The Ti*k*Z-Extensions Package

Manual for version 0.5

https://github.com/Qrrbrbirlbel/tikz-extensions

Qrrbrbirlbel

December 27, 2022

Contents

Part I
Introduction

1 Usage

This package is called `tikz-ext`, however, one can't load it via `\usepackage`.[1] Instead, this package consists of multiple PGF and Ti*k*Z libraries which are loaded by either `\usepgflibrary` or `\usetikzlibrary`.

2 Why do we need it?

Since I have been answering questions on TeX.sx I've noticed that some questions come up again and again, every time with a slightly different approach on how to solve them.

I don't like reinventing the wheel which is why I've gathered the solutions of my answers in this package.

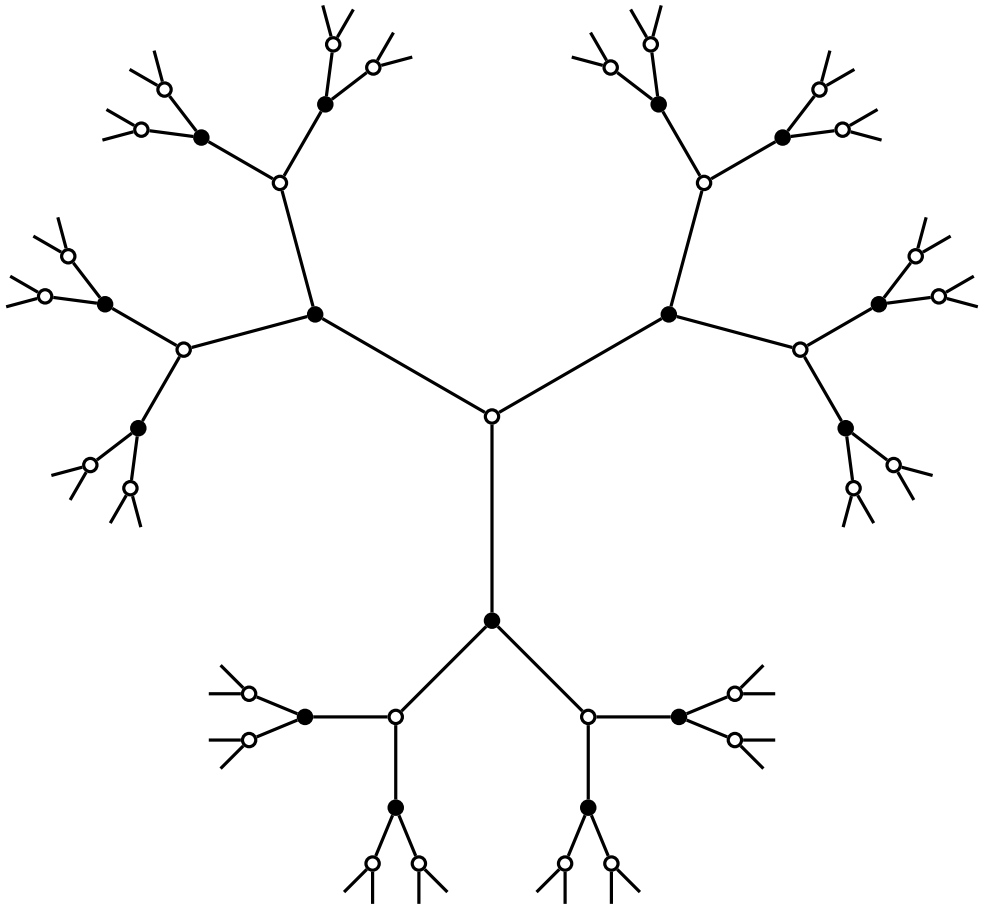
3 Having problems?

Don't hesitate to open an issue on GitHub.

[1] Except for `calendar-ext` of section 23.

Part II
Ti*k*Z Libraries

These libraries only work with Ti*k*Z.



4 Calendar

TikZ Library ext.calendar-plus

```
\usetikzlibrary{ext.calendar-plus} % LaTeX and plain TeX
\usetikzlibrary[ext.calendar-plus] % ConTeXt
```

This library extends the TikZ library calendar.

Q & A: [8, 9, 3] & [21, 37, 36]

4.1 Value-keys and nestable if key

The values of following keys are originally stored in some macros that are not accessible by the user. These are now simple value-keys. The @-protected macros are still available, of course.

/tikz/day xshift (initially 3ex)

/tikz/day yshift (initially 3.5ex)

/tikz/month xshift (initially 9ex)

/tikz/month yshift (initially 9ex)

It is now also possible to nest /tikz/if occurrences.

/tikz/if=(⟨*conditions*⟩)⟨*code or options*⟩else⟨*else code or options*⟩ (no default)

4.2 PGFmath functions

weeksinmonthofyear(*first weekday*, *month*, *year*)
\pgfmathweeksinmonthofyear{*first weekday*}{*month*}{*year*}

Returns the number of (partial) weeks in the month *month* of year *year* when this month begins on a *first weekday*.

lastdayinmonthofyear(*month*, *year*)
\pgfmathlastdayinmonthofyear{*month*}{*year*}

Returns the last day (28, 29, 30 or 31) of month *month* of year *year*.

4.3 Week numbering (ISO 8601)

The actual week number algorithm is implemented by the pgfcalendar-ext package/module in section 23.2.

/tikz/week code=⟨*code*⟩ (no default)

Works like /tikz/day code or /tikz/month code, only for weeks.

/tikz/week text=⟨*text*⟩ (no default)

Works like /tikz/day text or /tikz/month text, only for weeks.

/tikz/every week (style, no value)

Works like /tikz/every day or /tikz/every month, only for weeks.

/tikz/week label left (style, no value)

Places the week label to the left of the first day of the month. (For week list and month list where a week does not start on a Monday, the position is chosen "as if" the week had started on a Monday – which is usually exactly what you want.)

July

26					1	2	3
27	4	5	6	7	8	9	10
28	11	12	13	14	15	16	17
29	18	19	20	21	22	23	24
30	25	26	27	28	29	30	31

```
\usetikzlibrary {ext.calendar-plus}
\tikz
  \calendar [week list, month label above
centered,
             dates=2022-07-01 to 2022-07-31,
             week label left,
             every week/.append
style={gray!50!black,font=\sffamily}];
```

5 Nodes

Ti*k*Z Library `ext.layers`

```
\usetikzlibrary{ext.layers} % LaTeX and plain TeX
\usetikzlibrary[ext.layers] % ConTeXt
```

This library extends Ti*k*Z's functionalities to put nodes, edges, matrices and pics on a separate layer without having to use the `pgfonlayer` environment.

Consider this library experimental. If you can, avoid it and use the `pgfonlayer` environment or change the drawing order.

5.1 Internal keys

`/tikz-ext/patch`=⟨*specification*⟩ (no default)

Since this library is experimental, its functionality needs to be activated explicitly. The ⟨*specification*⟩ is one of

- `node`,
- `matrix`,
- `pic`[2],
- `edge` or
- `all` which applies all the patches at once.

These keys only work when a patch is applied but don't need to be used since the patching activated specific

`/tikz-ext/layers/in box`=⟨*box*⟩ (no default)

`/tikz-ext/layers/on layer`=⟨*layer*⟩ (no default)

5.2 User-level keys

`/tikz/node on layer`=⟨*layer*⟩ (no default)

`/tikz/node in box`=⟨*box*⟩ (no default)

`/tikz/matrix on layer`=⟨*layer*⟩ (no default)

`/tikz/matrix in box`=⟨*box*⟩ (no default)

`/tikz/edge on layer`=⟨*layer*⟩ (no default)

`/tikz/edge in box`=⟨*box*⟩ (no default)

`/tikz/pic on layer`=⟨*layer*⟩ (no default)

`/tikz/pic in box`=⟨*box*⟩ (no default)

[2] Only the normal `/tikz/pics/code` can be placed on different layers. Both `/tikz/pics/background code` and `/tikz/pics/foreground code` will not be affected.

6 Node Families

TikZ Library ext.node-families

```
\usetikzlibrary{ext.node-families} % LaTeX and plain TeX
\usetikzlibrary[ext.node-families] % ConTeXt
```

With this library the user can instruct multiple nodes to have the same width, height, text width, text height or text width. This uses the hook `/tikz/execute at end picture` to write the nodes' measurements to the AUX file.

Unfortunately, this does not work with the `external` library.[3]

Q & A: [10] & [23]

Before we get to the interesting keys, a common prefix can be set for the families' names. Initially this is `\pgfpictureid-` so that families of different pictures don't interact.

`/tikz/node family/prefix`=⟨*prefix*⟩ (no default, initially `\pgfpictureid-`)

The family names are prefixed with the value of `/tikz/node family/prefix`.

6.1 Text Box

The following keys – when setup, see below – work with every shape with one single node part.[4] Initially though, only `circle` and `rectangle` are set up that way.

`/tikz/node family/text height`=⟨*name*⟩ (no default, initially {})

Nodes with the same ⟨*name*⟩ will have the same text height. An empty ⟨*name*⟩ disables the evaluation by the library.

`/tikz/node family/text depth`=⟨*name*⟩ (no default, initially {})

Nodes with the same ⟨*name*⟩ will have the same text depth. An empty ⟨*name*⟩ disables the evaluation by the library.

`/tikz/node family/text width`=⟨*name*⟩ (no default, initially {})

Nodes with the same ⟨*name*⟩ will have the same text width. An empty ⟨*name*⟩ disables the evaluation by the library.

`/tikz/node family/text`=⟨*name*⟩ (no default)

Sets `text height`, `text depth` and `text width`.

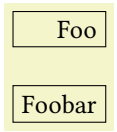
Since the width of the node's content's box is setup much earlier, the previous key only extends the width of that box which would make the text seem as if it where aligned to the left. With `text width family align` this can changed.

[3] First of all, I can't figure out how to use the AUX file during externalization since it gets written to the LOG instead. And then there's the question about how `external` would notice the need to export the picture again until it's stable …

[4] Technically, it will also work with shapes with multiple node parts but it will only affect the main node part.

`/tikz/node family/text width align`=⟨*alignment*⟩ (no default, initially `center`)

⟨*alignment*⟩ is one of `left`, `center` or `right`.




```
\usetikzlibrary {positioning,ext.node-families}
\tikzexternaldisable % ext.node-families does not work with active externalization
\begin{tikzpicture}[nodes={rectangle, draw, node family={text width=manual, text width align=right}}]
\node (a) {Foo};
\node[below=of a] (b) {Foobar};
\end{tikzpicture}
```

`/tikz/node family/setup shape`=⟨*shape*⟩ (no default)

This adds instructions to the ⟨*shape*⟩'s definition which adjust the text box's dimensions according to the family.

This should be only used once per shape.

6.2 Minimum Width/Height

While the keys of the previous subsection work well enough for nodes of the same shape (and the same `inner seps`), for different node shapes the text box dimensions will be used differently for the node's total dimension.

For this, the following keys are necessary. When one of the keys are used the values of `minimum width` and/or `minimum height` are set to `nf_width` or `nf_height` respectively.

`/tikz/node family/width`=⟨*name*⟩ (no default, initially `{}`)

Nodes with the same ⟨*name*⟩ will have the same `/pgf/minimum width` . An empty ⟨*name*⟩ disables the evaluation by the library.


```
\usetikzlibrary {positioning,ext.node-families}
\tikzexternaldisable % ext.node-families does not work with active externalization
\begin{tikzpicture}[nodes={rectangle, draw, node family/width=manual}]
\node (a) {Foo};
\node[below=of a] (b) {Foobar};
\end{tikzpicture}
```

`/tikz/node family/height`=⟨*name*⟩ (no default, initially `{}`)

Nodes with the same ⟨*name*⟩ will have the same `/pgf/minimum height` . An empty ⟨*name*⟩ disables the evaluation by the library.

`/tikz/node family/size`=⟨*name*⟩ (no default)

Sets both `height` and `width`.

6.3 More shapes that support the keys `width` and `height`

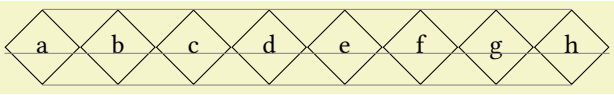
TikZ Library `ext.node-families.shapes.geometric`

```
\usetikzlibrary{ext.node-families.shapes.geometric} % LaTeX and plain TeX
\usetikzlibrary[ext.node-families.shapes.geometric] % ConTeXt
```

This library adds support for the keys `/tikz/node family/width` and `/tikz/node family/height` for the shapes of the PGF library `shapes.geometric`.

Q: [16]

The shapes are also setup for the keys from subsection 6.1.




```
\usetikzlibrary {ext.node-families.shapes.geometric}
\tikzexternaldisable % ext.node-families does not work with active externalization
\begin{tikzpicture}
\foreach \cnt[count=\Cnt] in {a,...,h}
  \node[draw, diamond, node family/text=aTOh] (\cnt)
    at (right:\Cnt) {\cnt};
\draw[help lines] (a.south) -- (h.south) (a.north) -- (h.north) (a.base-|a.west) -- (h.base-|h.east);
\end{tikzpicture}
```

7 Nodes

TikZ Library `ext.nodes`

```
\usetikzlibrary{ext.nodes} % LaTeX and plain TeX
\usetikzlibrary[ext.nodes] % ConTeXt
```

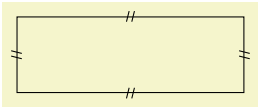
This library extends TikZ's functionalities when it comes to nodes.

Q & A: [7, 13] & [24, 32]

7.1 Pic as a node

`/tikz/pic=`⟨*boolean*⟩ (default `true, initially false`)

This key allows one to use a pic where usually only nodes are accepted, for example as a label.



```
\usetikzlibrary {ext.nodes, ext.misc}
\begin{tikzpicture}[
  slsl/.pic={\draw (-2pt, 1.5pt) -- (2pt, .5pt)
                   (2pt, -1.5pt) -- (-2pt, -
.5pt);}]
\node[draw, minimum width=3cm, minimum height=1cm,
     label={[pic           ]  east:slsl},
     label={[pic, rotate=90] north:slsl},
     label={[pic           ] west:slsl},
     label={[pic, rotate=-90]south:slsl}]{};
\end{tikzpicture}
```

7.2 Nodes on paths

When nodes are placed along paths they don't interrupt the path at that place. The decoration `markings` and its `/pgf/decoration/mark connection node` key can help but only works for straight paths and doesn't play nicely with arrow tips.

This library provides alternatives. These are separated into straight paths, i. e. `--`, and everything else (including any `to path`).

7.2.1 Nodes on Lines

`/tikz/node on line=`⟨*anchor specification*⟩ (style, default `{}`)

This installs a `/tikz/to path` that places *one* node along a straight line but connect the line with it.

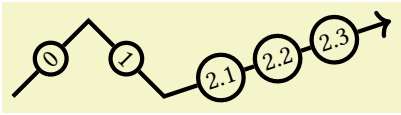
This allows a node to be placed *on* a straight line without having to use `fill = white` or similar tricks to make the line disappear beneath the node.

The optional ⟨*anchor specification*⟩ allows to specify the anchors to which the line should connect. It allows one or two anchors divided by `and` to be specified.

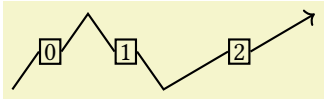
`/tikz/nodes on line` (style, no value)

This is similar to the previous key but allows multiple nodes to be placed on a straight line *if* they are in the correct order (from start to target), don't overlap with each other, the start or the target.

It allows *no* anchor specification.




```
\usetikzlibrary {ext.nodes, quotes}
\tikz[inner sep=.15em, circle, nodes=draw, sloped]
  \draw[ultra thick, ->, node on line] (0,0) to["0"] (1,1)
                                        to["1"] (2,0)
    to[nodes on line, "2.1" near start, "2.2", "2.3" near end] (5,1);
```




```
\usetikzlibrary {ext.nodes, quotes}
\tikz[inner sep=.15em, nodes=draw]
  \draw[thick, ->, node on line=west and east]
     (0,0) to["0"] (1,1)
           to["1"] (2,0)
           to["2"] (4,1);
```

7.2.2 Nodes on Curves

The following keys need the `intersections` and the `spath3` [41] library to be loaded. They will not be automatically loaded by this library.

Any `/pgf/outer sep` will be ignored.

If you can, use `fill=`⟨*bg color*⟩ instead of these keys, it will be much faster and easier.

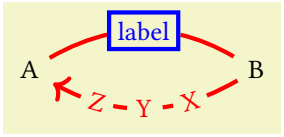
`/tikz/nodes on curve=`⟨*to path*⟩ (style, default `line to`)

Similar to `/tikz/nodes on line` , this key allows to have nodes on arbitrary paths.

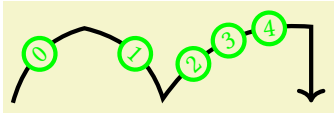
This is not suitable for paths connecting nodes.

`/tikz/nodes on curve'=`⟨*to path*⟩ (style, default `line to`)

As above but suitable for connecting nodes.




```
\usetikzlibrary {ext.nodes, intersections, quotes, spath3}
\begin{tikzpicture}[ultra thick]
  \node (A) at (0, 0) {A} ;
  \node (B) at (3, 0) {B} ;
  \draw [red, ->, nodes on curve'=bend left]
    (A) to node[blue,draw]{label} (B)
        to ["X" {sloped, near start},
            "Z" {sloped, near end},
            "Y"] (A);
\end{tikzpicture}
```



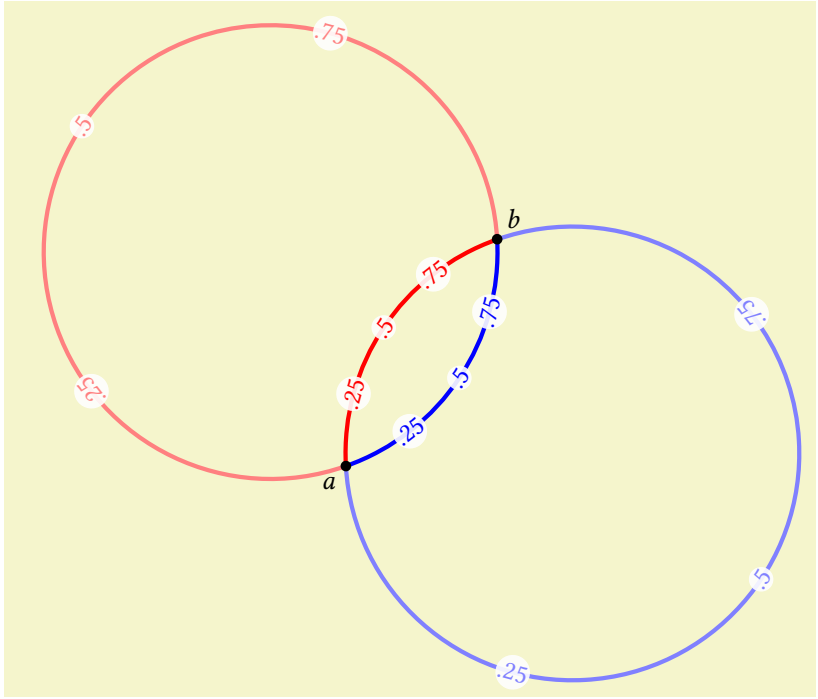

```
\usetikzlibrary {ext.nodes, intersections, quotes, spath3}
\tikz[inner sep=.15em, circle, nodes={draw, green}, sloped, ultra thick]
  \draw[->, nodes on curve=bend left] (0,0) to["0"] (1,1)
                                            to["1"] (2,0)
              to["2" near start, "3", "4" near end] (4,1)
                                            -- ++(down:1);
```

8 Arc *to* a point

TikZ Library `ext.paths.arcto`

```
\usetikzlibrary{ext.paths.arcto} % LaTeX and plain TeX
\usetikzlibrary[ext.paths.arcto] % ConTeXt
```

This library adds the new path operation `arc to` that specifies an arc *to* a point – without the user having to specify any angles.




```
\usetikzlibrary {ext.paths.arcto}
\begin{tikzpicture}[ultra thick,dot/.style={label={#1}}]
\coordinate[dot=below left:$a$] (a) at (0,0);
\coordinate[dot=above right:$b$] (b) at (2,3);
\begin{scope}[
  radius=3,
  nodes={
    shape=circle,
    fill=white,
    fill opacity=.9,
    text opacity=1,
    inner sep=+0pt,
    sloped,
    allow upside down
  }]
\draw[blue]    (a) arc to[]
  node[near start] {.25} node {.5} node[near end] {.75} (b);
\draw[red]     (a) arc to[clockwise]
  node[near start] {.25} node {.5} node[near end] {.75} (b);
\draw[blue!50] (a) arc to[large]
  node[near start] {.25} node {.5} node[near end] {.75} (b);
\draw[red!50]  (a) arc to[large, clockwise]
  node[near start] {.25} node {.5} node[near end] {.75} (b);
\end{scope}

\fill[radius=2pt] (a) circle[] (b) circle[];
\end{tikzpicture}
```

`\path` ... `arc to[`⟨*options*⟩`]`⟨*coordinate or cycle*⟩ ...;

When this operation is used, the path gets extended by an arc that goes through the current point and ⟨*coordinate*⟩.

For two points there exist two circles or four arcs that go through or connect these two points. Which one of these is constructed is determined by the following options that can be used inside of ⟨*options*⟩.

`/tikz/arc to/clockwise` (style, no value)

This constructs an arc that goes clockwise.

`/tikz/arc to/counter clockwise` (style, no value)

This constructs an arc that goes counter clockwise.

This is the default.

`/tikz/arc to/large` (style, no value)

This constructs an arc whose angle is larger than 180°.

`/tikz/arc to/small` (style, no value)

This constructs an arc whose angle is smaller than 180°.

`/tikz/arc to/rotate`=⟨*degree*⟩ (no default)

Rotates the arc by ⟨*degree*⟩. This is only noticeable when `x radius` and `y radius` are different.

`/tikz/arc to/x radius`=⟨*value*⟩ (no default)

This forwards the ⟨*value*⟩ to `/tikz/x radius` . Its ⟨*value*⟩ is used for the radius of the arc.

`/tikz/arc to/y radius`=⟨*value*⟩ (no default)

This forwards the ⟨*value*⟩ to `/tikz/y radius` . Its ⟨*value*⟩ is used for the radius of the arc.

`/tikz/arc to/radius`=⟨*value*⟩ (no default)

This forwards the ⟨*value*⟩ to both `/tikz/x radius` and `/tikz/y radius`. Its ⟨*value*⟩ is used for radius of the arc.

`/tikz/every arc to` (style, no value)

After `/tikz/every arc` this will also be applied before any ⟨*options*⟩ are set.

It should be noted that this uses `\pgfpatharcto` for which the Ti*k*Z manual warns:

> *The internal computations necessary for this command are numerically very unstable. In particular, the arc will not always really end at the ⟨target coordinate⟩, but may be off by up to several points. A more precise positioning is currently infeasible due to TEX's numerical weaknesses. The only case it works quite nicely is when the resulting angle is a multiple of* 90°.

The `arc to` path operation will also work only in the `canvas` coordinate system. The lengths of the vectors $(1, 0)$ and $(0, 1)$ will be used for the calculation of the radii but no further consideration is done.

9 More Horizontal and Vertical Lines

TikZ Library `ext.paths.ortho`

```
\usetikzlibrary{ext.paths.ortho} % LaTeX and plain TeX
\usetikzlibrary[ext.paths.ortho] % ConTeXt
```

This library adds new path specifications `|-|`, `-|-` as well as `r-ud`, `r-du`, `r-lr` and `r-rl`.

9.1 Zig-Zag

Similar to the path operations `|-` and `-|` this library adds the path operations `|-|` and `-|-`.

`\path` ... `|-|[`⟨*options*⟩`]`⟨*coordinate or cycle*⟩ ...;

This operation means “first vertical, then horizontal and then vertical again”.

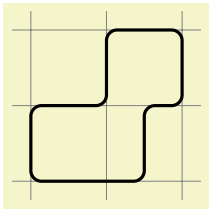
`\path` ... `-|-[`⟨*options*⟩`]`⟨*coordinate or cycle*⟩ ...;

This operation means “first horizontal, then vertical and then horizontal again”.

`/tikz/ortho/ratio`=⟨*ratio*⟩ (no default, initially `0.5`)

This sets the ratio for the middle part of the Zig-Zag connection.

For values ⟨*ratio*⟩ < 0 and ⟨*ratio*⟩ > 1 the Zig-Zag lines will look more like Zig-Zig lines.

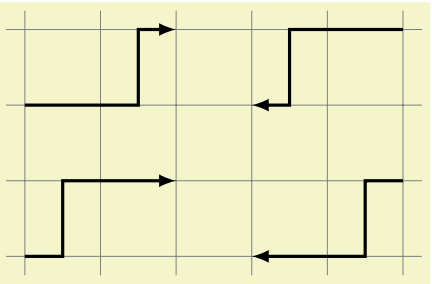


```
\usetikzlibrary {paths.ortho}
\begin{tikzpicture}[very thick, rounded corners]
\draw[help lines] (-.25, -1.25) grid (2.25, 1.25);
\draw (0, 0) -|-            (2, 1) --
      (2, 0) -|-[ratio=.25] (0,-1) -- cycle;
\end{tikzpicture}
```

`/tikz/ortho/distance`=⟨*distance*⟩ (no default)

This sets the distance between the start point and the middle part of the Zig-Zag connection.

For values ⟨*distance*⟩ < 0 the distance will be used for the target coordinate.

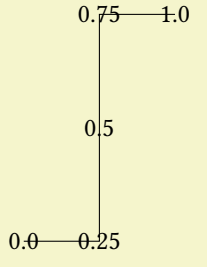


```
\usetikzlibrary {ext.paths.ortho}
\begin{tikzpicture}[very thick,-latex]
\draw[help lines,-] (-.25, -.25) grid (5.25, 3.25);
\draw (0, 0) -|-[distance= .5cm] ++(2, 1);
\draw (0, 2) -|-[distance=-.5cm] ++(2, 1);

\tikzset{xshift=3cm}
\draw (2, 1) -|-[distance= .5cm] ++(-2, -1);
\draw (2, 3) -|-[distance=-.5cm] ++(-2, -1);
\end{tikzpicture}
```

`/tikz/ortho/from center`=⟨*true or false*⟩ (default `true`)

When nodes get connected the placement of the middle part of the Zig-Zag and the Zig-Zig (see below) connections will be calculated from the border of these nodes. The middle part of the connections can be calculated from the nodes' center if this key is set to `true`.

New timers are setup for both the Zig-Zag and the Zig-Zig connections, these can be configured through the following keys.




```
\usetikzlibrary {paths.ortho}
\tikz \draw (0,0) -|- (2,3)
  foreach \p in {0.0, 0.25, 0.5, 0.75, 1.0}{
    node [pos=\p] {\p}};
```

`/tikz/ortho/spacing`=⟨*number*⟩ (no default, initially 4)

Unless ⟨*number*⟩ = 0 is set

- `pos = 0` will be at the start,
- `pos = 1` will be at the end,
- `pos =` $\frac{1}{\langle number \rangle}$ will be at the first kink,
- `pos =` $\frac{\langle number \rangle - 1}{\langle number \rangle}$ will be at the second kink and
- `pos = .5` will be in the middle of the middle part of the connection.

If ⟨*number*⟩ = 0 then

- `pos = -1` will be at the start,
- `pos = 2` will be at the end,
- `pos = 0` will be at the first kink,
- `pos = 1` will be at the second kink and
- `pos = .5` will still be in the middle of the middle part of the connection.

`/tikz/ortho/middle 0 to 1` (no value)

This is an alias for `spacing = 0`.

9.2 Zig-Zig

`\path ... r-ud[`⟨*options*⟩`]`⟨*coordinate or cycle*⟩ `...;`

This operation means "first up, then horizontal and then down".

`/tikz/ortho/ud distance=`⟨*length*⟩ (no default, initially `.5cm`)

This sets the distance between the start and the horizontal line to ⟨*length*⟩.

`\path ... r-du[`⟨*options*⟩`]`⟨*coordinate or cycle*⟩ `...;`

This operation means "first down, then horizontal and then up".

`/tikz/ortho/du distance=`⟨*length*⟩ (no default, initially `.5cm`)

This sets the distance between the start and the horizontal line to ⟨*length*⟩.

`\path ... r-lr[`⟨*options*⟩`]`⟨*coordinate or cycle*⟩ `...;`

This operation means "left down, then vertical and then right".

`/tikz/ortho/lr distance=`⟨*length*⟩ (no default, initially `.5cm`)

This sets the distance between the start and the vertical line to ⟨*length*⟩.

`\path ... r-rl[`⟨*options*⟩`]`⟨*coordinate or cycle*⟩ `...;`

This operation means "first right, then vertical and then down".

`/tikz/ortho/rl distance=`⟨*length*⟩ (no default, initially `.5cm`)

This sets the distance between the start and the vertical line to ⟨*length*⟩.

All distances can be set with one key.

`/tikz/ortho/udlr distance=`⟨*length*⟩ (no default)

Sets all the previous distances to the same value ⟨*length*⟩.

9.3 Even more Horizontal and Vertical Lines

The following keys can be used to access vertical and horizontal line path operations.

`/tikz/horizontal vertical` (style, no value)

This installs `to path = -| (\tikztotarget) \tikztonodes` that can be used with the path operations `to` or `edge`.

`/tikz/vertical horizontal` (style, no value)

This installs `to path = |- (\tikztotarget) \tikztonodes` that can be used with the path operations `to` or `edge`.

`/tikz/horizontal vertical horizontal` (style, no value)

This installs `to path = -|- (\tikztotarget) \tikztonodes` that can be used with the path operations `to` or `edge`.

`/tikz/vertical horizontal vertical` (style, no value)

This installs `to path = |-| (\tikztotarget) \tikztonodes` that can be used with the path operations `to` or `edge`.

When connecting rectangular nodes, these keys could be useful as well. They all need to be given to a `to` or `edge` path operation.

`/tikz/only vertical second=`⟨*length*⟩ (style, default `0pt`)

This draws a vertical line from the start point to the target point so that it connects to the target point in the center (or at its border in case it is a node).

The optional ⟨*length*⟩ can be used to shift the line orthogonally to its direction.

`/tikz/only horizontal second=`⟨*length*⟩ (style, default `0pt`)

This draws a horizontal line from the start point to the target point so that it connects to the target point in the center (or at its border in case it is a node).

The optional ⟨*length*⟩ can be used to shift the line orthogonally to its direction.

`/tikz/only vertical first=`⟨*length*⟩ (style, default `0pt`)

This draws a vertical line from the start point to the target point so that it connects to the start point in the center (or at its border in case it is a node).

The optional ⟨*length*⟩ can be used to shift the line orthogonally to its direction.

`/tikz/only horizontal first=`⟨*length*⟩ (style, default `0pt`)

This draws a horizontal line from the start point to the target point so that it connects to the start point in the center (or at its border in case it is a node).

The optional ⟨*length*⟩ can be used to shift the line orthogonally to its direction.

Since all previous key are rather cumbersome, one can install shortcuts for these.

`/tikz/ortho/install shortcuts` (style, no value)

Installs the following shortcuts:

```
|-   → vertical horizontal
-|   → horizontal vertical
-|-  → horizontal vertical horizontal
|-|  → vertical horizontal vertical
|*   → only vertical first
*|   → only vertical second
-*   → only horizontal first
*-   → only horizontal second
```

10 Extending the Path Timers

Ti*k*Z Library `ext.paths.timer`

```
\usetikzlibrary{ext.paths.timer} % LaTeX and plain TeX
\usetikzlibrary[ext.paths.timer] % ConTeXt
```

This library adds timers to the path specifications `rectangle`, `parabola`, `sin` and `cos`.

Q & A: [5, 4] & [30, 39]

In Ti*k*Z, the path specification `rectangle`, `parabola`, `sin` and `cos` do not provide their own timer, i. e. a node placing algorithm that is dependent on the actual path. For `rectangle` the timer of the straight line between the rectangle's corners is used, for the other paths, nodes, coordinates, pics, etc. are placed on the last coordinate.

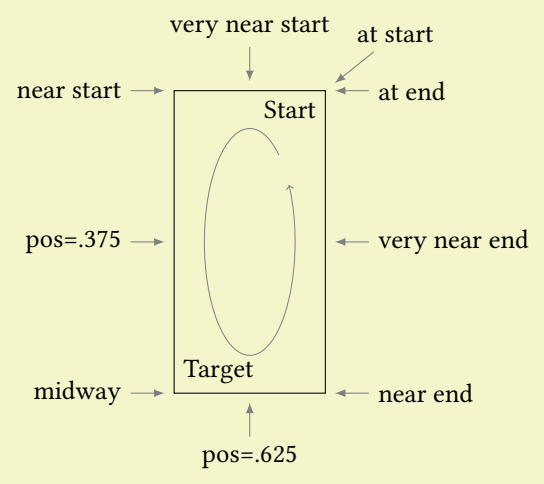
This library allows this.

10.1 Rectangle

For the `rectangle` path operator, the timer starts with `pos = 0` (= `at start`) from the starting coordinate in a counter-clockwise direction along the rectangle. The corners will be at positions 0.0, 0.25, 0.5, 0.75 and 1.0.

`/tikz/rectangle timer=line` or `rectangle` (default `rectangle`)

By default, the library activates the new (correct) timer for `rectangle`. With `rectangle timer = line` the original line timer can be reinstated.

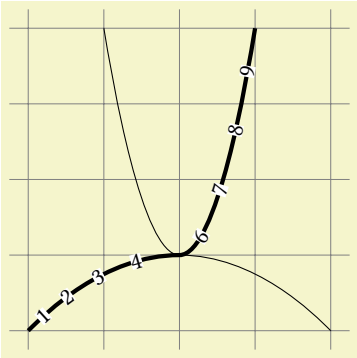



```
\usetikzlibrary {ext.paths.timer}
\begin{tikzpicture}[scale=2, every pin edge/.style={latex-, gray}]
\coordinate [label=above right:Target] (A) at (0,0);
\coordinate [label=below left:Start]   (B) at (1,2);
\draw[->, help lines] ([shift=(50:.3 and .75)] .5,1)
  arc[start angle=50, delta angle=340, x radius=.3, y radius=.75];
\draw (B) rectangle (A)
  foreach \pos/\ang in {at start/60, very near start/90, near start/180, pos=.375/180,
                        midway/180, pos=.625/270, near end/0, very near end/0, at end/0}{
    node[pin=\ang:\pos, style/.expanded=\pos]{}};
\end{tikzpicture}
```

10.2 Parabola

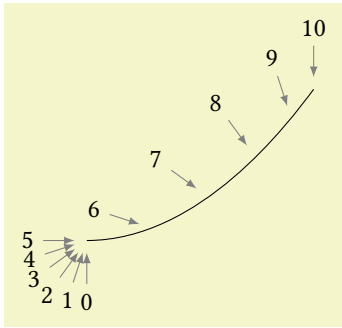
For the `parabola` path operator the timer is similar to the `.. controls ..` operator.

The position 0.5 will lie at the `bend`.




```
\usetikzlibrary {ext.paths.timer}
\begin{tikzpicture}
\draw[help lines]  (-2.25, -1.25) grid (2.25, 3.25);
\draw              ( 2,-1) parabola bend (0,0) (-1,3);
\draw[ultra thick] (-2,-1) parabola bend (0,0) ( 1,3)
  foreach \pos in {1,...,4,6,7,...,9}{
    node[
      pos=.\pos, sloped, fill=white, font=\small, inner sep=+0pt
    ] {\pos}
  };
\end{tikzpicture}
```

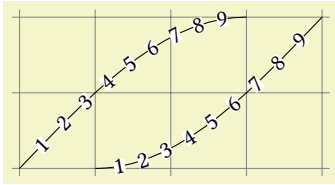
If no `bend` is specified half the positions will collapse into one end of the curve.




```
\usetikzlibrary {ext.paths.timer}
\begin{tikzpicture}[every pin edge/.style={latex-, shorten <=1pt, gray}]
\draw (-2,-2) parabola (1,0)
  foreach \pos in {0, 1, ..., 10} {
    node [pos=\pos/10, pin={[anchor=-18*\pos+90]-18*\pos+270:\pos}]{}
  };
\end{tikzpicture}
```

10.3 Sine/Cosine

The `sin` and `cos` path operators also allow placing of nodes along their paths.



```
\usetikzlibrary {ext.paths.timer}
\begin{tikzpicture}[mark nodes on line/.style={insert path={
  foreach \pos in {1, ..., 9} {node[
    sloped, fill=white, font=\small, inner sep=+0pt, pos=\pos/10] {\pos}}}}]
\draw[help lines] (-2.1,-2.1) grid (2.1,0.1);
\draw             (-2,-2) sin (1,0) [mark nodes on line];
\draw[shift=(0:1)](-2,-2) cos (1,0) [mark nodes on line];
\end{tikzpicture}
```

11 Using Images as a Pattern

TikZ Library ext.patterns.images

```
\usetikzlibrary{ext.patterns.images} % LaTeX and plain TeX
\usetikzlibrary[ext.patterns.images] % ConTeXt
```

This library allows to use an image to be used as a repeating pattern for a path.

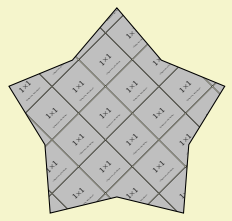
Q & A: [12] & [38]

With this library arbitrary images (or indeed PDF documents) can be used as a repeating pattern for the background of a path.
This is a two-step process:

1. Declaring an image as an “image-pattern”.
2. Using the “image-pattern”.

`\pgfsetupimageaspattern[`⟨*options*⟩`]{`⟨*name*⟩`}{`⟨*image*⟩`}`

`/tikz/image as pattern=`⟨*options*⟩ (default {})



```
\usetikzlibrary {ext.patterns.images,shapes.geometric}
\pgfsetupimageaspattern[width=.5cm]{grid}{example-image-1x1}
\tikz \node[star, minimum size=3cm, draw,
  image as pattern={name=grid,options={left, bottom, y=-.5cm, rotate=45}}] {};
```

`/tikz/image as pattern/name=`⟨*name*⟩ (no default)

Specifies the name of the “image-pattern” to be used.

`/tikz/image as pattern/option` (style, no value)

Options that will be used by the internal `\pgftext`, only keys from `/pgf/text` should be used.

`/tikz/image as pattern/options=`⟨*style*⟩ (style, no default)

Appends style `/tikz/image as pattern/option`.

12 Positioning Plus

Ti*k*Z Library ext.positioning-plus

```
\usetikzlibrary{ext.positioning-plus} % LaTeX and plain TeX
\usetikzlibrary[ext.positioning-plus] % ConTeXt
```

With the help of the `positioning` and the `fit` library this extends the placement of nodes.

12.1 Useful corner anchors

The anchors `corner north east`, `corner north west`, `corner south west` and `corner south east` are defined as "generic anchors", i. e. they are defined for all shapes. This is mostly useful for the placement of circular shapes.

`/tikz/corner above left=`⟨*specification*⟩ (style, default `0pt`)

Similar as `/tikz/above left` of the Ti*k*Z library `positioning` but uses the `corner north west` anchor.

`/tikz/corner below left=`⟨*specification*⟩ (style, default `0pt`)

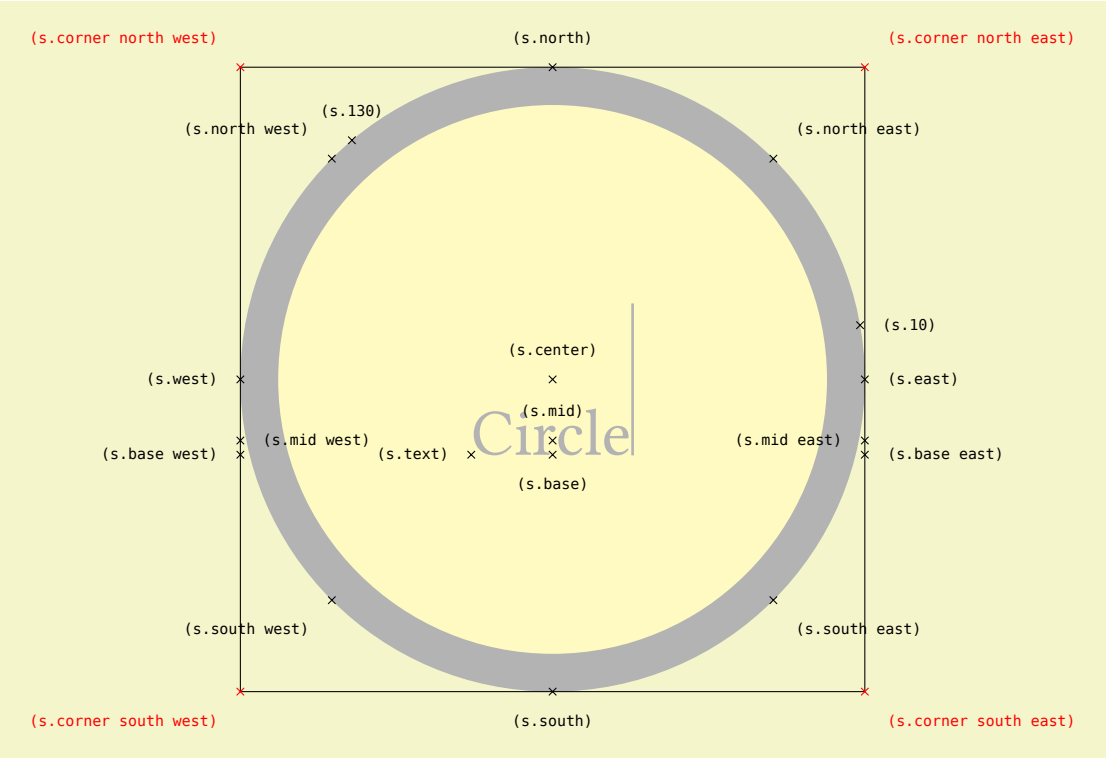
Similar as `/tikz/below left` of the Ti*k*Z library `positioning` but uses the `corner south west` anchor.

`/tikz/corner above right=`⟨*specification*⟩ (style, default `0pt`)

Similar as `/tikz/above right` of the Ti*k*Z library `positioning` but uses the `corner north east` anchor.

`/tikz/corner below right=`⟨*specification*⟩ (style, default `0pt`)

Similar as `/tikz/below right` of the Ti*k*Z library `positioning` but uses the `corner south east` anchor.



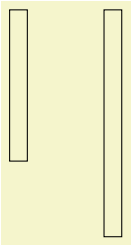

```
\usetikzlibrary {ext.positioning-plus}
\Huge
\begin{tikzpicture}
\node[name=s,shape=circle,shape example]
  {Circle\vrule width 1pt height 2cm};
\foreach \anchor/\placement in {
  north west/above left, north/above, north east/above right,
  west/left, center/above, east/right,
  mid west/right, mid/above, mid east/left,
  base west/left, base/below, base east/right,
  south west/below left, south/below, south east/below right,
  text/left, 10/right, 130/above}
  \draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
    node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\draw (s.corner north west) rectangle (s.corner south east);
\foreach \anchor/\placement in {
  corner north west/above left, corner north east/above right,
  corner south west/below left, corner south east/below right}
  \draw[red,shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
    node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```

12.2 Useful placement keys for vertical and horizontal alignment

`/tikz/north left=`⟨*specification*⟩ (style, default `0pt`)

Like `/tikz/left` but aligns the nodes at their `north` border.

This is basically the same as `left=of reference.north west, anchor=north east`.



```
\usetikzlibrary {ext.positioning-plus}
\begin{tikzpicture}[nodes=draw]
\node[minimum height=2cm] (a) {};
\node[minimum height=3cm, north right=of a] {};
\end{tikzpicture}
```

`/tikz/north right=`⟨*specification*⟩ (style, default `0pt`)

Like `/tikz/right` but aligns the nodes at their `north` border.

This is basically the same as `left=of reference.north east, anchor=north west`.

`/tikz/south left=`⟨*specification*⟩ (style, default `0pt`)

Like `/tikz/left` but aligns the nodes at their `south` border.

This is basically the same as `left=of reference.south west, anchor=south east`.

`/tikz/south right=`⟨*specification*⟩ (style, default `0pt`)

Like `/tikz/right` but aligns the nodes at their `south` border.

This is basically the same as `left=of reference.south east, anchor=south west`.

`/tikz/west above=`⟨*specification*⟩ (style, default `0pt`)

Like `/tikz/above` but aligns the nodes at their `west` border.

This is basically the same as `left=of reference.north west, anchor=south west`.

`/tikz/west below=`⟨*specification*⟩ (style, default `0pt`)

Like `/tikz/below` but aligns the nodes at their `west` border.

This is basically the same as `left=of reference.south west, anchor=north west`.

`/tikz/east above=`⟨*specification*⟩ (style, default `0pt`)

Like `/tikz/above` but aligns the nodes at their `east` border.

This is basically the same as `left=of reference.north east, anchor=south east`.

`/tikz/east below=`⟨*specification*⟩ (style, default `0pt`)

Like `/tikz/below` but aligns the nodes at their `east` border.

This is basically the same as `left=of reference.south east, anchor=north east`.

The same exist for the recently introduces corner anchors, too.

`/tikz/corner north left=`⟨*specification*⟩ (style, default `0pt`)

The same as `/tikz/north left` but uses the new corner anchors.

`/tikz/corner north right=`⟨*specification*⟩ (style, default `0pt`)

The same as `/tikz/north right` but uses the new corner anchors.

`/tikz/corner south left=`⟨*specification*⟩ (style, default `0pt`)

The same as `/tikz/south left` but uses the new corner anchors.

`/tikz/corner south right=`⟨*specification*⟩ (style, default `0pt`)

The same as `/tikz/south right` but uses the new corner anchors.

`/tikz/corner west above=`⟨*specification*⟩ (style, default `0pt`)

The same as `/tikz/west above` but uses the new corner anchors.

`/tikz/corner west below=`⟨*specification*⟩ (style, default `0pt`)

The same as `/tikz/west below` but uses the new corner anchors.

`/tikz/corner east above=`⟨*specification*⟩ (style, default `0pt`)

The same as `/tikz/east above` but uses the new corner anchors.

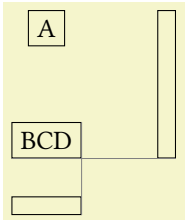
`/tikz/corner east below=`⟨*specification*⟩ (style, default `0pt`)

The same as `/tikz/east below` but uses the new corner anchors.

While the ⟨*specification*⟩ of all these keys still accept the same form as with Ti*k*Z, the `ext.positioning-plus` library extends this even more.

The specification after `of` can contain a list of coordinates (like the `fit` key of the `fit` library). This means that the new node will be placed in relation to a rectangular bounding box that fits around all this nodes in the list.

If this list is prefixed with `|`, `-` or `+`, the new node will also have the same height (`|`), the same width (`-`) or both as this bounding box.




```
\usetikzlibrary {ext.positioning-plus}
\begin{tikzpicture}[nodes=draw]
\node                       (A) {A};
\node[below=of A]           (BCD) {BCD};
\node[right=of |(A)(BCD)] (c) {};
\node[below=.5:of -(A)(BCD)] (d) {};
\draw[help lines] (BCD.south west) -- (c.south east)
                  (BCD.north east) -- (d.south east);
\end{tikzpicture}
```

This functionality is also available without the placement:

`/tikz/fit bounding box`=⟨*list of coordinates*⟩ (style, no default)

Creates a rectangular node with the name `fit bounding box` that encompasses the ⟨*list of coordinates*⟩.

`/tikz/span vertical`=⟨*list of coordinates*⟩ (style, no default)

Creates a rectangular node with the name `fit bounding box` that encompasses the ⟨*list of coordinates*⟩ *and* sets the `/pgfminimum height` to the height of this bounding box.

`/tikz/span horizontal`=⟨*list of coordinates*⟩ (style, no default)

Creates a rectangular node with the name `fit bounding box` that encompasses the ⟨*list of coordinates*⟩ *and* sets the `/pgfminimum width` to the width of this bounding box.

`/tikz/span`=⟨*list of coordinates*⟩ (style, no default)

Is a combination of `/tikz/span vertical` and `/tikz/span horizontal`.

As you maybe noticed in the example above, the ⟨*specification*⟩ also allows a prefix delimited by `:` which the `node distance` will be multiplied to with for the placement.[5]

[5] This is probably more useful when `/tikz/on grid` is used.

13 Scaling Pictures to a Specific Size

Ti*k*Z Library `ext.scalepicture`

```
\usetikzlibrary{ext.scalepicture} % LaTeX and plain TeX
\usetikzlibrary[ext.scalepicture] % ConTeXt
```

This library scales Ti*k*Z pictures to a specific width or height by scaling the whole picture.

If one of the keys below are used on a Ti*k*Z picture, i. e. as an option to `\tikzpicture` or `\begin{tikzpicture}` the size of the picture[6] will be measured and written to the AUX file so that it will be available at the next compilation run and an appropriate scaling for the picture can be installed.

`\tikzextpicturewidth`

Returns the last measured width of the picture.

This will expand to `0pt` if the picture hasn't been measured before.

`\tikzextpictureheight`

Returns the last measured height of the picture.

This will expand to `0pt` if the picture hasn't been measured before.

`/tikz/save picture size` (style, no value)

This key is usually used by the keys provided by this library. Normally, this is not needed to be explicitly given.

13.1 Keeping the aspect ratio

The following *unstarred* keys do not change the aspect ratio of the picture.

`/tikz/picture width`=⟨*dimension*⟩ (no default)

Scales the picture so that the width of the picture will be ⟨*dimension*⟩. This will keep the aspect ratio the same.

`/tikz/minimum picture width`=⟨*dimension*⟩ (no default)

As above but will not change the size of the picture if its width is greater than ⟨*dimension*⟩.

[6] This is the size of the pseudo-node `current bounding box`.

`/tikz/maximum picture width`=⟨*dimension*⟩ (no default)

As above but will not change the size of the picture if its width is less than ⟨*dimension*⟩.

`/tikz/picture height`=⟨*dimension*⟩ (no default)

Scales the picture so that the height of the picture will be ⟨*dimension*⟩. This will keep the aspect ratio the same.

`/tikz/minimum picture height`=⟨*dimension*⟩ (no default)

As above but will not change the size of the picture if its height is greater than ⟨*dimension*⟩.

`/tikz/maximum picture height`=⟨*dimension*⟩ (no default)

As above but will not change the size of the picture if its height is less than ⟨*dimension*⟩.

`/tikz/minimum picture size`={⟨*width*⟩}{⟨*height*⟩} (no default)

Scales the picture so that its height will be at least ⟨*width*⟩ and its height will be at least ⟨*height*⟩.

`/tikz/maximum picture size`={⟨*width*⟩}{⟨*height*⟩} (no default)

Scales the picture so that its height will be at most ⟨*width*⟩ and its height will be at most ⟨*height*⟩.

13.2 Changing the aspect ratio.

The following *starred* keys do change the aspect ratio.

`/tikz/picture width*`=⟨*dimension*⟩ (no default)

Scales the picture so that the width of the picture will be ⟨*dimension*⟩. This will only scale the *x* axis.

`/tikz/minimum picture width*`=⟨*dimension*⟩ (no default)

As above but will not change the size of the picture if its width is greater than ⟨*dimension*⟩.

`/tikz/maximum picture width*`=⟨*dimension*⟩ (no default)

As above but will not change the size of the picture if its width is less than ⟨*dimension*⟩.

`/tikz/picture height*`=⟨*dimension*⟩ (no default)

Scales the picture so that the height of the picture will be ⟨*dimension*⟩. This will only scale the y axis.

`/tikz/picture size*`={⟨*width*⟩}{⟨*height*⟩} (no default)

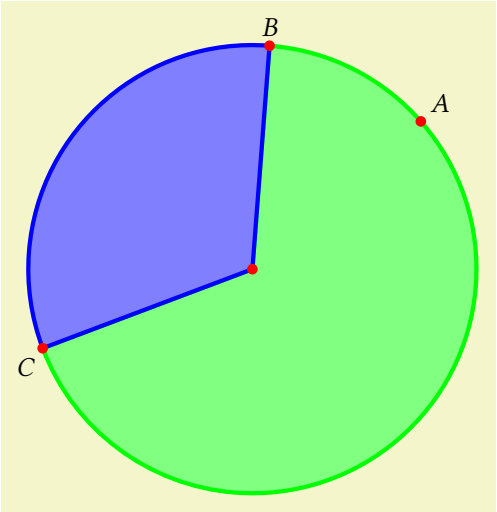
Scales the picture so that its width will be ⟨*width*⟩ and its height will be ⟨*height*⟩. This will scale both axes but independent from each other.

14 Arcs through Three Points

TikZ Library ext.topaths.arcthrough

```
\usetikzlibrary{ext.topaths.arcthrough} % LaTeX and plain TeX
\usetikzlibrary[ext.topaths.arcthrough] % ConTeXt
```

This library allows to use an arc defined by three points.




```
\usetikzlibrary {ext.topaths.arcthrough}
\begin{tikzpicture}
\coordinate[label=above right:$A$] (A) at ( 3, 1);
\coordinate[label=above:$B$]       (B) at ( 1, 2);
\coordinate[label=below left:$C$]  (C) at (-2,-2);

\draw[ultra thick, draw=green, fill=green!50]
  (B) to[arc through={clockwise,(A)}] (C)
  -- (arc through center) -- cycle;
\draw[ultra thick, draw=blue, fill=blue!50]
  (B) to[arc through=(A)]             (C)
  -- (arc through center) -- cycle;

\foreach \p in {A,B,C, arc through center} \fill[red] (\p) circle[radius=2pt];
\end{tikzpicture}
```

This can only by used for circles in the canvas coordinate system.

/tikz/arc through/through=⟨*coordinate*⟩ (no default, initially (0,0))

The coordinate on the circle that defines – together with the starting and target point – a circle.

/tikz/arc through/center suffix=⟨*suffix*⟩ (no default, initially)

The arc through will define a coordinate named arc through center⟨*suffix*⟩ so that it can be referenced later.

/tikz/arc through/clockwise (no value)

The resulting arc will go clockwise from the starting point to the target point. This will not necessarily go through the through point.

/tikz/arc through/counter clockwise (no value)

The resulting arc will go counter clockwise from the starting point to the target point. This will not necessarily go through the through point.

/tikz/arc through=⟨*key-value*⟩ (no default)

This key should be used with to or edge. A parameter other than center suffix, clockwise or counter clockwise will be assumed to be the through coordinate.

15 Mirror, Mirror on the Wall

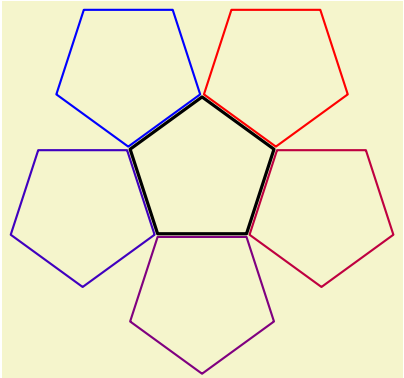
Ti*k*Z Library ext.transformations.mirror

```
\usetikzlibrary{ext.transformations.mirror} % LaTeX and plain TeX
\usetikzlibrary[ext.transformations.mirror] % ConTeXt
```

This library adds more transformations to Ti*k*Z.

As explained in section 16, there are two approaches to setting a mirror transformation. As with the commands in PGF, we'll be using a lowercase `m` for the reflection matrix and an uppercase `M` for the built-in approach.

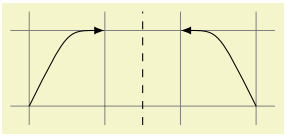
15.1 Using the reflection matrix



```
\usetikzlibrary {shapes.geometric,ext.transformations.mirror}
\begin{tikzpicture}[line join=round, thick, reg poly/.style={
  shape=regular polygon, regular polygon sides={#1}}]
\node[reg poly=5, minimum size=+2cm, draw, very thick] (a) {};
\foreach \i[evaluate={\col=(\i-1)/.04}] in {1,...,5}
  \node [mirror={(a.corner \i)--(a.side \i)}, transform shape,
         reg poly=5, minimum size=+2cm, draw=red!\col!blue] {};
\end{tikzpicture}
```

/tikz/xmirror=⟨*value or coordinate*⟩ (default 0pt)

Sets up a transformation that mirrors along a horizontal line that goes through point (⟨*value*⟩, 0) or ⟨*coordinate*⟩.



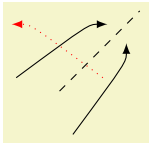
```
\usetikzlibrary {ext.transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-0.25, -.25) grid (3.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);
\draw[xmirror={(m)},-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`/tikz/ymirror=`⟨*value or coordinate*⟩ (default `0pt`)

Sets up a transformation that mirrors along a vertical line that goes through point $(0, \langle value \rangle)$ or ⟨*coordinate*⟩.

`/tikz/mirror x=`⟨*coordinate*⟩ (default `(0,0)`)

Similar to `/tikz/xmirror`, this however uses the `xyz` coordinate system instead of the `canvas` system.



```
\usetikzlibrary {ext.transformations.mirror}
\begin{tikzpicture}[x=.5cm, y=(45:1cm)]

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);

\draw[ xmirror=(m), -latex, red, dotted] (0,0) .. controls (.5,1) .. (1,1);
\draw[mirror x=(m), -latex]              (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`/tikz/mirror y=`⟨*coordinate*⟩ (default `(0,0)`)

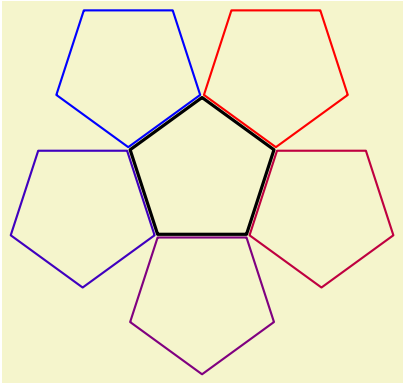
Similar to `/tikz/ymirror`, this however uses the `xyz` coordinate system instead of the `canvas` system.

`/tikz/mirror=`⟨*point A*⟩`--`⟨*point B*⟩ (no default)

Sets up a transformation that mirrors along a line that goes through ⟨*point A*⟩ and ⟨*point B*⟩.

When only ⟨*point A*⟩ is given that line goes through ⟨*point A*⟩ and the origin.

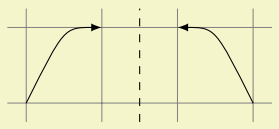
15.2 Using built-in transformations



```
\usetikzlibrary {shapes.geometric,ext.transformations.mirror}
\begin{tikzpicture}[line join=round, thick, reg poly/.style={
  shape=regular polygon, regular polygon sides={#1}}]
\node[reg poly=5, minimum size=+2cm, draw, very thick] (a) {};
\foreach \i[evaluate={\col=(\i-1)/.04}] in {1,...,5}
  \node [Mirror=(a.corner \i)--(a.side \i), transform shape,
         reg poly=5, minimum size=+2cm, draw=red!\col!blue] {};
\end{tikzpicture}
```

/tikz/xMirror=⟨*value or coordinate*⟩ (default 0pt)

Sets up a transformation that mirrors along a horizontal line that goes through point (⟨*value*⟩, 0) or ⟨*coordinate*⟩.



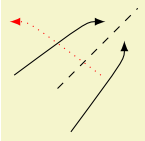
```
\usetikzlibrary {ext.transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-0.25, -.25) grid (3.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);
\draw[xMirror=(m),-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

/tikz/yMirror=⟨*value or coordinate*⟩ (default 0pt)

Sets up a transformation that mirrors along a vertical line that goes through point (0, ⟨*value*⟩) or ⟨*coordinate*⟩.

/tikz/Mirror x=⟨*coordinate*⟩ (default (0,0))

Similar to /tikz/xMirror, this however uses the xyz coordinate system instead of the canvas system.



```
\usetikzlibrary {ext.transformations.mirror}
\begin{tikzpicture}[x=.5cm, y=(45:1cm)]

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);

\draw[ xMirror=(m), -latex, red, dotted] (0,0) .. controls (.5,1) .. (1,1);
\draw[Mirror x=(m), -latex]              (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

/tikz/Mirror y=⟨*coordinate*⟩ (default (0,0))

Similar to /tikz/yMirror, this however uses the xyz coordinate system instead of the canvas system.

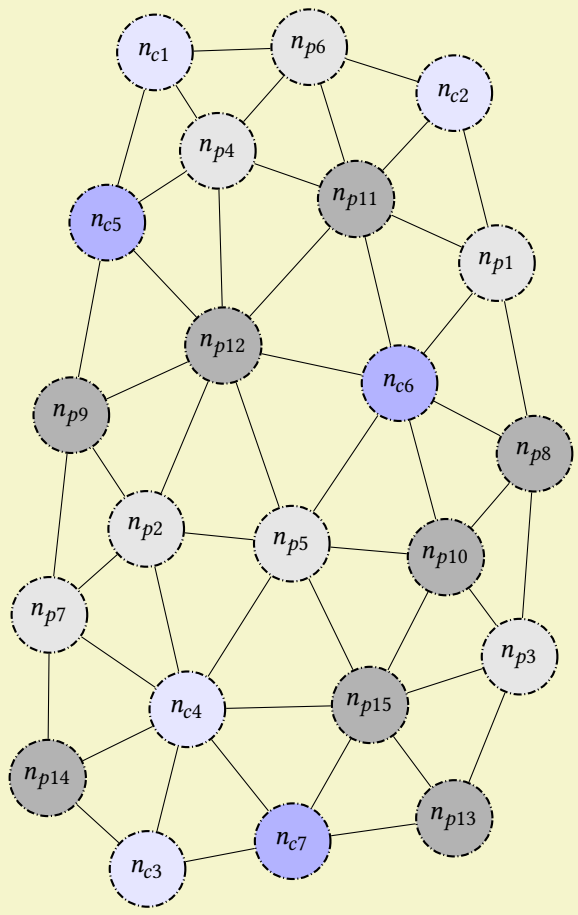
/tikz/Mirror=⟨*point A*⟩--⟨*point B*⟩ (no default)

Sets up a transformation that mirrors along a line that goes through ⟨*point A*⟩ and ⟨*point B*⟩.

When only ⟨*point A*⟩ is given that line goes through ⟨*point A*⟩ and the origin.

Part III
PGF Libraries

These libraries (should) work with both PGF and Ti*k*Z.




```
\usetikzlibrary {graphs,graphdrawing,ext.misc} \usegdlibrary {force}
\tikzset{
  mynode/.style={
    circle, minimum size=10mm, draw, densely dashdotted, thick,
    decide color/.expand once=#1},
  decide color/.style 2 args={
    /utils/TeX/if=c#1
      {/utils/TeX/ifnum={#2<5}{bluelight}{bluedark}}
      {/utils/TeX/ifnum={#2<8}{light}{dark}}},
  light/.style={fill=gray!20},  bluelight/.style={fill=blue!10},
  dark/.style ={fill=gray!60},  bluedark/.style ={fill=blue!30}}
\tikz\graph[
  spring electrical layout, vertical=c2 to p13,
  node distance=1.5cm, typeset=$n_{\tikzgraphnodetext}$,
  nodes={mynode=\tikzgraphnodetext}] {
  % outer ring
  c2 -- {p1, p11, p6};
    p1 -- {p8, c6, p11};
      p8 -- {p3, p10, c6};
       p3 -- {p13, p15, p10};
         p13 -- {p15, c7};
           c7  -- {c3, c4, p15};
           c3  -- {p14, c4};
           p14 -- {p7, c4};
         p7 -- {p9, p2, c4};
       p9 -- {c5, p12, p2};
     c5 -- {c1, p4, p12};
   c1 -- {p6, p4};
  p6 -- {p11, p4};
  % inner ring
  p11 -- {c6, p12, p4};
  p5 -- {c6 -- {p10, p12}, p10 -- p15, p15 -- c4, c4 -- p2, p2 -- p12, p12 -- p4};
};
```

16 Transformations: Mirroring

PGF Library ext.transformations.mirror

```
\usepgflibrary{ext.transformations.mirror}  % LaTeX and plain TeX
\usepgflibrary[ext.transformations.mirror]  % ConTeXt
```

This library adds mirror transformations to PGF.

Two approaches to mirror transformation exist:

1. Using the reflection matrix (see left column).

   This depends on `\pgfpointnormalised` which involves the sine and the cosine functions of PGFmath.

2. Using built-in transformations (see right column).

   This depends on `\pgfmathanglebetweenpoints` which involves the arctangent (`atan2`) function of PGFmath.

Which one is better? I don't know. Choose one you're comfortable with.

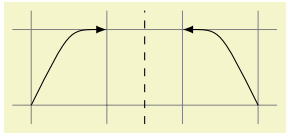
16.1 Using the reflection matrix

The following commands use the reflection matrix that sets the transformation matrix following

$$A = \frac{1}{\|\vec{l}\|^2}\begin{bmatrix} l_x^2 - l_y^2 & 2l_xl_y \\ 2l_xl_y & l_y^2 - l_x^2 \end{bmatrix}.$$

\pgftransformxmirror{⟨*value*⟩}

Sets up a transformation that mirrors along a vertical line that goes through point (⟨*value*⟩, 0).



```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-0.25, -
.25) grid (3.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (1.5, -.25) -- (1.5, 1.25);
\pgftransformxmirror{1.5}

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

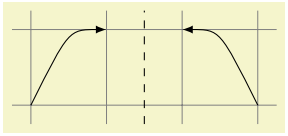
16.2 Using built-in transformations

The following commands use a combination of shifting, rotating, −1 scaling, rotating back and shifting back to reach the mirror transformation.

The commands are named the same as on the left side, only the `m` in `mirror` is capitalized.

\pgftransformxMirror{⟨*value*⟩}

Sets up a transformation that mirrors along a vertical line that goes through point (⟨*value*⟩, 0).



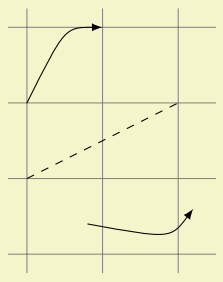
```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-0.25, -
.25) grid (3.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (1.5, -.25) -- (1.5, 1.25);
\pgftransformxMirror{1.5}

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`\pgftransformymirror{⟨value⟩}`

Sets up a transformation that mirrors along a horizontal line that goes through point $(0, \langle value \rangle)$.

`\pgftransformmmirror{⟨point A⟩}{⟨point B⟩}`

Sets up a transformation that mirrors along the line that goes through ⟨*point A*⟩ and ⟨*point B*⟩.



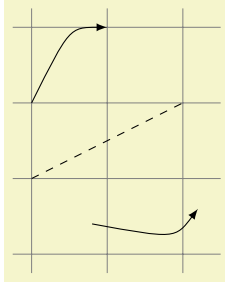
```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-.25, -2.25) grid (2.5, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (0, -1) -- (2, 0);
\pgftransformmmirror{\pgfpointxy{0}{-1}}
                    {\pgfpointxy{2}{ 0}}

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`\pgftransformmyMirror{⟨value⟩}`

Sets up a transformation that mirrors along a horizontal line that goes through point $(0, \langle value \rangle)$.

`\pgftransformMirror{⟨point A⟩}{⟨point B⟩}`

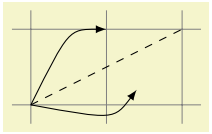
Sets up a transformation that mirrors along the line that goes through ⟨*point A*⟩ and ⟨*point B*⟩.



```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-.25, -2.25) grid (2.5, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (0, -1) -- (2, 0);
\pgftransformMirror{\pgfpointxy{0}{-1}}
                   {\pgfpointxy{2}{ 0}}

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`\pgfqtransformmmirror{⟨point A⟩}`

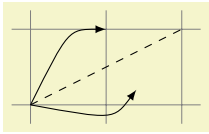
Sets up a transformation that mirrors along the line that goes through the origin and ⟨*point A*⟩.



```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-.25, -.25) grid (2.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (0, 0) -- (2, 1);
\pgfqtransformmmirror{\pgfpointxy{2}{1}}

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`\pgfqtransformMirror{⟨point A⟩}`

Sets up a transformation that mirrors along the line that goes through the origin and ⟨*point A*⟩.



```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-.25, -.25) grid (2.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (0, 0) -- (2, 1);
\pgfqtransformMirror{\pgfpointxy{2}{1}}

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

17 Shape: Circle Arrow

TikZ Library ext.shapes.circlearrow

```
\usepgflibrary{ext.shapes.circlearrow}  % LaTeX and plain TeX and pure pgf
\usepgflibrary[ext.shapes.circlearrow]  % ConTeXt and pure pgf
\usetikzlibrary{ext.shapes.circlearrow} % LaTeX and plain TeX when using TikZ
\usetikzlibrary[ext.shapes.circlearrow] % ConTeXt when using TikZ
```

A circular shape named `circle arrow` that has an arc as its background path that can have an arrow tip.

Q & A: [19] & [34]

Shape circle arrow

This shape is an arrow whose path is an arc – defined very similar to the `arc` path operation – that can possibly be customized with arrow tips.

/pgf/circle arrow start angle=⟨*start angle*⟩ (no default, initially {})

Sets the start angle.

/pgf/circle arrow end angle=⟨*end angle*⟩ (no default, initially {})

Sets the end angle.

/pgf/circle arrow delta angle=⟨*delta angle*⟩ (no default, initially {})

Sets the delta angle.

/pgf/circle arrow arrows=⟨*start arrow tip specification*⟩-⟨*end arrow tip specification*⟩ (no default, initially -)

The specification will be forwarded to `\pgfsetarrows` .

A few handful styles are pre-defined.

/pgf/circle arrow turn left north (no value)

Sets `circle arrow start angle = 100`, `circle arrow delta angle = 340` and `circle arrow arrows = ->`.

/pgf/circle arrow turn left east (no value)

As above but `circle arrow start angle = 10`.

/pgf/circle arrow turn left west (no value)

As above but `circle arrow start angle = 280`.

/pgf/circle arrow turn left south (no value)

As above but `circle arrow start angle = 190`.

/pgf/circle arrow turn right north (no value)

Sets `circle arrow start angle = 100`, `circle arrow delta angle = 340` and `circle arrow arrows = <-`.

/pgf/circle arrow turn right east (no value)

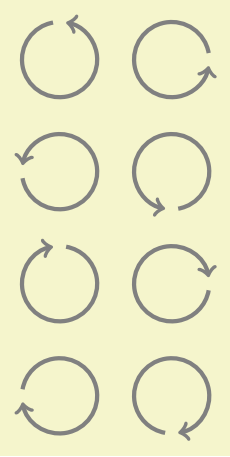
As above but `circle arrow start angle = 10`.

/pgf/circle arrow turn right west (no value)

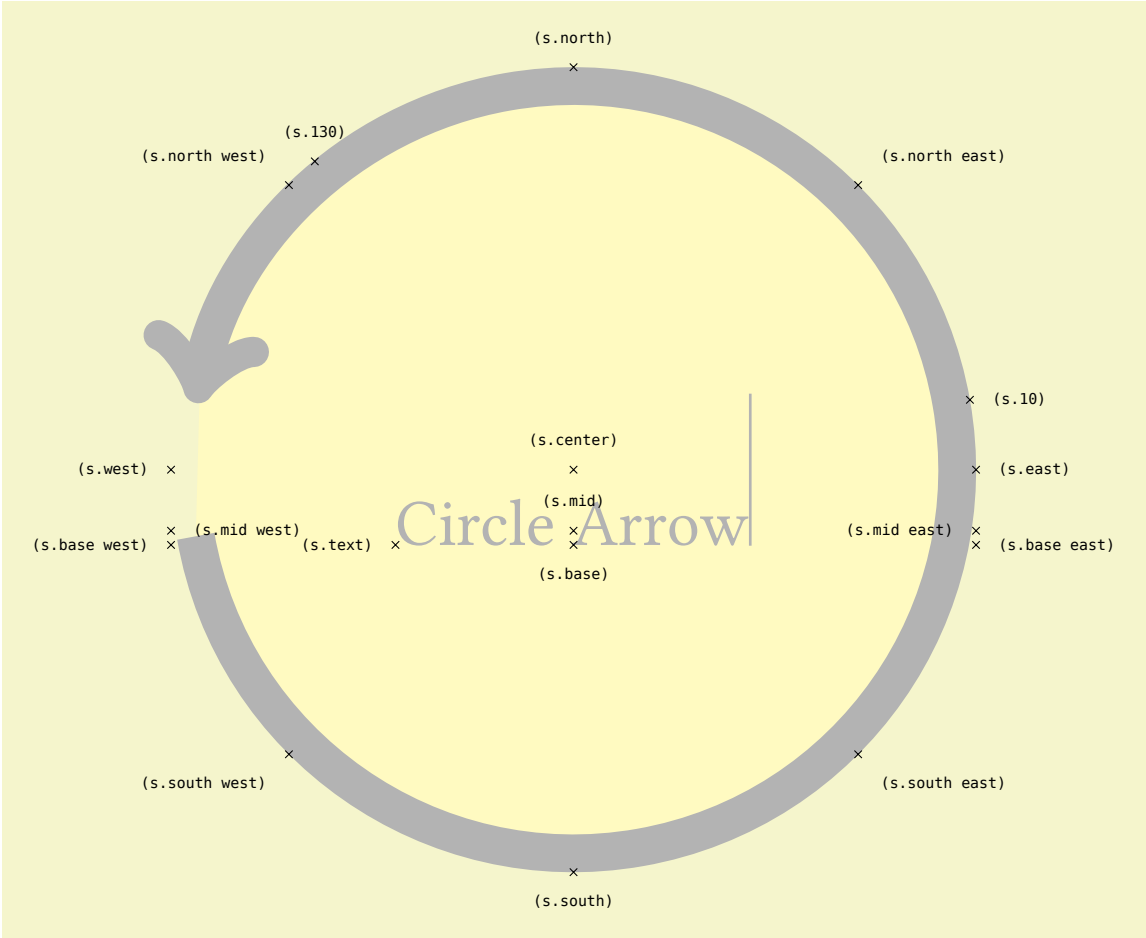
As above but `circle arrow start angle = 280`.

/pgf/circle arrow turn right south (no value)

As above but `circle arrow start angle = 190`.



```
\usetikzlibrary {ext.shapes.circlearrow,matrix}
\begin{tikzpicture}
\matrix[matrix of nodes, draw=none, row sep=1em, column sep=1em,
  every node/.style={draw=gray, shape=circle arrow, ultra thick, inner sep=1em}
] (m) {
  |[circle arrow turn left north]|  & |[circle arrow turn left east]|   \\
  |[circle arrow turn left west]|   & |[circle arrow turn left south]|  \\
  |[circle arrow turn right north]| & |[circle arrow turn right east]|  \\
  |[circle arrow turn right west]|  & |[circle arrow turn right south]| \\
};
\end{tikzpicture}
```




```
\usetikzlibrary {ext.shapes.circlearrow}
\begin{tikzpicture}\Huge
\node[name=s, shape=circle arrow,
  circle arrow turn left west, shape example]
  {Circle Arrow\vrule width 1pt height 2cm};
\foreach \anchor/\placement in
  {north west/above left, north/above, north east/above right,
   west/left, center/above, east/right,
   mid west/right, mid/above, mid east/left,
   base west/left, base/below, base east/right,
   south west/below left, south/below, south east/below right,
   text/left, 10/right, 130/above}
   \draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
     node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```

18 Shape: Circle Cross Split

TikZ Library ext.shapes.circlecrosssplit

```
\usepgflibrary{ext.shapes.circlecrosssplit}  % LaTeX and plain TeX and pure pgf
\usepgflibrary[ext.shapes.circlecrosssplit]  % ConTeXt and pure pgf
\usetikzlibrary{ext.shapes.circlecrosssplit} % LaTeX and plain TeX when using TikZ
\usetikzlibrary[ext.shapes.circlecrosssplit] % ConTeXt when using TikZ
```

A circular shape with four parts that can be individually filled.

Q & A: [14] & [35]

Shape circle cross split

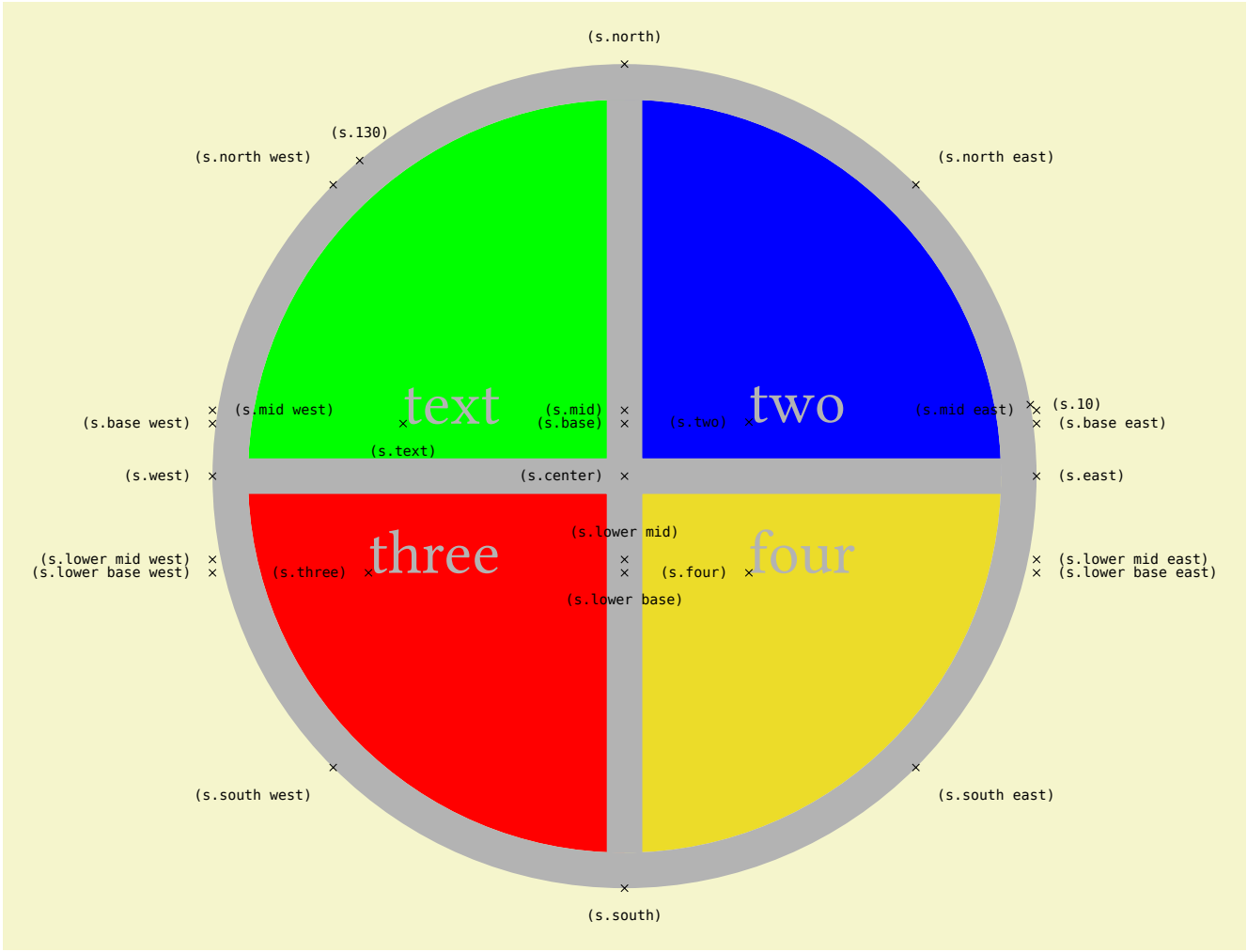
This shape has four node parts that are placed near the center of a circle.

/pgf/circle cross split part fill={⟨*list*⟩} (no default, initially none)

Sets the custom fill color for each node part shape. The items in ⟨*list*⟩ should be separated by commas (so if there is more than one item in ⟨*list*⟩, it must be surrounded by braces). If ⟨*list*⟩ has less entries than node parts, then the remaining node parts use the color from the last entry in the list. This key will automatically set /pgf/circle cross split uses custom fill.

/pgf/circle cross split uses custom fill=⟨*boolean*⟩ (default true)

This enables the use of a custom fill for each of the node parts (including the area covered by the inner sep). The background path for the shape should not be filled (e. g., in TikZ, the fill option for the node must be implicitly or explicitly set to none). Internally, this key sets the TeX-if \ifpgfcirclecrosssplitcustomfill appropriately.


(s.north)
(s.130)
(s.north west)
(s.north east)
text
two
(s.mid west)
(s.mid)
(s.base)
(s.two)
(s.mid east)
(s.10)
(s.base west)
(s.base east)
(s.text)
(s.west)
(s.center)
(s.east)
(s.lower mid)
three
four
(s.lower mid west)
(s.lower base west)
(s.three)
(s.four)
(s.lower mid east)
(s.lower base east)
(s.lower base)
(s.south west)
(s.south east)
(s.south)

```
\usepgflibrary {ext.shapes.circlecrosssplit}
\begin{tikzpicture}\Huge
\node[name=s, shape=circle cross split, shape example, inner xsep=1.5cm, fill=none,
  circle cross split part fill={green,blue,red,yellow!90!black}]
 {\nodepart{text}text\nodepart{two}two
        \nodepart{three}three\nodepart{four}four};
\foreach \anchor/\placement in
    {north west/above left, north/above,      north east/above right,
          west/left,      center/left,             east/right,
       mid west/right,        mid/left,         mid east/left,
      base west/left,        base/left,        base east/right,
lower base west/left,  lower base/below, lower base east/right,
 lower mid west/left,  lower mid/above,   lower mid east/right,
     south west/below left, south/below,      south east/below right,
   text/below, 10/right, 130/above, two/left, three/left, four/left}
   \draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
     node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```

19 Shape: Heatmark

TikZ Library ext.shapes.heatmark

```
\usepgflibrary{ext.shapes.heatmark}  % LaTeX and plain TeX and pure pgf
\usepgflibrary[ext.shapes.heatmark]  % ConTeXt and pure pgf
\usetikzlibrary{ext.shapes.heatmark} % LaTeX and plain TeX when using TikZ
\usetikzlibrary[ext.shapes.heatmark] % ConTeXt when using TikZ
```

A circular shape that has customizable rings around it.

Q & A: [2] & [27]

Shape heatmark

/pgf/heatmark arcs=⟨*arcs num*⟩ (no default, initially 3)

Sets the number of arc around the circle to ⟨*arcs num*⟩.

/pgf/heatmark arc width=⟨*arc width*⟩ (no default, initially 4pt)

Sets the width of the rings around the circle to ⟨*arc width*⟩.

/pgf/heatmark arc sep=⟨*sep length*⟩ (no default, initially 1pt)

Sets the whitespace between the rings to ⟨*sep length*⟩.

/pgf/heatmark arc rings=⟨*rings num*⟩ (no default, initially 3)

Sets the number of rings around the circle to ⟨*rings num*⟩

/pgf/heatmark arc sep angle=⟨*sep angle*⟩ (no default, initially 20)

Sets the whitespace angle between the arcs in one ring to ⟨*sep angle*⟩.

/pgf/heatmark inner opacity=⟨*inner opacity*⟩ (no default, initially 0.8)

Sets the opacity of the inner ring to ⟨*inner opacity*⟩.

/pgf/heatmark outer opacity=⟨*low opacity*⟩ (no default, initially 0.2)

Sets the opacity of the outer ring to ⟨*outer opacity*⟩.

The opacity of the rings between the outer and the inner ring will be interpolated by these two opacities.

This shape takes the value of /pgf/shape border rotate into consideration.

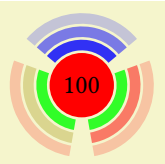
For every ring and for every arc the following styke keys are tried.

/pgf/heatmark ring ⟨*ring number*⟩ (style, no value)

`/pgf/heatmark arc ⟨arc number⟩` (style, no value)

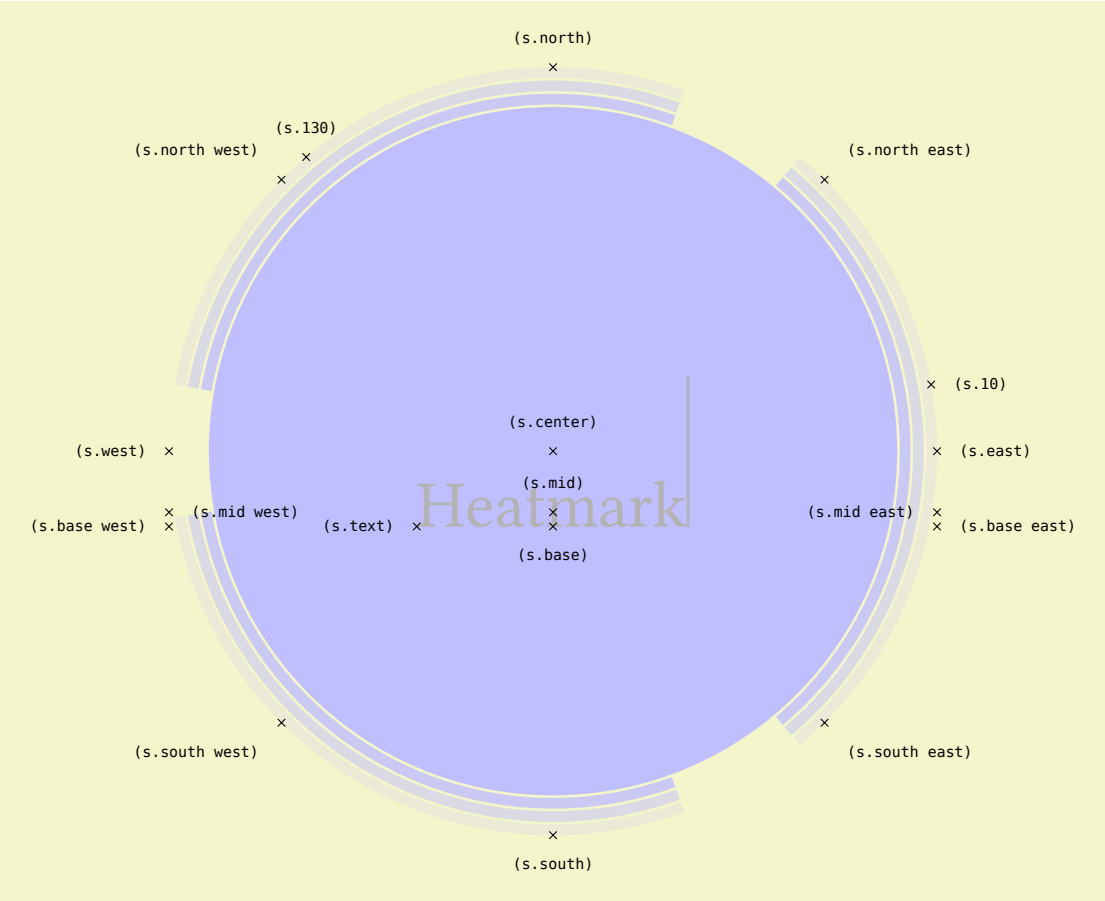
`/pgf/heatmark ring ⟨ring number⟩ arc ⟨arc number⟩` (style, no value)

The PGFshape is setup in a way that even Ti*k*Z styles can be used with a little bit work:




```
\usetikzlibrary {ext.shapes.heatmark}
\tikz[
  shape border rotate=90,
  /pgf/heatmark ring 1/.append style={/tikz/fill=green},
  /pgf/heatmark arc 1/.append style={/tikz/fill=blue},
  /pgf/heatmark ring 2 arc 2/.append style={/tikz/fill=yellow!70!black}
] \node[heatmark, fill=red] (n) {100};
```

It is best to use this shape with no actual border (`draw = none`) and the `outer sep` set to zero.




```
\usetikzlibrary {ext.shapes.heatmark}
\begin{tikzpicture}\Huge
\node[name=s, shape=heatmark, shape example,
  fill=blue!25, draw=none, outer sep=0pt]
  {Heatmark\vrule width 1pt height 2cm};
\foreach \anchor/\placement in
  {north west/above left, north/above,
                          north east/above right,
        west/left, center/above,      east/right,
     mid west/right,   mid/above,  mid east/left,
    base west/left,   base/below, base east/right,
   south west/below left, south/below,
                          south east/below right,
   text/left, 10/right, 130/above}
   \draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
     node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```

20 Shape: Rectangle with Rounded Corners

TikZ Library ext.shapes.rectangleroundedcorners

```
\usepgflibrary{ext.shapes.rectangleroundedcorners}  % LaTeX and plain TeX and pure pgf
\usepgflibrary[ext.shapes.rectangleroundedcorners]  % ConTeXt and pure pgf
\usetikzlibrary{ext.shapes.rectangleroundedcorners} % LaTeX and plain TeX when using TikZ
\usetikzlibrary[ext.shapes.rectangleroundedcorners] % ConTeXt when using TikZ
```

A rectangle with rounded corners.

Shape rectangle with rounded corners

This library provides a `rectangle with rounded corners` where every corner can have a different radius.

/pgf/rectangle with rounded corners north west radius=⟨*dimen*⟩ (no default, initially `.5\pgflinewidth`)

Sets the north west radius to ⟨*dimen*⟩.

/pgf/rectangle with rounded corners north east radius=⟨*dimen*⟩ (no default, initially `.5\pgflinewidth`)

Sets the north east radius to ⟨*dimen*⟩.

/pgf/rectangle with rounded corners south west radius=⟨*dimen*⟩ (no default, initially `.5\pgflinewidth`)

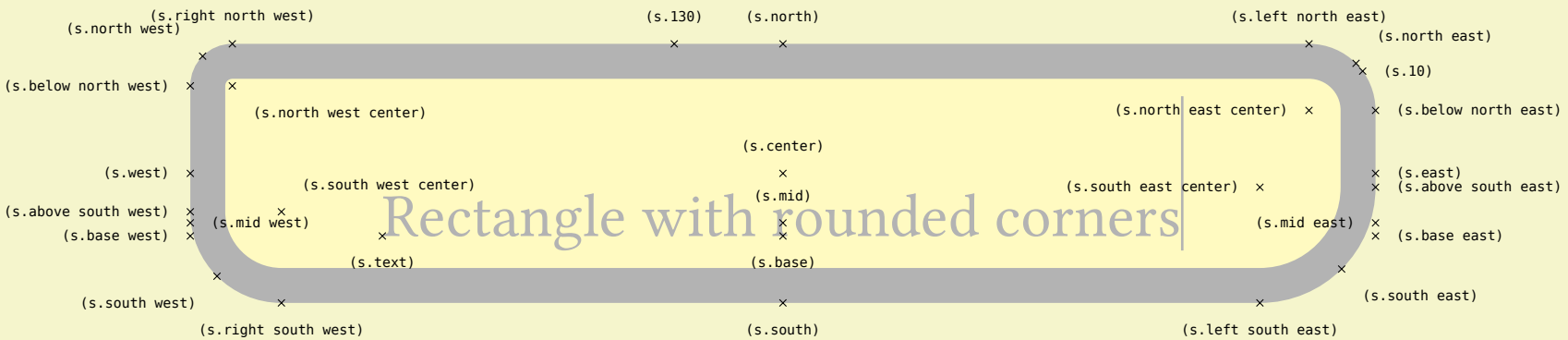
Sets the south west radius to ⟨*dimen*⟩.

/pgf/rectangle with rounded corners south east radius=⟨*dimen*⟩ (no default, initially `.5\pgflinewidth`)

Sets the south east radius to ⟨*dimen*⟩.

/pgf/rectangle with rounded corners radius=⟨*dimen*⟩ (no default)

Sets all radii to ⟨*dimen*⟩.




```
\usepgflibrary {ext.shapes.rectangleroundedcorners}
\begin{tikzpicture}\Huge
\node[name=s, shape=rectangle with rounded corners, shape example,
  rectangle with rounded corners north west radius=10pt,
  rectangle with rounded corners north east radius=20pt,
  rectangle with rounded corners south west radius=30pt,
  rectangle with rounded corners south east radius=40pt] {Rectangle with rounded corners\vrule width 1pt height 2cm};
\foreach \anchor/\placement in
  {north west/above left, north/above, north east/above right,
         west/left,      center/above,       east/right,
     mid west/right,        mid/above,   mid east/left,
    base west/left,        base/below,  base east/right,
   south west/below left, south/below, south east/below right,
   text/below, 10/right, 130/above,
   north west center/below right,      north east center/left,
   south west center/above right,      south east center/left,
   below north west/left,  above south west/left, above south east/right, below north east/right,
   right north west/above, right south west/below, left south east/below,  left north east/above}
   \draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
     node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```

21 Shape: Superellipse

TikZ Library `ext.shapes.superellipse`

```
\usepgflibrary{ext.shapes.superellipse}  % LaTeX and plain TeX and pure pgf
\usepgflibrary[ext.shapes.superellipse]  % ConTeXt and pure pgf
\usetikzlibrary{ext.shapes.superellipse} % LaTeX and plain TeX when using TikZ
\usetikzlibrary[ext.shapes.superellipse] % ConTeXt when using TikZ
```

Shape in the form of a “superellipse”.

Q & A: [40] & [22]

Shape `superellipse`

This shape is defined by formula

$$\left|\frac{x}{r_x}\right|^m + \left|\frac{y}{r_y}\right|^n = 1$$

and will be plotted by

$$x(t) = |\cos t|^{\frac{2}{m}} \cdot r_x \operatorname{sgn}(\cos t)$$
$$y(t) = |\sin t|^{\frac{2}{n}} \cdot r_y \operatorname{sgn}(\sin t)$$

where r_x is half the node's width and r_y is half the node's height.

`/pgf/superellipse x exponent`=⟨*x exponent*⟩ (no default, initially `2.5`)

This sets m.

`/pgf/superellipse y exponent`=⟨*y exponent*⟩ (no default, initially `2.5`)

This sets n.

`/pgf/superellipse step`=⟨*step*⟩ (no default, initially `5`)

This specifies the step of the underlying plot handler. The smaller ⟨*step*⟩ is, the slower computation will be.

Sensible values for ⟨*step*⟩ are integer dividers of 90, i. e. 2, 3, 5, 6, 9, 10, 15, 18, 30 and 45.

`/pgf/superellipse exponent`=⟨*exponent*⟩ (no default)

Sets both `superellipse x exponent` and `superellipse y exponent` to ⟨*exponent*⟩.

Notes on Implementation For implementing this shape, additional mathematical functions were declared.

`superellipsex(`t`, 2/m, `r_x`)`
`\pgfmathsuperellipsex{`t`}{2/m}{`r_x`}`

Returns the x value on a point of the superellipse with its center on the origin following

$$x = r_x \cos^{2/m} t$$

for values of $0 \leq t \leq 90$.

`superellipsey(`t`, 2/n, `r_y`)`
`\pgfmathsuperellipsey{`t`}{2/n}{`r_y`}`

Returns the y value on a point of the superellipse with its center on the origin following

$$y = r_y \cos^{2/n} t$$

for values of $0 \leq t \leq 90$.

Both PGFmath functions can be used at once with the following macro.

`\pgfmathsuperellipseXY{`$\langle t \rangle$`}{`$\langle 2/m \rangle$`}{`$\langle 2/n \rangle$`}{`$\langle a \rangle$`}{`$\langle b \rangle$`}`

Returns the x value (in `\pgfmathresultX`) and the y value (in `\pgfmathresultY`) of the superellipse with its center on the origin following

$$x = a \cos^{2/m} t$$
$$y = b \cos^{2/n} t$$

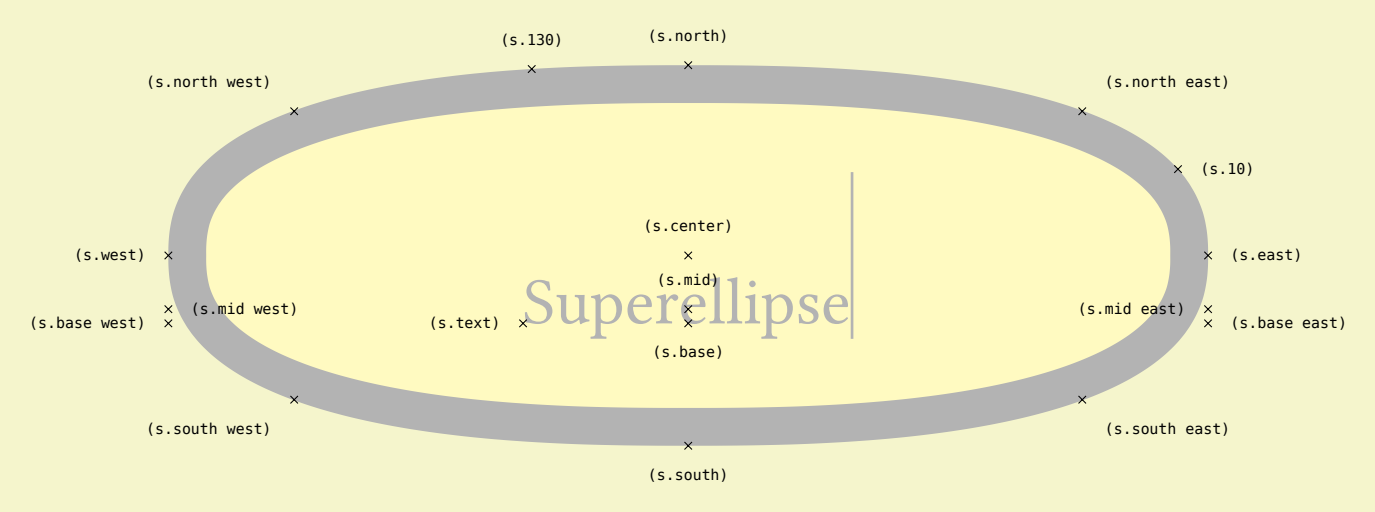
for values of $0 \leq t \leq 90$.

Note: all arguments must be a valid number since they will not be parsed by PGFmath.

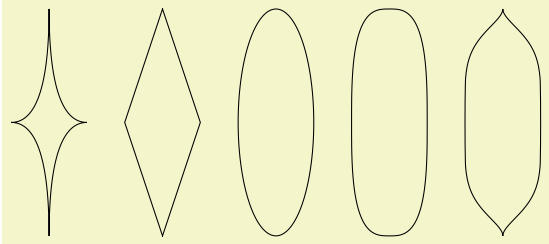
And additional internal macro was defined following the original naming scheme.

`\pgfutil@prefix@macrotomacro{`$\langle$ *macro 1* $\rangle$`}{`$\langle$ *macro 2* $\rangle$`}`

Adds the once-expansion of $\langle$ *macro 2* $\rangle$ in front of $\langle$ *macro 1* $\rangle$.




```
\usetikzlibrary {ext.shapes.superellipse}
\begin{tikzpicture}[superellipse step=1]\Huge
\node[name=s,shape=superellipse,shape example] {Superellipse\vrule width 1pt height 2cm};
\foreach \anchor/\placement in
  {north west/above left, north/above, north east/above right,
   west/left, center/above, east/right,
   mid west/right, mid/above, mid east/left,
   base west/left, base/below, base east/right,
   south west/below left, south/below, south east/below right,
   text/left, 10/right, 130/above}
   \draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
     node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
```



```
\usetikzlibrary {ext.shapes.superellipse}
\begin{tikzpicture}[minimum width=1cm, minimum height=3cm]
\foreach \xe/\ye[count=\i] in {.5/.5, 1/1, 2/2, 3/3, .5/5}
  \node[draw, superellipse, superellipse x exponent=\xe, superellipse y exponent=\ye] at (1.5*\i,0) {};
\end{tikzpicture}
```

22 Shape: Uncentered Rectangle

TikZ Library `ext.shapes.uncenteredrectangle`

```
\usepgflibrary{ext.shapes.uncenteredrectangle}  % LaTeX and plain TeX and pure pgf
\usepgflibrary[ext.shapes.uncenteredrectangle]  % ConTeXt and pure pgf
\usetikzlibrary{ext.shapes.uncenteredrectangle} % LaTeX and plain TeX when using TikZ
\usetikzlibrary[ext.shapes.uncenteredrectangle] % ConTeXt when using TikZ
```

A rectangle that has a variable horizontal center with three node parts.

Q & A: [42, 18] & [28, 25]

Shape `uncentered rectangle`

For some alignment problems, this shape could be useful.

It has three node parts: the standard `text` part, the `left` part that is to the left of `text` and the `right` part that is to the right of `text`.

When edges are to be connected with this shape, the following key changes to which inner center this shape will calculate the appropriate point on the border.

`/pgf/uncentered rectangle center`=⟨*left*⟩ or ⟨*text*⟩ or ⟨*right*⟩ or ⟨*real*⟩ (no default, initially `text`)

Sets the center that is to be used for connecting edges.

This will also move the anchors `north`, `mid`, `base` and `south` along. In the picture below, this are marked red.

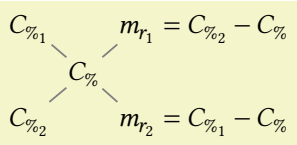
For support of the `cd` library of the `tikz-cd` package, this shape also supports a dynamic *y* value for its anchors `center`, `west` and `east`.

`/pgf/uncentered rectangle center yshift`=⟨*dimension*⟩ (no default, initially `{}`)

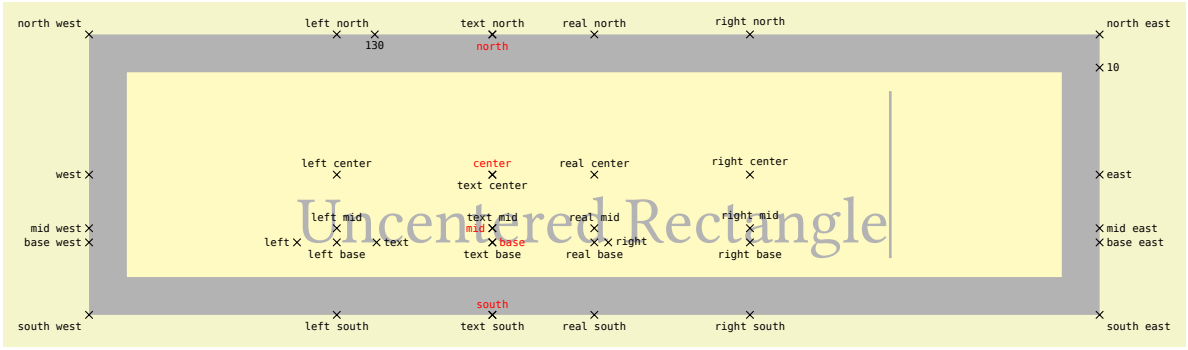
This determines the distance between the baseline and the `center` anchors.

If ⟨*dimension*⟩ is empty, the real center will be used which is the default.

For use with `cd`, set this to `axis_height`. Due to a bug with `/tikz/execute at end node` this needs a lot of fixing to be able to use in a commutative diagram, though.

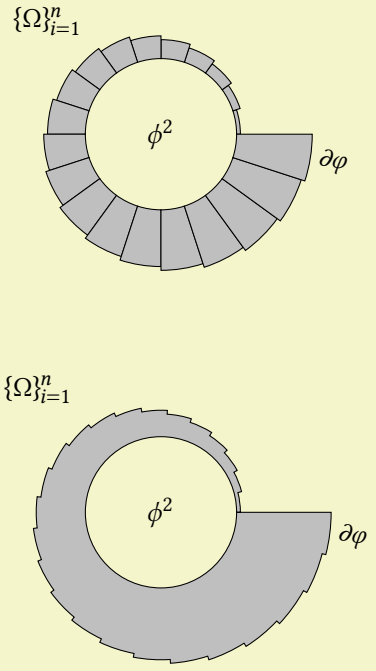



```
\usetikzlibrary {cd, ext.shapes.uncenteredrectangle}
\makeatletter
\tikzcdset{
  every diagram/.append style={
    /pgf/uncentered rectangle center=text,
    /tikz/math align/.style={
      shape=uncentered rectangle,
      /pgf/uncentered rectangle center yshift=axis_height,
      commutative diagrams/math mode=false},
    /tikz/math align left/.style={
      math align,
      /utils/exec={\def\mathalign####1=####2;{$####2$\nodepart{left}$####1={}$}}},
    /tikz/math align right/.style={
      math align,
      /utils/exec={\def\mathalign####1=####2;{$####1$\nodepart{right}${}=####2$}}},
    /tikz/matrix of math nodes/.style={
      matrix of nodes,
      nodes={
        execute at begin node=\iftikzcd@mathmode$\fi,
        execute at end node  =\iftikzcd@mathmode$\fi}}}}
\makeatother
\tikzcdset{install C shortcut/.code=\newcommand*\C[1]{C_{\%_{##1}}}}
\begin{tikzcd}[
  install C shortcut,
  sep=tiny,
  arrows={-, gray},
  cells={font=\strut, inner xsep=.2ex, inner ysep=.1ex}
]
\C{1} \drar &         & |[math align right]| \mathalign m_{r_1} = \C{2}-C_\%; \dlar\\
            & C_\% \\
\C{2} \urar &         & |[math align right]| \mathalign m_{r_2} = \C{1}-C_\%; \ular
\end{tikzcd}
```




```
\usepgflibrary {ext.shapes.uncenteredrectangle}
\begin{tikzpicture}[style north/.style=red, style south/.style=red, style center/.style=red, style base/.style=red, style mid/.style=red]
\Huge
\node[shape example, name=n, uncentered rectangle]
  {centered \nodepart{left} Un \nodepart{right} \space Rectangle\vrule width 1pt height 2cm}
  foreach \anchor/\pos in {
   north west/above left, north/below, north east/above right, real north/above,  left north/above, right north/above, text north/above,
         west/left,      center/above,       east/right,       real center/above, left center/above,right center/above,text center/below,
     mid west/left,         mid/left,    mid east/right,       real mid/above,    left mid/above,   right mid/above,   text mid/above,
    base west/left,        base/right,  base east/right,       real base/below,   left base/below,  right base/below,  text base/below,
   south west/below left, south/above, south east/below right, real south/below,  left south/below, right south/below, text south/below,
                           10/right,        130/below,                            left/left,        right/right,       text/right}{
    plot[mark=x, only marks] coordinates {(n.\anchor)}
    node[inner sep=.1em, style \anchor/.try, style/.expand once=\pos] {\tiny\ttfamily\anchor}};
\end{tikzpicture}
```

Part IV
Utilities




```
\usetikzlibrary {ext.misc}
\begin{tikzpicture}[
  declare function={bigR(\n)=smallR+.05*\n;},
  declare constant={smallR=1; segments=20;},
  full arc=segments]
\foreach \iN[evaluate={\endRadius=bigR(\iN+1);}, use int=0 to segments-1]
  \filldraw[fill=gray!50] (\iN R:\endRadius)
    arc [radius=\endRadius, start angle=\iN R, delta angle=+1R] -- (\iN R+1R:smallR)
    arc [radius=smallR,       end angle=\iN R, delta angle=-1R] -- cycle;

\node                                              {$\phi^2$};
\node at (north west:{sqrt 2 * bigR(segments/2)})  {$\{\Omega\}_{i=1}^n$};
\node[rotate=-.5R, right] at (-.5R: bigR segments) {$\partial \varphi$};

\tikzset{yshift=-5cm, declare constant={segments=25;}, full arc=segments}
\filldraw[fill=gray!50] (right:smallR)
  \foreach \iN[evaluate={\endRadius=bigR(\iN+1);}, use int=0 to segments-1] {
    -- (\iN R:\endRadius) arc[radius=\endRadius, start angle=\iN R, delta angle=1R]}
    -- (right:smallR)     arc[radius=smallR,     start angle=0,     delta angle=-360];

\node                                              {$\phi^2$};
\node at (north west:{sqrt 2 * bigR(segments/2)})  {$\{\Omega\}_{i=1}^n$};
\node[rotate=-.5R, right] at (-.5R: bigR segments) {$\partial \varphi$};
\end{tikzpicture}
```

23 Calendar: Weeknumbers and more conditionals

```
\usepackage{calendar-ext} % LaTeX
\input calendar-ext.tex   % plain TeX
```

This package adds week numbers and more conditionals to the PGF package `pgfcalendar`.

Q & A: [8, 9, 11] & [21, 37, 26]

23.1 Extensions

The following tests are added.

- `Jan` This test is passed by all dates that are in the month of January.
- `Feb` as above.
- `Mar` as above.
- `Apr` as above.
- `May` as above.
- `Jun` as above.
- `Jul` as above.
- `Aug` as above.
- `Sep` as above.
- `Oct` as above.
- `Nov` as above.
- `Dec` as above.
- `leap year=`⟨*year*⟩ This test checks whether the given year is a leap year. If ⟨*year*⟩ is omitted, it checks the year of the current date.
- `and={`⟨*tests*⟩`}` This test passes when all ⟨*tests*⟩ pass.
- `not={`⟨*tests*⟩`}` This test passes when ⟨*tests*⟩ do not pass.
- `week of month=`⟨*num*⟩ This test passes when the date is in ⟨*num*⟩th week of the month. The first week of the month start at day 1 and ends with day 7.
- `week of month'=`⟨*num*⟩ As above but counts from the last day of the month. For a month with 31 days, this means the "1st" week starts at day 25 and ends with day 31.
- `calendar week of month=`⟨*num*⟩ This test passes when the date is in ⟨*num*⟩th calendar week of the month. The first week starts at the first day of the month and ends at the next Sunday.
- `calendar week of month'=`⟨*num*⟩ As above but counts from the last day of the month.


```
\usetikzlibrary {ext.calendar-plus}
\tikz
  \calendar[
    dates=2022-10-01 to 2022-10-31,
    week list]
    if (week of month=2) [red]
    if (calendar week of month'=2) [blue];
```

- `yesterday={`⟨*tests*⟩`}` This test passes when the previous day passes ⟨*tests*⟩.
- `week=`⟨*num*⟩ This test passes when the current week of the year equals `{`⟨*num*⟩`}`.

The shorthands for `d-` and `m-` are slightly changed so that they are expandable. This makes it possible to use these shorthands inside of PGFmath. The shorthands for the week (see section 23.2) are added. These are

- `n-` (shortest numerical representation),
- `n=` (shortest but added horizontal space) and
- `n0` (leading zero when below 10).

23.2 Week numbering (ISO 8601)

`\pgfcalendarjulianyeartoweek`{⟨*Julian day*⟩}{⟨*year*⟩}{⟨*week count*⟩}

This command calculates the week for the ⟨*Julian day*⟩ of ⟨*year*⟩. The ⟨*week counter*⟩ must be a TEX count.

The calculation follows the rule of ISO 8601 where the first week has that year's first Thursday in it.

Inside of `\pgfcalendar` the command `\pgfcalendarcurrentweek` will be available.

`\pgfcalendarcurrentweek`

This command returns the current week number (always two digits – use shorthand `n.` to strip the leading zero).

Inside of `\ifdate` the command `\pgfcalendarifdateweek` will be available.

`\pgfcalendarifdateweek`

This command returns the week number (always two digits).

24 Repeating Things and Other Things

```
\usepackage{pgffor-ext} % LaTeX
\input pgffor-ext.tex   % plain TeX
```

This package adds small niceties to the `pgffor` package. Most of these additions are also available with the `ext.misc` library.

Warning: Consider this package experimental. At the very least, it will break the `...` notation and possibly gobbles spaces after the body.

Q & A: [1, 6, 43] & [29, 33, 31]

Instead of `\foreach \var in {start, start + delta, ..., end}` one can use `\foreach \var[use int=start to end step delta]`.

`/pgf/foreach/use int=`⟨*start*⟩`to`⟨*end*⟩`step`⟨*delta*⟩ (no default)

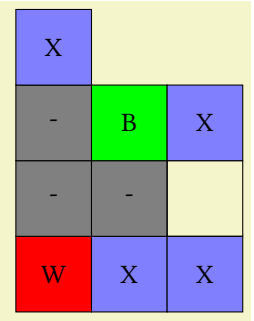
The values ⟨*start*⟩, ⟨*end*⟩ and ⟨*delta*⟩ are evaluates by PGFmath at initialization. The part `step` ⟨*delta*⟩ is optional (⟨*delta*⟩ = 1).

`/pgf/foreach/use float=`⟨*start*⟩`to`⟨*end*⟩`step`⟨*delta*⟩ (no default)

Same as above, however the results are not truncated.

`/pgf/foreach/no separator` (no value)

This key disables any separator between elements of the list. Every token is its own element. This also means that Unicode characters need to be grouped between `{` and `}` if LuaTeX isn't used. Spaces will be ignored.




```
\usetikzlibrary {ext.misc}
\newcommand*{\board}[3][]{%
  \begin{tikzpicture}[#1]
    \foreach[
      count=\i from 0,
      no separator,
      evaluate=\i as \colX using {mod(\i,#2)},
      evaluate=\i as \rowY using {int(\i/#2)}
    ] \elem in {#3} {
        \draw[black, board/\elem/.try, rectangle timer/.try=line]
          (\colX,\rowY) rectangle node {\elem} ++(1, 1);}
  \end{tikzpicture}}
\board[
  board/W/.style={fill=red},
  board/X/.style={fill=blue!50},
  board/B/.style={fill=green},
  board/-/.style={fill=gray},
]{3}{WXX---BXX}
```

`/pgf/foreach/normal list` (no value)

This key simply disables all other special parsers and returns to the original list parser.

The following keys only work with LaTeX and cannot be used when only the `ext.misc` library or the plainTeX `pgffor-ext.tex` are loaded. For this, you will need to use `\usepackage{pgffor-ext}`.

`/pgf/foreach/xparser`=`{`⟨*argument specification*⟩`}{`⟨*foreach value*⟩`}` (no default)

This key can be used to specify a `xparse` specification for each element in the list.

For this to work somewhat seamless, the following needs to observed:

- Every `{`⟨*argument specification*⟩`}` get appended `u,`. This means there's always one additional mandatory argument at the end of every element.
- The `{`⟨*foreach value*⟩`}` needs to correspond to the `/pgf/foreach/var` value.

`/pgf/foreach/xparser 0m` (no value)

Sets up a list whose elements may contain an optional argment inside `[]` which correspond to two `\foreach` variables, say `\Options/\Text`.

Key handler ⟨*key*⟩`/.list xparse`=`{`⟨*argument specification*⟩`}{`⟨*comma-separated list of values*⟩`}`

This handler causes the key to be used repeatedly, namely once for every element of the list of values. The ⟨*comma-separated list of values*⟩ is processed using `\foreach` and the given `xparse` ⟨*argument specification*⟩ with the aforementioned `xparser` key.

25 And a little bit more

Ti*k*Z Library `ext.misc`

```
\usetikzlibrary{ext.misc} % LaTeX and plain TeX
\usetikzlibrary[ext.misc] % ConTeXt
```

This library adds miscellaneous utilities to PGFmath, PGF or Ti*k*Z.

Q & A: [17] & [20]

25.1 PGFmath

25.1.1 Postfix operator R

Similar to `\segments[<num>]` in PSTricks, the postfix operator `R` allows the user to use an arbitrary number of segments of a circle to be used instead of an angle.

`/pgf/full arc=`⟨*num*⟩ (default {})

The number ⟨*num*⟩ of segments will be set up. Using `full arc` with an empty value disables the segmentation and `1R` equals 1°.

The given value ⟨*num*⟩ is evaluated when the key is used and doesn't change when ⟨*num*⟩ contains variables that change.

The `R` operator can then be used.

x`R` (postfix operator; uses the `fullarc` function)

Multiplies *x* with $\frac{360}{\langle num \rangle}$.

25.1.2 Functions

`strrepeat("`*Text*`", `*x*`)`
`\pgfmathstrrepeat{"`*Text*`"}{`*x*`}`

Returns a string with *Text* repeated *x* times.

foofoofoofoofoo

```
\pgfmathparse{strrepeat("foo", 5)}
\pgfmathresult
```

`isInString("`*String*`", "`*Text*`")`
`\pgfmathisInString{"`*String*`"}{"`*Text*`"}`

Returns `1` (true) if *Text* contains *String*, otherwise `0` (false).

0 and 1

```
\pgfmathparse{isInString("foo", "bar")}
\pgfmathresult \ and\
\pgfmathparse{isInString("foo", "foobar")}
\pgfmathresult
```

`strcat("`*Text A*`", "`*Text B*`", …)`
`\pgfmathstrcat{"`*Text A*`"}{"`*Text B*`"}{…}`

Returns the concatenation of all given parameters.

blue!21!green

```
\pgfmathparse{strcat("blue!", int(7*3), "!green")}
\pgfmathresult
```

`isEmpty("`*Text*`")`
`\pgfmathisEmpty{"`*Text*`"}`

Returns `1` (true) if *Text* is empty, otherwise `0` (false).

0 and 1 and 1

```
\pgfmathparse{isEmpty("foo")} \pgfmathresult\ and\
\pgfmathparse{isEmpty("")}    \pgfmathresult\ and\
\def\emptyText{}
\pgfmathparse{isEmpty("\emptyText")} \pgfmathresult
```

`atanXY(`*x*`,`*y*`)`

`\pgfmathatanXY{`*x*`}{`*y*`}`

Arctangent of $y \div x$ in degrees. This also takes into account the quadrant. This is just a argument-swapped version of `atan2` which makes it easier to use the `\p` commands of the `calc` library.

53.13011 `\pgfmathparse{atanXY(3,4)} \pgfmathresult`

`atanYX(`*y*`,`*x*`)`
`\pgfmathatanYX{`*y*`}{`*x*`}`

Arctangent of $y \div x$ in degrees. This also takes into account the quadrant.

53.13011 `\pgfmathparse{atanYX(4,3)} \pgfmathresult`

25.1.3 Functions: using coordinates

The following functions can only be used with PGF and/or Ti*k*Z. Since the arguments are usually plain text (and not numbers) one has to wrap them in `"`.

`anglebetween("`*p1*`", "`*p2*`")`
`\pgfmathanglebetween{"`*p1*`"}{"`*p2*`"}`

Return the angle between the centers of the nodes *p1* and *p2*.

`qanglebetween("`*p*`")`
`\pgfmathqanglebetween{"`*p*`"}`

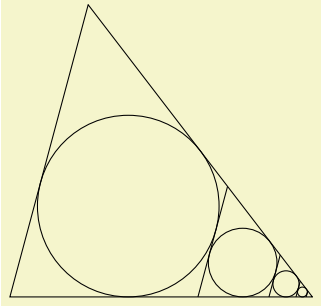
Return the angle between the origin and the center of the node *p*.

`distancebetween("`*p1*`", "`*p2*`")`
`\pgfmathdistancebetween{"`*p1*`"}{"`*p2*`"}`

Return the distance (in pt) between the centers of the nodes *p1* and *p2*.

`qdistancebetween("`*p*`")`
`\pgfmathqdistancebetween{"`*p*`"}`

Return the distance (in pt) between the origin and the center of the node *p*.



```
\usetikzlibrary {calc,ext.misc,through}
\begin{tikzpicture}
\path (0,0) coordinate (A) + (0:4) coordinate (B) +(75:4) coordinate (C);
\draw (A) -- (B) -- (C) -- cycle;
\foreach \cnt in {1,...,4}{
  \pgfmathsetmacro\triA{distancebetween("B","C")}
  \pgfmathsetmacro\triB{distancebetween("C","A")}
  \pgfmathsetmacro\triC{distancebetween("A","B")}
  \path (barycentric cs:A=\triA,B=\triB,C=\triC) coordinate (M)
       node [draw, circle through=($(A)!(M)!(C)$)] (M) {};
  \draw ($(C)-(A)$) coordinate (vecB)
      (M.75-90) coordinate (@)
      (intersection of @--[shift=(vecB)]@ and B--C) coordinate (C) --
      (intersection of @--[shift=(vecB)]@ and B--A) coordinate (A);}
\end{tikzpicture}
```

25.2 PGFfor

This library loads also most of the functions of the `pgffor-ext` of section 24 on page 59.

25.3 PGFkeys

pgfkeys Library `ext.pgfkeys-plus`

```
\usepgfkeyslibrary{ext.pgfkeys-plus} % LaTeX and plain TeX
\usepgfkeyslibrary[ext.pgfkeys-plus] % ConTeXt
```

This extends PGFkeys and adds helpful `/utils` keys as well as handlers. This library gets loaded by the `ext.misc` library.[7]

25.3.1 Conditionals

`/utils/if`={⟨*cond*⟩}{⟨*true*⟩}{⟨*false*⟩} (no default)

This key checks the conditional ⟨*cond*⟩ and applies the styles ⟨*true*⟩ if ⟨*cond*⟩ is true, otherwise ⟨*false*⟩. ⟨*cond*⟩ can be anything that PGFmath understands.

As a side effect on how PGFkeys parses argument, the ⟨*false*⟩ argument is actually optional.

The following keys use TEX' macros `\if`, `\ifx`, `\ifnum` and `\ifdim` for faster executions.

`/utils/TeX/if`=⟨*token A*⟩⟨*token B*⟩{⟨*true*⟩}{⟨*false*⟩} (no default)

This key checks via `\if` if ⟨*token A*⟩ matches ⟨*token B*⟩ and applies the styles ⟨*true*⟩ if it does, otherwise ⟨*false*⟩.

As a side effect on how PGFkeys parses argument, the ⟨*false*⟩ argument is actually optional.

`/utils/TeX/ifx`=⟨*token A*⟩⟨*token B*⟩{⟨*true*⟩}{⟨*false*⟩} (no default)

As above but via `\ifx`.

`/utils/TeX/ifnum`={⟨*num cond*⟩}{⟨*true*⟩}{⟨*false*⟩} (no default)

This key checks `\ifnum`⟨*num cond*⟩ and applies the styles ⟨*true*⟩ if true, otherwise ⟨*false*⟩. A delimiting `\relax` will be inserted after ⟨*num cond*⟩.

As a side effect on how PGFkeys parses arguments, the ⟨*false*⟩ argument is actually optional.

`/utils/TeX/ifdim`=⟨*dim cond*⟩⟨*true*⟩⟨*false*⟩ (no default)

As above but with `\ifdim`.

`/utils/TeX/ifempty`=⟨*Text*⟩⟨*true*⟩⟨*false*⟩ (no default)

This checks whether ⟨*Text*⟩ is empty and applies styles ⟨*true*⟩ if true, otherwise ⟨*false*⟩.

`/utils/TeX/ifxempty`=⟨*Text*⟩⟨*true*⟩⟨*false*⟩ (no default)

This checks whether fully expanded ⟨*Text*⟩ is empty and applies styles ⟨*true*⟩ if true, otherwise ⟨*false*⟩.

25.3.2 Handlers

While already a lot of values given to keys are evaluated by PGFmath at some point, not all of them are.

Key handler ⟨*key*⟩`/.pgfmath`=⟨*eval*⟩

This handler evaluates ⟨*eval*⟩ before it is handed to the key.

This handler works almost the same as the `.evaluated` handler but it does its evaluation in a group so that the result will not overwrite any other results.

Key handler ⟨*key*⟩`/.pgfmath int`=⟨*eval*⟩

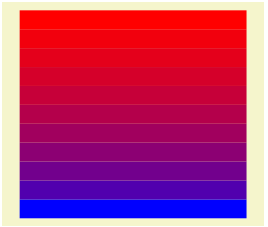
As above but truncates the result.

Key handler ⟨*key*⟩`/.pgfmath wrap`={⟨*wrapper*⟩}{⟨*eval*⟩}

This feeds the result of ⟨*eval*⟩ as `#1` to ⟨*wrapper*⟩.

In the example below, one could have used the `/pgf/foreach/evaluate` key from the `\foreach` loop.

[7] `\usepgfkeyslibrary` is an upcoming feature of PGF/TikZ. For now, you need to load `ext.misc` or manually `\input` the file `pgfkeyslibraryext.pgfkeys-plus.code.tex` with `@` being a letter.



```
\usetikzlibrary {ext.pgfkeys-plus}
\tikz\foreach \i in {0,10,...,100}
\draw[
  line width=+.25cm,
  % needs ## because its inside the \foreach body
  color/.pgfmath wrap={red!##1!blue}{sqrt(\i)*10}
]
  (0,\i/40) -- +(right:3);
```

Key handler ⟨*key*⟩**/.pgfmath if=**{⟨*cond*⟩}{⟨*true*⟩}{⟨*false*⟩}

Evaluates ⟨*cond*⟩ with PGFMath and returns ⟨*true*⟩ or ⟨*false*⟩ to the used key respectively.

Key handler ⟨*key*⟩**/.if=**⟨*token A*⟩⟨*token B*⟩{⟨*true*⟩}{⟨*false*⟩}

Checks via `\if` if ⟨*token A*⟩ matches ⟨*token B*⟩ and applies the value ⟨*true*⟩ if it does, otherwise ⟨*false*⟩.

Key handler ⟨*key*⟩**/.ifx=**⟨*token A*⟩⟨*token B*⟩{⟨*true*⟩}{⟨*false*⟩}

As above but via `\ifx`.

Key handler ⟨*key*⟩**/.ifnum=**{⟨*ifnum cond*⟩}{⟨*true*⟩}{⟨*false*⟩}

Checks via `\ifnum` if ⟨*ifnum cond*⟩ and applies the value ⟨*true*⟩ if it does, otherwise ⟨*false*⟩.

Key handler ⟨*key*⟩**/.ifdim=**{⟨*ifdim cond*⟩}{⟨*true*⟩}{⟨*false*⟩}

As above but via `\ifdim`.

Key handler ⟨*key*⟩**/.ifxempty=**{⟨*Text*⟩}{⟨*true*⟩}{⟨*false*⟩}

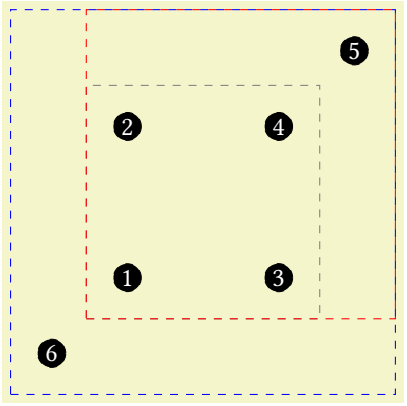
Checks whether a fully expanded ⟨*Text*⟩ is empty and applies the value ⟨*true*⟩ if it does, otherwise ⟨*false*⟩.

Key handler ⟨*key*⟩**/.ifempty=**{⟨*Text*⟩}{⟨*true*⟩}{⟨*false*⟩}

Checks whether ⟨*Text*⟩ is empty and applies the value ⟨*true*⟩ if it does, otherwise ⟨*false*⟩.

Key handler ⟨*key*⟩**/.List=**{⟨⟨*e1*⟩, ⟨*e2*⟩, ..., ⟨*en*⟩⟩}

This handler evaluates the given list with `\foreach` and concatenates the element and the result is then given to the used key.



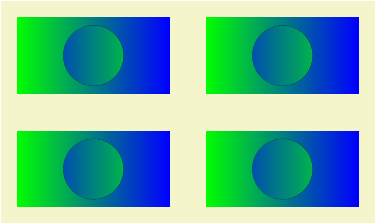

```
\usetikzlibrary {fit,ext.misc}
\begin{tikzpicture}[nodes={draw, dashed, inner sep=+10pt}]
  \foreach \point [count=\cnt] in {(0,0), (0,2), (2,0), (2,2), (3,3), (-1,-1)}
    \node[circle, fill, inner sep=1pt, text=white] (point-\cnt) at \point {\cnt};
  \node[gray, fit/.List={(point-1),(point-...),(point-4)}] {};
  \node[red,  fit/.List={(point-1),(point-...),(point-5)}] {};
  \node[blue, fit/.List={(point-1),(point-...),(point-6)}] {};
\end{tikzpicture}
```

`/tikz/reverse clip=`⟨*direction*⟩ (default `counter clockwise`)

This key installs a very big rectangle which is either constructed `counter clockwise` (like the `circle` path operation) or `clockwise`.

`/tikz/clip rule=`⟨*direction*⟩ (default `even odd`)

This key switches directly[8] to the specified rule which is either `even odd` or `nonzero`. This corresponds to the `/tikz/even odd rule` and `/tikz/nonzero rule` keys.



```
\usetikzlibrary {ext.misc}
\newcommand*\myDiagram[1]{
  \fill[left color=blue, right color=green] (0, 0) rectangle (2, 1);
  \clip (1, .5) #1 [reverse clip];
  \fill[left color=green, right color=blue] (0, 0) rectangle (2, 1);
}
\begin{tikzpicture}[radius=.4, row sep=5mm, column sep=5mm]
\matrix[
  row 2/.append style={clip rule=even odd},
  column 1/.append style={reverse clip/.default=clockwise}
]{
  \myDiagram{circle[]} &
  \myDiagram{+(0:.4) arc[start angle=0, delta angle=-360] -- cycle}
\\
  \myDiagram{circle[]} &
  \myDiagram{+(0:.4) arc[start angle=0, delta angle=-360] -- cycle}
\\};
\end{tikzpicture}
```

[8] Meaning, it directly executes `\pgfseteorule` /`\pgfsetnonzerorule` and doesn't accumulates where Ti*k*Z throws an error.

Part V
Changelog, Index & References

Changelog

Version 0.5

- Added package `pgffor-ext`.
- Added Ti*k*Z library `ext.nodes`.
- Bugfixes to `ext.calendar-plus`.
- Allow the original `rectangle` timer with `ext.paths.timer`.

Version 0.4.2

- Added Ti*k*Z library `ext.scalepicture`.
- Bugfixes to `shapes.uncenteredrectangle`, `paths.ortho`, `positioning-plus` and `pgfcalender-ext`.

Version 0.4.1

- Cleaned up directory structure of documentary.
- Added PGFkeys library `ext.pgfkeys-plus`.
- Added shape `uncentered rectangle` (PGF library `ext.shapes.uncenteredrectangle`).
- Fixed `ext.paths.arcto` – again [15].

Version 0.4

- CTAN version of 0.3.1

Version 0.3.1

- Fixed `ext.paths.ortho` keys `only vertical first` and `only horizontal first`.
- Moved all (except the `to paths`) to namespace `/tikz/ortho`. `/tikz/hvvh` and `/tikz/udlr` are considered deprecated.
- Fixed `\pgfcalendarjulianyeartoweek`.
- Added more calendar tests.
- Added directory structure.

Version 0.3

- Added shape `circle arrow` (PGF library `ext.shapes.circlearrow`).
- Added shape `circle cross split` (PGF library `ext.shapes.circlecrosssplit`).
- Added shape `heatmark` (PGF library `ext.shapes.heatmark`).
- Added shape `rectangle with rounded corners` (PGF library `ext.shapes.rectangleroundedcorners`).
- Added shape `superellipse` (PGF library `ext.shapes.superellipse`).
- Added Ti*k*Z library `ext.node-families.shapes.geometric`.
- Fixed `ext.node-families`' key `size`.
- Renamed internal macros to use custom namespace starting with `\tikzext@`.
- Added some references.

Version 0.2

- Added Ti*k*Z library `ext.positioning-plus`.

- Added Ti*k*Z library `ext.node-families`.

Version 0.1

- Added Ti*k*Z library `ext.calendar-plus`.
- Added Ti*k*Z library `ext.misc`.
- Added Ti*k*Z library `ext.paths.arcto`.
- Added Ti*k*Z library `ext.paths.ortho`.
- Added Ti*k*Z library `ext.paths.timer`.
- Added Ti*k*Z library `ext.patterns.images`.
- Added Ti*k*Z library `ext.topaths.arcthrough`.
- Added Ti*k*Z library `ext.transformations.mirror`.
- Added PGF library `ext.transformations.mirror`.

Index

This index contains automatically generated entries as well as references to original functionalities of PGF/Ti*k*Z and references to functionalities outside of PGF/Ti*k*Z.

References

[1] Foo Bar. *How to use declared TikZ functions in \foreach condition?* TeX - LaTeX Stack Exchange. Apr. 2013. URL: https://tex.stackexchange.com/q/110962 (visited on 09/24/2022) (cit. on p. 59).

[2] boje. *Heatmap over country like Google Map.* May 2013. URL: https://tex.stackexchange.com/q/112929 (visited on 09/24/2022) (cit. on p. 45).

[3] cis. *TikZ / calendar: Set the height of a monthly calendar.* Dec. 2018. URL: https://tex.stackexchange.com/q/464589 (visited on 09/24/2022) (cit. on p. 7).

[4] cis. *TikZ: How to place a coordinate at parabola-path-position?* May 2020. URL: https://tex.stackexchange.com/q/543251 (visited on 09/24/2022) (cit. on p. 21).

[5] CrazyArm. *Is It Possible to Combine TikZ Distance and Line-To Operations?* Apr. 2013. URL: https://tex.stackexchange.com/q/106558 (visited on 09/24/2022) (cit. on p. 21).

[6] daan. *String conditional tikz.* TeX - LaTeX Stack Exchange. Nov. 2022. URL: https://tex.stackexchange.com/q/666263 (visited on 12/03/2022) (cit. on p. 59).

[7] Dimitris. *Draw two concentric circles and a shaded area with associated text.* TeX - LaTeX Stack Exchange. Dec. 2022. URL: https://tex.stackexchange.com/q/667338 (visited on 12/12/2022) (cit. on p. 12).

[8] Fence. *Add week day to calendar.* Nov. 2019. URL: https://tex.stackexchange.com/q/517338 (visited on 09/24/2022) (cit. on pp. 7, 57).

[9] healyp. *TikZ calendar and conditional tests.* Oct. 2013. URL: https://tex.stackexchange.com/q/140948 (visited on 09/24/2022) (cit. on pp. 7, 57).

[10] Holene. *Dependent node size in TikZ.* Apr. 2017. URL: https://tex.stackexchange.com/q/107227 (visited on 09/24/2022) (cit. on p. 9).

[11] jd6. *Full weeks in Tikz Calendar.* TeX - LaTeX Stack Exchange. Dec. 2020. URL: https://tex.stackexchange.com/q/576673 (visited on 10/09/2022) (cit. on p. 57).

[12] knut. *TikZ: Define pattern with reference to external picture.* Mar. 2013. URL: https://tex.stackexchange.com/q/103980 (visited on 09/24/2022) (cit. on p. 24).

[13] Ben Liblit. *path with both mark connection node and arrow tip.* TeX - LaTeX Stack Exchange. Feb. 2013. URL: https://tex.stackexchange.com/q/99945 (visited on 12/12/2022) (cit. on p. 12).

[14] Marco. *TikZ - Four Colored Circle Split.* Apr. 2017. URL: https://tex.stackexchange.com/q/121686 (visited on 09/24/2022) (cit. on p. 42).

[15] marmotghost. *clockwise/counter clockwise does not seem to work reliably.* Oct. 2022. URL: https://github.com/Qrrbrbirlbel/tikz-extensions/issues/2 (visited on 10/23/2022) (cit. on p. 66).

[16] nkk. *How to prevent tikz custom node fill from covering the text when using node-families library.* June 2019. URL: https://tex.stackexchange.com/q/494862 (visited on 09/24/2022) (cit. on p. 11).

[17] Anthony Peter. *A rather difficult ring like picture to be drawn.* Apr. 2017. URL: https://tex.stackexchange.com/q/144293 (visited on 09/24/2022) (cit. on p. 61).

[18] projetmbc. *forest - automatic setting of the alignment of some labels.* TeX - LaTeX Stack Exchange. Oct. 2022. URL: https://tex.stackexchange.com/q/661726 (visited on 10/23/2022) (cit. on p. 53).

[19] projetmbc. *TikZ - "Circled" arrow.* TeX - LaTeX Stack Exchange. Jan. 2013. URL: https://tex.stackexchange.com/q/95221 (visited on 09/24/2022) (cit. on p. 39).

[20] Qrrbrbirlbel. *Answer to "A rather difficult ring like picture to be drawn".* Nov. 2013. URL: https://tex.stackexchange.com/a/144297 (visited on 09/24/2022) (cit. on p. 61).

[21] Qrrbrbirlbel. *Answer to "Add week day to calendar".* July 2022. URL: https://tex.stackexchange.com/a/651888 (visited on 09/24/2022) (cit. on pp. 7, 57).

[22] Qrrbrbirlbel. *Answer to "An oval surrounded a *long text* inside in TikZ [equivalent cover background of METAFUN]".* TeX - LaTeX Stack Exchange. Aug. 2022. URL: https://tex.stackexchange.com/a/654759 (visited on 09/24/2022) (cit. on p. 50).

[23] Qrrbrbirlbel. *Answer to "Dependent node size in TikZ"*. June 2013. URL: https://tex.stackexchange.com/a/121054 (visited on 09/24/2022) (cit. on p. 9).

[24] Qrrbrbirlbel. *Answer to "Draw two concentric circles and a shaded area with associated text"*. TeX - LaTeX Stack Exchange. Dec. 2022. URL: https://tex.stackexchange.com/a/667341 (visited on 12/12/2022) (cit. on p. 12).

[25] Qrrbrbirlbel. *Answer to "forest - automatic setting of the alignment of some labels"*. TeX - LaTeX Stack Exchange. Oct. 2022. URL: https://tex.stackexchange.com/a/661746 (visited on 10/23/2022) (cit. on p. 53).

[26] Qrrbrbirlbel. *Answer to "Full weeks in Tikz Calendar"*. TeX - LaTeX Stack Exchange. Oct. 2022. URL: https://tex.stackexchange.com/a/660335 (visited on 10/09/2022) (cit. on p. 57).

[27] Qrrbrbirlbel. *Answer to "Heatmap over country like Google Map"*. May 2013. URL: https://tex.stackexchange.com/a/113004 (visited on 09/24/2022) (cit. on p. 45).

[28] Qrrbrbirlbel. *Answer to "How to draw a mixing rule? #chemistry"*. TeX - LaTeX Stack Exchange. Sept. 2022. URL: https://tex.stackexchange.com/a/657449 (visited on 10/23/2022) (cit. on p. 53).

[29] Qrrbrbirlbel. *Answer to "How to use declared TikZ functions in \foreach condition?"* TeX - LaTeX Stack Exchange. Apr. 2013. URL: https://tex.stackexchange.com/a/110996 (visited on 09/24/2022) (cit. on p. 59).

[30] Qrrbrbirlbel. *Answer to "Is It Possible to Combine TikZ Distance and Line-To Operations?"* Apr. 2013. URL: https://tex.stackexchange.com/a/106571 (visited on 09/24/2022) (cit. on p. 21).

[31] Qrrbrbirlbel. *Answer to "Is there a package to implement this style of "Register diagrams with field descriptions""*. TeX - LaTeX Stack Exchange. Dec. 2022. URL: https://tex.stackexchange.com/a/667155 (visited on 12/03/2022) (cit. on p. 59).

[32] Qrrbrbirlbel. *Answer to "path with both mark connection node and arrow tip"*. TeX - LaTeX Stack Exchange. Dec. 2022. URL: https://tex.stackexchange.com/a/667487 (visited on 12/12/2022) (cit. on p. 12).

[33] Qrrbrbirlbel. *Answer to "String conditional tikz"*. TeX - LaTeX Stack Exchange. Nov. 2022. URL: https://tex.stackexchange.com/a/666265 (visited on 12/03/2022) (cit. on p. 59).

[34] Qrrbrbirlbel. *Answer to "TikZ - 'Circled' arrow"*. TeX - LaTeX Stack Exchange. Jan. 2013. URL: https://tex.stackexchange.com/a/95263 (visited on 09/24/2022) (cit. on p. 39).

[35] Qrrbrbirlbel. *Answer to "TikZ - Four Colored Circle Split"*. June 2013. URL: https://tex.stackexchange.com/a/121767 (visited on 09/24/2022) (cit. on p. 42).

[36] Qrrbrbirlbel. *Answer to "TikZ / calendar: Set the height of a monthly calendar"*. Aug. 2022. URL: https://tex.stackexchange.com/a/653146 (visited on 09/24/2022) (cit. on p. 7).

[37] Qrrbrbirlbel. *Answer to "TikZ calendar and conditional tests"*. Oct. 2013. URL: https://tex.stackexchange.com/a/141027 (visited on 09/24/2022) (cit. on pp. 7, 57).

[38] Qrrbrbirlbel. *Answer to "TikZ: Define pattern with reference to external picture"*. Apr. 2013. URL: https://tex.stackexchange.com/a/107144 (visited on 09/24/2022) (cit. on p. 24).

[39] Qrrbrbirlbel. *Answer to "TikZ: How to place a coordinate at parabola-path-position?"* Nov. 2021. URL: https://tex.stackexchange.com/a/621012 (visited on 09/24/2022) (cit. on p. 21).

[40] somenxavier. *An oval surrounded a *long text* inside in TikZ [equivalent cover background of METAFUN]*. TeX - LaTeX Stack Exchange. Aug. 2022. URL: https://tex.stackexchange.com/q/649144 (visited on 09/24/2022) (cit. on p. 50).

[41] Andrew Stacey. *spath3 TikZ library*. original-date: 2014-05-26T12:08:12Z. Dec. 2022. URL: https://github.com/loopspace/spath3 (visited on 12/10/2022) (cit. on p. 13).

[42] Michał Szymankiewicz. *How to draw a mixing rule? #chemistry*. TeX - LaTeX Stack Exchange. Sept. 2022. URL: https://tex.stackexchange.com/q/657432 (visited on 10/23/2022) (cit. on p. 53).

[43] uulinux. *Is there a package to implement this style of "Register diagrams with field descriptions"*. TeX - LaTeX Stack Exchange. Oct. 2021. URL: https://tex.stackexchange.com/q/618047 (visited on 12/03/2022) (cit. on p. 59).