (style, no default, initially empty)

If this key is set then the middle of the wheelchart will be filled with this style. This requires a fixed inner and outer radius for all slices. This key does *not* apply if a plot is used.

`/wheelchart/middle style={⟨options⟩}` (style, no default, initially empty)

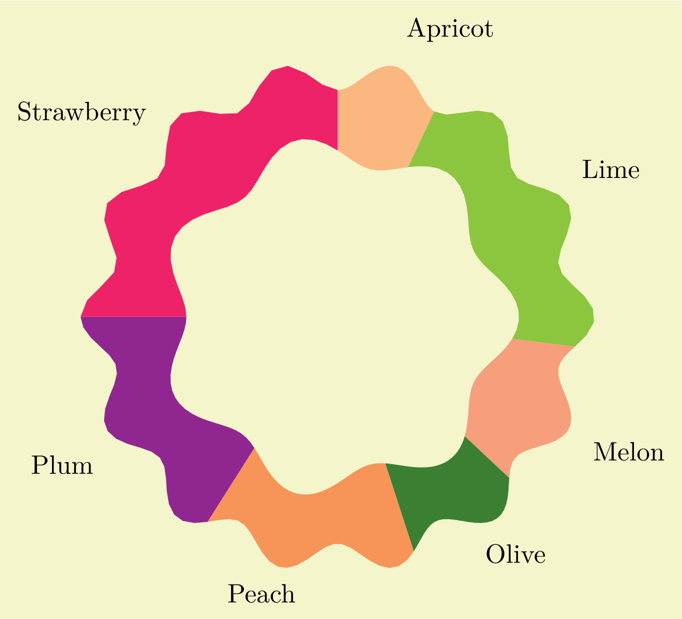
This key accepts a list of keys which will be applied to the node where the contents of the key `middle` is placed.

`/wheelchart/name={⟨name⟩}` (no default, initially `wheelchart@name`)

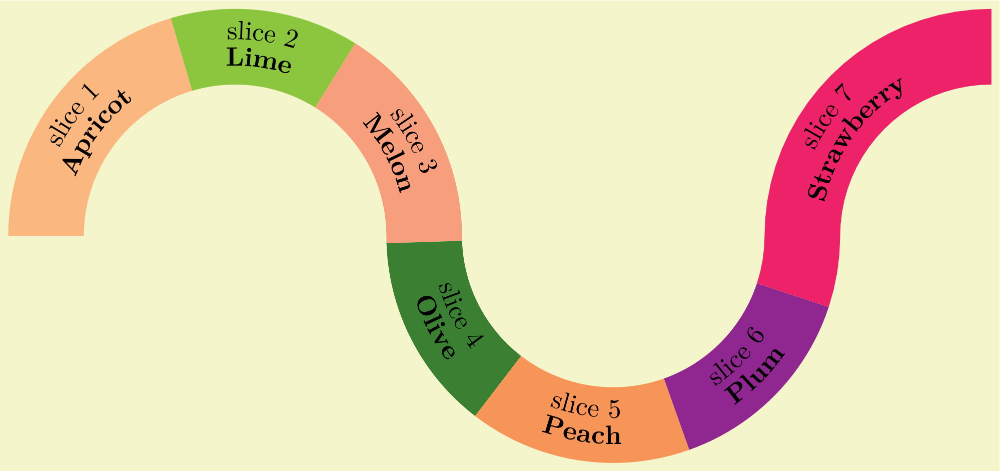
This key defines the ⟨*name*⟩ of the `local bounding box` around the wheelchart.

`/wheelchart/outer plot={⟨code⟩}` (no default)

This key is similar to the key `inner plot` but determines the outer parts of the slices of the wheelchart.



```
\begin{tikzpicture}
\wheelchart[
  inner plot={{#1}:{#2+0.2*(cos(#1*\WCtotalcount)+1)}},
  outer plot={{#1}:{#2+0.2*(cos(#1*\WCtotalcount*2)+1)}}
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\usetikzlibrary {decorations.text}
\begin{tikzpicture}
\wheelchart[
  arc data=slice \WCcount\\|\bfseries|\WCvarC,
  arc data expand=f,
  arc data pos=0.5,
  arc data line sep factor=1.5,
  data=,
  domain=0:450,
  inner plot={
    {int((#1)/180)*5+(0.5-((-1)^Mod(int((#1)/180),2))*2.5)*cos(#1)},
    {(2.5-((-1)^Mod(int((#1)/180),2))*0.5)*sin(#1)}
  },
  outer plot={
    {int((#1)/180)*5+(-0.5-((-1)^Mod(int((#1)/180),2))*2.5)*cos(#1)},
    {(2.5+((-1)^Mod(int((#1)/180),2))*0.5)*sin(#1)}
  },
  value=width("\WCvarC")
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/outer plot style={⟨options⟩}` (style, no default, initially empty)

This key accepts a list of keys which will be applied to the plot determined by the key `outer plot`.

`/wheelchart/outer radius={⟨value⟩}` (no default, initially `3`)

The ⟨*value*⟩ of this key defines the outer radius of the wheelchart.

`/wheelchart/perc precision={⟨number⟩}` (no default, initially `0`)

This key defines the number of decimals up to which the percentage in the macros `\WCperc` and `\WCpercentagerounded` is rounded. The rounding is performed with `l3fp`. With the initial setting, for example 49.5 and 50.5 are both rounded to 50. With `perc precision={0,1}`, 49.5 is rounded to 50 and 50.5 to 51.

`/wheelchart/pie=⟨boolean⟩` (default `true, initially false`)

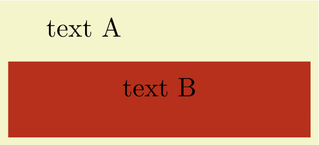
If true, the inner radius of the wheelchart is set to `0`.

`/wheelchart/plot={⟨code⟩}` (no default)

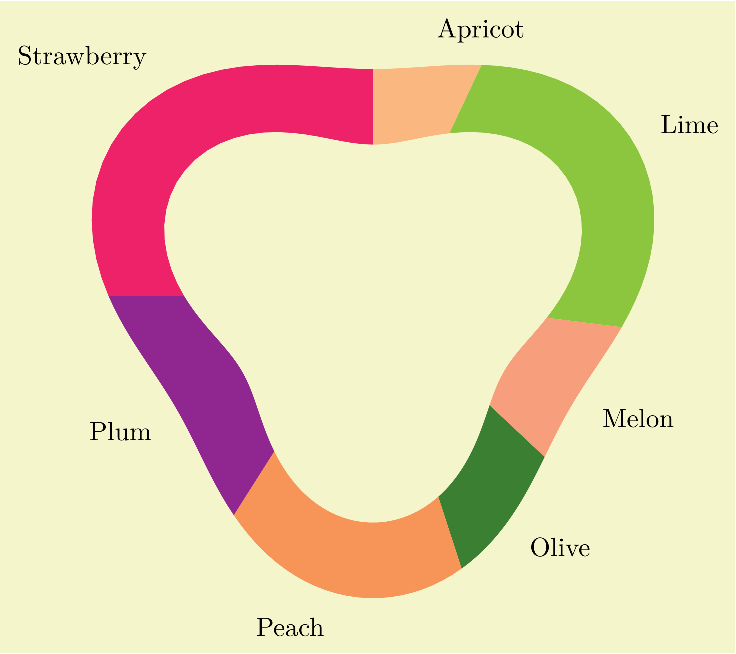
This key sets `inner plot` and `outer plot`.

Since the *let operation* from the TikZ library `calc` is used, it is not possible to use the variable names `\n`, `\p`, `\x` and `\y` inside the ⟨*code*⟩.

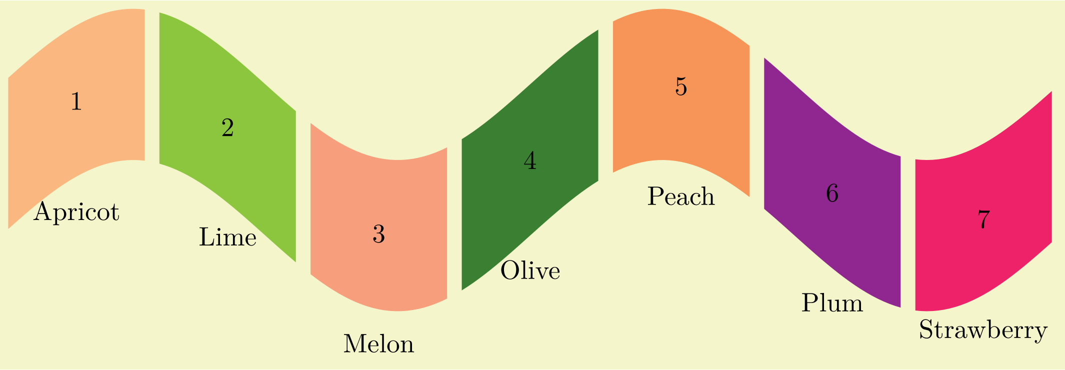
Note that positions depend on the `domain` and *not* on the length of the `plot`. For example below, `data angle pos=0.5`. The corresponding value of the `domain` is 1 which gives the x coordinate 1 which is *not* in the middle of the plot. Whereas `wheel data angle pos=sqrt(2)/2`. The corresponding value of the `domain` is $\sqrt{2}$ which gives the x coordinate 2 which is in the middle of the plot.



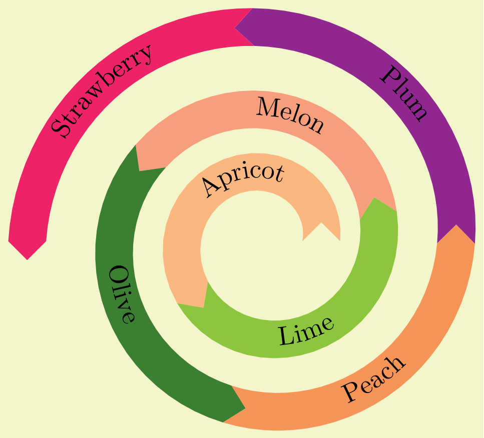
```
\begin{tikzpicture}
\wheelchart[
  domain=0:2,
  plot={{(#1)^2},{#2}},
  wheel data=text B,
  wheel data angle pos=sqrt(2)/2
]{1/BrickRed/text A}
\end{tikzpicture}
```



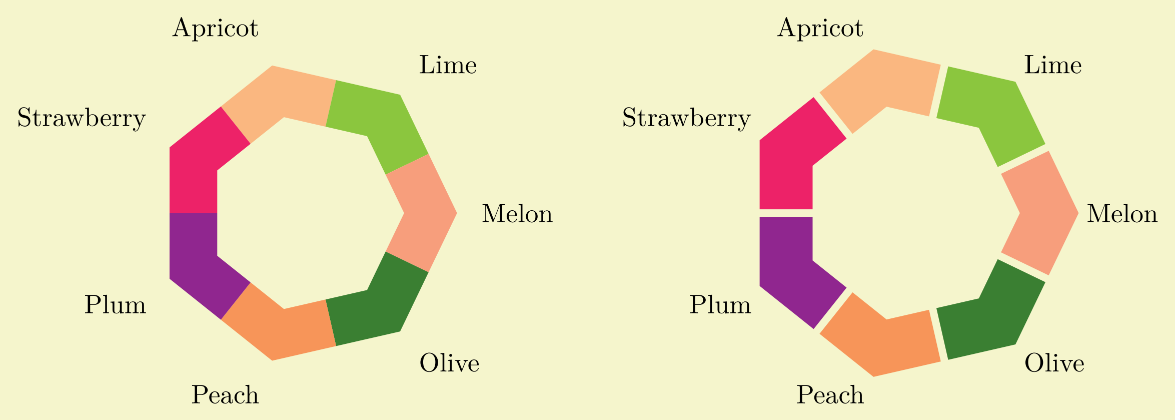
```
\begin{tikzpicture}
\wheelchart[
  plot={{#1}:{0.5*(sin(#1*3)+1)+#2}}
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\wheelchart[
  domain=0:720,
  gap polar=5,
  plot={{#1*3.5/180},{sin(#1)-#2}},
  radius={0}{2},
  value=1,
  wheel data=\WCcount,
  wheel data pos=0.5
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\usetikzlibrary {decorations.text}
\begin{tikzpicture}
\wheelchart[
  arc data=\WCvarC,
  arc data dir={\WCmidangle<180?-1:1},
  arc data pos=0.5,
  data=,
  domain=0:900,
  plot={{#1}:
    {(((#1)*pi/180+15)^2-1)/300
      +(#2)-0.25}},
  radius={0}{0.5},
  slices arrow={1}{0},
  value=sqrt(3+\WCcount*pi*(pi+6)/7)-
    sqrt(3+(\WCcount-1)*pi*(pi+6)/7)
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\pgfkeys{
  /wheelchart,
  gap,
  radius={1.3}{2},
  start angle=180*(1-2/\WCtotalcount),
  value=1
}
\wheelchart[
  plot={{#1}:{(#2)*cos(180/\WCtotalcount)/cos(Mod(#1,{360/\WCtotalcount})-180/\WCtotalcount)}}
]{\exampleforthismanual}
\wheelchart[
  at={(8,0)},
  slices inner arrow={-cot(90*(1-2/\WCtotalcount))}{0},
  slices outer arrow={cot(90*(1-2/\WCtotalcount))}{0}
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/plot style={⟨options⟩}` (style, no default, initially empty)

This key sets `inner plot style` and `outer plot style`.

`/wheelchart/radius={⟨inner radius⟩}{⟨outer radius⟩}` (no default)

This key defines the inner and outer radius of the wheelchart.

`/wheelchart/samples={⟨number⟩}` (no default, initially 25)

This key determines the ⟨*number*⟩ of samples used in the plots.

`/wheelchart/separator columns={⟨delimiter⟩}` (no default, initially `/`)

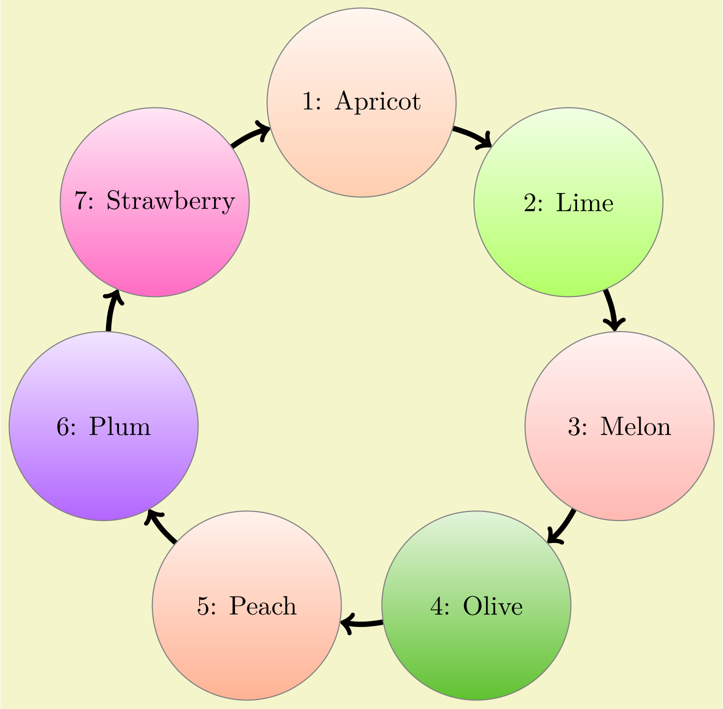
`/wheelchart/separator rows={⟨delimiter⟩}` (no default, initially `,`)

The ⟨*wheelchart data*⟩ in the command `\wheelchart` is a list in which the items are separated by the value of the key `separator rows`. Each item in this list corresponds to one slice of the wheelchart and consists of data separated by the value of the key `separator columns`.

`/wheelchart/slices={⟨path⟩}` (no default)

If this key is set then the shape of the slices of the wheelchart is defined by ⟨*path*⟩.

In the following example, a `;` is placed at the beginning of the argument for the key `slices` because there is no path to be filled. Thereafter, a node is placed still within the argument for the key `slices`.



```
\begin{tikzpicture}
\wheelchart[
  data=,
  radius={3.5}{3.5},
  slices={
    ;
    \node[
      bottom color=\WCvarB!60,
      top color=\WCvarB!10,
      circle,
      draw=gray,
      minimum width=2.5cm
    ]
    (WCslice\WCcount)
    {\WCcount: \WCvarC};
  },
  slices style={},
  start half,
  value=1
]{\exampleforthismanual}
\foreach\n in {1,...,7}{
  \pgfmathsetmacro{\k}{int(Mod(\n,7)+1)}
  \draw[->,line width=2pt] (WCslice\n)
    to[bend left=10] (WCslice\k);
}
\end{tikzpicture}
```

/wheelchart/slices angle pos={⟨*value*⟩} (no default, initially 0.5)

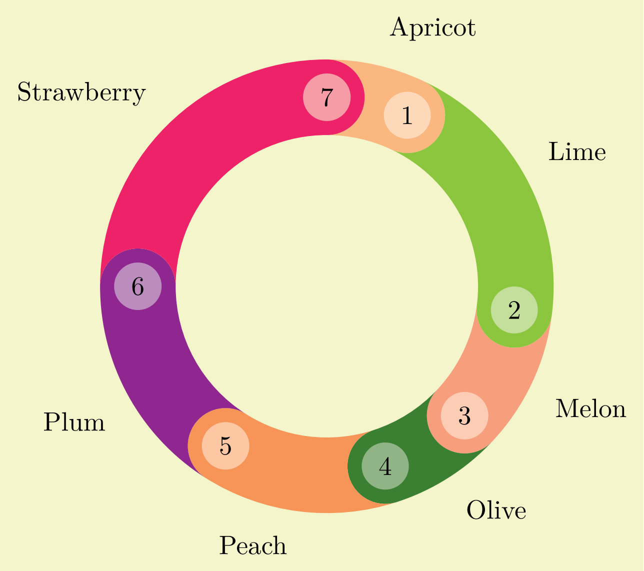
/wheelchart/slices angle shift={⟨*angle*⟩} (no default, initially 0)

These keys determine the position of the slices if the key `slices` is used similar as the corresponding keys for the key `data`.

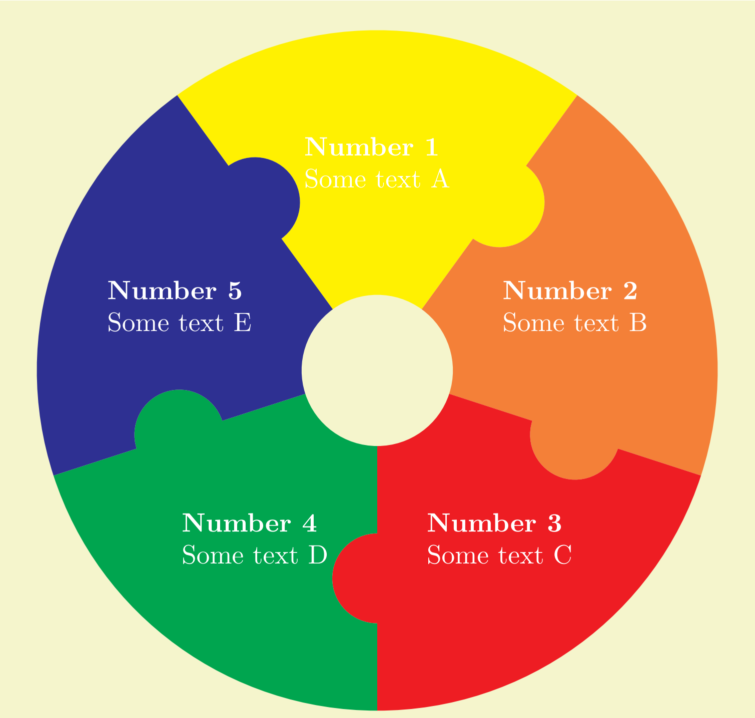
Below we list some keys to modify the shape of the slices. These keys only affect the shape of the slices and *not* the computation of the inner and outer sides. In particular, these keys do *not* affect the placement of `arc`, `arc data`, `data`, `inner data`, `lines`, `wheel data` and `wheel lines`. If this placement should be changed then the keys `inner plot` and `outer plot` can be used.

/wheelchart/slices arc={⟨*value 1*⟩}{⟨*value 2*⟩} (no default)

This key sets `slices end arc` and `slices start arc` but uses the opposite of ⟨*value 1*⟩ for `slices start arc`.



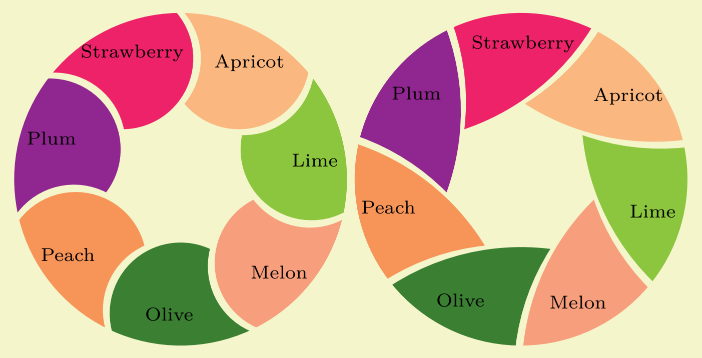
```
\begin{tikzpicture}
\wheelchart[
  slices arc={1}{0},
  wheel data=\WCcount,
  wheel data angle pos=1,
  wheel data pos=0.5,
  wheel data style={
    circle,
    fill=\WCvarB!50
  }
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\wheelchart[
  data=,
  radius={1}{4.5},
  slices arc={1}{0.66},
  slices style=\WCvarA,
  start half,
  value=1,
  wheel data={%
    \textbf{Number \WCcount}\\%
    \WCvarB%
  },
  wheel data pos=0.5,
  wheel data style=White
]{%
  Yellow/Some text A,
  Orange/Some text B,
  Red/Some text C,
  Green/Some text D,
  Blue/Some text E%
}
\end{tikzpicture}
```

/wheelchart/slices arc inner end=⟨*boolean*⟩ (default `true`, initially `false`)

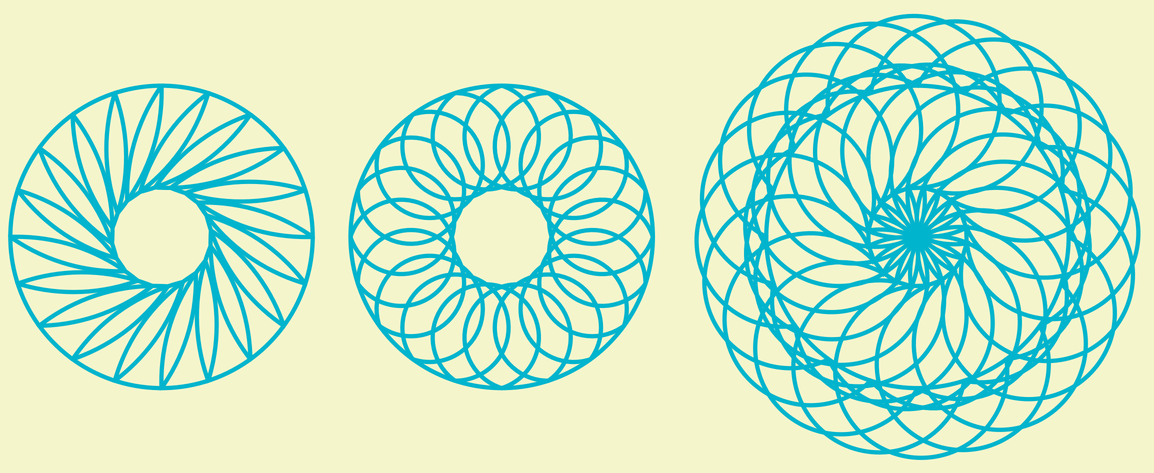
If true then the keys `slices end arc`, `slices inner arc` and `slices start arc` are set such that the inner part and the end of each of the slices of the wheelchart form one arc and such that the start has the opposite curvature as the end.



```
\begin{tikzpicture}[font=\scriptsize]
\foreach\a/\x in {0/0,45/4.5}{
  \wheelchart[
    at={(\x,0)},
    data=,
    gap,
    radius={1}{2.2},
    slices arc inner end,
    slices outer angle shift=\a,
    value=1,
    wheel data=\WCvarC,
    wheel data angle pos=0.6
  ]{\exampleforthismanual}
}
\end{tikzpicture}
```

/wheelchart/slices arc inner end start=⟨*boolean*⟩ (default `true`, initially `false`)

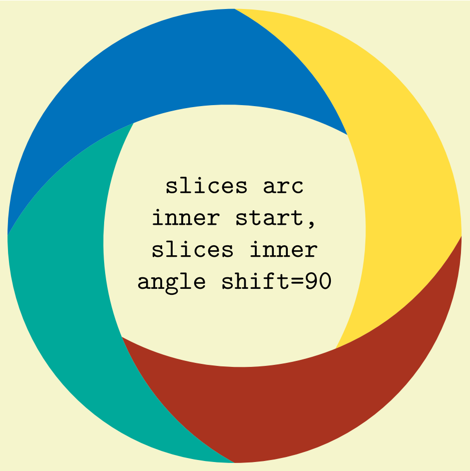
If true then the keys `slices end arc`, `slices inner arc` and `slices start arc` are set such that the inner part and the end of each of the slices of the wheelchart form one arc and such that the start has the same curvature as the end.



```
\begin{tikzpicture}
\foreach\a/\x in {-60/0,0/4.5,60/10}{
  \wheelchart[
    at={(\x,0)},
    radius={0.66}{2},
    slices arc inner end start,
    slices inner angle shift=\a,
    slices style={fill=none,draw=Turquoise,ultra thick},
    total count=20
  ]{}
}
\end{tikzpicture}
```

/wheelchart/slices arc inner start=⟨*boolean*⟩ (default `true`, initially `false`)

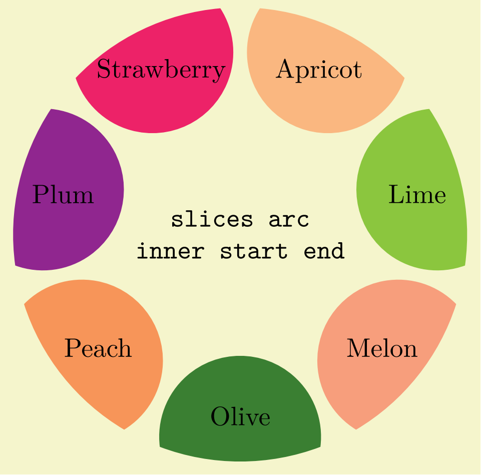
If true then the keys `slices end arc`, `slices inner arc` and `slices start arc` are set such that the inner part and the start of each of the slices of the wheelchart form one arc and such that the end has the opposite curvature as the start.



```
\begin{tikzpicture}
\wheelchart[
  middle={%
    slices arc\\%
    inner start,\\%
    slices inner\\%
    angle shift=90%
  },
  middle style={font=\ttfamily},
  slices arc inner start,
  slices inner angle shift=90
]{%
  1/Goldenrod/,
  1/Mahogany/,
  1/JungleGreen/,
  1/RoyalBlue/%
}
\end{tikzpicture}
```

`/wheelchart/slices arc inner start end=`⟨*boolean*⟩ (default `true`, initially `false`)

If true then the keys `slices end arc`, `slices inner arc` and `slices start arc` are set such that the inner part and the start of each of the slices of the wheelchart form one arc and such that the end has the same curvature as the start.

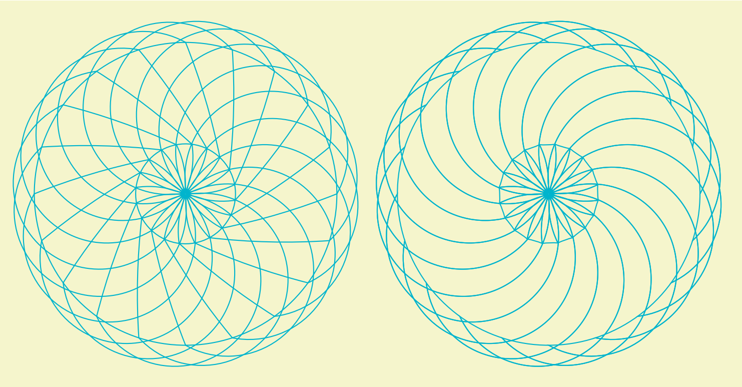


```
\begin{tikzpicture}
\wheelchart[
  data=,
  gap polar=5,
  middle={%
    slices arc\\%
    inner start end%
  },
  middle style={font=\ttfamily},
  slices arc inner start end,
  value=1,
  wheel data=\WCvarC,
  wheel data pos=0.4
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/slices arc match={`⟨*arg 1*⟩`}{`⟨*num 1*⟩`}{`⟨*num 2*⟩`}{`⟨*num 3*⟩`}{`⟨*arg 2*⟩`}{`⟨*arg 3*⟩`}{`⟨*arg 4*⟩`}`
(no default)

This key modifies the shape of the slices according to the 7 arguments.

Here, ⟨*arg 1*⟩ must be `end`, `inner`, `outer` or `start` and ⟨*arg 2*⟩, ⟨*arg 3*⟩ and ⟨*arg 4*⟩ must be `inner end`, `inner start`, `outer end` or `outer start`. For example, the key `slices arc inner end` sets `slices arc match={inner}{1}{-1}{1}{inner end}{inner start}{outer end}`.



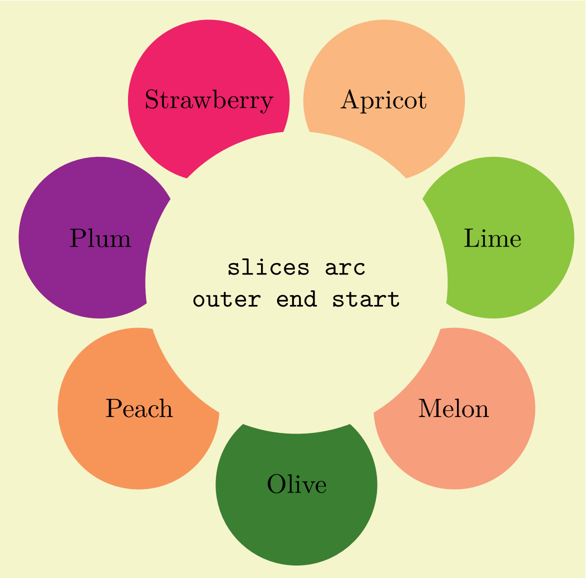
```
\begin{tikzpicture}
\foreach\a/\b/\x in
  {end/1/0,inner/-1/4.8}{
  \wheelchart[
    at={(\x,0)},
    radius={0.66}{2},
    slices arc match=
      {\a}{\b}{1}{1}{inner end}
      {inner start}{outer end},
    slices inner angle shift=60,
    slices style={
      fill=none,
      draw=Turquoise
    },
    total count=20
  ]{}
}
\end{tikzpicture}
```

`/wheelchart/slices arc outer end=`⟨*boolean*⟩ (default `true`, `initially false`)

If true then the keys `slices end arc`, `slices outer arc` and `slices start arc` are set such that the outer part and the end of each of the slices of the wheelchart form one arc and such that the start has the opposite curvature as the end.

`/wheelchart/slices arc outer end start=`⟨*boolean*⟩ (default `true`, `initially false`)

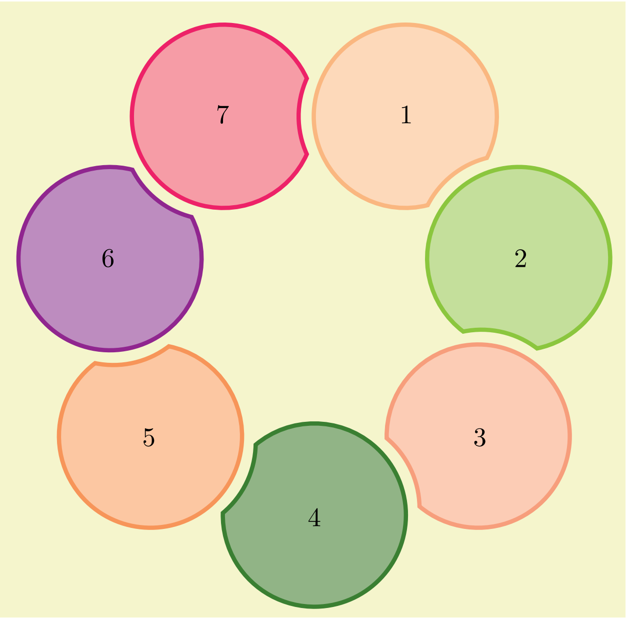
If true then the keys `slices end arc`, `slices outer arc` and `slices start arc` are set such that the outer part and the end of each of the slices of the wheelchart form one arc and such that the start has the same curvature as the end.



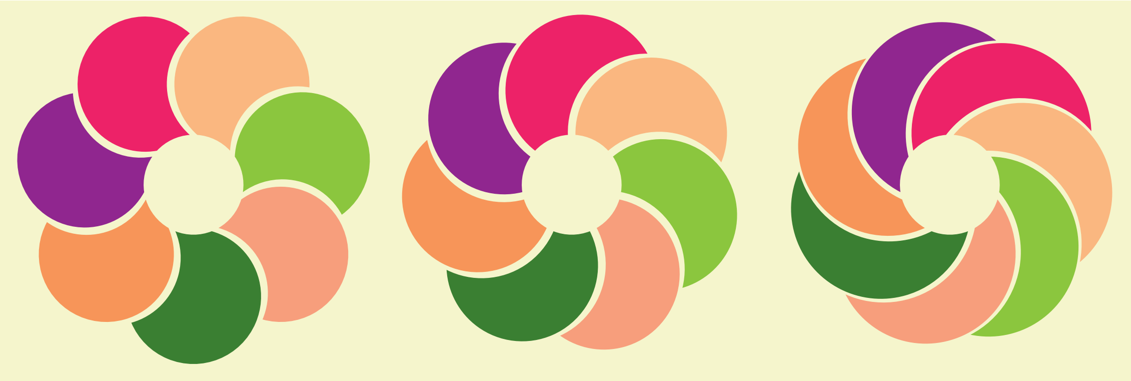
```
\begin{tikzpicture}
\wheelchart[
  data=,
  gap polar=5,
  middle={%
    slices arc\\%
    outer end start%
  },
  middle style={font=\ttfamily},
  slices arc outer end start,
  value=1,
  wheel data=\WCvarC
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/slices arc outer start=`⟨*boolean*⟩ (default `true`, `initially false`)

If true then the keys `slices end arc`, `slices outer arc` and `slices start arc` are set such that the outer part and the start of each of the slices of the wheelchart form one arc and such that the end has the opposite curvature as the start.



```
\begin{tikzpicture}
\wheelchart[
  data=,
  gap=0.1,
  slices arc inner start,
  slices arc outer start,
  slices style={
    \WCvarB!50,
    draw=\WCvarB,
    ultra thick
  },
  value=1,
  wheel data=\WCcount,
  wheel data pos=0.8
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\foreach\a/\x in {0/0,45/5,90/10}{
  \wheelchart[
    at={(\x,0)},
    data=,
    gap,
    radius={0.66}{2},
    slices arc outer start,
    slices outer angle shift=\a,
    value=1
  ]{\exampleforthismanual}
}
\end{tikzpicture}
```

`/wheelchart/slices arc outer start end=`⟨*boolean*⟩ (default `true`, initially `false`)

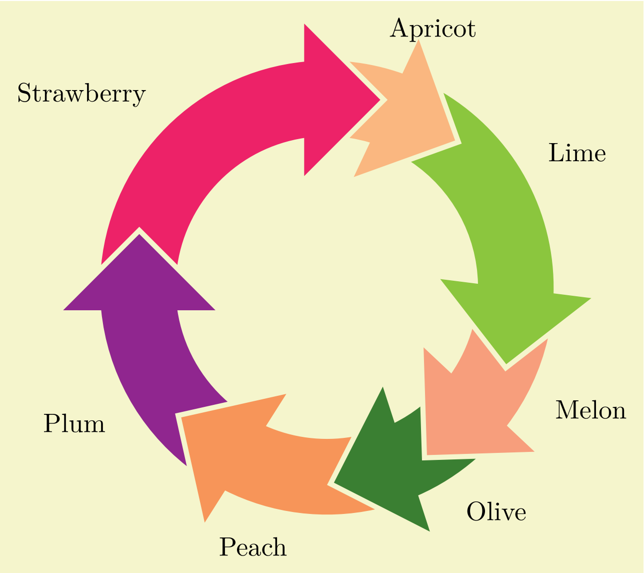
If true then the keys `slices end arc`, `slices outer arc` and `slices start arc` are set such that the outer part and the start of each of the slices of the wheelchart form one arc and such that the end has the same curvature as the start.

`/wheelchart/slices Arrow={`⟨*angle*⟩`}` (no default)

This key sets `slices end` to `--(\WCpoint{1}{`⟨*angle*⟩`}{0.5}{0})--(\WCpoint{1}{0}{0}{0})` and `slices start` to `--(\WCpoint{0}{`⟨*angle*⟩`}{0.5}{0})--cycle`.

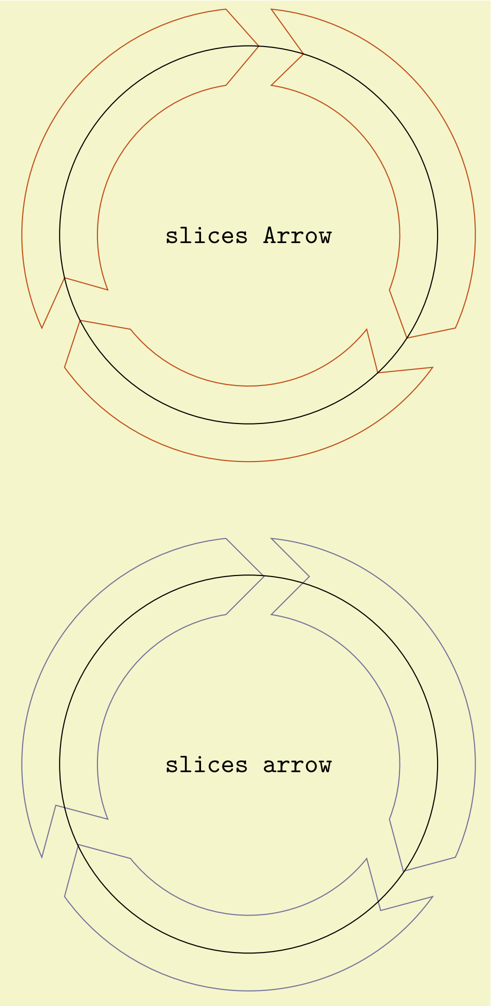
`/wheelchart/slices arrow={`⟨*value 1*⟩`}{`⟨*value 2*⟩`}` (no default)

This key is similar to the key `slices arc` but draws an arrow.



```
\begin{tikzpicture}
\wheelchart[
  gap=0.3,
  slices arrow={1}{-1}
]{\exampleforthismanual}
\end{tikzpicture}
```

The example below compares arrows constructed with the key `slices Arrow` and the key `slices arrow`. Using the key `slices Arrow`, the arrow tip lies on the circle but the line segments do *not* have the same length. Using the key `slices arrow`, the arrow tip does *not* lie on the circle but the line segments have the same length.



```
\begin{tikzpicture}
\wheelchart[
  gap=0.3,
  middle=slices Arrow,
  middle style={font=\ttfamily},
  slices Arrow=10,
  slices style={
    fill=none,
    draw=Bittersweet
  },
  total count=3
]{}
\draw (0,0) circle[radius=2.5];
\wheelchart[
  at={(0,-7)},
  gap=0.3,
  middle=slices arrow,
  middle style={font=\ttfamily},
  slices arrow={1}{0},
  slices style={
    fill=none,
    draw=CadetBlue
  },
  total count=3
]{}
\draw (0,-7) circle[radius=2.5];
\end{tikzpicture}
```

`/wheelchart/slices end={⟨path⟩}` (no default)

This key determines the end of the slice. Initially, this is a line segment from the outer end to the inner end of the slice.

`/wheelchart/slices end arc={`⟨*value 1*⟩`}{`⟨*value 2*⟩`}` (no default)

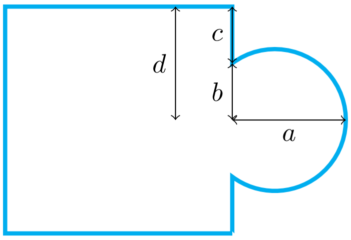
This key determines the end of the slice.

The effect of ⟨*value 1*⟩ and ⟨*value 2*⟩ is shown in the figure and the table below.

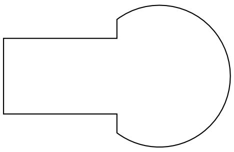
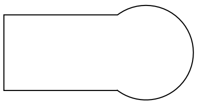
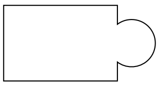
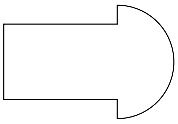
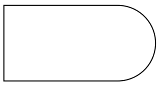
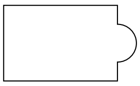
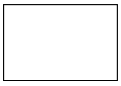
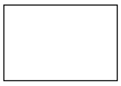
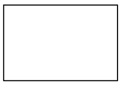
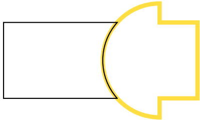
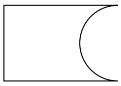
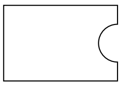
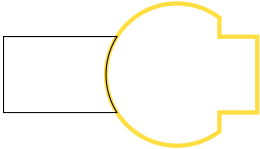
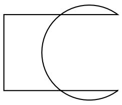
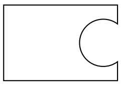
If ⟨*value 1*⟩ > 0 then the arc points outwards the slice. If ⟨*value 1*⟩ < 0 then the arc points inwards the slice. Here, outwards and inwards are relative to the orientation of the four-sided polygon formed by the points whose coordinates are determined by the inner and outer radius of the first slice and the start angle and the angle at the inverse of the key `samples` between the start angle and the end angle of the first slice. If the start angle and the end angle of the first slice are equal then the end angle of the last slice is used instead. If this test is inconclusive then the orientation is set according to the key `counterclockwise`.

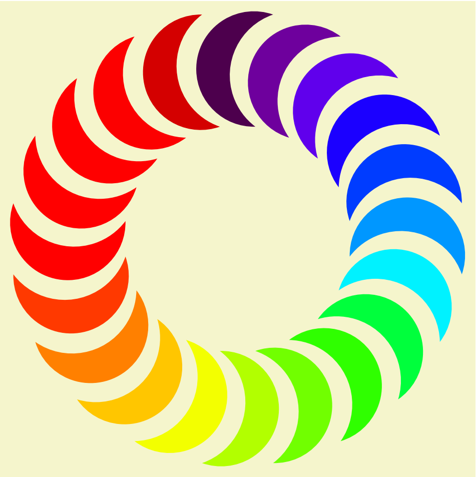
If ⟨*value 1*⟩ $= 0$ then a line segment is drawn.

If ⟨*value 1*⟩ and ⟨*value 2*⟩ are negative then an arc is drawn which behaves the same as an arc with ⟨*value 2*⟩ $= 0$ and such that its radius matches the radius of the arc corresponding to setting ⟨*value 1*⟩ to its opposite. This is illustrated in the table below.



$|\langle \textit{value 1} \rangle| = \frac{a}{b},\ |\langle \textit{value 2} \rangle| = \frac{c}{d}$

	⟨*value 2*⟩ $= -0.5$	⟨*value 2*⟩ $= 0$	⟨*value 2*⟩ $= 0.5$
⟨*value 1*⟩ $= 2$			
⟨*value 1*⟩ $= 1$			
⟨*value 1*⟩ $= 0$			
⟨*value 1*⟩ $= -1$			
⟨*value 1*⟩ $= -2$			



```
\begin{tikzpicture}
\wheelchart[
  for loop start={
    \definecolor{WCcolor}{wave}{
    \fpeval{380+(\WCcount-1)*
    340/(\WCtotalcount-1)}}
  },
  gap polar=180/\WCtotalcount,
  radius={1.5}{3},
  slices end arc={-0.6}{0},
  slices start arc={1.2}{0},
  slices style=WCcolor,
  total count=20
]{}
\end{tikzpicture}
```

`/wheelchart/slices end arrow={⟨value 1⟩}{⟨value 2⟩}` (no default)

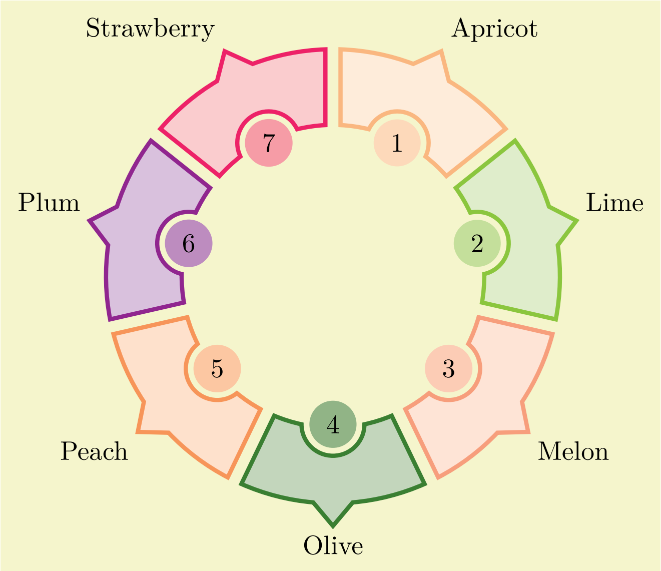
This key is similar to the key `slices end arc` but draws an arrow.

`/wheelchart/slices end to={⟨value 1⟩}{⟨value 2⟩}` (no default)

This key sets the `to` path operation for the end of the slice. The angle at the inner side is determined by ⟨*value 1*⟩ and the angle at the outer side is determined by ⟨*value 2*⟩.

`/wheelchart/slices inner={⟨path⟩}` (no default)

This key determines the inner part of the slice. Initially, this is an arc from the inner end to the inner start of the slice.



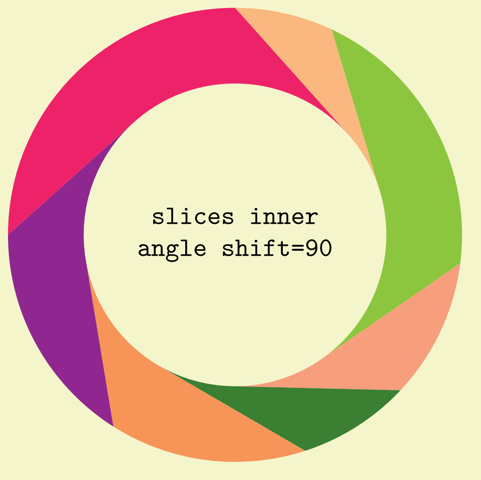
```
\begin{tikzpicture}
\def\a{5}
\def\b{12}
\wheelchart[
  data sep=0.3,
  gap=0.1,
  inner data=\WCcount,
  inner data sep=2-2*cos(\b),
  inner data style={
    circle,
    fill=\WCvarB!50
  },
  slices inner={
    arc[
      start angle=\WCangle{1}{0}{0}{0},
      end angle=\WCangle{0.5}{\b}{0}{0},
      radius=\WCradius{0}{0}
    ]
    arc[
      start angle=\WCmidangle-90,
      end angle=\WCmidangle+90,
      radius=2*sin(\b)
    ]
    arc[
      start angle=
        \WCangle{0.5}{-\b}{0}{0},
      end angle=\WCangle{0}{0}{0}{0},
      radius=\WCradius{0}{0}
    ]
  },
  slices outer={
    arc[
      start angle=\WCangle{0}{0}{1}{0},
      end angle=\WCangle{0.5}{-\a}{1}{0},
      radius=\WCradius{1}{0}
    ]
    --(\WCpoint{0.5}{0}{1}{0.3})
    --(\WCpoint{0.5}{\a}{1}{0})
    arc[
      start angle=
        \WCangle{0.5}{\a}{1}{0},
      end angle=\WCangle{1}{0}{1}{0},
      radius=\WCradius{1}{0}
    ]
  },
  slices style={
    draw=\WCvarB,
    fill=\WCvarB!25,
    ultra thick
  },
  value=1
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/slices inner angle reduce`={⟨*angle*⟩} (no default)

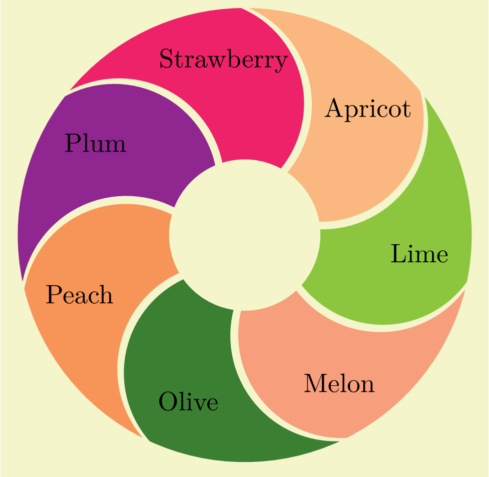
This key sets `slices inner end angle shift` to −⟨*angle*⟩ and `slices inner start angle shift` to ⟨*angle*⟩.

`/wheelchart/slices inner angle shift`={⟨*angle*⟩} (no default)

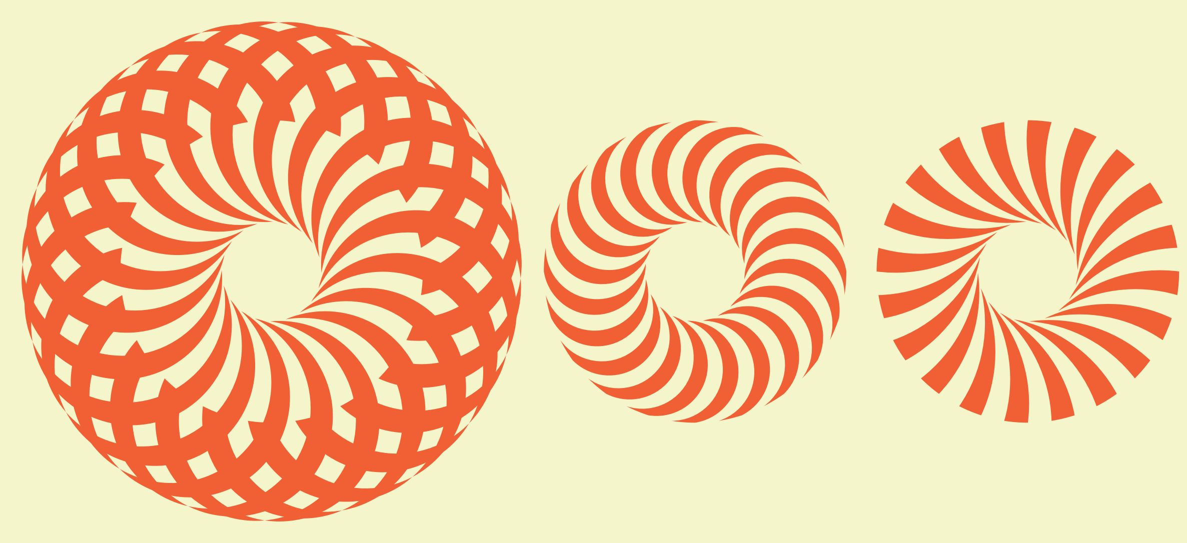
This key sets `slices inner end angle shift` and `slices inner start angle shift` to ⟨*angle*⟩.



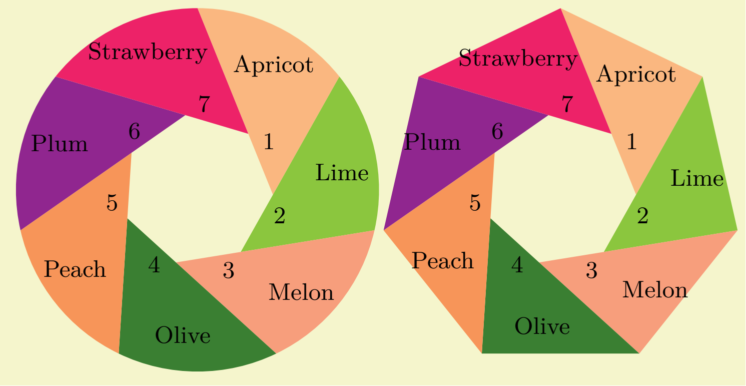
```
\begin{tikzpicture}
\wheelchart[
  data=,
  middle={%
    slices inner\\%
    angle shift=90%
  },
  middle style={font=\ttfamily},
  slices inner angle shift=90
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\wheelchart[
  data=,
  gap,
  radius={1}{3},
  slices arc={0.5}{0},
  slices inner angle shift=45,
  value=1,
  wheel data=\WCvarC,
  wheel data angle pos=0.8
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\foreach\a/\x in {-60/0,0/5.6,60/10}{
  \wheelchart[
    at={(\x,0)},
    radius={0.66}{2},
    slices arc inner start,
    slices inner angle shift=\a,
    slices style={fill=\WClistcolors},
    total count=40,
    WClistcolors={RedOrange,none}
  ]{}
}
\end{tikzpicture}
```



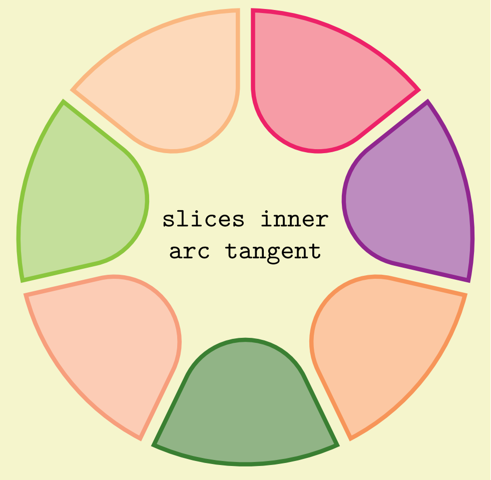
```
\begin{tikzpicture}[font=\small]
\pgfkeys{
  /wheelchart,
  data=,
  inner data=\WCcount,
  inner data pos=0.1,
  inner data sep=0,
  radius={1}{2.4},
  slices inner angle shift=
    90-180/\WCtotalcount,
  slices inner arc={0}{0},
  value=1,
  wheel data=\WCvarC
}
\wheelchart{\exampleforthismanual}
\wheelchart[
  at={(4.8,0)},
  slices outer arc={0}{0},
  wheel data pos=0.58
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/slices inner arc={⟨value 1⟩}{⟨value 2⟩}` (no default)

This key is similar to the key `slices end arc` but sets the inner part of the slice.

`/wheelchart/slices inner arc tangent=⟨boolean⟩` (default `true`, initially `false`)

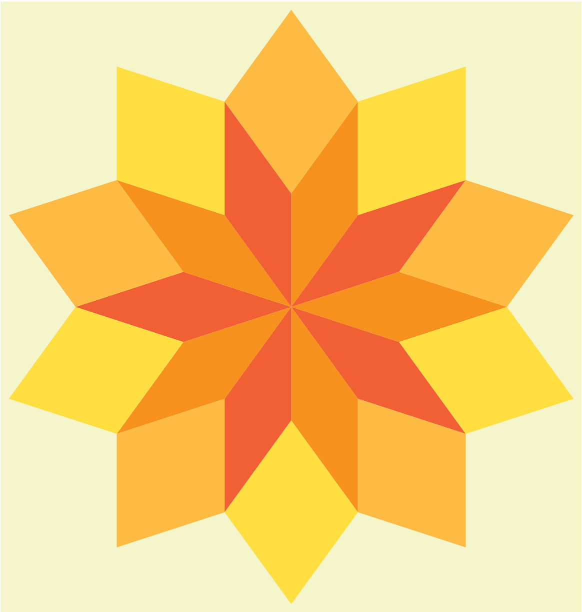
If true then the key `slices inner arc` is set such that the arc is tangent to the end and start of the slice if possible. Note that this is not possible for all settings for keys such as `plot` and `slices inner angle shift`.



```
\begin{tikzpicture}
\wheelchart[
  counterclockwise,
  data=,
  gap=0.1,
  middle=slices inner\\arc tangent,
  middle style={font=\ttfamily},
  slices inner arc tangent,
  slices style={
    draw=\WCvarB,
    fill=\WCvarB!50,
    ultra thick
  },
  value=1
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/slices inner arrow={⟨value 1⟩}{⟨value 2⟩}` (no default)

This key is similar to the key `slices end arrow` but sets the inner part of the slice.



```
\begin{tikzpicture}
\def\n{10}
\wheelchart[
  radius={0}{1.5},
  slices outer arrow={cot(180/\n)}{0},
  slices style{list}={
    BurntOrange,RedOrange
  },
  total count=\n
]{}
\wheelchart[
  radius={3*cos(180/\n)}
    {3*cos(180/\n)},
  slices inner arrow={cot(360/\n)}{0},
  slices outer arrow={cot(360/\n)}{0},
  slices style{list}={
    Dandelion,Goldenrod
  },
  start half,
  total count=\n
]{}
\end{tikzpicture}
```

`/wheelchart/slices inner end angle shift={⟨angle⟩}` (no default, initially `0`)

The end angle of the inner part of the slice will be modified such that the angle between the end and the inner part of the slice is shifted with ⟨*angle*⟩ (taking into account the key `counterclockwise`). The behavior of this key depends on whether a plot is used.

`/wheelchart/slices inner start angle shift={⟨angle⟩}` (no default, initially `0`)

This key is similar to the key `slices inner end angle shift` but modifies the start angle of the inner part of the slice.

`/wheelchart/slices inner to={⟨value 1⟩}{⟨value 2⟩}` (no default)

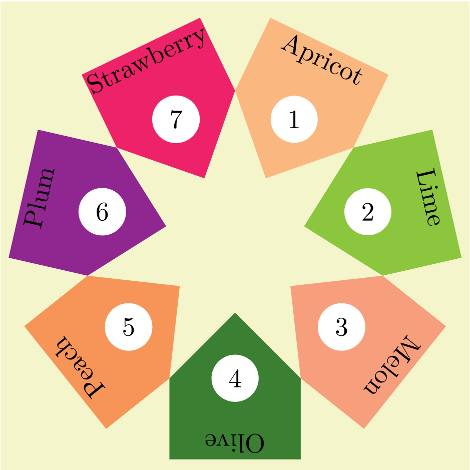
This key sets the `to` path operation for the inner part of the slice. The angle at the start is determined by ⟨*value 1*⟩ and the angle at the end is determined by ⟨*value 2*⟩.

`/wheelchart/slices outer={⟨path⟩}` (no default)

This key determines the outer part of the slice. Initially, this is an arc from the outer start to the outer end of the slice.

`/wheelchart/slices outer angle reduce={⟨angle⟩}` (no default)

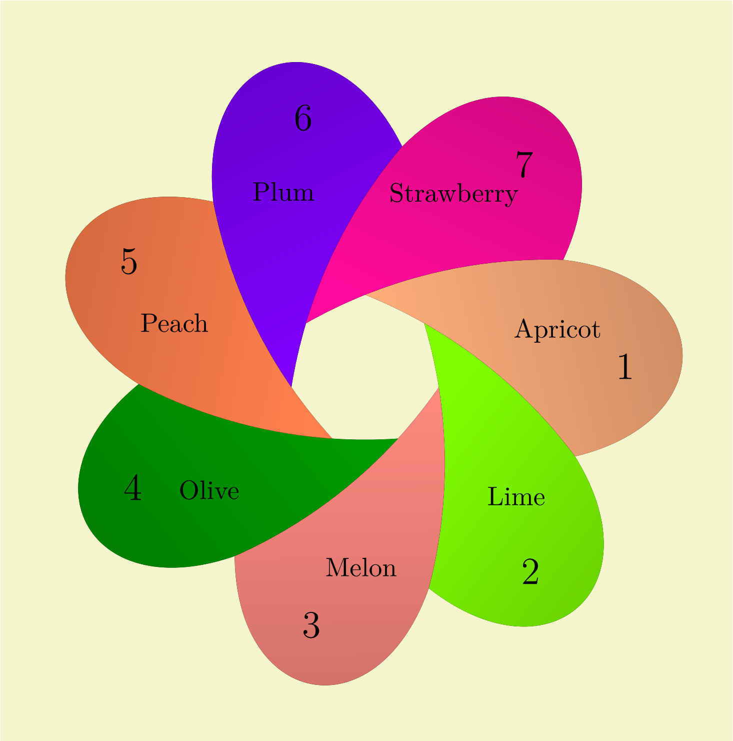
This key sets `slices outer end angle shift` to −⟨*angle*⟩ and `slices outer start angle shift` to ⟨*angle*⟩.



```
\begin{tikzpicture}
\wheelchart[
  data=,
  inner data=\WCcount,
  inner data style={
    circle,
    fill=white
  },
  slices inner arrow={1}{0},
  slices outer angle reduce=
    180/\WCtotalcount,
  slices outer arrow={0}{0},
  value=1,
  wheel data=\WCvarC,
  wheel data style={
    rotate=\WCmidangle-90
  }
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/slices outer angle shift={⟨angle⟩}` (no default)

This key sets `slices outer end angle shift` and `slices outer start angle shift` to ⟨*angle*⟩.



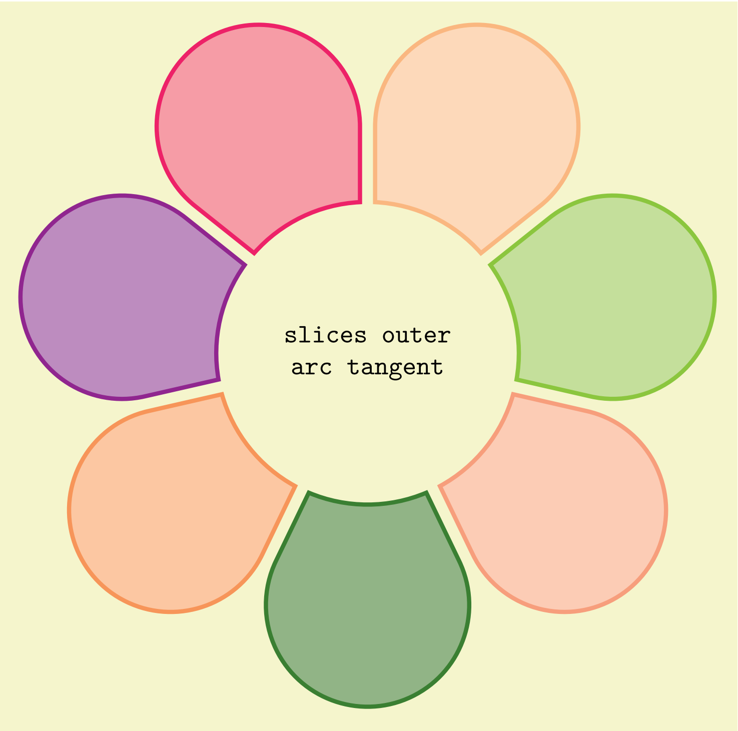
```
\begin{tikzpicture}[looseness=2]
\wheelchart[
  data=,
  inner data={\Large\WCcount},
  inner data pos=1.1,
  radius={1}{3},
  slices arc inner end,
  slices outer angle shift=80,
  slices outer to={80}{80},
  slices style={
    bottom color=\WCvarB,
    top color=\WCvarB!80!black,
    shading angle=\WCmidangle-90
  },
  value=1,
  wheel data=\WCvarC,
  wheel data angle pos=0.4,
  wheel data pos=0.8
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/slices outer arc={⟨value 1⟩}{⟨value 2⟩}` (no default)

This key is similar to the key `slices end arc` but sets the outer part of the slice.

`/wheelchart/slices outer arc tangent=⟨boolean⟩` (default `true`, initially `false`)

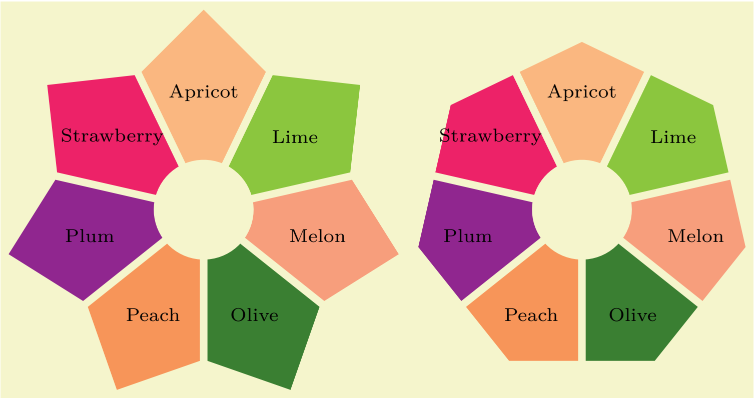
If true then the key `slices outer arc` is set such that the arc is tangent to the end and start of the slice if possible. Note that this is not possible for all settings for keys such as `plot` and `slices inner angle shift`.



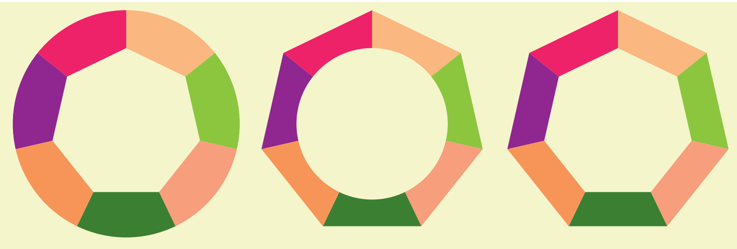
```
\begin{tikzpicture}
\wheelchart[
  data=,
  gap=0.1,
  middle=slices outer\\arc tangent,
  middle style={font=\ttfamily},
  slices outer arc tangent,
  slices style={
    draw=\WCvarB,
    fill=\WCvarB!50,
    ultra thick
  },
  value=1
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/slices outer arrow={⟨value 1⟩}{⟨value 2⟩}` (no default)

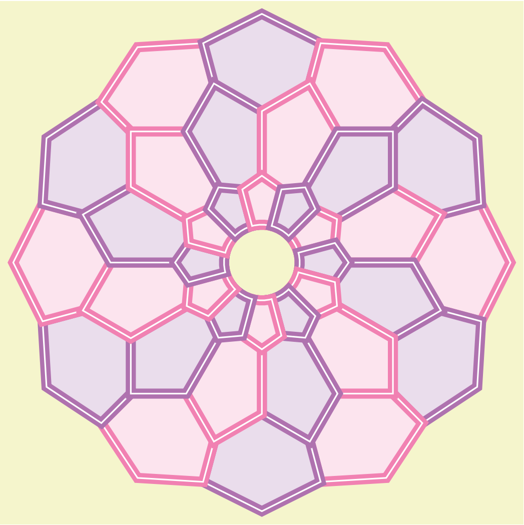
This key is similar to the key `slices end arrow` but sets the outer part of the slice.



```
\begin{tikzpicture}[font=\scriptsize]
\foreach\a/\x in
  {1/0,{tan(180/\WCtotalcount)}/5}{
  \wheelchart[
    at={(\x,0)},
    data=,
    gap,
    radius={0.66}{2},
    slices outer arrow={\a}{0},
    start half,
    value=1,
    wheel data=\WCvarC
  ]{\exampleforthismanual}
}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\pgfkeys{
  /wheelchart,
  data=,
  radius={1}{1.5},
  value=1
}
\wheelchart[
  slices inner arrow={0}{0}
]{\exampleforthismanual}
\wheelchart[
  at={(3.25,0)},
  slices outer arrow={0}{0}
]{\exampleforthismanual}
\wheelchart[
  at={(6.5,0)},
  slices inner arrow={0}{0},
  slices outer arrow={0}{0}
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\begin{tikzpicture}
\foreach\r/\s/\a in
  {3/0/0.5,2/15/1,1/30/0.7}{
  \wheelchart[
    radius={0.5}{\r},
    slices outer arrow={\a}{0},
    slices style={
      fill=\WClistcolors!20,
      draw=\WClistcolors,
      ultra thick,
      double
    },
    start half=\s,
    total count=12,
    WClistcolors={CarnationPink,Orchid}
  ]{}
}
\end{tikzpicture}
```

`/wheelchart/slices outer end angle shift={⟨angle⟩}` (no default, initially `0`)

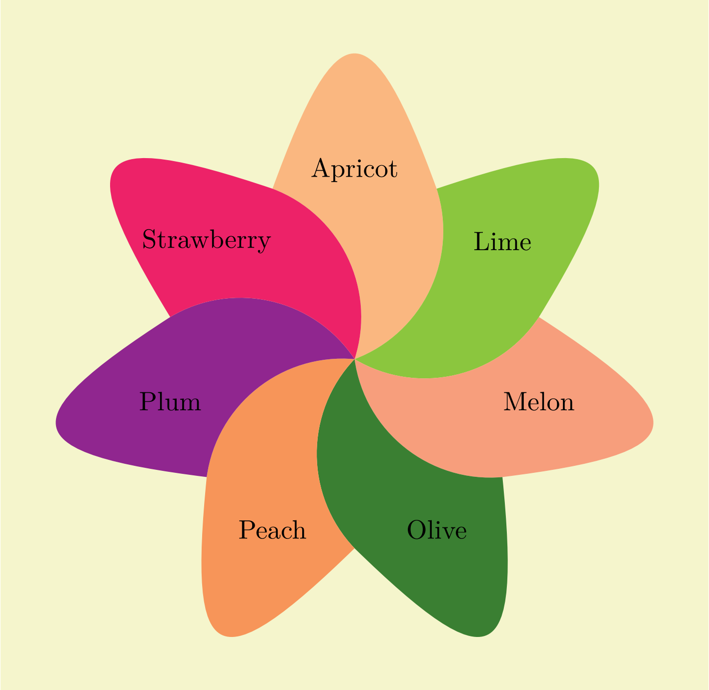
The end angle of the outer part of the slice will be modified such that the angle between the end and the inner (not the outer) part of the slice is shifted with ⟨*angle*⟩ (taking into account the key `counterclockwise`). The behavior of this key depends on whether a plot is used.

`/wheelchart/slices outer start angle shift={⟨angle⟩}` (no default, initially `0`)

This key is similar to the key `slices outer end angle shift` but modifies the start angle of the outer part of the slice.

`/wheelchart/slices outer to={⟨value 1⟩}{⟨value 2⟩}` (no default)

This key sets the `to` path operation for the outer part of the slice. The angle at the start is determined by ⟨*value 1*⟩ and the angle at the end is determined by ⟨*value 2*⟩.



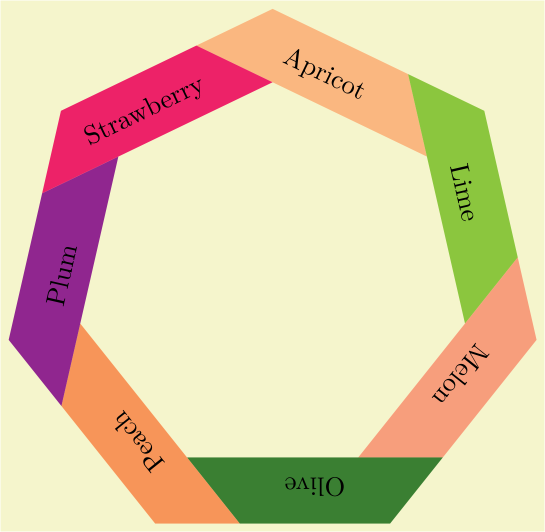
```
\begin{tikzpicture}[looseness=3]
\wheelchart[
  data=,
  radius={0}{2.5},
  slices arc={0.4}{0},
  slices outer to={70}{70},
  start half,
  value=1,
  wheel data=\WCvarC,
  wheel data pos=1
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/slices pos={⟨value⟩}` (no default, initially `0.5`)

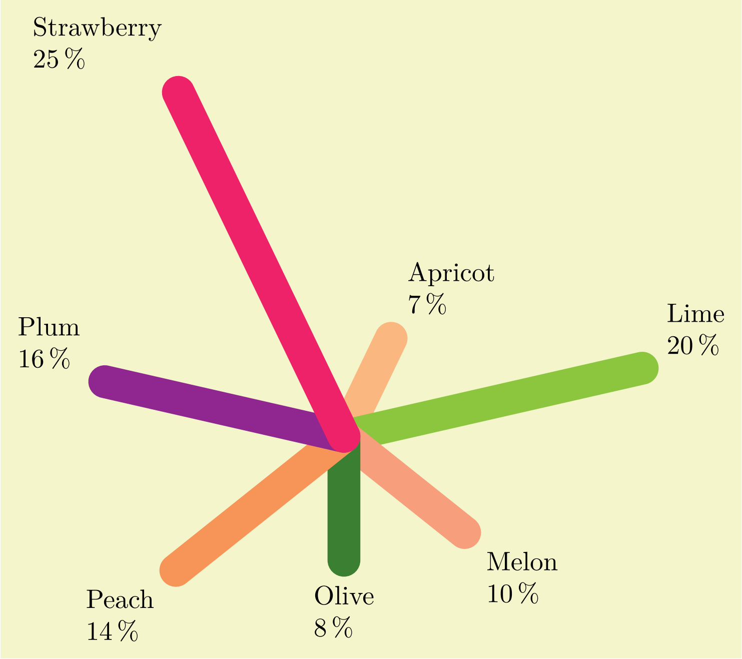
This key determines the position of the slices if the key `slices` is used similar as the corresponding key for the key `data`.

/wheelchart/slices scope={⟨*options*⟩} (style, no default, initially empty)

This key accepts a list of keys which will be applied to the scope in which the slices of the wheelchart, the wheel lines determined by the key `wheel lines` and the different kinds of data are placed.



```
\begin{tikzpicture}
\wheelchart[
  data=,
  radius={3.9}{4.5},
  slices inner arc={0}{0},
  slices outer angle reduce=5*90/7,
  slices outer arc={0}{0},
  slices scope={
    shift={
      ($(90+\WCmidangle:0.559572)
      +(\WCmidangle:-1.16196)$)
    }
  },
  value=1,
  wheel data=\WCvarC,
  wheel data pos=0,
  wheel data style={
    rotate=\WCmidangle-90
  }
]{\exampleforthismanual}
\end{tikzpicture}
```



```
\usepackage {siunitx}
\begin{tikzpicture}
\wheelchart[
  data=\WCvarC\\%
    \qty{\fpeval{\WCvarA/2}}{\percent},
  radius={0.5}{0.5+0.1*\WCvarA},
  slices inner arc tangent,
  slices outer angle reduce=
    180/\WCtotalcount,
  slices outer arc tangent,
  slices scope={shift={(\WCmidangle:
    {-cos(180/\WCtotalcount)/2})}},
  value=1
]{\exampleforthismanual}
\end{tikzpicture}
```

/wheelchart/slices sep={⟨*value*⟩} (no default, initially `0`)

This key determines the position of the slices if the key `slices` is used similar as the corresponding key for the key `data`.

/wheelchart/slices start={⟨*path*⟩} (no default)

This key determines the start of the slice. Initially, this is a line segment from the inner start to the outer start of the slice.

/wheelchart/slices start arc={⟨*value 1*⟩}{⟨*value 2*⟩} (no default)

This key is similar to the key `slices end arc` but sets the start of the slice.

/wheelchart/slices start arrow={⟨*value 1*⟩}{⟨*value 2*⟩} (no default)

This key is similar to the key `slices end arrow` but sets the start of the slice.

`/wheelchart/slices start to={⟨value 1⟩}{⟨value 2⟩}` (no default)

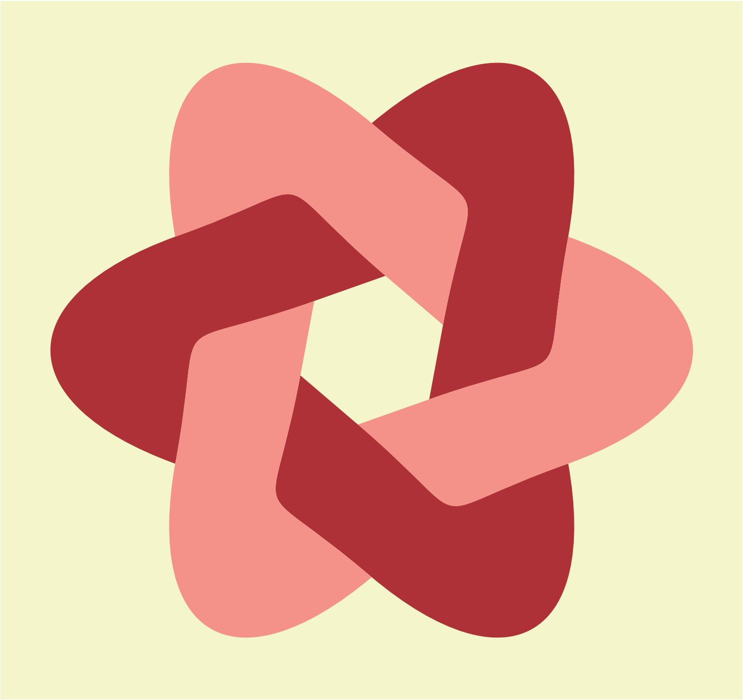
This key sets the `to` path operation for the start of the slice. The angle at the inner side is determined by ⟨*value 1*⟩ and the angle at the outer side is determined by ⟨*value 2*⟩.

`/wheelchart/slices style={⟨options⟩}` (style, no default, initially `\WCvarB`)

This key defines the style of the slices of the wheelchart.

`/wheelchart/slices to={⟨value 1⟩}{⟨value 2⟩}` (no default)

This key sets `slices end to` and `slices start to` but uses the opposite respective values for `slices start to`.



```
\begin{tikzpicture}[looseness=2]
\wheelchart[
  radius={1}{3},
  slices inner angle shift=90,
  slices inner arc={0}{0},
  slices outer to={70}{70},
  slices style{list}={Maroon,Salmon},
  slices to={30}{30},
  total count=6
]{}
\end{tikzpicture}
```

`/wheelchart/slice{⟨range⟩}={⟨options⟩}` (style, no default, initially empty)

This key accepts a list of keys from the wheelchart key family. The ⟨*range*⟩ is mandatory and must be non-empty. It is processed with `\foreach` with the option `parse=true`. Hereafter the elements are processed with `\fp_eval:n`. The ⟨*options*⟩ will only be applied to a slice if the number of the slice is in the ⟨*range*⟩. The ⟨*range*⟩ only makes sense for a key which is processed for each slice. For example, the ⟨*range*⟩ does not make sense for the key `middle`.

`/wheelchart/start angle={⟨angle⟩}` (no default, initially `90`)

This key defines the ⟨*angle*⟩ in degrees at which the first slice of the wheelchart starts.

`/wheelchart/start half={⟨angle⟩}` (default `90`)

This key sets the start angle such that the middle of the first slice of the wheelchart is positioned at ⟨*angle*⟩ in degrees.

`/wheelchart/title={⟨text⟩}` (no default, initially empty)

This key contains the ⟨*text*⟩ which will be placed above the wheelchart. The ⟨*text*⟩ is placed in a node. The x coordinate of this node is the x coordinate of the center of the wheelchart, which is defined by the key `at`. In general, this is *not* the same as the x coordinate of the center of the `local bounding box` around the wheelchart. The y coordinate of this node is at a value determined by the key `title sep` above the north of the `local bounding box` around the wheelchart. The style of this node is given as follows. First, the options `anchor=south,align=center` are given. Thereafter, the style of the key `title style` is added.

`/wheelchart/title left={⟨text⟩}` (no default, initially empty)

This key contains the ⟨*text*⟩ which will be placed above left of the wheelchart. The ⟨*text*⟩ is placed in a node. This node is placed at a value determined by the key `title left sep` above the north west of the `local bounding box` around the wheelchart. The style of this node is given as follows. First, the options `anchor=south west,align=left` are given. Thereafter, the style of the key `title left style` is added.

`/wheelchart/title left sep={`⟨*value*⟩`}` (no default, initially `0.5`)

The node where the contents of the key `title left` is placed is at ⟨*value*⟩ above the north west of the `local bounding box` around the wheelchart.

`/wheelchart/title left style={`⟨*options*⟩`}` (style, no default, initially empty)

This key accepts a list of keys which will be applied to the node where the contents of the key `title left` is placed.

`/wheelchart/title sep={`⟨*value*⟩`}` (no default, initially `0.5`)

The y coordinate of the node where the contents of the key `title` is placed is at ⟨*value*⟩ above the north of the `local bounding box` around the wheelchart.

`/wheelchart/title style={`⟨*options*⟩`}` (style, no default, initially empty)

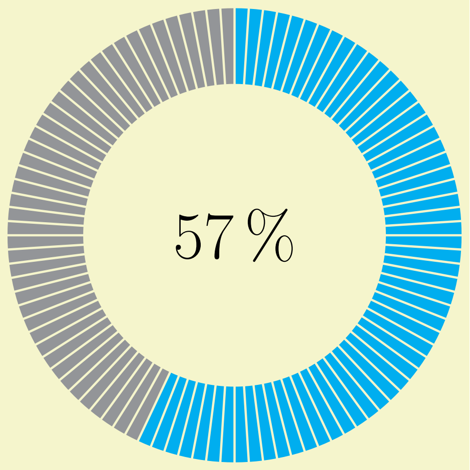
This key accepts a list of keys which will be applied to the node where the contents of the key `title` is placed.

`/wheelchart/total angle={`⟨*angle*⟩`}` (no default, initially `360`)

This key defines the total ⟨*angle*⟩ in degrees of the wheelchart.

`/wheelchart/total count={`⟨*number*⟩`}` (no default)

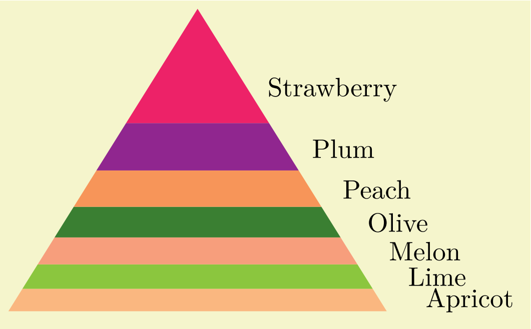
If this key is set then the number of slices of the wheelchart is determined by ⟨*number*⟩. Moreover, `\WCvarA` is defined as `1` and `\WCvarB` and `\WCvarC` are defined to be empty.



```
\usepackage {siunitx}
\begin{tikzpicture}
\def\n{57}
\wheelchart[
  gap=0.015,
  middle={\Huge\qty{\n}{\percent}},
  slices style=Gray,
  slices style{1,...,\n}=Cyan,
  total count=100
]{}
\end{tikzpicture}
```

`/wheelchart/triangle proportional area={`⟨*width*⟩`}{`⟨*height*⟩`}` (no default)

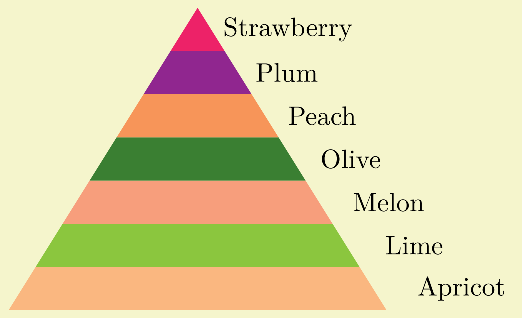
This key configures the plot such that a triangular shape is obtained. The value is proportional to the area and *not* to the height. Moreover, it sets `samples=2` and `wheel data pos=0.5`. The point $(0, 0)$ is at the top. This can be shifted with the key `at`.



```
\begin{tikzpicture}
\wheelchart[
  triangle proportional area={5}{4},
  value=1
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/triangle proportional height={⟨width⟩}{⟨height⟩}` (no default)

This key configures the plot such that a triangular shape is obtained. The value is proportional to the height and *not* to the area. Moreover, it sets `samples=2` and `wheel data pos=0.5`. The point $(0,0)$ is at the top. This can be shifted with the key `at`.



```
\begin{tikzpicture}
\wheelchart[
  triangle proportional height={5}{4},
  value=1
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/value={⟨value⟩}` (no default, initially `\WCvarA`)

This key defines the ⟨*value*⟩ which corresponds to the size of each slice of the wheelchart.

`/wheelchart/WClist⟨name⟩={⟨list⟩}` (no default)

This key locally defines a macro `\WClist`⟨*name*⟩ which gives the element in the ⟨*list*⟩ with as index `\WCcount` modulo the length of the ⟨*list*⟩. The ⟨*list*⟩ is expanded once and processed using a `clist`. In particular, blank arguments are ignored. An empty argument in the ⟨*list*⟩ can be obtained with `{}`. Items containing a `,` can be obtained by surrounding it with `{` and `}` such as `WClistA={{a,b},{c,d}}`.

If `\def\mylist{a,b,c}` and `WClistA=\mylist` then `\WClistA` gives `a,b,c` for each slice. On the other hand, if `WClistA/.expanded=\mylist` then `\WClistA` alternates between `a`, `b` and `c`.

`/wheelchart/wheel data={⟨text⟩}` (no default, initially empty)

This key contains the ⟨*text*⟩ which will be placed on top of each slice of the wheelchart. The ⟨*text*⟩ is placed in a node. The style of this node is given as follows. First, the option `align=left` is given. Thereafter, the style of the key `wheel data style` is added.

`/wheelchart/wheel data angle pos={⟨value⟩}` (no default, initially `0.5`)

`/wheelchart/wheel data angle shift={⟨angle⟩}` (no default, initially `0`)

`/wheelchart/wheel data pos={⟨value⟩}` (no default, initially `0.66`)

`/wheelchart/wheel data sep={⟨value⟩}` (no default, initially `0`)

These keys determine the position of the contents of the key `wheel data` similar as the corresponding keys for the key `data`.

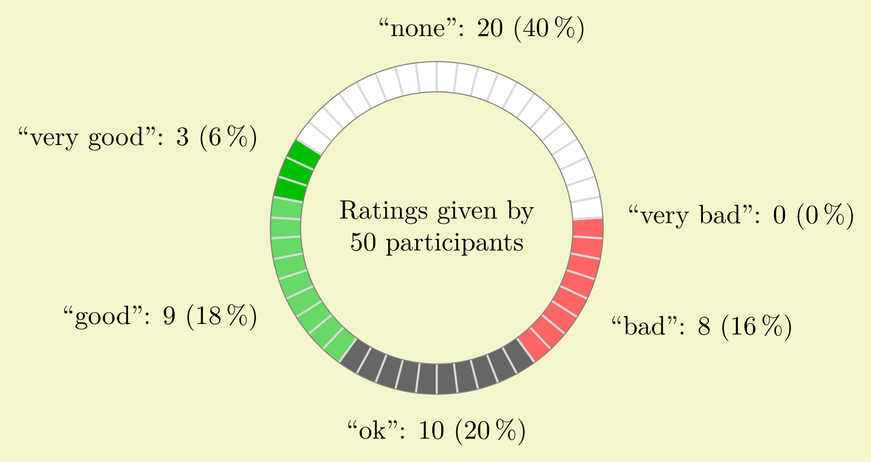
`/wheelchart/wheel data style={⟨options⟩}` (style, no default, initially empty)

This key accepts a list of keys which will be applied to the node where the contents of the key `wheel data` is placed.

`/wheelchart/wheel lines={⟨options⟩}` (style, no default, initially empty)

If this key is set then lines with the style determined by this key will be drawn inside the slices of the wheelchart. The number of these lines depends on the value of the key `value`.

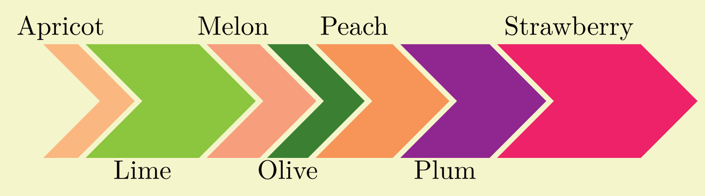
Below is the example from [5, Subsection 7.6] recreated with the package `wheelchart`.



```
\usepackage {siunitx}
\begin{tikzpicture}
\colorlet{good}{green!75!black}
\colorlet{bad}{red}
\colorlet{neutral}{black!60}
\colorlet{none}{white}
\wheelchart[
  anchor xsep=15,
  contour=gray,
  data=``\WCvarC'': \WCvarA{} (\WCperc),
  middle=Ratings given by\\\WCtotalnum{} participants,
  radius={1.8}{2.2},
  start half=270,
  wheel lines={black!15,thick}
]{%
  10/neutral/ok,
  9/good!60!white/good,
  3/good/very good,
  20/none/none,
  0/bad/very bad,
  8/bad!60!white/bad%
}
\end{tikzpicture}
```

`/wheelchart/xbar={⟨width⟩}{⟨height⟩}` (no default)

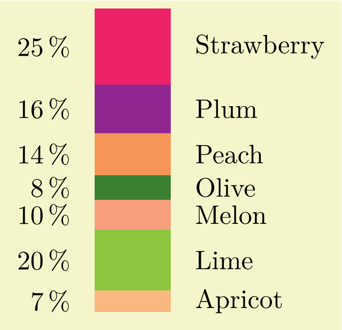
This key sets `domain=0:{⟨width⟩}`, `plot={{#1},{#2}}`, `radius={0}{⟨height⟩}`, `samples=2` and also `wheel data pos=0.5`. The point $(0, 0)$ is below left of the bar. This can be shifted with the key `at`. Note that since this key sets in particular the outer parts of the slices, keys such as `slices outer arc` must be placed *after* the key `xbar` to be applied.



```
\begin{tikzpicture}
\wheelchart[
  data pos{list}={1,0},
  data style={anchor=mid},
  gap polar=0.05,
  slices arrow={1}{0},
  xbar={8}{1.5}
]{\exampleforthismanual}
\end{tikzpicture}
```

`/wheelchart/ybar={⟨width⟩}{⟨height⟩}` (no default)

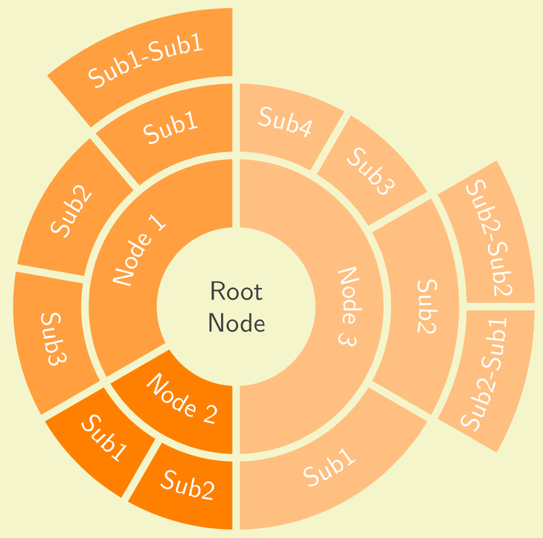
This key sets `domain=0:{⟨height⟩}`, `plot={{#2},{#1}}`, `radius={0}{⟨width⟩}`, `samples=2` and also `wheel data pos=0.5`. The point $(0, 0)$ is below left of the bar. This can be shifted with the key `at`.



```
\begin{tikzpicture}
\wheelchart[
  inner data=\WCperc,
  inner data style={anchor=east},
  ybar={1}{4}
]{\exampleforthismanual}
\end{tikzpicture}
```

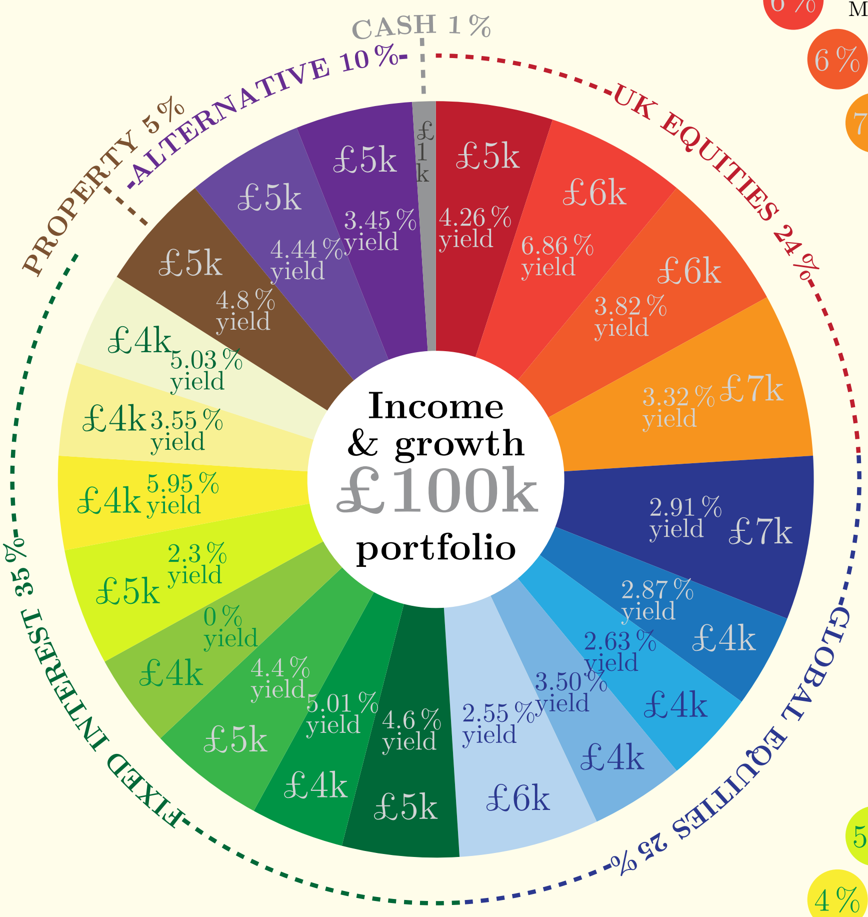
5 Additional examples

The following example is an answer to the question on https://tex.stackexchange.com/questions/433848/is-there-a-way-to-make-sunburst-charts-multi-level-pie-charts-in-latex.



```
\usepackage {etoolbox} \usepackage {listofitems} \usetikzlibrary {decorations.text}
\begin{tikzpicture}
\sffamily
\readlist\WCcolors{orange!50,orange!75,orange}
\pgfkeys{
  /wheelchart,
  arc data=\WCvarB,
  arc data dir={\WCmidangle<180?-1:1},
  arc data pos=0.5,
  arc data style={text color=white},
  counterclockwise,
  data=,
  gap,
  gap radius,
  slices style={
    /utils/exec={
      \ifdefempty{\WCvarB}{
        \def\WCcolor{none}
        \def\WCoverlay{true}
      }{
        \edef\WCcolor{\WCcolors[\fpeval{\WCmidangle<90?1:(\WCmidangle<210?2:(\WCmidangle<270?3:1))}]}
        \def\WCoverlay{false}
      }
    },
    fill=\WCcolor,
    overlay=\WCoverlay
  }
}
\wheelchart[
  middle=Root\\Node,
  middle style=darkgray,
  radius={1}{2}
]{2/Node 1,1/Node 2,3/Node 3}
\wheelchart[
  radius={2}{3}
]{4/Sub1,4/Sub2,4/Sub3,3/Sub1,3/Sub2,6/Sub1,6/Sub2,3/Sub3,3/Sub4}
\wheelchart[
  radius={3}{4}
]{4/Sub1-Sub1,20/,3/Sub2-Sub1,3/Sub2-Sub2,6/}
\end{tikzpicture}
```

The following example is an answer to the question on https://tex.stackexchange.com/questions/447920/pie-chart-with-color-palette-info-inside-and-legend.



5 %	Miton Multi-Cap Income
6 %	Schroder Income Maximiser
6 %	Trojan Income
7 %	CF Woodford Equity Income
7 %	Artemis Global Income
4 %	First State Global Listed Infrastructure
4 %	Lazard Global Listed Infrastructure
4 %	Legg Mason RARE Global Income
6 %	Newton Global Income
5 %	Henderson Strategic Bond
4 %	Invesco Perpetual Monthly Income Plus
5 %	Jupiter Strategic Bond
4 %	L&G All Stocks Index Linked Gilt Index
5 %	L&G Short Dated Sterling Corporate Bond Index
4 %	Royal London Short Duration Global High Yield Bond
4 %	TwentyFour Corporate Bond
4 %	TwentyFour Dynamic Bond
5 %	F&C Property Growth & Income
5 %	Aviva Multi Strategy Target Income
5 %	Invesco Perpetual Global Targeted Income
1 %	Cash

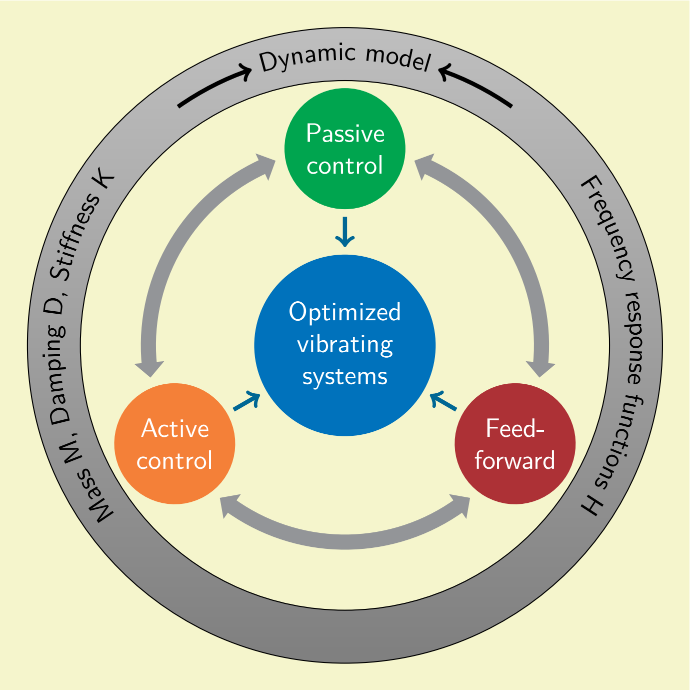
Portfolio income £3,785

or 3.79 %

Source: Whitechurch Securities

```
\usepackage {siunitx} \usetikzlibrary {decorations.text}
\begin{tikzpicture}
\ExplSyntaxOn
\seq_set_from_clist:Nn \l_tmpa_seq { 190~ 30~ 46 , 240~ 65~ 54 , 241~ 90~ 43 , 247~148~ 30 ,  43~ 56~144 ,  28~117~188 ,  40~170~225 ,
                                     119~179~225 , 181~212~239 ,   0~104~ 56 ,   0~148~ 69 ,  57~181~ 74 , 141~199~ 63 , 215~244~ 34 ,
                                     249~237~ 50 , 248~241~148 , 242~245~205 , 123~ 82~ 49 , 104~ 73~158 , 102~ 45~145 , 148~149~151 }
\seq_map_indexed_inline:Nn \l_tmpa_seq { \definecolor { slice#1 } { RGB } {#2} }
\ExplSyntaxOff
\definecolor{background}{RGB}{255 253 234}\definecolor{disc}{RGB}{ 15 119 188}
\definecolor{text1}{RGB}{209 211 212}\definecolor{text2}{RGB}{ 67  66  63}
\sisetup{group-separator={,},group-minimum-digits=4,text-series-to-math=true}
\fill[background] (-6.8,-8) rectangle (13.8,8);
\pgfkeys{/wheelchart,data=,radius={1.7}{5}}
\wheelchart[
  arc data{18,21}=/\bfseries/\WCvarE{} \WCperc,
  arc data pos=1.2,
  arc data style={text color=slice\WCcount},
  inner data{1,...,20}=\qty{\WCvarC}{\percent}\\[-4pt]yield,
  inner data pos=0.5,
  inner data style=\WCvarB,
  legend entry={
    \fill[slice\WCcount,shift={({int((\WCcount-1)/17)*4.5-3},0)}] ({45-Mod({\WCcount-1},17)*90/16}:10) circle[radius=0.4]
    node[\WCvarB,font=\large] {\WCperc}
    node[black,shift={(0.6,0)},anchor=west,font=\footnotesize,align=left,execute at begin node={\baselineskip=7pt}] {\WCvarD};
  },
  lines{18,21}=0.75,
  lines sep=0.1,
  lines style={slice\WCcount,dashed,ultra thick},
  middle=Income\\[-4pt]\& growth\\{\Huge\textcolor{slice21}{\pounds 100k}}\\portfolio,
  middle fill=white,
  middle style={font=\bfseries\Large},
  slices style=slice\WCcount,
  wheel data={\Large \pounds\WCvarA k},
  wheel data{21}=\pounds\\[-4pt]\WCvarA\\[-4pt]k,
  wheel data pos=0.8,
  wheel data style=\WCvarB
]{%
  5/text1/4.26/Miton Multi-Cap\\Income/,
  6/text1/6.86/Schroder Income\\Maximiser/,
  6/text1/3.82/Trojan\\Income/,
  7/text1/3.32/CF Woodford\\Equity Income/,
  7/text1/2.91/Artemis Global\\Income/,
  4/text1/2.87/First State Global\\Listed Infrastructure/,
  4/slice5/2.63/Lazard Global\\Listed Infrastructure/,
  4/slice5/3.50/Legg Mason RARE\\Global Income/,
  6/slice5/2.55/Newton Global\\Income/,
  5/text1/4.6/Henderson\\Strategic Bond/,
  4/text1/5.01/Invesco Perpetual\\Monthly Income Plus/,
  5/text1/4.4/Jupiter Strategic\\Bond/,
  4/slice11/0/L\&G All Stocks Index\\Linked Gilt Index/,
  5/slice11/2.3/L\&G Short Dated Sterling\\Corporate Bond Index/,
  4/slice11/5.95/Royal London Short Duration\\Global High Yield Bond/,
  4/slice10/3.55/TwentyFour\\Corporate Bond/,
  4/slice10/5.03/TwentyFour\\Dynamic Bond/,
  5/text1/4.8/F\&C Property Growth\\\& Income/PROPERTY,
  5/text1/4.44/Aviva Multi Strategy\\Target Income/,
  5/text1/3.45/Invesco\\Perpetual\\Global Targeted\\Income/,
  1/text2/0.01/Cash/CASH%
}
\pgfkeys{/wheelchart,arc={draw=\WCvarB,dashed,ultra thick},arc around text,arc data{1,2,3,5}=/\bfseries/\WCvarC{} \WCvarA{\,}
{\unit{\percent}},arc data pos=1.1,arc data style={text color=\WCvarB},arc pos=1.1,slices style={fill=none},value{5}=12}
\wheelchart{%
  24/slice1/UK EQUITIES,
  25/slice5/GLOBAL EQUITIES,
  35/slice10/FIXED INTEREST,
  3/none/,
  10/slice20/ALTERNATIVE,
  1/none/%
}
\fill[disc] (12,-5.5) circle[radius=1.7]
  node[white,font=\Large\bfseries,align=center] {Portfolio\\[-4pt]income\\\pounds\num{3785}\\[10pt]{\large or \qty{3.79}{\percent}}};
\node[rotate=270,anchor=north west] at (13.8,8) {\emph{Source: Whitechurch Securities}};
\end{tikzpicture}
```

The following example is an answer to the question on https://tex.stackexchange.com/questions/477310/cyclic-flowchart-in-tikz.


Dynamic model
Frequency response functions H
Mass M, Damping D, Stiffness K
Passive control
Optimized vibrating systems
Active control
Feed-forward

```
\usetikzlibrary {decorations.text}
\begin{tikzpicture}
\sffamily
\wheelchart[
  data=,
  middle=Optimized\\vibrating\\systems,
  middle fill=RoyalBlue,
  middle style=white,
  radius={1.2}{4},
  slices={(0,0) circle[radius=0.8];},
  slices style=\WCvarA,
  start half,
  value=1,
  wheel data=\WCvarB,
  wheel data pos=0.5,
  wheel data style={
    white,
    align=center
  }
]{%
  Green/Passive\\control,
  Maroon/Feed-\\forward,
  Orange/Active\\control%
}
\wheelchart[
  gap polar=25,
  radius={2.5}{2.7},
  slices end arrow={1}{-1},
  slices start arrow={1}{-1},
  slices style=Gray,
  total count=3
]{}
\foreach\n in {-30,90,210}{
  \draw[->,MidnightBlue,ultra thick]
    (\n:1.7)--(\n:1.3);
}
\fill[
  top color=Gray!50,
  bottom color=Gray,
  draw,
  even odd rule
] (0,0) circle[radius=3.5]
  circle[radius=4.2];
\wheelchart[
  arc{2}={
    <-,
    ultra thick
  },
  arc around text,
  arc data=~\WCvarA~,
  arc data pos=0.5,
  arc pos=0.5,
  data=,
  gap polar=10,
  radius={3.5}{4.2},
  slices style={fill=none},
  start half=180,
  value=1
]{%
  {Mass M, Damping D, Stiffness K},
  Dynamic model,
  Frequency response functions H,
  %
}
\end{tikzpicture}
```

6 Version history

Version 1.0 (2022/09/11) First version.

Version 2.0 (2023/12/03)

- The package now mainly uses LaTeX3 syntax.
- Improved the definition of the path of the slices.
- Many internal computations are now performed with `\fp_eval:n` instead of `pgfmath` for higher accuracy and to allow larger values. This applies in particular to the computation of `\WCpercentage`, `\WCpercentagerounded` and `\WCtotalnum`. Hence `\WCpercentagerounded` can be parsed by `siunitx` since its definition does not involve `\pgfmathprintnumberto` anymore and `\WCtotalnum` does not end with `.0` if it is an integer.
- The number of data which can be given to each slice of the wheelchart and accessed by `\WCvarA` and so on is not limited to 26 anymore.
- Added the macros `\WCcountdiscrete`, `\WCetocthelinkedname`, `\WCetocthelinkednumber`, `\WCetocthelinkedpage`, `\WCetocthename`, `\WCetocthenumber`, `\WCetocthenumberofpages`, `\WCetocthepage`, `\WClegend`, `\WClist`⟨*name*⟩ and `\`⟨*prefix*⟩⟨*name*⟩.
- Added the keys `after slices`, `arc`, `arc around text`, `arc data`, `arc data align`, `arc data angle pos`, `arc data angle shift`, `arc data dir`, `arc data pos`, `arc data sep`, `arc data style`, `arc first half`, `arc pos`, `arc second half`, `arc sep`, `before slices`, `caption left sep`, `caption sep`, `data angle pos`, `data pos`, `discrete`, `discrete factor`, `discrete partitioning`, `discrete pic`, `discrete sort`, `discrete space at borders`, `domain`, `etoc code`, `etoc count total pages`, `etoc level`, `etoc name`, `etoc use name`, `expand list items`, `for loop end`, `for loop start`, `gap max angle`, `gap radius`, `header`, `header prefix`, `inner data angle pos`, `inner data angle shift`, `inner data pos`, `inner plot`, `inner plot style`, `legend columns`, `legend only`, `legend row`, `lines angle pos`, `lines angle shift`, `lines ext dir`, `lines ext fixed left`, `lines ext fixed right`, `lines pos`, `outer plot`, `outer plot style`, `parse`, `plot`, `plot style`, `samples`, `separator columns`, `separator rows`, `slices angle pos`, `slices angle shift`, `slices arc inner end`, `slices arc inner end start`, `slices arc inner start`, `slices arc inner start end`, `slices arc match`, `slices arc outer end`, `slices arc outer end start`, `slices arc outer start`, `slices arc outer start end`, `slices end to`, `slices inner angle reduce`, `slices inner angle shift`, `slices inner arc`, `slices inner arc tangent`, `slices inner arrow`, `slices inner end angle shift`, `slices inner start angle shift`, `slices inner to`, `slices outer angle reduce`, `slices outer angle shift`, `slices outer arc`, `slices outer arc tangent`, `slices outer arrow`, `slices outer end angle shift`, `slices outer start angle shift`, `slices outer to`, `slices pos`, `slices scope`, `slices sep`, `slices start to`, `slices to`, `slice{`⟨*range*⟩`}`, `title left sep`, `title sep`, `triangle proportional area`, `triangle proportional height`, `WClist`⟨*name*⟩, `wheel data angle pos`, `wheel data angle shift`, `wheel data sep`, `xbar` and `ybar`.
- Added the possibility to give a ⟨*range*⟩ to the keys such that the options given to the key will only be applied to a slice if the number of the slice is in the ⟨*range*⟩.
- Added the possibility to give a ⟨*list*⟩ to the keys.
- The ⟨*wheelchart data*⟩ are not processed with `\foreach` anymore but instead with one of `\seq_set_split:Nee`, `\seq_set_split:Nen` or `\seq_set_split:Neo` depending on the keys `expand list` and `expand list items`. Thus syntax which is specific to how `\foreach` processes a list does not work anymore, such as the dots notation and the repeating of the last entry if some entry in the list has fewer entries than required.
- If the key `start angle` is set after the key `start half` then v1.0 preserved the setting of the key `start half`. In v2.0, the setting is determined by the key which is set last.

- In v1.0, the value of the key `data angle shift` was also applied to `inner data`, `lines` and `wheel data`. In v2.0, this is not the case anymore. Instead there are now separate keys `inner data angle shift`, `lines angle shift`, `wheel data angle shift` and also `arc data angle shift`.
- In v1.0, the key `data sep` was not applied if the key `lines ext` was used. In v2.0, this is not the case anymore.
- In v1.0, a negative value for the key `lines` was not applied. In v2.0, this is not the case anymore.

Version 3.0 (2024/03/08)

- Improved the parametrization of the slices in the case that no plot is used. In particular, the `arc` and `arc data` are placed with an arc if no plot is used whereas in v2.0, these were placed with a plot even if no plot was used. Also, the computation of `\WCdataangle` and `\WCmidangle` is more precise than in v2.0.
- Optimized the code. The compilation is faster than in v2.0.
- Added the commands `\WCangle`, `\WCcoordinate`, `\WCpoint` and `\WCradius`.
- Added the keys `arc around line`, `arc data expand`, `arc data line sep factor`, `slices Arrow`, `slices end`, `slices inner`, `slices outer` and `slices start`.
- Changed the definition of `\WCperc` in the key `arc data` so that `\WCperc` follows the arc or plot.
- Added the possibility that the contents of the key `arc data` consists of multiple lines separated by `\\`.
- Reduced the functionality of the keys `contour` and `middle fill` to require a fixed inner and outer radius for all slices.
- Removed the key `parse`. The values of applicable keys are parsed with `\pgfmathparse`. If a value should be parsed with `l3fp` then `\fpeval` can be used.
- In v2.0, the key `arc data angle shift` was not taken into account for the key `arc` in combination with the key `arc around text`. This is fixed in v3.0.
- In v2.0, the number of items for each slice in the ⟨*wheelchart data*⟩ which can be accessed with the macros `\WCvarA` and so on was determined by the number of items for the last slice. For example, `data{1}=\WCvarD` in combination with the ⟨*wheelchart data*⟩ `1/black/A/a,2/gray/B` was not possible with v2.0. This is not a limitation anymore with v3.0.

Version 4.0 (2024/07/28)

- Added the keys `arc data lines pos` and `arc data lines shift`.
- Solved an incompatibility if `\\` is used in a key such as `data` inside an environment such as `center`.

References

[1] Jake, *How can I produce a 'ring (or wheel) chart' like that on page 88 of the* PGF *manual?*, https://tex.stackexchange.com/questions/17898/how-can-i-produce-a-ring-or-wheel-chart-like-that-on-page-88-of-the-pgf-manu/18105#18105, 2011.

[2] Jens-Uwe Morawski, `piechartMP`, Manual for Preliminary Version, https://ctan.org/pkg/piechartmp, 2002.

[3] Dominique Rodriguez, Michael Sharpe, Herbert Voß, `pstricks-add` additionals Macros for `pstricks`, Manual for version 3.94, https://ctan.org/pkg/pstricks-add, 2023.

[4] Nicola L.C. Talbot, *User Manual for datatool bundle version 2.32*, https://ctan.org/pkg/datatool, 2019.

[5] Till Tantau, *The TikZ and* PGF *Packages*, Manual for version 3.1.10, https://ctan.org/pkg/pgf, 2023.

[6] Yuan Xu, *Drawing Pie Chart by using* `pgf-pie`, Manual for version 0.7, https://ctan.org/pkg/pgf-pie, 2022.

Index

A The source code

```
%% wheelchart.sty
%% Copyright 2022-2024 Matthias Floré
%
% This work may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either version 1.3c
% of this license or (at your option) any later version.
% The latest version of this license is in
%   http://www.latex-project.org/lppl.txt
% and version 1.3c or later is part of all distributions of LaTeX
% version 2005/12/01 or later.
%
% This work has the LPPL maintenance status `maintained'.
%
% The Current Maintainer of this work is Matthias Floré.
%
% This work consists of the files wheelchart.pdf, wheelchart.sty,
% wheelchart.tex and README.md.
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{tikz}
\usetikzlibrary{calc}
\ProvidesExplPackage{wheelchart}{2024/07/28}{4.0}{Diagrams with circular or other shapes using TikZ and LaTeX3}
```

A.1 Variables

```
\newcounter { g__wheelchart_WCcount_counter }

\bool_new:N \l__wheelchart_arc_bool
\bool_new:N \l__wheelchart_contour_bool
\bool_new:N \g__wheelchart_def_angle_radius_shift_bool
\bool_new:N \l__wheelchart_discrete_bool
\bool_new:N \l__wheelchart_etoc_use_name_bool
\bool_new:N \l__wheelchart_legend_only_bool
\bool_new:N \l__wheelchart_legend_row_bool
\bool_new:N \l__wheelchart_lines_ext_dir_bool
\bool_new:N \l__wheelchart_middle_fill_bool
\bool_new:N \l__wheelchart_pie_bool
```

```
\bool_new:N \l__wheelchart_plot_bool
\bool_new:N \l__wheelchart_slices_bool
\bool_new:N \l__wheelchart_wheel_lines_bool

\box_new:N \l__wheelchart_arc_data_box
\box_new:N \g__wheelchart_if_text_box

\clist_new:N \l__wheelchart_header_clist
\clist_new:N \g__wheelchart_slice_range_for_loop_clist
\clist_new:N \l__wheelchart_slice_range_local_clist

\fp_new:N \l__wheelchart_abs_half_angle_minus_new_angle_minus_gap_polar_fp
\fp_new:N \l__wheelchart_anchor_xsep_fp
\fp_new:N \l__wheelchart_anchor_ysep_fp
\fp_new:N \g__wheelchart_angle_fp
\fp_new:N \l__wheelchart_arc_around_line_fp
\fp_new:N \l__wheelchart_arc_data_angle_pos_fp
\fp_new:N \l__wheelchart_arc_data_angle_shift_fp
\fp_new:N \g__wheelchart_arc_data_aux_i_fp
\fp_new:N \g__wheelchart_arc_data_aux_ii_fp
\fp_new:N \l__wheelchart_arc_data_dir_fp
\fp_const:Nn \c__wheelchart_arc_data_end_factor_center_fp { 0.5 }
\fp_const:Nn \c__wheelchart_arc_data_end_factor_left_fp { 1 }
\fp_const:Nn \c__wheelchart_arc_data_end_factor_right_fp { 0 }
\fp_new:N \l__wheelchart_arc_data_line_sep_factor_fp
\fp_new:N \l__wheelchart_arc_data_lines_pos_fp
\fp_new:N \l__wheelchart_arc_data_lines_shift_fp
\fp_new:N \l__wheelchart_arc_data_pos_fp
\fp_new:N \l__wheelchart_arc_data_radius_plot_false_fp
\fp_new:N \l__wheelchart_arc_data_sep_fp
\fp_new:N \g__wheelchart_arc_data_slice_length_fp
\fp_new:N \l__wheelchart_arc_data_start_angle_plot_false_fp
\fp_const:Nn \c__wheelchart_arc_data_start_factor_center_fp { -0.5 }
\fp_const:Nn \c__wheelchart_arc_data_start_factor_left_fp { 0 }
\fp_const:Nn \c__wheelchart_arc_data_start_factor_right_fp { -1 }
\fp_new:N \l__wheelchart_arc_data_text_pos_fp
\fp_new:N \l__wheelchart_arc_data_total_angle_plot_false_fp
\fp_new:N \l__wheelchart_arc_pos_fp
```

```
\fp_new:N \l__wheelchart_arc_radius_fp
\fp_new:N \l__wheelchart_arc_sep_fp
\fp_new:N \l__wheelchart_arc_start_angle_fp
\fp_new:N \l__wheelchart_coord_determinant_fp
\fp_new:N \g__wheelchart_coord_x_fp
\fp_new:N \g__wheelchart_coord_y_fp
\fp_new:N \l__wheelchart_counter_or_clockwise_fp
\fp_new:N \g__wheelchart_def_angle_angle_fp
\fp_new:N \l__wheelchart_discrete_end_length_fp
\fp_new:N \l__wheelchart_discrete_factor_fp
\fp_new:N \l__wheelchart_discrete_inner_length_fp
\fp_new:N \l__wheelchart_discrete_level_fp
\fp_new:N \l__wheelchart_discrete_level_end_length_fp
\fp_new:N \l__wheelchart_discrete_level_start_length_fp
\fp_new:N \l__wheelchart_discrete_levels_sum_fp
\fp_new:N \l__wheelchart_discrete_outer_length_fp
\fp_new:N \l__wheelchart_discrete_start_length_fp
\fp_new:N \l__wheelchart_discrete_sublevel_end_length_fp
\fp_new:N \l__wheelchart_discrete_sublevel_start_length_fp
\fp_new:N \l__wheelchart_gap_fp
\fp_new:N \l__wheelchart_gap_max_angle_fp
\fp_new:N \l__wheelchart_gap_max_angle_def_fp
\fp_new:N \l__wheelchart_gap_polar_fp
\fp_new:N \l__wheelchart_gap_radius_fp
\fp_new:N \g__wheelchart_half_ex_over_one_cm_fp
\fp_new:N \l__wheelchart_inner_data_angle_pos_fp
\fp_new:N \l__wheelchart_inner_data_angle_shift_fp
\fp_new:N \l__wheelchart_inner_data_pos_fp
\fp_new:N \l__wheelchart_inner_data_sep_fp
\fp_new:c { g__wheelchart_inner~end_x_fp }
\fp_new:c { g__wheelchart_inner~end_y_fp }
\fp_new:N \l__wheelchart_inner_radius_fp
\fp_new:c { g__wheelchart_inner~start_x_fp }
\fp_new:c { g__wheelchart_inner~start_y_fp }
\fp_new:N \l__wheelchart_lines_fp
\fp_new:N \l__wheelchart_lines_angle_pos_fp
\fp_new:N \l__wheelchart_lines_angle_shift_fp
\fp_new:N \l__wheelchart_lines_ext_fp
\fp_new:N \l__wheelchart_lines_ext_dirsep_fp
```

```
\fp_new:N \l__wheelchart_lines_ext_fixed_left_fp
\fp_new:N \l__wheelchart_lines_ext_fixed_right_fp
\fp_new:N \l__wheelchart_lines_pos_fp
\fp_new:N \l__wheelchart_lines_sep_fp
\fp_new:N \g__wheelchart_new_angle_fp
\fp_new:c { g__wheelchart_outer~end_x_fp }
\fp_new:c { g__wheelchart_outer~end_y_fp }
\fp_new:N \l__wheelchart_outer_radius_fp
\fp_new:c { g__wheelchart_outer~start_x_fp }
\fp_new:c { g__wheelchart_outer~start_y_fp }
\fp_new:N \g__wheelchart_previous_coord_x_fp
\fp_new:N \g__wheelchart_previous_coord_y_fp
\fp_new:N \l__wheelchart_samples_fp
\fp_new:N \l__wheelchart_slices_angle_fp
\fp_new:N \l__wheelchart_slices_angle_pos_fp
\fp_new:N \l__wheelchart_slices_angle_shift_fp
\fp_new:N \l__wheelchart_slices_arc_A_fp
\fp_new:N \l__wheelchart_slices_arc_A_abs_fp
\fp_new:N \l__wheelchart_slices_arc_angle_fp
\fp_new:N \l__wheelchart_slices_arc_B_fp
\fp_new:N \l__wheelchart_slices_arc_coord_fp
\fp_new:N \l__wheelchart_slices_arc_rotate_fp
\fp_new:N \l__wheelchart_slices_arrow_A_fp
\fp_new:N \l__wheelchart_slices_arrow_B_fp
\fp_new:N \l__wheelchart_slices_arrow_coord_fp
\fp_new:N \l__wheelchart_slices_inner_end_angle_shift_fp
\fp_new:N \l__wheelchart_slices_inner_start_angle_shift_fp
\fp_new:N \g__wheelchart_slices_orientation_fp
\fp_new:N \l__wheelchart_slices_orientation_new_angle_fp
\fp_new:N \l__wheelchart_slices_outer_end_angle_shift_fp
\fp_new:N \l__wheelchart_slices_outer_start_angle_shift_fp
\fp_new:N \l__wheelchart_slices_pos_fp
\fp_new:N \l__wheelchart_slices_sep_fp
\fp_new:N \l__wheelchart_start_angle_fp
\fp_new:N \l__wheelchart_total_angle_fp
\fp_new:N \l__wheelchart_total_count_fp
\fp_new:N \l__wheelchart_wheel_data_angle_pos_fp
\fp_new:N \l__wheelchart_wheel_data_angle_shift_fp
\fp_new:N \l__wheelchart_wheel_data_pos_fp
```

```
\fp_new:N \l__wheelchart_wheel_data_sep_fp

\int_new:N \g__wheelchart_discrete_count_int
\int_new:N \l__wheelchart_discrete_levels_int
\int_new:N \l__wheelchart_discrete_partitioning_first_index_int
\int_new:N \l__wheelchart_discrete_partitioning_second_index_int
\int_new:N \l__wheelchart_discrete_sort_int
\int_new:N \l__wheelchart_discrete_space_at_borders_int
\int_set:Nn \l__wheelchart_discrete_space_at_borders_int { -1 }
\int_new:N \l__wheelchart_discrete_sublevels_int
\int_new:N \l__wheelchart_etoc_count_total_pages_int
\int_new:N \l__wheelchart_legend_columns_int
\int_new:N \l__wheelchart_legend_rows_int
\int_new:N \l__wheelchart_lines_ext_bottom_dir_int
\int_new:N \l__wheelchart_lines_ext_dir_int
\int_const:Nn \c__wheelchart_lines_ext_dir_left_int { -1 }
\int_const:Nn \c__wheelchart_lines_ext_dir_right_int { 1 }
\int_new:N \l__wheelchart_lines_ext_top_dir_int
\int_new:N \l__wheelchart_max_list_items_int

\regex_const:Nn \c__wheelchart_braces_regex { \{(.+)\} }
\regex_const:Nn \c__wheelchart_key_braces_regex { [\w\s]+\{(.+)\} }

\seq_new:N \l__wheelchart_arc_data_seq
\seq_new:N \l__wheelchart_discrete_coefficients_first_seq
\seq_new:N \l__wheelchart_discrete_coefficients_second_seq
\seq_new:N \l__wheelchart_discrete_points_seq
\seq_new:N \l__wheelchart_list_seq
\seq_new:N \l__wheelchart_list_items_seq

\tl_new:N \WClegend
\tl_const:Nn \c__wheelchart_arc_around_text_aux_tl
  {
    sign ( \l__wheelchart_arc_data_dir_fp ) *
      (
        \cs:w c__wheelchart_arc_data_start_factor_\l__wheelchart_arc_data_align_tl _fp \cs_end:
        + \cs:w c__wheelchart_arc_data_end_factor_\l__wheelchart_arc_data_align_tl _fp \cs_end:
      )
  }
\tl_new:N \l__wheelchart_arc_data_align_tl
\tl_new:N \l__wheelchart_data_anchor_tl
\tl_new:N \l__wheelchart_etoc_level_tl
\tl_new:N \l__wheelchart_etoc_name_tl
\tl_new:N \l__wheelchart_expand_list_tl
\tl_new:N \l__wheelchart_expand_list_items_tl
\tl_new:N \l__wheelchart_inner_plot_variable_tl
\tl_new:N \l__wheelchart_key_name_tl
\tl_new:N \l__wheelchart_key_range_tl
\tl_new:N \l__wheelchart_legend_row_tl
\tl_new:N \g__wheelchart_name_tl
\tl_new:N \l__wheelchart_outer_plot_variable_tl
\tl_new:N \l__wheelchart_plot_variable_tl
\tl_new:N \l__wheelchart_slice_range_index_tl
\tl_new:N \l__wheelchart_slices_tl
\tl_new:N \g__wheelchart_totalcount_tl
\tl_new:N \l__wheelchart_type_tl
\tl_set:Nn \l__wheelchart_type_tl { default }
```

A.2 Functions

```
\cs_generate_variant:Nn \seq_set_split:Nnn { Nen , Neo , Nnf }
\cs_generate_variant:Nn \tl_build_put_right:Nn { NV }

\cs_new_protected:Npn \__wheelchart_arc_around_text_plot_false:nn #1#2
  {
    \fp_set:Nn \l__wheelchart_arc_start_angle_fp
      {
        \__wheelchart_def_angle_plot_false:nnnnn
          { \WCcount }
          { \l__wheelchart_arc_data_angle_pos_fp }
          {
            ( 0.5 * \c__wheelchart_arc_around_text_aux_tl + #2 - 0.5 ) * \g__wheelchart_arc_data_aux_ii_fp
            + \l__wheelchart_arc_data_angle_shift_fp
          }
          { \l__wheelchart_arc_pos_fp }
          { \l__wheelchart_arc_sep_fp + \g__wheelchart_half_ex_over_one_cm_fp }
```

```
   }
  \path
   [ draw , / wheelchart / arc_style , / wheelchart / arc_#1_half ]
   ( \fp_use:N \l__wheelchart_arc_start_angle_fp \c_colon_str \fp_use:N \l__wheelchart_arc_radius_fp )
   arc
    [
     start~angle = \fp_use:N \l__wheelchart_arc_start_angle_fp ,
     end~angle =
      \__wheelchart_def_angle_plot_false:nnnnn
       { \WCcount }
       {#2}
       { 0 }
       { \l__wheelchart_arc_pos_fp }
       { \l__wheelchart_arc_sep_fp + \g__wheelchart_half_ex_over_one_cm_fp } ,
     radius = \fp_use:N \l__wheelchart_arc_radius_fp
    ]
   ;
 }

\cs_new_protected:Npn \__wheelchart_arc_around_text_plot_true:nnn #1#2#3
 {
  \__wheelchart_convex_comb_coord_plot:nnnnnnn
   { draw , / wheelchart / arc_style , / wheelchart / arc_#1_half }
   { 1 }
   { 0 }
   {
    \l__wheelchart_plot_variable_tl *
    \fp_eval:n
     {
      \l__wheelchart_arc_data_angle_pos_fp + 0.5 * \g__wheelchart_arc_data_aux_ii_fp *
       ( \c__wheelchart_arc_around_text_aux_tl + #2 )
     }
    + (#3) * ( 1 - \l__wheelchart_plot_variable_tl )
   }
   { \l__wheelchart_plot_variable_tl * \l__wheelchart_arc_data_angle_shift_fp }
   { \l__wheelchart_arc_pos_fp }
   { \l__wheelchart_arc_sep_fp }
 }
```

```
\cs_new_protected:Npn \__wheelchart_caption_and_title:nnnnn #1#2#3#4#5
  {
    \__wheelchart_if_text:nnn {#1} { o }
      {
        \node [ anchor = #2 , align = #3 , / wheelchart / #1_style ]
          at ( $ (#4) + ( 0 , { #5 * ( \pgfkeysvalueof { / wheelchart / #1~sep } ) } ) $ )
          { \pgfkeysvalueof { / wheelchart / #1 } } ;
      }
  }

\cs_new:Npn \__wheelchart_convex_comb_coord_aux:n #1 { (#1) }

\cs_generate_variant:Nn \__wheelchart_convex_comb_coord_aux:n { o }

\cs_new:Npn \__wheelchart_convex_comb_coord_def:nnnn #1#2#3#4
  {
    $
      (
        \__wheelchart_inner_plot:nn
          {
            \fp_eval:n
              {
                ( 1 - (#1) ) * \cs:w g__wheelchart_slice_inner_start_angle_\WCcount _fp \cs_end:
                + (#1) * \cs:w g__wheelchart_slice_inner_end_angle_\WCcount _fp \cs_end:
                + \l__wheelchart_counter_or_clockwise_fp * (#2)
              }
          }
          { \fp_eval:n { \cs:w g__wheelchart_inner_radius_\WCcount _fp \cs_end: - (#4) } }
      )
      ! { \fp_eval:n {#3} } !
      (
        \__wheelchart_outer_plot:nn
          {
            \fp_eval:n
              {
                ( 1 - (#1) ) * \cs:w g__wheelchart_slice_outer_start_angle_\WCcount _fp \cs_end:
                + (#1) * \cs:w g__wheelchart_slice_outer_end_angle_\WCcount _fp \cs_end:
                + \l__wheelchart_counter_or_clockwise_fp * (#2)
```

```
       }
     }
     { \fp_eval:n { \cs:w g__wheelchart_outer_radius_\WCcount _fp \cs_end: + (#4) } }
   )
  $%note the () around the #i's to keep these contents together
 }

\cs_new_protected:Npn \__wheelchart_convex_comb_coord_plot_aux:nnnn #1#2#3#4
 {
  \path [#1] plot
   [
    domain = {#2} \c_colon_str {#3} ,
    samples = \fp_use:c { g__wheelchart_samples_\WCcount _fp } ,
    variable = \l__wheelchart_plot_variable_tl
   ]
   ( {#4} ) ;
 }

\cs_generate_variant:Nn \__wheelchart_convex_comb_coord_plot_aux:nnnn { nnno }

\cs_new_protected:Npn \__wheelchart_convex_comb_coord_plot:nnnnnnn #1#2#3#4#5#6#7
 {
  \__wheelchart_convex_comb_coord_plot_aux:nnno {#1} {#2} {#3}
   { \__wheelchart_convex_comb_coord_def:nnnn {#4} {#5} {#6} { \g__wheelchart_half_ex_over_one_cm_fp + (#7) } }
 }

\cs_new_protected:Npn \__wheelchart_def_angle_aux:
 {
  \fp_gset:Nn \g__wheelchart_def_angle_angle_fp
   {
    atand
     (
      (
       \pgf@xx * ( \y { l__wheelchart_def_angle_2 } - \y { l__wheelchart_def_angle_1 } )
       - \pgf@xy * ( \x { l__wheelchart_def_angle_2 } - \x { l__wheelchart_def_angle_1 } )
      )
      / \l__wheelchart_coord_determinant_fp ,
      (
              \pgf@yy * ( \x { l__wheelchart_def_angle_2 } - \x { l__wheelchart_def_angle_1 } )
              - \pgf@yx * ( \y { l__wheelchart_def_angle_2 } - \y { l__wheelchart_def_angle_1 } )
            )
            / \l__wheelchart_coord_determinant_fp
          )
      }%\pgf@xx and so on are necessary if an option such as [x={(-0.5,0)},y={(0,0.5)}] is given to the tikzpicture
  }

\cs_new_protected:Npn \__wheelchart_def_angle:nnnn #1#2#3#4
  {
    \bool_if:NTF \l__wheelchart_plot_bool
      {
        \path let
          \p { l__wheelchart_def_angle_1 } = \__wheelchart_point_plot_true:nnnnn { \WCcount } {#1} {#2} { 0 } {#4} ,
          \p { l__wheelchart_def_angle_2 } = \__wheelchart_point_plot_true:nnnnn { \WCcount } {#1} {#2} { 1 } {#4}
          in
            [
              / utils / exec =
                {
                  \bool_gset:Nn \g__wheelchart_def_angle_radius_shift_bool
                    {
                      \fp_compare_p:n
                        {
                          \y { l__wheelchart_def_angle_2 } - \y { l__wheelchart_def_angle_1 } == 0
                          &&
                          \x { l__wheelchart_def_angle_2 } - \x { l__wheelchart_def_angle_1 } == 0
                        }
                    }
                  \bool_if:NF \g__wheelchart_def_angle_radius_shift_bool
                    { \__wheelchart_def_angle_aux: }
                }
            ]
          ;
        \bool_if:NT \g__wheelchart_def_angle_radius_shift_bool
          {
            \path let
              \p { l__wheelchart_def_angle_1 } =
                \__wheelchart_point_plot_true:nnnnn { \WCcount } {#1} {#2} { 0 }
                  { \fp_eval:n { (#4) + 1 / \cs:w g__wheelchart_samples_\WCcount _fp \cs_end: } } ,
       \p { l__wheelchart_def_angle_2 } =
        \__wheelchart_point_plot_true:nnnnn { \WCcount } {#1} {#2} { 1 }
         { \fp_eval:n { (#4) + 1 / \cs:w g__wheelchart_samples_\WCcount _fp \cs_end: } }
       in [ / utils / exec = { \__wheelchart_def_angle_aux: } ]
       ;
     }
    \pgfmathparse { Mod ( \fp_use:N \g__wheelchart_def_angle_angle_fp , 360 ) }
   }
   {
    \pgfmathparse { Mod ( \__wheelchart_def_angle_plot_false:nnnnn { \WCcount } {#1} {#2} {#3} {#4} , 360 ) }
   }
 }

\cs_new:Npn \__wheelchart_def_angle_plot_false_aux_angle:nn #1#2
 {
  ( 1 - (#2) ) *
   (
    \cs:w g__wheelchart_slice_outer_start_angle_#1_fp \cs_end:
    - \cs:w g__wheelchart_slice_inner_start_angle_#1_fp \cs_end:
   )
  + (#2) * ( \cs:w g__wheelchart_slice_outer_end_angle_#1_fp \cs_end: - \cs:w g__wheelchart_slice_inner_end_angle_#1_fp \cs_end: )
 }

\cs_new:Npn \__wheelchart_def_angle_plot_false:nnnnn #1#2#3#4#5
 {
  \fp_eval:n
   {
    \fp_compare:nNnTF { \__wheelchart_def_angle_plot_false_aux_angle:nn {#1} {#2} } = { 0 }
     { 0 }
     {
      asind
       (
        (
         sqrt
          (
           ( \cs:w g__wheelchart_outer_radius_#1_fp \cs_end: + \cs:w g__wheelchart_inner_radius_#1_fp \cs_end: ) ^ 2
           - \cs:w g__wheelchart_outer_radius_#1_fp \cs_end: * \cs:w g__wheelchart_inner_radius_#1_fp \cs_end: *
            (
             2 + 2 * cosd ( \__wheelchart_def_angle_plot_false_aux_angle:nn {#1} {#2} )
             + \cs:w g__wheelchart_outer_radius_#1_fp \cs_end: * \cs:w g__wheelchart_inner_radius_#1_fp \cs_end:
              *
               (
                (
                 sind ( \__wheelchart_def_angle_plot_false_aux_angle:nn {#1} {#2} )
                 / \__wheelchart_def_radius:nnn {#1} {#4} {#5}
                )
                ^ 2
               )
            )
          )
         + \cs:w g__wheelchart_inner_radius_#1_fp \cs_end: *
          (
           \cs:w g__wheelchart_inner_radius_#1_fp \cs_end:
           - \cs:w g__wheelchart_outer_radius_#1_fp \cs_end:
            * cosd ( \__wheelchart_def_angle_plot_false_aux_angle:nn {#1} {#2} )
          )
          / \__wheelchart_def_radius:nnn {#1} {#4} {#5}
        )
        * \cs:w g__wheelchart_outer_radius_#1_fp \cs_end: * sind ( \__wheelchart_def_angle_plot_false_aux_angle:nn {#1} {#2} )
        /
        (
         ( \cs:w g__wheelchart_outer_radius_#1_fp \cs_end: + \cs:w g__wheelchart_inner_radius_#1_fp \cs_end: ) ^ 2
         - 2 * \cs:w g__wheelchart_outer_radius_#1_fp \cs_end: * \cs:w g__wheelchart_inner_radius_#1_fp \cs_end:
          * ( 1 + cosd ( \__wheelchart_def_angle_plot_false_aux_angle:nn {#1} {#2} ) )
        )
       )
     }
    + ( 1 - (#2) ) * ( \cs:w g__wheelchart_slice_inner_start_angle_#1_fp \cs_end: )
    + (#2) * ( \cs:w g__wheelchart_slice_inner_end_angle_#1_fp \cs_end: )
    + \l__wheelchart_counter_or_clockwise_fp * (#3)
   }
 }

\cs_new_protected:Npn \__wheelchart_def_coord:nnnn #1#2#3#4
 {
  \path let \p { l__wheelchart_coord } =
   ( \cs:w __wheelchart_#2_plot:nn \cs_end: {#4} { \fp_use:c { g__wheelchart_#2_radius_\WCcount _fp } } )
   in
    [
     / utils / exec =
      {
       \fp_gset:cn { g__wheelchart_#1_x_fp }
        {
         ( \pgf@yy * \x { l__wheelchart_coord } - \pgf@yx * \y { l__wheelchart_coord } ) / \l__wheelchart_coord_determinant_fp
        }
       \fp_gset:cn { g__wheelchart_#1_y_fp }
        {
         ( \pgf@xx * \y { l__wheelchart_coord } - \pgf@xy * \x { l__wheelchart_coord } ) / \l__wheelchart_coord_determinant_fp
        }
      }
    ]
    coordinate ( g__wheelchart_slice_\WCcount _#2~#3_coordinate ) at ( \p { l__wheelchart_coord } )
   ;
 }

\cs_new_protected:Npn \__wheelchart_def_fp:nn #1#2
 {
  \pgfmathparse { \pgfkeysvalueof { / wheelchart / #2 } }
  \fp_set:cn { l__wheelchart_#1_fp } { \pgfmathresult }
 }

\cs_new_protected:Npn \__wheelchart_def_gap:nn #1#2
 {
  \fp_gset:cn { g__wheelchart_#1_gap_\WCcount _fp }
   {
    (#2) * sind ( min ( \l__wheelchart_abs_half_angle_minus_new_angle_minus_gap_polar_fp , \l__wheelchart_gap_max_angle_def_fp ) )
    <
    \l__wheelchart_gap_fp
    ?
    min ( \cs:w g__wheelchart_abs_half_angle_minus_new_angle_\WCcount _fp \cs_end: , \l__wheelchart_gap_max_angle_def_fp )
    :
    asind ( min ( \l__wheelchart_gap_fp / ( (#2) + 1 - sign (#2) ) , 1 ) ) + \l__wheelchart_gap_polar_fp
    %note the min ( ... , 1 ) so that the asind is always defined
    %also note the + 1 - sign (#2) so that the denominator is also nonzero if #2 = 0
   }
  }

\cs_new_protected:Npn \__wheelchart_def_inner_radius:
  {
    \bool_if:NTF \l__wheelchart_pie_bool
      { \fp_set:Nn \l__wheelchart_inner_radius_fp { 0 } }
      {
        \pgfmathparse { \pgfkeysvalueof { / wheelchart / inner~radius } }
        \fp_set:Nn \l__wheelchart_inner_radius_fp { \pgfmathresult + \l__wheelchart_gap_radius_fp }
      }
  }

\cs_new_protected:Npn \__wheelchart_def_orientation:
  {%determine the orientation, this is necessary even if no plot is used, for example if
  %inner radius > outer radius then \g__wheelchart_slices_orientation_fp is different from
  %\l__wheelchart_counter_or_clockwise_fp
    \fp_set:Nn \l__wheelchart_slices_orientation_new_angle_fp
      {
        \g__wheelchart_angle_fp +
        (
          \g__wheelchart_new_angle_fp == \g__wheelchart_angle_fp
          ?
          \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_total_angle_fp
          :
          \g__wheelchart_new_angle_fp - \g__wheelchart_angle_fp
        )
        / \cs:w g__wheelchart_samples_1_fp \cs_end:
      }
    \path let
      \p { l__wheelchart_slices_orientation_1 } =
        (
          \__wheelchart_outer_plot:nn
            { \fp_use:N \g__wheelchart_angle_fp }
            { \fp_use:c { g__wheelchart_outer_radius_1_fp } }
        ) ,
      \p { l__wheelchart_slices_orientation_2 } =
        (
          \__wheelchart_outer_plot:nn
    { \fp_use:N \l__wheelchart_slices_orientation_new_angle_fp }
    { \fp_use:c { g__wheelchart_outer_radius_1_fp } }
  ) ,
 \p { l__wheelchart_slices_orientation_3 } =
  (
   \__wheelchart_inner_plot:nn
    { \fp_use:N \l__wheelchart_slices_orientation_new_angle_fp }
    { \fp_use:c { g__wheelchart_inner_radius_1_fp } }
  ) ,
 \p { l__wheelchart_slices_orientation_4 } =
  (
   \__wheelchart_inner_plot:nn
    { \fp_use:N \g__wheelchart_angle_fp }
    { \fp_use:c { g__wheelchart_inner_radius_1_fp } }
  )
 in
  [
   / utils / exec =
    {
     \fp_gset:Nn \g__wheelchart_slices_orientation_fp
      {
       sign
        (
         0.1 * \y { l__wheelchart_slices_orientation_1 }
         * ( \x { l__wheelchart_slices_orientation_4 } - \x { l__wheelchart_slices_orientation_2 } )
         + 0.1 * \y { l__wheelchart_slices_orientation_2 }
         * ( \x { l__wheelchart_slices_orientation_1 } - \x { l__wheelchart_slices_orientation_3 } )
         + 0.1 * \y { l__wheelchart_slices_orientation_3 }
         * ( \x { l__wheelchart_slices_orientation_2 } - \x { l__wheelchart_slices_orientation_4 } )
         + 0.1 * \y { l__wheelchart_slices_orientation_4 }
         * ( \x { l__wheelchart_slices_orientation_3 } - \x { l__wheelchart_slices_orientation_1 } )
        )
       * sign ( \l__wheelchart_coord_determinant_fp )
      }
    }
  ]
;%the terms are multiplied with 0.1 to try to avoid an overflow
\fp_compare:nNnT { \g__wheelchart_slices_orientation_fp } = { 0 }
 { \fp_gset_eq:NN \g__wheelchart_slices_orientation_fp \l__wheelchart_counter_or_clockwise_fp }
```

```
  }

\cs_new_protected:Npn \__wheelchart_def_outer_radius:
  {
    \pgfmathparse { \pgfkeysvalueof { / wheelchart / outer~radius } }
    \fp_set:Nn \l__wheelchart_outer_radius_fp { \pgfmathresult - \l__wheelchart_gap_radius_fp }
  }

\cs_new:Npn \__wheelchart_def_radius:nnn #1#2#3
  {
    \fp_eval:n
      {
        ( 1 - (#2) ) * ( \cs:w g__wheelchart_inner_radius_#1_fp \cs_end: - (#3) )
        + (#2) * ( \cs:w g__wheelchart_outer_radius_#1_fp \cs_end: + (#3) )
      }
  }

\cs_new_protected:Npn \__wheelchart_def_slice_angle:nnnn #1#2#3#4
  {
    \fp_gzero_new:c { g__wheelchart_slice_#1_#2_angle_\WCcount _fp }
    \bool_if:NTF \l__wheelchart_plot_bool
      {
        \fp_gset:cn { g__wheelchart_slice_#1_#2_angle_\WCcount _fp }
          {
            \cs:w g__wheelchart_#3angle_fp \cs_end: +
              (
                \l__wheelchart_counter_or_clockwise_fp *
                  ( (#4) * \cs:w g__wheelchart_#1_gap_\WCcount _fp \cs_end: + \cs:w l__wheelchart_slices_#1_#2_angle_shift_fp \cs_end: )
              )
          }
      }
      {
        \fp_gset:cn { g__wheelchart_slice_#1_#2_angle_\WCcount _fp }
          {
            \cs:w g__wheelchart_#3angle_fp \cs_end: +
              (
                \l__wheelchart_counter_or_clockwise_fp *
                  (
          (#4) * \cs:w g__wheelchart_#1_gap_\WCcount _fp \cs_end: + \cs:w l__wheelchart_slices_#1_#2_angle_shift_fp \cs_end:
          - asind
           (
            \cs:w g__wheelchart_inner_radius_\WCcount _fp \cs_end:
            * sind ( \cs:w l__wheelchart_slices_#1_#2_angle_shift_fp \cs_end: )
            / \cs:w g__wheelchart_outer_radius_\WCcount _fp \cs_end:
           )
         )
       )
     }
   }
 }

\cs_new_protected:Npn \__wheelchart_def_slice_keys:n #1
 {
  {%note the double braces {{...}} so that the contents is in a group
  %and in particular, pgfkeys which are specific to the current slice are local for this slice
   \clist_if_in:NVT \l__wheelchart_slice_range_local_clist \WCcount
    {
     \pgfkeys { / wheelchart , slice_final /. expanded = { \exp_not:v { l__wheelchart_slice_\WCcount _keys_clist } } }
     \pgfkeys{ / wheelchart , slice_final_style }
    }
   #1
  }
 }

\cs_new_protected:Npn \__wheelchart_def_WClegend:
 {
  \int_set:Nn \l__wheelchart_legend_columns_int
   { \fp_eval:n { ceil ( \WCtotalcount / ceil ( \WCtotalcount / ( \pgfkeysvalueof { / wheelchart / legend~columns } ) ) ) } }
  \tl_build_begin:N \WClegend
   \int_compare:nNnTF { \l__wheelchart_legend_columns_int } = { 1 }
    { \int_step_inline:nnn { 2 } { \WCtotalcount } { \__wheelchart_legend_append:nn {##1} { \\ } } }
    {
     \int_set:Nn \l__wheelchart_legend_rows_int
      { \fp_eval:n { ceil ( \WCtotalcount / \l__wheelchart_legend_columns_int ) } }
     \int_step_inline:nn { \l__wheelchart_legend_rows_int - 1 }
      {
          \int_step_inline:nn { \l__wheelchart_legend_columns_int - 2 }
            { \__wheelchart_legend_append:nn { ##1 + \l__wheelchart_legend_rows_int * ####1 } { & } }
          \int_compare:nNnF { ##1 + ( \l__wheelchart_legend_columns_int - 1 ) * \l__wheelchart_legend_rows_int } > { \WCtotalcount }
            {
              \__wheelchart_legend_append:nn
                { ##1 + ( \l__wheelchart_legend_columns_int - 1 ) * \l__wheelchart_legend_rows_int }
                { & }
            }
          \__wheelchart_legend_append:nn { ##1 + 1 } { \\ }
        }
      \int_step_inline:nn { \l__wheelchart_legend_columns_int - 2 }
        { \__wheelchart_legend_append:nn { \l__wheelchart_legend_rows_int * ( ##1 + 1 ) } { & } }
      \int_compare:nNnF { \l__wheelchart_legend_columns_int * \l__wheelchart_legend_rows_int } > { \WCtotalcount }
        { \__wheelchart_legend_append:nn { \l__wheelchart_legend_columns_int * \l__wheelchart_legend_rows_int } { & } }
    }
  \__wheelchart_legend_append:nn { 1 } { \\ }%at the moment it is unnecessary to set g__wheelchart_WCcount_counter to 1
  %but this is done to be future-proof if the contents of \WClegend would be parsed in a way that prohibits the value for
  %g__wheelchart_WCcount_counter to be larger than \WCtotalcount
\tl_build_end:N \WClegend
\cs_set:Npn \WCcount { \theg__wheelchart_WCcount_counter }
\cs_set:Npn \WCpercentage { \cs:w l__wheelchart_WCpercentage_\theg__wheelchart_WCcount_counter \cs_end: }
\cs_set:Npn \WCpercentagerounded { \cs:w l__wheelchart_WCpercentagerounded_\theg__wheelchart_WCcount_counter \cs_end: }
\str_if_eq:eeTF { \l__wheelchart_type_tl } { etoc }
  {
    \cs_set:Npn \WCetocthelinkedname
      { \cs:w g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_linked_name_\theg__wheelchart_WCcount_counter \cs_end: }
    \cs_set:Npn \WCetocthelinkednumber
      { \cs:w g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_linked_number_\theg__wheelchart_WCcount_counter \cs_end: }
    \cs_set:Npn \WCetocthelinkedpage
      { \cs:w g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_linked_page_\theg__wheelchart_WCcount_counter \cs_end: }
    \cs_set:Npn \WCetocthename
      { \cs:w g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_name_\theg__wheelchart_WCcount_counter \cs_end: }
    \cs_set:Npn \WCetocthenumber
      { \cs:w g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_number_\theg__wheelchart_WCcount_counter \cs_end: }
    \cs_set:Npn \WCetocthepage
      { \cs:w g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_page_\theg__wheelchart_WCcount_counter \cs_end: }
    \cs_set:Npn \WCetocthenumberofpages
      { \cs:w g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_number_of_pages_\theg__wheelchart_WCcount_counter \cs_end: }
  }
```

```
      {
        \clist_if_empty:NTF \l__wheelchart_header_clist
          {
            \int_step_inline:nn { \l__wheelchart_max_list_items_int }
              {
                \cs_set:cpn { WCvar\int_to_Alph:n {##1} }
                  { \cs:w l__wheelchart_item_WCvar\int_to_Alph:n {##1}_\theg__wheelchart_WCcount_counter \cs_end: }
              }
          }
          {
            \int_step_inline:nn { \l__wheelchart_max_list_items_int }
              {
                \cs_set:cpn { \pgfkeysvalueof { / wheelchart / header~prefix } \clist_item:Nn \l__wheelchart_header_clist {##1} }
                  {
                    \cs:w
                      l__wheelchart_item_\pgfkeysvalueof { / wheelchart / header~prefix }
                      \clist_item:Nn \l__wheelchart_header_clist {##1}_\theg__wheelchart_WCcount_counter
                    \cs_end:
                  }
              }
          }
      }
    \setcounter { g__wheelchart_WCcount_counter } { 1 }
  }

\cs_new:Npn \__wheelchart_diff_atan:nnnn #1#2#3#4
  {
    Mod
      (
        \fp_eval:n
          {
            \g__wheelchart_slices_orientation_fp *
              (
                atand
                  (
                    \cs:w g__wheelchart_#3_y_fp \cs_end: - \cs:w g__wheelchart_#4_y_fp \cs_end: ,
                    \cs:w g__wheelchart_#3_x_fp \cs_end: - \cs:w g__wheelchart_#4_x_fp \cs_end:
                  )
                - atand
```

```
         (
          \cs:w g__wheelchart_#1_y_fp \cs_end: - \cs:w g__wheelchart_#2_y_fp \cs_end: ,
          \cs:w g__wheelchart_#1_x_fp \cs_end: - \cs:w g__wheelchart_#2_x_fp \cs_end:
         )
       )
     }
    ,
    360
   )%note the Mod 360 because for example cos(90/2)\neq cos(-270/2)
 }

\cs_new_protected:Npn \__wheelchart_discrete_algorithm:
 {
  \__wheelchart_def_fp:nn { discrete_factor } { discrete~factor }
  \__wheelchart_def_fp:nn { gap_radius } { gap~radius }
  \__wheelchart_def_outer_radius:
  \__wheelchart_def_inner_radius:
  \int_compare:nNnT { \l__wheelchart_discrete_space_at_borders_int } = { -1 }
   {
    \pgfkeys
     {
      / errors / boolean~expected /. expanded =
       { discrete~space~at~borders }
       { \pgfkeysvalueof { / wheelchart / discrete~space~at~borders } }
     }
   }
  \seq_clear:N \l__wheelchart_discrete_points_seq
  \bool_if:NTF \l__wheelchart_plot_bool
   {
    \__wheelchart_def_fp:nn { samples } { samples }
    \fp_zero:N \l__wheelchart_discrete_outer_length_fp
    \__wheelchart_discrete_def_coord:nn { outer } { 0 }
    \int_step_inline:nn { \fp_use:N \l__wheelchart_samples_fp - 1 }
     {
      \__wheelchart_discrete_def_coord:nn { outer } {##1}
      \fp_add:Nn \l__wheelchart_discrete_outer_length_fp
       {
        sqrt
         (
       ( \g__wheelchart_coord_x_fp - \g__wheelchart_previous_coord_x_fp ) ^ 2
       + ( \g__wheelchart_coord_y_fp - \g__wheelchart_previous_coord_y_fp ) ^ 2
      )
    }
  }
 \__wheelchart_discrete_def_coord:nn { inner } { \l__wheelchart_samples_fp - 1 }
 \fp_set:Nn \l__wheelchart_discrete_end_length_fp
  {
   sqrt
    (
     ( \g__wheelchart_coord_x_fp - \g__wheelchart_previous_coord_x_fp ) ^ 2
     + ( \g__wheelchart_coord_y_fp - \g__wheelchart_previous_coord_y_fp ) ^ 2
    )
  }
 \fp_zero:N \l__wheelchart_discrete_inner_length_fp
 \int_step_inline:nnnn { \fp_use:N \l__wheelchart_samples_fp - 2 } { -1 } { 0 }
  {
   \__wheelchart_discrete_def_coord:nn { inner } {##1}
   \fp_add:Nn \l__wheelchart_discrete_inner_length_fp
    {
     sqrt
      (
       ( \g__wheelchart_coord_x_fp - \g__wheelchart_previous_coord_x_fp ) ^ 2
       + ( \g__wheelchart_coord_y_fp - \g__wheelchart_previous_coord_y_fp ) ^ 2
      )
    }
  }
 \__wheelchart_discrete_def_coord:nn { outer } { 0 }
 \fp_set:Nn \l__wheelchart_discrete_start_length_fp
  {
   sqrt
    (
     ( \g__wheelchart_coord_x_fp - \g__wheelchart_previous_coord_x_fp ) ^ 2
     + ( \g__wheelchart_coord_y_fp - \g__wheelchart_previous_coord_y_fp ) ^ 2
    )
  }
}
{
 \fp_set:Nn \l__wheelchart_discrete_outer_length_fp
```

```
   { abs ( \l__wheelchart_total_angle_fp * deg * \l__wheelchart_outer_radius_fp ) }
  \fp_set:Nn \l__wheelchart_discrete_end_length_fp { abs ( \l__wheelchart_outer_radius_fp - \l__wheelchart_inner_radius_fp ) }
  \fp_set:Nn \l__wheelchart_discrete_inner_length_fp
   { abs ( \l__wheelchart_total_angle_fp * deg * \l__wheelchart_inner_radius_fp ) }
  %note the abs ( ... ) because \l__wheelchart_total_angle_fp can be negative
  %and \l__wheelchart_outer_radius_fp can be smaller than \l__wheelchart_inner_radius_fp
  \fp_set_eq:NN \l__wheelchart_discrete_start_length_fp \l__wheelchart_discrete_end_length_fp
 }
\str_case:enF { \pgfkeysvalueof { / wheelchart / discrete~partitioning } }
 {
  { radius }
   {
    \int_set:Nn \l__wheelchart_discrete_partitioning_first_index_int { 1 }
    \int_set:Nn \l__wheelchart_discrete_partitioning_second_index_int { 2 }
    \fp_set_eq:NN \l__wheelchart_discrete_level_start_length_fp \l__wheelchart_discrete_inner_length_fp
    \fp_set_eq:NN \l__wheelchart_discrete_level_end_length_fp \l__wheelchart_discrete_outer_length_fp
    \fp_set_eq:NN \l__wheelchart_discrete_sublevel_start_length_fp \l__wheelchart_discrete_start_length_fp
    \fp_set_eq:NN \l__wheelchart_discrete_sublevel_end_length_fp \l__wheelchart_discrete_end_length_fp
   }
  { angle }
   {
    \int_set:Nn \l__wheelchart_discrete_partitioning_first_index_int { 2 }
    \int_set:Nn \l__wheelchart_discrete_partitioning_second_index_int { 1 }
    \fp_set_eq:NN \l__wheelchart_discrete_level_start_length_fp \l__wheelchart_discrete_start_length_fp
    \fp_set_eq:NN \l__wheelchart_discrete_level_end_length_fp \l__wheelchart_discrete_end_length_fp
    \fp_set_eq:NN \l__wheelchart_discrete_sublevel_start_length_fp \l__wheelchart_discrete_inner_length_fp
    \fp_set_eq:NN \l__wheelchart_discrete_sublevel_end_length_fp \l__wheelchart_discrete_outer_length_fp
    \int_set:Nn \l__wheelchart_discrete_sort_int { 3 - \l__wheelchart_discrete_sort_int }
   }
 }
 {
  \pgfkeys
   {
    / errors / unknown~choice~value /. expanded =
     { discrete~partitioning }
     { \pgfkeysvalueof { / wheelchart / discrete~partitioning } }
   }
 }
\int_set:Nn \l__wheelchart_discrete_levels_int
```

```
  {
    \fp_eval:n
      {
        max
          (
            round
              (
                sqrt
                  (
                    (
                      ( \l__wheelchart_discrete_sublevel_start_length_fp + \l__wheelchart_discrete_sublevel_end_length_fp )
                      * \l__wheelchart_discrete_factor_fp
                      / ( \l__wheelchart_discrete_level_start_length_fp + \l__wheelchart_discrete_level_end_length_fp )
                    )
                    * \WCtotalnum
                  )
              )
            ,
            1
          )
      }
  }
\int_gzero:N \g__wheelchart_discrete_count_int
\fp_zero:N \l__wheelchart_discrete_levels_sum_fp
\int_step_inline:nn { \l__wheelchart_discrete_levels_int - 1 }
  {
    \fp_set:Nn \l__wheelchart_discrete_level_fp
      {
        ( ##1 - 0.5 * ( 1 + \l__wheelchart_discrete_space_at_borders_int ) )
        / ( \l__wheelchart_discrete_levels_int - \l__wheelchart_discrete_space_at_borders_int )
      }
    \fp_add:Nn \l__wheelchart_discrete_levels_sum_fp { \l__wheelchart_discrete_level_fp }
    \int_set:Nn \l__wheelchart_discrete_sublevels_int
      {
        \fp_eval:n
          {
            round
              (
                (
       (
        ( ##1 ) * \l__wheelchart_discrete_level_start_length_fp
        + \l__wheelchart_discrete_levels_sum_fp
        * ( \l__wheelchart_discrete_level_end_length_fp - \l__wheelchart_discrete_level_start_length_fp )
       )
       /
       (
        \l__wheelchart_discrete_levels_int * 0.5
        * ( \l__wheelchart_discrete_level_start_length_fp + \l__wheelchart_discrete_level_end_length_fp )
       )
      )
      * \WCtotalnum - \g__wheelchart_discrete_count_int
     )
   }
 }
\int_gadd:Nn \g__wheelchart_discrete_count_int { \l__wheelchart_discrete_sublevels_int }
\int_compare:nNnTF { \l__wheelchart_discrete_sublevels_int } = { 1 }
 {
  \seq_put_right:Ne \l__wheelchart_discrete_points_seq
   {
    0.5
    /
    \fp_use:N \l__wheelchart_discrete_level_fp
   }
 }
 {
  \int_step_inline:nn { \l__wheelchart_discrete_sublevels_int }
   {
    \seq_put_right:Ne \l__wheelchart_discrete_points_seq
     {
      \fp_eval:n
       {
        ( ####1 - 0.5 * ( 1 + \l__wheelchart_discrete_space_at_borders_int ) )
        / ( \l__wheelchart_discrete_sublevels_int - \l__wheelchart_discrete_space_at_borders_int )
       }
      %the denominator is 0 if \l__wheelchart_discrete_sublevels_int = \l__wheelchart_discrete_space_at_borders_int = 1
      %thus the case when \l__wheelchart_discrete_sublevels_int = 1 is treated separately above
      /
      \fp_use:N \l__wheelchart_discrete_level_fp
```

```
              }
          }
      }
  }
\int_compare:nNnTF { \l__wheelchart_discrete_levels_int } = { 1 }
  { \fp_set:Nn \l__wheelchart_discrete_level_fp { 0.5 } }
  {
    \fp_set:Nn \l__wheelchart_discrete_level_fp
      {
        1 - 0.5 * ( 1 - \l__wheelchart_discrete_space_at_borders_int )
        / ( \l__wheelchart_discrete_levels_int - \l__wheelchart_discrete_space_at_borders_int )
      }
  }
\int_set:Nn \l__wheelchart_discrete_sublevels_int { \fp_eval:n { round ( \WCtotalnum - \g__wheelchart_discrete_count_int ) } }
\int_compare:nNnTF { \l__wheelchart_discrete_sublevels_int } = { 1 }
  {
    \seq_put_right:Ne \l__wheelchart_discrete_points_seq
      {
        0.5
        /
        \fp_use:N \l__wheelchart_discrete_level_fp
      }
  }
  {
    \int_step_inline:nn { \l__wheelchart_discrete_sublevels_int }
      {
        \seq_put_right:Ne \l__wheelchart_discrete_points_seq
          {
            \fp_eval:n
              {
                ( ##1 - 0.5 * ( 1 + \l__wheelchart_discrete_space_at_borders_int ) )
                / ( \l__wheelchart_discrete_sublevels_int - \l__wheelchart_discrete_space_at_borders_int )
              }
            /
            \fp_use:N \l__wheelchart_discrete_level_fp
          }
      }
  }
\seq_sort:Nn \l__wheelchart_discrete_points_seq
```

```
  {
    \seq_set_split:Nnn \l__wheelchart_discrete_coefficients_first_seq { / } {##1}
    \seq_set_split:Nnn \l__wheelchart_discrete_coefficients_second_seq { / } {##2}
    \fp_compare:nNnTF
      { \seq_item:Nn \l__wheelchart_discrete_coefficients_first_seq { \l__wheelchart_discrete_sort_int } }
      >
      { \seq_item:Nn \l__wheelchart_discrete_coefficients_second_seq { \l__wheelchart_discrete_sort_int } }
      { \sort_return_swapped: }
      { \sort_return_same: }
  }
\int_gzero:N \g__wheelchart_discrete_count_int
\__wheelchart_for_loop:n
  {
    \pgfkeysvalueof { / wheelchart / before~slices }
    \int_step_inline:nn { \fp_eval:n { round ( \cs:w g__wheelchart_value_\WCcount _fp \cs_end: ) } }
    %note that \fp_eval:n { round ( ... ) } is necessary even when the value is an integer because pgfmath
    %could have added .0 and then \int_step_inline:nn { \cs:w g__wheelchart_value_\WCcount _fp \cs_end: } would give the messages
    %Missing character: There is no . in font nullfont! Missing character: There is no 0 in font nullfont!
      {
        \int_gincr:N \g__wheelchart_discrete_count_int
        \cs_set:Npe \WCcountdiscrete { \int_use:N \g__wheelchart_discrete_count_int }
        \seq_set_split:Nne \l__wheelchart_discrete_coefficients_first_seq { / }
          { \seq_item:Nn \l__wheelchart_discrete_points_seq { \g__wheelchart_discrete_count_int } }
        %Naturally, an error occurs if the sum of the rounded values of the key value is
        %greater than the rounded value of \WCtotalnum.
        %For example if there are 2 values 1.6 and 1.7 then these numbers are 4 and 3 and then there is no 4-th item in the list.
        %However only positive integer values make practical sense for this diagram.
        \bool_if:NTF \l__wheelchart_plot_bool
          {
            \coordinate ( g__wheelchart_slice_##1_####1_coordinate ) at
              (
                $
                  (
                    \__wheelchart_inner_plot:nn
                      {
                        \fp_eval:n
                          {
                            \l__wheelchart_start_angle_fp + \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_total_angle_fp *
                              (
           \seq_item:Nn \l__wheelchart_discrete_coefficients_first_seq
            { \l__wheelchart_discrete_partitioning_first_index_int }
          )
        }
      }
      { \fp_use:N \l__wheelchart_inner_radius_fp }
    )
    !
     {
      \seq_item:Nn \l__wheelchart_discrete_coefficients_first_seq
       { \l__wheelchart_discrete_partitioning_second_index_int }
     }
    !
    (
     \__wheelchart_outer_plot:nn
      {
       \fp_eval:n
        {
         \l__wheelchart_start_angle_fp + \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_total_angle_fp *
          (
           \seq_item:Nn \l__wheelchart_discrete_coefficients_first_seq
            { \l__wheelchart_discrete_partitioning_first_index_int }
          )
        }
      }
      { \fp_use:N \l__wheelchart_outer_radius_fp }
    )
   $
  ) ;
}
{
 \coordinate ( g__wheelchart_slice_##1_####1_coordinate ) at
  (
   \fp_eval:n
    {
     \l__wheelchart_start_angle_fp + \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_total_angle_fp *
      (
       \seq_item:Nn \l__wheelchart_discrete_coefficients_first_seq
        { \l__wheelchart_discrete_partitioning_first_index_int }
```

```
                          )
                      }
                    \c_colon_str
                    \fp_eval:n
                      {
                        \l__wheelchart_inner_radius_fp
                        + \seq_item:Nn \l__wheelchart_discrete_coefficients_first_seq
                          { \l__wheelchart_discrete_partitioning_second_index_int }
                          * ( \l__wheelchart_outer_radius_fp - \l__wheelchart_inner_radius_fp )
                      }
                  ) ;
              }
            \pic [ / wheelchart / slices_style ] at ( g__wheelchart_slice_##1_####1_coordinate )
              { code = { \pgfkeysvalueof { / wheelchart / discrete~pic } } } ;
          }
        \pgfkeysvalueof { / wheelchart / after~slices }
      }
  }

\cs_new_protected:Npn \__wheelchart_discrete_def_coord:nn #1#2
  {
    \fp_gset_eq:NN \g__wheelchart_previous_coord_x_fp \g__wheelchart_coord_x_fp
    \fp_gset_eq:NN \g__wheelchart_previous_coord_y_fp \g__wheelchart_coord_y_fp
    \__wheelchart_def_coord:nnnn
      { coord }
      {#1}
      {}
      {
        \fp_eval:n
          {
            \l__wheelchart_start_angle_fp + ( ( #2 ) / ( \l__wheelchart_samples_fp - 1 ) )
            * \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_total_angle_fp
          }
      }
  }

\cs_new_protected:Npn \__wheelchart_for_loop:n #1
  {
    \__wheelchart_for_loop_initial:n
      {
        \__wheelchart_def_slice_keys:n
          {
            \cs_set_eq:Nc \WCpercentage { l__wheelchart_WCpercentage_\WCcount }
            \cs_set_eq:Nc \WCpercentagerounded { l__wheelchart_WCpercentagerounded_\WCcount }
            \cs_set:Npe \WCdataangle
              { \fp_use:c { g__wheelchart_WCdataangle_\WCcount _fp } }
            \cs_set:Npe \WCmidangle
              { \fp_use:c { g__wheelchart_WCmidangle_\WCcount _fp } }
            \pgfkeysvalueof { / wheelchart / for~loop~start }%this must be placed after the definition of macros such as \WCpercentage
            %so that these macros can be used in the key for loop start
            \begin { scope }
              [
                shift = { ( \WCmidangle \c_colon_str \fp_use:c { g__wheelchart_explode_\WCcount _fp } ) } ,
                / wheelchart / slices_scope
              ]
              #1
            \end { scope }
            \pgfkeysvalueof { / wheelchart / for~loop~end }
          }
      }
  }

\cs_new_protected:Npn \__wheelchart_for_loop_initial:n #1
  {
    \str_case:en { \l__wheelchart_type_tl }
      {
        { default }
          {
            \clist_if_empty:NTF \l__wheelchart_header_clist
              {
                \int_step_inline:nn { \WCtotalcount }
                  {
                    \cs_set:Npe \WCcount {##1}
                    \int_step_inline:nn { \l__wheelchart_max_list_items_int }
                      { \cs_set_eq:cc { WCvar\int_to_Alph:n {####1} } { l__wheelchart_item_WCvar\int_to_Alph:n {####1}_##1 } }
                    #1
                  }
```

```
   }
   {
    \int_step_inline:nn { \WCtotalcount }
     {
      \cs_set:Npe \WCcount {##1}
      \int_step_inline:nn { \l__wheelchart_max_list_items_int }
       {
        \cs_set_eq:cc
         { \pgfkeysvalueof { / wheelchart / header~prefix } \clist_item:Nn \l__wheelchart_header_clist {####1} }
         {
          l__wheelchart_item_\pgfkeysvalueof { / wheelchart / header~prefix }
          \clist_item:Nn \l__wheelchart_header_clist {####1}_##1
         }
       }
      #1
     }
   }
 }
{ totalcount }
 {
  \int_step_inline:nn { \WCtotalcount }
   {
    \cs_set:Npe \WCcount {##1}
    #1
   }
 }
{ etoc }
 {
  \int_step_inline:nn { \WCtotalcount }
   {
    \cs_set:Npe \WCcount {##1}
    \cs_set_eq:Nc \WCetocthelinkedname { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_linked_name_##1 }
    \cs_set_eq:Nc \WCetocthelinkednumber { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_linked_number_##1 }
    \cs_set_eq:Nc \WCetocthelinkedpage { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_linked_page_##1 }
    \cs_set_eq:Nc \WCetocthename { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_name_##1 }
    \cs_set_eq:Nc \WCetocthenumber { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_number_##1 }
    \cs_set_eq:Nc \WCetocthepage { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_page_##1 }
    \cs_set_eq:Nc \WCetocthenumberofpages { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _the_number_of_pages_##1 }
    #1
```

```
       }
     }
   }
 }

\cs_new_protected:Npn \__wheelchart_gdef_count_fp:nn #1#2
 {
  \fp_gzero_new:c { g__wheelchart_#1_\WCcount _fp }
  \pgfmathparse { \pgfkeysvalueof { / wheelchart / #2 } }
  \fp_gset:cn { g__wheelchart_#1_\WCcount _fp } { \pgfmathresult }
 }

\cs_new_protected:Npn \__wheelchart_if_text:nnn #1#2#3
 {
 %https://tex.stackexchange.com/questions/42280/expand-away-empty-macros-within-ifthenelse
 %https://tex.stackexchange.com/questions/44919/proper-way-to-detect-empty-blank-text
  \begin { pgfinterruptpicture }
   \DeclareDocumentCommand \\ {#2} {}
   %for arc data, \\ is used with seq_set_split so has no optional argument thus then #2 is empty
   %otherwise, #2 is o
   %no s because an optional star does not apply in a node
   %no ! because a space between \\ and its optional argument is allowed in a node
   %https://tex.stackexchange.com/questions/459853/savebox-within-tikzpicture-results-in-an-empty-savebox
   \hbox_gset:Nn \g__wheelchart_if_text_box { \pgfkeysvalueof { / wheelchart / #1 } }
  \end { pgfinterruptpicture }
  \dim_compare:nNnT { \box_wd:N \g__wheelchart_if_text_box } > { 0 pt }
   { {#3} }
 }

\cs_new_protected:Npn \__wheelchart_initial:n #1
 {
  \str_case:en { \l__wheelchart_type_tl }
   {
    { default }
     {
      \tl_if_empty:nTF {#1}
       { \cs_set:Npn \WCtotalcount { 0 } }
       {
```

```
    \cs:w seq_set_split:Ne\l__wheelchart_expand_list_tl \cs_end:
     \l__wheelchart_list_seq
     { \pgfkeysvalueof { / wheelchart / separator~rows } }
     {#1}
    \cs_set:Npe \WCtotalcount { \seq_count:N \l__wheelchart_list_seq }
    \int_zero:N \l__wheelchart_max_list_items_int
    \seq_map_indexed_inline:Nn \l__wheelchart_list_seq
     {
      \cs:w seq_set_split:Ne\l__wheelchart_expand_list_items_tl \cs_end:
       \l__wheelchart_list_items_seq
       { \pgfkeysvalueof { / wheelchart / separator~columns } }
       {##2}
      \int_compare:nNnT { \seq_count:N \l__wheelchart_list_items_seq } > { \l__wheelchart_max_list_items_int }
       { \int_set:Nn \l__wheelchart_max_list_items_int { \seq_count:N \l__wheelchart_list_items_seq } }
      %make sure that the namespace l__wheelchart_item_ below is unique
      \clist_if_empty:NTF \l__wheelchart_header_clist
       {
        \seq_map_indexed_inline:Nn \l__wheelchart_list_items_seq
         { \cs_set:cpn { l__wheelchart_item_WCvar\int_to_Alph:n {####1}_##1 } {####2} }
       }
       {
        \seq_map_indexed_inline:Nn \l__wheelchart_list_items_seq
         {
          \cs_set:cpn
           {
            l__wheelchart_item_\pgfkeysvalueof { / wheelchart / header~prefix }
            \clist_item:Nn \l__wheelchart_header_clist {####1}_##1
           }
           {####2}
         }
       }
     }
   }
 }
{ totalcount }
 {
  \cs_set:Npe \WCtotalcount { \fp_use:N \l__wheelchart_total_count_fp }
  \cs_set:Npn \WCvarA { 1 }
  \cs_set:Npn \WCvarB {}
```

```
    \cs_set:Npn \WCvarC {}
  }
{ etoc }
  {
    \bool_if:NTF \l__wheelchart_etoc_use_name_bool
      {
        \int_if_exist:cTF { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
          { \cs_set:Npe \WCtotalcount { \int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int } } }
          { \cs_set:Npn \WCtotalcount { 0 } }
      }
      {
        \IfPackageLoadedTF { etoc } {}
          { \PackageError { wheelchart } { The~package~etoc~must~be~loaded~to~use~the~key~etoc~level } {} }
        \etocsetlevel { part } { 0 }
        \etocsetlevel { chapter } { 0 }
        \etocsetlevel { section } { 0 }
        \etocsetlevel { subsection } { 0 }
        \etocsetlevel { subsubsection } { 0 }
        \etocsetlevel { paragraph } { 0 }
        \etocsetlevel { subparagraph } { 0 }
        \etocsetlevel { \l__wheelchart_etoc_level_tl } { -1 }%these level changes are local to the current group
        \etocsetnexttocdepth { -1 }%only for the next toc
        \etocsetstyle { \l__wheelchart_etoc_level_tl } {} {}
          {
            \int_compare:nNnT { \cs:w g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int \cs_end: } > { 0 }
              {
                \cs_gset:cpe
                  {
                    g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl
                    _the_number_of_pages_\int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
                  }
                  {
                    \int_eval:n
                      {
                        \etocthepage -
                          \cs:w
                            g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl
                            _the_page_\int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
                          \cs_end:
          }
      }
  }
\int_gincr:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
\cs_gset_eq:cN
  {
    g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl
    _the_linked_name_\int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
  }
  \etocthelinkedname
\cs_gset_eq:cN
  {
    g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl
    _the_linked_number_\int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
  }
  \etocthelinkednumber
\cs_gset_eq:cN
  {
    g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl
    _the_linked_page_\int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
  }
  \etocthelinkedpage
\cs_gset_eq:cN
  {
    g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl
    _the_name_\int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
  }
  \etocthename
\cs_gset_eq:cN
  {
    g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl
    _the_number_\int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
  }
  \etocthenumber
\cs_gset_eq:cN
  {
    g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl
    _the_page_\int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
  }
```

```
         \etocthepage
       }
       {}
      \int_gzero_new:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
      \pgfkeysvalueof { / wheelchart / etoc~code }
      \int_compare:nNnT { \cs:w g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int \cs_end: } > { 0 }
       {
        \cs_gset:cpe
         {
          g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl
          _the_number_of_pages_\int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
         }
         {
          \int_eval:n
           {
            \l__wheelchart_etoc_count_total_pages_int + 1 -
            \cs:w
             g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl
             _the_page_\int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int }
            \cs_end:
           }
         }
       }
      \cs_set:Npe \WCtotalcount { \int_use:c { g__wheelchart_etoc_item_\l__wheelchart_etoc_name_tl _count_int } }
     }
   }
 }
\cs_set:Npn \WCtotalnum { 0 }
\__wheelchart_for_loop_initial:n
 {
  \__wheelchart_def_slice_keys:n
   { \__wheelchart_gdef_count_fp:nn { value } { value } }
  \cs_set:Npe \WCtotalnum { \fp_eval:n { \WCtotalnum + \cs:w g__wheelchart_value_\WCcount _fp \cs_end: } }
 }
\int_step_inline:nn { \WCtotalcount }
 {
  \cs_set:cpe { l__wheelchart_WCpercentage_##1 }
   { \fp_eval:n { \cs:w g__wheelchart_value_##1_fp \cs_end: / ( \WCtotalnum / 100 ) } }
  \cs_set:cpe { l__wheelchart_WCpercentagerounded_##1 }
```

```
          { \fp_eval:n { round ( \cs:w l__wheelchart_WCpercentage_##1 \cs_end: , \pgfkeysvalueof { / wheelchart / perc~precision } ) } }
      }
  }

\cs_new_protected:Npn \__wheelchart_inner_and_wheel_data:n #1
  {
    \__wheelchart_if_text:nnn { #1~data } { o }
      {
        \__wheelchart_def_fp:nn { #1_data_angle_pos } { #1~data~angle~pos }
        \__wheelchart_def_fp:nn { #1_data_angle_shift } { #1~data~angle~shift }
        \__wheelchart_def_fp:nn { #1_data_pos } { #1~data~pos }
        \__wheelchart_def_fp:nn { #1_data_sep } { #1~data~sep }
        \node [ align = left , / wheelchart / #1_data_style ] at
          \cs:w __wheelchart_point_plot_\bool_to_str:N \l__wheelchart_plot_bool :nnnnn \cs_end:
            { \WCcount }
            { \cs:w l__wheelchart_#1_data_angle_pos_fp \cs_end: }
            { \cs:w l__wheelchart_#1_data_angle_shift_fp \cs_end: }
            { \cs:w l__wheelchart_#1_data_pos_fp \cs_end: }
            { \cs:w l__wheelchart_#1_data_sep_fp \cs_end: }
          { \pgfkeysvalueof { / wheelchart / #1~data } } ;
      }
  }

\cs_new:Npn \__wheelchart_inner_plot:nn #1#2
  { {#1} \c_colon_str {#2} }

\cs_new_protected:Npn \__wheelchart_legend_append:nn #1#2
  {
    \tl_build_put_right:NV \WClegend \l__wheelchart_legend_row_tl%note the V specifier so that \WClegend can also be combined
    %with an S column of the package siunitx and so that \WClegend can be used in a tblr environment of the package tabularray
    %and then the option expand=\WClegend needs to be given to the tblr environment
    \tl_build_put_right:Nn \WClegend { \protect \setcounter { g__wheelchart_WCcount_counter } }%note the \protect for in case that
    %\WClegend would be parsed in a particular way
    %note that the counter g__wheelchart_WCcount_counter is defined globally and at the end of the previous \l__wheelchart_legend_row_tl
    %so that this value is defined and has the right value at the start of the next \l__wheelchart_legend_row_tl
    %if \WClegend is used in a tblr environment of the package tabularray then \UseTblrLibrary{counter} is required
    \tl_build_put_right:Ne \WClegend { { \int_eval:n {#1} } }
    \tl_build_put_right:Nn \WClegend {#2}
```

```
 }

\cs_new:Npn \__wheelchart_mod:n #1
 {
  \int_eval:n
   {
    \int_mod:nn { (#1) - \int_compare:nNnTF {#1} > { 0 } { 1 } { 0 } } { \g__wheelchart_totalcount_tl }
    + \int_compare:nNnTF {#1} > { 0 } { 1 } { \g__wheelchart_totalcount_tl }
   }
 }

\cs_new:Npn \__wheelchart_outer_plot:nn #1#2
 { {#1} \c_colon_str {#2} }

\cs_new:Npn \__wheelchart_point_plot_false:nnnnn #1#2#3#4#5
 { ( \__wheelchart_def_angle_plot_false:nnnnn {#1} {#2} {#3} {#4} {#5} \c_colon_str \__wheelchart_def_radius:nnn {#1} {#4} {#5} ) }

\cs_new:Npn \__wheelchart_point_plot_true:nnnnn #1#2#3#4#5
 { \__wheelchart_convex_comb_coord_aux:o { \__wheelchart_convex_comb_coord_def:nnnn {#2} {#3} {#4} {#5} } }

\cs_new_protected:Npn \__wheelchart_slices_arc:nnnnnn #1#2#3#4#5#6
 {
  {
   [
    / utils / exec =
     {
      \pgfmathparse {#1}
      \fp_set:Nn \l__wheelchart_slices_arc_A_fp { \pgfmathresult }
      #6
      \fp_set:Nn \l__wheelchart_slices_arc_A_abs_fp { abs ( \l__wheelchart_slices_arc_A_fp ) }
      \fp_compare:nNnF { \l__wheelchart_slices_arc_A_abs_fp } < { 0.01 }
       {
        \pgfmathparse {#2}
        \fp_set:Nn \l__wheelchart_slices_arc_B_fp { \pgfmathresult }
        \fp_set:Nn \l__wheelchart_slices_arc_rotate_fp
         {
          atand
```

```
                (
                  \cs:w g__wheelchart_#3_y_fp \cs_end: - \cs:w g__wheelchart_#4_y_fp \cs_end: ,
                  \cs:w g__wheelchart_#3_x_fp \cs_end: - \cs:w g__wheelchart_#4_x_fp \cs_end:
                )
            }
          \fp_set:Nn \l__wheelchart_slices_arc_coord_fp
            {
              \l__wheelchart_slices_arc_A_fp < 0 && \l__wheelchart_slices_arc_B_fp < 0
              ?
              0
              :
              \l__wheelchart_slices_arc_B_fp
            }
          \fp_set:Nn \l__wheelchart_slices_arc_angle_fp
            {
              \l__wheelchart_slices_arc_A_fp < 0 && \l__wheelchart_slices_arc_B_fp < 0
              ?
              acosd
                (
                  2 /
                    (
                      ( min ( \l__wheelchart_slices_arc_B_fp , 0 ) - 1 )
                      * ( ( 1 / \l__wheelchart_slices_arc_A_fp ) + \l__wheelchart_slices_arc_A_fp )
                    )
                )
              :
              atand ( ( \l__wheelchart_slices_arc_A_fp - ( 1 / \l__wheelchart_slices_arc_A_fp ) ) / 2 )
            }%note the min ( \l__wheelchart_slices_arc_B_fp , 0 ) so that the acosd is always defined
        }
    }
]
\fp_compare:nNnTF { \l__wheelchart_slices_arc_A_abs_fp } < { 0.01 }
  {#5}
  {
    \fp_compare:nNnT { \l__wheelchart_slices_arc_B_fp } < { 1 }
      {
        \fp_compare:nNnF { \l__wheelchart_slices_arc_coord_fp } = { 0 }
          {
            --
```

```
   (
    $
     ( g__wheelchart_slice_\WCcount _#3_coordinate )
     ! { \fp_eval:n { \l__wheelchart_slices_arc_coord_fp / 2 } } !
     ( g__wheelchart_slice_\WCcount _#4_coordinate )
    $
   )
 }
arc
 [
  start~angle =
   {
    \fp_eval:n
     {
      \l__wheelchart_slices_arc_rotate_fp
      - \g__wheelchart_slices_orientation_fp * \l__wheelchart_slices_arc_angle_fp
     }
   } ,
  end~angle =
   {
    \fp_eval:n
     {
      \l__wheelchart_slices_arc_rotate_fp + \g__wheelchart_slices_orientation_fp *
      ( sign ( \l__wheelchart_slices_arc_A_fp ) * 180 + \l__wheelchart_slices_arc_angle_fp )
     }
   } ,
  radius =
   {
    \fp_eval:n
     {
      0.25 * ( 1 - \l__wheelchart_slices_arc_B_fp )
      * abs ( ( 1 / \l__wheelchart_slices_arc_A_fp ) + \l__wheelchart_slices_arc_A_fp )
      * sqrt
       (
        ( \cs:w g__wheelchart_#3_x_fp \cs_end: - \cs:w g__wheelchart_#4_x_fp \cs_end: ) ^ 2
        + ( \cs:w g__wheelchart_#3_y_fp \cs_end: - \cs:w g__wheelchart_#4_y_fp \cs_end: ) ^ 2
       )
     }
   }
```

```
                ]
            }
          \fp_compare:nNnF { \l__wheelchart_slices_arc_coord_fp } = { 0 }
            {#5}
        }
    }
  }

\cs_new_protected:Npn \__wheelchart_slices_arrow:nnnnn #1#2#3#4#5
  {
    {
      [
        / utils / exec =
          {
            \pgfmathparse {#1}
            \fp_set:Nn \l__wheelchart_slices_arrow_A_fp { \pgfmathresult }
            \fp_compare:nNnF { \l__wheelchart_slices_arrow_A_fp } = { 0 }
              {
                \pgfmathparse {#2}
                \fp_set:Nn \l__wheelchart_slices_arrow_B_fp { \pgfmathresult }
                \fp_set:Nn \l__wheelchart_slices_arrow_coord_fp
                  {
                    \l__wheelchart_slices_arrow_A_fp < 0 && \l__wheelchart_slices_arrow_B_fp < 0 ? 0 : \l__wheelchart_slices_arrow_B_fp
                  }
              }
          }
      ]
      \fp_compare:nNnTF { \l__wheelchart_slices_arrow_A_fp } = { 0 }
        {#5}
        {
          \fp_compare:nNnT { \l__wheelchart_slices_arrow_B_fp } < { 1 }
            {
              \fp_compare:nNnF { \l__wheelchart_slices_arrow_coord_fp } = { 0 }
                {
                  --
                    (
                      $
                        ( g__wheelchart_slice_\WCcount _#3_coordinate )
                        ! { \fp_eval:n { \l__wheelchart_slices_arrow_coord_fp / 2 } } !
          ( g__wheelchart_slice_\WCcount _#4_coordinate )
        $
      )
  }
--
  (
    \fp_eval:n
      {
        (
          \cs:w g__wheelchart_#3_x_fp \cs_end: + \cs:w g__wheelchart_#4_x_fp \cs_end: +
            (
              \g__wheelchart_slices_orientation_fp * ( \l__wheelchart_slices_arrow_coord_fp - 1 )
              * \l__wheelchart_slices_arrow_A_fp
              * ( \cs:w g__wheelchart_#3_y_fp \cs_end: - \cs:w g__wheelchart_#4_y_fp \cs_end: )
            )
        )
        / 2
      }
    ,
    \fp_eval:n
      {
        (
          \cs:w g__wheelchart_#3_y_fp \cs_end: + \cs:w g__wheelchart_#4_y_fp \cs_end: +
            (
              \g__wheelchart_slices_orientation_fp * ( \l__wheelchart_slices_arrow_coord_fp - 1 )
              * \l__wheelchart_slices_arrow_A_fp
              * ( \cs:w g__wheelchart_#4_x_fp \cs_end: - \cs:w g__wheelchart_#3_x_fp \cs_end: )
            )
        )
        / 2
      }
  )
\fp_compare:nNnF { \l__wheelchart_slices_arrow_coord_fp } = { 0 }
  {
    --
      (
        $
          ( g__wheelchart_slice_\WCcount _#4_coordinate )
          ! { \fp_eval:n { \l__wheelchart_slices_arrow_coord_fp / 2 } } !
```

```
            ( g__wheelchart_slice_\WCcount _#3_coordinate )
           $
          )
        }
      }
     #5
    }
  }
 }
```

```
\cs_new:Npn \__wheelchart_slices_to:nn #1#2
 {
  to
   [
    out = { \fp_eval:n { - \g__wheelchart_slices_orientation_fp * sign ( \l__wheelchart_coord_determinant_fp ) * (#1) } } ,
    in = { \fp_eval:n { \g__wheelchart_slices_orientation_fp * sign ( \l__wheelchart_coord_determinant_fp ) * ( (#2) - 180 ) } } ,
    relative
   ]
 }
```

```
\cs_new:Npn \__wheelchart_wheel_lines_aux:nn #1#2
 {
  \fp_eval:n
   {
    ( 1 - (#1) / max ( round ( \cs:w g__wheelchart_value_\WCcount _fp \cs_end: ) , 1 ) )
     * \cs:w g__wheelchart_slice_#2_start_angle_\WCcount _fp \cs_end:
    + (#1) * \cs:w g__wheelchart_slice_#2_end_angle_\WCcount _fp \cs_end:
     / max ( round ( \cs:w g__wheelchart_value_\WCcount _fp \cs_end: ) , 1 )
   }
 }
```

A.3 Pgfkeys

```
\pgfkeys
 {
  / wheelchart /. is~family ,
  / wheelchart ,
after~slices /. initial = {} ,
anchor~xsep /. initial = 5 ,
anchor~ysep /. initial = 5 ,
arc /. style =
  {
    bool_set_true = \l__wheelchart_arc_bool ,
    arc_style /. style = {#1}
  } ,
arc_style /. style = {} ,
arc~around~line /. initial = 1 ,
arc~around~text /. default = true ,%this key is not set up with /.is choice or \bool_set:Nn
%so that for example \WCvarA can be used as value for this key
arc~around~text /. initial = false ,
arc~data /. initial = {} ,
arc~data~align /. code = { \tl_set:Nn \l__wheelchart_arc_data_align_tl {#1} } ,%this key is not set up with /.is choice
%so that for example \WCvarA can be used as value for this key
arc~data~align = center ,
arc~data~angle~pos /. initial = 0.5 ,
arc~data~angle~shift /. initial = 0 ,
arc~data~dir /. initial = 1 ,
arc~data~expand /. initial = n ,
arc~data~line~sep~factor /. initial = 1 ,
arc~data~lines~pos /. initial = 0.5 ,
arc~data~lines~shift /. initial = 0 ,
arc~data~pos /. initial = 1 ,
arc~data~sep /. initial = 1 ex / 1 cm ,
arc~data~style /. style = { arc_data_style /. style = {#1} } ,
arc_data_style /. style = {} ,
arc~first~half /. style = { arc_first_half /. style = {#1} } ,
arc_first_half /. style = {} ,
arc~pos /. initial = 1 ,
arc~second~half /. style = { arc_second_half /. style = {#1} } ,
arc_second_half /. style = {} ,
arc~sep /. initial = 1 ex / 1 cm ,
at /. initial = { ( 0 , 0 ) } ,
before~slices /. initial = {} ,
bool_set_true /. code = \bool_set_true:N #1 ,
caption /. initial = {} ,
caption~left /. initial = {} ,
caption~left~sep /. initial = 0.5 ,
caption~left~style /. style = { caption~left_style /. style = {#1} } ,
caption~left_style /. style = {} ,
caption~sep /. initial = 0.5 ,
caption~style /. style = { caption_style /. style = {#1} } ,
caption_style /. style = {} ,
contour /. style =
  {
    bool_set_true = \l__wheelchart_contour_bool ,
    contour_style /. style = {#1}
  } ,
contour_style /. style = {} ,
counterclockwise /. is~choice ,
counterclockwise / false /. code = { \fp_set:Nn \l__wheelchart_counter_or_clockwise_fp { -1 } } ,
counterclockwise / false /. value~forbidden ,
counterclockwise / true /. code = { \fp_set:Nn \l__wheelchart_counter_or_clockwise_fp { 1 } } ,
counterclockwise / true /. value~forbidden ,
counterclockwise /. default = true ,
counterclockwise = false ,
data /. initial = { \WCvarC } ,
data~angle~pos /. initial = 0.5 ,
data~angle~shift /. initial = 0 ,
data~pos /. initial = 1 ,
data~sep /. initial = 0.2 ,
data~style /. style = { data_style /. style = {#1} } ,
data_style /. style = {} ,
discrete /. is~choice ,
discrete / false /. code = \bool_set_false:N \l__wheelchart_discrete_bool ,
discrete / false /. value~forbidden ,
discrete / true /. code = \bool_set_true:N \l__wheelchart_discrete_bool ,
discrete / true /. value~forbidden ,
discrete /. default = true ,
discrete = false ,
discrete~factor /. initial = 1 ,
discrete~partitioning /. initial = radius ,
discrete~pic /. initial = {} ,
discrete~sort /. is~choice ,
discrete~sort / angle /. code = { \int_set:Nn \l__wheelchart_discrete_sort_int { 1 } } ,
discrete~sort / angle /. value~forbidden ,
discrete~sort / radius /. code = { \int_set:Nn \l__wheelchart_discrete_sort_int { 2 } } ,
discrete~sort / radius /. value~forbidden ,
discrete~sort = angle ,
discrete~space~at~borders /. is~choice ,%this key is not set up with /.is if because an initial value is unwanted for this key
discrete~space~at~borders / false /. code = { \int_set:Nn \l__wheelchart_discrete_space_at_borders_int { 1 } } ,
discrete~space~at~borders / false /. value~forbidden ,
discrete~space~at~borders / true /. code = { \int_set:Nn \l__wheelchart_discrete_space_at_borders_int { 0 } } ,
discrete~space~at~borders / true /. value~forbidden ,
discrete~space~at~borders /. default = true ,
domain /. style~args /. expanded = { ##1 \c_colon_str ##2 }
  {
    counterclockwise ,
    start~angle = {##1} ,
    total~angle = { (##2) - (##1) }
  } ,
etoc~code /. initial = { \tableofcontents } ,
etoc~count~total~pages /. code = { \int_set:Nn \l__wheelchart_etoc_count_total_pages_int {#1} } ,
etoc~level /. code =
  {
    \tl_set:Nn \l__wheelchart_type_tl { etoc }
    \tl_set:Nn \l__wheelchart_etoc_level_tl {#1}
  } ,
etoc~name /. code = { \tl_set:Nn \l__wheelchart_etoc_name_tl {#1} } ,
etoc~name = {} ,
etoc~use~name /. code =
  {
    \tl_set:Nn \l__wheelchart_etoc_name_tl {#1}
    \tl_set:Nn \l__wheelchart_type_tl { etoc }
    \bool_set_true:N \l__wheelchart_etoc_use_name_bool
  } ,
expand~list /. is~choice ,
expand~list / false /. code = { \tl_set:Nn \l__wheelchart_expand_list_tl { n } } ,
expand~list / false /. value~forbidden ,
expand~list / once /. code = { \tl_set:Nn \l__wheelchart_expand_list_tl { o } } ,
expand~list / once /. value~forbidden ,
expand~list / true /. code = { \tl_set:Nn \l__wheelchart_expand_list_tl { e } } ,
expand~list / true /. value~forbidden ,
expand~list = once ,
expand~list~items /. is~choice ,
expand~list~items / false /. code = { \tl_set:Nn \l__wheelchart_expand_list_items_tl { n } } ,
expand~list~items / false /. value~forbidden ,
expand~list~items / once /. code = { \tl_set:Nn \l__wheelchart_expand_list_items_tl { o } } ,
expand~list~items / once /. value~forbidden ,
expand~list~items / true /. code = { \tl_set:Nn \l__wheelchart_expand_list_items_tl { e } } ,
expand~list~items / true /. value~forbidden ,
expand~list~items = false ,
explode /. initial = 0 ,
explode /. default = 0.2 ,
for~loop~end /. initial = {} ,
for~loop~start /. initial = {} ,
gap /. initial = 0 ,
gap /. default = 0.05 ,
gap~max~angle /. initial = 180 ,
gap~polar /. initial = 0 ,
gap~polar /. default = 1 ,
gap~radius /. initial = 0 ,
gap~radius /. default = 0.05 ,%the same default value as for gap
header /. code = { \clist_set:Nn \l__wheelchart_header_clist {#1} } ,
header~prefix /. initial = WC ,
inner~data /. initial = {} ,
inner~data~angle~pos /. initial = 0.5 ,
inner~data~angle~shift /. initial = 0 ,
inner~data~pos /. initial = 0 ,
inner~data~sep /. initial = 0.2 ,
inner~data~style /. style = { inner_data_style /. style = {#1} } ,
inner_data_style /. style = {} ,
inner~plot /. style =
  {
    bool_set_true = \l__wheelchart_plot_bool ,
    / utils / exec = { \cs_set:Npn \__wheelchart_inner_plot:nn ##1##2 {#1} } ,
    slices~inner =
      {
        -- plot
          [
            domain =
              \fp_use:c { g__wheelchart_slice_inner_end_angle_\WCcount _fp }
              \c_colon_str
              \fp_use:c { g__wheelchart_slice_inner_start_angle_\WCcount _fp } ,
            samples = \fp_use:c { g__wheelchart_samples_\WCcount _fp } ,
            variable = \l__wheelchart_inner_plot_variable_tl ,
            / wheelchart / inner_plot_style
          ]
          (
            \__wheelchart_inner_plot:nn
              { \l__wheelchart_inner_plot_variable_tl }
              { \fp_use:c { g__wheelchart_inner_radius_\WCcount _fp } }
          )
      }
  } ,
inner~plot~style /. style = { inner_plot_style /. style = {#1} } ,
inner_plot_style /. style = {} ,
inner~radius /. initial = 2 ,
legend /. initial = {} ,
legend~columns /. initial = 1 ,
legend~entry /. initial = {} ,
legend~only /. code = \bool_set:Nn \l__wheelchart_legend_only_bool { \cs:w c_#1_bool \cs_end: } ,
legend~only /. default = true ,
legend~only = false ,
legend~row /. code =
  {
    \bool_set_true:N \l__wheelchart_legend_row_bool
    \tl_set:Nn \l__wheelchart_legend_row_tl {#1}
  } ,
lines /. initial = 0 ,
lines /. default = 1 ,
lines~angle~pos /. initial = 0.5 ,
lines~angle~shift /. initial = 0 ,
lines~ext /. initial = 0 ,
lines~ext /. default = 0.5 ,
lines~ext~bottom~dir /. code = { \int_set_eq:Nc \l__wheelchart_lines_ext_bottom_dir_int { c__wheelchart_lines_ext_dir_#1_int } } ,
lines~ext~bottom~dir = right ,
lines~ext~dir /. code =
  {
    \bool_set_true:N \l__wheelchart_lines_ext_dir_bool
    \int_set_eq:Nc \l__wheelchart_lines_ext_dir_int { c__wheelchart_lines_ext_dir_#1_int }
  } ,
lines~ext~dirsep /. initial = 0 ,
lines~ext~fixed /. default = true ,%this key is not set up with /.is choice or \bool_set:Nn
%so that for example \WCvarA can be used as value for this key
lines~ext~fixed /. initial = false ,
lines~ext~fixed~left /. initial =
  {
    \fp_eval:n
      {
        \l__wheelchart_lines_ext_dir_int *
        (
          \cs:w g__wheelchart_outer_radius_\WCcount _fp \cs_end: + \l__wheelchart_lines_sep_fp
          + \l__wheelchart_lines_fp + \l__wheelchart_lines_ext_fp
        )
      }
  } ,
lines~ext~fixed~right /. initial =
  {
    \fp_eval:n
      {
        \l__wheelchart_lines_ext_dir_int *
        (
          \cs:w g__wheelchart_outer_radius_\WCcount _fp \cs_end: + \l__wheelchart_lines_sep_fp
          + \l__wheelchart_lines_fp + \l__wheelchart_lines_ext_fp
        )
      }
  } ,
lines~ext~left~anchor /. initial = mid~east ,
lines~ext~right~anchor /. initial = mid~west ,
lines~ext~top~dir /. code = { \int_set_eq:Nc \l__wheelchart_lines_ext_top_dir_int { c__wheelchart_lines_ext_dir_#1_int } } ,
lines~ext~top~dir = right ,
lines~pos /. initial = 1 ,
lines~sep /. initial = 0.2 ,
lines~style /. style = { lines_style /. style = {#1} } ,
lines_style /. style = {} ,
middle /. initial = {} ,
middle~fill /. style =
  {
    bool_set_true = \l__wheelchart_middle_fill_bool ,
    middle_fill /. style = {#1}
  } ,
middle_fill /. style = {} ,
middle~style /. style = { middle_style /. style = {#1} } ,
middle_style /. style = {} ,
name /. code = { \tl_gset:Ne \g__wheelchart_name_tl {#1} } ,
name = wheelchart@name ,
outer~plot /. style =
  {
    bool_set_true = \l__wheelchart_plot_bool ,
    / utils / exec = { \cs_set:Npn \__wheelchart_outer_plot:nn ##1##2 {#1} } ,
    slices~outer =
      {
        -- plot
        [
          domain =
            \fp_use:c { g__wheelchart_slice_outer_start_angle_\WCcount _fp }
            \c_colon_str
            \fp_use:c { g__wheelchart_slice_outer_end_angle_\WCcount _fp } ,
          samples = \fp_use:c { g__wheelchart_samples_\WCcount _fp } ,
          variable = \l__wheelchart_outer_plot_variable_tl ,
          / wheelchart / outer_plot_style
        ]
        (
          \__wheelchart_outer_plot:nn
            { \l__wheelchart_outer_plot_variable_tl }
            { \fp_use:c { g__wheelchart_outer_radius_\WCcount _fp } }
        )
      }
  } ,
outer~plot~style /. style = { outer_plot_style /. style = {#1} } ,
outer_plot_style /. style = {} ,
outer~radius /. initial = 3 ,
perc~precision /. initial = 0 ,
pie /. code = \bool_set:Nn \l__wheelchart_pie_bool { \cs:w c_#1_bool \cs_end: } ,
pie /. default = true ,
pie = false ,
plot /. style =
  {
    inner~plot = {#1} ,
    outer~plot = {#1}
  } ,
plot~style /. style =
  {
    inner~plot~style = {#1} ,
    outer~plot~style = {#1}
  } ,
radius /. style~2~args =
  {
    inner~radius = {#1} ,
    outer~radius = {#2}
  } ,
samples /. initial = 25 ,%the same number as /tikz/samples
separator~columns /. initial = / ,
separator~rows /. initial = { , } ,
slice_final /. style = { slice_final_style /. style = {#1} } ,
slice_final_style /. style = {} ,
slices /. code =
  {
    \bool_set_true:N \l__wheelchart_slices_bool
    \tl_set:Nn \l__wheelchart_slices_tl {#1}
  } ,
slices~angle~pos /. initial = 0.5 ,
slices~angle~shift /. initial = 0 ,
slices~arc /. style~2~args =
  {
    slices~start~arc = { - (#1) } {#2} ,
    slices~end~arc = {#1} {#2}
  } ,
slices~arc~inner~end /. is~choice ,
slices~arc~inner~end / false /. style = {} ,
slices~arc~inner~end / false /. value~forbidden ,
slices~arc~inner~end / true /. style =
  { slices~arc~match = { inner } { 1 } { -1 } { 1 } { inner~end } { inner~start } { outer~end } } ,
slices~arc~inner~end / true /. value~forbidden ,
slices~arc~inner~end /. initial = false ,
slices~arc~inner~end /. default = true ,
slices~arc~inner~end~start /. is~choice ,
slices~arc~inner~end~start / false /. style = {} ,
slices~arc~inner~end~start / false /. value~forbidden ,
slices~arc~inner~end~start / true /. style =
  { slices~arc~match = { inner } { 1 } { 1 } { 1 } { inner~end } { inner~start } { outer~end } } ,
slices~arc~inner~end~start / true /. value~forbidden ,
slices~arc~inner~end~start /. initial = false ,
slices~arc~inner~end~start /. default = true ,
slices~arc~inner~start /. is~choice ,
slices~arc~inner~start / false /. style = {} ,
slices~arc~inner~start / false /. value~forbidden ,
slices~arc~inner~start / true /. style =
  { slices~arc~match = { inner } { 1 } { -1 } { -1 } { inner~start } { inner~end } { outer~start } } ,
slices~arc~inner~start / true /. value~forbidden ,
slices~arc~inner~start /. initial = false ,
slices~arc~inner~start /. default = true ,
slices~arc~inner~start~end /. is~choice ,
slices~arc~inner~start~end / false /. style = {} ,
slices~arc~inner~start~end / false /. value~forbidden ,
slices~arc~inner~start~end / true /. style =
  { slices~arc~match = { inner } { -1 } { -1 } { -1 } { inner~start } { inner~end } { outer~start } } ,
slices~arc~inner~start~end / true /. value~forbidden ,
slices~arc~inner~start~end /. initial = false ,
slices~arc~inner~start~end /. default = true ,
slices~arc~match /. style~n~args = { 7 }
  {
    slices~end~arc = { (#2) * tan ( \__wheelchart_diff_atan:nnnn {#7} {#6} {#5} {#6} / 2 ) } { 0 } ,
    slices~start~arc = { (#3) * tan ( \__wheelchart_diff_atan:nnnn {#7} {#6} {#5} {#6} / 2 ) } { 0 } ,
    slices~#1~arc = { (#4) * tan ( \__wheelchart_diff_atan:nnnn {#5} {#7} {#6} {#7} / 2 ) } { 0 }
  } ,
slices~arc~outer~end /. is~choice ,
slices~arc~outer~end / false /. style = {} ,
slices~arc~outer~end / false /. value~forbidden ,
slices~arc~outer~end / true /. style =
  { slices~arc~match = { outer } { -1 } { 1 } { -1 } { outer~end } { outer~start } { inner~end } } ,
slices~arc~outer~end / true /. value~forbidden ,
slices~arc~outer~end /. initial = false ,
slices~arc~outer~end /. default = true ,
slices~arc~outer~end~start /. is~choice ,
slices~arc~outer~end~start / false /. style = {} ,
slices~arc~outer~end~start / false /. value~forbidden ,
slices~arc~outer~end~start / true /. style =
  { slices~arc~match = { outer } { -1 } { -1 } { -1 } { outer~end } { outer~start } { inner~end } } ,
slices~arc~outer~end~start / true /. value~forbidden ,
slices~arc~outer~end~start /. initial = false ,
slices~arc~outer~end~start /. default = true ,
slices~arc~outer~start /. is~choice ,
slices~arc~outer~start / false /. style = {} ,
slices~arc~outer~start / false /. value~forbidden ,
slices~arc~outer~start / true /. style =
  { slices~arc~match = { outer } { -1 } { 1 } { 1 } { outer~start } { outer~end } { inner~start } } ,
slices~arc~outer~start / true /. value~forbidden ,
slices~arc~outer~start /. initial = false ,
slices~arc~outer~start /. default = true ,
slices~arc~outer~start~end /. is~choice ,
slices~arc~outer~start~end / false /. style = {} ,
slices~arc~outer~start~end / false /. value~forbidden ,
slices~arc~outer~start~end / true /. style =
  { slices~arc~match = { outer } { 1 } { 1 } { 1 } { outer~start } { outer~end } { inner~start } } ,
slices~arc~outer~start~end / true /. value~forbidden ,
slices~arc~outer~start~end /. initial = false ,
slices~arc~outer~start~end /. default = true ,
slices~Arrow /. style =
  {
    slices~end =
      {
        -- ( \WCpoint { 1 } {#1} { 0.5 } { 0 } )
        -- ( \WCpoint { 1 } { 0 } { 0 } { 0 } )
      } ,
    slices~start =
      {
        -- ( \WCpoint { 0 } {#1} { 0.5 } { 0 } )
        -- cycle
      }
  } ,
slices~arrow /. style~2~args =
  {
    slices~start~arrow = { - (#1) } {#2} ,
    slices~end~arrow = {#1} {#2}
  } ,
slices~end /. initial = { -- ( g__wheelchart_slice_\WCcount _inner~end_coordinate ) } ,
slices~end~arc /. style~2~args =
 {
  slices~end =
   {
    \__wheelchart_slices_arc:nnnnnn
     {#1}
     {#2}
     { outer~end }
     { inner~end }
     { -- ( g__wheelchart_slice_\WCcount _inner~end_coordinate ) }
     {}
   }
 } ,
slices~end~arrow /. style~2~args =
 {
  slices~end =
   {
    \__wheelchart_slices_arrow:nnnnn
     {#1}
     {#2}
     { outer~end }
     { inner~end }
     { -- ( g__wheelchart_slice_\WCcount _inner~end_coordinate ) }
   }
 } ,
slices~end~to /. style~2~args =
 {
  slices~end =
   {
    \__wheelchart_slices_to:nn {#2} {#1}
    ( g__wheelchart_slice_\WCcount _inner~end_coordinate )
   }
 } ,
slices~inner /. initial =
 {
  \fp_compare:nNnT { \cs:w g__wheelchart_inner_radius_\WCcount _fp \cs_end: } > { 0 }
   {
    \fp_compare:nNnT
     { \cs:w g__wheelchart_inner_gap_\WCcount _fp \cs_end: }
     <
     { \cs:w g__wheelchart_abs_half_angle_minus_new_angle_\WCcount _fp \cs_end: }
     {
      arc
       [
        start~angle = \fp_use:c { g__wheelchart_slice_inner_end_angle_\WCcount _fp } ,
        end~angle = \fp_use:c { g__wheelchart_slice_inner_start_angle_\WCcount _fp } ,
        radius = \fp_use:c { g__wheelchart_inner_radius_\WCcount _fp }
       ]
     }
   }
 } ,
slices~inner~angle~reduce /. style =
 {
  slices~inner~end~angle~shift = { - (#1) } ,
  slices~inner~start~angle~shift = {#1}
 } ,
slices~inner~angle~shift /. style =
 {
  slices~inner~end~angle~shift = {#1} ,
  slices~inner~start~angle~shift = {#1}
 } ,
slices~inner~arc /. style~2~args =
 {
  slices~inner =
   {
    \__wheelchart_slices_arc:nnnnnn
     {#1}
     {#2}
     { inner~end }
     { inner~start }
     { -- ( g__wheelchart_slice_\WCcount _inner~start_coordinate ) }
     {}
   }
 } ,
slices~inner~arc~tangent /. is~choice ,
slices~inner~arc~tangent / false /. code = {} ,
slices~inner~arc~tangent / false /. value~forbidden ,
slices~inner~arc~tangent / true /. style =
 {
  slices~inner =
   {
    \__wheelchart_slices_arc:nnnnnn
     { \__wheelchart_diff_atan:nnnn { outer~start } { inner~start } { outer~end } { inner~end } }
     { 0 }
     { inner~end }
     { inner~start }
     { -- ( g__wheelchart_slice_\WCcount _inner~start_coordinate ) }
     {
      \fp_compare:nNnTF { \l__wheelchart_slices_arc_A_fp } > { 359.99 }
       { \fp_set:Nn \l__wheelchart_slices_arc_A_fp { 1 } }
       { \fp_set:Nn \l__wheelchart_slices_arc_A_fp { tand ( 45 - \l__wheelchart_slices_arc_A_fp / 4 ) } }
     }
   }
 } ,
slices~inner~arc~tangent / true /. value~forbidden ,
slices~inner~arc~tangent /. initial = false ,
slices~inner~arc~tangent /. default = true ,
slices~inner~arrow /. style~2~args =
 {
  slices~inner =
   {
    \__wheelchart_slices_arrow:nnnnn
    {#1}
    {#2}
    { inner~end }
    { inner~start }
    { -- ( g__wheelchart_slice_\WCcount _inner~start_coordinate ) }
   }
 } ,
slices~inner~end~angle~shift /. initial = 0 ,
slices~inner~start~angle~shift /. initial = 0 ,
slices~inner~to /. style~2~args =
 {
  slices~inner =
   {
    \__wheelchart_slices_to:nn {#2} {#1}
    ( g__wheelchart_slice_\WCcount _inner~start_coordinate )
```

```
      }
  } ,
slices~outer /. initial =
  {
    arc
      [
        start~angle = \fp_use:c { g__wheelchart_slice_outer_start_angle_\WCcount _fp } ,
        end~angle = \fp_use:c { g__wheelchart_slice_outer_end_angle_\WCcount _fp } ,
        radius = \fp_use:c { g__wheelchart_outer_radius_\WCcount _fp }
      ]
  } ,
slices~outer~angle~reduce /. style =
  {
    slices~outer~end~angle~shift = { - (#1) } ,
    slices~outer~start~angle~shift = {#1}
  } ,
slices~outer~angle~shift /. style =
  {
    slices~outer~end~angle~shift = {#1} ,
    slices~outer~start~angle~shift = {#1}
  } ,
slices~outer~arc /. style~2~args =
  {
    slices~outer =
      {
        \__wheelchart_slices_arc:nnnnnn
          {#1}
          {#2}
          { outer~start }
          { outer~end }
          { -- ( g__wheelchart_slice_\WCcount _outer~end_coordinate ) }
          {}
      }
  } ,
slices~outer~arc~tangent /. is~choice ,
slices~outer~arc~tangent / false /. code = {} ,
slices~outer~arc~tangent / false /. value~forbidden ,
slices~outer~arc~tangent / true /. style =
  {
    slices~outer =
      {
        \__wheelchart_slices_arc:nnnnnn
          { \__wheelchart_diff_atan:nnnn { outer~start } { inner~start } { outer~end } { inner~end } }
          { 0 }
          { outer~start }
          { outer~end }
          { -- ( g__wheelchart_slice_\WCcount _outer~end_coordinate ) }
          {
            \fp_compare:nNnTF { \l__wheelchart_slices_arc_A_fp } > { 359.99 }
              { \fp_set:Nn \l__wheelchart_slices_arc_A_fp { 1 } }
              {
                \fp_compare:nNnTF { \l__wheelchart_slices_arc_A_fp } = { 180 }
                  { \fp_set:Nn \l__wheelchart_slices_arc_A_fp { 0 } }
                  { \fp_set:Nn \l__wheelchart_slices_arc_A_fp { cotd ( 45 - \l__wheelchart_slices_arc_A_fp / 4 ) } }
              }
          }
      }
  } ,
slices~outer~arc~tangent / true /. value~forbidden ,
slices~outer~arc~tangent /. initial = false ,
slices~outer~arc~tangent /. default = true ,
slices~outer~arrow /. style~2~args =
  {
    slices~outer =
      {
        \__wheelchart_slices_arrow:nnnnn
          {#1}
          {#2}
          { outer~start }
          { outer~end }
          { -- ( g__wheelchart_slice_\WCcount _outer~end_coordinate ) }
      }
  } ,
slices~outer~end~angle~shift /. initial = 0 ,
slices~outer~start~angle~shift /. initial = 0 ,
slices~outer~to /. style~2~args =
  {
    slices~outer =
   {
    \__wheelchart_slices_to:nn {#1} {#2}
    ( g__wheelchart_slice_\WCcount _outer~end_coordinate )
   }
  } ,
slices~pos /. initial = 0.5 ,
slices~scope /. style = { slices_scope /. style = {#1} } ,
slices_scope /. style = {} ,
slices~sep /. initial = 0 ,
slices~start /. initial = { -- cycle } ,
slices~start~arc /. style~2~args =
 {
  slices~start =
   {
    \__wheelchart_slices_arc:nnnnnn
     {#1}
     {#2}
     { inner~start }
     { outer~start }
     { -- cycle }
     {}
   }
 } ,
slices~start~arrow /. style~2~args =
 {
  slices~start =
   {
    \__wheelchart_slices_arrow:nnnnn
     {#1}
     {#2}
     { inner~start }
     { outer~start }
     { -- cycle }
   }
 } ,
slices~start~to /. style~2~args = { slices~start = { \__wheelchart_slices_to:nn {#1} {#2} cycle } } ,
slices~style /. style = { slices_style /. style = {#1} } ,
slices_style /. style = {} ,
slices~style = { \WCvarB } ,
slices~to /. style~2~args =
  {
    slices~end~to = {#1} {#2} ,
    slices~start~to = { - (#1) } { - (#2) }
  } ,
start~angle /. initial = 90 ,
start~half /. style =
  {
    start~angle =
      {
        (#1) -
        \fp_eval:n
          {
            \l__wheelchart_counter_or_clockwise_fp * \cs:w g__wheelchart_value_1_fp \cs_end: * 0.5
            * ( \l__wheelchart_total_angle_fp / \WCtotalnum )
          }
      }
  } ,
start~half /. default = 90 ,
title /. initial = {} ,
title~left /. initial = {} ,
title~left~sep /. initial = 0.5 ,
title~left~style /. style = { title~left_style /. style = {#1} } ,
title~left_style /. style = {} ,
title~sep /. initial = 0.5 ,
title~style /. style = { title_style /. style = {#1} } ,
title_style /. style = {} ,
total~angle /. initial = 360 ,
total~count /. code =
  {
    \tl_set:Nn \l__wheelchart_type_tl { totalcount }
    \pgfmathparse {#1}
    \fp_set:Nn \l__wheelchart_total_count_fp { \pgfmathresult }
  } ,
triangle~proportional~area /. style~2~args =
  {
    domain /. expanded = 0 \c_colon_str 1 ,
    plot = { { (##2) * sqrt ( 1 - (##1) ) * (#1) / 2 } , { - sqrt ( 1 - (##1) ) * (#2) } } ,
    radius = { -1 } { 1 } ,
    samples = 2 ,
    wheel~data~pos = 0.5
  } ,
triangle~proportional~height /. style~2~args =
  {
    domain /. expanded = 0 \c_colon_str 1 ,
    plot = { { (##2) * ( 1 - (##1) ) * (#1) / 2 } , { ( (##1) - 1 ) * (#2) } } ,
    radius = { -1 } { 1 } ,
    samples = 2 ,
    wheel~data~pos = 0.5
  } ,
value /. initial = { \WCvarA } ,
WC_list /. code~2~args =
  {
    \cs_set:cpn {#1}
      { \use:e { \clist_item:nn {#2} { \int_mod:nn { \WCcount - 1 } { \clist_count:n {#2} } + 1 } } }
      %note the \use:e so that \WClist<name> also works when given as an argument to pgfmath
      %if the list contains a macro, for example
      %\begin{tikzpicture}
      %\def\n{1}
      %\wheelchart[
      %  value=\WClistA,
      %  WClistA=\n
      %]{\exampleforthismanual}
      %\end{tikzpicture}
      %https://tex.stackexchange.com/questions/671298/clist-item-and-pgfmathsetmacro-causing-an-error
  } ,
wheel~data /. initial = {} ,
wheel~data~angle~pos /. initial = 0.5 ,
wheel~data~angle~shift /. initial = 0 ,
wheel~data~pos /. initial = 0.66 ,
wheel~data~sep /. initial = 0 ,
wheel~data~style /. style = { wheel_data_style /. style = {#1} } ,
wheel_data_style /. style = {} ,
wheel~lines /. style =
  {
    bool_set_true = \l__wheelchart_wheel_lines_bool ,
    wheel_lines /. style = {#1}
  } ,
    wheel_lines /. style = {} ,
    xbar /. style~2~args =
      {
        domain /. expanded = 0 \c_colon_str {#1} ,
        plot = { {##1} , {##2} } ,
        radius = { 0 } {#2} ,
        samples = 2 ,
        wheel~data~pos = 0.5
      } ,
    ybar /. style~2~args =
      {
        domain /. expanded = 0 \c_colon_str {#2} ,
        plot = { {##2} , {##1} } ,
        radius = { 0 } {#1} ,
        samples = 2 ,
        wheel~data~pos = 0.5
      } ,
  }

\pgfkeys
  {
    / wheelchart /. unknown /. code =
      {
        \tl_set:Ne \l__wheelchart_key_name_tl { \pgfkeyscurrentname }%it is necessary to define \l__wheelchart_key_name_tl
        %because \pgfkeyscurrentname will be overwritten by / errors / unknown~key /. expanded
        \regex_match:NVTF \c__wheelchart_braces_regex \l__wheelchart_key_name_tl
          {
            \tl_set:Ne \l__wheelchart_key_range_tl { \pgfkeyscurrentname }
            \regex_replace_all:NnN \c__wheelchart_braces_regex {} \l__wheelchart_key_name_tl
            \regex_replace_all:NnN \c__wheelchart_key_braces_regex { \1 } \l__wheelchart_key_range_tl
            \str_if_eq:eeTF { \l__wheelchart_key_range_tl } { list }
              {
                \pgfkeys { / wheelchart , WC_list = { l__wheelchart_list_\l__wheelchart_key_name_tl } {#1} }
                \pgfkeys
                  {
                    / wheelchart ,
                    \l__wheelchart_key_name_tl /. expand~once = { \cs:w l__wheelchart_list_\l__wheelchart_key_name_tl \cs_end: }
                  }
              }
```

```
       {
        \clist_gclear:N \g__wheelchart_slice_range_for_loop_clist
        \foreach \l__wheelchart_slice_range_index_tl [ parse = true ] in \l__wheelchart_key_range_tl
         {
          \clist_gput_right:Ne \g__wheelchart_slice_range_for_loop_clist
           { \fp_eval:n { \l__wheelchart_slice_range_index_tl } }
         }
        \clist_map_inline:Nn \g__wheelchart_slice_range_for_loop_clist
         {
          \clist_if_in:NnF \l__wheelchart_slice_range_local_clist {##1}
           { \clist_put_right:Nn \l__wheelchart_slice_range_local_clist {##1} }
          \clist_if_exist:cF { l__wheelchart_slice_##1_keys_clist }
           { \clist_new:c { l__wheelchart_slice_##1_keys_clist } }
          \str_if_eq:eeTF { \l__wheelchart_key_name_tl } { slice }
           { \clist_put_right:cn { l__wheelchart_slice_##1_keys_clist } {#1} }
           {
            \clist_put_right:ce { l__wheelchart_slice_##1_keys_clist }
             { \exp_not:V \l__wheelchart_key_name_tl \exp_not:n { = {#1} } }
           }
         }
       }
     }
     {
      \str_if_eq:eeTF { \str_range:Nnn \l__wheelchart_key_name_tl { 1 } { 6 } } { WClist }
       { \pgfkeys { / wheelchart , WC_list = { \l__wheelchart_key_name_tl } {#1} } }
       { \pgfkeys { / errors / unknown~key /. expanded = { \l__wheelchart_key_name_tl } {#1} } }
     }
   }
 }%this gives an error message if a key of the form <unknown key for wheelchart>{<range>} is given
```

A.4 Additional commands

```
\NewExpandableDocumentCommand \WCangle { O { \WCcount } m m m m }
 { \__wheelchart_def_angle_plot_false:nnnnn { \__wheelchart_mod:n {#1} } {#2} {#3} {#4} {#5} }

\NewExpandableDocumentCommand \WCcoordinate { O { \WCcount } m }
 { g__wheelchart_slice_\__wheelchart_mod:n {#1}_#2_coordinate }
```

```
\NewExpandableDocumentCommand \WCpoint { O { \WCcount } m m m m }
  { \WCangle [#1] {#2} {#3} {#4} {#5} \c_colon_str \WCradius [#1] {#4} {#5} }

\NewExpandableDocumentCommand \WCradius { O { \WCcount } m m }
  { \__wheelchart_def_radius:nnn { \__wheelchart_mod:n {#1} } {#2} {#3} }
```

A.5 The command \wheelchart

```
\NewDocumentCommand \wheelchart { O {} m }
  {
    {%note the double braces {{...}} so that the contents is in a group and so that & can be used in pgfmath in a tabular
      \pgfkeys { / wheelchart , #1 }
      \IfPackageLoadedTF { siunitx }
        { \cs_set:Npn \WCperc { \qty { \WCpercentagerounded } { \percent } } }
        { \cs_set:Npn \WCperc { \WCpercentagerounded \, \% } }%the definition of \WCperc is placed inside the command \wheelchart
        %so that \WCperc is not defined outside this command
      \bool_if:NTF \l__wheelchart_legend_only_bool
        {
          \__wheelchart_initial:n {#2}
          \bool_if:NT \l__wheelchart_legend_row_bool
            { \__wheelchart_def_WClegend: }
          \pgfkeysvalueof { / wheelchart / legend }
        }
        {
          \fp_set:Nn \l__wheelchart_coord_determinant_fp { \pgf@yy * \pgf@xx - \pgf@yx * \pgf@xy }
          \__wheelchart_def_fp:nn { total_angle } { total~angle }
          \__wheelchart_initial:n {#2}
          \tl_gset:Ne \g__wheelchart_totalcount_tl { \WCtotalcount }%\WCtotalcount is local and \g__wheelchart_totalcount_tl is global
          %because it is used in commands such as \WCangle thus must be available after the command \wheelchart
          \__wheelchart_def_fp:nn { start_angle } { start~angle }
          \fp_gset_eq:NN \g__wheelchart_angle_fp \l__wheelchart_start_angle_fp
          \__wheelchart_for_loop_initial:n
            {
              \__wheelchart_def_slice_keys:n
                {
                  \cs_set_eq:Nc \WCpercentage { l__wheelchart_WCpercentage_\WCcount }
                  \cs_set_eq:Nc \WCpercentagerounded { l__wheelchart_WCpercentagerounded_\WCcount }
                  \fp_compare:nNnTF { \WCcount } = { \WCtotalcount }
  {
    \fp_gset:Nn \g__wheelchart_new_angle_fp
      { \l__wheelchart_start_angle_fp + \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_total_angle_fp }
  }
  {
    \fp_gset:Nn \g__wheelchart_new_angle_fp
      {
        \g__wheelchart_angle_fp +
          (
            \l__wheelchart_counter_or_clockwise_fp * \cs:w g__wheelchart_value_\WCcount _fp \cs_end:
            * ( \l__wheelchart_total_angle_fp / \WCtotalnum )
          )
      }
  }
\__wheelchart_gdef_count_fp:nn { data_angle_pos } { data~angle~pos }
\__wheelchart_def_fp:nn { gap } { gap }
\__wheelchart_def_fp:nn { gap_max_angle } { gap~max~angle }
\__wheelchart_def_fp:nn { gap_polar } { gap~polar }
\__wheelchart_def_fp:nn { gap_radius } { gap~radius }
\__wheelchart_gdef_count_fp:nn { samples } { samples }
\__wheelchart_def_fp:nn { slices_inner_start_angle_shift } { slices~inner~start~angle~shift }
\__wheelchart_def_fp:nn { slices_inner_end_angle_shift } { slices~inner~end~angle~shift }
\__wheelchart_def_fp:nn { slices_outer_end_angle_shift } { slices~outer~end~angle~shift }
\__wheelchart_def_fp:nn { slices_outer_start_angle_shift } { slices~outer~start~angle~shift }
\__wheelchart_def_outer_radius:
\__wheelchart_def_inner_radius:
\fp_gzero_new:c { g__wheelchart_inner_radius_\WCcount _fp }
\fp_gset_eq:cN { g__wheelchart_inner_radius_\WCcount _fp } \l__wheelchart_inner_radius_fp
\fp_gzero_new:c { g__wheelchart_outer_radius_\WCcount _fp }
\fp_gset_eq:cN { g__wheelchart_outer_radius_\WCcount _fp } \l__wheelchart_outer_radius_fp
\fp_gzero_new:c { g__wheelchart_abs_half_angle_minus_new_angle_\WCcount _fp }
\fp_gset:cn { g__wheelchart_abs_half_angle_minus_new_angle_\WCcount _fp }
  { abs ( \g__wheelchart_angle_fp - \g__wheelchart_new_angle_fp ) / 2 }
\fp_set:Nn \l__wheelchart_abs_half_angle_minus_new_angle_minus_gap_polar_fp
  { \cs:w g__wheelchart_abs_half_angle_minus_new_angle_\WCcount _fp \cs_end: - \l__wheelchart_gap_polar_fp }
\fp_gzero_new:c { g__wheelchart_outer_gap_\WCcount _fp }
\fp_gzero_new:c { g__wheelchart_inner_gap_\WCcount _fp }
\bool_if:NTF \l__wheelchart_plot_bool
  {
```

```
  \fp_gset_eq:cN { g__wheelchart_outer_gap_\WCcount _fp } \l__wheelchart_gap_polar_fp
  \fp_gset_eq:cN { g__wheelchart_inner_gap_\WCcount _fp } \l__wheelchart_gap_polar_fp
}
{
  \fp_set:Nn \l__wheelchart_gap_max_angle_def_fp
    {
      \cs:w g__wheelchart_inner_radius_\WCcount _fp \cs_end: > 0
      ?
      90
      :
      (
        sind ( \l__wheelchart_abs_half_angle_minus_new_angle_minus_gap_polar_fp ) < 0.001
        ?
        (
          90 < \l__wheelchart_gap_max_angle_fp && \l__wheelchart_gap_max_angle_fp < 180
          ?
          \l__wheelchart_gap_max_angle_fp
          :
          90
        )
        :
        (
          \l__wheelchart_gap_max_angle_fp < 90 || \l__wheelchart_gap_max_angle_fp > 180
          ?
          180
          :
          \l__wheelchart_gap_max_angle_fp
        )
      )
    }
  \__wheelchart_def_gap:nn { outer } { \cs:w g__wheelchart_outer_radius_\WCcount _fp \cs_end: }
  \__wheelchart_def_gap:nn { inner } { \cs:w g__wheelchart_inner_radius_\WCcount _fp \cs_end: }
  \fp_compare:nNnT { \l__wheelchart_abs_half_angle_minus_new_angle_minus_gap_polar_fp } > { 0.01 }
    {
      \fp_gset:cn { g__wheelchart_inner_radius_\WCcount _fp }
        {
          max
            (
              \l__wheelchart_gap_fp
        / sind
         (
          min
           (
            \l__wheelchart_abs_half_angle_minus_new_angle_minus_gap_polar_fp ,
            \l__wheelchart_gap_max_angle_def_fp
           )
         )
        ,
        \cs:w g__wheelchart_inner_radius_\WCcount _fp \cs:end:
       )
     }
   }
 }
\__wheelchart_def_slice_angle:nnnn { inner } { end } { new_ } { -1 }
\__wheelchart_def_slice_angle:nnnn { inner } { start } {} { 1 }
\__wheelchart_def_slice_angle:nnnn { outer } { end } { new_ } { -1 }
\__wheelchart_def_slice_angle:nnnn { outer } { start } {} { 1 }
\__wheelchart_def_angle:nnnn { 0.5 } { 0 } { 0.5 } { 0 }
\cs_set:Npe \WCmidangle { \pgfmathresult }
\__wheelchart_gdef_count_fp:nn { data_angle_shift } { data~angle~shift }
\__wheelchart_gdef_count_fp:nn { data_pos } { data~pos }
\__wheelchart_gdef_count_fp:nn { data_sep } { data~sep }
\__wheelchart_def_angle:nnnn
 { \cs:w g__wheelchart_data_angle_pos_\WCcount _fp \cs_end: }
 { \cs:w g__wheelchart_data_angle_shift_\WCcount _fp \cs_end: }
 { \cs:w g__wheelchart_data_pos_\WCcount _fp \cs_end: }
 { \cs:w g__wheelchart_data_sep_\WCcount _fp \cs_end: }
\cs_set:Npe \WCdataangle { \pgfmathresult }
\__wheelchart_gdef_count_fp:nn { explode } { explode }
\pgfkeysvalueof { / wheelchart / for~loop~start }%this must be placed after the definition of macros such as
%\WCpercentage so that these macros can be used in the key for loop start
\int_compare:nNnT { \WCcount } = { 1 }
 {
  \begin { scope }
   [
    shift = { ( \WCmidangle \c_colon_str \fp_use:c { g__wheelchart_explode_\WCcount _fp } ) } ,
    / wheelchart / slices_scope
   ]
```

```
       \__wheelchart_def_orientation:
      \end { scope }
     }
    \fp_gzero_new:c { g__wheelchart_WCdataangle_\WCcount _fp }
    \fp_gset:cn { g__wheelchart_WCdataangle_\WCcount _fp } { \WCdataangle }
    \fp_gzero_new:c { g__wheelchart_WCmidangle_\WCcount _fp }
    \fp_gset:cn { g__wheelchart_WCmidangle_\WCcount _fp } { \WCmidangle }
    \fp_gset_eq:NN \g__wheelchart_angle_fp \g__wheelchart_new_angle_fp
    \pgfkeysvalueof { / wheelchart / for~loop~end }
   }
 }
\begin { scope } [ shift /. expanded = { \pgfkeysvalueof { / wheelchart / at } } ]
 \begin { scope } [ local~bounding~box /. expanded = \g__wheelchart_name_tl ]
  \bool_if:NT \l__wheelchart_middle_fill_bool
   {
    \bool_if:NF \l__wheelchart_plot_bool
     {
      \__wheelchart_def_inner_radius:
      \fill [ / wheelchart / middle_fill ]
       \fp_compare:nNnTF { \l__wheelchart_total_angle_fp } = { 360 }
        { ( 0 , 0 ) circle [ radius = \fp_use:N \l__wheelchart_inner_radius_fp ] }
        {
         ( 0 , 0 )
         -- ( \fp_use:N \l__wheelchart_start_angle_fp \c_colon_str \fp_use:N \l__wheelchart_inner_radius_fp )
         arc
          [
           start~angle = \fp_use:N \l__wheelchart_start_angle_fp ,
           end~angle =
            \fp_eval:n
             {
              \l__wheelchart_start_angle_fp
              + \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_total_angle_fp
             }
            ,
           radius = \fp_use:N \l__wheelchart_inner_radius_fp
          ]
         -- cycle
        }
       ;
   }
 }
\bool_if:NTF \l__wheelchart_discrete_bool
 { \__wheelchart_discrete_algorithm: }
 {
  \__wheelchart_for_loop:n
   {
    \pgfkeysvalueof { / wheelchart / before~slices }
    \bool_if:NTF \l__wheelchart_slices_bool
     {
      \__wheelchart_def_fp:nn { slices_angle_pos } { slices~angle~pos }
      \__wheelchart_def_fp:nn { slices_angle_shift } { slices~angle~shift }
      \__wheelchart_def_fp:nn { slices_pos } { slices~pos }
      \__wheelchart_def_fp:nn { slices_sep } { slices~sep }
      \__wheelchart_def_angle:nnnn
       { \l__wheelchart_slices_angle_pos_fp }
       { \l__wheelchart_slices_angle_shift_fp }
       { \l__wheelchart_slices_pos_fp }
       { \l__wheelchart_slices_sep_fp }
      \fp_set:Nn \l__wheelchart_slices_angle_fp { \pgfmathresult }
      \begin { scope }
       [
        shift /. expanded =
         {
          \cs:w __wheelchart_point_plot_\bool_to_str:N \l__wheelchart_plot_bool :nnnnn \cs_end:
           { \WCcount }
           { \l__wheelchart_slices_angle_pos_fp }
           { \l__wheelchart_slices_angle_shift_fp }
           { \l__wheelchart_slices_pos_fp }
           { \l__wheelchart_slices_sep_fp }
         } ,
        rotate = \fp_use:N \l__wheelchart_slices_angle_fp
       ]
       \fill [ / wheelchart / slices_style ] \l__wheelchart_slices_tl
      \end { scope }
     }
     {
      %We do not use the let operation in the path \fill[/wheelchart/slices_style] ... because then
      %\n, \p, \x and \y can not be used as macro names inside the argument of a key which is applied
```

```
            %on this path such as the key slices inner arc.
            \__wheelchart_def_coord:nnnn { inner~end } { inner } { end }
              { \fp_use:c { g__wheelchart_slice_inner_end_angle_\WCcount _fp } }
            \__wheelchart_def_coord:nnnn { inner~start } { inner } { start }
              { \fp_use:c { g__wheelchart_slice_inner_start_angle_\WCcount _fp } }
            \__wheelchart_def_coord:nnnn { outer~end } { outer } { end }
              { \fp_use:c { g__wheelchart_slice_outer_end_angle_\WCcount _fp } }
            \__wheelchart_def_coord:nnnn { outer~start } { outer } { start }
              { \fp_use:c { g__wheelchart_slice_outer_start_angle_\WCcount _fp } }
            \fill [ / wheelchart / slices_style ]
              ( g__wheelchart_slice_\WCcount _outer~start_coordinate )
              \pgfkeysvalueof { / wheelchart / slices~outer }
              \pgfkeysvalueof { / wheelchart / slices~end }
              \pgfkeysvalueof { / wheelchart / slices~inner }
              \pgfkeysvalueof { / wheelchart / slices~start }
              ;
          }
        \pgfkeysvalueof { / wheelchart / after~slices }
      }
  }
\__wheelchart_for_loop:n
  {
    \bool_if:NT \l__wheelchart_wheel_lines_bool
    %this is placed inside \__wheelchart_for_loop:n so that wheel lines can be applied for specific slices
      {
        \int_step_inline:nnn { 0 } { \fp_eval:n { round ( \cs:w g__wheelchart_value_\WCcount _fp \cs_end: ) } }
        %note the \fp_eval:n { round ( ... ) } to avoid the messages Missing character: There is no ... in font nullfont!
          {
            \bool_if:NTF \l__wheelchart_plot_bool
              {
                \draw [ / wheelchart / wheel_lines ]
                  (
                    \__wheelchart_inner_plot:nn
                      { \__wheelchart_wheel_lines_aux:nn {####1} { inner } }
                      { \fp_use:c { g__wheelchart_inner_radius_\WCcount _fp } }
                  )
                  --
                  (
                    \__wheelchart_outer_plot:nn
```

```
                      { \__wheelchart_wheel_lines_aux:nn {####1} { outer } }
                      { \fp_use:c { g__wheelchart_outer_radius_\WCcount _fp } }
                  )
                  ;
              }
              {
                \draw [ / wheelchart / wheel_lines ]
                  (
                    \__wheelchart_wheel_lines_aux:nn {####1} { inner }
                    \c_colon_str
                    \fp_use:c { g__wheelchart_inner_radius_\WCcount _fp }
                  )
                  --
                  (
                    \__wheelchart_wheel_lines_aux:nn {####1} { outer }
                    \c_colon_str
                    \fp_use:c { g__wheelchart_outer_radius_\WCcount _fp }
                  )
                  ;
              }
          }
      }
  }
\bool_if:NT \l__wheelchart_contour_bool
  {
    \bool_if:NF \l__wheelchart_plot_bool
      {
        \__wheelchart_def_outer_radius:
        \__wheelchart_def_inner_radius:
        \fp_compare:nNnTF { \l__wheelchart_total_angle_fp } = { 360 }
          {
            \draw [ / wheelchart / contour_style ]
              ( 0 , 0 ) circle [ radius = \fp_use:N \l__wheelchart_inner_radius_fp ] ;
            \draw [ / wheelchart / contour_style ]
              ( 0 , 0 ) circle [ radius = \fp_use:N \l__wheelchart_outer_radius_fp ] ;
          }
          {
            \draw [ / wheelchart / contour_style ]
              ( \fp_use:N \l__wheelchart_start_angle_fp \c_colon_str \fp_use:N \l__wheelchart_inner_radius_fp )
```

```
       arc
        [
         start~angle = \fp_use:N \l__wheelchart_start_angle_fp ,
         end~angle =
          \fp_eval:n
           {
            \l__wheelchart_start_angle_fp
            + \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_total_angle_fp
           }
          ,
         radius = \fp_use:N \l__wheelchart_inner_radius_fp
        ]
       --
        (
         \fp_eval:n
          {
           \l__wheelchart_start_angle_fp
           + \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_total_angle_fp
          }
         \c_colon_str
         \fp_use:N \l__wheelchart_outer_radius_fp
        )
       arc
        [
         start~angle =
          \fp_eval:n
           {
            \l__wheelchart_start_angle_fp
            + \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_total_angle_fp
           }
          ,
         end~angle = \fp_use:N \l__wheelchart_start_angle_fp ,
         radius = \fp_use:N \l__wheelchart_outer_radius_fp
        ]
       -- cycle ;
     }
   }
 }
\__wheelchart_for_loop:n
```

```
{
 \__wheelchart_def_fp:nn { lines } { lines }
 \__wheelchart_def_fp:nn { lines_angle_pos } { lines~angle~pos }
 \__wheelchart_def_fp:nn { lines_angle_shift } { lines~angle~shift }
 \__wheelchart_def_fp:nn { lines_ext } { lines~ext }
 \__wheelchart_def_fp:nn { lines_pos } { lines~pos }
 \__wheelchart_def_fp:nn { lines_sep } { lines~sep }
 \fp_compare:nNnF { \l__wheelchart_lines_ext_fp } = { 0 }
  {
   \bool_if:NF \l__wheelchart_lines_ext_dir_bool
    {
     \__wheelchart_def_fp:nn { lines_ext_dirsep } { lines~ext~dirsep }
     \int_set:Nn \l__wheelchart_lines_ext_dir_int
      {
       \fp_eval:n
        {
         (
          \WCdataangle < 90 - \l__wheelchart_lines_ext_dirsep_fp
          ?
          1
          :
          (
           \WCdataangle <= 90 + \l__wheelchart_lines_ext_dirsep_fp
           ?
           \l__wheelchart_lines_ext_top_dir_int
           :
           (
            \WCdataangle < 270 - \l__wheelchart_lines_ext_dirsep_fp
            ?
            -1
            :
            (
             \WCdataangle <= 270 + \l__wheelchart_lines_ext_dirsep_fp
             ?
             \l__wheelchart_lines_ext_bottom_dir_int
             :
             1
            )
           )
         )
        )
       }
     }
   }
  \__wheelchart_def_fp:nn { lines_ext_fixed_left } { lines~ext~fixed~left }
  \__wheelchart_def_fp:nn { lines_ext_fixed_right } { lines~ext~fixed~right }
 }
\fp_compare:nF { \l__wheelchart_lines_fp == 0 && \l__wheelchart_lines_ext_fp == 0 }
 {
  \draw [ / wheelchart / lines_style ] let \p { l__wheelchart_lines_1 } =
   \cs:w __wheelchart_point_plot_\bool_to_str:N \l__wheelchart_plot_bool :nnnnn \cs_end:
    { \WCcount }
    { \cs:w g__wheelchart_data_angle_pos_\WCcount _fp \cs_end: }
    { \cs:w g__wheelchart_data_angle_shift_\WCcount _fp \cs_end: }
    { \cs:w g__wheelchart_data_pos_\WCcount _fp \cs_end: }
    { \l__wheelchart_lines_sep_fp + \l__wheelchart_lines_fp }
   in
    \cs:w __wheelchart_point_plot_\bool_to_str:N \l__wheelchart_plot_bool :nnnnn \cs_end:
     { \WCcount }
     { \l__wheelchart_lines_angle_pos_fp }
     { \l__wheelchart_lines_angle_shift_fp }
     { \l__wheelchart_lines_pos_fp }
     { \l__wheelchart_lines_sep_fp }
    -- ( \p { l__wheelchart_lines_1 } )
  \fp_compare:nNnF { \l__wheelchart_lines_ext_fp } = { 0 }
   {
    \str_case:enF { \pgfkeysvalueof { / wheelchart / lines~ext~fixed } }
     {
      { true }
       {
        --
         (
          {
           \fp_eval:n
            {
             \l__wheelchart_lines_ext_dir_int == 1
             ?
             \l__wheelchart_lines_ext_fixed_right_fp
             :
             \l__wheelchart_lines_ext_fixed_left_fp
            }
          }
          ,
          \y { l__wheelchart_lines_1 }
         )
       }
      { false }
       { --++ ( { \fp_eval:n { \l__wheelchart_lines_ext_dir_int * \l__wheelchart_lines_ext_fp } } , 0 ) }
     }
     {
      \pgfkeys
       {
        / errors / boolean~expected /. expanded =
         { lines~ext~fixed }
         { \pgfkeysvalueof { / wheelchart / lines~ext~fixed } }
       }
     }
    coordinate
     [
      shift =
       {
        (
         {
          \fp_eval:n
           {
            \l__wheelchart_lines_ext_dir_int * \cs:w g__wheelchart_data_sep_\WCcount _fp \cs_end:
           }
         } ,
         0
        )
       }
     ]
     ( g__wheelchart_data_coordinate )
   }
  ;
 }
\__wheelchart_if_text:nnn { data } { o }
```

```
{
  \fp_compare:nNnTF { \l__wheelchart_lines_ext_fp } = { 0 }
    {
      \__wheelchart_def_fp:nn { anchor_xsep } { anchor~xsep }
      \__wheelchart_def_fp:nn { anchor_ysep } { anchor~ysep }
      \pgfmathparse
        {
          ( \WCdataangle == 0 ? "west" \c_colon_str
          ( \WCdataangle == 90 ? "south" \c_colon_str
          ( \WCdataangle == 180 ? "east" \c_colon_str
          ( \WCdataangle == 270 ? "north" \c_colon_str
          ( \WCdataangle <= \fp_use:N \l__wheelchart_anchor_ysep_fp ? "west" \c_colon_str
          ( \WCdataangle < 90 - \fp_use:N \l__wheelchart_anchor_xsep_fp ? "south~west" \c_colon_str
          ( \WCdataangle <= 90 + \fp_use:N \l__wheelchart_anchor_xsep_fp ? "south" \c_colon_str
          ( \WCdataangle < 180 - \fp_use:N \l__wheelchart_anchor_ysep_fp ? "south~east" \c_colon_str
          ( \WCdataangle <= 180 + \fp_use:N \l__wheelchart_anchor_ysep_fp ? "east" \c_colon_str
          ( \WCdataangle < 270 - \fp_use:N \l__wheelchart_anchor_xsep_fp ? "north~east" \c_colon_str
          ( \WCdataangle <= 270 + \fp_use:N \l__wheelchart_anchor_xsep_fp ? "north" \c_colon_str
          ( \WCdataangle < 360 - \fp_use:N \l__wheelchart_anchor_ysep_fp ? "north~west" \c_colon_str
          "west"
          ))))))))))))
        }
      \tl_set:Ne \l__wheelchart_data_anchor_tl { \pgfmathresult }
      \coordinate
        [
          at =
            \cs:w __wheelchart_point_plot_\bool_to_str:N \l__wheelchart_plot_bool :nnnnn \cs_end:
              { \WCcount }
              { \cs:w g__wheelchart_data_angle_pos_\WCcount _fp \cs_end: }
              { \cs:w g__wheelchart_data_angle_shift_\WCcount _fp \cs_end: }
              { \cs:w g__wheelchart_data_pos_\WCcount _fp \cs_end: }
              {
                \cs:w g__wheelchart_data_sep_\WCcount _fp \cs_end: +
                (
                  \l__wheelchart_lines_fp == 0
                  ?
                  0
                  :
                  \l__wheelchart_lines_sep_fp + \l__wheelchart_lines_fp
```

```
         )
        }
     ]
     ( g__wheelchart_data_coordinate ) ;
   }
   {
    \pgfmathparse { \int_use:N \l__wheelchart_lines_ext_dir_int == 1 ? "right" \c_colon_str "left" }
    \tl_set:Ne \l__wheelchart_data_anchor_tl
     { \pgfkeysvalueof { / wheelchart / lines~ext~\pgfmathresult \c_space_tl anchor } }
   }
  \node [ anchor = \l__wheelchart_data_anchor_tl , align = left , / wheelchart / data_style ]
   at ( g__wheelchart_data_coordinate )
   { \pgfkeysvalueof { / wheelchart / data } } ;%a separate \node and not at the end of the \draw with lines_style
   %so that the key lines style is not passed to this \node
 }
\__wheelchart_def_fp:nn { arc_around_line } { arc~around~line }
\__wheelchart_def_fp:nn { arc_data_angle_pos } { arc~data~angle~pos }
\__wheelchart_def_fp:nn { arc_data_angle_shift } { arc~data~angle~shift }
\__wheelchart_def_fp:nn { arc_data_dir } { arc~data~dir }
%these are needed for arc data and arc
\pgfinterruptpicture%
 \fp_gset:Nn \g__wheelchart_half_ex_over_one_cm_fp { 0.5 ex / 1 cm }%
\endpgfinterruptpicture%
\fp_gset:Nn \g__wheelchart_arc_data_aux_ii_fp { 0 }
\__wheelchart_if_text:nnn { arc~data } {}
 {
  \cs_set:Npn \WCperc { \WCpercentagerounded { \, } { \% } }%so that \WCperc follows the arc if used in arc data
  %this redefinition of \WCperc is local to the group of arc data
  \__wheelchart_def_fp:nn { arc_data_pos } { arc~data~pos }
  \__wheelchart_def_fp:nn { arc_data_sep } { arc~data~sep }
  \__wheelchart_def_fp:nn { arc_data_line_sep_factor } { arc~data~line~sep~factor }
  \__wheelchart_def_fp:nn { arc_data_lines_pos } { arc~data~lines~pos }
  \__wheelchart_def_fp:nn { arc_data_lines_shift } { arc~data~lines~shift }
  \cs:w seq_set_split:Nn\pgfkeysvalueof { / wheelchart / arc~data~expand } \cs_end:
   \l__wheelchart_arc_data_seq
   { \\ }
   { \pgfkeysvalueof { / wheelchart / arc~data } }
  \seq_map_indexed_inline:Nn \l__wheelchart_arc_data_seq
   {
```

```
\fp_set:Nn \l__wheelchart_arc_data_text_pos_fp
  {
    \l__wheelchart_arc_data_pos_fp + 4 * \g__wheelchart_slices_orientation_fp *
      (
        ####1 - 1 - \l__wheelchart_arc_data_lines_pos_fp * ( \seq_count:N \l__wheelchart_arc_data_seq - 1 )
        + \l__wheelchart_arc_data_lines_shift_fp
      )
      * sign ( \l__wheelchart_arc_data_dir_fp )
      * \l__wheelchart_arc_data_line_sep_factor_fp * \g__wheelchart_half_ex_over_one_cm_fp
      /
      (
        \cs:w g__wheelchart_outer_radius_\WCcount _fp \cs_end:
        - \cs:w g__wheelchart_inner_radius_\WCcount _fp \cs_end:
        + 2 * \l__wheelchart_arc_data_sep_fp
      )
  }%the sign is needed because \l__wheelchart_arc_data_dir_fp is not necessarily 1 or -1
\hbox_set:Nn \l__wheelchart_arc_data_box
  { \pgfinterruptpicture {####2} \endpgfinterruptpicture }
\fp_gset:Nn \g__wheelchart_arc_data_slice_length_fp { 1 }%this is necessary if the value is 0
\bool_if:NTF \l__wheelchart_plot_bool
  {
    \__wheelchart_convex_comb_coord_plot:nnnnnnn
      {
        overlay ,
        decorate ,
        decoration =
          {
            text~along~path ,
            text =
              { {} { \fp_gset:Nn \g__wheelchart_arc_data_slice_length_fp { \pgfdecoratedpathlength } } } ,
            raise = -0.5 ex ,
            text~align = \l__wheelchart_arc_data_align_tl ,
            / wheelchart / arc_data_style
          }
      }%get the length of the path
      %note the option overlay so that this does not increase the bounding box
      %note the {} at the start of text and the braces around \fp_gset:Nn ...
      %so that the compilation does not stall
      { 0 }
  { 1 }
  { \l__wheelchart_plot_variable_tl }
  { 0 }
  { \l__wheelchart_arc_data_text_pos_fp }
  { \l__wheelchart_arc_data_sep_fp }
\__wheelchart_convex_comb_coord_plot:nnnnnnn
  {
    decorate ,
    decoration =
      {
        text~along~path ,
        text =
          {
            { { \fp_gset:Nn \g__wheelchart_arc_data_aux_i_fp { \the \pgfdecoratedcompleteddistance } } }
            ####2
            {
              {
                \fp_gset:Nn \g__wheelchart_arc_data_aux_i_fp
                  {
                    ( \the \pgfdecoratedcompleteddistance - \g__wheelchart_arc_data_aux_i_fp )
                    / \g__wheelchart_arc_data_slice_length_fp
                  }
              }
            }
          } ,
        raise = -0.5 ex ,
        text~align = \l__wheelchart_arc_data_align_tl ,
        / wheelchart / arc_data_style
      }
  }%note the double braces around \fp_gset:Nn ...
  %so that for example arc data=text {\qty{5}{\percent}} is allowed
  %note that \def\mytext{}\path[decorate,decoration={text along path,text={\mytext}}] (0,0)--(1,1);
  %gives the message Missing character: There is no ... in font nullfont!, then
  %text={\pgfkeysvalueof { / wheelchart / arc~data }{}} can be used
  %if the \fp_gset:Nn ... would not be present
  { \fp_use:c { c__wheelchart_arc_data_start_factor_\l__wheelchart_arc_data_align_tl _fp } }
  { \fp_use:c { c__wheelchart_arc_data_end_factor_\l__wheelchart_arc_data_align_tl _fp } }
  {
    \l__wheelchart_plot_variable_tl * \l__wheelchart_arc_data_dir_fp * 1.1
```

```
   * ( \dim_to_fp:n { \box_wd:N \l__wheelchart_arc_data_box } / \g__wheelchart_arc_data_slice_length_fp )
   + \l__wheelchart_arc_data_angle_pos_fp
  }
  { \l__wheelchart_arc_data_angle_shift_fp }
  { \l__wheelchart_arc_data_text_pos_fp }
  { \l__wheelchart_arc_data_sep_fp }
}
{
 \fp_set:Nn \l__wheelchart_arc_data_radius_plot_false_fp
  {
   \__wheelchart_def_radius:nnn
    { \WCcount }
    { \l__wheelchart_arc_data_text_pos_fp }
    { \l__wheelchart_arc_data_sep_fp + \g__wheelchart_half_ex_over_one_cm_fp }
  }
 \fp_set:Nn \l__wheelchart_arc_data_total_angle_plot_false_fp
  {
   \box_wd:N \l__wheelchart_arc_data_box * 1.1 /
    (
     sqrt ( abs ( \l__wheelchart_coord_determinant_fp ) )%this is necessary if an option such as
     %[x={(-0.5,0)},y={(0,0.5)}] is given to the tikzpicture
     * \l__wheelchart_arc_data_radius_plot_false_fp * deg
    )
  }
 \fp_set:Nn \l__wheelchart_arc_data_start_angle_plot_false_fp
  {
   \__wheelchart_def_angle_plot_false:nnnnn
    { \WCcount }
    { \l__wheelchart_arc_data_angle_pos_fp }
    { \l__wheelchart_arc_data_angle_shift_fp }
    { \l__wheelchart_arc_data_text_pos_fp }
    { \l__wheelchart_arc_data_sep_fp + \g__wheelchart_half_ex_over_one_cm_fp }
   + \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_arc_data_dir_fp
    * \cs:w c__wheelchart_arc_data_start_factor_\l__wheelchart_arc_data_align_tl _fp \cs_end:
    * \l__wheelchart_arc_data_total_angle_plot_false_fp
  }
 \path
  [
   decorate ,
 decoration =
  {
   text~along~path ,
   text =
    {
     { { \fp_gset:Nn \g__wheelchart_arc_data_aux_i_fp { \the \pgfdecoratedcompleteddistance } } }
     ####2
     {
      {
       \fp_gset:Nn \g__wheelchart_arc_data_aux_i_fp
        {
         ( \the \pgfdecoratedcompleteddistance - \g__wheelchart_arc_data_aux_i_fp )
         / \l__wheelchart_arc_data_radius_plot_false_fp
        }
      }
     }
    } ,
   raise = -0.5 ex ,
   text~align = \l__wheelchart_arc_data_align_tl ,
   / wheelchart / arc_data_style
  }
]
(
 \fp_use:N \l__wheelchart_arc_data_start_angle_plot_false_fp
 \c_colon_str
 \fp_use:N \l__wheelchart_arc_data_radius_plot_false_fp
)
arc
 [
  start~angle = \fp_use:N \l__wheelchart_arc_data_start_angle_plot_false_fp ,
  end~angle =
   \fp_eval:n
    {
     \l__wheelchart_arc_data_start_angle_plot_false_fp
     + \l__wheelchart_counter_or_clockwise_fp * \l__wheelchart_arc_data_dir_fp
     * \l__wheelchart_arc_data_total_angle_plot_false_fp
    }
   ,
  radius = \fp_use:N \l__wheelchart_arc_data_radius_plot_false_fp
```

```
        ]
       ;
     }
    \fp_compare:nNnF { \g__wheelchart_arc_data_aux_i_fp } > { 0 }
     {
      \PackageWarning { wheelchart }
       {
        The~arc~data~in~slice~\WCcount \c_space_tl did~(possibly)~not~fit.~
        Increase~the~absolute~value~of~arc~data~dir.
       }%refer to \WCcount and not to \pgfkeysvalueof { / wheelchart / arc~data }
       %because the latter is not necessarily unique
     }
    \int_compare:nNnT {####1} = { \fp_use:N \l__wheelchart_arc_around_line_fp }
     { \fp_gset_eq:NN \g__wheelchart_arc_data_aux_ii_fp \g__wheelchart_arc_data_aux_i_fp }
   }
 }
\bool_if:NT \l__wheelchart_arc_bool
 {
  \__wheelchart_def_fp:nn { arc_pos } { arc~pos }
  \__wheelchart_def_fp:nn { arc_sep } { arc~sep }
  \str_case:enF { \pgfkeysvalueof { / wheelchart / arc~around~text } }
   {
    { true }
     {
      \bool_if:NTF \l__wheelchart_plot_bool
       {
        \__wheelchart_arc_around_text_plot_true:nnn { first } { -1 } { 0 }
        \__wheelchart_arc_around_text_plot_true:nnn { second } { 1 } { 1 }
       }
       {
        \fp_gset:Nn \g__wheelchart_arc_data_aux_ii_fp
         {
          \g__wheelchart_arc_data_aux_ii_fp
          / ( sqrt ( abs ( \l__wheelchart_coord_determinant_fp ) ) * deg )
         }
        \fp_set:Nn \l__wheelchart_arc_radius_fp
         {
          \__wheelchart_def_radius:nnn
           { \WCcount }
       { \l__wheelchart_arc_pos_fp }
       { \l__wheelchart_arc_sep_fp + \g__wheelchart_half_ex_over_one_cm_fp }
     }
    \__wheelchart_arc_around_text_plot_false:nn { first } { 0 }
    \__wheelchart_arc_around_text_plot_false:nn { second } { 1 }
   }
 }
{ false }
 {
  \bool_if:NTF \l__wheelchart_plot_bool
   {
    \__wheelchart_convex_comb_coord_plot:nnnnnnn
     { draw , / wheelchart / arc_style }
     { 0 }
     { 1 }
     { \l__wheelchart_plot_variable_tl }
     { 0 }
     { \l__wheelchart_arc_pos_fp }
     { \l__wheelchart_arc_sep_fp }
   }
   {
    \fp_set:Nn \l__wheelchart_arc_radius_fp
     {
      \__wheelchart_def_radius:nnn
       { \WCcount }
       { \l__wheelchart_arc_pos_fp }
       { \l__wheelchart_arc_sep_fp + \g__wheelchart_half_ex_over_one_cm_fp }
     }
    \fp_set:Nn \l__wheelchart_arc_start_angle_fp
     {
      \__wheelchart_def_angle_plot_false:nnnnn
       { \WCcount }
       { 0 }
       { 0 }
       { \l__wheelchart_arc_pos_fp }
       { \l__wheelchart_arc_sep_fp + \g__wheelchart_half_ex_over_one_cm_fp }
     }
    \path
     [ draw , / wheelchart / arc_style ]
```

```
            ( \fp_use:N \l__wheelchart_arc_start_angle_fp \c_colon_str \fp_use:N \l__wheelchart_arc_radius_fp )
            arc
             [
              start~angle = \fp_use:N \l__wheelchart_arc_start_angle_fp ,
              end~angle =
               \__wheelchart_def_angle_plot_false:nnnnn
                { \WCcount }
                { 1 }
                { 0 }
                { \l__wheelchart_arc_pos_fp }
                { \l__wheelchart_arc_sep_fp + \g__wheelchart_half_ex_over_one_cm_fp } ,
              radius = \fp_use:N \l__wheelchart_arc_radius_fp
             ]
            ;
          }
        }
      }
      {
       \pgfkeys
        {
         / errors / boolean~expected /. expanded =
          { arc~around~text }
          { \pgfkeysvalueof { / wheelchart / arc~around~text } }
        }
      }
    }
   \__wheelchart_inner_and_wheel_data:n { inner }
   \__wheelchart_inner_and_wheel_data:n { wheel }
   \pgfkeysvalueof { / wheelchart / legend~entry }
  }
 \bool_if:NT \l__wheelchart_legend_row_bool
  { \__wheelchart_def_WClegend: }
 \__wheelchart_if_text:nnn { middle } { o }
  { \node [ align = center , / wheelchart / middle_style ] at ( 0 , 0 ) { \pgfkeysvalueof { / wheelchart / middle } } ; }
 \pgfkeysvalueof { / wheelchart / legend }
\end { scope }
\__wheelchart_caption_and_title:nnnnn
 { caption~left } { north~west } { left } { \g__wheelchart_name_tl .south~west } { -1 }
\__wheelchart_caption_and_title:nnnnn
```

```
              { caption } { north } { center } { \g__wheelchart_name_tl .south -| 0 , 0 } { -1 }
            \__wheelchart_caption_and_title:nnnnn
              { title~left } { south~west } { left } { \g__wheelchart_name_tl .north~west } { 1 }
            \__wheelchart_caption_and_title:nnnnn
              { title } { south } { center } { \g__wheelchart_name_tl .north -| 0 , 0 } { 1 }
          \end { scope }
        }
    }
  }

\endinput
```