

The TikZ-Extensions Package

Manual for version 0.6

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

Qrrbrbirlbel

November 2, 2023

Contents

I	Introduction	5
1	Usage	5
2	Why do we need it?	5
3	Having problems?	5
II	TikZ Libraries	6
4	Arrow Pics	7
4.1	Arrow pic types	7
4.2	Arrow keys	8
4.3	Shifted and bended arrows for the decorations.markings library	9
5	Calendar	10
5.1	Value-keys and nestable if key	10
5.2	PGFmath functions	10
5.3	Week numbering (ISO 8601)	10

6	Layers	11
6.1	Internal keys	11
6.2	User-level keys	11
7	Node Families	12
7.1	Text Box	12
7.2	Minimum Width/Height	13
7.3	More shapes that support the keys width and height	14
8	Nodes	15
8.1	Pic as a node	15
8.2	Nodes on paths	15
8.2.1	Nodes on Lines	15
8.2.2	Nodes on Curves	16
8.3	Automatic placement of nodes	16
8.3.1	More than left and right	16
8.3.2	Offset	16
8.3.3	Precise placement	17
9	Arc to a point	18
10	More Horizontal and Vertical Lines	20
10.1	Zig-Zag	20
10.2	Zig-Zig	22
10.3	Even more Horizontal and Vertical Lines	23
11	Extending the Path Timers	26
11.1	Rectangle	26
11.2	Parabola	27
11.3	Sine/Cosine	27
12	Using Images as a Pattern	29
13	Positioning Plus	30
13.1	Useful corner anchors	30
13.2	Useful placement keys for vertical and horizontal alignment	31

14	Scaling Pictures to a Specific Size	35
14.1	Keeping the aspect ratio	35
14.2	Changing the aspect ratio.	35
15	Arcs through Three Points	37
16	Arcs through Three Points	38
17	Mirror, Mirror on the Wall	40
17.1	Using the reflection matrix	40
17.2	Using built-in transformations	41
III	PGF Libraries	43
18	Arrow Tips	44
18.1	Centered	45
18.1.1	Barbed Arrow Tips	45
18.1.2	Geometric Arrow Tips	45
18.1.3	Special Arrow Tips	45
18.2	Untipped	46
18.2.1	Barbed Arrow Tips	46
18.2.2	Geometric Arrow Tips	46
18.3	Original Arrow Tips	46
19	Transformations: Mirroring	48
19.1	Using the reflection matrix	48
19.2	Using built-in transformations	48
20	Shape: Circle Arrow	50
21	Shape: Circle Cross Split	53
22	Shape: Heatmark	56
23	Shape: Rectangle with Rounded Corners	59
24	Shape: Superellipse	61
25	Shape: Uncentered Rectangle	64

IV	Utilities	67
26	Calendar: Weeknumbers and more conditionals	68
26.1	Extensions	68
26.2	Week numbering (ISO 8601)	69
27	Repeating Things and Other Things	70
28	And a little bit more	72
28.1	PGFmath	72
28.1.1	Postfix operator R	72
28.1.2	Functions	72
28.1.3	Functions: using coordinates	73
28.2	PGFfor	73
28.3	PGFkeys	74
28.3.1	Conditionals	74
28.3.2	Handlers	74
28.4	TikZ	76
V	Changelog, Index & References	77
	Changelog	77
	Index	79
	References	91

Part I

Introduction

1 Usage

This package is called `tikz-ext`, however, one can't load it via `\usepackage`.¹ Instead, this package consists mostly of `pgf` and `TikZ` libraries which are loaded by either `\usepgflibrary` or `\usetikzlibrary`.

2 Why do we need it?

Since I have been answering questions on [TeX.sx](https://tex.stackexchange.com/) 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?

Note however, that most of these extensions haven't been stress-tested properly and might be considered experimental.

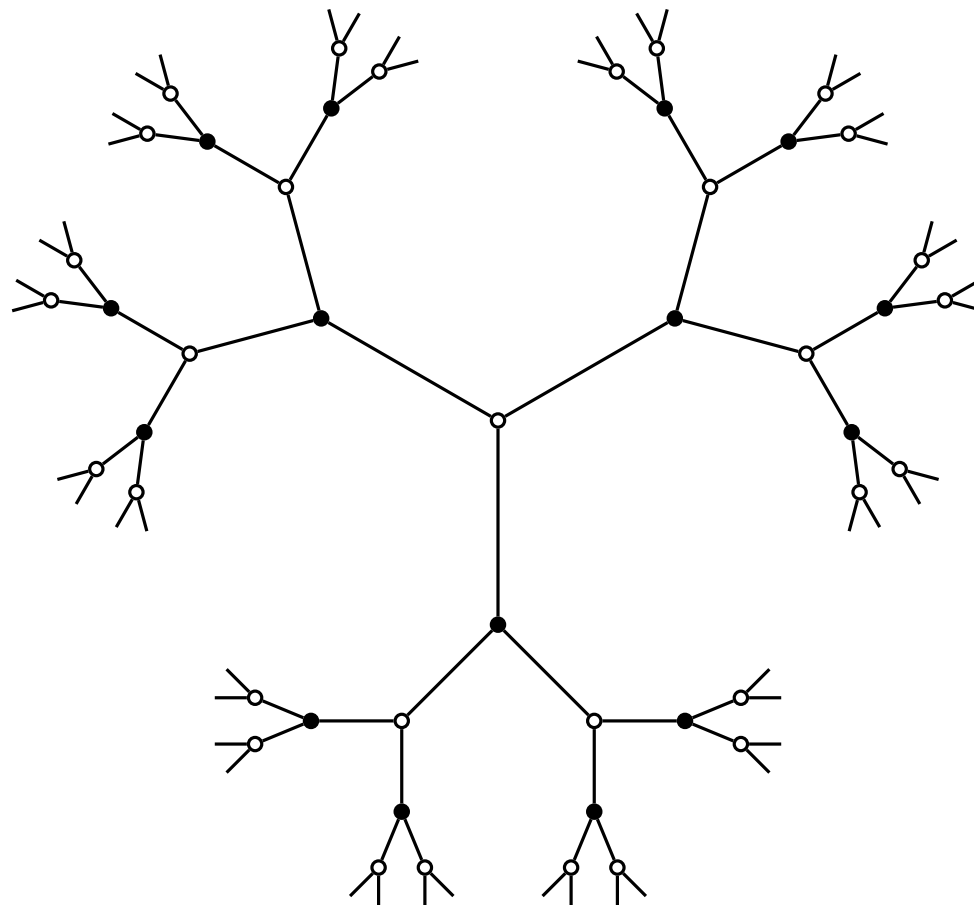
Don't hesitate to open an issue on GitHub. You probably found a bug.

¹Except for `pgfcalendar-ext` and `pgffor-ext`.

Part II

TikZ Libraries

These libraries only work with TikZ.



4 Arrow Pics

TikZ Library `ext.arrows-plus`

```
\usetikzlibrary{ext.arrows-plus} % LATEX and plain TEX
\usetikzlibrary[ext.arrows-plus] % ConTEXt
```

This library defines pics and keys that can be used to place (bended) arrow tips on paths.

The markings decoration already provides the functionality to place arrow tips along the path. The pics and keys provided by this library serve as an alternative.

Many of the pics and keys share various keys that specify where and how the arrow tips are placed.

`/tikz/pos <=<(value)` (no default, initially 0.0)

If the pic type supports it and an start arrow tip sequence is provided this specifies the position of that sequence.

`/tikz/pos >=>(value)` (no default, initially 0.5)

This is an alias for `/tikz/pos`, if an end arrow tip sequence is provided, it is placed at this position.

`/tikz/arrow shift mode=<shift mode>` (no default, initially total length)

This key is used to set the `<shift mode>` for the arrow tip. It can be one of the following.

arrow shift mode=off This disables the shifting.

arrow shift mode=total length The total length of the whole arrow tip sequence will be used.

arrow shift mode=total This is an alias for `total length`.

arrow shift mode=length until line end The length of the whole arrow tip until the line end will be used – as reported by pgf which might not always be the expected one.

arrow shift mode=line end This is an alias for `length until line end`.

For single arrow tips it might be better to use the Centered arrow tip variants of the `ext.arrows` library (see sec 18) and disabled `arrow shift mode`.

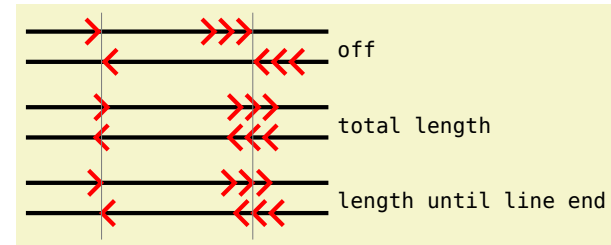
When an arrow tip sequence is to be drawn depending on the shift mode its total length or its length until the line end will be determined and multiplied with the

arrow shift factor. The result of this evaluation is used to shift the arrow tip sequence in the tip's direction.

`/tikz/arrow shift factor=<value>` (no default, initially 0.5)

This determines the shift factor.

The default value is probably good for most cases.



```
\usetikzlibrary {ext.arrows-plus}
\begin{tikzpicture}[>={Straight Barb[color=red]}, ultra thick]
\ttfamily
\foreach[count=\y] \shiftmode in {off, total length, length until line end}
\draw[arrow shift mode=\shiftmode] (0, -\y )
-- pic {arrow=>} ++(right:2)
-- pic {arrow=>.>} ++(right:2) node[below right] {\shiftmode}
++(down:.4) -- pic {arrow=>.>} ++( left:2)
-- pic {arrow=>} ++( left:2);
\draw[thin, gray] (1,-.75) -- +(down:3) (3,-.75) -- +(down:3);
\end{tikzpicture}
```

4.1 Arrow pic types

This library provides the following pics:

arrow This is the simplest implementation to place an arrow tip along a path. It uses the current timer that is also used to place nodes.

It can be used without any adjustment for every path operation that provides such a timer. These do *not* include circle, ellipse, plot and grid. For rectangle, parabola, sin and cos, the ext.paths.timer library is recommended or even necessary (see section 11).

The arrow tips will never be bended. For this the following pic types or the /tikz/arc arrows key will be necessary.

Due to [1] with an active transformation, the arrow tips won't be placed correctly in many cases. For this *and* bended arrow tips the following pics are necessary.

softpath arrows This pic type places a possible bended arrow tip on the last segment of the path.

For the path operators --, |- and -| this works even with a non-identity transformation. If possible, the current timer will be used to take more segments into account so that the arrow tip can be placed along the recent path operation.

This won't work for arcs, for this the /tikz/arc arrows key will be necessary.

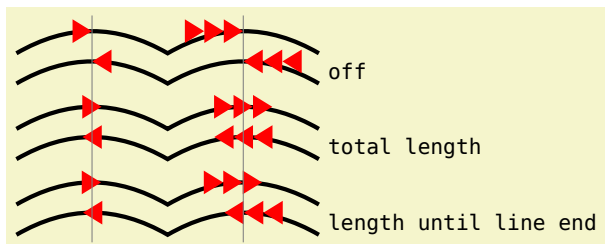
This pic type can place two tip specification, one at pos > and one at pos < in the reversed direction.

softpath arrow This is an alias for softpath arrows with an empty start arrow tip specification.

Pic type arrow=<arrow tip specification>

This pic draws the given <arrow tip specification> (default >).

This obviously is best use as a pic along a path segment that supports it. It *does not* support bended arrow tips.



```
\usetikzlibrary {bending, ext.arrows-plus}
\begin{tikzpicture}[>={Triangle[color=red]}, arrows={[bend]}, ultra thick]
\ttfamily
\foreach[count=\y] \shiftmode in {off, total length, length until line end}
\draw[arrow shift mode=\shiftmode] (0, -\y )
    to[bend left] pic {arrow=>} ++(right:2)
    to[bend left] pic {arrow=>.>} ++(right:2)
    node[below right] {\shiftmode}
    ++(down:.4) to[bend right] pic {arrow=>.>} ++(left:2)
    to[bend right] pic {arrow=>} ++(left:2);
\draw[thin, gray] (1,-.5) -- +(down:3) (3,-.5) -- +(down:3);
\end{tikzpicture}
```

Pic type softpath arrows=<start tip specification>-<end tip specification>

This pic draws the given arrow tip specification along the previous path segment (a curve or a line). It supports the pos < key.

Note: For arcs with an angle greater than 90° this will not work as expected. Use the arc arrows key instead.

Tip: Use an empty option specification [] after the first arrow tip in the start specification to help it correctly apply the shifting distance.

Pic type softpath arrow=<end tip specification>

This pic type is an alias for softpath arrows = -<end tip specification>.

4.2 Arrow keys

The last pic type softpath arrows is also available as a key which is the preferred version.

/tikz/softpath arrows=<options> (default ->)

This key adds arrow tips to the previous path segment (a curve or a line).

/tikz/every softpath arrows (style, initially {})

This style will be applied for every instance of softpath arrows (key version, not the pic).

For arcs the following key needs to be used directly after the arc path operation.

`/tikz/arc arrows=<options>` (default ->)

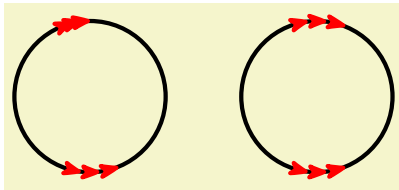
This key adds arrow tips to the previous arc segment.

`/tikz/every softpath arrows` (style, initially {})

This style will be applied for every instance of `arc arrows`.

Tip: Use an arc with the full 360° to place bended arrow tips along a circle or an ellipse.

Tip: Use an empty option specification [] after the first arrow tip in the start specification to help it correctly apply the shifting distance.



```
\usetikzlibrary {bending, ext.arrows-plus}
\tikz[>={Stealth[color=red, round]}, arrows={bend}], ultra thick]
\draw[start angle=0, end angle=360, radius=1,
every arc arrows/.style={pos <=.25, pos >=.75}]
(0, 0) arc[] [arc arrows= < <<->>>] -- cycle
(3, 0) arc[] [arc arrows={<[]<<->>>}] -- cycle;
```

4.3 Shifted and bended arrows for the decorations.markings library

Many paths are not properly accessible by the previous methods. If this library is loaded *after* the decorations.markings library, both the `\arrow` and the `\arrowreversed` macros are enhanced.

`\arrow**[<options>]{<arrow end tip>}`

This macro works the same as before but the one-starred version applies the shifting as specified by `/tikz/arrow shift mode` and `/tikz/arrow shift factor` where as the two-starred version also bends the arrow tip.

`\arrowreversed**[<options>]{<arrow end tip>}`

As above, only the arrow end tip is flipped and points in the other direction.



```
\usetikzlibrary {bending, decorations.markings, ext.arrows-plus}
\tikzset{
arr/.style={
postaction=decorate,
decoration={
name=markings,
mark={between positions .25 and 1 step .25 with
\arrow#1[red]{> _ < _ >}}}}
\tikz[y=1.5cm, >={Stealth, arrows={round}}, nodes={circle, draw}]
\path node[arr= ]{Ti\emph kZ} % \arrow
(0,-1) node[arr=* ]{Ti\emph kZ} % \arrow*
(0,-2) node[arr=**]{Ti\emph kZ} % \arrow**
;
```

5 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: [11, 12, 5] & [28, 49, 47]

5.1 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)

5.2 PGFmath functions

`weeks in month of year(first weekday, month, year)`

`\pgfmathweeks in month of year{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*.

`last day in month of year(month, year)`

`\pgfmathlast day in month of year{month}{year}`

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

5.3 Week numbering (ISO 8601)

The actual week number algorithm is implemented by the `pgfcalendar-ext` package/module in section 26.2.

`/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=\sfamily}};
```

6 Layers

TikZ Library `ext.layers`

```
\usetikzlibrary{ext.layers} % LATEX and plain TEX
\usetikzlibrary[ext.layers] % ConTEXt
```

This library extends TikZ’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.

6.1 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)

6.2 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)

²Only 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.

7 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: [14] & [31]

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`.

7.1 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 were aligned to the left. With `text width family align` this can be changed.

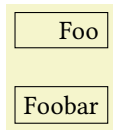
³First of all, I can't figure out how to use the AUX file during externalization since it gets written to the LOG instead. And then there's the question about how `external` would notice the need to export the picture again until it's stable ...

⁴Technically, 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.

7.2 Minimum Width/Height

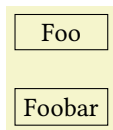
While the keys of the previous subsection work well enough for nodes of the same shape (and the same inner seps), for different node shapes the text box dimensions will be used differently for the node's total dimension.

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.

7.3 More shapes that support the keys width and height

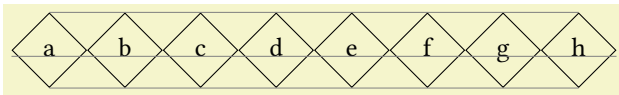
TikZ Library `ext.node-families.shapes.geometric`

```
\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: [23]

The shapes are also setup for the keys from subsection 7.1.



```
\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}
```

8 Nodes

TikZ Library `ext.nodes`

```
\usetikzlibrary{ext.nodes} % LATEX and plain TEX
\usetikzlibrary[ext.nodes] % ConTEXt
```

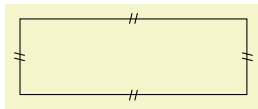
This library extends TikZ's functionalities when it comes to nodes.

Q & A: [10, 18] & [33, 42]

8.1 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}[
  \sll/.pic={\draw(-2pt, 1.5pt)--( 2pt, .5pt)
              ( 2pt,-1.5pt)--(-2pt,-.5pt);}]
\node[
  draw, minimum width=3cm, minimum height=1cm,
  label={[pic] east:sll},
  label={[pic, rotate= 90]north:sll},
  label={[pic] west:sll},
  label={[pic, rotate=-90]south:sll}]{};
\end{tikzpicture}
```

8.2 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).

8.2.1 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.

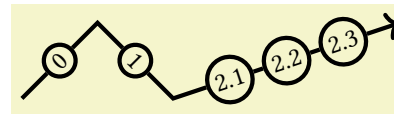
This allows a node to be placed *on* a straight line without having to use `fill = white` or similar tricks to make the line disappear beneath the node.

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);
```

8.2.2 Nodes on Curves

The following keys need the intersections and the ext.spath3 [54] 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.

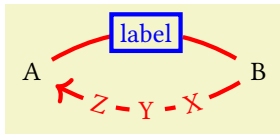
`/tikz/nodes on curve=<to path>` (style, default line to)

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);
```

8.3 Automatic placement of nodes

The `/tikz/auto` key allows automatic placement of nodes along a path segment. This library extends this in various ways.

8.3.1 More than left and right

Besides left and right that are provided by TikZ the following placement mechanism are provided:

- Left will place a node to the left of the direction of the line,
- Right will place a node to the right of the direction of the line,
- Above will place a node towards the direction of the line,
- Below will place a node against the direction of the line,
- West will place a node towards the left side of the paper,
- East will place a node towards the right side of the paper,
- North will place a node towards the upper side of the paper and
- South will place a node towards the lower side of the paper.

The placement mechanism Left and Right are like the original left and right mechanism but don't swap sides when `/tikz/sloped` is used.

Certain cases exist for West, East, North and South placements where it is not clear how a node should be placed. These cases and their behavior can be seen in figure 1.

8.3.2 Offset

Nodes are usually placed with their border (including any outer sep) on the line. With the following option, a node will be shifted a certain offset distance.

`/tikz/auto with offset=<true or false>` (default true)

This key activated the offset function.

`/tikz/auto offset` (initially 1cm)

The offset distance itself.

For the brace decoration, the following keys are provided which needs the `decorations.pathreplacing` loaded before they can be used.

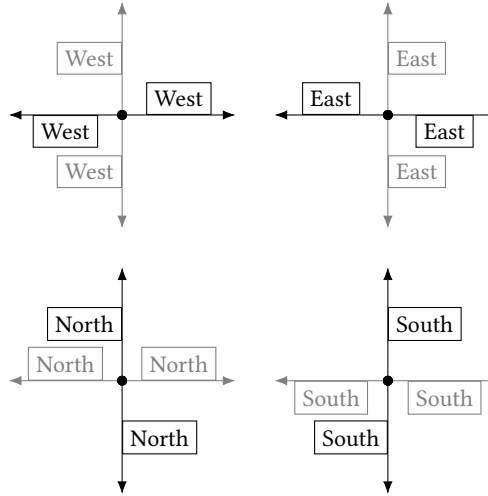


Figure 1: Behavior of West, East, North and South in certain cases

`/tikz-ext/nodes/install auto offset for brace decoration=<distance>`
(default 0pt)

This key installs the necessary customizations for the `/pgf/decorationraise` key so that the given value is available as an offset.

It also makes available the following keys.

`/tikz/auto offset for brace decoration` (style, no value)

This sets `/tikz/auto offset` to `\pgfdecorationsegmentamplitude+`
(`\pgfkeysvalueof/pgf/decoration/raise`).

`/tikz/every brace node` (style, no value)

Using this key on a node along a path that's decorated by the brace decoration will offset the node so that it will be placed at the tip of the brace.

8.3.3 Precise placement

The default behavior of the auto placement mechanism is to snap to one of the eight compass directions.

`/tikz/precise auto angle=<true or false>` (default true)

With this option set to true, the auto placement won't snap to one of the eight compass directions.

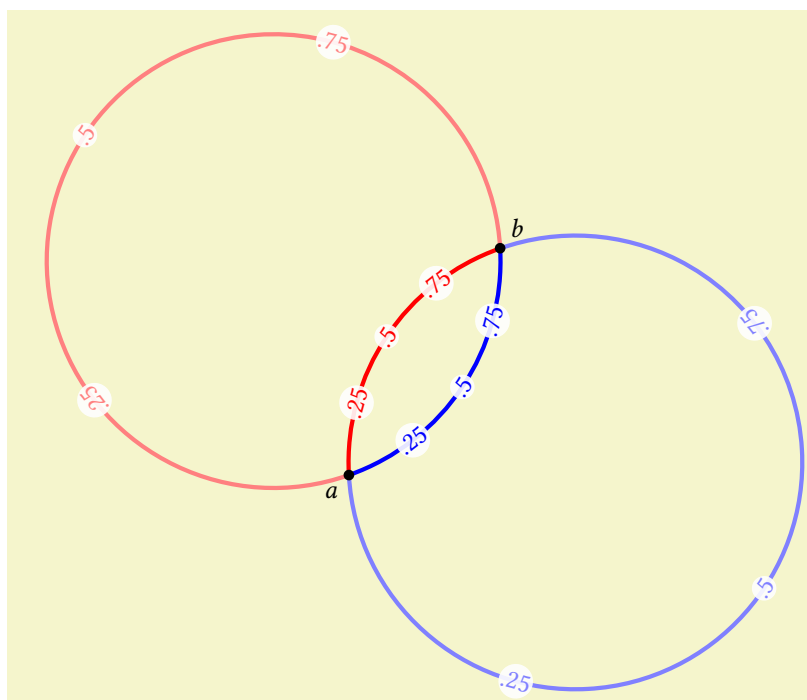
This key disables the `/tikz/sloped` option which in turn will disable this option.

9 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` 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= $\langle degree \rangle$`

(no default)

Rotates the arc by $\langle degree \rangle$. This is only noticeable when x radius and y radius are different.

`/tikz/arc to/x radius= $\langle value \rangle$`

(no default)

This forwards the $\langle value \rangle$ to `/tikz/x radius`. Its $\langle value \rangle$ is used for the radius of the arc.

`/tikz/arc to/y radius= $\langle value \rangle$`

(no default)

This forwards the $\langle value \rangle$ to `/tikz/y radius`. Its $\langle value \rangle$ is used for the radius of the arc.

`/tikz/arc to/radius= $\langle value \rangle$`

(no default)

This forwards the $\langle value \rangle$ to both `/tikz/x radius` and `/tikz/y radius`. Its $\langle value \rangle$ 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 $\langle options \rangle$ are set.

It should be noted that this uses `\pgfpatharcto` for which the TikZ manual warns:

The internal computations necessary for this command are numerically very unstable. In particular, the arc will not always really end at the $\langle target coordinate \rangle$, 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.

10 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`.

10.1 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 ...;`

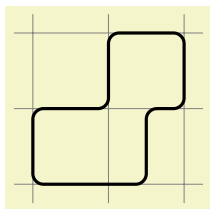
This operation means “first horizontal, then vertical and then horizontal again”.

`/tikz/ortho/ratio= $\langle ratio \rangle$`

(no default, initially 0.5)

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

For values $\langle ratio \rangle < 0$ and $\langle ratio \rangle > 1$ the Zig-Zag lines will look more like Zig-Zig lines.



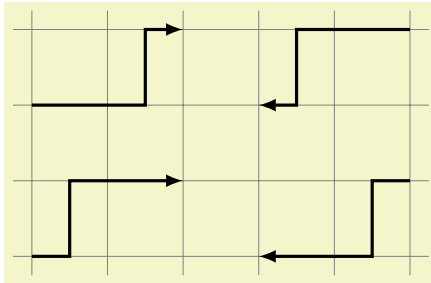
```
\usetikzlibrary{ext.paths.ortho}
\begin{tikzpicture}[very thick, rounded corners]
\draw[help lines] (-.25, -1.25) grid (2.25, 1.25);
\draw (0, 0) -|- (2, 1) --
(2, 0) -|- [ratio=.25] (0, -1) -- cycle;
\end{tikzpicture}
```

`/tikz/ortho/distance= $\langle distance \rangle$`

(no default)

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

For values $\langle distance \rangle < 0$ 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);

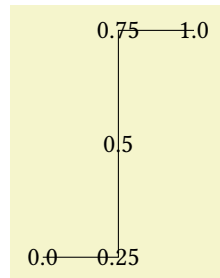
\tikzset{xshift=3cm}
\draw (2, 1) -|-[distance=.5cm] ++(-2, -1);
\draw (2, 3) -|-[distance=-.5cm] ++(-2, -1);
\end{tikzpicture}
```

`/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 {ext.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 $\langle number \rangle = 0$ 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 $\langle number \rangle = 0$ 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`.

10.2 Zig-Zig

`\path ... r-ud[$\langle options \rangle$] $\langle coordinate or cycle \rangle$...;`

This operation means “first up, then horizontal and then down”.

`/tikz/ortho/ud distance= $\langle length \rangle$`

(no default, initially .5cm)

This sets the distance between the start and the horizontal line to $\langle length \rangle$.

`\path ... r-du[$\langle options \rangle$] $\langle coordinate or cycle \rangle$...;`

This operation means “first down, then horizontal and then up”.

`/tikz/ortho/du distance= $\langle length \rangle$`

(no default, initially .5cm)

This sets the distance between the start and the horizontal line to $\langle length \rangle$.

`\path ... r-lr[$\langle options \rangle$] $\langle coordinate or cycle \rangle$...;`

This operation means “left down, then vertical and then right”.

`/tikz/ortho/lr distance= $\langle length \rangle$`

(no default, initially .5cm)

This sets the distance between the start and the vertical line to $\langle length \rangle$.

`\path ... r-rl[$\langle options \rangle$] $\langle coordinate or cycle \rangle$...;`

This operation means “first right, then vertical and then down”.

`/tikz/ortho/rl distance= $\langle length \rangle$`

(no default, initially .5cm)

This sets the distance between the start and the vertical line to $\langle length \rangle$.

All distances can be set with one key.

`/tikz/ortho/udlr distance= $\langle length \rangle$`

(no default)

Sets all the previous distances to the same value $\langle length \rangle$.

10.3 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=<options>` (style, no default)

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

`/tikz/vertical horizontal vertical=<options>` (style, no default)

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

`/tikz/up horizontal down=<options>` (style, no default)

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

`/tikz/down horizontal up=<options>` (style, no default)

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

`/tikz/left vertical right=<options>` (style, no default)

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

`/tikz/right vertical left=<options>` (style, no default)

This installs to `path = r-rl (\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
- `r-ud` → up horizontal down
- `r-du` → down horizontal up
- `r-lr` → left vertical right
- `r-rl` → right vertical left

11 Extending the Path Timers

TikZ 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: [7, 6] & [39, 51]

In TikZ, 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.

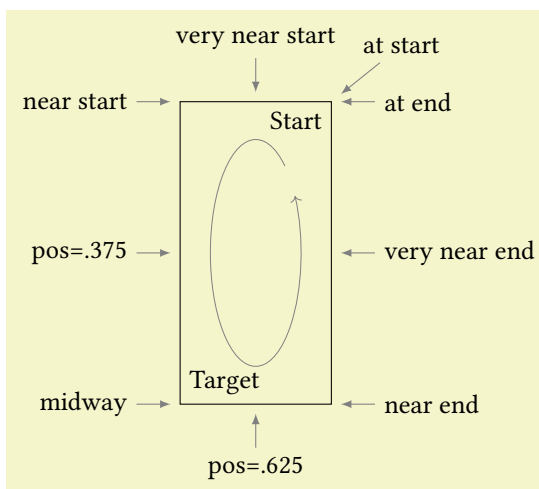
11.1 Rectangle

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}
```

11.2 Parabola

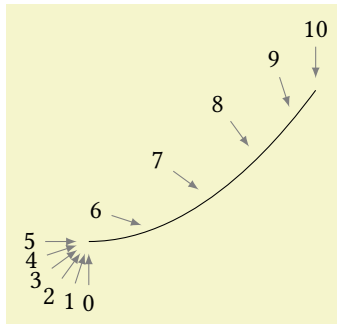
For the parabola path operator the timer is similar to the `.. controls ..` operator.

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}
```

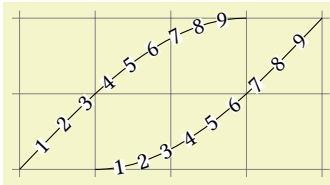
If no bend is specified half the positions will collapse into one end of the curve.



```
\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}
```

11.3 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}
```

12 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: [17] & [50]

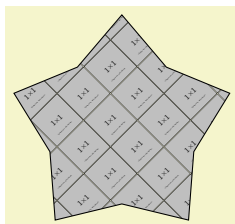
With this library arbitrary images (or indeed PDF documents) can be used as a repeating pattern for the background of a path. This is a two-step process:

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`.

13 Positioning Plus

TikZ 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.

13.1 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 TikZ library positioning but uses the `corner north west` anchor.

`/tikz/corner below left=<specification>` (style, default 0pt)

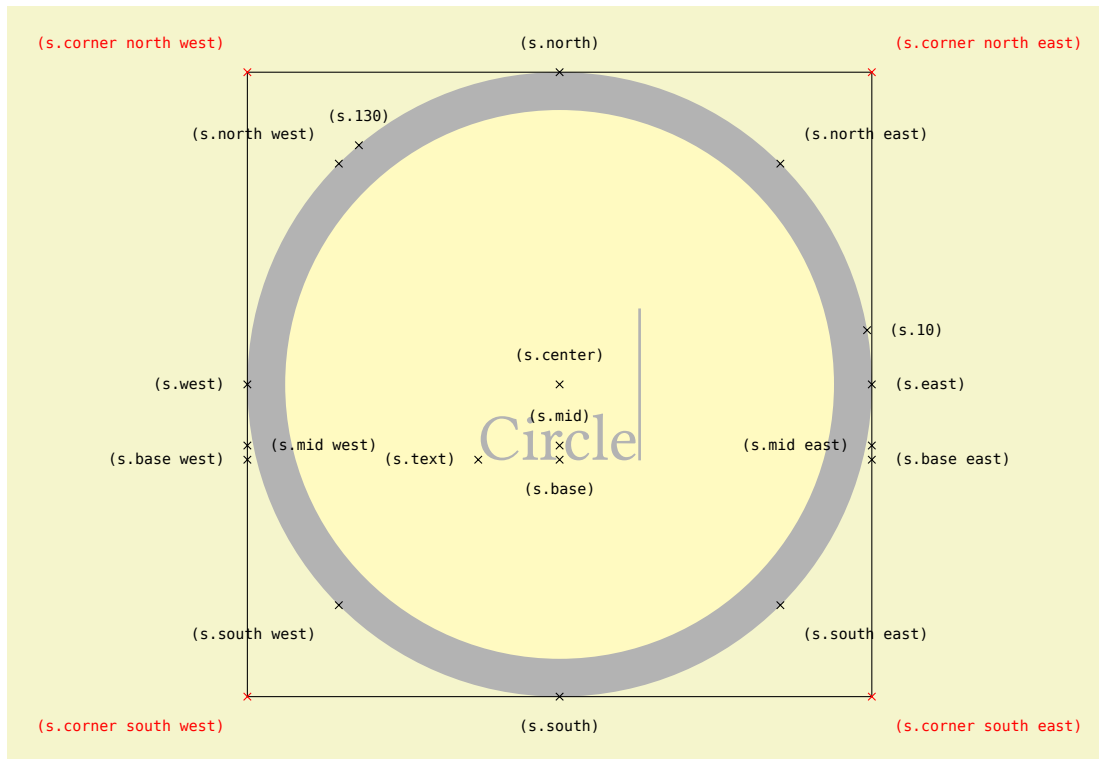
Similar as `/tikz/below left` of the TikZ library positioning but uses the `corner south west` anchor.

`/tikz/corner above right=<specification>` (style, default 0pt)

Similar as `/tikz/above right` of the TikZ library positioning but uses the `corner north east` anchor.

`/tikz/corner below right=<specification>` (style, default 0pt)

Similar as `/tikz/below right` of the TikZ 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}
```

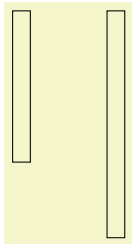
13.2 Useful placement keys for vertical and horizontal alignment

`/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.

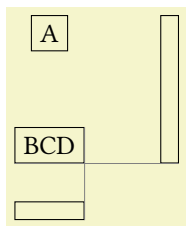
`/tikz/corner east below=<specification>` (style, default 0pt)

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 TikZ, 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 `/pgf/minimum 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 `/pgf/minimum 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

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

14 Scaling Pictures to a Specific Size

TikZ Library `ext.scalepicture`

```
\usetikzlibrary{ext.scalepicture} % LATEX and plain TEX
\usetikzlibrary[ext.scalepicture] % ConTEXt
```

This library scales TikZ pictures to a specific width or height by scaling the whole picture.

If one of the keys below are used on a TikZ picture, i.e. as an option to `\tikzpicture` or `\begin{tikzpicture}` the size of the picture⁶ 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.

14.1 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>*.

`/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>*.

14.2 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.

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

`/tikz/minimum picture width*= $\langle dimension \rangle$` (no default)

As above but will not change the size of the picture if its width is greater than $\langle dimension \rangle$.

`/tikz/maximum picture width*= $\langle dimension \rangle$` (no default)

As above but will not change the size of the picture if its width is less than $\langle dimension \rangle$.

`/tikz/picture height*= $\langle dimension \rangle$` (no default)

Scales the picture so that the height of the picture will be $\langle dimension \rangle$. This will only scale the y axis.

`/tikz/picture size*={ $\langle width \rangle$ }{ $\langle height \rangle$ }` (no default)

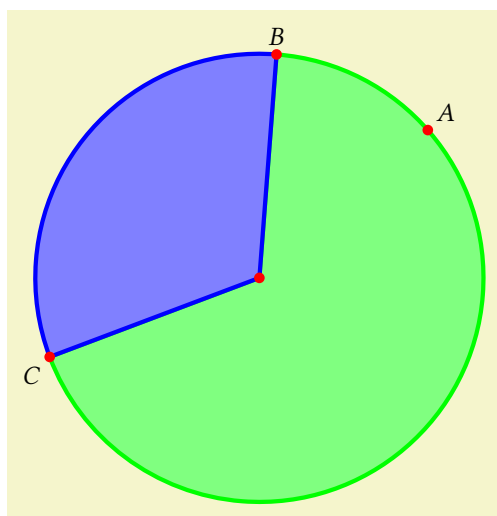
Scales the picture so that its width will be $\langle width \rangle$ and its height will be $\langle height \rangle$. This will scale both axes but independent from each other.

15 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 be 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.

16 Arcs through Three Points

TikZ Library `ext.topaths.autobend`

```
\usetikzlibrary{ext.topaths.autobend} % LATEX and plain TEX  
\usetikzlibrary[ext.topaths.autobend] % ConTEXt
```

This library provides various bended to paths that bend in the specified direction.

Q & A: [22] & [32]

The keys `/tikz/bend left` and `/tikz/bend right` from TikZ bend the requested curve in relation of the connecting coordinates/nodes.

The keys provided by this library bend the curve in the direction relative to the paper (north, south, west and east) or relative to the current coordinate system (up, down, left and right).

`/tikz/autobend north=angle` (default last value)

Works like the `bend left` and `bend right` options but bends the curve to the top of the page (i. e. it ignores the current transformation).

`/tikz/autobend south=angle` (default last value)

Works like `autobend north` but bends the curve to the bottom of the page.

`/tikz/autobend west=angle` (default last value)

Works like `autobend north` but bends the curve to the left of the page.

`/tikz/autobend east=angle` (default last value)

Works like `autobend north` but bends the curve to the right of the page.

`/tikz/autobend up=angle` (default last value)

Works like the `bend left` and `bend right` options but bends the curve upwards (i. e. it observes the current transformation).

`/tikz/autobend down=angle` (default last value)

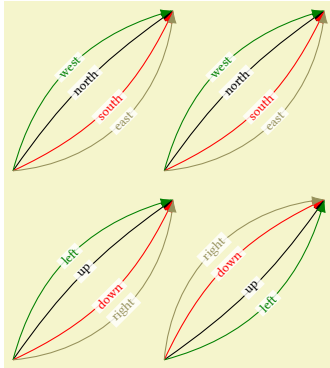
Works like `autobend up` but bends the curve downwards.

`/tikz/autobend left=angle` (default last value)

Works like `autobend up` but bends the curve leftwards.

`/tikz/autobend right=angle` (default last value)

Works like `autobend up` but bends the curve rightwards.



```
\usetikzlibrary {arrows.meta, ext.topaths.autobend}
\begin{tikzpicture}[
  every path/.append style=-Latex,
  nodes={sloped, fill=white, inner ysep=+.1em, fill opacity=.8, text opacity=1, scale=.5}]
\foreach[count=\i] \c/\d in {black/north, red/south,
  green!50!black/west, yellow!50!black/east}
  \draw[\c] (0,0) to[autobend \d=\i\theta] node{\d} +(45:3);
\foreach[count=\i] \c/\d in {black/north, red/south,
  green!50!black/west, yellow!50!black/east}
  \draw[shift=(right:2), rotate=180, \c]
    (0,0) to[autobend \d=\i\theta] node{\d} +(45:-3);

\tikzset{shift=(down:2.5)}
\foreach[count=\i] \c/\d in {black/up, red/down,
  green!50!black/left, yellow!50!black/right}
  \draw[\c] (0,0) to[autobend \d=\i\theta] node{\d} +(45:3);
\foreach[count=\i] \c/\d in {black/up, red/down,
  green!50!black/left, yellow!50!black/right}
  \draw[shift=(right:2), rotate=180, \c]
    (0,0) to[autobend \d=\i\theta] node{\d} +(45:-3);
\end{tikzpicture}
```

17 Mirror, Mirror on the Wall

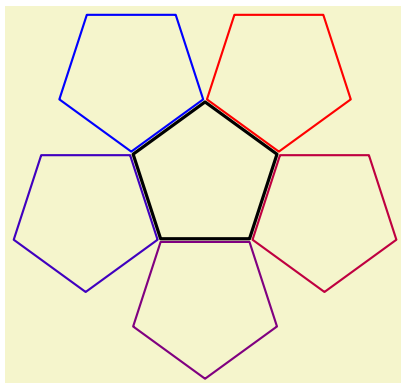
TikZ Library `ext.transformations.mirror`

```
\usetikzlibrary{ext.transformations.mirror} % LATEX and plain TEX
\usetikzlibrary[ext.transformations.mirror] % ConTEXt
```

This library adds more transformations to TikZ.

As explained in section 19, 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.

17.1 Using the reflection matrix

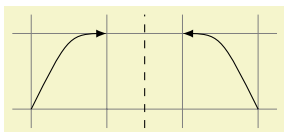


```
\usetikzlibrary {shapes.geometric,ext.transformations.mirror}
\begin{tikzpicture}[line join=round, thick, reg poly/.style={
  shape=regular polygon, regular polygon sides={#1}}]
\node[reg poly=5, minimum size=+2cm, draw, very thick] (a) {};
\foreach \i[evaluate={\col=(\i-1)/.04}] in {1,...,5}
  \node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

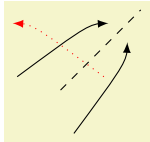
(default 0pt)

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

`/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.

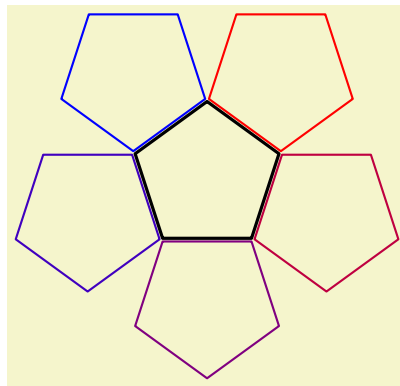
`/tikz/mirror=<point A>--<point B>`

(no default)

Sets up a transformation that mirrors along a line that goes through $\langle point A \rangle$ and $\langle point B \rangle$.

When only $\langle point A \rangle$ is given that line goes through $\langle point A \rangle$ and the origin.

17.2 Using built-in transformations

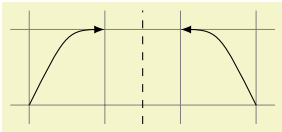


```
\usetikzlibrary {shapes.geometric,ext.transformations.mirror}
\begin{tikzpicture}[line join=round, thick, reg poly/.style={
  shape=regular polygon, regular polygon sides={#1}}]
\node[reg poly=5, minimum size=+2cm, draw, very thick] (a) {};
\foreach \i[evaluate={\col=(\i-1)/.04}] in {1,...,5}
  \node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

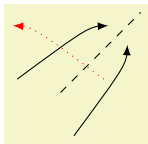
(default 0pt)

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)

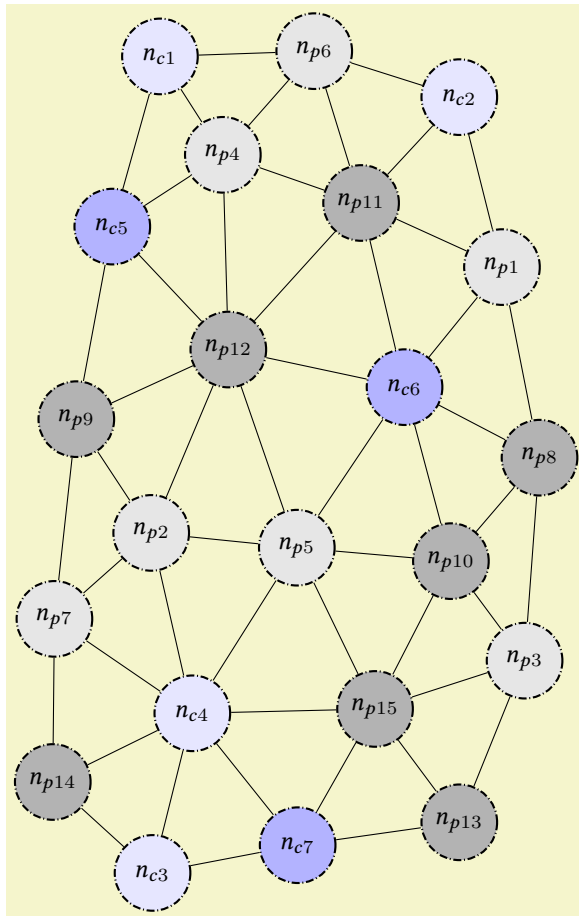
Sets up a transformation that mirrors along a line that goes through *<point A>* and *<point B>*.

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 TikZ.



```
\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}{blue!light}{blue!dark}}
      {/utils/TeX/ifnum={#2<8}{light}{dark}}},
    light/.style={fill=gray!20}, blue!light/.style={fill=blue!10},
    dark/.style={fill=gray!60}, blue!dark/.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};
};
```

18 Arrow Tips

TikZ Library `ext.arrows`

```
\usepgflibrary{ext.arrows} % LATEX and plain TEX and pure pgf
\usepgflibrary[ext.arrows] % ConTEXt and pure pgf
\usetikzlibrary{ext.arrows} % LATEX and plain TEX when using TikZ
\usetikzlibrary[ext.arrows] % ConTEXt when using TikZ
```

This library adds arrows to PGF/TikZ.

























Q & A: [15, 9, 4, 13] & [43, 30, 48, 41]

The arrow tips of the `arrows.meta` library always just touch the end of original line – which is usually what you want.

But for some arrow tips (and when they lie along a path) it makes sense that these tips shoot a bit over the end of the line. This is why these arrow tips exist. They can be categorized into three groups:

1. Centered
2. Untipped
3. Overtipped⁷

Not all original arrow tips got all variants. For a summary, refer to table on the right side. As with the original tips of the `arrows.meta` library these can be organized in the following categories.

Group	Original	Centered	Untipped	Overtipped
Barbed	Arc Barb			–
	Parenthesis			–
	Hooks		–	–
	Straight Barb		–	–
	Tee Barb			–
	Bar			–
	Bracket			–
Geometric	Circle			–
	Ellipse			–
	Kite		–	–
	Diamond		–	–
	Turned Square		–	–
	LaTeX	–	–	–
	Square		–	–
	Rectangle		–	–
	Stealth		–	–
	Triangle		–	–
Rays	Rays		–	–

⁷The Overtipped arrow tips aren't yet implemented.

18.1 Centered

18.1.1 Barbed Arrow Tips

Arrow Tip Kind Centered Arc Barb

This is a variant of the Arc Barb tip. The center of the arc lies on the original end of the path.

Arrow Tip Kind Centered Bar

A variant of the simple Bar tip. This is a simple instance of Centered Tee Barb for length zero.

The middle of the line will lie on original end of the path.

Arrow Tip Kind Centered Bracket

This is a variant of the Bracket tip and therefore an instance of the Centered Tee Barb arrow tip that results in something resembling a bracket.

The middle of the vertical part will lie on the original end of the path.

Arrow Tip Kind Centered Hooks

A variant of the Hooks tip. The starting point of the hooks will lie on the original end of the path.

Arrow Tip Kind Centered Parenthesis

This is a variant of the Parenthesis tip and thus an instance of the Centered Arc Barb arrow tip.

Arrow Tip Kind Centered Straight Barb

A variant of the Straight Barb tip.

Arrow Tip Kind Centered Tee Barb

A variant of the Tee Barb tip.

The middle of the vertical part will lie on the original end of the path.

18.1.2 Geometric Arrow Tips

Arrow Tip Kind Centered Circle

A variant of the Circle tip. The center of the circle will lie on the original end of the path.

Arrow Tip Kind Centered Diamond

This is a variant of the Diamond tip and thus an instance of Centered Kite where the length is larger than the width.

Arrow Tip Kind Centered Ellipse

This is a variant of the Ellipse tip and thus another name for the Centered Circle tip that is twice as wide as high.

Arrow Tip Kind Centered Kite

A variant of the Kite tip.

The widest part will lie on the original end of the path.

Arrow Tip Kind Centered Rectangle

A variant of the Rectangle tip. By default, it is twice as long as high.

Arrow Tip Kind Centered Square

A variant of the Square tip.

Arrow Tip Kind Centered Stealth

This is a variant of the Stealth tip.

The weighted center will lie at the original end of the path.

Arrow Tip Kind Centered Triangle

This is a variant of the Triangle tip and thus an instance of the Centered Kite tip with zero inset.

Arrow Tip Kind Centered Turned Square

This is a variant of the Turned Square tip and thus an instance of the Centered Kite tip with identical width and height and mid-inset.

18.1.3 Special Arrow Tips

Arrow Tip Kind Centered Rays

A variant of the Rays tip. The origin of the rays will lie on the original end of the path.

18.2 Untipped

18.2.1 Barbed Arrow Tips

Arrow Tip Kind Centered Arc Barb

This is a variant of the Arc Barb tip. The arrow tip will protrude half its line width over the original end of the path.

Arrow Tip Kind Untipped Bar

A variant of the simple Bar tip. This is a simple instance of Untipped Tee Barb for length zero.

The middle of the line will lie on original end of the path.

Arrow Tip Kind Untipped Bracket

This is a variant of the Bracket tip and therefore an instance of the Untipped Tee Barb arrow tip that results in something resembling a bracket.

The arrow tip will protrude half its line width over the original end of the path.

Arrow Tip Kind Untipped Parenthesis

This is a variant of the Parenthesis tip and thus an instance of the Untipped Arc Barb arrow tip.

Arrow Tip Kind Untipped Tee Barb

A variant of the Tee Barb tip.

The middle of the vertical part will lie on the original end of the path.

18.2.2 Geometric Arrow Tips

Arrow Tip Kind Untipped Circle

A variant of the Circle tip. This tip will protrude half its line width over the original end of the path.

Arrow Tip Kind Untipped Ellipse

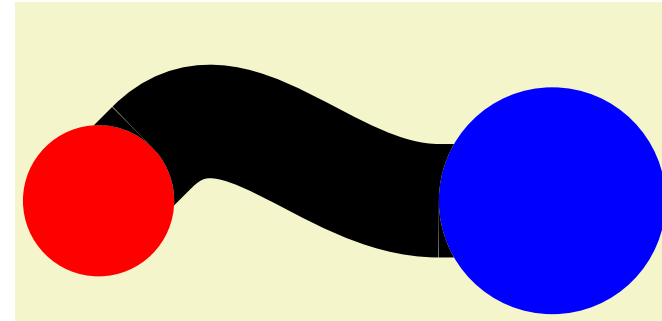
This is a variant of the Ellipse tip and thus another name for the Untipped Circle tip that is twice as wide as high.

18.3 Original Arrow Tips

Arrow Tip Kind Hug Cap

This arrow tip will hug a circle that would touch the end of the path.

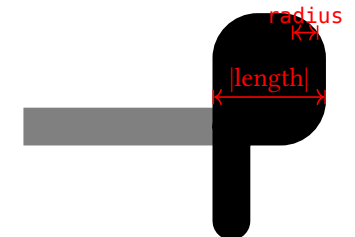
Use the `/pgf/arrow keys/length` key to set up the radius of that circle.



```
\usepgflibrary {ext.arrows}
\begin{tikzpicture}[
  dot/.style 2 args={
    shape=circle, outer sep=+0pt, fill={#1}, minimum size={#2}}]
\node[dot={red} {2cm}] (A) {};
\node[dot={blue}{3cm}] (B) at (6,0) {};
\draw[
  line width=1.5cm,
  arrows={Hug Cap[length=1cm]-Hug Cap[length=1.5cm]}
] (A) to[out=45, in=180] (B);
\end{tikzpicture}
```

Arrow Tip Kind Loop

This arrow tip attaches a one-sided loop to the end of the line. The length refers to the length of the whole tip while the radius specifies the radius of the three rounded corners. The width of the tip is twice the length (but can't specified independently).



Appearance of the below at line width

Loop[]

Loop[sep] Loop[]

0.4pt

—● thin

—●● thin

0.8pt

—● thick

—●● thick

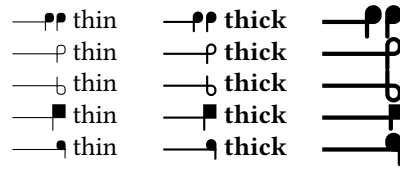
1.6pt

—●●●

```

Loop[sep] . Loop[]
Loop[open]
Loop[open, swap]
Loop[length=5pt, radius=0pt]
Loop[reversed]

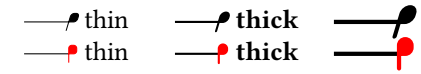
```



```

Loop[slant=.3]
Loop[red]

```



The following options have no effect: harpoon, round, line width.
On double lines, the arrow tip will not look correct.

19 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.

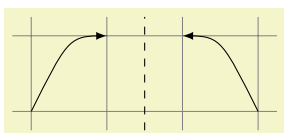
19.1 Using the reflection matrix

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

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

`\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] (-.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}
```

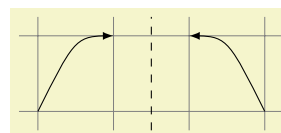
19.2 Using built-in transformations

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] (-.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}

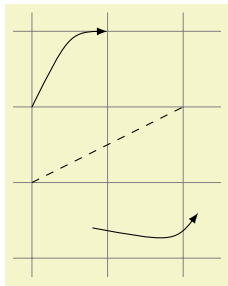
\draw[latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```


`\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 $\langle point A \rangle$ and $\langle point B \rangle$.



```
\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}}

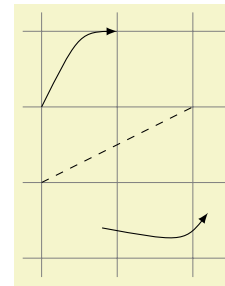
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`\pgftransformyMirror{⟨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 $\langle point A \rangle$ and $\langle point B \rangle$.



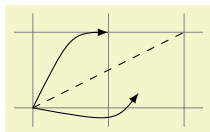
```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-.25, -2.25) grid (2.5, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

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

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

`\pgfqtransformmirror{⟨point A⟩}`

Sets up a transformation that mirrors along the line that goes through the origin and $\langle point A \rangle$.



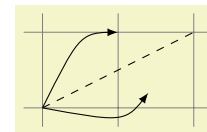
```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-.25, -.25) grid (2.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

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

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

`\pgfqtransformMirror{⟨point A⟩}`

Sets up a transformation that mirrors along the line that goes through the origin and $\langle point A \rangle$.



```
\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}
```

20 Shape: Circle Arrow

TikZ Library `ext.shapes.circulararrow`

```
\usepgflibrary{ext.shapes.circulararrow} % LATEX and plain TEX and pure pgf
\usepgflibrary[ext.shapes.circulararrow] % ConTEXt and pure pgf
\usetikzlibrary{ext.shapes.circulararrow} % LATEX and plain TEX when using TikZ
\usetikzlibrary[ext.shapes.circulararrow] % 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: [26] & [45]

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`

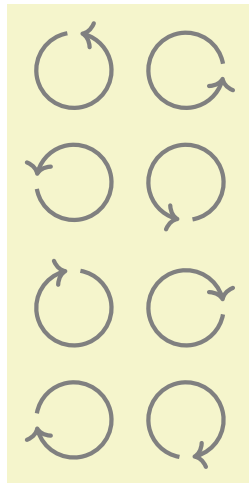
(no value)

As above but circle arrow start angle = 280.

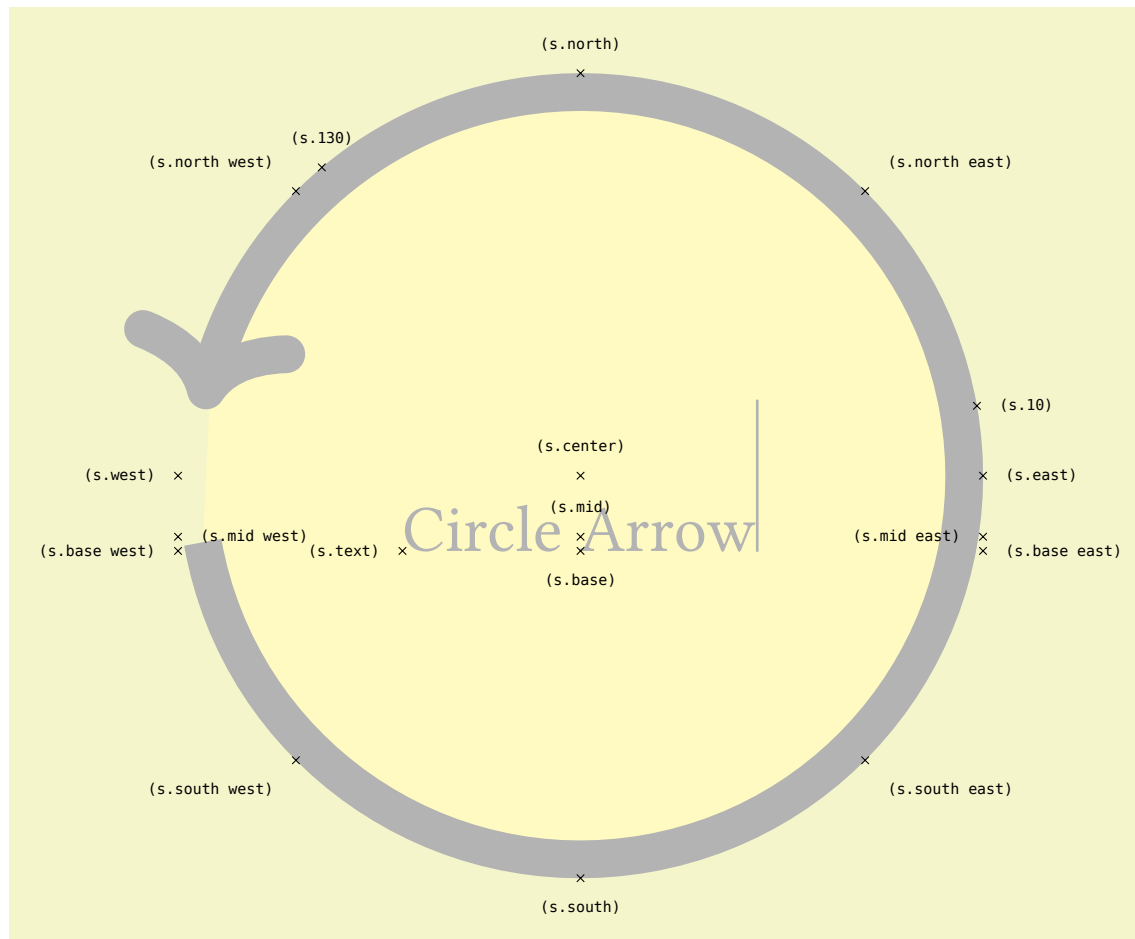
`/pgf/circle arrow turn right south`

(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.circulararrow}
\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}
```

21 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: [19] & [46]

Shape `circle cross split`

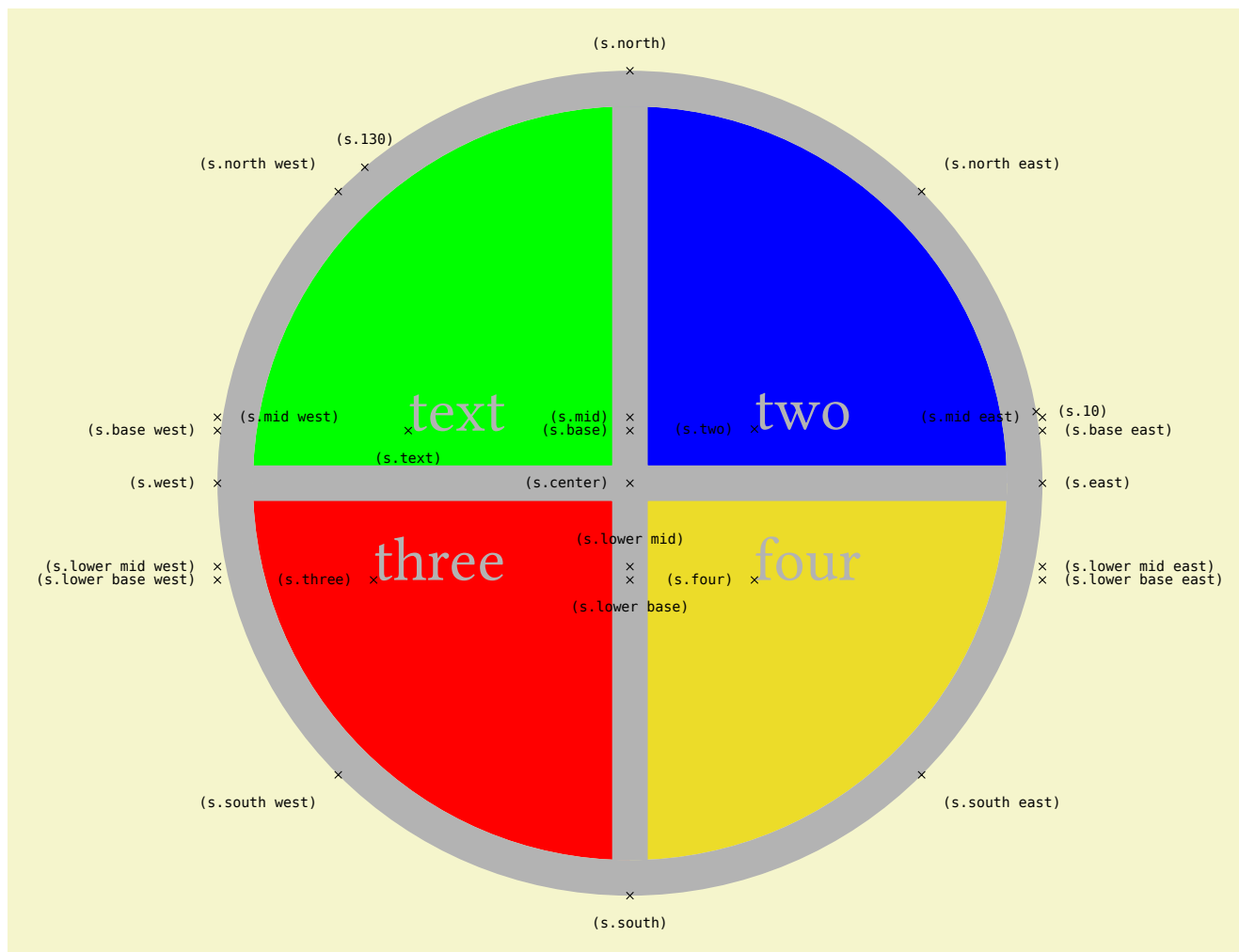
This shape has four node parts that are placed near the center of a circle.

`/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 T_EX-if `\ifpgfcirclecrosssplitcustomfill` appropriately.



```

\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}

```

22 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: [3] & [36]

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)

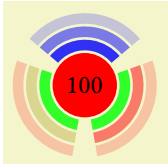
`/pgf/heatmark arc <arc number>`

(style, no value)

`/pgf/heatmark ring <ring number> arc <arc number>`

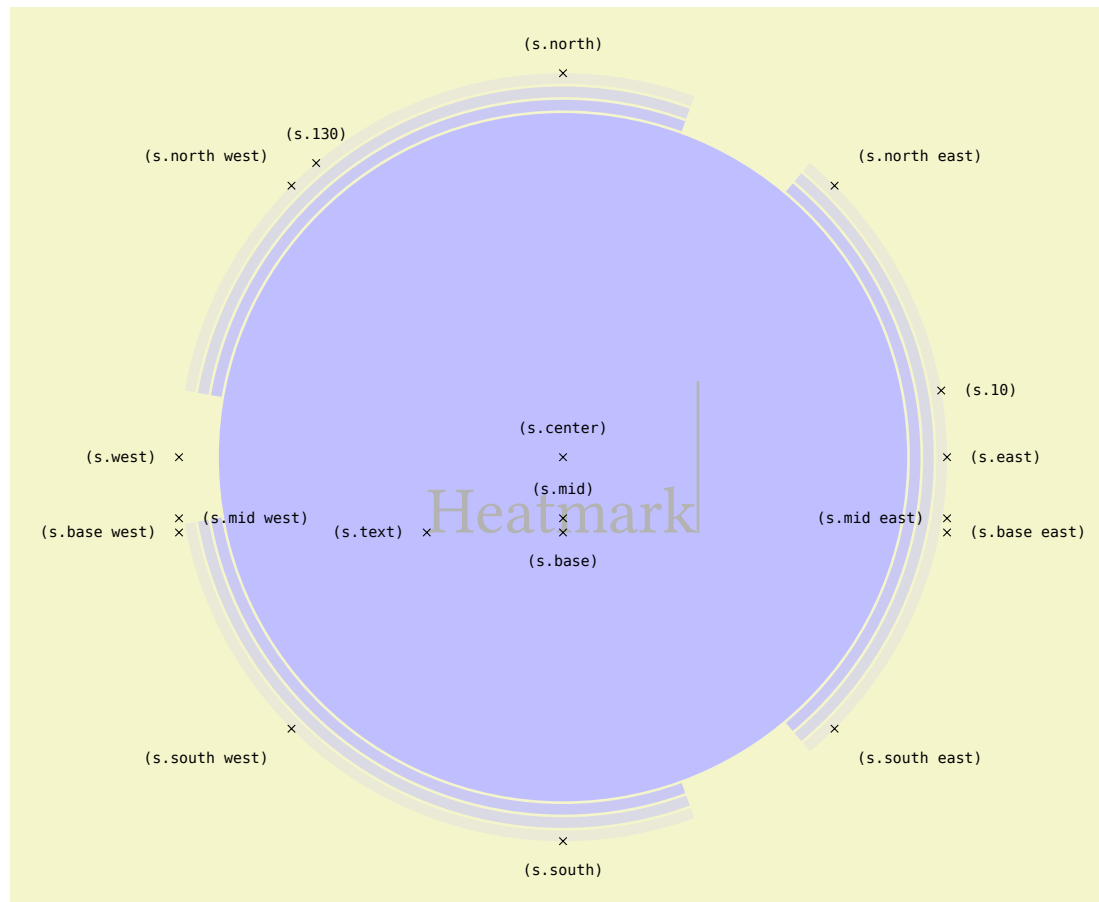
(style, no value)

The `pgfshape` is setup in a way that even `TikZ` 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}
```

23 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`)

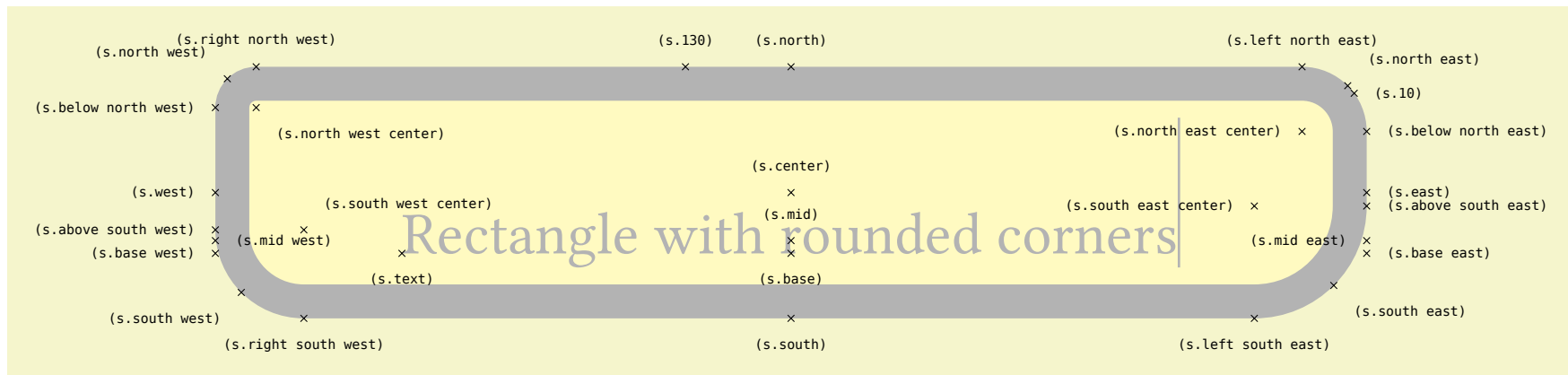
Sets the south west radius to `<dimen>`.

`/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}
```

24 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: [52] & [29]

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 $\langle step \rangle$ is, the slower computation will be.

Sensible values for $\langle step \rangle$ 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 $\langle exponent \rangle$.

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

```
superellipsex(t,  $2/m$ ,  $r_x$ )  
\pgfmathsuperellipse{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,  $2/n$ ,  $r_y$ )  
\pgfmathsuperellipse{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{t}{ $2/m$ }{ $2/n$ }{a}{b}
```

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{macro 1}{macro 2}
```

Adds the once-expansion of *macro 2* in front of *macro 1*.

25 Shape: Uncentered Rectangle

PGF Library `ext.shapes.uncenteredrectangle`

```
\usepgflibrary{ext.shapes.uncenteredrectangle} % LATEX and plain TEX  
\usepgflibrary[ext.shapes.uncenteredrectangle] % ConTEXt
```

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

Q & A: [55, 25] & [37, 34]

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.

`/pgf/uncentered rectangle use saved center=<true> or <false>` (default `true`)

When this is set to `true`, the border anchors will use the horizontal center that was used when the node was created.

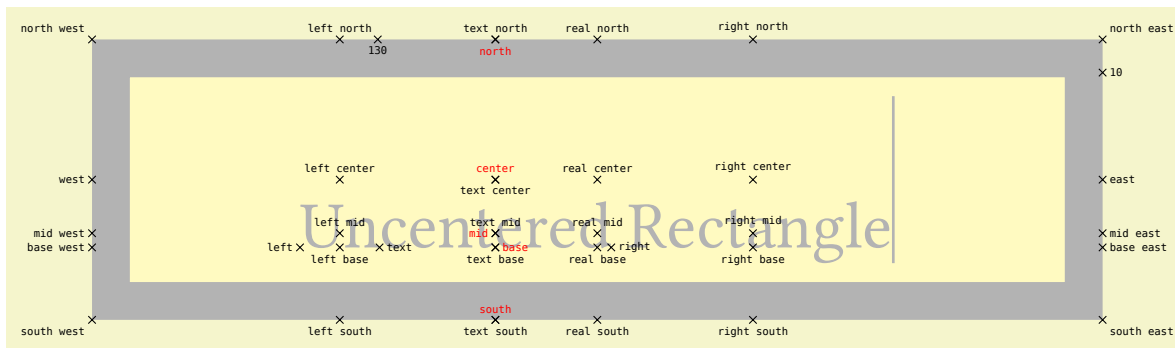
For support of the `cd` library of the `tikz-cd` package, this shape also supports a dynamic `y` value for its anchors `center`, `west` and `east`.

`/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 vertical center will be used.

For use with `cd`, set this to `axis_height`.



```
\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}
```

TikZ Library `ext.shapes.uncenteredrectangle`

`\usetikzlibrary{ext.shapes.uncenteredrectangle}` % \LaTeX and plain \TeX

`\usetikzlibrary[ext.shapes.uncenteredrectangle]` % \ConTeXt

This library extends the `cd` library (from the `tikz-cd` package) so that it can be used with the `uncentered rectangle` shape.

Q: [53]

This library provides only one key.

`/tikz-ext/tikz-cd fix`

(style, no value)

This key installs various “fixes” to the `/tikz/commutative diagrams/every diagram` style:

- Firstly, it defines a `/tikz/matrix of math nodes` key (only for the `tikzcd` environment) which allows to toggle the `/tikz/commutative diagrams/math mode` for

each node.⁸

- The helpful macro `\uncrec` will be installed.

`\uncrec{<left>}{<center>}{<right>}`

When used as the content of `uncentered rectangle`, the node parts will be setup so that `<left>` is in the left part of the node part etc.

- Since math mode will be disabled with the `uncentered rectangle`, it is automatically enabled for each node part with `\uncrec` but it can be disabled with the following key.

`/tikz/uncrec math mode=<true> or <false>` (default true)

When enabled the contents of `\uncrec` will be set in math mode.

- For easy access to the `uncentered rectangle` shape, the following keys are available inside a Commutative Diagram.

`/tikz/uncrec=<left> or <text> or <right> or <real>` (style, no default, initially text)

This key sets the shape to `uncentered rectangle` and `/pgf/uncentered rectangle center` to its argument.

`/tikz/commutative diagrams/install uncentered rectangle in columns=<column>` (style, no default)

All nodes in column `<column>` will be set to the `uncentered rectangle` shape.

$$\begin{array}{ccc} C_{\%_1} & & m_{r_1} = C_{\%_2} - C_{\%} \\ & \swarrow \quad \searrow & \\ & C_{\%} & \\ & \swarrow \quad \searrow & \\ C_{\%_2} & & m_{r_2} = C_{\%_1} - C_{\%} \end{array}$$

```
\usetikzlibrary {cd, ext.shapes.uncenteredrectangle}
\tikzcdset{/tikz-ext/tikz-cd fix}
\newcommand*\C[1]{C_{\%_{#1}}}
\begin{tikzcd}[
  sep=tiny,
  arrows={-, gray},
  cells={font=\strut, inner xsep=.2ex, inner ysep=.1ex},
  install uncentered rectangle in column=3
]
\C{1} \drar & & \uncrec{m_{r_1}}{=} = \C{2}-C_{\%} \dlar \\
& & C_{\%} \\
\C{2} \urar & & \uncrec{m_{r_2}}{=} = \C{1}-C_{\%} \ular
\end{tikzcd}
```

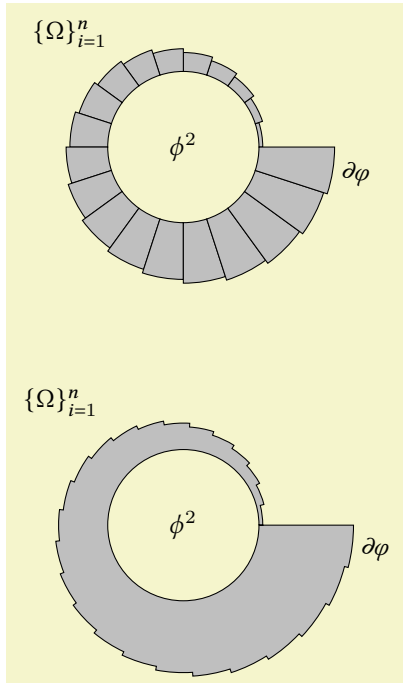
$$\begin{array}{ccc} S \supset U_{\tau} & \xrightarrow{\varphi_0} & U_{\pi} \subset T \\ \sim \downarrow \tau & & \sim \downarrow \pi \\ \mathrm{Bl}_{(0,0)}(\mathbb{A}^2) \supset V_{\tau} & \xrightarrow{\epsilon} & V_{\pi} \subset \mathbb{A}^2 \end{array}$$

```
\usetikzlibrary {cd, ext.shapes.uncenteredrectangle}
\tikzset{/tikz-ext/tikz-cd fix}
\begin{tikzcd}[install uncentered rectangle in column/.list={1,2}]
  \uncrec{S \supset {} }{U_{\tau}}{} \arrow[r, "\varphi_0"]
  & & \uncrec{U_{\pi} \subset T}{} \arrow[d, "\tau", "\sim"]
  \\
  \uncrec{\operatorname{Bl}_{(0,0)}(\mathbb{A}^2) \supset {} }{V_{\tau}}{} \arrow[r, "\epsilon"]
  & & \uncrec{V_{\pi} \subset \mathbb{A}^2}{} \arrow[d, "\pi", "\sim"]
\end{tikzcd}
```

⁸Due to a bug with `/tikz/execute at end node`, the “automatic” math mode in matrices can’t be used with multipart nodes.

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=+IR] -- (\iN R+1R:smallR)
  arc [radius=smallR, end angle=\iN R, delta angle=-IR] -- 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) {${\partial\phi}$};

\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=IR]}
    -- (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) {${\partial\phi}$};
\end{tikzpicture}
```

26 Calendar: Weeknumbers and more conditionals

```
\usepackage{pgfcalendar-ext} % LATEX
\input pgfcalendar-ext.tex % plain TEX
```

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

Q & A: [11, 12, 16] & [28, 49, 35]

26.1 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**= $\langle year \rangle$ This test checks whether the given year is a leap year. If $\langle year \rangle$ is omitted, it checks the year of the current date.
- **and**= $\{\langle tests \rangle\}$ This test passes when all $\langle tests \rangle$ pass.
- **not**= $\{\langle tests \rangle\}$ This test passes when $\langle tests \rangle$ do not pass.
- **week of month**= $\langle num \rangle$ This test passes when the date is in $\langle num \rangle$ th week of the month. The first week of the month start at day 1 and ends with day 7.

- **week of month**'= $\langle num \rangle$ 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**= $\langle num \rangle$ This test passes when the date is in $\langle num \rangle$ 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**'= $\langle num \rangle$ As above but counts from the last day of the month.

<pre> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 </pre>	<pre> \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]; </pre>
--	--

- **yesterday**= $\{\langle tests \rangle\}$ This test passes when the previous day passes $\langle tests \rangle$.
- **week**= $\langle num \rangle$ This test passes when the current week of the year equals $\{\langle num \rangle\}$.

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 26.2) are added. These are

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

26.2 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).

27 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: [2, 8, 56] & [38, 44, 40]

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 Lua_T_EX isn't used. Spaces will be ignored.

B	X	X
-	-	-
W	X	X

```
\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 plain \TeX `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 be 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 Om` (no value)

Sets up a list whose elements may contain an optional argument 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.

28 And a little bit more

TikZ Library `ext.misc`

```
\usetikzlibrary{ext.misc} % LATEX and plain TEX
\usetikzlibrary{ext.misc} % ConTEXt
```

This library adds miscellaneous utilities to PGFmath, PGF or TikZ.

Q & A: [24] & [27]

28.1 PGFmath

28.1.1 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.

`xR` (postfix operator; uses the `fullarc` function)

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

28.1.2 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 `\pgfmath` commands of the `calc` library.

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

`atanYX(y,x)`

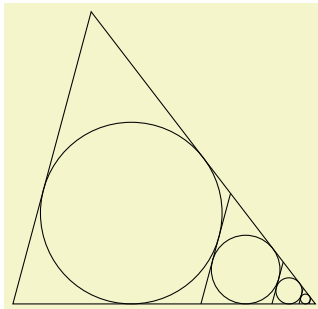
`\pgfmathatanYX{y}{x}`

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

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

28.1.3 Functions: using coordinates

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



```
\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}
```

`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*.

28.2 PGFfor

This library loads also most of the functions of the `pgffor-ext` of section 27 on page 70.

28.3 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.⁹

28.3.1 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 T_EX' 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/isempty=⟨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⟩`.

28.3.2 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⟩`

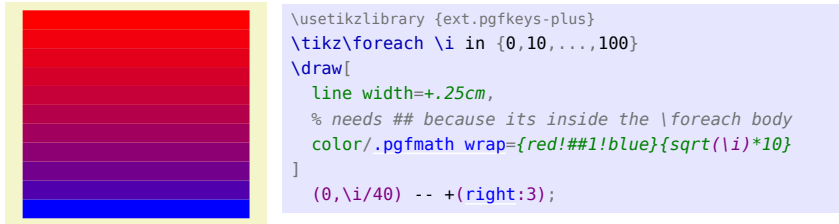
As above but truncates the result.

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.

⁹`\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.



Key handler $\langle key \rangle /.pgfmath\ if=\{\langle cond \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

Evaluates $\langle cond \rangle$ with PGFMath and returns $\langle true \rangle$ or $\langle false \rangle$ to the used key respectively.

Key handler $\langle key \rangle /.if=\langle token A \rangle \langle token B \rangle \{\langle true \rangle\}\{\langle false \rangle\}$

Checks via $\backslash if$ if $\langle token A \rangle$ matches $\langle token B \rangle$ and applies the value $\langle true \rangle$ if it does, otherwise $\langle false \rangle$.

Key handler $\langle key \rangle /.ifx=\langle token A \rangle \langle token B \rangle \{\langle true \rangle\}\{\langle false \rangle\}$

As above but via $\backslash ifx$.

Key handler $\langle key \rangle /.ifnum=\{\langle ifnum cond \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

Checks via $\backslash ifnum$ if $\langle ifnum cond \rangle$ and applies the value $\langle true \rangle$ if it does, otherwise $\langle false \rangle$.

Key handler $\langle key \rangle /.ifdim=\{\langle ifdim cond \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

As above but via $\backslash ifdim$.

Key handler $\langle key \rangle /.ifxempty=\{\langle Text \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

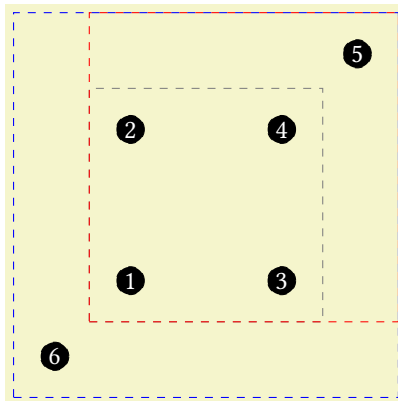
Checks whether a fully expanded $\langle Text \rangle$ is empty and applies the value $\langle true \rangle$ if it does, otherwise $\langle false \rangle$.

Key handler $\langle key \rangle /.ifempty=\{\langle Text \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

Checks whether $\langle Text \rangle$ is empty and applies the value $\langle true \rangle$ if it does, otherwise $\langle false \rangle$.

Key handler $\langle key \rangle /.List=\{\langle e1 \rangle, \langle e2 \rangle, ..., \langle en \rangle\}$

This handler evaluates the given list with $\backslash foreach$ 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}

```

28.4 TikZ

`/tikz/reverse clip=<direction>`

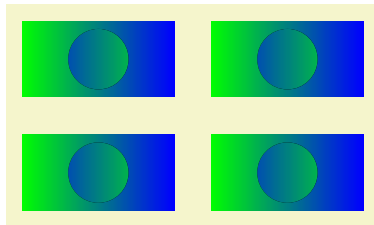
(default counter clockwise)

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 directly¹⁰ 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}
```

¹⁰Meaning, it directly executes `\pgfseteorule` `/\pgfsetnonzerorule` and doesn't accumulate where TikZ throws an error.

Part V

Changelog, Index & References

Changelog

Version 0.6

- Added six new auto placement mechanism: above, below, west, east, north and south.
- Added auto offset for auto placement.
- Added precise auto angle.
- Added TikZ library `ext.arrows-plus`.

Version 0.5.1

- Added PGF library `ext.arrows`.
- Bugfix to `ext.pgfkeys-plus`. [21]

Version 0.5

- Added package `pgffor-ext`.
- Added TikZ library `ext.nodes`.
- Added TikZ library `ext.layers`.
- Bugfixes to `ext.calendar-plus`.
- Allow the original rectangle timer with `ext.paths.timer`.

Version 0.4.2

- Added TikZ 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 [20].

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 TikZ 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 TikZ library `ext.positioning-plus`.
- Added TikZ library `ext.node-families`.

Version 0.1

- Added TikZ library `ext.calendar-plus`.
- Added TikZ library `ext.misc`.
- Added TikZ library `ext.paths.arcto`.
- Added TikZ library `ext.paths.ortho`.
- Added TikZ library `ext.paths.timer`.
- Added TikZ library `ext.patterns.images`.
- Added TikZ library `ext.topaths.arctthrough`.
- Added TikZ 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/TikZ` and [references](#) to functionalities outside of `PGF/TikZ`.

- path operation, [8](#), [15](#)
- | - | path operation, [20](#)
- | - path operation, [20](#)
- | path operation, [20](#)
- | - path operation, [20](#)

- above key, [32](#)
- above left key, [30](#)
- above right key, [30](#)
- and date test, [68](#)
- anglebetween math function, [73](#)
- Apr date test, [68](#)
- arc path operation, [8](#), [50](#)
- arc arrows key, [8](#), [9](#)
- Arc Barb arrow tip, [45](#), [46](#)
- arc through key, [37](#)
- arc to path operation, [18](#)
- \arrow, [9](#)
- arrow pic type, [8](#)
- arrow shift factor key, [7](#), [9](#)
- arrow shift mode key, [7](#), [9](#)
- Arrow tips
 - Arc Barb, [45](#), [46](#)
 - Bar, [45](#), [46](#)
 - Bracket, [45](#), [46](#)
 - Centered Arc Barb, [45](#), [46](#)
 - Centered Bar, [45](#)
 - Centered Bracket, [45](#)
 - Centered Circle, [45](#)
 - Centered Diamond, [45](#)
 - Centered Ellipse, [45](#)
 - Centered Hooks, [45](#)
 - Centered Kite, [45](#)
 - Centered Parenthesis, [45](#)
 - Centered Rays, [45](#)
 - Centered Rectangle, [45](#)
 - Centered Square, [45](#)
 - Centered Stealth, [45](#)
 - Centered Straight Barb, [45](#)
 - Centered Tee Barb, [45](#)
 - Centered Triangle, [45](#)
 - Centered Turned Square, [45](#)
 - Circle, [45](#), [46](#)
 - Diamond, [45](#)
 - Ellipse, [45](#), [46](#)
 - Hooks, [45](#)
 - Hug Cap, [46](#)
 - Kite, [45](#)
 - Loop, [46](#)
 - Parenthesis, [45](#), [46](#)
 - Rays, [45](#)
 - Rectangle, [45](#)
 - Square, [45](#)
 - Stealth, [45](#)
 - Straight Barb, [45](#)
 - Tee Barb, [45](#), [46](#)
 - Triangle, [45](#)
 - Turned Square, [45](#)
 - Untipped Bar, [46](#)
 - Untipped Bracket, [46](#)
 - Untipped Circle, [46](#)
 - Untipped Ellipse, [46](#)
 - Untipped Parenthesis, [46](#)
 - Untipped Tee Barb, [46](#)
- \arrowreversed, [9](#)
- `ext.arrows` library, [7](#)
- `arrows.meta` library, [44](#)
- atan2 math function, [48](#), [73](#)
- atanXY math function, [72](#)
- atanYX math function, [73](#)
- Aug date test, [68](#)

auto key, 16
auto offset key, 16, 17
auto offset for brace decoration key, 17
auto with offset key, 16
autobend down key, 38
autobend east key, 38
autobend left key, 38
autobend north key, 38
autobend right key, 38
autobend south key, 38
autobend up key, 38
autobend west key, 38

background code key, 11
Bar arrow tip, 45, 46
below key, 32
below left key, 30
below right key, 30
bend left key, 38
brace decoration, 16
Bracket arrow tip, 45, 46

calc library, 73
calendar library, 10
calendar week of month date test, 68
calendar week of month' date test, 68
cd library, 64, 65
center suffix key, 37
Centered Arc Barb arrow tip, 45, 46
Centered Bar arrow tip, 45
Centered Bracket arrow tip, 45
Centered Circle arrow tip, 45
Centered Diamond arrow tip, 45
Centered Ellipse arrow tip, 45
Centered Hooks arrow tip, 45
Centered Kite arrow tip, 45
Centered Parenthesis arrow tip, 45
Centered Rays arrow tip, 45
Centered Rectangle arrow tip, 45
Centered Square arrow tip, 45
Centered Stealth arrow tip, 45

Centered Straight Barb arrow tip, 45
Centered Tee Barb arrow tip, 45
Centered Triangle arrow tip, 45
Centered Turned Square arrow tip, 45
Circle arrow tip, 45, 46
circle path operation, 8, 76
circle shape, 12
circle arrow shape, 50
circle arrow arrows key, 50
circle arrow delta angle key, 50
circle arrow end angle key, 50
circle arrow start angle key, 50
circle arrow turn left east key, 50
circle arrow turn left north key, 50
circle arrow turn left south key, 50
circle arrow turn left west key, 50
circle arrow turn right east key, 51
circle arrow turn right north key, 51
circle arrow turn right south key, 51
circle arrow turn right west key, 51
circle cross split shape, 53
circle cross split part fill key, 53
circle cross split uses custom fill key, 53
clip rule key, 76
clockwise key, 18, 37
code key, 11
corner above left key, 30
corner above right key, 30
corner below left key, 30
corner below right key, 30
corner east above key, 33
corner east below key, 33
corner north left key, 33
corner north right key, 33
corner south left key, 33
corner south right key, 33
corner west above key, 33
corner west below key, 33
cos math function, 48
cos path operation, 8, 27
counter clockwise key, 19, 37

Date tests

- and, [68](#)
- Apr, [68](#)
- Aug, [68](#)
- calendar week of month, [68](#)
- calendar week of month', [68](#)
- Dec, [68](#)
- Feb, [68](#)
- Jan, [68](#)
- Jul, [68](#)
- Jun, [68](#)
- leap year, [68](#)
- Mar, [68](#)
- May, [68](#)
- not, [68](#)
- Nov, [68](#)
- Oct, [68](#)
- Sep, [68](#)
- week, [68](#)
- week of month, [68](#)
- week of month', [68](#)
- yesterday, [68](#)

day code key, [10](#)

day text key, [10](#)

day xshift key, [10](#)

day yshift key, [10](#)

Dec date test, [68](#)

decorationraise key, [17](#)

Decorations

- brace, [16](#)
- markings, [7](#)

decorations.markings library, [9](#)

decorations.pathreplacing library, [16](#)

Diamond arrow tip, [45](#)

distance key, [20](#)

distancebetween math function, [73](#)

down horizontal up key, [23](#)

du distance key, [22](#)

east above key, [32](#)

east below key, [32](#)

edge path operation, [37](#)

edge in box key, [11](#)

edge on layer key, [11](#)

Ellipse arrow tip, [45, 46](#)

ellipse path operation, [8](#)

Environments

- pgfonlayer, [11](#)
- tikzcd, [65](#)

evaluate key, [74](#)

.evaluated handler, [74](#)

even odd rule key, [76](#)

every arc to key, [19](#)

every brace node key, [17](#)

every day key, [10](#)

every diagram key, [65](#)

every month key, [10](#)

every softpath arrows key, [8, 9](#)

every week key, [10](#)

execute at end node key, [66](#)

execute at end picture key, [12](#)

ext.arrows library, [44](#)

ext.arrows-plus library, [7](#)

ext.calendar-plus library, [10](#)

ext.layers library, [11](#)

ext.misc library, [70–72](#)

ext.node-families library, [12](#)

ext.node-families.shapes.geometric library, [14](#)

ext.nodes library, [15](#)

ext.paths.arcto library, [18](#)

ext.paths.ortho library, [20](#)

ext.paths.timer library, [26](#)

ext.patterns.images library, [29](#)

ext.pgfkeys-plus pgfkeys library, [74](#)

ext.positioning-plus library, [30](#)

ext.scalepicture library, [35](#)

ext.shapes.circlearrow library, [50](#)

ext.shapes.circlecrosssplit library, [53](#)

ext.shapes.heatmark library, [56](#)

ext.shapes.rectangleroundedcorners library, [59](#)

ext.shapes.superellipse library, [61](#)

ext.shapes.uncenteredrectangle library, [64, 65](#)

ext.topaths.arctthrough library, 37
ext.topaths.autobend library, 38
ext.transformations.mirror library, 40, 48
external library, 12

Feb date test, 68
fit library, 30
fit bounding box key, 34
foreground code key, 11
from center key, 21
full arc key, 72

grid path operation, 8

heatmark shape, 56
heatmark arc *(arc number)* key, 57
heatmark arc rings key, 56
heatmark arc sep key, 56
heatmark arc sep angle key, 56
heatmark arc width key, 56
heatmark arcs key, 56
heatmark inner opacity key, 56
heatmark outer opacity key, 56
heatmark ring *(ring number)* key, 56
heatmark ring *(ring number)* arc *(arc number)* key, 57
height key, 13, 14
Hooks arrow tip, 45
horizontal vertical key, 23
horizontal vertical horizontal key, 23
Hug Cap arrow tip, 46

.if handler, 75
if key, 10, 74
\ifdate, 69
.ifdim handler, 75
ifdim key, 74
.ifempty handler, 75
ifempty key, 74
.ifnum handler, 75
ifnum key, 74
.ifx handler, 75
ifx key, 74

.ifxempty handler, 75
ifxempty key, 74
image as pattern key, 29
in box key, 11
install auto offset for brace decoration key, 17
install shortcuts key, 25
install uncentered rectangle in columns key, 66
intersections library, 16
isEmpty math function, 72
isInString math function, 72

Jan date test, 68
Jul date test, 68
Jun date test, 68

Key handlers

.List, 75
.evaluated, 74
.if, 75
.ifdim, 75
.ifempty, 75
.ifnum, 75
.ifx, 75
.ifxempty, 75
.list xparse, 71
.pgfmath, 74
.pgfmath if, 75
.pgfmath int, 74
.pgfmath wrap, 74

Kite arrow tip, 45

large key, 19
lastdayinmonthofyear math function, 10
leap year date test, 68
left key, 31, 32
left vertical right key, 23
length key, 46

Libraries

arrows.meta, 44
calc, 73
calendar, 10

- cd, [64](#), [65](#)
- decorations.markings, [9](#)
- decorations.pathreplacing, [16](#)
- ext.arrows, [7](#)
- ext.arrows, [44](#)
- ext.arrows-plus, [7](#)
- ext.calendar-plus, [10](#)
- ext.layers, [11](#)
- ext.misc, [70–72](#)
- ext.node-families, [12](#)
- ext.node-families.shapes.geometric, [14](#)
- ext.nodes, [15](#)
- ext.paths.arcto, [18](#)
- ext.paths.ortho, [20](#)
- ext.paths.timer, [8](#)
- ext.paths.timer, [26](#)
- ext.patterns.images, [29](#)
- ext.positioning-plus, [30](#)
- ext.scalepicture, [35](#)
- ext.shapes.circulararrow, [50](#)
- ext.shapes.circlexsplit, [53](#)
- ext.shapes.heatmark, [56](#)
- ext.shapes.rectangleroundedcorners, [59](#)
- ext.shapes.superellipse, [61](#)
- ext.shapes.uncenteredrectangle, [64](#), [65](#)
- ext.spath3, [16](#)
- ext.topaths.arctthrough, [37](#)
- ext.topaths.autobend, [38](#)
- ext.transformations.mirror, [40](#), [48](#)
- external, [12](#)
- fit, [30](#)
- intersections, [16](#)
- markings, [15](#)
- positioning, [30](#)
- shapes.geometric, [14](#)
- .List handler, [75](#)
- .list xparse handler, [71](#)
- Loop arrow tip, [46](#)
- lr distance key, [22](#)
- Mar date test, [68](#)

- mark connection node key, [15](#)
- markings decoration, [7](#)
- markings library, [15](#)
- Math functions
 - anglebetween, [73](#)
 - atan2, [48](#), [73](#)
 - atanXY, [72](#)
 - atanYX, [73](#)
 - cos, [48](#)
 - distancebetween, [73](#)
 - isEmpty, [72](#)
 - isInString, [72](#)
 - lastdayinmonthofyear, [10](#)
 - qanglebetween, [73](#)
 - qdistancebetween, [73](#)
 - sin, [48](#)
 - strcat, [72](#)
 - strrepeat, [72](#)
 - superellipse, [62](#)
 - superellipse, [62](#)
 - weeksmonthofyear, [10](#)
- math mode key, [65](#)
- Math operators
 - R, [72](#)
- matrix in box key, [11](#)
- matrix of math nodes key, [65](#)
- matrix on layer key, [11](#)
- maximum picture height key, [35](#)
- maximum picture size key, [35](#)
- maximum picture width key, [35](#)
- maximum picture width* key, [36](#)
- May date test, [68](#)
- middle 0 to 1 key, [22](#)
- minimum height key, [13](#), [34](#)
- minimum picture height key, [35](#)
- minimum picture size key, [35](#)
- minimum picture width key, [35](#)
- minimum picture width* key, [36](#)
- minimum width key, [13](#), [34](#)
- Mirror key, [42](#)
- mirror key, [41](#)

Mirror x key, 42
mirror x key, 41
Mirror y key, 42
mirror y key, 41
month code key, 10
month text key, 10
month xshift key, 10
month yshift key, 10

name key, 29
no separator key, 70
node in box key, 11
node on layer key, 11
node on line key, 15
nodes on curve key, 16
nodes on curve' key, 16
nodes on line key, 15, 16
nonzero rule key, 76
normal list key, 70
north left key, 31
north right key, 32
not date test, 68
Nov date test, 68

Oct date test, 68
on grid key, 34
on layer key, 11
only horizontal first key, 24
only horizontal second key, 23
only vertical first key, 24
only vertical second key, 23
option key, 29
options key, 29
outer sep key, 16

Packages and files

pgfcalendar-ext, 68
pgffor, 70
pgffor-ext, 70, 73
xparse-ext, 71
parabola path operation, 8, 27

Parenthesis arrow tip, 45, 46

patch key, 11

Path operations

--, 8, 15
|-|, 20
-|-, 20
-|, 20
|-, 20
arc, 8, 50
arc to, 18
circle, 8, 76
cos, 8, 27
edge, 37
ellipse, 8
grid, 8
parabola, 8, 27
plot, 8
r-du, 22
r-lr, 22
r-rl, 22
r-ud, 22
rectangle, 8, 26
sin, 8, 27
to, 37

ext.paths.timer library, 8

/pgf/

arrow keys/
length, 46
circle arrow arrows, 50
circle arrow delta angle, 50
circle arrow end angle, 50
circle arrow start angle, 50
circle arrow turn left east, 50
circle arrow turn left north, 50
circle arrow turn left south, 50
circle arrow turn left west, 50
circle arrow turn right east, 51
circle arrow turn right north, 51
circle arrow turn right south, 51
circle arrow turn right west, 51
circle cross split part fill, 53

- circle cross split uses custom fill, [53](#)
- decoration/
 - mark connection node, [15](#)
- decorationraise, [17](#)
- foreach/
 - evaluate, [74](#)
 - no separator, [70](#)
 - normal list, [70](#)
 - use float, [70](#)
 - use int, [70](#)
 - var, [71](#)
 - xparser, [71](#)
 - xparser 0m, [71](#)
- full arc, [72](#)
- heatmark arc *<arc number>*, [57](#)
- heatmark arc rings, [56](#)
- heatmark arc sep, [56](#)
- heatmark arc sep angle, [56](#)
- heatmark arc width, [56](#)
- heatmark arcs, [56](#)
- heatmark inner opacity, [56](#)
- heatmark outer opacity, [56](#)
- heatmark ring *<ring number>*, [56](#)
- heatmark ring *<ring number>* arc *<arc number>*, [57](#)
- minimum height, [13](#), [34](#)
- minimum width, [13](#), [34](#)
- outer sep, [16](#)
- rectangle with rounded corners north east radius, [59](#)
- rectangle with rounded corners north west radius, [59](#)
- rectangle with rounded corners radius, [59](#)
- rectangle with rounded corners south east radius, [59](#)
- rectangle with rounded corners south west radius, [59](#)
- shape border rotate, [56](#)
- superellipse exponent, [61](#)
- superellipse step, [61](#)
- superellipse x exponent, [61](#)
- superellipse y exponent, [61](#)
- text, [29](#)
- uncentered rectangle center, [64](#), [66](#)
- uncentered rectangle center yshift, [64](#)
- uncentered rectangle use saved center, [64](#)

- [\pgfcalendar](#), [69](#)
- pgfcalendar-ext package, [68](#)
- [\pgfcalendarcurrentweek](#), [69](#)
- [\pgfcalendarifdateweek](#), [69](#)
- [\pgfcalendarjulianyeartoweek](#), [69](#)
- pgffor package, [70](#)
- pgffor-ext package, [70](#), [73](#)
- pgfkeys Libraries
 - ext.pgfkeys-plus, [74](#)
- .pgfmath handler, [74](#)
- .pgfmath if handler, [75](#)
- .pgfmath int handler, [74](#)
- .pgfmath wrap handler, [74](#)
- [\pgfmathanglebetween](#), [73](#)
- [\pgfmathanglebetweenpoints](#), [48](#)
- [\pgfmathatanXY](#), [73](#)
- [\pgfmathatanYX](#), [73](#)
- [\pgfmathdistancebetween](#), [73](#)
- [\pgfmathisEmpty](#), [72](#)
- [\pgfmathisInString](#), [72](#)
- [\pgfmathlastdayinmonthofyear](#), [10](#)
- [\pgfmathqanglebetween](#), [73](#)
- [\pgfmathqdistancebetween](#), [73](#)
- [\pgfmathstrcat](#), [72](#)
- [\pgfmathstrrepeat](#), [72](#)
- [\pgfmathsuperellipseX](#), [62](#)
- [\pgfmathsuperellipseXY](#), [62](#)
- [\pgfmathsuperellipseY](#), [62](#)
- [\pgfmathweeksinsmonthofyear](#), [10](#)
- pgfonlayer environment, [11](#)
- [\pgfpatharcto](#), [19](#)
- [\pgfpointnormalised](#), [48](#)
- [\pgfqtransformMirror](#), [49](#)
- [\pgfqtransformmirror](#), [49](#)
- [\pgfsetarrows](#), [50](#)
- [\pgfseteorule](#), [76](#)
- [\pgfsetnonzerorule](#), [76](#)
- [\pgfsetupimageaspattern](#), [29](#)
- [\pgftext](#), [29](#)
- [\pgfttransformMirror](#), [49](#)
- [\pgfttransformmirror](#), [49](#)

- `\pgftransformxMirror`, 48
- `\pgftransformxmirror`, 48
- `\pgftransformyMirror`, 49
- `\pgftransformymirror`, 49
- `\pgfutil@prefix@macrotomacro`, 62
- `pic` key, 15
- `pic in box` key, 11
- `pic on layer` key, 11
- Pic Types
 - arrow, 8
 - softpath arrow, 8
 - softpath arrows, 8
- `picture height` key, 35
- `picture height*` key, 36
- `picture size*` key, 36
- `picture width` key, 35
- `picture width*` key, 35
- `plot path` operation, 8
- `pos` key, 21, 26
- `pos` key, 7
- `pos <` key, 7
- `pos >` key, 7
- positioning library, 30
- `precise auto angle` key, 17
- `prefix` key, 12
- `qanglebetween` math function, 73
- `qdistancebetween` math function, 73
- R postfix math operator, 72
- `r-du` path operation, 22
- `r-lr` path operation, 22
- `r-rl` path operation, 22
- `r-ud` path operation, 22
- `radius` key, 19
- `ratio` key, 20
- Rays arrow tip, 45
- Rectangle arrow tip, 45
- `rectangle path` operation, 8, 26
- `rectangle shape`, 12
- `rectangle timer` key, 26

- `rectangle with rounded corners shape`, 59
- `rectangle with rounded corners north east radius` key, 59
- `rectangle with rounded corners north west radius` key, 59
- `rectangle with rounded corners radius` key, 59
- `rectangle with rounded corners south east radius` key, 59
- `rectangle with rounded corners south west radius` key, 59
- `reverse clip` key, 76
- `right` key, 32
- `right vertical left` key, 23
- `rl distance` key, 22
- `rotate` key, 19
- `save picture size` key, 35
- Sep date test, 68
- `setup shape` key, 13
- `shape border rotate` key, 56
- Shapes
 - circle, 12
 - circle arrow, 50
 - circle cross split, 53
 - heatmark, 56
 - rectangle, 12
 - rectangle with rounded corners, 59
 - superellipse, 61
 - uncentered rectangle, 64
- `shapes.geometric` library, 14
- `sin` math function, 48
- `sin path` operation, 8, 27
- `size` key, 13
- `sloped` key, 16, 17
- `small` key, 19
- `softpath arrow` pic type, 8
- `softpath arrows` key, 8
- `softpath arrows` pic type, 8
- `south left` key, 32
- `south right` key, 32
- `spacing` key, 21
- `span` key, 34
- `span horizontal` key, 34
- `span vertical` key, 34
- `ext.spath3` library, 16

- Square arrow tip, [45](#)
- Stealth arrow tip, [45](#)
- Straight Barb arrow tip, [45](#)
- strcat math function, [72](#)
- strepeat math function, [72](#)
- superellipse shape, [61](#)
- superellipse exponent key, [61](#)
- superellipse step key, [61](#)
- superellipse x exponent key, [61](#)
- superellipse y exponent key, [61](#)
- superellipsex math function, [62](#)
- superellipsey math function, [62](#)
- Tee Barb arrow tip, [45](#), [46](#)
- text key, [29](#)
- text key, [12](#)
- text depth key, [12](#)
- text height key, [12](#)
- text width key, [12](#)
- text width align key, [13](#)
- through key, [37](#)
- /tikz/
 - above, [32](#)
 - above left, [30](#)
 - above right, [30](#)
 - arc arrows, [8](#), [9](#)
 - arc through/
 - center suffix, [37](#)
 - clockwise, [37](#)
 - counter clockwise, [37](#)
 - through, [37](#)
 - arc through, [37](#)
 - arc to/
 - clockwise, [18](#)
 - counter clockwise, [19](#)
 - large, [19](#)
 - radius, [19](#)
 - rotate, [19](#)
 - small, [19](#)
 - x radius, [19](#)
 - y radius, [19](#)

- arrow shift factor, [7](#), [9](#)
- arrow shift mode, [7](#), [9](#)
- auto, [16](#)
- auto offset, [16](#), [17](#)
- auto offset for brace decoration, [17](#)
- auto with offset, [16](#)
- autobend down, [38](#)
- autobend east, [38](#)
- autobend left, [38](#)
- autobend north, [38](#)
- autobend right, [38](#)
- autobend south, [38](#)
- autobend up, [38](#)
- autobend west, [38](#)
- below, [32](#)
- below left, [30](#)
- below right, [30](#)
- bend left, [38](#)
- clip rule, [76](#)
- commutative diagrams/
 - every diagram, [65](#)
 - install uncentered rectangle in columns, [66](#)
 - math mode, [65](#)
- corner above left, [30](#)
- corner above right, [30](#)
- corner below left, [30](#)
- corner below right, [30](#)
- corner east above, [33](#)
- corner east below, [33](#)
- corner north left, [33](#)
- corner north right, [33](#)
- corner south left, [33](#)
- corner south right, [33](#)
- corner west above, [33](#)
- corner west below, [33](#)
- day code, [10](#)
- day text, [10](#)
- day xshift, [10](#)
- day yshift, [10](#)
- down horizontal up, [23](#)
- east above, [32](#)

- east below, [32](#)
- edge in box, [11](#)
- edge on layer, [11](#)
- even odd rule, [76](#)
- every arc to, [19](#)
- every brace node, [17](#)
- every day, [10](#)
- every month, [10](#)
- every softpath arrows, [8](#), [9](#)
- every week, [10](#)
- execute at end node, [66](#)
- execute at end picture, [12](#)
- fit bounding box, [34](#)
- horizontal vertical, [23](#)
- horizontal vertical horizontal, [23](#)
- if, [10](#)
- image as pattern/
 - name, [29](#)
 - option, [29](#)
 - options, [29](#)
- image as pattern, [29](#)
- left, [31](#), [32](#)
- left vertical right, [23](#)
- matrix in box, [11](#)
- matrix of math nodes, [65](#)
- matrix on layer, [11](#)
- maximum picture height, [35](#)
- maximum picture size, [35](#)
- maximum picture width, [35](#)
- maximum picture width*, [36](#)
- minimum picture height, [35](#)
- minimum picture size, [35](#)
- minimum picture width, [35](#)
- minimum picture width*, [36](#)
- Mirror, [42](#)
- mirror, [41](#)
- Mirror x, [42](#)
- mirror x, [41](#)
- Mirror y, [42](#)
- mirror y, [41](#)
- month code, [10](#)

- month text, [10](#)
- month xshift, [10](#)
- month yshift, [10](#)
- node family/
 - height, [13](#), [14](#)
 - prefix, [12](#)
 - setup shape, [13](#)
 - size, [13](#)
 - text, [12](#)
 - text depth, [12](#)
 - text height, [12](#)
 - text width, [12](#)
 - text width align, [13](#)
 - width, [13](#), [14](#)
- node in box, [11](#)
- node on layer, [11](#)
- node on line, [15](#)
- nodes on curve, [16](#)
- nodes on curve', [16](#)
- nodes on line, [15](#), [16](#)
- nonzero rule, [76](#)
- north left, [31](#)
- north right, [32](#)
- on grid, [34](#)
- only horizontal first, [24](#)
- only horizontal second, [23](#)
- only vertical first, [24](#)
- only vertical second, [23](#)
- ortho/
 - distance, [20](#)
 - du distance, [22](#)
 - from center, [21](#)
 - install shortcuts, [25](#)
 - lr distance, [22](#)
 - middle 0 to 1, [22](#)
 - ratio, [20](#)
 - rl distance, [22](#)
 - spacing, [21](#)
 - ud distance, [22](#)
 - udlr distance, [22](#)
- pic, [15](#)

- pic in box, [11](#)
- pic on layer, [11](#)
- pics/
 - background code, [11](#)
 - code, [11](#)
 - foreground code, [11](#)
- picture height, [35](#)
- picture height*, [36](#)
- picture size*, [36](#)
- picture width, [35](#)
- picture width*, [35](#)
- pos, [21](#), [26](#)
- pos, [7](#)
- pos <, [7](#)
- pos >, [7](#)
- precise auto angle, [17](#)
- rectangle timer, [26](#)
- reverse clip, [76](#)
- right, [32](#)
- right vertical left, [23](#)
- save picture size, [35](#)
- sloped, [16](#), [17](#)
- softpath arrows, [8](#)
- south left, [32](#)
- south right, [32](#)
- span, [34](#)
- span horizontal, [34](#)
- span vertical, [34](#)
- to path, [15](#), [23](#)
- unrec, [66](#)
- unrec math mode, [66](#)
- up horizontal down, [23](#)
- vertical horizontal, [23](#)
- vertical horizontal vertical, [23](#)
- week code, [10](#)
- week label left, [10](#)
- week text, [10](#)
- west above, [32](#)
- west below, [32](#)
- x radius, [19](#)
- xMirror, [42](#)

- mirror, [40](#)
 - y radius, [19](#)
 - yMirror, [42](#)
 - ymirror, [41](#)
- tikz-cd fix key, [65](#)
- /tikz-ext/
 - layers/
 - in box, [11](#)
 - on layer, [11](#)
 - nodes/
 - install auto offset for brace decoration, [17](#)
 - patch, [11](#)
 - tikz-cd fix, [65](#)
- tikzcd environment, [65](#)
- \tikzextpictureheight, [35](#)
- \tikzextpicturewidth, [35](#)
- to path operation, [37](#)
- to path key, [15](#), [23](#)
- Triangle arrow tip, [45](#)
- Turned Square arrow tip, [45](#)
- ud distance key, [22](#)
 - udlr distance key, [22](#)
 - uncentered rectangle shape, [64](#)
 - uncentered rectangle center key, [64](#), [66](#)
 - uncentered rectangle center yshift key, [64](#)
 - uncentered rectangle use saved center key, [64](#)
 - \unrec, [66](#)
 - unrec key, [66](#)
 - unrec math mode key, [66](#)
 - Untipped Bar arrow tip, [46](#)
 - Untipped Bracket arrow tip, [46](#)
 - Untipped Circle arrow tip, [46](#)
 - Untipped Ellipse arrow tip, [46](#)
 - Untipped Parenthesis arrow tip, [46](#)
 - Untipped Tee Barb arrow tip, [46](#)
 - up horizontal down key, [23](#)
 - use float key, [70](#)
 - use int key, [70](#)
- /utils/
 - if, [74](#)

TeX/

if, [74](#)

ifdim, [74](#)

ifempty, [74](#)

ifnum, [74](#)

ifx, [74](#)

ifxempty, [74](#)

var key, [71](#)

vertical horizontal key, [23](#)

vertical horizontal vertical key, [23](#)

week date test, [68](#)

week code key, [10](#)

week label left key, [10](#)

week of month date test, [68](#)

week of month' date test, [68](#)

week text key, [10](#)

weeksinmonthofyear math function, [10](#)

west above key, [32](#)

west below key, [32](#)

width key, [13](#), [14](#)

x radius key, [19](#)

x radius key, [19](#)

xMirror key, [42](#)

xmirror key, [40](#)

xparse-ext package, [71](#)

xparser key, [71](#)

xparser Om key, [71](#)

y radius key, [19](#)

y radius key, [19](#)

yesterday date test, [68](#)

yMirror key, [42](#)

ymirror key, [41](#)

References

- [1] ‘sloped’ should consider the current transformation · Issue #1058 · pgf-tikz/pgf. URL: <https://github.com/pgf-tikz/pgf/issues/1058> (visited on 10/21/2023) (cit. on p. 8).
- [2] 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. 70).
- [3] boje. *Heatmap over country like Google Map*. May 2013. URL: <https://tex.stackexchange.com/q/112929> (visited on 09/24/2022) (cit. on p. 56).
- [4] Christian. *TikZ arrow tip is displaced*. TeX - LaTeX Stack Exchange. Apr. 2013. URL: <https://tex.stackexchange.com/q/111051> (visited on 04/02/2023) (cit. on p. 44).
- [5] 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. 10).
- [6] 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. 26).
- [7] 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. 26).
- [8] 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. 70).
- [9] Alejandro DC. *Better fitting line to node in TiKZ*. TeX - LaTeX Stack Exchange. Apr. 2015. URL: <https://tex.stackexchange.com/q/241074> (visited on 04/01/2023) (cit. on p. 44).
- [10] 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. 15).
- [11] Fence. *Add week day to calendar*. Nov. 2019. URL: <https://tex.stackexchange.com/q/517338> (visited on 09/24/2022) (cit. on pp. 10, 68).
- [12] healyp. *TikZ calendar and conditional tests*. Oct. 2013. URL: <https://tex.stackexchange.com/q/140948> (visited on 09/24/2022) (cit. on pp. 10, 68).
- [13] Jan Hlavacek. *Modifying * and o style tikz arrows so that they are centered at the end of line*. TeX - LaTeX Stack Exchange. Feb. 2011. URL: <https://tex.stackexchange.com/q/11871> (visited on 04/02/2023) (cit. on p. 44).
- [14] Holene. *Dependent node size in TikZ*. Apr. 2017. URL: <https://tex.stackexchange.com/q/107227> (visited on 09/24/2022) (cit. on p. 12).
- [15] Edgar A. Bering IV. *Set the color of a tikz-cd Glyph arrow tip with xelatex*. TeX - LaTeX Stack Exchange. Oct. 2020. URL: <https://tex.stackexchange.com/q/565010> (visited on 04/01/2023) (cit. on p. 44).
- [16] 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. 68).
- [17] 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. 29).
- [18] 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. 15).
- [19] Marco. *TikZ - Four Colored Circle Split*. Apr. 2017. URL: <https://tex.stackexchange.com/q/121686> (visited on 09/24/2022) (cit. on p. 53).
- [20] 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. 77).
- [21] marmotghost. *Latest version of ext.misc on CTAN appears to have a typo*. Mar. 2023. URL: <https://github.com/Qrrbrbirlbel/tikz-extensions/issues/6> (visited on 04/01/2023) (cit. on p. 77).

- [22] Alan Munn. *Determine TikZ bend direction automatically*. TeX - LaTeX Stack Exchange. Oct. 2023. URL: <https://tex.stackexchange.com/q/699883> (visited on 10/31/2023) (cit. on p. 38).
- [23] 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. 14).
- [24] 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. 72).
- [25] 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. 64).
- [26] 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. 50).
- [27] 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. 72).
- [28] 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. 10, 68).
- [29] 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. 61).
- [30] Qrrbrbirlbel. *Answer to "Better fitting line to node in TiKZ"*. TeX - LaTeX Stack Exchange. Apr. 2015. URL: <https://tex.stackexchange.com/a/241303> (visited on 04/01/2023) (cit. on p. 44).
- [31] 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. 12).
- [32] Qrrbrbirlbel. *Answer to "Determine TikZ bend direction automatically"*. TeX - LaTeX Stack Exchange. Oct. 2023. URL: <https://tex.stackexchange.com/a/699919> (visited on 10/31/2023) (cit. on p. 38).
- [33] 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. 15).
- [34] 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. 64).
- [35] 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. 68).
- [36] 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. 56).
- [37] 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. 64).
- [38] 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. 70).
- [39] 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. 26).
- [40] 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. 70).

- [41] Qrrbrbirlbel. *Answer to “Modifying * and o style tikz arrows so that they are centered at the end of line”*. TeX - LaTeX Stack Exchange. Sept. 2022. URL: <https://tex.stackexchange.com/a/656241> (visited on 04/02/2023) (cit. on p. 44).
- [42] 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. 15).
- [43] Qrrbrbirlbel. *Answer to “Set the color of a tikz-cd Glyph arrow tip with xelatex”*. TeX - LaTeX Stack Exchange. Apr. 2023. URL: <https://tex.stackexchange.com/a/681474> (visited on 04/01/2023) (cit. on p. 44).
- [44] 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. 70).
- [45] 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. 50).
- [46] 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. 53).
- [47] 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. 10).
- [48] Qrrbrbirlbel. *Answer to “TikZ arrow tip is displaced”*. TeX - LaTeX Stack Exchange. Apr. 2013. URL: <https://tex.stackexchange.com/a/111053> (visited on 04/02/2023) (cit. on p. 44).
- [49] 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. 10, 68).
- [50] 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. 29).
- [51] 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. 26).
- [52] 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. 61).
- [53] sro5h. *Achieve desired alignment of arrows in tikz-cd diagram*. TeX - LaTeX Stack Exchange. July 2022. URL: <https://tex.stackexchange.com/q/652540> (visited on 02/19/2023) (cit. on p. 65).
- [54] 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. 16).
- [55] 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. 64).
- [56] 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. 70).