

# The TikZ-Extensions Package

## Manual for version 0.6.3 (11)

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

Qrrbrbirlbel

June 29, 2026

## Contents

<b>I</b>	<b>Introduction</b>	<b>6</b>
1	Usage	6
2	Why do we need it?	6
3	Having problems?	6
4	Namespaces and <i>TikZ</i> -Extensions macros	6
5	Compatibility with older versions	7
<b>II</b>	<b><i>TikZ</i> Libraries</b>	<b>8</b>
6	Arrow Pics	9
6.1	Arrow pic types . . . . .	10
6.2	Arrow keys . . . . .	11
6.3	Shifted and bended arrows for the <code>decorations.markings</code> library . . . . .	11

<b>7</b>	<b>Beamer with TikZ</b>	<b>13</b>
7.1	Helpers . . . . .	13
7.2	BEAMER . . . . .	13
7.2.1	Stop Jumping . . . . .	14
7.2.2	BEAMER Function and keys . . . . .	14
7.2.3	BEAMER Shortcuts . . . . .	15
7.2.4	Key Handler . . . . .	17
<b>8</b>	<b>Calendar</b>	<b>18</b>
8.1	Value-keys and nestable if key . . . . .	18
8.2	PGFmath functions . . . . .	18
8.3	Week numbering (ISO 8601) . . . . .	18
<b>9</b>	<b>Layers</b>	<b>19</b>
<b>10</b>	<b>Node Families</b>	<b>20</b>
10.1	Externalization . . . . .	20
10.2	Text Box . . . . .	20
10.3	Minimum Width/Height . . . . .	21
10.4	More shapes that support the keys width and height . . . . .	23
<b>11</b>	<b>Nodes</b>	<b>24</b>
11.1	Pic as a node . . . . .	24
11.2	Node Pictures . . . . .	24
11.2.1	Overview . . . . .	25
11.2.2	Styles . . . . .	25
11.2.3	Settings . . . . .	26
11.2.4	More and Examples . . . . .	27
11.3	Nodes on paths . . . . .	29
11.3.1	Nodes on Lines . . . . .	29
11.3.2	Nodes on Curves . . . . .	30
11.4	Automatic placement of nodes . . . . .	31
11.4.1	More than left and right . . . . .	31
11.4.2	Offset . . . . .	31
11.4.3	Precise placement . . . . .	32
<b>12</b>	<b>Arc to a point</b>	<b>33</b>

<b>13</b>	<b>More Horizontal and Vertical Lines</b>	<b>35</b>
13.1	Timers . . . . .	35
13.2	Zig-Zag . . . . .	36
13.3	Zig-Zig . . . . .	37
13.4	Even more Horizontal and Vertical Lines . . . . .	38
<b>14</b>	<b>Extending the Path Timers</b>	<b>39</b>
14.1	Rectangle . . . . .	39
14.2	Parabola . . . . .	40
14.3	Sine/Cosine . . . . .	40
<b>15</b>	<b>Using Images as a Pattern</b>	<b>42</b>
<b>16</b>	<b>Positioning Plus</b>	<b>43</b>
16.1	Useful corner anchors . . . . .	43
16.2	Useful placement keys for vertical and horizontal alignment . . . . .	44
<b>17</b>	<b>Scaling Pictures to a Specific Size</b>	<b>46</b>
17.1	Externalization . . . . .	46
17.2	Keeping the aspect ratio . . . . .	46
17.3	Changing the aspect ratio . . . . .	47
<b>18</b>	<b>Arcs through Three Points</b>	<b>48</b>
<b>19</b>	<b>Autobending</b>	<b>49</b>
<b>20</b>	<b>Mirror, Mirror on the Wall</b>	<b>51</b>
20.1	Using the reflection matrix . . . . .	51
20.2	Using built-in transformations . . . . .	53
<b>III</b>	<b>PGF Libraries</b>	<b>55</b>
<b>21</b>	<b>Arrow Tips</b>	<b>56</b>
21.1	Centered . . . . .	57
21.1.1	Barbed Arrow Tips . . . . .	57
21.1.2	Geometric Arrow Tips . . . . .	57
21.1.3	Special Arrow Tips . . . . .	58
21.2	Untipped . . . . .	58

21.2.1	Barbed Arrow Tips . . . . .	58
21.2.2	Geometric Arrow Tips . . . . .	58
21.3	Original Arrow Tips . . . . .	59
<b>22</b>	<b>Transformations: Mirroring</b>	<b>61</b>
22.1	Using the reflection matrix . . . . .	61
22.2	Using built-in transformations . . . . .	61
<b>23</b>	<b>Shape: Circle Arrow</b>	<b>63</b>
<b>24</b>	<b>Shape: Circle Cross Split</b>	<b>67</b>
<b>25</b>	<b>Shape: Heatmark</b>	<b>70</b>
<b>26</b>	<b>Shape: Rectangle with Rounded Corners</b>	<b>73</b>
<b>27</b>	<b>Shape: Superellipse</b>	<b>75</b>
<b>28</b>	<b>Shape: Uncentered Rectangle</b>	<b>78</b>
<b>IV</b>	<b>Utilities</b>	<b>81</b>
<b>29</b>	<b>Calendar: Weeknumbers and more conditionals</b>	<b>82</b>
29.1	Extensions . . . . .	82
29.2	Week numbering (ISO 8601) . . . . .	83
<b>30</b>	<b>Repeating Things and Other Things</b>	<b>84</b>
<b>31</b>	<b>And a little bit more</b>	<b>86</b>
31.1	PGFmath . . . . .	86
31.1.1	Postfix operator R . . . . .	86
31.1.2	Functions . . . . .	86
31.1.3	Functions: using coordinates . . . . .	87
31.2	PGFfor . . . . .	88
31.3	PGFkeys . . . . .	88
31.3.1	Conditionals . . . . .	88
31.3.2	Handlers . . . . .	88
31.4	TikZ . . . . .	91

<b>V</b>	<b>Changelog, Index &amp; References</b>	<b>92</b>
	Changelog	92
	Index	94
	References	108

## Part I

# Introduction

## 1 Usage

This package is called `tikz-ext`, however, one can't load it via `\usepackage`.<sup>1</sup> 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.

## 4 Namespaces and `TikZ-Extensions` macros

Since some parts of this package have existed in some form since 2013, the choice for key names and in which `pgfkeys` namespace they reside is not always optimal. They often reside in the main `/tikz` or `/pgf` path. Similar applies to macro names.

Starting with version 0.6, the namespace for almost all keys is `/tikz/ext` or `/pgf/ext`. The same applies to macros that shall be starting with `\tikzext` or `\pgfext`.

Starting with version 0.6, `TikZ-Extensions` provides commands that return the current version for compatibility testing. The second simply increments with every release so that the first doesn't need to be parsed.

`\tikzextversion`

Returns 0.6.3.

`\tikzextversionnumber`

Returns 11.

Also there's `\tikzextset` and `\pgfextset`.

`\tikzextset{<options>}`

This command will process the `<options>` using the `\pgfkeys` command with the default path set to `/tikz/ext`.

`\pgfextset{<options>}`

This command will process the `<options>` using the `\pgfkeys` command with the default path set to `/pgf/ext`.

---

<sup>1</sup>Except for `pgfcalendar-ext` and `pgffor-ext`.

## 5 Compatibility with older versions

As discussed in the previous section, keys and commands of extensions that existed before version 0.6 that do not appear in this manual are considered deprecated.

`/tikz/ext/compat=pre 0.6|0.6|warn|newest` (default pre 0.6)

This sets the global compatibility setting for every extension of this package (whether already loaded or not).

The choice `warn` gives out warning for deprecated keys or commands but still executes them if they were not in use when an extension was loaded.

Starting with version 0.6.3, the default compatibility setting is `newest`, i.e. 0.6.

The following table shows the compatibility settings for each extension. A ✓ denotes an available setting where ✓ denotes the default compatibility setting. A – denotes that it is not different than the newest setting.

Extension	warn	pre 0.6	0.6
pgfcalendar-ext	✓	✓	✓/–
ext.calendar-plus			
ext.arrows	✓	✓	✓/–
ext.layers	✓	✓	✓/–
ext.node-families	✓	✓	✓/–
ext.nodes	✓	✓	✓/–
ext.paths.arcto	✓	✓	✓/–
ext.paths.ortho	✓	✓	✓/–
ext.paths.timer	✓	✓	✓/–
ext.patterns.images	✓	✓	✓/–
ext.pgffor-ext	✓	✓	✓/–
ext.pgfkeys-plus	✓	✓	✓/–
ext.positioning-plus	✓	✓	✓/–
ext.scalepicture	✓	✓	✓/–
ext.shapes	✓	✓	✓/–
ext.transformations.mirror	✓	✓	✓/–
ext.topaths.arctthrough	✓	✓	✓/–

For each available extension the compatibility setting can be adjusted as well after the extension is loaded.

`/tikz/ext/compat/pgfcalendar-ext=<version>` (default pre 0.6)  
`/tikz/ext/compat/arrows=<version>` (default pre 0.6)  
`/tikz/ext/compat/layers=<version>` (default pre 0.6)  
`/tikz/ext/compat/nodes=<version>` (default pre 0.6)  
`/tikz/ext/compat/node-families=<version>` (default pre 0.6)  
`/tikz/ext/compat/paths.arcto=<version>` (default pre 0.6)  
`/tikz/ext/compat/paths.ortho=<version>` (default pre 0.6)  
`/tikz/ext/compat/paths.timer=<version>` (default pre 0.6)  
`/tikz/ext/compat/patterns.images=<version>` (default pre 0.6)  
`/tikz/ext/compat/pgffor-ext=<version>` (default pre 0.6)  
`/tikz/ext/compat/pgfkeys-plus=<version>` (default pre 0.6)  
`/tikz/ext/compat/positioning-plus=<version>` (default pre 0.6)  
`/tikz/ext/compat/scalepicture=<version>` (default pre 0.6)  
`/tikz/ext/compat/shapes=<version>` (default pre 0.6)  
`/tikz/ext/compat/transformations.mirror=<version>` (default pre 0.6)  
`/tikz/ext/compat/topaths.arctthrough=<version>` (default pre 0.6)

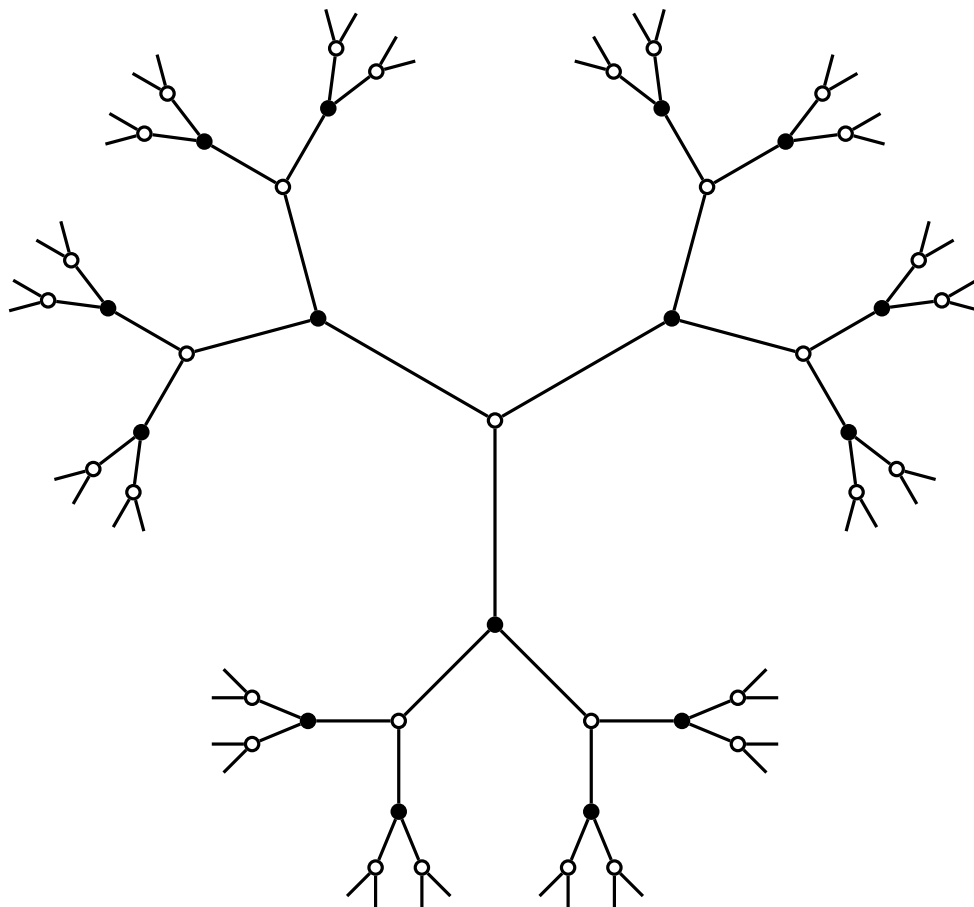
For `<version>` the same choices are valid as for the main `compat` key. It should be noted that at this point, a compatibility setting can't really be reversed since they only forward arguments from an old key or command to the new version.

The old names are given as a subtitle to the new one in the sections that introduce them.

## Part II

# TikZ Libraries

These libraries only work with TikZ.





## 6 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/ext/pos <=<value>` (no default, initially 0.0)

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

`/tikz/ext/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/ext/pos < angle=<angle>` (no default)

For tips along an arc the angle along that arc can be specified for the start tip sequence.

`/tikz/ext/pos > angle=<angle>` (no default)

For tips along an arc the angle along that arc can be specified for the end tip sequence.

`/tikz/ext/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.

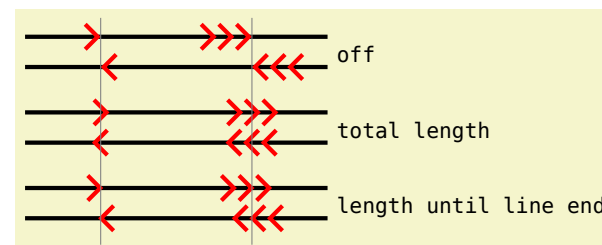
**arrow shift mode=off** This disables the shifting.

**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.



```
\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[ext/arrow shift mode=\shiftmode] (0, -\y )
-- pic {ext/arrow=>} ++(right:2)
-- pic {ext/arrow=>.>} ++(right:2) node[below right] {\shiftmode}
++(down:.4) -- pic {ext/arrow=>.>} ++( left:2)
-- pic {ext/arrow=>} ++( left:2);
\draw[thin, gray] (1,-.75) -- +(down:3) (3,-.75) -- +(down:3);
\end{tikzpicture}
```

For single arrow tips it might be better to use the Centered arrow tip variants of the `ext.arrows` library (see sec 21) 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/ext/arrow shift factor start=<value>` (no default, initially 0.5)

This determines the shift factor for the start tip sequence.

The default value is probably good for most cases.

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

This determines the shift factor for the end tip sequence.

The default value is probably good for most cases.

`/tikz/ext/arrow shift factor=<value>` (no default)

This sets both the start and end shift factor.

## 6.1 Arrow pic types

This library provides the following pics:

**ext/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 14).

The arrow tips will never be bended. For this the following pic types or the `/tikz/ext/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.

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

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

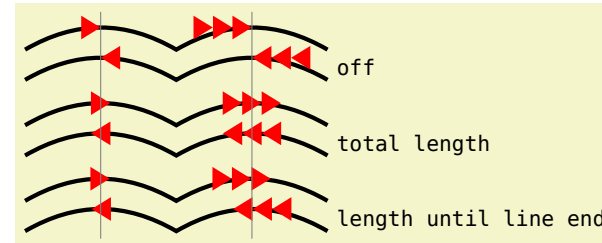
This pic type can place two tip specification, one at `pos >` and one at `pos <` in the reversed direction.

**ext/softpath arrow** This is an alias for `softpath arrows` with an empty start arrow tip specification.

**Pic type** `ext/arrow=<arrow tip specification>`

This pic draws the given `<arrow tip specification>` (defaults to the end tip specification of the path).

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



```
\usetikzlibrary {arrows.meta, 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[ext/arrow shift mode=\shiftmode] (0, -\y)
    to[bend left] pic {ext/arrow=>} ++(right:2)
    to[bend left] pic {ext/arrow=>>.>} ++(right:2)
    node[below right] {\shiftmode}
    ++(down:.4) to[bend right] pic {ext/arrow=>>.>} ++(left:2)
    to[bend right] pic {ext/arrow=>} ++(left:2);
\draw[thin, gray] (1,-.5) -- +(down:3) (3,-.5) -- +(down:3);
\end{tikzpicture}
```

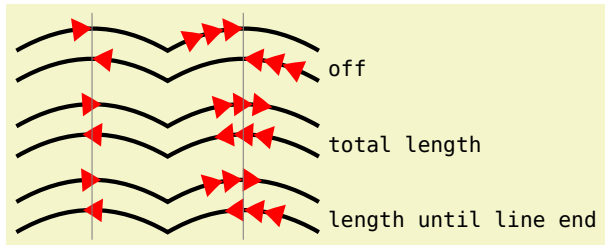
**Pic type** `ext/softpath arrows=<start tip specification>-<end tip specification>`

This pic draws the given arrow tip specification (defaults to the already present tip specification of the path) 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 `/tikz/ext/arc arrows` key instead.

**Pic type** `ext/softpath arrow=<end tip specification>`

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



```
\usetikzlibrary {arrows.meta, 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[ext/arrow shift mode=\shiftmode] (0, -\y)
    to[bend left] pic {ext/softpath arrow=>} ++(right:2)
    to[bend left] pic {ext/softpath arrow=>.>} ++(right:2)
    node[below right] {\shiftmode}
    ++(down:.4) to[bend right] pic {ext/softpath arrow=>.>} ++(left:2)
    to[bend right] pic {ext/softpath arrow=>} ++(left:2);
\draw[thin, gray] (1,-.5) -- +(down:3) (3,-.5) -- +(down:3);
\end{tikzpicture}
```

## 6.2 Arrow keys

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

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

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

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

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

- from /tikz/pos > to /tikz/ext/pos > and
- from /tikz/pos < to /tikz/ext/pos >.

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

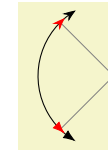
`/tikz/ext/arc arrows=<options>` (default ->)

This key adds arrow tips to the previous arc segment.

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

This style will be applied for every instance of arc arrows. It also sets up forwarding

- from /tikz/pos > to /tikz/ext/pos >,
- from /tikz/pos < to /tikz/ext/pos > as well as
- from /tikz/pos > angle to /tikz/ext/pos > angle and
- from /tikz/pos < angle to /tikz/ext/pos < angle.



```
\usetikzlibrary {arrows.meta, ext.arrows-plus}
\begin{tikzpicture}
\draw (0,0) -- (135:1)
      (0,0) -- (225:1) [help lines];
\draw[arrows={[bend] Stealth-Latex}]
      (120:1) arc[start angle=120,
                  end angle=240, radius=1]
      [ext/arc arrows={pos < angle=135,
                      pos > angle=225,
                      arrows=[red]}];
\end{tikzpicture}
```

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

## 6.3 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 arrow shift mode and 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**
;
```

## 7 Beamer with TikZ

**TikZ Library** `ext.beamer`

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

This library can help create TikZ diagrams in the BEAMER class.

### 7.1 Helpers

These helpers are always available, even if this library is loaded outside of BEAMER.

`/tikz/ext/ignore line width` (no value)

If this key is used on a scope (or the TikZ picture itself), the line widths of paths will not contribute to the bounding box of the diagram.

`/tikz/ext/max bounding box=<name>` (no default)

This key is to be used on multiple `tikzpicture` environments. All TikZ diagram with the same `<name>` will have the same bounding box. Refrain from using (unprotected) commas (,) in the `<name>`.

This uses the AUX file and is therefore incompatible with the external library.

However, it is made compatible with the `memoize` [65] package, even if it takes a few compilations until it is stable again after a new diagram is added to the group.

### 7.2 BEAMER

While TikZ has some rudimentary support for the BEAMER class, i. e. in the form of `\path<overlay specification>`, this uses BEAMER's `\alt` command internally so that on overlays that are not included in `<overlay specification>`, the path will not be typeset and will therefore not contribute to the diagram's bounding box.

This in turn will lead to the diagram “jumping around” as every overlay will contain a different diagram with different dimensions. The `aobs-tikz` package solves this by setting the opacity to zero for all those slides an element shouldn't be visible on.

I believe we can do better.

Though, remember that for many simple diagrams, you can simply use `\onslide`. The following diagram will show

- nodes and edges transparent on overlay 1,
- nodes fully visible and edges transparent on overlay 2 and
- all elements fully visible on overlay 3.

```

\usetikzlibrary {ext.beamer} \setbeamercovered {transparent}
\begin{tikzpicture}
\onslide<2->
\path node (a) {A}
      node (b) at (1,2) {B};
\onslide<3->
\path (a) edge[bend right] (b);
\onslide
\end{tikzpicture}

```

### 7.2.1 Stop Jumping

One solution to this is to have the same TikZ diagram have the same size on every overlay. The `/tikz/ext/ignore line width` might help if all that changes between overlays is the line width of elements.

Another one is the following key.

`/tikz/ext/sync bounding box` (no value)

This key uses `ext/max bounding box` with a specific *<name>* that is stable across overlays.

If you find yourself often rearrange diagrams or changing overlays, you might be better off using the `ext/sync bounding box` key directly with a distinct *<name>*.

### 7.2.2 BEAMER Function and keys

`/tikz/ext/beamer function=original|alt|only|uncover|visible|invisible` (no default)

This key changes how the *<overlay specification>* in `\path<overlay specification>` is applied internally. The choices `original`, `alt` and `only` are all the same and will result in the default behavior of TikZ.

The same overlays as above can be created now with the following diagram.

```

\usetikzlibrary {ext.beamer} \setbeamercovered {transparent}
\begin{tikzpicture}[ext/beamer function=uncover]
\path<2-> node (a) {A}
      node (b) at (1,2) {B};
\path<3-> (a) edge[bend right] (b);
\end{tikzpicture}

```

`/tikz/ext/uncover=<overlay specification>` (default all)

`/tikz/ext/cover=<overlay specification>` (default all)

`/tikz/ext/visible=<overlay specification>` (default all)

`/tikz/ext/invisible=<overlay specification>` (default all)

These keys work similar to BEAMER's own `\onslide` command but only apply to the element it is used on.

The implementation of this is rather experimental and should be used carefully. Multiple uses of these options *don't* stack, only the last one wins – this includes the use of the first syntax shortcuts introduced in the next subsection.

The same overlays as above can be created with the first of the following diagram. In the second diagram `ext/uncover` is only used on the (actual) empty path. Just like `draw/\draw`, the nodes will not observe the request to be covered on overlay 1.

```
\usetikzlibrary {ext.beamer} \setbeamercovered {transparent}
\begin{tikzpicture}
\path[nodes={ext/uncover=2-}]
  node (a)          {A}
  node (b) at (1,2) {B}
  (a) edge[bend right, ext/uncover=3-] (b);
\end{tikzpicture}
\begin{tikzpicture}
\draw[ext/uncover=2-]
  node (a)          {A}
  node (b) at (1,2) {B};
\end{tikzpicture}
```

`/tikz/ext/aobs visible=<overlay specification>`

(default all)

`/tikz/ext/aobs invisible=<overlay specification>`

(default all)

In case one wants to use the method of simply setting the opacity of elements to zero to hide them, these keys are also available.

Of course, an extension to Beamer is not complete without the following keys.

`/utils/ext/only={<overlay specification>}{<key-value list>}`

(no default)

Applies the `<key-value list>` only on `<overlay specification>`.

`/utils/ext/alt={<overlay specification>}{<default kv list>}{<alternative kv list>}`

(no default)

Applies the `<default kv list>` on `<overlay specification>`, otherwise the `<alternative kv list>`.

`/utils/ext/temporal={<overlay specification>}{<before kv list>}{<default kv list>}{<after kv list>}`

(no default)

Applies the specific list depending whether the current overlay is before, on or after the specified `<overlay specification>`.

### 7.2.3 BEAMER Shortcuts

But, of course, no one wants to write `/utils/ext/only={2}{red}` to make an element red on overlay 2.

`/tikz/ext/beamer shortcuts={<key-value list>}`

(no default)

This executes the `<key-value list>` in the namespace `/tikz/ext/beamer shortcuts`.

`/tikz/ext/beamer shortcuts/aot`

(no value)

This forwards the keys `/tikz/alt`, `/tikz/only` and `/tikz/temporal` to the aforementioned homonymous `/utils/ext` keys.

`/tikz/ext/beamer shortcuts/first char=uncover|cover|visible|invisible|aobs visible|aobs invisible`

(no default, initially uncover)

The value of this key will be used for the first char shorthands that can be enabled with the following keys.

`/tikz/ext/beamer shortcuts/enable first char <`

(no value)

This install a “first char” handler with the character `<`.

This allows the example diagram to specified in the following way.

```
\usetikzlibrary {ext.beamer} \setbeamercovered {transparent}
\begin{tikzpicture}[ext/beamer shortcuts={enable first char <}]
\node (a) [<2->] {A};
\node (b) [<2->] at (1,2) {B};
\path (a) edge[<3->, bend right] (b);
\end{tikzpicture}
```

Internally, this will be converted to `ext/uncover={⟨overlay specification⟩}`.

Actually, the full syntax is much more versatile

`<⟨overlay specification⟩>'⟨options⟩`  
`<⟨overlay specification⟩>' {⟨options A⟩}{⟨options B⟩}{⟨options C⟩}`

If no options are present the `⟨overlay specification⟩` will be forwarded to one of the keys explained in the subsection above – depending on `ext/beamer shortcuts/first char`. The optional `'` after `>` will invert the `⟨overlay specification⟩`.

If `⟨options⟩` or `{⟨options A⟩}` are present, these will only applied on `⟨overlay specifications⟩` (or the inverse of them with the `'`). If two sets of options are present, they will be `\alted`, three options will be `\temporal`d.

The `'` will swap two sets of options while for three the first and the last will be swapped.

`/tikz/ext/beamer shortcuts/enable first char={⟨character⟩}`

(no default)

As the `<` character might lead to problems as it conflicts with the TikZ shorthand of specifying arrow tip sequences (i. e. the famous `<->`<sup>2</sup>) and the graphs library’s own first char syntax an alternative is presented here.

This key enables a first char syntax with `⟨character⟩` where the full syntax is the same as above:

`⟨character⟩<⟨overlay specification⟩>'⟨options⟩`  
`⟨character⟩<⟨overlay specification⟩>' {⟨options A⟩}{⟨options B⟩}{⟨options C⟩}`

This means, the example diagram can be created in the following way.

---

<sup>2</sup>Though, remember, you can always write arrows = `<->`.



```

\usetikzlibrary {ext.beamer} \setbeamercovered {transparent}
\begin{tikzpicture}[ext/beamer shortcuts={enable first char=?}]
\node (a) [?<2->] {A};
\node (b) [?<2->] at (1,2) {B};
\path (a) edge[?<3->, bend right] (b);
\end{tikzpicture}

```

`/tikz/graphs/ext/better beamer shortcut`

(no value)

This sets up a few things that enables the < first char syntax in the appropriate places. For nodes, it tries to be smart and checks for the presence of a > to decide whether the BEAMER shortcut or the the graphs library's own shortcut should be used.

#### 7.2.4 Key Handler

Maybe this is a syntax that someone wants ...

**Key handler**  $\langle key \rangle / .ext\_< \langle overlay specification \rangle > value$

This handler applies the key on  $\langle overlay specification \rangle$ . If  $\langle value \rangle$  is missing, then the key is also used without a value. For an empty value, use {}. If the  $\langle value \rangle$  contains comas or equal signs, as always, you will need to protect those with {}.

`/tikz/ext/beamer shortcuts/enable handler`

(no value)

If ext\_ is too much, using this key activates the .< handler.

**Key handler**  $\langle key \rangle / .< \langle overlay specification \rangle > value$

This handler is then an alias for the .ext\_< handler.

## 8 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] & [31, 52, 50]

### 8.1 Value-keys and nestable if key

```
/tikz/day xshift           (initially 3ex)
/tikz/day yshift          (initially 3.5ex)
/tikz/month xshift        (initially 9ex)
/tikz/month yshift        (initially 9ex)
```

The values of these 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/if=(⟨conditions⟩)⟨code or options⟩else⟨else code or options⟩ (no default)
```

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

### 8.2 PGFmath functions

```
ext_weeksinmonthofyear(first weekday, month, year)
\pgfmathextweeksinmonthofyear{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*.

```
ext_lastdayinmonthofyear(month, year)
\pgfmathextlastdayinmonthofyear{month}{year}
```

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

### 8.3 Week numbering (ISO 8601)

The actual week number algorithm is implemented by the `pgfcalendar-ext` package/module in section 29.2.

```
/tikz/ext/week code=⟨code⟩ (no default)
```

pre 0.6 `/tikz/week code`

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

```
/tikz/ext/week text=⟨text⟩ (no default)
```

pre 0.6 `/tikz/week text`

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

```
/tikz/ext/every week (style, no value)
```

pre 0.6 `/tikz/every week`

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

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

pre 0.6 `/tikz/week label left`

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
28	11	12	13	14	15	16
29	18	19	20	21	22	23
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,
  ext/week label left,
  ext/every week/.append style={
    gray!50!black, font=\sfamily};
```

## 9 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.

`/tikz/ext/layers/patch=node|matrix|pic|edge|all` (no default)

pre 0.6 `/tikz-ext/layers/patch`

Since this library is experimental, its functionality needs to be activated explicitly. Patches exist for

- `node`,
- `matrix`,
- `pic`<sup>3</sup>,
- `edge` or
- `all` which applies all the patches at once.

`/tikz/ext/node on layer=<layer>` (no default)

pre 0.6 `/tikz/node on layer`

If the `node patch` is applied, this key places a node on layer `<layer>`.

`/tikz/ext/matrix on layer=<layer>` (no default)

pre 0.6 `/tikz/matrix on layer`

If the `matrix patch` is applied, this key places the matrix on layer `<layer>`.

`/tikz/ext/edge on layer=<layer>` (no default)

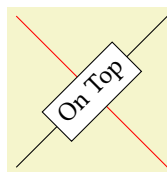
pre 0.6 `/tikz/edge on layer`

If the `edge patch` is applied, this key places the edge on layer `<layer>`.

`/tikz/ext/pic on layer=<layer>` (no default)

pre 0.6 `/tikz/pic on layer`

If the `pic patch` is applied, this key places the main code of a pic on layer `<layer>`.



```
\usetikzlibrary {ext.layers}
\pgfdeclarelayer{front}
\begin{tikzpicture}[ext/layers/patch=node]
\pgfsetlayers{main,front}
\draw (0, -1) -- node[
    ext/node on layer=front,
    draw, fill=white, sloped
] {On Top} (2,1);
\draw[red] (0, 1) -- (2, -1);
\end{tikzpicture}
```

<sup>3</sup>Only the normal `/tikz/pics/code` can be placed on different layers. Both `/tikz/pics/background` code and `/tikz/pics/foreground` code will not be affected.

## 10 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.

**Q & A:** [14] & [34]

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/ext/node family/prefix=<prefix> (no default, initially \pgfpictureid-)
pre 0.6 /tikz/node family/prefix
```

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

### 10.1 Externalization

As this library usually needs multiple compilations to produce stable pictures it is incompatible with the `external` library.

However, the library provides support for the `memoize` [65] package.

### 10.2 Text Box

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

```
/tikz/ext/node family/text height=<name> (no default, initially {})
pre 0.6 /tikz/node family/text height
```

Nodes with the same `<name>` will have the same text height. An empty `<name>` disables the evaluation by the library.

```
/tikz/ext/node family/text depth=<name> (no default, initially {})
pre 0.6 /tikz/node family/text depth
```

Nodes with the same `<name>` will have the same text depth. An empty `<name>` disables the evaluation by the library.

```
/tikz/ext/node family/text width=<name> (no default, initially {})
pre 0.6 /tikz/node family/text width
```

Nodes with the same `<name>` will have the same text width. An empty `<name>` disables the evaluation by the library.

```
/tikz/ext/node family/text=<name> (no default)
```

---

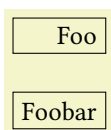
<sup>4</sup>Technically, it will also work with shapes with multiple node parts but it will only affect the main node part.

pre 0.6 /tikz/node family/text

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.

`/tikz/ext/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, ext/node family={text width=manual, text width align=right}}]
\node (a) {Foo};
\node[below=of a] (b) {Foobar};
\end{tikzpicture}
```

`/tikz/ext/node family/setup shape=<shape>` (no default)

pre 0.6 /tikz/node family/setup shape

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.

### 10.3 Minimum Width/Height

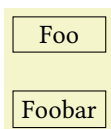
While the keys of the previous subsection work well enough for nodes of the same shape (and the same inner seps), for different node shapes the text box dimensions will be used differently for the node's total dimension.

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 `ext_nf_width` or `nf_height` respectively.

`/tikz/ext/node family/width=<name>` (no default, initially {})

pre 0.6 /tikz/node family/width

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, ext/node family/width=manual}]
\node (a) {Foo};
\node[below=of a] (b) {Foobar};
\end{tikzpicture}
```

`/tikz/ext/node family/height=<name>` (no default, initially {})

```
pre 0.6 /tikz/node family/height
```

Nodes with the same  $\langle name \rangle$  will have the same /pgf/minimum height. An empty  $\langle name \rangle$  disables the evaluation by the library.

```
/tikz/ext/node family/size= $\langle name \rangle$ 
```

(no default)

```
pre 0.6 /tikz/node family/size
```

Sets both height and width.

## 10.4 More shapes that support the keys width and height

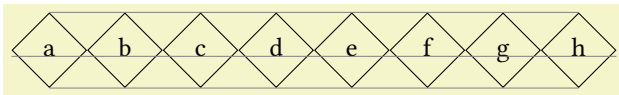
**TikZ Library** `ext.node-families.shapes.geometric`

```
\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/ext/node family/width` and `/tikz/ext/node family/height` for the shapes of the pgf library `shapes.geometric`.

Q: [25]

The shapes are also setup for the keys from subsection 10.2.



```
\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, ext/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}
```

## 11 Nodes

### TikZ Library `ext.nodes`

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

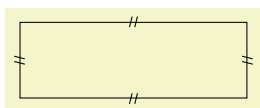
This library extends TikZ’s functionalities around nodes.

Q & A: [10, 19] & [36, 45]

### 11.1 Pic as a node

`/tikz/ext/pic=<boolean>` (default true, initially false)  
pre 0.6 `/tikz/pic`

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



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

### 11.2 Node Pictures

A Node Picture is a mix of a pic, a PGFmatrix and nesting TikZ pictures as well as using the fit library while sharing advantages and disadvantages with all of these:

Feature	nest	ref out	ref in	trans	inherit	placing
PGF matrix	–	–	✓	–	†	✓
TikZ pic	✓	✓	✓	✓	✓	†
Nested TikZpicture	✓	†	†	✓	✓	✓
fit	†	✓	✓	–	–	–
Node Picture	✓	–	✓	✓	†	✓

In this table, “✓” means that the functionality is supported by the feature, “–” means it is not and “†” means the functionality is supported but with some caveat. The functionalities are:

**nest** whether the feature can be nested

PGF matrices cannot be nested. Nodes that are fitted don’t actually contain the nodes they are fitted to.

**ref out** whether one can reference coordinates/nodes outside of a feature

It is not possible to reference a node in the outer TikZpicture from inside a matrix. Referencing node from the outer scope while inside a Node Picture can be done if using the absolute key (see below).

TikZpictures need remember picture to be able to reference a node in the outer TikZpicture.



**ref in** whether one can reference coordinates/nodes from outside of a feature

TikZpictures need remember picture to be able to reference a node in the inner TikZpicture.

**trans** whether the feature can be transformed

A PGF matrix can only be shifted, a fitted node can be transformed but its “content” will remain where it has been placed.

**inherit** whether settings from outside the feature gets inherited

This is usually something one has to fight against and can be quite messy to deal with. Only the pgfmatrix allows some protection against it by using the /tikz/every outer matrix style. The content a node is fitted around doesn’t inherit any settings.

**placing** whether a feature can be placed well

A TikZ pic doesn’t provide any anchors and can’t be placed other than at the origin but it can be placed along a path. A fitted node surely can be placed somewhere else but then it doesn’t fit anymore. Once its “content” is placed, it can’t be moved anymore.

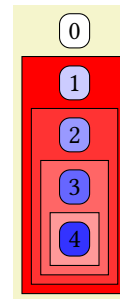
### 11.2.1 Overview

The Node Picture shall offer a compromise between a TikZ pic and a PGF matrix without having to nest TikZpictures. It also shares the visual output with a node that has been fitted around its content but with more flexibility, after all the whole node including its Node Picture can be moved around.

Here are some rules and guidelines for using this:

- The Node Picture will be typeset as the text of the node.
- The baseline of this text is the vertical center of the Node Picture’s bounding box. This can be changed with the ext/node picture/baseline option – similar to TikZ’s /tikz/baseline option. But unlike TikZ’s baseline it need to not be set inside the Node Picture.
- Only the text node part is available. The \nodepart command may not be used.
- A Node Picture is *not* a matrix. It only contains one “cell”.

- Just like with a pic or a matrix, certain settings and values will be inherited by the Node Picture. (It’s a gift and a curse.)
- The bounding box of the picture will be evaluated automatically and the result will be used to determine the size of the “text” of the node. The key ext/node picture/ignore line width changes whether line widths contribute to the size. If they do not and with any inner seps set to zero, paths of the (sub)picture can lie on the border of the node.
- You can use a Node Pictures inside a Node Picture.



```
\usetikzlibrary {ext.nodes}
\tikz[
  nodes=draw,
  do/.code={
    \node[
      rounded corners,
      fill=blue!\interval{\tikzextnodepicture0*2}]
      (L) {\tikzextnodepicture};
    \ifnum\tikzextnodepicture<#1
      \node[
        anchor=north, yshift=-1mm, ext/node picture,
        fill=red!\interval{100-\tikzextnodepicture0*2}]
        at (L.south) {\tikzset{do=#1}};
    \fi
  },
  do=4]{}

```

Most of the time, all that is needed is to add /tikz/ext/node picture to a node and it will be a Node Picture node. (That key’s default value is true which is actually the key that activates it.) Let’s look at further keys and tools for this.

### 11.2.2 Styles

**/tikz/ext/every node picture=<Node Picture level>** (style, no default)

This style will be applied for every node that is enabled to be a Node Picture’s node, this will be executed on the same level as /tikz/every node, i.e. not inside the actual Node Picture.

Its parameter is the current Node Picture level.

**/tikz/ext/every outer node picture=<Node Picture level>** (style, no default)

This style will be applied for every node that is enabled to be a Node Picture's node but it will be executed after the Node Picture has been constructed. This can be used to change options that don't get inherited by the Node Picture.

Its parameter is the current Node Picture level.

`/tikz/ext/every node picture scope=<Node Picture level>` (style, no default)

This style will be applied on the scope that encompasses the whole Node Picture.

Its parameter is the current (outer) Node Picture level.

### 11.2.3 Settings

`/tikz/ext/node picture=<true or false or any of the options below>` (default true)

This is only a key to set keys in the namespace `/tikz/ext/node picture`. However, without any options used, it activates a Node Picture as it uses true as the default value. To return to a normal node, use the argument false.

`/tikz/ext/node picture/prefix name=<true or false>` (default true, initially true)

By default, every node's name inside a Node Picture gets the Node Picture's outer node's name prefixed – similar to how a TikZ pic works. This can be disabled by setting this to false.

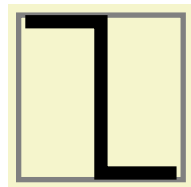
It is not possible to set the Node Picture's name to an empty value.

`/tikz/ext/node picture/name prefix ..` (no value)

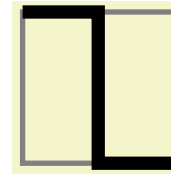
Similar to a TikZ pic, `/tikz/name prefix` is set to the name of the outer node. This key will return to the prefix that was set beforehand.

`/tikz/ext/node picture/ignore line width=<true or false>` (default true)

This changes whether line widths contribute to the size of a Node Picture.



```
\usetikzlibrary {ext.nodes}
\begin{tikzpicture}
  \node[
    ext/node picture,
    inner sep=+0pt, line width=2pt, draw=gray]{
    \draw[line width=5pt, black]
      (-1,1) -| (0,0) |- (1,-1);
  };
\end{tikzpicture}
```



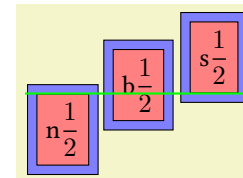
```
\usetikzlibrary {ext.nodes}
\begin{tikzpicture}
  \node[
    ext/node picture={true, ignore line width},
    inner sep=+0pt, line width=2pt, draw=gray]{
    \draw[line width=5pt, black]
      (-1,1) -| (0,0) |- (1,-1);
    };
\end{tikzpicture}
```

`/tikz/ext/node picture/reset graphic state` (no value)

This key sets up `/tikz/ext/every node picture` so that it resets many settings and values to their PGF/TikZ default.

`/tikz/ext/node picture/baseline=<dimension or coordinate>` (default empty)

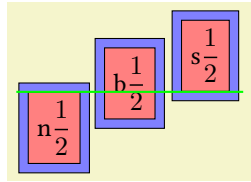
This changes where the baseline of the Node Picture's node lies. This must be used *outside* of a Node Picture to have an effect on that Node Picture. A coordinate must be enclosed in parenthesis, like in the following example.



```
\usetikzlibrary {ext.nodes}
\begin{tikzpicture}
  \foreach[count=\i] \b/\c in {n/orth, b/ase, s/outh}
  \node[
    ext/node picture={true, baseline=(n.\b\c)},
    anchor=base, draw, fill=blue!50
  ] at (right:\i) {
    \node[fill=red!50, draw] (n)
      {\b$\displaystyle\frac{1}{2}$};
  };
  \draw[thick, green] (.5,0) -- (3.5,0);
\end{tikzpicture}
```

`/tikz/ext/node picture/anchor=<coordinate>` (no default, initially empty)

Similar to `/tikz/matrix anchor` this allows to specify any coordinate in the Node Picture's coordinate system to be used as an anchor for the node. Like its inspiration, the parentheses aren't to be included in the `<coordinate>` specification. Unlike its inspiration, any coordinate specification is allowed, not only named nodes/coordinates.

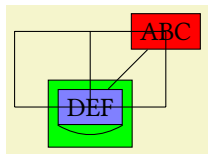


```
\usetikzlibrary {ext.nodes}
\begin{tikzpicture}
\foreach[count=\i] \b/\c in {n/orth, b/ase, s/outh}
\begin{node[
  ext/node picture={true, anchor=n.\b\c},
  draw, fill=blue!50
] at (right:\i) {
  \node[fill=red!50, draw] (n)
  {\b$\displaystyle\frac{1}{2}$};
};
\draw[thick, green] (.5,0) -- (3.5,0);
\end{tikzpicture}
```

`/tikz/ext/node picture/absolute=<coordinate>` (default 0,0)

This applies a transformation so that the `<coordinate>` inside the Node Picture lies at the same `<coordinate>` in the outer scope. This simply sets both `/tikz/ext/node picture/anchor` and `/tikz/at` to the same `<coordinate>`.

This also may make it possible to successfully reference nodes outside of a Node Picture unless you add further transformation to the outer node. You probably also want to use `overlay` on any connections.



```
\usetikzlibrary {ext.nodes}
\begin{tikzpicture}[nodes=draw]
\node[fill=red] at (2,1) (ABC) {ABC};
\node[ext/node picture={true, absolute}, fill=green]{
  \node[fill=blue!50] (DEF) at (1,0) {DEF}
  edge[overlay] (ABC);
  \draw (DEF.south west) to[bend
  right] (DEF.south east);
};
\draw (0,0) grid (2,1);
\end{tikzpicture}
```

#### 11.2.4 More and Examples

A  $\text{\TeX}$ -macro is available for accessing the current Node Picture level.

`\tikzextnodepicture`

This command returns the current level of Node Picture. The outside (i. e. the original `TikZpicture`) is level 0. Remember that when this command is used on – not in – the node that contains the Node Picture it will not return the level of its picture.

A Node Picture also supports two predefined nodes.

#### Predefined node `current` bounding box

Just like the original `current` bounding box, this will be available inside a Node Picture and refers to its bounding box.

#### Predefined node `ext_current` node picture bounding box

Similar to `current` bounding box this returns the bounding box of the Node Picture with an ignored line width.

Further more, anchors can be added to a Node Picture's node.

`/tikz/ext/node picture/add anchor={<n>}{<c>}` (no default)

This will add the anchor `<n>` to the Node Picture's node at `<c>` which will evaluated at the end of the Node Picture.

For rectangular nodes, these keys might be helpful as well.

`/tikz/ext/node picture/add anchor/rectangle east={<n>}{<c>}` (no default)

This key works like the previous key but it also shifts the anchor by the amount of `inner xsep` and `outer xsep` to the right.

`/tikz/ext/node picture/add anchor/rectangle west={<n>}{<c>}` (no default)

As above but to the left.

`/tikz/ext/node picture/add anchor/rectangle north={<n>}{<c>}` (no default)

This key works like the previous keys but it shifts the anchor by the amount of `inner ysep` and `outer ysep` upwards.

`/tikz/ext/node picture/add anchor/rectangle south={<n>}{<c>}` (no default)

As above but downwards.

Those are special versions of these more general keys for which the given factors must be purely numerical as no evaluation are done for them.

`/tikz/ext/node picture/add anchor osep={<n>}{<c>}{<ox>}{<osep y>}` (no default)

Thus key works like the `add anchor` key but it shifts the anchor by the `outer xsep` and `outer ysep` values multiplied by the given factors `<ox>` and `<oy>`.

`/tikz/ext/node picture/add anchor osep={⟨n⟩}{⟨c⟩}{⟨ix⟩}{⟨iy⟩}` (no default)

As above but with the values of inner xsep and inner ysep.

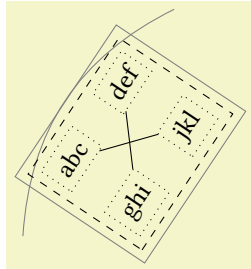
`/tikz/ext/node picture/add anchor sep={⟨n⟩}{⟨c⟩}{⟨ix⟩}{⟨iy⟩}{⟨ox⟩}{⟨oy⟩}`  
(no default)

This is a combination of the last two keys. This is the most general version to add anchors.

A PGF matrix alone is not able to be transformed but as part of this construct, it is possible. As you can see in the next example, you can reference the nodes defined inside the (sub)picture from the outside.

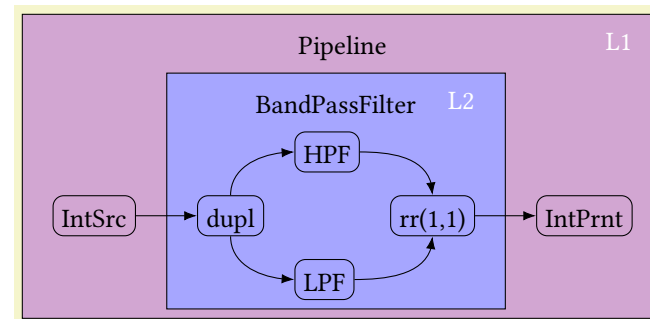
Unfortunately, due to how a TikZ matrix sets up the naming of its nodes, a name prefix would be applied twice (see [56]), you'll have to *first* name the matrix and *then* clear the name prefix.

The example below also shows that all nodes include their own inner sep: The outer node (solid line), the matrix node (dashed line) as well as the nodes in the matrix themselves (dotted lines).



```
\usetikzlibrary {ext.nodes, matrix}
\begin{tikzpicture}
\draw (0,0) to[bend left]
node[ext/node picture, sloped, draw, below] (n) {
  \matrix[
    matrix of nodes, row sep=5mm, column sep=5mm,
    draw, dashed, nodes={draw, dotted},
    name=-m, ext/node picture/name prefix ..
  ] { abc & def \\ ghi & jkl \\ };
\draw (-m-1-1) -- (-m-2-2);
} (2,3) [gray];
\draw (n-m-2-1) -- (n-m-1-2);
\end{tikzpicture}
```

The following example is adapted from [58].



```

\usetikzlibrary {ext.nodes, arrows.meta, positioning, graphs}
\begin{tikzpicture}[
  ext/node picture={prefix name=false, reset graphic state},
  fixed height/.style={text depth=+0pt, text height=+.7\baselineskip},
  n/.style={shape=rectangle, rounded corners, draw, fixed height},
  b/.style={
    shape=rectangle, draw, ext/node picture, inner xsep=4mm,
    fill=blue!\interval{50*(\tikzextnodepicture+1)}!red!35,
  },
  h/.style={above=+.333em of current bounding box, fixed height},
  node distance=3mm and 4mm
]
\NewDocumentCommand{\nbox}{0}{m d() m }{%
  \node[b,#1,style/.expanded=\IfValueT{#3}{ext/node picture/baseline=({#3.base})}]{
    #4
    \node[h]{#2};
    \node[
      white, inner xsep=+0pt,
      below left=0pt and 0pt of current bounding box.north east
    ] {\tikzextnodepicture};
  };
}
\nbox{Pipeline}{
  \node[n]{IS}{IntSrc};
  \nbox[base right=of IS, name=BPF]{BandPassFilter}{dup}{
    \node[n]{dup}{dupl};
    \node[n, above right=of dup]{HPF}{HPF};
    \node[n, below right=of dup]{LPF}{LPF};
    \node[n, below right=of HPF]{rr}{rr(1,1)};
  };
  \node[n, base right=of BPF]{IP}{IntPrnt};
}
\graph[edge=-Latex, use existing nodes]{
  IS -> dup -> [in=180] {HPF[>out= 90, <in= 90],
    LPF[>out=-90, <in=-90]}
    -> [in distance=5mm, out=0] rr -> IP
};
\end{tikzpicture}

```

## 11.3 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).

### 11.3.1 Nodes on Lines

`/tikz/ext/node on line` = *<anchor specification>* (style, default {})

pre 0.6 /tikz/node on line

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.

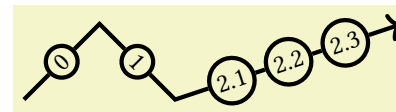
The optional *<anchor specification>* allows to specify the anchors to which the line should connect. It allows one or two anchors divided by and to be specified.

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

pre 0.6 /tikz/nodes on line

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, ->, ext/node on line] (0,0) to["0"] (1,1)
    to["1"] (2,0)
    to[ext/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, ->, ext/node on line=west and east]
  (0,0) to["0"] (1,1)
  to["1"] (2,0)
  to["2"] (4,1);
```

```
\usetikzlibrary {ext.nodes, intersections, quotes, spath3}
\tikz[inner sep=.15em, circle, nodes={draw, green}, sloped, ultra thick]
\draw[->, ext/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);
```

### 11.3.2 Nodes on Curves

The following keys need the intersections and the spath3 [61] 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.

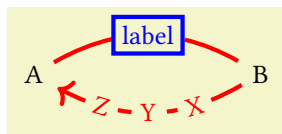
`/tikz/ext/nodes on curve=<to path>` (style, default line to)  
pre 0.6 /tikz/nodes on curve

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

This is not suitable for paths connecting nodes.

`/tikz/ext/nodes on curve'=<to path>` (style, default line to)  
pre 0.6 /tikz/nodes on curve'

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, ->, ext/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}
```



## 11.4 Automatic placement of nodes

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

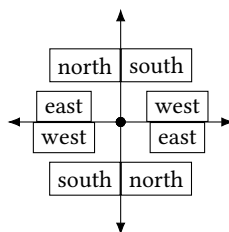
### 11.4.1 More than left and right

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

- `ext/left` will place a node to the left of the direction of the line,
- `ext/right` will place a node to the right of the direction of the line,
- `ext/above` will place a node towards the direction of the line,
- `ext/below` will place a node against the direction of the line,
- `ext/west` will place a node towards the left side of the paper,
- `ext/east` will place a node towards the right side of the paper,
- `ext/north` will place a node towards the upper side of the paper and
- `ext/south` will place a node towards the lower side of the paper.

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

Certain cases exist for `ext/west`, `ext/east`, `ext/north` and `ext/south` placements where it is not clear how a node should be placed. These cases and their behavior can be seen in the following figure.



### 11.4.2 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/ext/auto with offset=<true or false>` (default true)

This key activates the offset function.

`/tikz/ext/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.

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

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

It also makes available the following keys.

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

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

`/tikz/ext/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.

**Implementation note:** This redefines the keys `/tikz/auto`, `/tikz/swap` and `/tikz/sloped`. One can install custom auto placement rules by using the following key.

`/tikz/ext/nodes/install auto={{<left>}}{{<right>}}` (no default)

This key defines `/tikz/auto/<left>` which activates the auto placement and installs the appropriate placement function. Further more, the key `/tikz/swap/<left>` will be defined to activate the `<right>` placement function.

The key `/tikz/swap` has been defined to apply `/tikz/swap/<dir>` where `<dir>` is the current placement function.

### 11.4.3 Precise placement

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

`/tikz/ext/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.

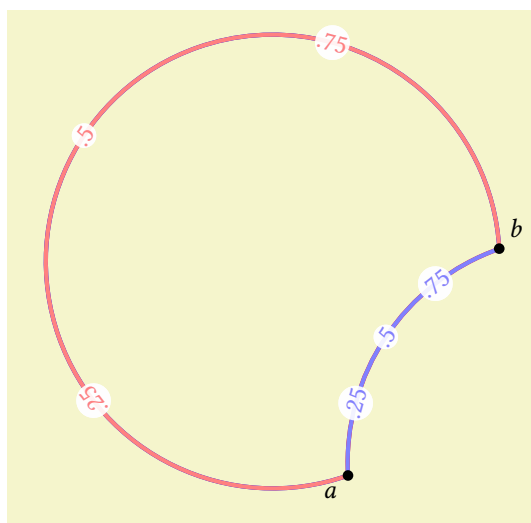


## 12 Arc to a point

**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/ext/arc to/clockwise` (style, no value)

This constructs an arc that goes clockwise.

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

This constructs an arc that goes counter clockwise.

This is the default.

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

This constructs an arc whose angle is larger than  $180^\circ$ .

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

This constructs an arc whose angle is smaller than  $180^\circ$ .

`/tikz/ext/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/ext/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/ext/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/ext/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/ext/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  $\text{\TeX}$ 's numerical weaknesses. The only case it works quite nicely is when the resulting angle is a multiple of  $90^\circ$ .*

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.

## 13 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`.

### 13.1 Timers

New timers are setup for both the Zig-Zag and the Zig-Zig connections that will be introduced in this section. These can be configured through the following keys.

`/tikz/ext/ortho/spacing=<number>` (no default, initially 4)

pre 0.6 `/tikz/ortho/spacing`

For  $\langle number \rangle \geq 2$

- `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  $\langle number \rangle \leq 1$  then

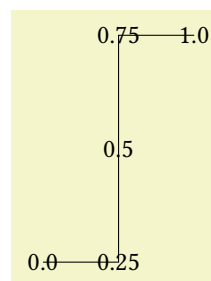
- `pos = -1` will be at the start,
- `pos = 2` will be at the end,
- `pos = 0` will be at the first kink,
- `pos = 1` will be at the second kink and
- `pos = .5` will still be in the middle of the middle part of the connection.

`/tikz/ext/ortho/middle 0 to 1`

(no value)

pre 0.6 `/tikz/ortho/middle 0 to 1`

This is an alias for `spacing = 0`.



```
\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}};
```

## 13.2 Zig-Zag

Similar to the path operations `| -` and `- |` this library adds the path operations `| - |` (“vhv”) and `- | -` (“hvh”).

```
\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”.

Where the middle part (horizontal for `| - |` and vertical for `- | -`) of these path operation end up can be specified by a ratio, a distance or a factor of one base vector of the xy coordinate system.

If used with nodes, the key `from center` toggles from where these are measured.

```
/tikz/ext/ortho/from center=<true or false> (default true)
```

pre 0.6 /tikz/ortho/from center

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 will be calculated from the nodes’ center if this key is set to true.

```
/tikz/ext/ortho/hvh ratio=<ratio> (no default, initially 0.5)
```

```
/tikz/ext/ortho/vhv ratio=<ratio> (no default, initially 0.5)
```

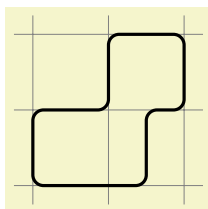
```
/tikz/ext/ortho/hvvh ratio=<ratio> (no default)
```

```
/tikz/ext/ortho/ratio=<ratio> (no default)
```

pre 0.6 /tikz/ortho/ratio

This sets the ratio for the middle part of the `- | -` and/or the `| - |` operation.

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

For specifying a distance or a factor in the xy coordinate system the same option will be used.

```
/tikz/ext/ortho/hvh distance=<distance> (no default)
```

```
/tikz/ext/ortho/vhv distance=<distance> (no default)
```

```
/tikz/ext/ortho/hvvh distance=<distance> (no default)
```

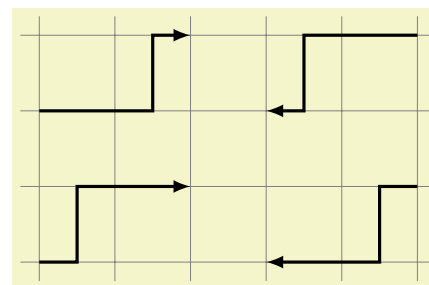
```
/tikz/ext/ortho/distance=<distance> (no default)
```

pre 0.6 /tikz/ortho/distance

If  $\langle distance \rangle$  contains a unit, this will be used as the absolute distance for the middle part of the `- | -` and/or the `| - |` operation. If  $\langle distance \rangle$  doesn’t contain a unit it will be interpreted as the factor for the base y (`| - |`) or x vector (`- | -`) in the xy coordinate system.

This distance is measured from the target coordinate if it’s negative, otherwise from the start coordinate.

The distance option also sets the distance of the Zig-Zig path operations below.



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

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

### 13.3 Zig-Zig

```
\path ... r-ud[<options>](coordinate or cycle) ...;
```

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

```
\path ... r-du[<options>](coordinate or cycle) ...;
```

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

```
\path ... r-lr[<options>](coordinate or cycle) ...;
```

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

```
\path ... r-rl[<options>](coordinate or cycle) ...;
```

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

```
/tikz/ext/ortho/ud distance=<distance>      (no default, initially .5cm)
/tikz/ext/ortho/du distance=<distance>      (no default, initially .5cm)
/tikz/ext/ortho/lr distance=<distance>      (no default, initially .5cm)
/tikz/ext/ortho/rl distance=<distance>      (no default, initially .5cm)
/tikz/ext/ortho/udlr distance=<distance>    (no default)
/tikz/ext/ortho/distance=<distance>        (no default)
```

pre 0.6 These were all in the /tikz/ortho namespace.

If *<distance>* contains a unit, this will be used as the absolute distance for the middle part of the previously introduced path operation. If *<distance>* doesn't contain a unit it will be interpreted as the factor for the base y (ud/du) or x vector (lr/rl) in the xy coordinate system.

The distance is measured from the start, never from the target coordinate.

The distance option also sets the distance of the Zig-Zag path operations above.

## 13.4 Even more Horizontal and Vertical Lines

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

`/tikz/ext/horizontal vertical` (no value)  
`/tikz/ext/vertical horizontal` (no value)

pre 0.6 `/tikz/horizontal vertical`, `/tikz/vertical horizontal`

These install the path operations `-|` or `| -` respectively as to paths that can be used with the path operations `to` or `edge`.

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

pre 0.6 `/tikz/horizontal vertical horizontal`, `/tikz/vertical horizontal vertical`

These installs the operations `-| -` or `| -|` respectively as to paths that can be used with the path operations `to` or `edge`.

`/tikz/ext/up horizontal down=<options>` (no default)  
`/tikz/ext/down horizontal up=<options>` (no default)  
`/tikz/ext/left vertical right=<options>` (no default)  
`/tikz/ext/right vertical left=<options>` (no default)

pre 0.6 These were all in the `/tikz` namespace.

These install the Zig-Zig path operations as to paths 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/ext/only vertical second=<length>` (style, default 0pt)

pre 0.6 `/tikz/only vertical second`

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/ext/only horizontal second=<length>` (style, default 0pt)

pre 0.6 `/tikz/only horizontal second`

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/ext/only vertical first=<length>` (style, default 0pt)

pre 0.6 `/tikz/only vertical first`

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/ext/only horizontal first=<length>` (style, default 0pt)

pre 0.6 `/tikz/only horizontal first`

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

pre 0.6 `/tikz/ortho/install shortcuts`

Installs the following shortcuts:

<code>  -</code>	→ vertical horizontal
<code>- </code>	→ horizontal vertical
<code>-  -</code>	→ horizontal vertical horizontal
<code>  - </code>	→ vertical horizontal vertical
<code>  *</code>	→ only vertical first
<code>* </code>	→ only vertical second
<code>- *</code>	→ only horizontal first
<code>* -</code>	→ only horizontal second
<code>r-ud</code>	→ up horizontal down
<code>r-du</code>	→ down horizontal up
<code>r-lr</code>	→ left vertical right
<code>r-rl</code>	→ right vertical left

## 14 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] & [42, 54]

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.

### 14.1 Rectangle

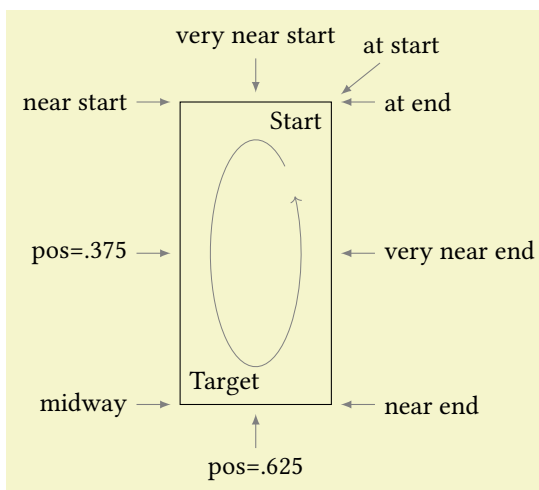
For the `rectangle` path operator, the timer starts with `pos = 0` (= at start) from the starting coordinate in a counter-clockwise direction along the rectangle. The corners will be at positions 0.0, 0.25, 0.5, 0.75 and 1.0.

`/tikz/ext/rectangle timer=`line or rectangle

(no default)

pre 0.6 `/tikz/rectangle timer`

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

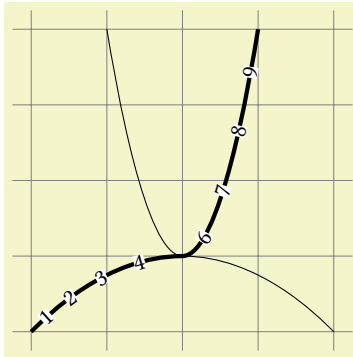


```
\usetikzlibrary {ext.paths.timer}
\begin{tikzpicture}[scale=2, every pin edge/.style={latex-, gray}]
\coordinate [label=above right:Target] (A) at (0,0);
\coordinate [label=below left:Start] (B) at (1,2);
\draw[->, help lines] ([shift=(50:.3 and .75)] .5,1)
  arc[start angle=50, delta angle=340, 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}
```

## 14.2 Parabola

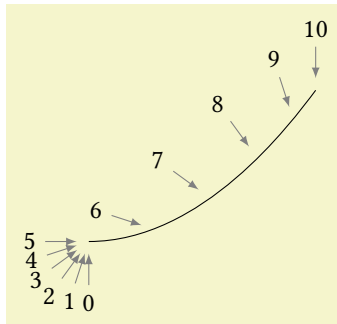
For the parabola path operator the timer is similar to the `.. controls ..` operator.

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.

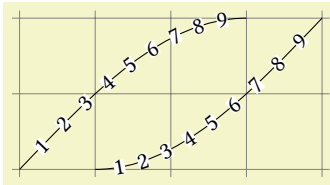


```
\usetikzlibrary {ext.paths.timer}
\begin{tikzpicture}[every pin edge/.style={latex-, shorten <=1pt, gray}]
\draw (-2,-2) parabola (1,0)
  foreach \pos in {0, 1, ..., 10} {
    node [pos=\pos/10, pin={\anchor=-18*\pos+90-18*\pos+270:\pos}]{}
  };
\end{tikzpicture}
```

## 14.3 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}
```

## 15 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:** [18] & [53]

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”.

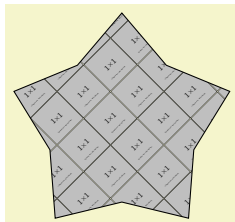
```
\tikzextsetupimageaspattern[options]{name}{image}
```

pre 0.6 \pgfsetupimageaspattern

```
/tikz/ext/image as pattern=options
```

(default {})

pre 0.6 /tikz/image as pattern



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

```
/tikz/ext/image as pattern/name=name
```

(no default)

pre 0.6 /tikz/image as pattern/name

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

```
/tikz/ext/image as pattern/option
```

(style, no value)

pre 0.6 /tikz/image as pattern/*option*

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

```
/tikz/ext/image as pattern/options=style
```

(no default)

pre 0.6 /tikz/image as pattern/*options*

Appends to style /tikz/ext/image as pattern/*option*.

## 16 Positioning Plus

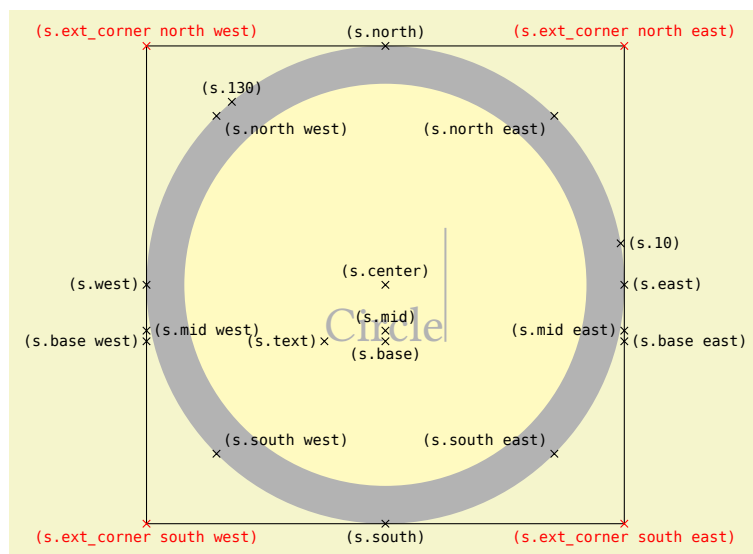
**TikZ Library** `ext.positioning-plus`

```
\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.

### 16.1 Useful corner anchors

The anchors `ext_corner north east`, `ext_corner north west`, `ext_corner south west` and `ext_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.



```
\usetikzlibrary {ext.positioning-plus}
\Huge
\begin{tikzpicture}
\node[name=s,shape=circle,shape example,scale=.75,outer sep=auto]
{Circle\vrule width 1pt height 2cm};
\foreach \anchor/\placement in {
north west/below right, north/above, north east/below left,
west/left, center/above, east/right,
mid west/right, mid/above, mid east/left,
base west/left, base/below, base east/right,
south west/above right, south/below, south east/above left,
text/left, 10/right, 130/above}
\draw[node font=\scriptsize, shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
node[\placement] {\texttt{(s.\anchor)}};
\draw (s.ext_corner north west) rectangle (s.ext_corner south east);
\foreach \anchor/\placement in {
corner north west/above, corner north east/above,
corner south west/below, corner south east/below}
\draw[node font=\scriptsize, red, shift=(s.ext_\anchor)] plot[mark=x] coordinates{(0,0)}
node[\placement] {\texttt{(s.ext\textunderscore\anchor)}};
\end{tikzpicture}
```

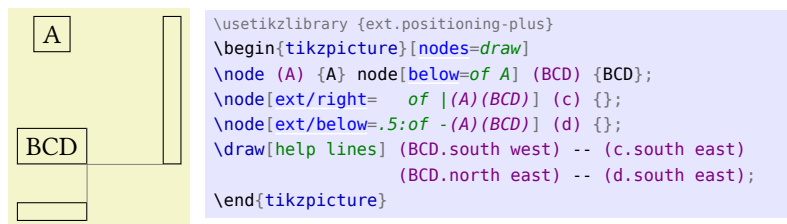
## 16.2 Useful placement keys for vertical and horizontal alignment

<code>/tikz/ext/left=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/right=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/above=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/below=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/above left=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/below left=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/above right=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/below right=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/mid left=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/mid right=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/base left=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/base right=&lt;specification&gt;</code>	(default 0pt)

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.



As you may have 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.<sup>5</sup>

The fitting functionality is also available without the placement.

<code>/tikz/ext/fit bounding box=&lt;list of coordinates&gt;</code>	(no default)
<code>/tikz/ext/span vertical=&lt;list of coordinates&gt;</code>	(no default)
<code>/tikz/ext/span horizontal=&lt;list of coordinates&gt;</code>	(no default)
<code>/tikz/ext/span=&lt;list of coordinates&gt;</code>	(no default)

These all create a rectangular node with the name `ext_fit` bounding box that encompasses the *<list of coordinates>*.

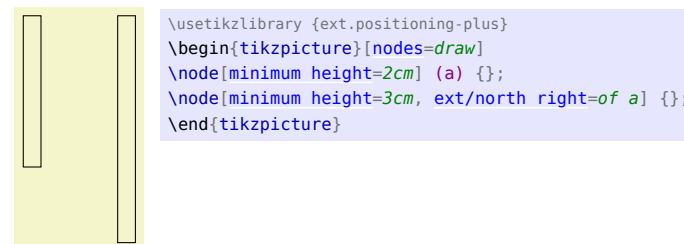
The `span vertical` key will also set `/pgf/minimum height` to the height of this bounding box

The `span horizontal` key will also set `/pgf/minimum width` to the width of this bounding box

The last one combines `span vertical` and `span horizontal`.

<code>/tikz/ext/north left=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/south left=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/north right=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/south right=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/west above=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/west below=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/east above=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/east below=&lt;specification&gt;</code>	(default 0pt)

These work similarly to `left`, `right`, `above` and `below` but they are north- or south-aligned.



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

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

<code>/tikz/ext/corner above left=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/corner below left=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/corner above right=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/corner below right=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/corner north left=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/corner south left=&lt;specification&gt;</code>	(default 0pt)

<code>/tikz/ext/corner north right=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/corner south right=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/corner west above=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/corner west below=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/corner east above=&lt;specification&gt;</code>	(default 0pt)
<code>/tikz/ext/corner east below=&lt;specification&gt;</code>	(default 0pt)

These work the same as above `left`, `below left`, ... but they use the new generic corner anchors

## 17 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, meaning as an option to `\tikzpicture` or `\begin{tikzpicture}`, the size of the picture<sup>6</sup> 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/ext/save picture size` (style, no value)

```
pre 0.6 /tikz/save picture size
```

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

### 17.1 Externalization

As this library usually needs multiple compilations to produce stable pictures it is incompatible with the external library.

However, the library provides support for the `memoize` [65] package. When it is used the arguments to the keys below will be saved as the context of the memo. This means that the arguments need to be a valid `\dimexpr` expression.

### 17.2 Keeping the aspect ratio

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

<sup>6</sup>This is the size of the pseudo-node `current bounding box`.

#### `/tikz/ext/picture width=<dimension>` (no default)

```
pre 0.6 /tikz/picture width
```

Scales the picture so that the width of the picture will be `<dimension>`. This will keep the aspect ratio the same.

#### `/tikz/ext/minimum picture width=<dimension>` (no default)

```
pre 0.6 /tikz/minimum picture width
```

As above but will not change the size of the picture if its width is greater than `<dimension>`.

#### `/tikz/ext/maximum picture width=<dimension>` (no default)

```
pre 0.6 /tikz/maximum picture width
```

As above but will not change the size of the picture if its width is less than `<dimension>`.

#### `/tikz/ext/picture height=<dimension>` (no default)

```
pre 0.6 /tikz/picture height
```

Scales the picture so that the height of the picture will be `<dimension>`. This will keep the aspect ratio the same.

#### `/tikz/ext/minimum picture height=<dimension>` (no default)

```
pre 0.6 /tikz/minimum picture height
```

As above but will not change the size of the picture if its height is greater than `<dimension>`.

#### `/tikz/ext/maximum picture height=<dimension>` (no default)

```
pre 0.6 /tikz/maximum picture height
```

As above but will not change the size of the picture if its height is less than `<dimension>`.

#### `/tikz/ext/minimum picture size={<width>}{<height>}` (no default)

pre 0.6 /tikz/minimum picture size

Scales the picture so that its width will be at least  $\langle width \rangle$  and its height will be at least  $\langle height \rangle$ .

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

pre 0.6 /tikz/maximum picture size

Scales the picture so that its width will be at most  $\langle width \rangle$  and its height will be at most  $\langle height \rangle$ .

### 17.3 Changing the aspect ratio

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

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

pre 0.6 /tikz/picture width\*

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

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

pre 0.6 /tikz/minimum picture width\*

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

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

pre 0.6 /tikz/maximum picture width\*

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

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

pre 0.6 /tikz/picture height\*

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

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

pre 0.6 /tikz/minimum picture height\*

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

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

pre 0.6 /tikz/maximum picture height\*

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

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

pre 0.6 /tikz/picture size\*

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.

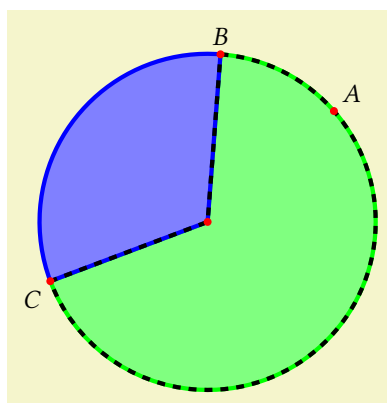
## 18 Arcs through Three Points

**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.

Q & A: [64] & [55]



```
\usetikzlibrary {ext.topaths.arcthrough}
\begin{tikzpicture}[scale=.75]
\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[ext/arc through={clockwise, (A)}] (C) -- (arc through center) -- cycle;
\draw[ultra thick, draw=blue, fill=blue!50]
(B) to[ext/arc through=(A)] (C) -- (arc through center) -- cycle;
\draw[ultra thick, draw=black, dashed]
(B) to[ext/arc through={auto, (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/ext/arc through=<key-value>` (no default)  
pre 0.6 /tikz/arc trough/arc through

This key should be used with `to` or `edge`. A parameter other than the key described below will be assumed to be the through coordinate.

`/tikz/ext/arc through/through=<coordinate>` (no default, initially (0,0))  
pre 0.6 /tikz/arc through/through

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

`/tikz/ext/arc through/center suffix=<suffix>` (no default, initially )  
pre 0.6 /tikz/arc trough/center suffix

The arc through will define a coordinate named `arc through center<suffix>` so that it can be referenced later.

`/tikz/ext/arc through/clockwise` (no value)  
pre 0.6 /tikz/arc trough/clockwise

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

`/tikz/ext/arc through/counter clockwise` (no value)  
pre 0.6 /tikz/arc trough/counter clockwise

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/ext/arc through/auto` (no value)

The resulting arc will go automatically clockwise or counter clockwise so that it goes through the through point.



## 19 Autobending

**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:** [24] & [35]

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/ext/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/ext/autobend south=<angle>` (default last value)

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

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

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

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

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

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

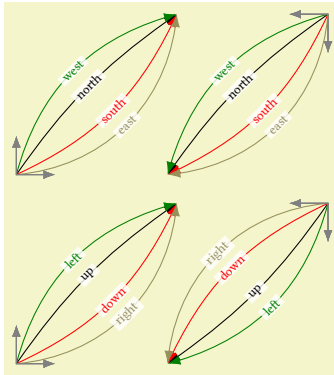
Works like `autobend up` but bends the curve downwards.

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

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

`/tikz/ext/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,
  pics/cs/.style={
    /tikz/transform shape,
    code={\draw[help lines, Latex-Latex] (up:1) |- (right:1);}
  },
  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) pic {cs} to[ext/autobend \d=\i0] 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]
    (45:-3) pic {cs} to[ext/autobend \d=\i0] node{\d} (0,0);

\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) pic {cs} to[ext/autobend \d=\i0] 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]
    (45:-3) pic {cs} to[ext/autobend \d=\i0] node{\d} (0,0);
\end{tikzpicture}

```

## 20 Mirror, Mirror on the Wall

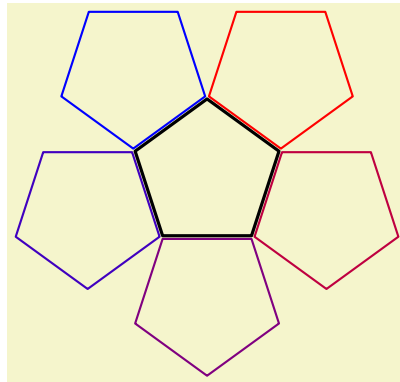
**TikZ Library** `ext.transformations.mirror`

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

This library adds more transformations to TikZ.

As explained in section 22, 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.

### 20.1 Using the reflection matrix



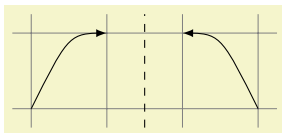
```
\usetikzlibrary {shapes.geometric,ext.transformations.mirror}
\begin{tikzpicture}[line join=round, thick, reg poly/.style={
  shape=regular polygon, regular polygon sides={#1}}]
\node[reg poly=5, minimum size=+2cm, draw, very thick] (a) {};
\foreach \i[evaluate={\col=(\i-1)/.04}] in {1,...,5}
  \node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

`/tikz/ext/xmirror=<value or coordinate>`

(default 0pt)

pre 0.6 /tikz/xmirror

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



```
\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[ext/xmirror=(m),-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`/tikz/ext/ymirror=<value or coordinate>`

(default 0pt)

pre 0.6 /tikz/ymirror

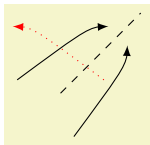
Sets up a transformation that mirrors along a vertical line that goes through point  $(0, \langle value \rangle)$  or  $\langle coordinate \rangle$ .

`/tikz/ext/mirror x=<coordinate>`

(default  $(0,0)$ )

pre 0.6 /tikz/mirror x

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

`/tikz/ext/mirror y=<coordinate>`

(default  $(0,0)$ )

pre 0.6 /tikz/mirror y

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

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

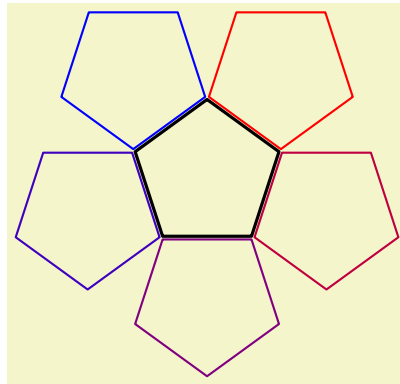
(no default)

pre 0.6 /tikz/mirror

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.

## 20.2 Using built-in transformations



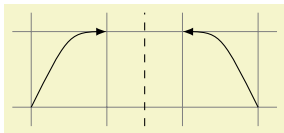
```
\usetikzlibrary {shapes.geometric,ext.transformations.mirror}
\begin{tikzpicture}[line join=round, thick, reg poly/.style={
  shape=regular polygon, regular polygon sides={#1}}]
\node[reg poly=5, minimum size=+2cm, draw, very thick] (a) {};
\foreach \i[evaluate={\col=(\i-1)/.04}] in {1,...,5}
  \node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

`/tikz/ext/xMirror=<value or coordinate>`

(default 0pt)

pre 0.6 /tikz/xMirror

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



```
\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[ext/xMirror=(m),-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`/tikz/ext/yMirror=<value or coordinate>`

(default 0pt)

pre 0.6 /tikz/yMirror

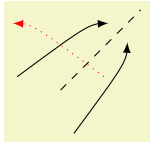
Sets up a transformation that mirrors along a vertical line that goes through point (0,  $\langle value \rangle$ ) or  $\langle coordinate \rangle$ .

`/tikz/ext/Mirror x=<coordinate>`

(default (0,0))

pre 0.6 /tikz/Mirror x

Similar to 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[ ext/xMirror=(m), -latex, red, dotted] (0,0) .. controls (.5,1) .. (1,1);
\draw[ext/Mirror_x=(m), -latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`/tikz/ext/Mirror y=<coordinate>`

(default (0,0))

pre 0.6 /tikz/Mirror y

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

`/tikz/ext/Mirror=<point A>--<point B>`

(no default)

pre 0.6 /tikz/Mirror

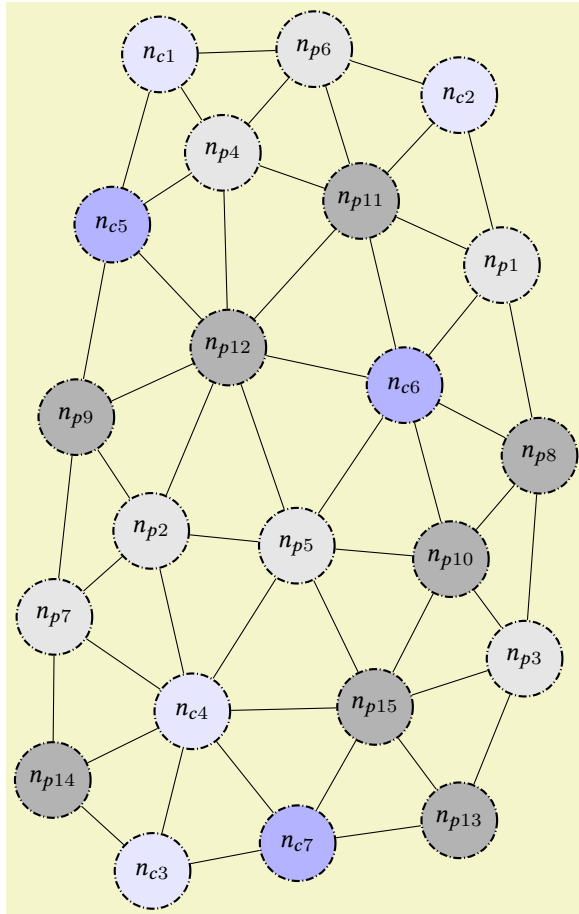
Sets up a transformation that mirrors along a line that goes through <point A> and <point B>.

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/ext/if=c#1
      {/utils/ext/ifnum={#2<5}{blue!light}{blue!dark}}
      {/utils/ext/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};
};
```

## 21 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, 20] & [46, 33, 51, 44, 30]

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. Overtipped<sup>7</sup>

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		–	–

<sup>7</sup>The Overtipped arrow tips aren't yet implemented.



## 21.1 Centered

### 21.1.1 Barbed Arrow Tips

#### **Arrow Tip Kind** ext\_Centered Arc Barb

pre 0.6 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** ext\_Centered Bar

pre 0.6 Centered Bar

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

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

#### **Arrow Tip Kind** ext\_Centered Bracket

pre 0.6 Centered Bracket

This is a variant of the Bracket tip and therefore an instance of the ext\_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** ext\_Centered Hooks

pre 0.6 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** ext\_Centered Parenthesis

pre 0.6 Centered Parenthesis

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

#### **Arrow Tip Kind** ext\_Centered Straight Barb

pre 0.6 Centered Straight Barb

A variant of the Straight Barb tip.

#### **Arrow Tip Kind** ext\_Centered Tee Barb

pre 0.6 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.

### 21.1.2 Geometric Arrow Tips

#### **Arrow Tip Kind** ext\_Centered Circle

pre 0.6 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** ext\_Centered Diamond

pre 0.6 Centered Diamond

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

#### **Arrow Tip Kind** ext\_Centered Ellipse

pre 0.6 Centered Ellipse

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

#### **Arrow Tip Kind** ext\_Centered Kite

pre 0.6 Centered Kite

A variant of the Kite tip.

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

#### **Arrow Tip Kind** ext\_Centered Rectangle

pre 0.6 Centered Rectangle

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

#### **Arrow Tip Kind** ext\_Centered Square

pre 0.6 Centered Square

A variant of the Square tip.

#### **Arrow Tip Kind** ext\_Centered Stealth

pre 0.6 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** ext\_Centered Triangle

pre 0.6 Centered Triangle

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

### **Arrow Tip Kind** ext\_Centered Turned Square

pre 0.6 Centered Turned Square

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

### **21.1.3 Special Arrow Tips**

### **Arrow Tip Kind** ext\_Centered Rays

pre 0.6 Centered Rays

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

## **21.2 Untipped**

### **21.2.1 Barbed Arrow Tips**

### **Arrow Tip Kind** ext\_Centered Arc Barb

pre 0.6 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** ext\_Untipped Bar

pre 0.6 Untipped Bar

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

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

### **Arrow Tip Kind** ext\_Untipped Bracket

pre 0.6 Untipped Bracket

This is a variant of the Bracket tip and therefore an instance of the ext\_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** ext\_Untipped Parenthesis

pre 0.6 Untipped Parenthesis

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

### **Arrow Tip Kind** ext\_Untipped Tee Barb

pre 0.6 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.

### **21.2.2 Geometric Arrow Tips**

### **Arrow Tip Kind** ext\_Untipped Circle

pre 0.6 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** ext\_Untipped Ellipse

pre 0.6 Untipped Ellipse

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

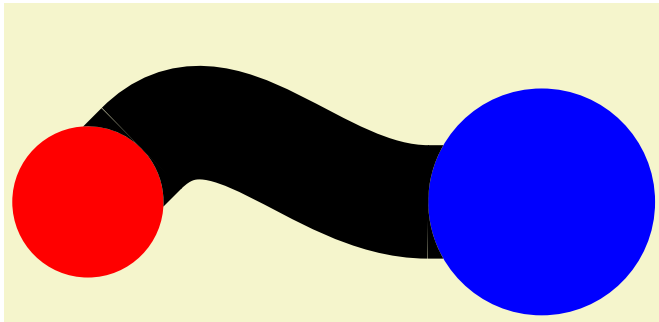
## 21.3 Original Arrow Tips

### Arrow Tip Kind ext\_Hug Cap

pre 0.6 Hug Cap

This arrow tips 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.

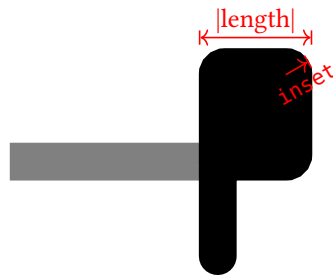


```
\usepgflibrary {ext.arrows}
\begin{tikzpicture}[
  dot/.style 2 args={
    shape=circle, outer sep=+0pt, fill={#1}, minimum size={#2}}]
\node[dot={red} {2cm}] (A) {};
\node[dot={blue}{3cm}] (B) at (6,0) {};
\draw[
  line width=1.5cm,
  arrows={ext_Hug Cap[length=1cm]-ext_Hug Cap[length=1.5cm]}
]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

### Arrow Tip Kind ext\_Loop

pre 0.6 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 inset 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

ext\_Loop[]

ext\_Loop[sep] ext\_Loop[]

ext\_Loop[sep] . ext\_Loop[]

ext\_Loop[open]

ext\_Loop[open, swap]

ext\_Loop[length=5pt,inset=0pt]

ext\_Loop[reversed]

ext\_Loop[slant=.3]

ext\_Loop[red]

0.4pt

— thin

— thin

— thin

— thin

— thin

— thin

— thin

— thin

— thin

0.8pt

— thick

— thick

— thick

— thick

— thick

— thick

— thick

— thick

— thick

1.6pt

—

—

—

—

—

—

—

—

—

The following options have no effect: harpoon, round, line width.

On double lines, the arrow tip will not look correct.

### Arrow Tip Kind ext\_Double Stealth

This arrow tip is similar to the original Stealth, its back is left open so that it aligns neatly to a doubled path.

### Arrow Tip Kind ext\_Double Triangle

This arrow tip is similar to the original Triangle, its back is left open so that it aligns neatly to a doubled path.

### Arrow Tip Kind ext\_Double Cap

This arrow tip closes a doubled line so that it not left open.



```
\usepgflibrary {ext.arrows}
\begin{tikzpicture}
\draw[
  ext_Double Cap-ext_Double Stealth,
  double distance=1cm,
  line width=3mm
] (0,0) to[bend left] (right:9);
\end{tikzpicture}
```

## 22 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.

### 22.1 Using the reflection matrix

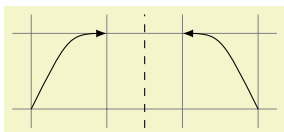
The following commands use the reflection matrix that sets the transformation matrix following

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

`\pgfexttransformxmirror{⟨value⟩}`

pre 0.6 `\pgftransformxmirror`

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



```
\usepgflibrary {ext.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);
\pgfexttransformxmirror{1.5}

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

### 22.2 Using built-in transformations

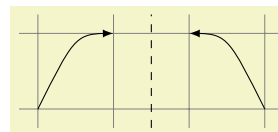
The following commands use a combination of shifting, rotating,  $-1$  scaling, rotating back and shifting back to reach the mirror transformation.

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

`\pgfexttransformxMirror{⟨value⟩}`

pre 0.6 `\pgftransformxMirror`

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



```
\usepgflibrary {ext.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);
\pgfexttransformxMirror{1.5}

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

`\pgfexttransformymirror{⟨value⟩}`

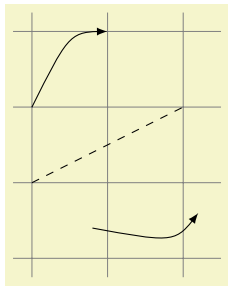
pre 0.6 `\pgftransformymirror`

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

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

pre 0.6 `\pgftransformmirror`

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



```
\usepgflibrary {ext.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);
\pgfexttransformmirror{\pgfpointxy{0}{-1}}
{\pgfpointxy{2}{0}}

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

`\pgfexttransformyMirror{⟨value⟩}`

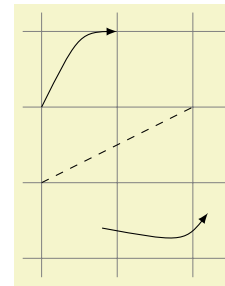
pre 0.6 `\pgftransformyMirror`

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

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

pre 0.6 `\pgftransformMirror`

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



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

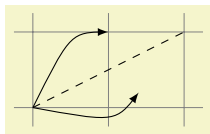
\draw[dashed] (0, -1) -- (2, 0);
\pgfexttransformMirror{\pgfpointxy{0}{-1}}
{\pgfpointxy{2}{0}}

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

`\pgfextqttransformmirror{⟨point A⟩}`

pre 0.6 `\pgfqttransformmirror`

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



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

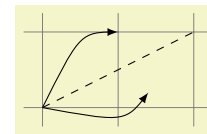
\draw[dashed] (0, 0) -- (2, 1);
\pgfextqttransformmirror{\pgfpointxy{2}{1}}

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

`\pgfextqttransformMirror{⟨point A⟩}`

pre 0.6 `\pgfqttransformMirror`

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



```
\usepgflibrary {ext.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);
\pgfextqttransformMirror{\pgfpointxy{2}{1}}

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

## 23 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: [28] & [48]

**Shape** `ext_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/ext/circle arrow start angle=<start angle>` (no default, initially {})

pre 0.6 /pgf/circle arrow start angle

Sets the start angle.

`/pgf/ext/circle arrow end angle=<end angle>` (no default, initially {})

pre 0.6 /pgf/circle arrow end angle

Sets the end angle.

`/pgf/ext/circle arrow delta angle=<delta angle>` (no default, initially {})

pre 0.6 /pgf/circle arrow delta angle

Sets the delta angle.

`/pgf/ext/circle arrow arrows=<start arrow tip specification>-<end arrow tip specification>` (no default, initially -)

pre 0.6 /pgf/circle arrow arrows

The specification will be forwarded to `\pgfsetarrows`.

A few handful styles are pre-defined.

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

pre 0.6 /pgf/circle arrow turn left north

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

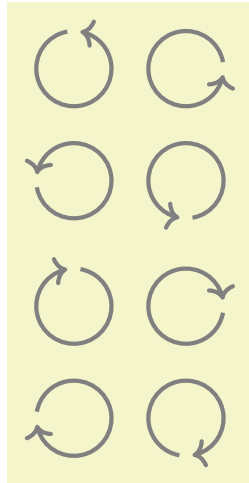
`/pgf/ext/circle arrow turn left east` (no value)

pre 0.6 /pgf/circle arrow turn left east

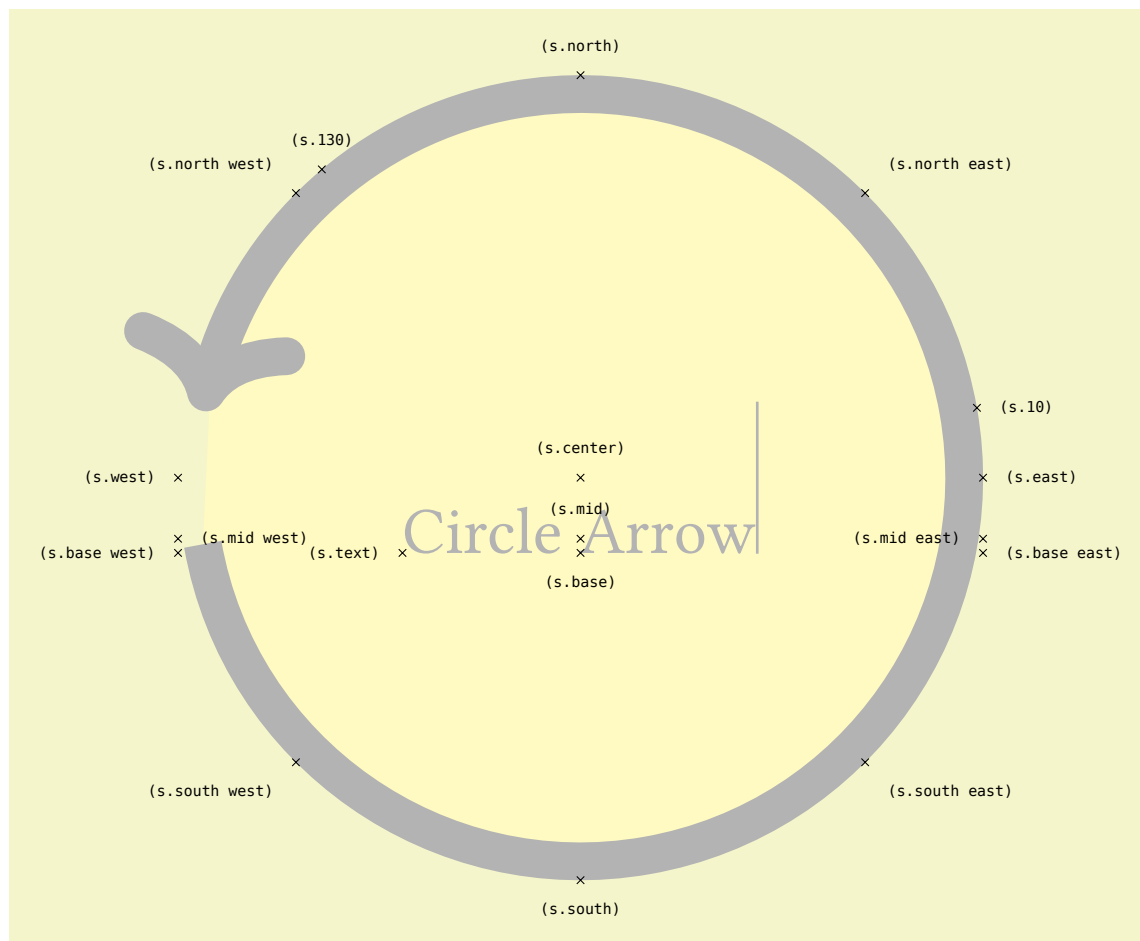
As above but `circle arrow start angle = 10`.

<pre> /pgf/ext/circle arrow turn left west pre 0.6 /pgf/circle arrow turn left west     As above but circle arrow start angle = 280. </pre>	(no value)
<pre> /pgf/ext/circle arrow turn left south pre 0.6 /pgf/circle arrow turn left south     As above but circle arrow start angle = 190. </pre>	(no value)
<pre> /pgf/ext/circle arrow turn right north pre 0.6 /pgf/circle arrow turn right north     Sets circle arrow start angle = 100, circle arrow delta angle = 340 and circle arrow arrows = &lt;-. </pre>	(no value)
<pre> /pgf/ext/circle arrow turn right east pre 0.6 /pgf/circle arrow turn right east     As above but circle arrow start angle = 10. </pre>	(no value)
<pre> /pgf/ext/circle arrow turn right west pre 0.6 /pgf/circle arrow turn right west     As above but circle arrow start angle = 280. </pre>	(no value)
<pre> /pgf/ext/circle arrow turn right south pre 0.6 /pgf/circle arrow turn right south     As above but circle arrow start angle = 190. </pre>	(no value)





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



```
\usetikzlibrary {ext.shapes.circulararrow}
\begin{tikzpicture}\Huge
\node[name=s, shape=ext_circle arrow,
ext/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}
```

## 24 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:** [21] & [49]

**Shape** `ext_circle_cross_split`

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

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

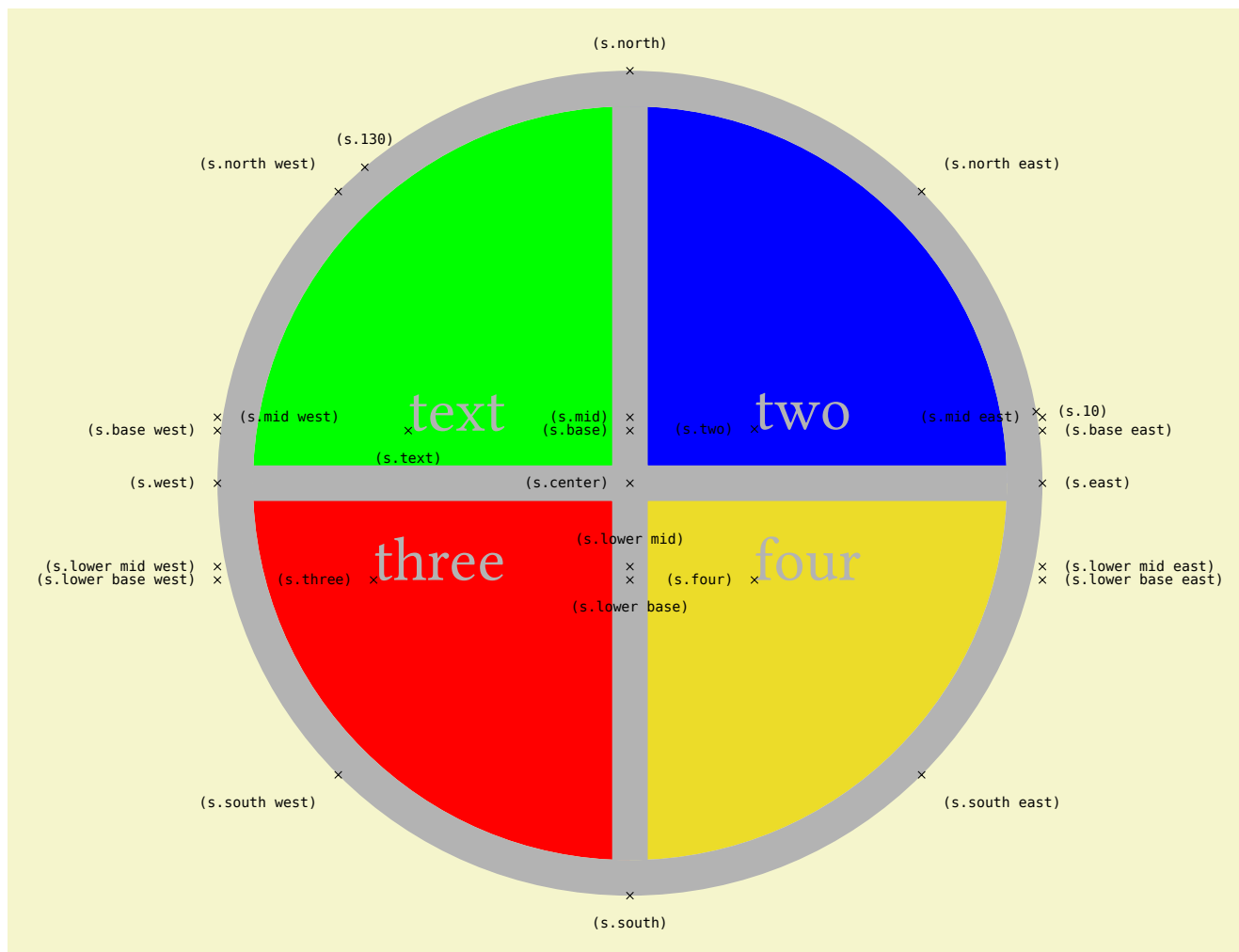
pre 0.6 /pgf/circle cross split part fill

Sets the custom fill color for each node part shape. The items in  $\langle list \rangle$  should be separated by commas (so if there is more than one item in  $\langle list \rangle$ , it must be surrounded by braces). If  $\langle list \rangle$  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/ext/circle cross split uses custom fill=⟨boolean⟩ (default true)
```

pre 0.6 /pgf/circle cross split uses custom fill

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<sub>E</sub>X-if `\ifextpgfcirclecrosssplitcustomfill` appropriately.



```

\usepgflibrary {ext.shapes.circlecrosssplit}
\begin{tikzpicture}\Huge
\node[name=s, shape=ext_circle cross split, shape example, inner xsep=1.5cm, fill=none,
  ext/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}

```

## 25 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] & [39]

**Shape** `ext_heatmark`

<pre>/pgf/ext/heatmark arcs=&lt;arcs num&gt;</pre>	(no default, initially 3)
<pre>pre 0.6 /pgf/heatmark arcs</pre>	
Sets the number of arc around the circle to <i>&lt;arcs num&gt;</i> .	
<pre>/pgf/ext/heatmark arc width=&lt;arc width&gt;</pre>	(no default, initially 4pt)
<pre>pre 0.6 /pgf/heatmark arc width</pre>	
Sets the width of the rings around the circle to <i>&lt;arc width&gt;</i> .	
<pre>/pgf/ext/heatmark arc sep=&lt;sep length&gt;</pre>	(no default, initially 1pt)
<pre>pre 0.6 /pgf/heatmark arc sep</pre>	
Sets the whitespace between the rings to <i>&lt;sep length&gt;</i> .	
<pre>/pgf/ext/heatmark arc rings=&lt;rings num&gt;</pre>	(no default, initially 3)
<pre>pre 0.6 /pgf/heatmark arc rings</pre>	
Sets the number of rings around the circle to <i>&lt;rings num&gt;</i> .	
<pre>/pgf/ext/heatmark arc sep angle=&lt;sep angle&gt;</pre>	(no default, initially 20)
<pre>pre 0.6 /pgf/heatmark arc sep angle</pre>	
Sets the whitespace angle between the arcs in one ring to <i>&lt;sep angle&gt;</i> .	
<pre>/pgf/ext/heatmark inner opacity=&lt;inner opacity&gt;</pre>	(no default, initially 0.8)
<pre>pre 0.6 /pgf/heatmark inner opacity</pre>	
Sets the opacity of the inner ring to <i>&lt;inner opacity&gt;</i> .	
<pre>/pgf/ext/heatmark outer opacity=&lt;low opacity&gt;</pre>	(no default, initially 0.2)
<pre>pre 0.6 /pgf/heatmark outer opacity</pre>	
Sets the opacity of the outer ring to <i>&lt;outer opacity&gt;</i> .	

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/ext/heatmark ring <ring number>`

(style, no value)

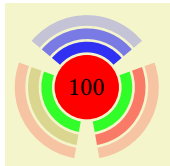
`/pgf/ext/heatmark arc <arc number>`

(style, no value)

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

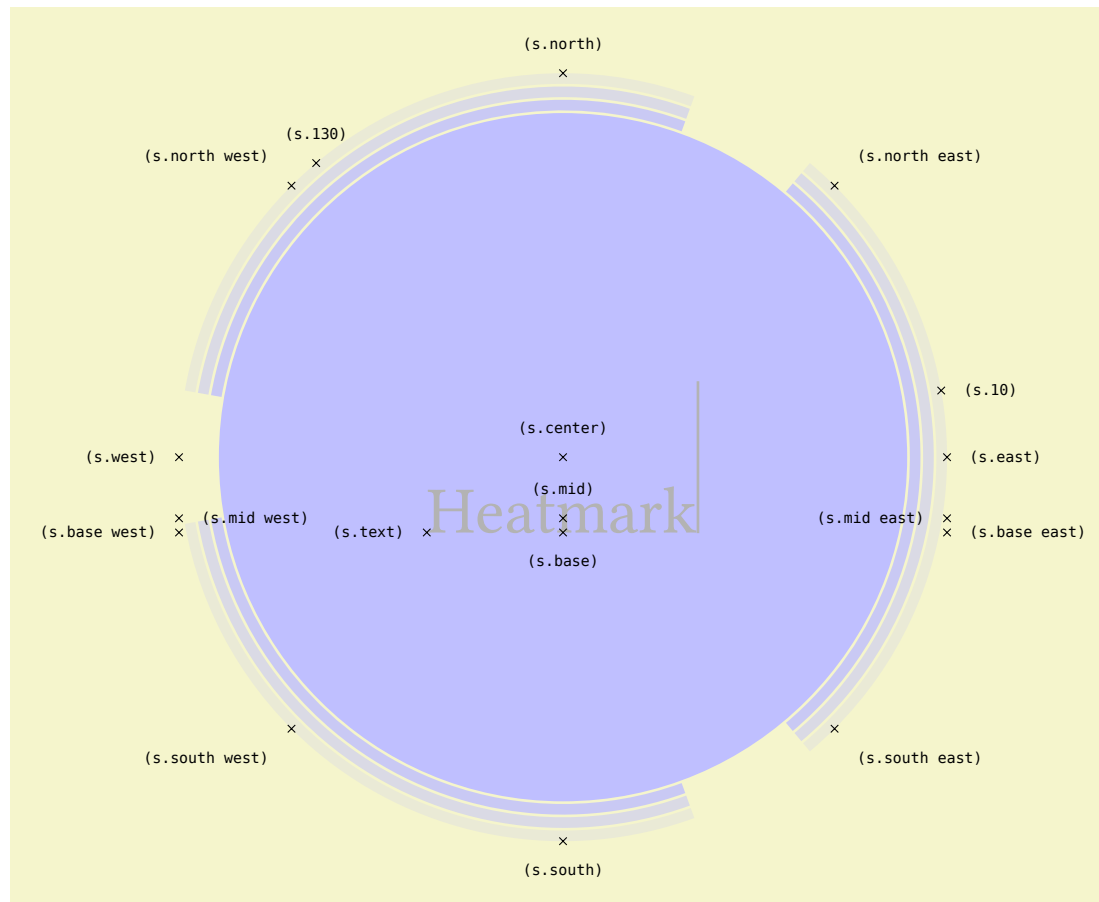
(style, no value)

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/ext/heatmark ring 1/.append style={/tikz/fill=green},
  /pgf/ext/heatmark arc 1/.append style={/tikz/fill=blue},
  /pgf/ext/heatmark ring 2 arc 2/.append style={/tikz/fill=yellow!70!black}
] \node[ext_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=ext_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}
```



## 26 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** `ext_rectangle with rounded corners`

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

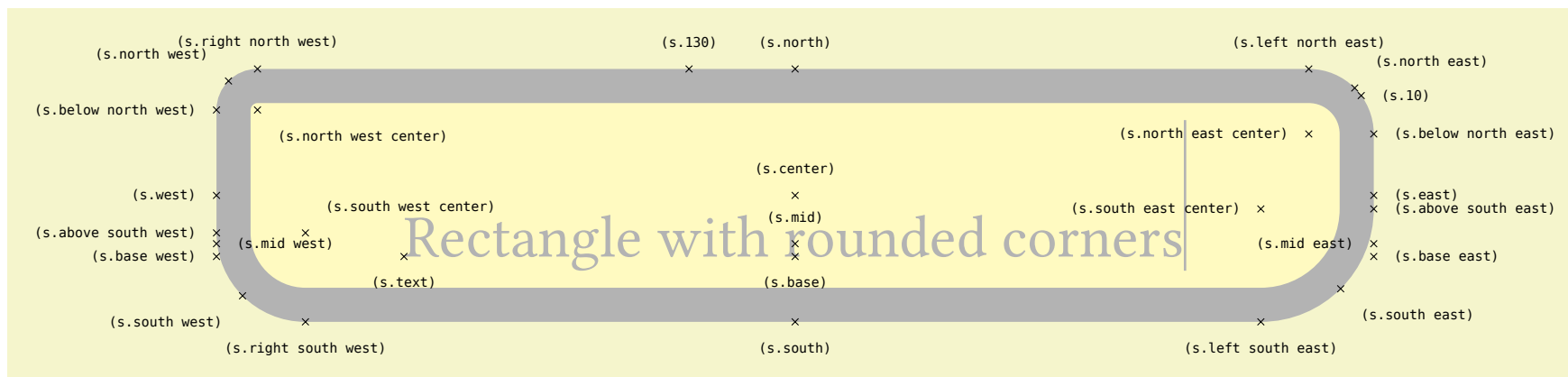
```
/pgf/ext/rectangle with rounded corners north west radius=<dimen> (no default, initially .5\pgflinewidth)
pre 0.6 /pgf/rectangle with rounded corners north west radius
Sets the north west radius to <dimen>.
```

```
/pgf/ext/rectangle with rounded corners north east radius=<dimen> (no default, initially .5\pgflinewidth)
pre 0.6 /pgf/rectangle with rounded corners north east radius
Sets the north east radius to <dimen>.
```

```
/pgf/ext/rectangle with rounded corners south west radius=<dimen> (no default, initially .5\pgflinewidth)
pre 0.6 /pgf/rectangle with rounded corners south west radius
Sets the south west radius to <dimen>.
```

```
/pgf/ext/rectangle with rounded corners south east radius=<dimen> (no default, initially .5\pgflinewidth)
pre 0.6 /pgf/rectangle with rounded corners south east radius
Sets the south east radius to <dimen>.
```

```
/pgf/ext/rectangle with rounded corners radius=<dimen> (no default)
pre 0.6 /pgf/rectangle with rounded corners radius
Sets all radii to <dimen>.
```



```
\usepgflibrary {ext.shapes.rectangleroundedcorners}
\begin{tikzpicture}\Huge
\node[name=s, shape=ext_rectangle with rounded corners, shape example,
  ext/rectangle with rounded corners north west radius=10pt,
  ext/rectangle with rounded corners north east radius=20pt,
  ext/rectangle with rounded corners south west radius=30pt,
  ext/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}
```

## 27 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:** [59] & [32]

**Shape** `ext_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

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

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

`/pgf/ext/superellipse x exponent=<x exponent>`

(no default, initially 2.5)

pre 0.6 /pgf/superellipse x exponent

This sets  $m$ .

`/pgf/ext/superellipse y exponent=<y exponent>`

(no default, initially 2.5)

pre 0.6 /pgf/superellipse y exponent

This sets  $n$ .

`/pgf/ext/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/ext/superellipse exponent=<exponent>`

(no default)

pre 0.6 /pgf/superellipse exponent

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.

`ext_superellipsex( $t$ ,  $2/m$ ,  $r_x$ )`

`\pgfmathextsuperellipse{ $t$ }{ $2/m$ }{ $r_x$ }`

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  $0 \leq t \leq 90$ .

`ext_superellipsey( $t$ ,  $2/n$ ,  $r_y$ )`

`\pgfmathextsuperellipsey{ $t$ }{ $2/n$ }{ $r_y$ }`

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.

`\pgfextmathsuperellipseXY{ $t$ }{ $2/m$ }{ $2/n$ }{ $a$ }{ $b$ }`

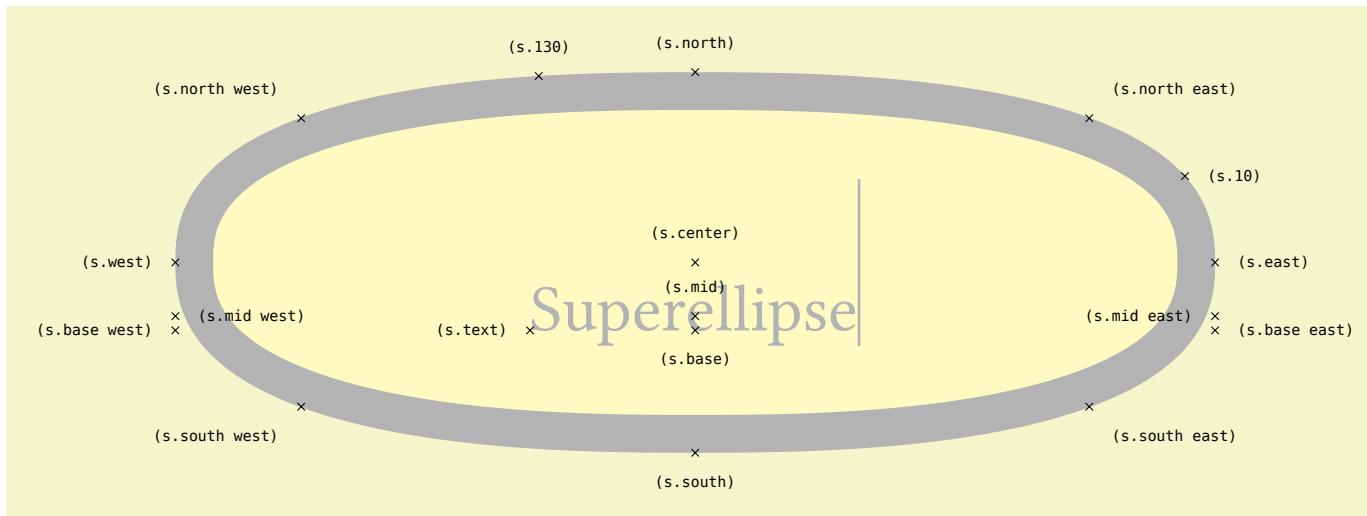
Returns the  $x$  value (in `\pgfmathresultX`) and the  $y$  value (in `\pgfmathresultY`) of the superellipse with its center on the origin following

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

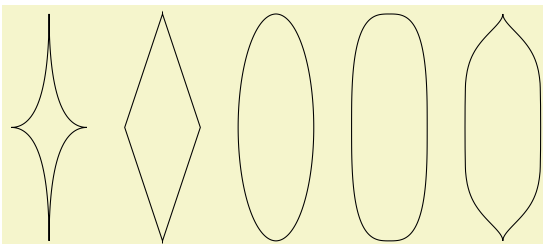
$$y = b \cos^{2/n} t$$

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

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



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



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

## 28 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: [62, 27] & [40, 37]

**Shape** `ext_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/ext/uncentered rectangle center=<left> or <text> or <right> or <real>` (no default, initially `text`)

pre 0.6 /pgf/uncentered rectangle center

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

pre 0.6 /pgf/uncentered rectangle use saved center

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`.

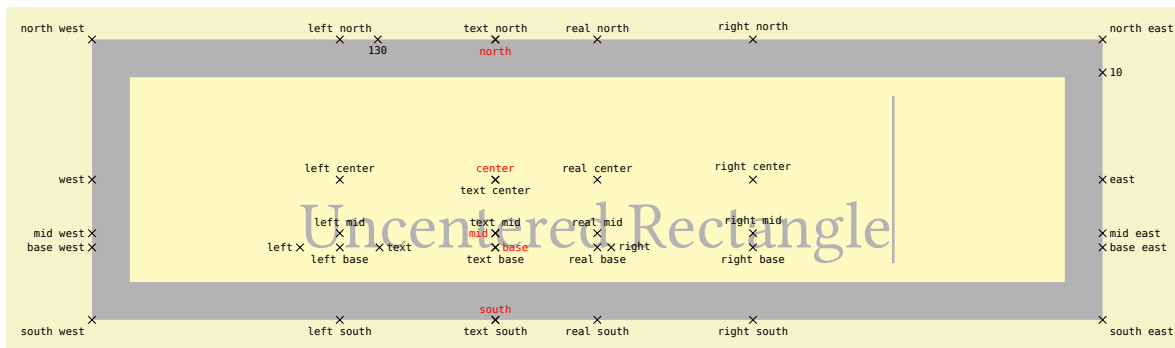
`/pgf/ext/uncentered rectangle center yshift=<dimension>` (no default, initially `{}`)

pre 0.6 /pgf/uncentered rectangle center yshift

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, ext_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}` %  $\text{\LaTeX}$  and plain  $\text{\TeX}$

`\usetikzlibrary[ext.shapes.uncenteredrectangle]` % Con $\text{\TeX}$ t

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

Q: [60]

This library provides only one key.

`/tikz/ext/tikz-cd fix`

(style, no value)

pre 0.6 `/tikz-ext/tikz-cd fix`

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.<sup>8</sup>

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

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

When used as the content of a `ext_uncentered rectangle` shape, 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 `ext_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 `ext_uncentered rectangle` and `/pgf/ext/uncentered rectangle center` to its argument.

`/tikz/commutative diagrams/install uncentered rectangle in column=<column>`

(style, no default)

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

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

```
\usetikzlibrary {cd, ext.shapes.uncenteredrectangle}
\tikzextset{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_{\%_1} \dlar \\
& & C_{\%_2} & \\
\C{2} \urar & & & \uncrec{m_{r_2}}{} = \C{1}-C_{\%_2} \ular \\
\end{tikzcd}
```

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

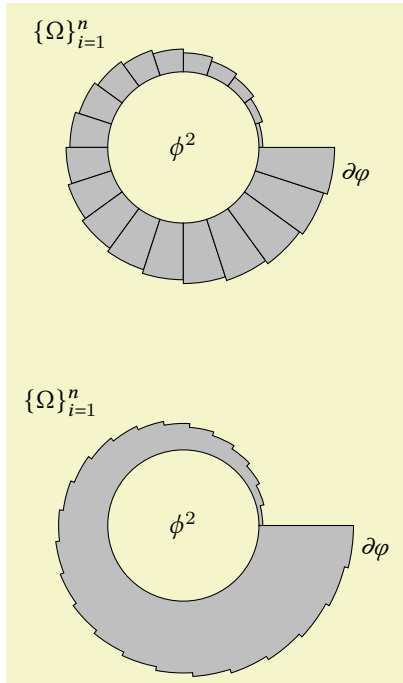
```
\usetikzlibrary {cd, ext.shapes.uncenteredrectangle}
\tikzextset{tikz-cd fix}
\begin{tikzcd}[install uncentered rectangle in column/.list={1,2}]
  \uncrec{S \supset {} }{U_{\tau}}{} & & \arrow[r, "\varphi_0"] \\
  & & \arrow[d, "\tau", "\sim"] \\
  \uncrec{U_{\pi}}{} \subset T & & \arrow[d, "\pi",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

<sup>8</sup>Due 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;},
  ext/declare constant={smallR=1; segments=20;},
  ext/full arc=segments]
\foreach \iN[evaluate={\endRadius=bigR(\iN+1);}, ext/use int=0 to segments-1]
\filldraw[fill=gray!50] (\iN R:\endRadius)
  arc [radius=\endRadius, start angle=\iN R, delta angle=+1R] -- (\iN R+1R:smallR)
  arc [radius=smallR, end angle=\iN R, delta angle=-1R] -- cycle;

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

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

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

## 29 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, 17] & [31, 52, 38]

### 29.1 Extensions

The following tests are added. In version pre 0.6, they're missing the prefix ext/.

- **ext/Jan** This test is passed by all dates that are in the month of January.
- **ext/Feb** as above.
- **ext/Mar** as above.
- **ext/Apr** as above.
- **ext/May** as above.
- **ext/Jun** as above.
- **ext/Jul** as above.
- **ext/Aug** as above.
- **ext/Sep** as above.
- **ext/Oct** as above.
- **ext/Nov** as above.
- **ext/Dec** as above.
- **ext/leap year=<year>** This test checks whether the given year is a leap year. If <year> is omitted, it checks the year of the current date.
- **ext/and={<tests>}** This test passes when all <tests> pass.
- **ext/not={<tests>}** This test passes when <tests> do not pass.
- **ext/week of month=<num>** This test passes when the date is in <num>th week of the month. The first week of the month start at day 1 and ends with day 7.

- **ext/week of month'=<num>** As above but counts from the last day of the month. For a month with 31 days, this means the “1st” week starts at day 25 and ends with day 31.
- **ext/calendar week of month=<num>** This test passes when the date is in <num>th calendar week of the month. The first week starts at the first day of the month and ends at the next Sunday.
- **ext/calendar week of month'=<num>** As above but counts from the last day of the month.

```

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
```

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

- **ext/yesterday={<tests>}** This test passes when the previous day passes <tests>.
- **ext/week=<num>** This test passes when the current week of the year equals <num>.

The shorthands for d- and m- are slightly changed so that they are expandable. This makes it possible to use these shorthands inside of pgfmath. The shorthands for the week (see section 29.2) are added. These are

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

## 29.2 Week numbering (ISO 8601)

`\pgfextcalendarjulianyearthoweeek{<Julian day>}{<year>}{<week count>}`

pre 0.6 `\pgfcalendarjulianyearthoweeek`

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 `\pgfextcalendarcurrentweek` will be available.

`\pgfextcalendarcurrentweek`

pre 0.6 `\pgfcalendarcurrentweek`

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

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

`\pgfextcalendarifdateweek`

pre 0.6 `\pgfcalendarifdateweek`

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

## 30 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, 63] & [41, 47, 43]

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

`/pgf/foreach/ext/use int=<start>to<end>step<delta>` (no default)

pre 0.6 /pgf/foreach/use int

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

`/pgf/foreach/ext/use float=<start>to<end>step<delta>` (no default)

pre 0.6 /pgf/foreach/use float

Same as above, however the results are not truncated.

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

pre 0.6 /pgf/foreach/no separator

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 LuaT<sub>E</sub>X 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,
      ext/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, ext/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/ext/normal list` (no value)  
pre 0.6 /pgf/foreach/normal list

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/ext/xparser={\langle argument specification \rangle}{\langle foreach value \rangle}` (no default)  
pre 0.6 /pgf/foreach/xparser

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

`/pgf/foreach/ext/xparser 0m=default` (default `{}`)  
pre 0.6 /pgf/foreach/xparser 0m

Sets up a list whose elements may contain an optional argument inside `[]` which correspond to two `\foreach` variables, say `\Options/\Text`. The `default` value is the default value if the optional argument is missing.

**Key handler** `\key/.ext_list xparse={\langle argument specification \rangle}{\langle comma-separated list of values \rangle}`

pre 0.6 .list xparse

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

## 31 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: [26] & [29]

### 31.1 PGFmath

#### 31.1.1 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/ext/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  $1^\circ$ .

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 `extfullarc` function)

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

#### 31.1.2 Functions

```
extstrrepeat("Text", x)
```

```
pre 0.6 strrepeat
\pgfmathextstrrepeat{"Text"}{x}
```

Returns a string with `Text` repeated  $x$  times.

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

```
extisInString("String", "Text")
pre 0.6 isInString
\pgfmathextisInString{"String"}{"Text"}
```

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

```
0 and 1 \pgfmathparse{extisInString("foo", "bar")}
\pgfmathresult \ and\
\pgfmathparse{extisInString("foo", "foobar")}
\pgfmathresult
```

```
extstrcat("Text A", "Text B", ...)
pre 0.6 strcat
\pgfmathextstrcat{"Text A"}{"Text B"}{...}
```

Returns the concatenation of all given parameters.

```
blue!21!green \pgfmathparse{extstrcat("blue!", int(7*3), "!green")}
\pgfmathresult
```

```
extisEmpty("Text")
pre 0.6 isEmpty
\pgfmathextisEmpty{"Text"}
```

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

```
0 and 1 and 1 \pgfmathparse{extisEmpty("foo")} \pgfmathresult\ and\
\pgfmathparse{extisEmpty("")} \pgfmathresult\ and\
\def\emptyText{}
\pgfmathparse{extisEmpty("\emptyText")} \pgfmathresult
```

```
extatanXY(x,y)
pre 0.6 atanXY
\pgfmathextatanXY{x}{y}
```

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

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

```
extatanYX(y,x)
pre 0.6 atanYX
\pgfmathextatanYX{y}{x}
```

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

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

### 31.1.3 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 `"`.

```
extanglebetween("p1", "p2")
pre 0.6 anglebetween
\pgfmathextanglebetween{"p1"}{"p2"}
```

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

```
extqanglebetween("p")
pre 0.6 qanglebetween
\pgfmathextqanglebetween{"p"}
```

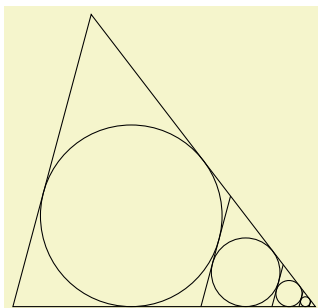
Return the angle between the origin and the center of the node  $p$ .

```
extdistancebetween("p1", "p2")
pre 0.6 distancebetween
\pgfmathextdistancebetween{"p1"}{"p2"}
```

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

```
extqdistancebetween("p")
pre 0.6 qdistancebetween
\pgfmathextqdistancebetween{"p"}
```

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



```
\usetikzlibrary {calc,ext.misc,through}
\begin{tikzpicture}
\path (0,0) coordinate (A) + (0:4) coordinate (B) +(75:4) coordinate (C);
\draw (A) -- (B) -- (C) -- cycle;
\foreach \cnt in {1,...,4}{
\pgfmathsetmacro\triA{extdistancebetween("B","C")}
\pgfmathsetmacro\triB{extdistancebetween("C","A")}
\pgfmathsetmacro\triC{extdistancebetween("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}
```

## 31.2 PGFfor

This library loads also most of the functions of the pgffor-ext of section 30 on page 84.

## 31.3 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.

#### 31.3.1 Conditionals

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

pre 0.6 /utils/if

This key checks the conditional  $\langle cond \rangle$  and applies the styles  $\langle true \rangle$  if  $\langle cond \rangle$  is true, otherwise  $\langle false \rangle$ .  $\langle cond \rangle$  can be anything that pgfmath understands.

As a side effect on how pgfkeys parses argument, the  $\langle false \rangle$  argument is actually optional.

The following keys use T<sub>E</sub>X' macros  $\backslash if$ ,  $\backslash ifx$ ,  $\backslash ifnum$  and  $\backslash ifdim$  for faster executions.

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

pre 0.6 /utils/TeX/if

This key checks via  $\backslash if$  if  $\langle token A \rangle$  matches  $\langle token B \rangle$  and applies the styles  $\langle true \rangle$  if it does, otherwise  $\langle false \rangle$ .

As a side effect on how pgfkeys parses argument, the  $\langle false \rangle$  argument is actually optional.

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

pre 0.6 /utils/TeX/ifx

As above but via  $\backslash ifx$ .

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

pre 0.6 /utils/TeX/ifnum

This key checks  $\backslash ifnum \langle num cond \rangle$  and applies the styles  $\langle true \rangle$  if true, otherwise  $\langle false \rangle$ . A delimiting  $\backslash relax$  will be inserted after  $\langle num cond \rangle$ .

As a side effect on how pgfkeys parses arguments, the  $\langle false \rangle$  argument is actually optional.

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

pre 0.6 /utils/TeX/ifdim

As above but with  $\backslash ifdim$ .

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

pre 0.6 /utils/TeX/ifempty

This checks whether  $\langle Text \rangle$  is empty and applies styles  $\langle true \rangle$  if true, otherwise  $\langle false \rangle$ .

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

pre 0.6 /utils/TeX/ifxempty

This checks whether fully expanded  $\langle Text \rangle$  is empty and applies styles  $\langle true \rangle$  if true, otherwise  $\langle false \rangle$ .

#### 31.3.2 Handlers

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

**Key handler**  $\langle key \rangle/.ext\_pgfmath=\langle eval \rangle$

pre 0.6 .pgfmath

This handler evaluates  $\langle eval \rangle$  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**  $\langle key \rangle/.ext\_pgfmath int=\langle eval \rangle$



```
pre 0.6 .pgfmath int
```

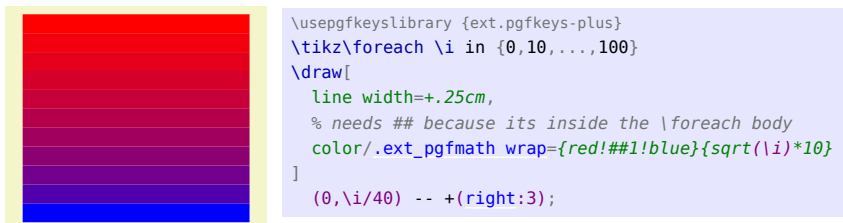
As above but truncates the result.

**Key handler**  $\langle key \rangle /.ext\_pgfmath\ wrap=\{\langle wrapper \rangle\}\{\langle eval \rangle\}$

```
pre 0.6 .pgfmath wrap
```

This feeds the result of  $\langle eval \rangle$  as #1 to  $\langle wrapper \rangle$ .

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



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

```
pre 0.6 .pgfmath if
```

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

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

```
pre 0.6 .if
```

Checks via `\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**  $\langle key \rangle /.ext\_ifx=\langle token A \rangle \langle token B \rangle \{\langle true \rangle\}\{\langle false \rangle\}$

```
pre 0.6 .ifx
```

As above but via `\ifx`.

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

```
pre 0.6 .ifnum
```

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

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

```
pre 0.6 .ifdim
```

As above but via `\ifdim`.

**Key handler**  $\langle key \rangle /.ext\_ifxempty=\{\langle Text \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

```
pre 0.6 .ifxempty
```

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

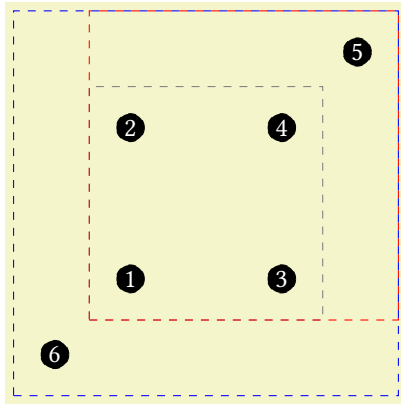
**Key handler**  $\langle key \rangle /.ext\_ifempty=\{\langle Text \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

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

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

```
pre 0.6 .List
```

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



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

## 31.4 TikZ

`/tikz/ext/reverse clip=<direction>` (default counter clockwise)

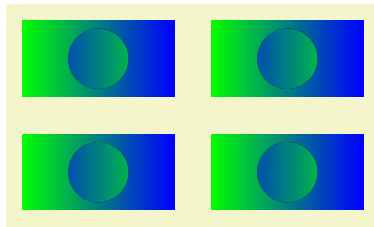
pre 0.6 /tikz/reverse clip

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

`/tikz/ext/clip rule=<direction>` (default even odd)

pre 0.6 /tikz/clip rule

This key switches directly<sup>9</sup> 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 [ext/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={ext/clip rule=even odd},
  column 1/.append style={ext/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}
```

<sup>9</sup>Meaning, it directly executes `\pgfseteorule` / `\pgfsetnonzerorule` and doesn't accumulate where TikZ throws an error.

## Part V

# Changelog, Index & References

## Changelog

Version 0.6.3 (2026-06-29)

- Bugfix to `ext.positioning-plus`. [\[16\]](#)
- Bugfix to `ext.beamer`.
- Bugfix to `ext.paths.ortho`.
- Added node pictures to `ext.nodes`.
- Added auto key to `ext.topaths.arctthrough`.
- `warn` is not the default compatibility setting anymore.

Version 0.6.2 (2025-04-24)

- Bugfix to and slight refactoring of `ext.paths.ortho`.
- Bugfix to `pgffor-ext/ext.misc`.
- Added better support for graphs library for `ext.beamer`.

Version 0.6.1 (2025-04-12)

- Added TikZ library `ext.beamer`.
- Added new tips `ext_Double Cap`, `ext_Double Stealth` and `ext_Double Triange`.
- Bugfix to `ext.arrows-plus`. [\[57\]](#)

Version 0.6 (2025-03-18)

- Added `\tikzextset`, `\tikzextversion` and `\tikzextversionnumber`
- Added six new auto placement mechanisms: `ext/above`, `ext/below`, `ext/west`, `ext/east`, `ext/north` and `ext/south`.

- Added `ext/auto offset` for auto placement.
- Added `ext/precise auto angle`.
- Added TikZ library `ext.arrows-plus`.
- Added TikZ library `ext.topaths.autobend`.
- Made `ext.node-families` and `ext.scalepicture` memoizable.

Version 0.5.1 (2023-04-02)

- Added PGF library `ext.arrows`.
- Bugfix to `ext.pgfkeys-plus`. [\[23\]](#)

Version 0.5 (2023-03-17)

- 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 (2022-10-30)

- Added TikZ library `ext.scalepicture`.
- Bugfixes to `shapes.uncenteredrectangle`, `paths.ortho`, `positioning-plus` and `pgfcalendar-ext`.

Version 0.4.1 (2022-10-23)

- 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 [22].

Version 0.4 (2022-10-10)

- CTAN version of 0.3.1

Version 0.3.1 (2022-10-09)

- 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 (2022-09-24)

- 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 (2022-08-21)

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

Version 0.1 (2022-08-16)

- 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, [29](#)
- | - | path operation, [36](#)
- | - path operation, [36](#)
- | path operation, [36](#)
- .< handler, [17](#)
- | - path operation, [36](#)
  
- above key, [44](#)
- above left key, [44](#)
- above right key, [44](#)
- absolute key, [27](#)
- add anchor key, [27](#)
- add anchor osep key, [27](#), [28](#)
- add anchor sep key, [28](#)
- alt key, [15](#), [16](#)
- anchor key, [26](#)
- aobs invisible key, [15](#)
- aobs visible key, [15](#)
- aobs-tikz package, [13](#)
- aot key, [16](#)
- arc path operation, [10](#), [63](#)
- arc arrows key, [10](#), [11](#)
- Arc Barb arrow tip, [57](#), [58](#)
- arc through key, [48](#)
- arc to path operation, [33](#)
- \arrow, [11](#)
- arrow shift factor key, [10](#)
- arrow shift factor end key, [10](#)
- arrow shift factor start key, [10](#)
- arrow shift mode key, [9](#)
- Arrow tips
  - Arc Barb, [57](#), [58](#)
  - Bar, [57](#), [58](#)
  - Bracket, [57](#), [58](#)
  - Circle, [57](#), [58](#)
  - Diamond, [57](#)
  - Ellipse, [57](#), [58](#)
  - ext\_Centered Arc Barb, [57](#), [58](#)
  - ext\_Centered Bar, [57](#)
  - ext\_Centered Bracket, [57](#)
  - ext\_Centered Circle, [57](#)
  - ext\_Centered Diamond, [57](#)
  - ext\_Centered Ellipse, [57](#)
  - ext\_Centered Hooks, [57](#)
  - ext\_Centered Kite, [57](#)
  - ext\_Centered Parenthesis, [57](#)
  - ext\_Centered Rays, [58](#)
  - ext\_Centered Rectangle, [57](#)
  - ext\_Centered Square, [57](#)
  - ext\_Centered Stealth, [57](#)
  - ext\_Centered Straight Barb, [57](#)
  - ext\_Centered Tee Barb, [57](#)
  - ext\_Centered Triangle, [57](#)
  - ext\_Centered Turned Square, [58](#)
  - ext\_Double Cap, [59](#)
  - ext\_Double Stealth, [59](#)
  - ext\_Double Triangle, [59](#)
  - ext\_Hug Cap, [59](#)
  - ext\_Loop, [59](#)
  - ext\_Untipped Bar, [58](#)
  - ext\_Untipped Bracket, [58](#)
  - ext\_Untipped Circle, [58](#)
  - ext\_Untipped Ellipse, [58](#)
  - ext\_Untipped Parenthesis, [58](#)
  - ext\_Untipped Tee Barb, [58](#)
  - Hooks, [57](#)
  - Kite, [57](#)
  - Parenthesis, [57](#), [58](#)
  - Rays, [58](#)
  - Rectangle, [57](#)
  - Square, [57](#)

- Stealth, [57](#), [59](#)
- Straight Barb, [57](#)
- Tee Barb, [57](#), [58](#)
- Triangle, [57](#), [59](#)
- Turned Square, [58](#)
- `\arrowreversed`, [11](#)
- arrows key, [7](#)
- `arrows.meta` library, [56](#)
- `atan2` math function, [61](#), [87](#)
- auto key, [31](#)
- auto key, [48](#)
- auto offset key, [31](#)
- auto offset for brace decoration key, [31](#)
- auto with offset key, [31](#)
- autobend down key, [49](#)
- autobend east key, [49](#)
- autobend left key, [49](#)
- autobend north key, [49](#)
- autobend right key, [49](#)
- autobend south key, [49](#)
- autobend up key, [49](#)
- autobend west key, [49](#)
- 
- background code key, [19](#)
- Bar arrow tip, [57](#), [58](#)
- base left key, [44](#)
- base right key, [44](#)
- baseline key, [25](#)
- baseline key, [26](#)
- beamer function key, [14](#)
- beamer shortcuts key, [15](#)
- below key, [44](#)
- below left key, [44](#)
- below right key, [44](#)
- bend left key, [49](#)
- better beamer shortcut key, [17](#)
- brace decoration, [31](#)
- Bracket arrow tip, [57](#), [58](#)
- 
- `calc` library, [87](#)
- calendar library, [18](#)

- `cd` library, [78](#), [79](#)
- center suffix key, [48](#)
- Circle arrow tip, [57](#), [58](#)
- circle path operation, [10](#), [91](#)
- circle shape, [20](#)
- circle arrow arrows key, [63](#)
- circle arrow delta angle key, [63](#)
- circle arrow end angle key, [63](#)
- circle arrow start angle key, [63](#)
- circle arrow turn left east key, [63](#)
- circle arrow turn left north key, [63](#)
- circle arrow turn left south key, [64](#)
- circle arrow turn left west key, [64](#)
- circle arrow turn right east key, [64](#)
- circle arrow turn right north key, [64](#)
- circle arrow turn right south key, [64](#)
- circle arrow turn right west key, [64](#)
- circle cross split part fill key, [67](#)
- circle cross split uses custom fill key, [67](#)
- clip rule key, [91](#)
- clockwise key, [33](#), [48](#)
- code key, [19](#)
- compat key, [7](#)
- corner above left key, [45](#)
- corner above right key, [45](#)
- corner below left key, [45](#)
- corner below right key, [45](#)
- corner east above key, [45](#)
- corner east below key, [45](#)
- corner north left key, [45](#)
- corner north right key, [45](#)
- corner south left key, [45](#)
- corner south right key, [45](#)
- corner west above key, [45](#)
- corner west below key, [45](#)
- `cos` math function, [61](#)
- `cos` path operation, [10](#), [40](#)
- counter clockwise key, [33](#), [48](#)
- cover key, [14](#)
- current bounding box shape, [27](#)
- current bounding box node, [27](#)

## Date tests

- ext/and, 82
- ext/Apr, 82
- ext/Aug, 82
- ext/calendar week of month, 82
- ext/calendar week of month', 82
- ext/Dec, 82
- ext/Feb, 82
- ext/Jan, 82
- ext/Jul, 82
- ext/Jun, 82
- ext/leap year, 82
- ext/Mar, 82
- ext/May, 82
- ext/not, 82
- ext/Nov, 82
- ext/Oct, 82
- ext/Sep, 82
- ext/week, 82
- ext/week of month, 82
- ext/week of month', 82
- ext/yesterday, 82

day code key, 18

day text key, 18

day xshift key, 18

day yshift key, 18

## Decorations

- brace, 31
- markings, 9

decorations.markings library, 11

decorations.pathreplacing library, 31

## Diamond arrow tip, 57

\dimexpr, 46

distance key, 36, 37

down horizontal up key, 38

du distance key, 37

east above key, 44

east below key, 44

edge path operation, 48

edge on layer key, 19

Ellipse arrow tip, 57, 58

ellipse path operation, 10

enable first char key, 16

enable first char < key, 16

enable handler key, 17

## Environments

- pgfonlayer, 19

- tikzcd, 80

evaluate key, 89

.evaluated handler, 88

even odd rule key, 91

every arc arrows key, 11

every arc to key, 34

every brace node key, 31

every day key, 18

every diagram key, 79

every month key, 18

every node key, 25

every node picture key, 25

every node picture scope key, 26

every outer matrix key, 25

every outer node picture key, 25

every softpath arrows key, 11

every week key, 18

execute at end node key, 80

execute at end picture key, 20

ext.arrows library, 9, 56

ext.arrows-plus library, 9

ext.beamer library, 13

ext.calendar-plus library, 18

ext.layers library, 19

ext.misc library, 84–86

ext.node-families library, 20

ext.node-families.shapes.geometric library, 23

ext.nodes library, 24

ext.paths.arcto library, 33

ext.paths.ortho library, 35

ext.paths.timer library, 10, 39

ext.patterns.images library, 42

ext.pgfkeys-plus pgfkeys library, 88

ext.positioning-plus library, 43



- ext.scalepicture library, [46](#)
- ext.shapes.circlearrow library, [63](#)
- ext.shapes.circlecrosssplit library, [67](#)
- ext.shapes.heatmark library, [70](#)
- ext.shapes.rectangleroundedcorners library, [73](#)
- ext.shapes.superellipse library, [75](#)
- ext.shapes.uncenteredrectangle library, [78](#), [79](#)
- ext.topaths.arctthrough library, [48](#)
- ext.topaths.autobend library, [49](#)
- ext.transformations.mirror library, [51](#), [61](#)
- ext/and date test, [82](#)
- ext/Apr date test, [82](#)
- ext/arrow pic type, [10](#)
- ext/Aug date test, [82](#)
- ext/calendar week of month date test, [82](#)
- ext/calendar week of month' date test, [82](#)
- ext/Dec date test, [82](#)
- ext/Feb date test, [82](#)
- ext/Jan date test, [82](#)
- ext/Jul date test, [82](#)
- ext/Jun date test, [82](#)
- ext/leap year date test, [82](#)
- ext/Mar date test, [82](#)
- ext/May date test, [82](#)
- ext/not date test, [82](#)
- ext/Nov date test, [82](#)
- ext/Oct date test, [82](#)
- ext/Sep date test, [82](#)
- ext/softpath arrow pic type, [10](#)
- ext/softpath arrows pic type, [10](#)
- ext/week date test, [82](#)
- ext/week of month date test, [82](#)
- ext/week of month' date test, [82](#)
- ext/yesterday date test, [82](#)
- .ext\_< handler, [17](#)
- ext\_Centered Arc Barb arrow tip, [57](#), [58](#)
- ext\_Centered Bar arrow tip, [57](#)
- ext\_Centered Bracket arrow tip, [57](#)
- ext\_Centered Circle arrow tip, [57](#)
- ext\_Centered Diamond arrow tip, [57](#)
- ext\_Centered Ellipse arrow tip, [57](#)

- ext\_Centered Hooks arrow tip, [57](#)
- ext\_Centered Kite arrow tip, [57](#)
- ext\_Centered Parenthesis arrow tip, [57](#)
- ext\_Centered Rays arrow tip, [58](#)
- ext\_Centered Rectangle arrow tip, [57](#)
- ext\_Centered Square arrow tip, [57](#)
- ext\_Centered Stealth arrow tip, [57](#)
- ext\_Centered Straight Barb arrow tip, [57](#)
- ext\_Centered Tee Barb arrow tip, [57](#)
- ext\_Centered Triangle arrow tip, [57](#)
- ext\_Centered Turned Square arrow tip, [58](#)
- ext\_circle arrow shape, [63](#)
- ext\_circle cross split shape, [67](#)
- ext\_current node picture bounding box node, [27](#)
- ext\_Double Cap arrow tip, [59](#)
- ext\_Double Stealth arrow tip, [59](#)
- ext\_Double Triangle arrow tip, [59](#)
- ext\_heatmark shape, [70](#)
- ext\_Hug Cap arrow tip, [59](#)
- .ext\_if handler, [89](#)
- .ext\_ifdim handler, [89](#)
- .ext\_ifempty handler, [89](#)
- .ext\_ifnum handler, [89](#)
- .ext\_ifx handler, [89](#)
- .ext\_ifxempty handler, [89](#)
- ext\_lastdayinmonthofyear math function, [18](#)
- .ext\_List handler, [89](#)
- .ext\_list xparse handler, [85](#)
- ext\_Loop arrow tip, [59](#)
- .ext\_pgfmth handler, [88](#)
- .ext\_pgfmth if handler, [89](#)
- .ext\_pgfmth int handler, [88](#)
- .ext\_pgfmth wrap handler, [89](#)
- ext\_rectangle with rounded corners shape, [73](#)
- ext\_superellipse shape, [75](#)
- ext\_superellipsex math function, [76](#)
- ext\_superellipsey math function, [76](#)
- ext\_uncentered rectangle shape, [78](#)
- ext\_Untipped Bar arrow tip, [58](#)
- ext\_Untipped Bracket arrow tip, [58](#)
- ext\_Untipped Circle arrow tip, [58](#)

- ext\_Untipped Ellipse arrow tip, 58
- ext\_Untipped Parenthesis arrow tip, 58
- ext\_Untipped Tee Barb arrow tip, 58
- ext\_weeksinmonthofyear math function, 18
- extanglebetween math function, 87
- extatanXY math function, 87
- extatanYX math function, 87
- extdistancebetween math function, 87
- external library, 13, 20, 46
- extisEmpty math function, 86
- extIsInString math function, 86
- extqanglebetween math function, 87
- extqdistancebetween math function, 87
- extstrcat math function, 86
- extstrrepeat math function, 86

- first char key, 16
- fit library, 24, 43
- fit bounding box key, 44
- foreground code key, 19
- from center key, 36
- full arc key, 86

- graphs library, 16
- grid path operation, 10

- heatmark arc *(arc number)* key, 71
- heatmark arc rings key, 70
- heatmark arc sep key, 70
- heatmark arc sep angle key, 70
- heatmark arc width key, 70
- heatmark arcs key, 70
- heatmark inner opacity key, 70
- heatmark outer opacity key, 70
- heatmark ring *(ring number)* key, 71
- heatmark ring *(ring number)* arc *(arc number)* key, 71
- height key, 21, 23
- Hooks arrow tip, 57
- horizontal vertical key, 38
- horizontal vertical horizontal key, 38
- hvh distance key, 36

- hvh ratio key, 36
- hvvh distance key, 36
- hvvh ratio key, 36

- if key, 18, 88
- \ifdate, 83
- ifdim key, 88
- ifempty key, 88
- ifnum key, 88
- ifx key, 88
- ifxempty key, 88
- ignore line width key, 13, 26
- image as pattern key, 42
- install auto key, 31
- install auto offset for brace decoration key, 31
- install shortcuts key, 38
- install uncentered rectangle in column key, 80
- intersections library, 30
- invisible key, 14

#### Key handlers

- .<, 17
- .evaluated, 88
- .ext\_<, 17
- .ext\_List, 89
- .ext\_if, 89
- .ext\_ifdim, 89
- .ext\_ifempty, 89
- .ext\_ifnum, 89
- .ext\_ifx, 89
- .ext\_ifxempty, 89
- .ext\_list xparse, 85
- .ext\_pgfmth, 88
- .ext\_pgfmth if, 89
- .ext\_pgfmth int, 88
- .ext\_pgfmth wrap, 89

- Kite arrow tip, 57

- large key, 34
- layers key, 7
- left key, 44

left vertical right key, 38

length key, 59

#### Libraries

arrows.meta, 56

calc, 87

calendar, 18

cd, 78, 79

decorations.markings, 11

decorations.pathreplacing, 31

ext.arrows, 9, 56

ext.arrows-plus, 9

ext.beamer, 13

ext.calendar-plus, 18

ext.layers, 19

ext.misc, 84–86

ext.node-families, 20

ext.node-families.shapes.geometric, 23

ext.nodes, 24

ext.paths.arcto, 33

ext.paths.ortho, 35

ext.paths.timer, 10, 39

ext.patterns.images, 42

ext.positioning-plus, 43

ext.scalepicture, 46

ext.shapes.circulararrow, 63

ext.shapes.circlecrosssplit, 67

ext.shapes.heatmark, 70

ext.shapes.rectangleroundedcorners, 73

ext.shapes.superellipse, 75

ext.shapes.uncenteredrectangle, 78, 79

ext.topaths.arcthrough, 48

ext.topaths.autobend, 49

ext.transformations.mirror, 51, 61

external, 13, 20, 46

fit, 24, 43

graphs, 16

intersections, 30

markings, 29

positioning, 43

shapes.geometric, 23

spath3, 30

lr distance key, 37

mark connection node key, 29

markings decoration, 9

markings library, 29

#### Math functions

atan2, 61, 87

cos, 61

ext\_lastdayinmonthofyear, 18

ext\_superellipse, 76

ext\_superellipse, 76

ext\_weeksinmonthofyear, 18

extanglebetween, 87

extatanXY, 87

extatanYX, 87

extdistancebetween, 87

extisEmpty, 86

extisinstring, 86

extqanglebetween, 87

extqdistancebetween, 87

extstrcat, 86

extstrrepeat, 86

sin, 61

math mode key, 80

#### Math operators

R, 86

matrix of math nodes key, 80

matrix on layer key, 19

max bounding box key, 13

maximum picture height key, 46

maximum picture height\* key, 47

maximum picture size key, 47

maximum picture width key, 46

maximum picture width\* key, 47

memoize package, 13, 20, 46

mid left key, 44

mid right key, 44

middle 0 to 1 key, 35

minimum height key, 22, 44

minimum picture height key, 46

minimum picture height\* key, 47

- minimum picture size key, 46
- minimum picture width key, 46
- minimum picture width\* key, 47
- minimum width key, 21, 44
- Mirror key, 54
- mirror key, 52
- Mirror x key, 53
- mirror x key, 52
- Mirror y key, 54
- mirror y key, 52
- month code key, 18
- month text key, 18
- month xshift key, 18
- month yshift key, 18
- name key, 42
- name prefix key, 26
- name prefix .. key, 26
- no separator key, 84
- node on layer key, 19
- node on line key, 29
- node picture key, 26
- node-families key, 7
- \nodepart, 25
- nodes key, 7
- nodes on curve key, 30
- nodes on curve' key, 30
- nodes on line key, 29
- nonzero rule key, 91
- normal list key, 85
- north left key, 44
- north right key, 44
- on grid key, 44
- only key, 15, 16
- only horizontal first key, 38
- only horizontal second key, 38
- only vertical first key, 38
- only vertical second key, 38
- \onslide, 13
- option key, 42

- options key, 42
- outer sep key, 30
- Packages and files
  - aobs-tikz, 13
  - memoize, 13, 20, 46
  - pgfcalendar-ext, 82
  - pgffor, 84
  - pgffor-ext, 84, 88
  - xparse, 85
- parabola path operation, 10, 40
- Parenthesis arrow tip, 57, 58
- patch key, 19
- Path operations
  - , 29
  - |-|, 36
  - |-, 36
  - |, 36
  - |-, 36
  - arc, 10, 63
  - arc to, 33
  - circle, 10, 91
  - cos, 10, 40
  - edge, 48
  - ellipse, 10
  - grid, 10
  - parabola, 10, 40
  - plot, 10
  - r-du, 37
  - r-lr, 37
  - r-rl, 37
  - r-ud, 37
  - rectangle, 10, 39
  - sin, 10, 40
  - to, 48
- paths.arc to key, 7
- paths.ortho key, 7
- paths.timer key, 7
- patterns.images key, 7
- /pgf/
  - arrow keys/

- length, [59](#)
- decoration/
  - mark connection node, [29](#)
  - raise, [31](#)
- foreach/
  - evaluate, [89](#)
  - var, [85](#)
- minimum height, [22](#), [44](#)
- minimum width, [21](#), [44](#)
- outer sep, [30](#)
- shape border rotate, [71](#)
- text, [42](#)

/pgf/ext/

- circle arrow arrows, [63](#)
- circle arrow delta angle, [63](#)
- circle arrow end angle, [63](#)
- circle arrow start angle, [63](#)
- circle arrow turn left east, [63](#)
- circle arrow turn left north, [63](#)
- circle arrow turn left south, [64](#)
- circle arrow turn left west, [64](#)
- circle arrow turn right east, [64](#)
- circle arrow turn right north, [64](#)
- circle arrow turn right south, [64](#)
- circle arrow turn right west, [64](#)
- circle cross split part fill, [67](#)
- circle cross split uses custom fill, [67](#)
- full arc, [86](#)
- heatmark arc  $\langle arc\ number \rangle$ , [71](#)
- heatmark arc rings, [70](#)
- heatmark arc sep, [70](#)
- heatmark arc sep angle, [70](#)
- heatmark arc width, [70](#)
- heatmark arcs, [70](#)
- heatmark inner opacity, [70](#)
- heatmark outer opacity, [70](#)
- heatmark ring  $\langle ring\ number \rangle$ , [71](#)
- heatmark ring  $\langle ring\ number \rangle$  arc  $\langle arc\ number \rangle$ , [71](#)
- rectangle with rounded corners north east radius, [73](#)
- rectangle with rounded corners north west radius, [73](#)
- rectangle with rounded corners radius, [73](#)

- rectangle with rounded corners south east radius, [73](#)
- rectangle with rounded corners south west radius, [73](#)
- superellipse exponent, [75](#)
- superellipse step, [75](#)
- superellipse x exponent, [75](#)
- superellipse y exponent, [75](#)
- uncentered rectangle center, [78](#), [80](#)
- uncentered rectangle center yshift, [78](#)
- uncentered rectangle use saved center, [78](#)

/pgf/foreach/ext/

- no separator, [84](#)
- normal list, [85](#)
- use float, [84](#)
- use int, [84](#)
- xparser, [85](#)
- xparser 0m, [85](#)

\pgfcalendar, [83](#)

pgfcalendar-ext key, [7](#)

pgfcalendar-ext package, [82](#)

\pgfextcalendarcurrentweek, [83](#)

\pgfextcalendarifdateweek, [83](#)

\pgfextcalendarjulianyeartoweek, [83](#)

\pgfextmathsuperellipseXY, [76](#)

\pgfextqtransformMirror, [62](#)

\pgfextqtransformmirror, [62](#)

\pgfextset, [6](#)

\pgfextttransformMirror, [62](#)

\pgfextttransformmirror, [62](#)

\pgfextttransformxMirror, [61](#)

\pgfextttransformxmMirror, [61](#)

\pgfextttransformyMirror, [62](#)

\pgfextttransformymirror, [62](#)

pgffor package, [84](#)

pgffor-ext key, [7](#)

pgffor-ext package, [84](#), [88](#)

pgfkeys Libraries

- ext.pgfkeys-plus, [88](#)

pgfkeys-plus key, [7](#)

pgfmath if key, [88](#)

\pgfmathanglebetweenpoints, [61](#)

\pgfmathextanglebetween, [87](#)

- `\pgfmathextatanXY`, 87
- `\pgfmathextatanYX`, 87
- `\pgfmathextdistancebetween`, 87
- `\pgfmathextisempty`, 86
- `\pgfmathextisinstring`, 86
- `\pgfmathextlastdayinmonthofyear`, 18
- `\pgfmathextqanglebetween`, 87
- `\pgfmathextqdistancebetween`, 87
- `\pgfmathextstrcat`, 86
- `\pgfmathextstrrepeat`, 86
- `\pgfmathextsuperellipseX`, 76
- `\pgfmathextsuperellipseY`, 76
- `\pgfmathextweeks in month of year`, 18
- `pgfonlayer` environment, 19
- `\pgfpatharcto`, 34
- `\pgfpointnormalised`, 61
- `\pgfsetarrows`, 63
- `\pgfseteorule`, 91
- `\pgfsetnonzerorule`, 91
- `\pgftext`, 42
- `pic` key, 24
- `pic on layer` key, 19
- Pic Types
  - `ext/arrow`, 10
  - `ext/softpath arrow`, 10
  - `ext/softpath arrows`, 10
- `picture` height key, 46
- `picture` height\* key, 47
- `picture` size\* key, 47
- `picture` width key, 46
- `picture` width\* key, 47
- `plot` path operation, 10
- `pos` key, 35, 39
- `pos` key, 9
- `pos <` key, 9
- `pos < angle` key, 9
- `pos >` key, 9
- `pos > angle` key, 9
- `positioning` library, 43
- `positioning-plus` key, 7
- `precise auto angle` key, 32

- Predefined node
  - current bounding box, 27
  - `ext_current` node picture bounding box, 27
- prefix key, 20
- prefix name key, 26
- R postfix math operator, 86
- `r-du` path operation, 37
- `r-lr` path operation, 37
- `r-rl` path operation, 37
- `r-ud` path operation, 37
- radius key, 34
- raise key, 31
- ratio key, 36
- Rays arrow tip, 58
- Rectangle arrow tip, 57
- rectangle path operation, 10, 39
- rectangle shape, 20
- rectangle east key, 27
- rectangle north key, 27
- rectangle south key, 27
- rectangle timer key, 39
- rectangle west key, 27
- rectangle with rounded corners north east radius key, 73
- rectangle with rounded corners north west radius key, 73
- rectangle with rounded corners radius key, 73
- rectangle with rounded corners south east radius key, 73
- rectangle with rounded corners south west radius key, 73
- reset graphic state key, 26
- reverse clip key, 91
- right key, 44
- right vertical left key, 38
- `rl` distance key, 37
- rotate key, 34
- save picture size key, 46
- `scalecture` key, 7
- `setup` shape key, 21
- shape border rotate key, 71
- Shapes
  - circle, 20

- current bounding box, 27
- ext\_circle arrow, 63
- ext\_circle cross split, 67
- ext\_heatmark, 70
- ext\_rectangle with rounded corners, 73
- ext\_superellipse, 75
- ext\_uncentered rectangle, 78
- rectangle, 20
- shapes key, 7
- shapes.geometric library, 23
- sin math function, 61
- sin path operation, 10, 40
- size key, 22
- sloped key, 31, 32
- small key, 34
- softpath arrows key, 11
- south left key, 44
- south right key, 44
- spacing key, 35
- span key, 44
- span horizontal key, 44
- span vertical key, 44
- spath3 library, 30
- Square arrow tip, 57
- Stealth arrow tip, 57, 59
- Straight Barb arrow tip, 57
- superellipse exponent key, 75
- superellipse step key, 75
- superellipse x exponent key, 75
- superellipse y exponent key, 75
- swap key, 31
- sync bounding box key, 14
- Tee Barb arrow tip, 57, 58
- temporal key, 15, 16
- text key, 42
- text key, 20
- text depth key, 20
- text height key, 20
- text width key, 20
- text width align key, 21

- through key, 48
- /tikz/
  - alt, 16
  - auto, 31
  - baseline, 25
  - bend left, 49
  - commutative diagrams/
    - every diagram, 79
    - install uncentered rectangle in column, 80
    - math mode, 80
  - day code, 18
  - day text, 18
  - day xshift, 18
  - day yshift, 18
  - even odd rule, 91
  - every day, 18
  - every month, 18
  - every node, 25
  - every outer matrix, 25
  - execute at end node, 80
  - execute at end picture, 20
  - if, 18
  - matrix of math nodes, 80
  - month code, 18
  - month text, 18
  - month xshift, 18
  - month yshift, 18
  - name prefix, 26
  - nonzero rule, 91
  - on grid, 44
  - only, 16
  - pics/
    - background code, 19
    - code, 19
    - foreground code, 19
  - pos, 35, 39
  - pos, 9
  - sloped, 31, 32
  - swap, 31
  - temporal, 16
  - to path, 29, 38

- uncrec, [80](#)
- uncrec math mode, [80](#)
- x radius, [34](#)
- y radius, [34](#)
- tikz-cd fix key, [79](#)
- /tikz/ext/
  - above, [44](#)
  - above left, [44](#)
  - above right, [44](#)
  - aobs invisible, [15](#)
  - aobs visible, [15](#)
  - arc arrows, [10](#), [11](#)
  - arc through/
    - auto, [48](#)
    - center suffix, [48](#)
    - clockwise, [48](#)
    - counter clockwise, [48](#)
    - through, [48](#)
  - arc through, [48](#)
  - arc to/
    - clockwise, [33](#)
    - counter clockwise, [33](#)
    - large, [34](#)
    - radius, [34](#)
    - rotate, [34](#)
    - small, [34](#)
    - x radius, [34](#)
    - y radius, [34](#)
  - arrow shift factor, [10](#)
  - arrow shift factor end, [10](#)
  - arrow shift factor start, [10](#)
  - arrow shift mode, [9](#)
  - auto offset, [31](#)
  - auto offset for brace decoration, [31](#)
  - auto with offset, [31](#)
  - autobend down, [49](#)
  - autobend east, [49](#)
  - autobend left, [49](#)
  - autobend north, [49](#)
  - autobend right, [49](#)
  - autobend south, [49](#)

- autobend up, [49](#)
- autobend west, [49](#)
- base left, [44](#)
- base right, [44](#)
- beamer function, [14](#)
- beamer shortcuts/
  - aot, [16](#)
  - enable first char, [16](#)
  - enable first char <, [16](#)
  - enable handler, [17](#)
  - first char, [16](#)
- beamer shortcuts, [15](#)
- below, [44](#)
- below left, [44](#)
- below right, [44](#)
- clip rule, [91](#)
- compat/
  - arrows, [7](#)
  - layers, [7](#)
  - node-families, [7](#)
  - nodes, [7](#)
  - paths.arcto, [7](#)
  - paths.ortho, [7](#)
  - paths.timer, [7](#)
  - patterns.images, [7](#)
  - pgfcalendar-ext, [7](#)
  - pgffor-ext, [7](#)
  - pgfkeys-plus, [7](#)
  - positioning-plus, [7](#)
  - scalepicture, [7](#)
  - shapes, [7](#)
  - topaths.arctthrough, [7](#)
  - transformations.mirror, [7](#)
- compat, [7](#)
- corner above left, [45](#)
- corner above right, [45](#)
- corner below left, [45](#)
- corner below right, [45](#)
- corner east above, [45](#)
- corner east below, [45](#)
- corner north left, [45](#)



- corner north right, [45](#)
- corner south left, [45](#)
- corner south right, [45](#)
- corner west above, [45](#)
- corner west below, [45](#)
- cover, [14](#)
- down horizontal up, [38](#)
- east above, [44](#)
- east below, [44](#)
- edge on layer, [19](#)
- every arc arrows, [11](#)
- every arc to, [34](#)
- every brace node, [31](#)
- every node picture, [25](#)
- every node picture scope, [26](#)
- every outer node picture, [25](#)
- every softpath arrows, [11](#)
- every week, [18](#)
- fit bounding box, [44](#)
- horizontal vertical, [38](#)
- horizontal vertical horizontal, [38](#)
- ignore line width, [13](#)
- image as pattern/
  - name, [42](#)
  - option, [42](#)
  - options, [42](#)
- image as pattern, [42](#)
- invisible, [14](#)
- layers/
  - patch, [19](#)
- left, [44](#)
- left vertical right, [38](#)
- matrix on layer, [19](#)
- max bounding box, [13](#)
- maximum picture height, [46](#)
- maximum picture height\*, [47](#)
- maximum picture size, [47](#)
- maximum picture width, [46](#)
- maximum picture width\*, [47](#)
- mid left, [44](#)
- mid right, [44](#)

- minimum picture height, [46](#)
- minimum picture height\*, [47](#)
- minimum picture size, [46](#)
- minimum picture width, [46](#)
- minimum picture width\*, [47](#)
- Mirror, [54](#)
- mirror, [52](#)
- Mirror x, [53](#)
- mirror x, [52](#)
- Mirror y, [54](#)
- mirror y, [52](#)
- node family/
  - height, [21](#), [23](#)
  - prefix, [20](#)
  - setup shape, [21](#)
  - size, [22](#)
  - text, [20](#)
  - text depth, [20](#)
  - text height, [20](#)
  - text width, [20](#)
  - text width align, [21](#)
  - width, [21](#), [23](#)
- node on layer, [19](#)
- node on line, [29](#)
- node picture, [26](#)
- nodes/
  - install auto, [31](#)
  - install auto offset for brace decoration, [31](#)
- nodes on curve, [30](#)
- nodes on curve', [30](#)
- nodes on line, [29](#)
- north left, [44](#)
- north right, [44](#)
- only horizontal first, [38](#)
- only horizontal second, [38](#)
- only vertical first, [38](#)
- only vertical second, [38](#)
- ortho/
  - distance, [36](#), [37](#)
  - du distance, [37](#)
  - from center, [36](#)

- hvh distance, 36
- hvh ratio, 36
- hvvh distance, 36
- hvvh ratio, 36
- install shortcuts, 38
- lr distance, 37
- middle 0 to 1, 35
- ratio, 36
- rl distance, 37
- spacing, 35
- ud distance, 37
- udlr distance, 37
- vhv distance, 36
- vhv ratio, 36
- pic, 24
- pic on layer, 19
- picture height, 46
- picture height\*, 47
- picture size\*, 47
- picture width, 46
- picture width\*, 47
- pos <, 9
- pos < angle, 9
- pos >, 9
- pos > angle, 9
- precise auto angle, 32
- rectangle timer, 39
- reverse clip, 91
- right, 44
- right vertical left, 38
- save picture size, 46
- softpath arrows, 11
- south left, 44
- south right, 44
- span, 44
- span horizontal, 44
- span vertical, 44
- sync bounding box, 14
- tikz-cd fix, 79
- uncover, 14
- up horizontal down, 38

- vertical horizontal, 38
- vertical horizontal vertical, 38
- visible, 14
- week code, 18
- week label left, 18
- week text, 18
- west above, 44
- west below, 44
- xMirror, 53
- xmirror, 51
- yMirror, 53
- ymirror, 52
- /tikz/ext/node picture/
  - absolute, 27
  - add anchor/
    - rectangle east, 27
    - rectangle north, 27
    - rectangle south, 27
    - rectangle west, 27
  - add anchor, 27
  - add anchor osep, 27, 28
  - add anchor sep, 28
  - anchor, 26
  - baseline, 26
  - ignore line width, 26
  - name prefix .., 26
  - prefix name, 26
  - reset graphic state, 26
- /tikz/graphs/ext/
  - better beamer shortcut, 17
- tikzcd environment, 80
- \tikzextnodepicture, 27
- \tikzextpictureheight, 46
- \tikzextpicturewidth, 46
- \tikzextset, 6
- \tikzextsetupimageaspattern, 42
- \tikzextversion, 6
- \tikzextversionnumber, 6
- to path operation, 48
- to path key, 29, 38
- topaths.arctthrough key, 7

transformations.mirror key, 7

Triangle arrow tip, 57, 59

Turned Square arrow tip, 58

ud distance key, 37

udlr distance key, 37

uncentered rectangle center key, 78, 80

uncentered rectangle center yshift key, 78

uncentered rectangle use saved center key, 78

uncover key, 14

\uncrec, 80

uncrec key, 80

uncrec math mode key, 80

up horizontal down key, 38

use float key, 84

use int key, 84

/utils/ext/

alt, 15

if, 88

ifdim, 88

ifempty, 88

ifnum, 88

ifx, 88

ifxempty, 88

only, 15

pgfmath if, 88

temporal, 15

var key, 85

vertical horizontal key, 38

vertical horizontal vertical key, 38

vhv distance key, 36

vhv ratio key, 36

visible key, 14

week code key, 18

week label left key, 18

week text key, 18

west above key, 44

west below key, 44

width key, 21, 23

x radius key, 34

x radius key, 34

xMirror key, 53

xmirror key, 51

xparse package, 85

xparser key, 85

xparser Om key, 85

y radius key, 34

y radius key, 34

yMirror key, 53

ymirror key, 52

## 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. 10).
- [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. 84).
- [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. 70).
- [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. 56).
- [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. 18).
- [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. 39).
- [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. 39).
- [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. 84).
- [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. 56).
- [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. 24).
- [11] Fence. *Add week day to calendar*. Nov. 2019. URL: <https://tex.stackexchange.com/q/517338> (visited on 09/24/2022) (cit. on pp. 18, 82).
- [12] healyp. *TikZ calendar and conditional tests*. Oct. 2013. URL: <https://tex.stackexchange.com/q/140948> (visited on 09/24/2022) (cit. on pp. 18, 82).
- [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. 56).
- [14] Holene. *Dependent node size in TikZ*. Apr. 2017. URL: <https://tex.stackexchange.com/q/107227> (visited on 09/24/2022) (cit. on p. 20).
- [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. 56).
- [16] jamesvenning. *Example no longer works*. Jan. 2026. URL: <https://github.com/Qrrbrbirlbel/tikz-extensions/issues/24> (visited on 01/27/2026) (cit. on p. 92).
- [17] 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. 82).
- [18] 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. 42).
- [19] 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. 24).
- [20] lucky1928. *add customize line header not smooth connected*. TeX - LaTeX Stack Exchange. ZSCC: NoCitationData[s0]. July 2024. URL: <https://tex.stackexchange.com/q/722350> (visited on 03/28/2025) (cit. on p. 56).
- [21] Marco. *TikZ - Four Colored Circle Split*. Apr. 2017. URL: <https://tex.stackexchange.com/q/121686> (visited on 09/24/2022) (cit. on p. 67).

- [22] 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. 93).
- [23] 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. 92).
- [24] 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. 49).
- [25] 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. 23).
- [26] 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. 86).
- [27] 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. 78).
- [28] 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. 63).
- [29] 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. 86).
- [30] Qrrbrbirlbel. *Answer to "add customize line header not smooth connected"*. TeX - LaTeX Stack Exchange. ZSCC: NoCitationData[s0]. July 2024. URL: <https://tex.stackexchange.com/a/722413> (visited on 03/28/2025) (cit. on p. 56).
- [31] 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. 18, 82).
- [32] 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. 75).
- [33] 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. 56).
- [34] 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. 20).
- [35] 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. 49).
- [36] 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. 24).
- [37] 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. 78).
- [38] 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. 82).
- [39] 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. 70).
- [40] 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. 78).

- [41] 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. 84).
- [42] 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. 39).
- [43] 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. 84).
- [44] 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. 56).
- [45] 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. 24).
- [46] 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. 56).
- [47] 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. 84).
- [48] 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. 63).
- [49] 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. 67).
- [50] 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. 18).
- [51] 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. 56).
- [52] 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. 18, 82).
- [53] 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. 42).
- [54] 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. 39).
- [55] Qrrbrbirlbel. *Answer to “What’s the easiest way to draw the arc defined by three points in TikZ?”* TeX - LaTeX Stack Exchange. Sept. 2013. URL: <https://tex.stackexchange.com/a/132184> (visited on 06/04/2025) (cit. on p. 48).
- [56] Qrrbrbirlbel. *Answer to “Why don’t pic prefixes in TikZ work with matrix cell names?”* TeX - LaTeX Stack Exchange. Dec. 2023. URL: <https://tex.stackexchange.com/a/703436> (visited on 05/21/2025) (cit. on p. 28).
- [57] Qrrbrbirlbel. *arrow shifting for start tip sequence doesn’t work and shift amount needs to be expanded*. Mar. 2025. URL: <https://github.com/Qrrbrbirlbel/tikz-extensions> (visited on 03/29/2025) (cit. on p. 92).

- [58] Manuel Selva. *How to draw nested nodes?* TeX - LaTeX Stack Exchange. Dec. 2012. URL: <https://tex.stackexchange.com/q/85536> (visited on 06/03/2026) (cit. on p. 28).
- [59] 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. 75).
- [60] 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. 79).
- [61] 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. 30).
- [62] 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. 78).
- [63] 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. 84).
- [64] David Z. *What's the easiest way to draw the arc defined by three points in TikZ?* TeX - LaTeX Stack Exchange. Apr. 2011. URL: <https://tex.stackexchange.com/q/15972> (visited on 06/04/2025) (cit. on p. 48).
- [65] Sašo Živanović. *Memoize*. original-date: 2020-05-19T09:58:52Z. Oct. 2023. URL: <https://github.com/sasozivanovic/memoize> (visited on 11/05/2023) (cit. on pp. 13, 20, 46).