

The TikZ-Extensions Package
Manual for version 0.6.3 (11)
<https://github.com/Qrrbrbirlbel/tikz-extensions>

Qrrbrbirlbel

May 21, 2025

Contents

I	Introduction	6
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
II	<i>TikZ</i> Libraries	8
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

7	Beamer with TikZ	13
7.1	Helpers	13
7.2	Beamer	13
7.2.1	Stop Jumping	14
7.2.2	BEAMERFunction and keys	14
7.2.3	BEAMERShortcuts	15
7.2.4	Key Handler	17
8	Calendar	18
8.1	Value-keys and nestable if key	18
8.2	PGFmath functions	18
8.3	Week numbering (ISO 8601)	18
9	Layers	19
10	Node Families	20
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
11	Nodes	24
11.1	Pic as a node	24
11.2	Node pictures	24
11.3	Nodes on paths	26
11.3.1	Nodes on Lines	26
11.3.2	Nodes on Curves	26
11.4	Automatic placement of nodes	27
11.4.1	More than left and right	27
11.4.2	Offset	28
11.4.3	Precise placement	28
12	Arc to a point	30
13	More Horizontal and Vertical Lines	32
13.1	Timers	32
13.2	Zig-Zag	33
13.3	Zig-Zig	34
13.4	Even more Horizontal and Vertical Lines	35

14	Extending the Path Timers	36
14.1	Rectangle	36
14.2	Parabola	37
14.3	Sine/Cosine	37
15	Using Images as a Pattern	39
16	Positioning Plus	40
16.1	Useful corner anchors	40
16.2	Useful placement keys for vertical and horizontal alignment	41
17	Scaling Pictures to a Specific Size	43
17.1	Externalization	43
17.2	Keeping the aspect ratio	43
17.3	Changing the aspect ratio	44
18	Arcs through Three Points	45
19	Autobending	46
20	Mirror, Mirror on the Wall	48
20.1	Using the reflection matrix	48
20.2	Using built-in transformations	50
III	PGF Libraries	52
21	Arrow Tips	53
21.1	Centered	54
21.1.1	Barbed Arrow Tips	54
21.1.2	Geometric Arrow Tips	54
21.1.3	Special Arrow Tips	55
21.2	Untipped	55
21.2.1	Barbed Arrow Tips	55
21.2.2	Geometric Arrow Tips	55
21.3	Original Arrow Tips	56

22	Transformations: Mirroring	58
22.1	Using the reflection matrix	58
22.2	Using built-in transformations	58
23	Shape: Circle Arrow	60
24	Shape: Circle Cross Split	64
25	Shape: Heatmark	67
26	Shape: Rectangle with Rounded Corners	70
27	Shape: Superellipse	72
28	Shape: Uncentered Rectangle	75
IV	Utilities	78
29	Calendar: Weeknumbers and more conditionals	79
29.1	Extensions	79
29.2	Week numbering (ISO 8601)	80
30	Repeating Things and Other Things	81
31	And a little bit more	83
31.1	PGFmath	83
31.1.1	Postfix operator R	83
31.1.2	Functions	83
31.1.3	Functions: using coordinates	84
31.2	PGFfor	85
31.3	PGFkeys	85
31.3.1	Conditionals	85
31.3.2	Handlers	85
31.4	TikZ	88
V	Changelog, Index & References	89
	Changelog	89

Index	91
References	105

Part I

Introduction

1 Usage

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

2 Why do we need it?

Since I have been answering questions on [TeX.sx](https://tex.stackexchange.com/) I've noticed that some questions come up again and again, every time with a slightly different approach on how to solve them.

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

3 Having problems?

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

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

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`.

¹Except 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.

For version 0.6 this is actually the default settings so that active documents keep working – for now.

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 future version, it is planned that the default compatibility setting will not be `warn`.

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)

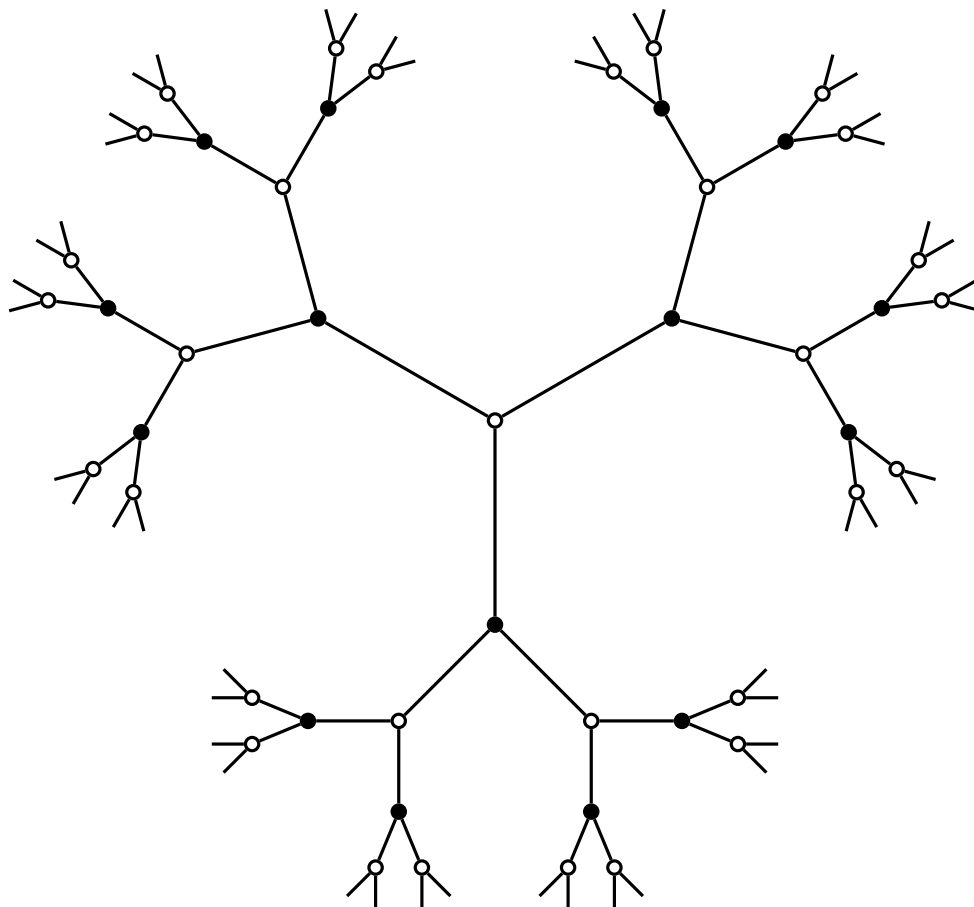
For `<version>` the same choices are valid as for the main `compat` key. It should be noted that at this point, a compatibility setting can't really be reversed since they only forward arguments from an old key or command to the new version.

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.

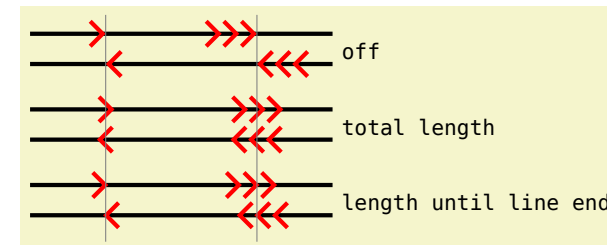
arrow shift mode=off This disables the shifting.

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.

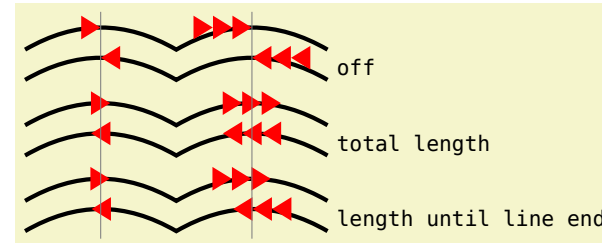
This pic type can place two tip specification, one at `pos >` and one at `pos <` in the reversed direction.

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}
```

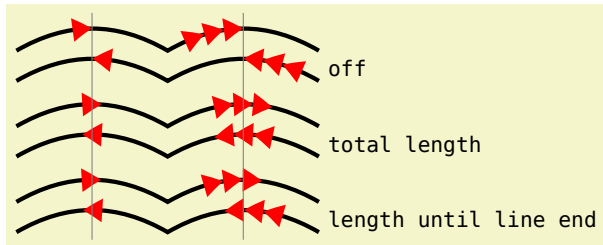
Pic type `ext/softpath arrows=<start tip specification>-<end tip specification>`

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.

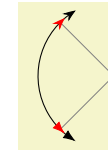
`/tikz/ext/arc arrows=<options>` (default ->)

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` [61] 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 BEAMERFunction 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 BEAMERShortcuts

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 `<->`²) 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.

²Though, 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] & [30, 51, 49]

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 10
28	11	12	13	14	15	16 17
29	18	19	20	21	22	23 24
30	25	26	27	28	29	30 31

```
\usetikzlibrary {ext.calendar-plus}
\tikz
\calendar[
  week list, month label above centered,
  dates=2022-07-01 to 2022-07-31,
  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`³,
- `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
```

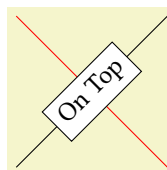
If the `matrix` patch is applied, this key places the matrix on layer `<layer>`.

```
/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}
```

³Only 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] & [33]

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` [61] package.

10.2 Text Box

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

```
/tikz/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)
```

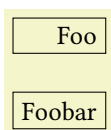
⁴Technically, it will also work with shapes with multiple node parts but it will only affect the main node part.

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

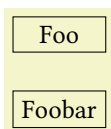
While the keys of the previous subsection work well enough for nodes of the same shape (and the same inner seps), for different node shapes the text box dimensions will be used differently for the node's total dimension.

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

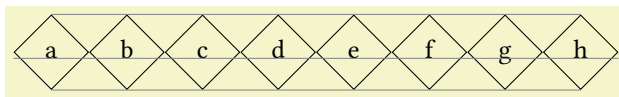
TikZ Library `ext.node-families.shapes.geometric`

```
\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: [24]

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
```

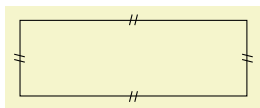
This library extends TikZ’s functionalities around nodes.

Q & A: [10, 18] & [35, 44]

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

11.2 Node pictures

A pic allows any (sub)picture to be handled like a node – with one caveat: it doesn’t provide anchors like a node and is thus not able to be placed like one.

A PGF matrix on the other hand allows multiple (sub)pictures in a node but the matrix cannot be transformed.

The here introduced *node picture* should find a compromise between these two. Here are some rules and guidelines for using this:

- The node picture will be typeset as the text of the node.
- Only the text node part is available. The `\nodepart` command may not be used. A node picture is *not* a matrix. It only exists of one “cell”.
- The baseline of the node is the picture’s y value of 0.
- Just like with a pic or a matrix, certain settings and values will be inherited by the node picture.
- However, other options might not be correctly applied like with other nodes.
- 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.

However, the line width will not contribute to the size. This has the advantage that with any inner seps set to zero, paths of the (sub)picture can lie on the border of the node.

- This is not nestable! You cannot use a node picture inside a node picture.

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

Setting this to true activates the aforementioned procedure for the node's text to be a picture. Setting it to false returns to the default behavior of a normal node.

`/tikz/ext/every node picture` (style, no value)

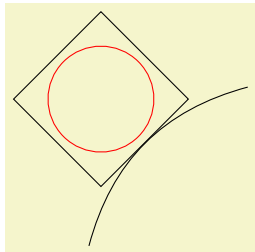
This key will be applied at the start of every node picture.

`/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/name prefix ..` (no value)

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

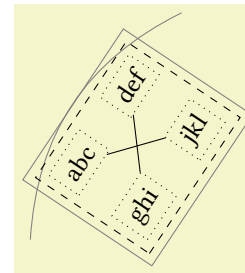


```
\usetikzlibrary {ext.nodes}
\begin{tikzpicture}[x=+7mm, y=+7mm]
\draw (0,0) to[bend left]
  node[ext/node picture, sloped, draw, above]{
    \draw[red] (0,0) circle[radius=1];
  } (3,3);
\end{tikzpicture}
```

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 [54]), 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/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}
```

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.

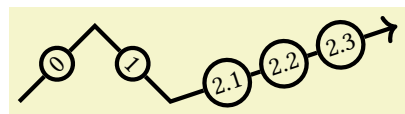
The optional `<anchor specification>` allows to specify the anchors to which the line should connect. It allows one or two anchors divided by `and` to be specified.

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

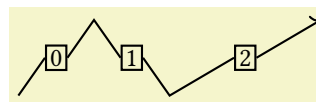
pre 0.6 `/tikz/nodes on line`

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);
```

11.3.2 Nodes on Curves

The following keys need the `intersections` and the `spath3` [58] 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`

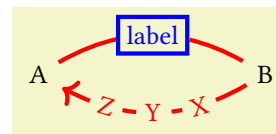
Similar to `nodes on line`, this key allows to have nodes on arbitrary paths.

This is not suitable for paths connecting nodes.

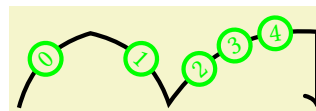
`/tikz/ext/nodes on curve'=<to path>` (style, default `line to`)

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}
```



```

\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);

```

11.4 Automatic placement of nodes

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

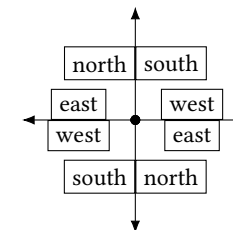
11.4.1 More than left and right

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 active 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.

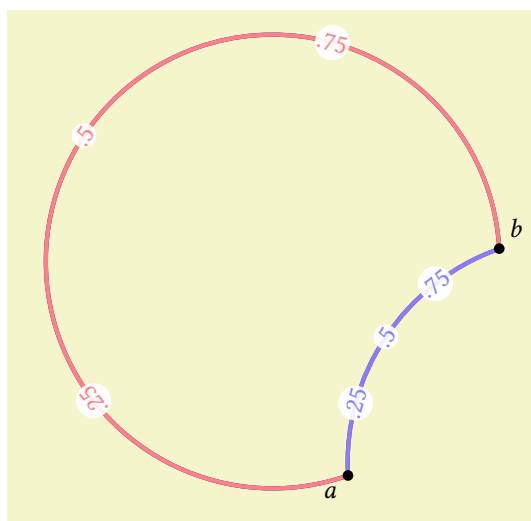
This key disables the `/tikz/sloped` option which in turn will disable this option.

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° .

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

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

`/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 \TeX 's numerical weaknesses. The only case it works quite nicely is when the resulting angle is a multiple of 90° .

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

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

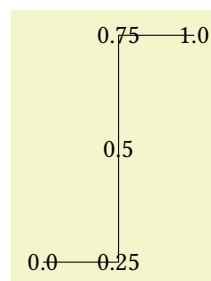
- `pos = -1` will be at the start,
- `pos = 2` will be at the end,
- `pos = 0` will be at the first kink,
- `pos = 1` will be at the second kink and
- `pos = .5` will still be in the middle of the middle part of the connection.

`/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}};
```


13.3 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] & [41, 53]

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

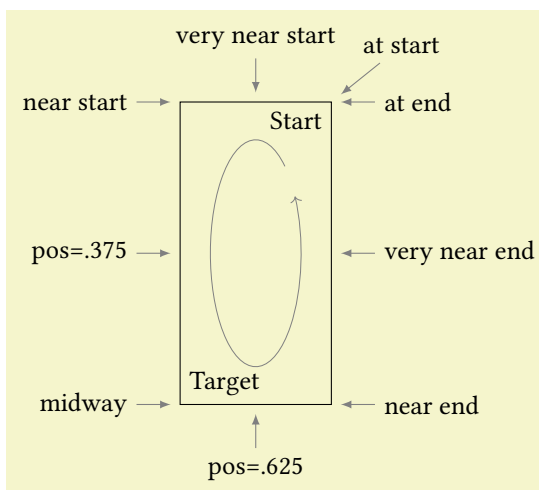
For the `rectangle` path operator, the timer starts with `pos = 0` (= at start) from the starting coordinate in a counter-clockwise direction along the rectangle. The corners will be at positions 0.0, 0.25, 0.5, 0.75 and 1.0.

`/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.

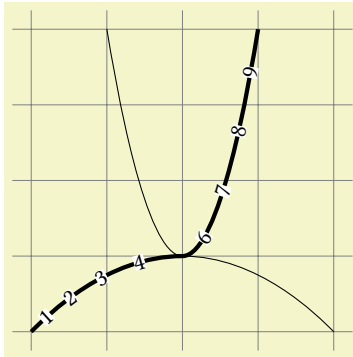


```
\usetikzlibrary {ext.paths.timer}
\begin{tikzpicture}[scale=2, every pin edge/.style={latex-, gray}]
\coordinate [label=above right:Target] (A) at (0,0);
\coordinate [label=below left:Start] (B) at (1,2);
\draw[->, help lines] ([shift=(50:.3 and .75)] .5,1)
  arc[start angle=50, delta angle=340,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

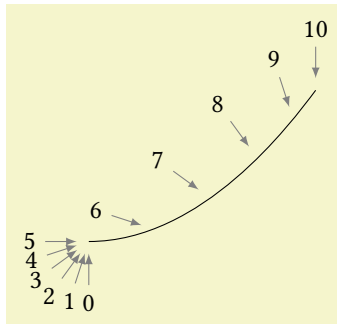
For the parabola path operator the timer is similar to the `.. controls ..` operator.

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}
```

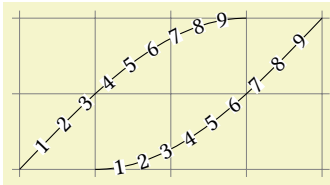
If no bend is specified half the positions will collapse into one end of the curve.



```
\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: [17] & [52]

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”.

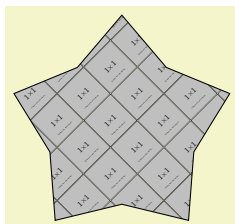
```
\tikzextsetupimageaspattern[⟨options⟩]{⟨name⟩}{⟨image⟩}
```

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

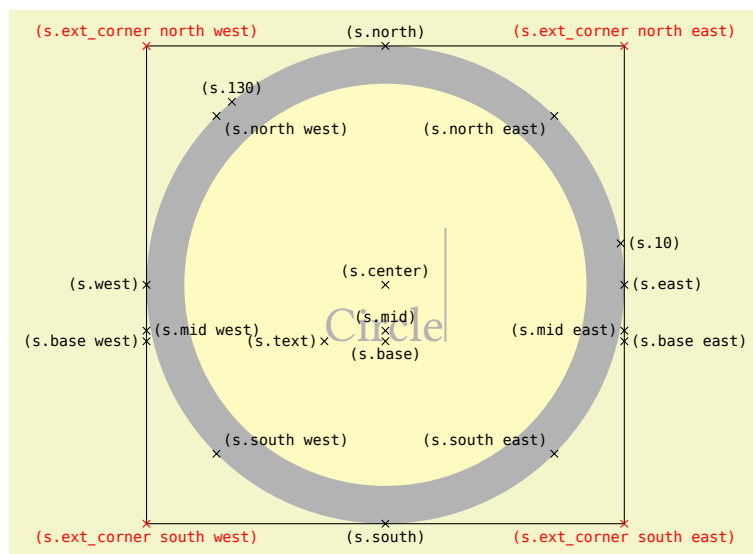
TikZ Library `ext.positioning-plus`

```
\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}
```

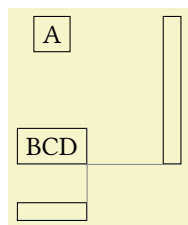

16.2 Useful placement keys for vertical and horizontal alignment

<code>/tikz/ext/left=<specification></code>	(default 0pt)
<code>/tikz/ext/right=<specification></code>	(default 0pt)
<code>/tikz/ext/above=<specification></code>	(default 0pt)
<code>/tikz/ext/below=<specification></code>	(default 0pt)
<code>/tikz/ext/above left=<specification></code>	(default 0pt)
<code>/tikz/ext/below left=<specification></code>	(default 0pt)
<code>/tikz/ext/above right=<specification></code>	(default 0pt)
<code>/tikz/ext/below right=<specification></code>	(default 0pt)
<code>/tikz/ext/mid left=<specification></code>	(default 0pt)
<code>/tikz/ext/mid right=<specification></code>	(default 0pt)
<code>/tikz/ext/base left=<specification></code>	(default 0pt)
<code>/tikz/ext/base right=<specification></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.



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

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.⁵

The fitting functionality is also available without the placement.

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

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

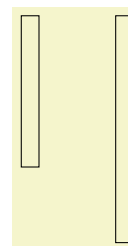
The `span vertical` key will also set `/pgf/minimum height` to the height of this bounding box

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

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



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

⁵This 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=<specification></code>	(default 0pt)
<code>/tikz/ext/corner below left=<specification></code>	(default 0pt)
<code>/tikz/ext/corner above right=<specification></code>	(default 0pt)
<code>/tikz/ext/corner below right=<specification></code>	(default 0pt)
<code>/tikz/ext/corner north left=<specification></code>	(default 0pt)
<code>/tikz/ext/corner south left=<specification></code>	(default 0pt)

<code>/tikz/ext/corner north right=<specification></code>	(default 0pt)
<code>/tikz/ext/corner south right=<specification></code>	(default 0pt)
<code>/tikz/ext/corner west above=<specification></code>	(default 0pt)
<code>/tikz/ext/corner west below=<specification></code>	(default 0pt)
<code>/tikz/ext/corner east above=<specification></code>	(default 0pt)
<code>/tikz/ext/corner east below=<specification></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 picture⁶ will be measured and written to the AUX file so that it will be available at the next compilation run and an appropriate scaling for the picture can be installed.

`\tikzextpicturewidth`

Returns the last measured width of the picture.

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

`\tikzextpictureheight`

Returns the last measured height of the picture.

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

```
/tikz/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` [61] 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.

⁶This 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 $\langle dimension \rangle$. 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 $\langle dimension \rangle$.

```
/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 $\langle dimension \rangle$.

```
/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 $\langle dimension \rangle$. 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 $\langle dimension \rangle$.

```
/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 $\langle dimension \rangle$.

```
/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 $\langle height \rangle$.

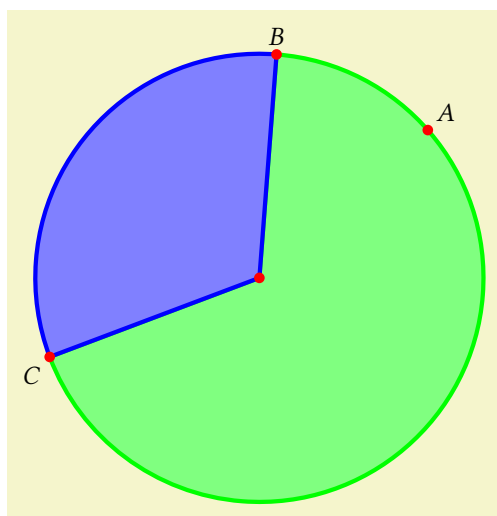
This will scale both axes but independent from each other.

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.



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

\draw[ultra thick, draw=green, fill=green!50]
(B) to[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;

\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/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 through/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 through/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 through/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=<key-value>` (no default)

pre 0.6 /tikz/arc through/arc through

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

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: [23] & [34]

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)

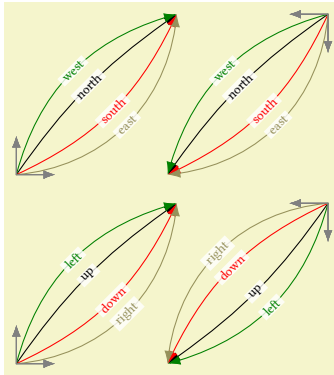
Works like `autobend up` but bends the curve downwards.

`/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

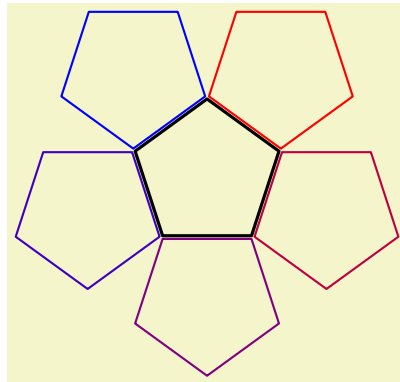
TikZ Library `ext.transformations.mirror`

```
\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



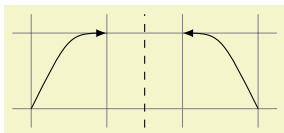
```
\usetikzlibrary {shapes.geometric,ext.transformations.mirror}
\begin{tikzpicture}[line join=round, thick, reg poly/.style={
  shape=regular polygon, regular polygon sides={#1}}]
\node[reg poly=5, minimum size=+2cm, draw, very thick] (a) {};
\foreach \i[evaluate={\col=(\i-1)/.04}] in {1,...,5}
  \node [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 (`<value>`, 0) or `<coordinate>`.



```
\usetikzlibrary {ext.transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-0.25, -.25) grid (3.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);
\draw[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

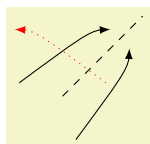
Sets up a transformation that mirrors along a vertical line that goes through point $(0, \langle value \rangle)$ or $\langle coordinate \rangle$.

`/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>`

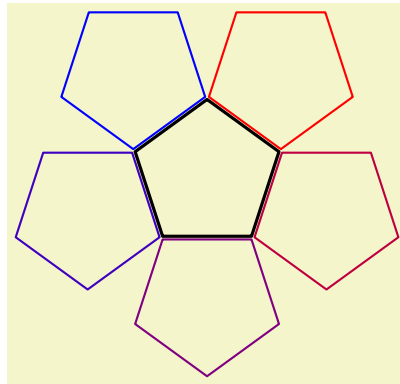
(no default)

pre 0.6 /tikz/mirror

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



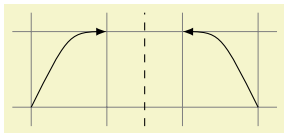
```
\usetikzlibrary {shapes.geometric,ext.transformations.mirror}
\begin{tikzpicture}[line join=round, thick, reg poly/.style={
  shape=regular polygon, regular polygon sides={#1}}]
\node[reg poly=5, minimum size=+2cm, draw, very thick] (a) {};
\foreach \i[evaluate={\col=(\i-1)/.04}] in {1,...,5}
  \node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

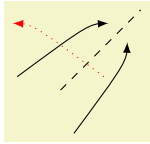
Sets up a transformation that mirrors along a vertical line that goes through point (0, $\langle value \rangle$) or $\langle coordinate \rangle$.

`/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

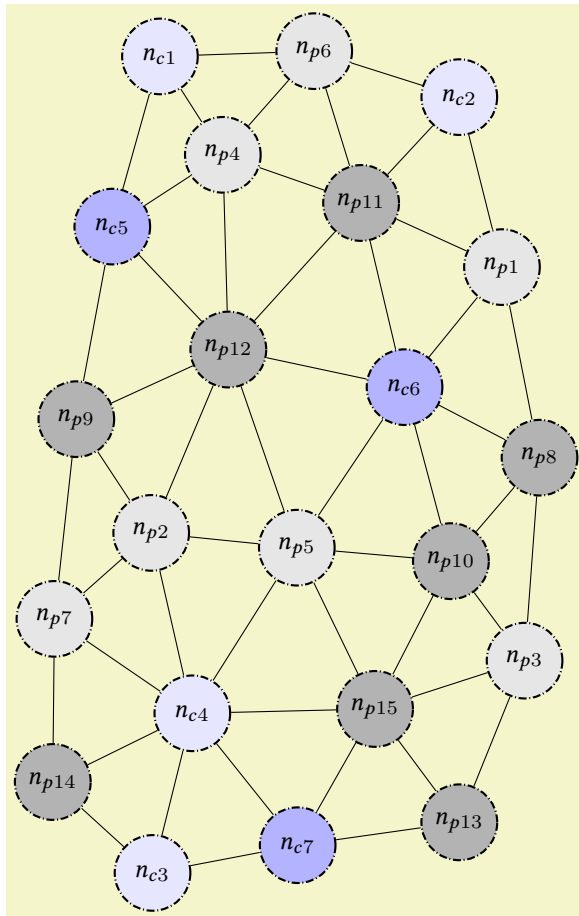
Sets up a transformation that mirrors along a line that goes through <point A> and <point B>.

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, 19] & [45, 32, 50, 43, 29]

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

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

1. Centered
2. Untipped
3. Overtipped⁷

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

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

⁷The Overtipped arrow tips aren't yet implemented.

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.

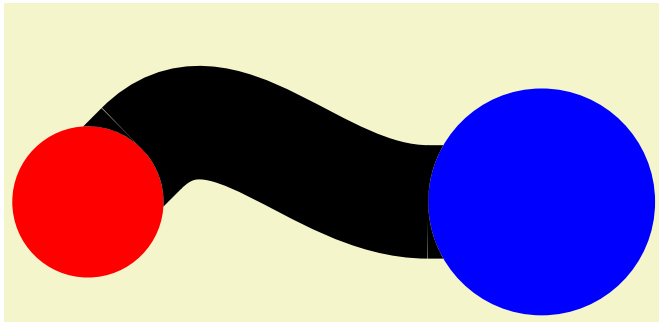
21.3 Original Arrow Tips

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.

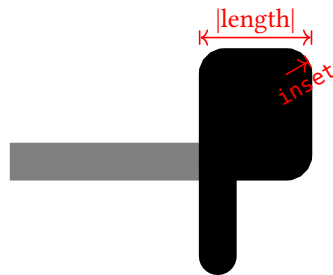


```
\usepgflibrary {ext.arrows}
\begin{tikzpicture}[
  dot/.style 2 args={
    shape=circle, outer sep=+0pt, fill={#1}, minimum size={#2}}]
\node[dot={red} {2cm}] (A) {};
\node[dot={blue}{3cm}] (B) at (6,0) {};
\draw[
  line width=1.5cm,
  arrows={ext_Hug Cap[length=1cm]-ext_Hug Cap[length=1.5cm]}
]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

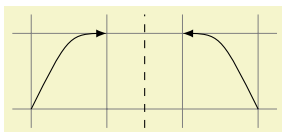
The following commands use the reflection matrix that sets the transformation matrix following

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

`\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

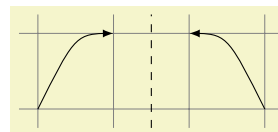
The following commands use a combination of shifting, rotating, -1 scaling, rotating back and shifting back to reach the mirror transformation.

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⟩}`

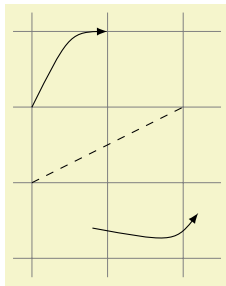
pre 0.6 `\pgftransformymirror`

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⟩}`

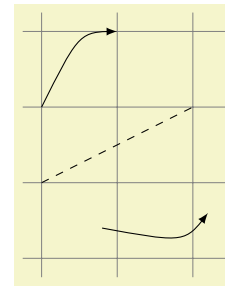
pre 0.6 `\pgftransformyMirror`

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);

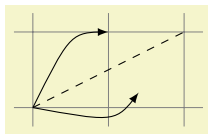
\draw[dashed] (0, -1) -- (2, 0);
\pgfexttransformMirror{\pgfpointxy{0}{-1}}
{\pgfpointxy{2}{0}}

\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);

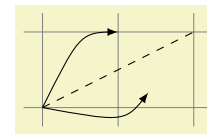
\draw[dashed] (0, 0) -- (2, 1);
\pgfextqttransformmirror{\pgfpointxy{2}{1}}

\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: [27] & [47]

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

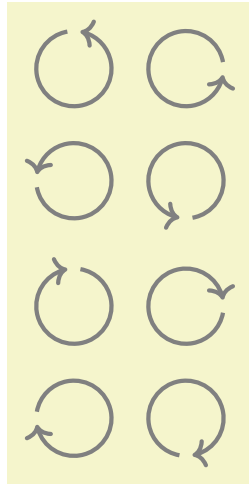
Sets `circle arrow start angle = 100`, `circle arrow delta angle = 340` and `circle arrow arrows = ->`.

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

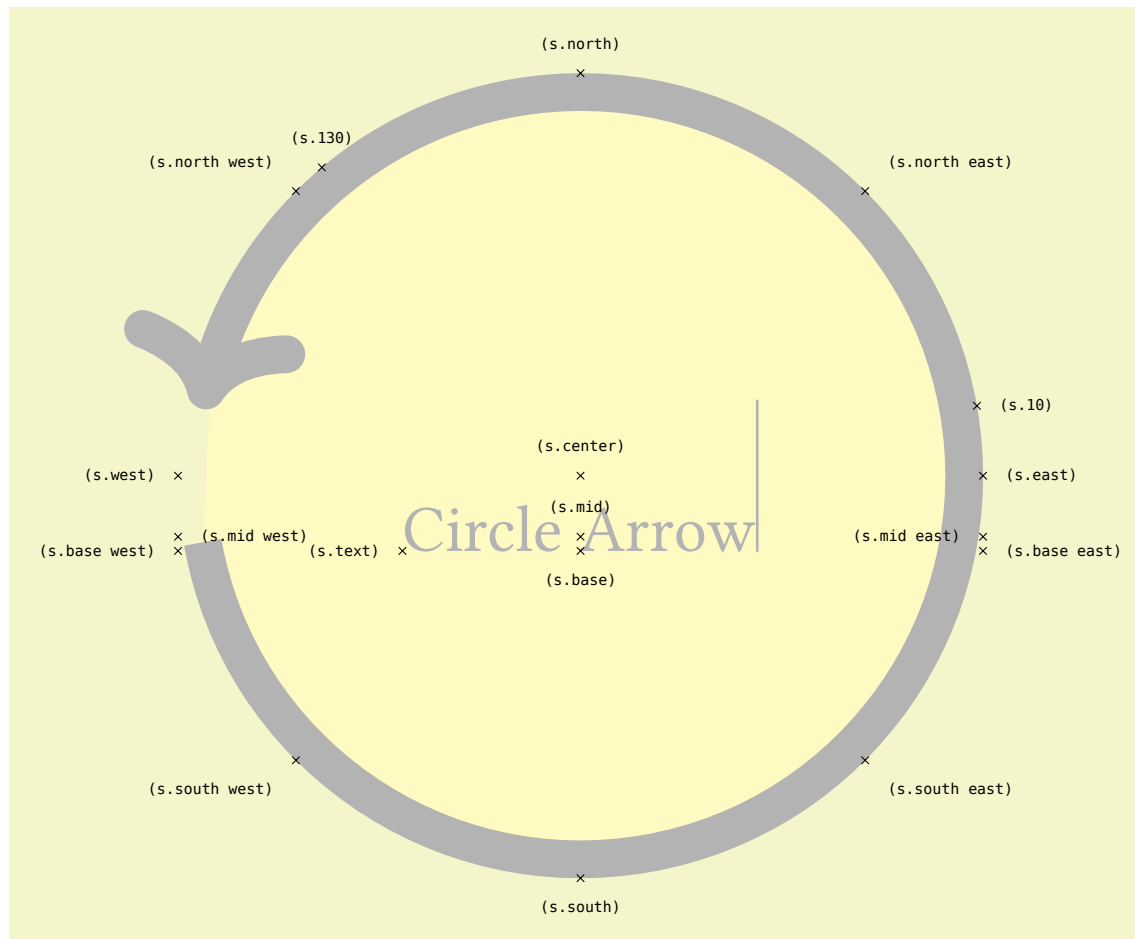
pre 0.6 /pgf/circle arrow turn left east

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 = <-. </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: [20] & [48]

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)
```

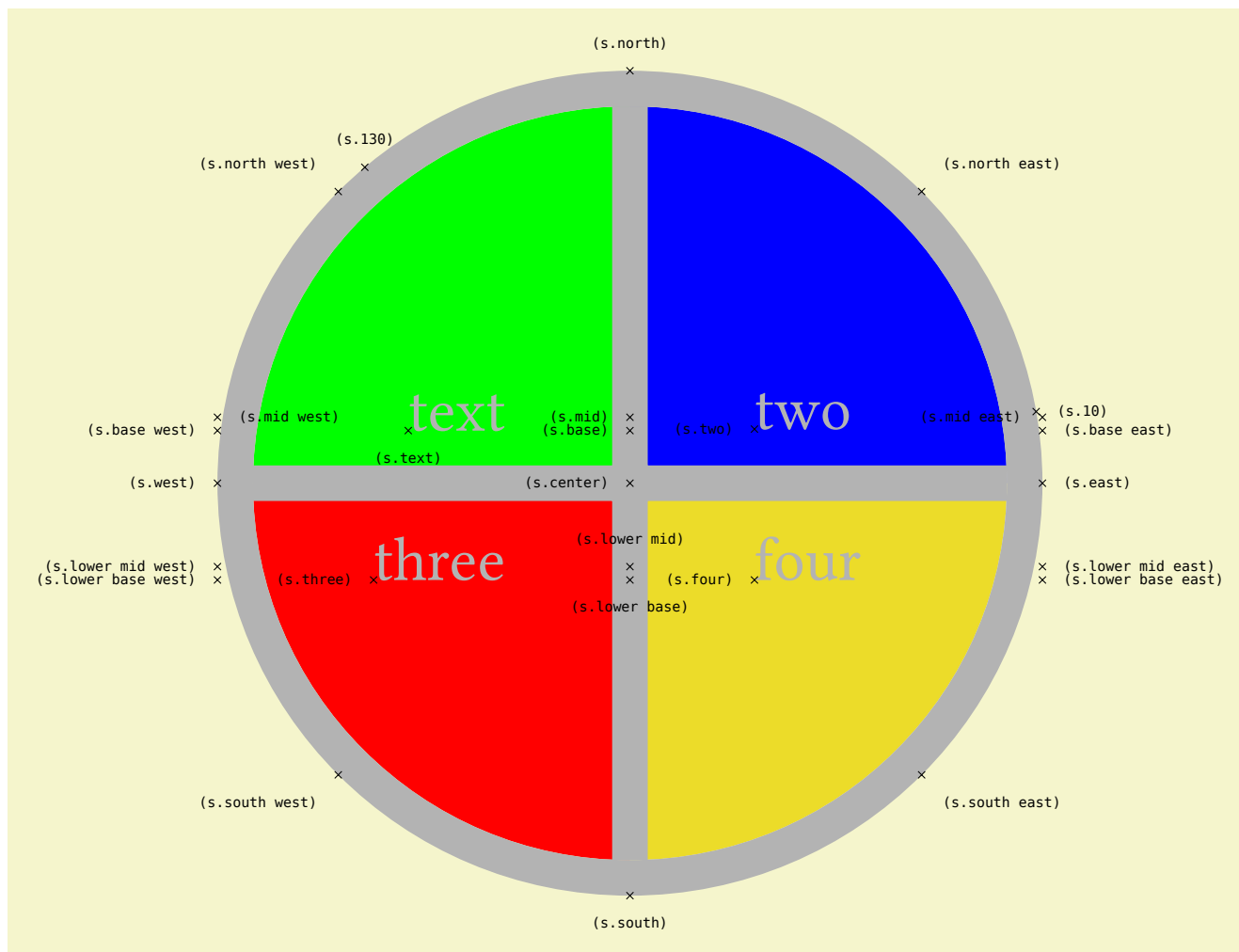
pre 0.6 /pgf/circle cross split part fill

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_EX-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] & [38]

Shape `ext_heatmark`

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

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)

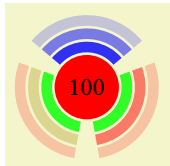
`/pgf/ext/heatmark arc <arc number>`

(style, no value)

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

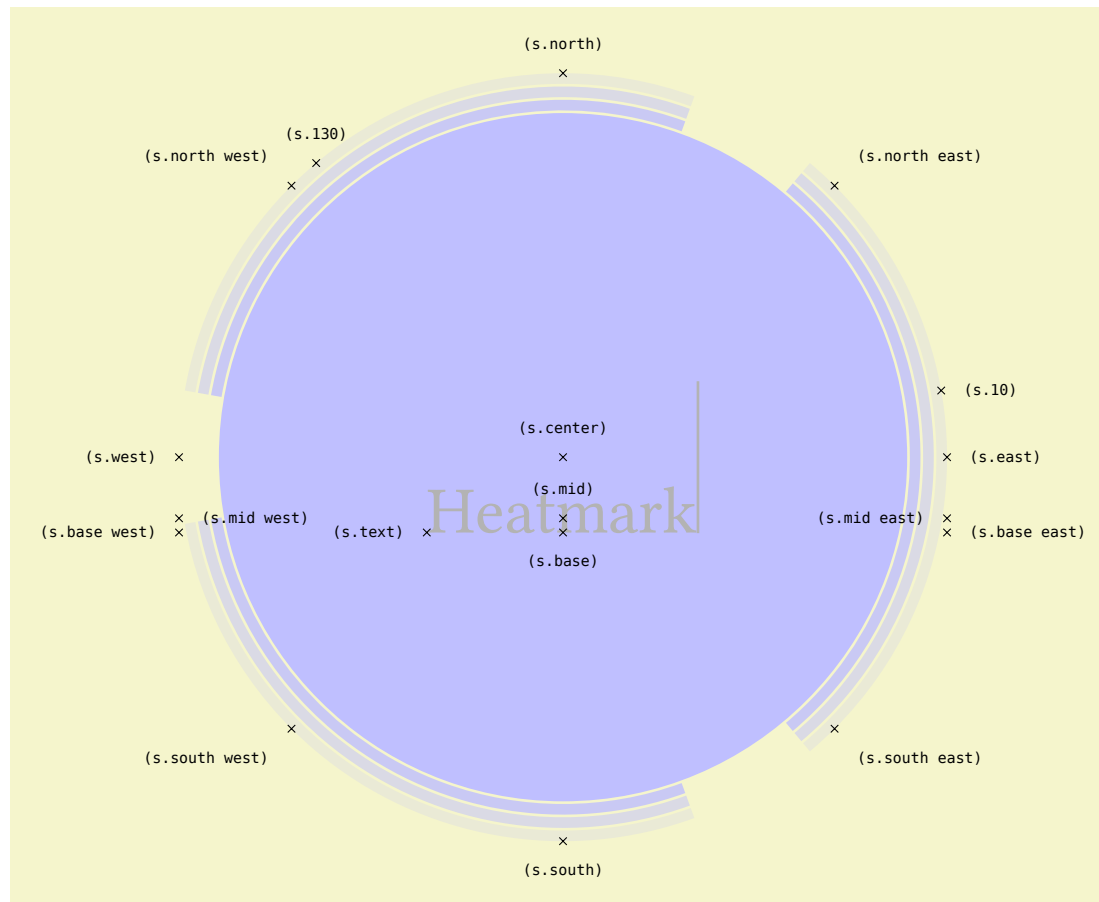
(style, no value)

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.

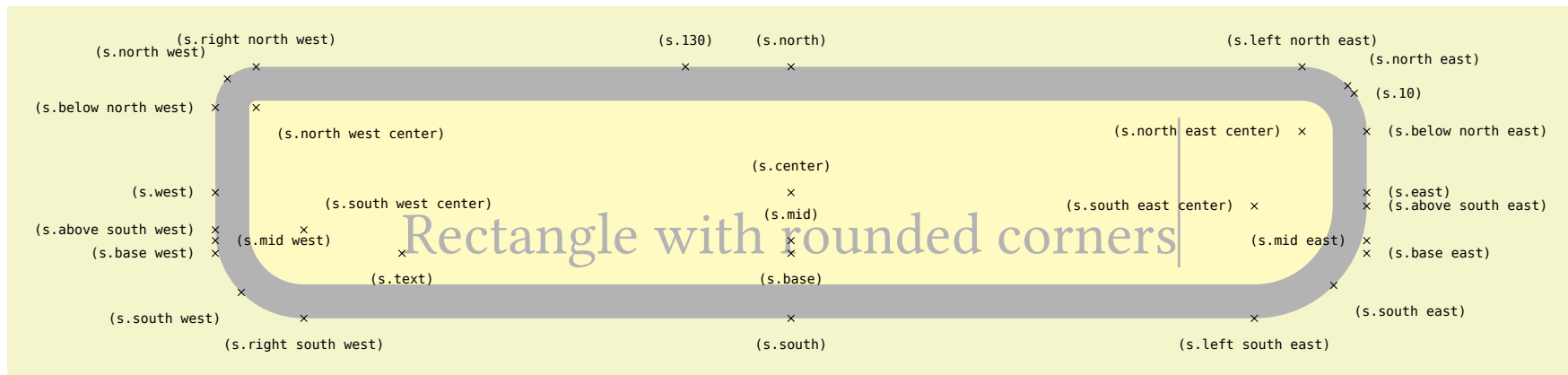
```
/pgf/ext/rectangle with rounded corners north west radius=<dimen> (no default, initially .5\pgflinewidth)
pre 0.6 /pgf/rectangle with rounded corners north west radius
Sets the north west radius to <dimen>.
```

```
/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: [56] & [31]

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

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

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

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

`/pgf/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 }`

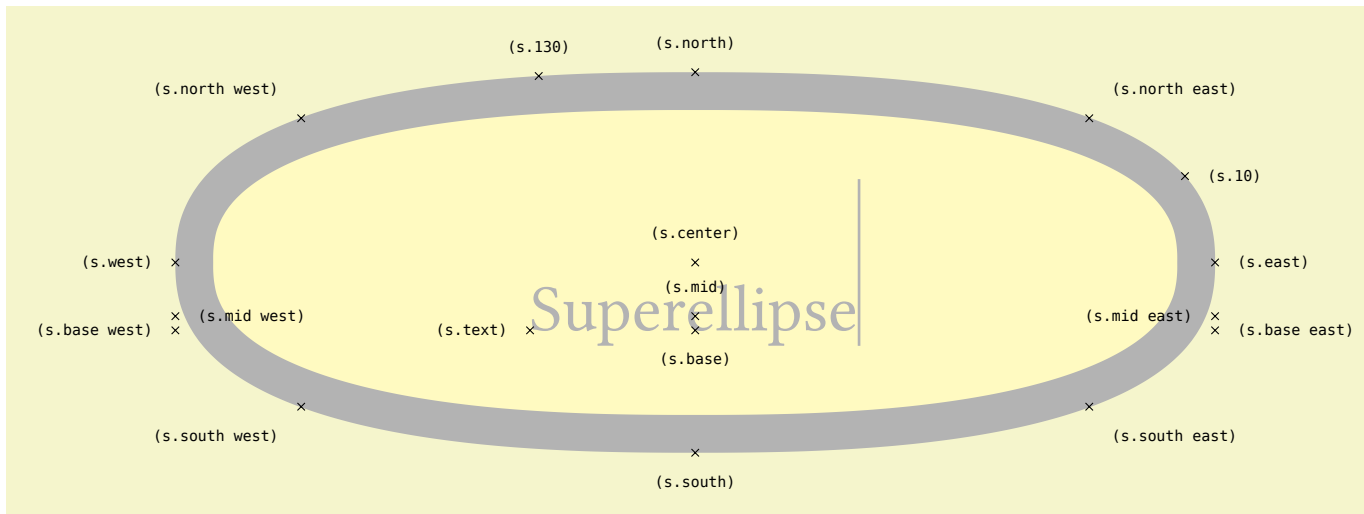
Returns the x value (in `\pgfmathresultX`) and the y value (in `\pgfmathresultY`) of the superellipse with its center on the origin following

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

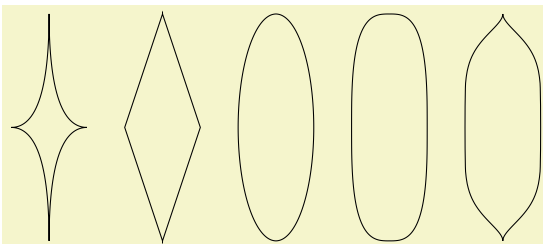
$$y = b \cos^{2/n} t$$

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: [59, 26] & [39, 36]

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`.

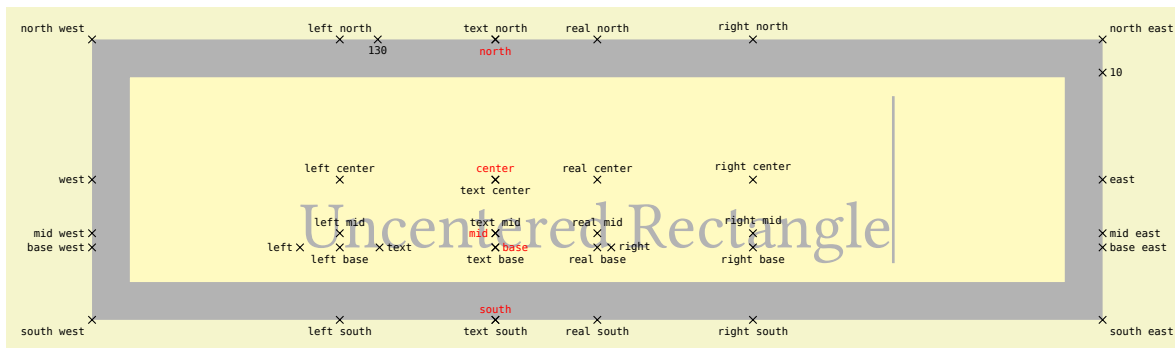
`/pgf/ext/uncentered rectangle center yshift=<dimension>` (no default, initially `{}`)

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}` % \LaTeX and plain \TeX

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

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

Q: [57]

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.⁸

- 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.

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

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

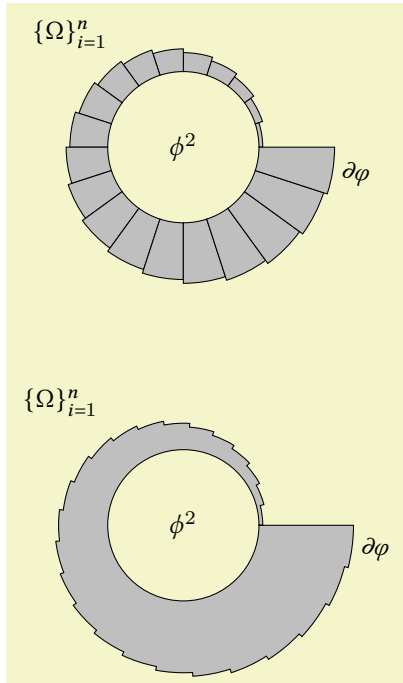
$$\begin{array}{ccc} S \supset U_\tau & \xrightarrow{\varphi_0} & U_\pi \subset T \\ \sim \downarrow \tau & & \sim \downarrow \pi \\ \text{Bl}_{(0,0)}(\mathbb{A}^2) \supset V_\tau & \xrightarrow{\epsilon} & V_\pi \subset \mathbb{A}^2 \end{array}$$

```
\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"]
  & & \uncrec{U_\pi}{} \subset T \arrow[d, "\tau", "\sim"]
  & & \uncrec{V_\pi}{} \subset \mathbb{A}^2 \arrow[d, "\pi", "\sim"]
  \uncrec{\operatorname{Bl}_{(0,0)}(\mathbb{A}^2) \supset {} }{V_\tau}{} \arrow[r, "\epsilon"]
  & & \uncrec{V_\pi}{} \subset \mathbb{A}^2
\end{tikzcd}
```

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

Part IV

Utilities



```
\usetikzlibrary {ext.misc}
\begin{tikzpicture}[
  declare function={bigR(\n)=smallR+.05*\n;},
  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=+IR] -- (\iN R+1R:smallR)
  arc [radius=smallR, end angle=\iN R, delta angle=-IR] -- cycle;

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

\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=IR]}
    -- (right:smallR) arc[radius=smallR, start angle=0, delta angle=-360];

\node                                {$\phi^2$};
\node at (north west:{sqrt 2 * bigR(segments/2)}) {$\{\Omega\}_{i=1}^n$};
\node[rotate=-.5R, right] at (-.5R: bigR segments) {$\partial \varphi$};
\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, 16] & [30, 51, 37]

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, 60] & [40, 46, 42]

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_EX isn't used. Spaces will be ignored.

B	X	X
-	-	-
W	X	X

```
\usetikzlibrary {ext.misc}
\newcommand*{\board}[3][[]]{%
  \begin{tikzpicture}[#1]
    \foreach[
      count=\i from 0,
      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={⟨argument specification⟩}{⟨foreach value⟩}` (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 `{⟨argument specification⟩}` get appended `u,.` This means there's always one additional mandatory argument at the end of every element.
- The `{⟨foreach value⟩}` needs to correspond to the `/pgf/foreach/var` value.

`/pgf/foreach/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={⟨argument specification⟩}{⟨comma-separated list of values⟩}`

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 `⟨comma-separated list of values⟩` is processed using `\foreach` and the given `xparse ⟨argument specification⟩` 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: [25] & [28]

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°.

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
```

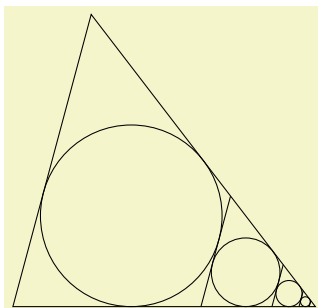
```
extatanYX(y,x)
pre 0.6 atanYX
\pgfmathextatanYX{y}{x}
```

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 `"`.



```
\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}
```

```
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*.

31.2 PGFfor

This library loads also most of the functions of the pgffor-ext of section 30 on page 81.

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 *⟨cond⟩* and applies the styles *⟨true⟩* if *⟨cond⟩* is true, otherwise *⟨false⟩*. *⟨cond⟩* can be anything that pgfmath understands.

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

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

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

pre 0.6 /utils/TeX/if

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

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

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

pre 0.6 /utils/TeX/ifx

As above but via *\ifx*.

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

pre 0.6 /utils/TeX/ifnum

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

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

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

pre 0.6 /utils/TeX/ifdim

As above but with *\ifdim*.

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

pre 0.6 /utils/TeX/ifempty

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

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

pre 0.6 /utils/TeX/ifxempty

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

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 *⟨key⟩/.ext_pgfm_{ath}=⟨eval⟩*

pre 0.6 .pgfm_{ath}

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

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

Key handler *⟨key⟩/.ext_pgfm_{ath} int=⟨eval⟩*

```
pre 0.6 .pgfmath int
```

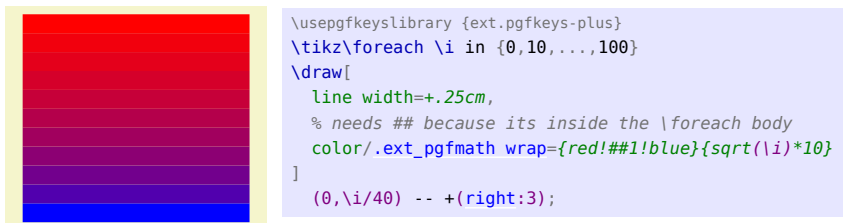
As above but truncates the result.

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 $\langle false \rangle$.

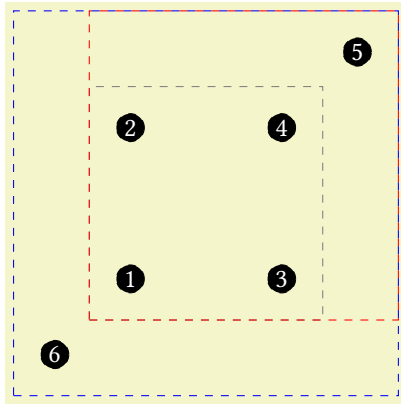
Key handler $\langle key \rangle /.ext_ifempty=\{\langle Text \rangle\}\{\langle true \rangle\}\{\langle false \rangle\}$

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)}
\node[circle, fill, inner sep=1pt, text=white] (point-\cnt) at \point {\cnt};
\node[gray, fit/.ext_List={(point-1),(point-...),(point-4)}] {};
\node[red, fit/.ext_List={(point-1),(point-...),(point-5)}] {};
\node[blue, fit/.ext_List={(point-1),(point-...),(point-6)}] {};
\end{tikzpicture}
```

31.4 TikZ

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

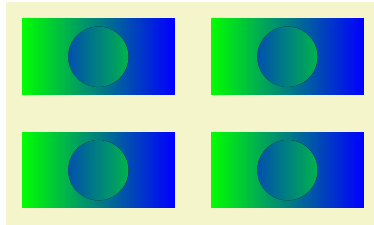
pre 0.6 /tikz/reverse clip

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 directly⁹ 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}
```

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

Part V

Changelog, Index & References

Changelog

Version 0.6.3 (2025-05-21)

- Small bugfix to `ext.paths.ortho`.
- Added node pictures to `ext.nodes`.

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`. [55]

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`. [22]

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 [21].

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, [26](#)
- | - | path operation, [33](#)
- | - path operation, [33](#)
- | path operation, [33](#)
- .< handler, [17](#)
- | - path operation, [33](#)

- above key, [41](#)
- above left key, [41](#)
- above right key, [41](#)
- alt key, [15](#), [16](#)
- aobs invisible key, [15](#)
- aobs visible key, [15](#)
- aobs-tikz package, [13](#)
- aot key, [16](#)
- arc path operation, [10](#), [60](#)
- arc arrows key, [10](#), [11](#)
- Arc Barb arrow tip, [54](#), [55](#)
- arc through key, [45](#)
- arc to path operation, [30](#)
- \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, [54](#), [55](#)
 - Bar, [54](#), [55](#)
 - Bracket, [54](#), [55](#)
 - Circle, [54](#), [55](#)
 - Diamond, [54](#)
 - Ellipse, [54](#), [55](#)
 - ext_Centered Arc Barb, [54](#), [55](#)
 - ext_Centered Bar, [54](#)
 - ext_Centered Bracket, [54](#)
 - ext_Centered Circle, [54](#)
 - ext_Centered Diamond, [54](#)
 - ext_Centered Ellipse, [54](#)
 - ext_Centered Hooks, [54](#)
 - ext_Centered Kite, [54](#)
 - ext_Centered Parenthesis, [54](#)
 - ext_Centered Rays, [55](#)
 - ext_Centered Rectangle, [54](#)
 - ext_Centered Square, [54](#)
 - ext_Centered Stealth, [54](#)
 - ext_Centered Straight Barb, [54](#)
 - ext_Centered Tee Barb, [54](#)
 - ext_Centered Triangle, [54](#)
 - ext_Centered Turned Square, [55](#)
 - ext_Double Cap, [56](#)
 - ext_Double Stealth, [56](#)
 - ext_Double Triangle, [56](#)
 - ext_Hug Cap, [56](#)
 - ext_Loop, [56](#)
 - ext_Untipped Bar, [55](#)
 - ext_Untipped Bracket, [55](#)
 - ext_Untipped Circle, [55](#)
 - ext_Untipped Ellipse, [55](#)
 - ext_Untipped Parenthesis, [55](#)
 - ext_Untipped Tee Barb, [55](#)
 - Hooks, [54](#)
 - Kite, [54](#)
 - Parenthesis, [54](#), [55](#)
 - Rays, [55](#)
 - Rectangle, [54](#)
 - Square, [54](#)
 - Stealth, [54](#), [56](#)
 - Straight Barb, [54](#)
 - Tee Barb, [54](#), [55](#)
 - Triangle, [54](#), [56](#)
 - Turned Square, [55](#)

- `\arrowreversed`, 11
- arrows key, 7
- `arrows.meta` library, 53
- `atan2` math function, 58, 84
- auto key, 27, 28
- auto offset key, 28
- auto offset for brace decoration key, 28
- auto with offset key, 28
- autobend down key, 46
- autobend east key, 46
- autobend left key, 46
- autobend north key, 46
- autobend right key, 46
- autobend south key, 46
- autobend up key, 46
- autobend west key, 46
- background code key, 19
- Bar arrow tip, 54, 55
- base left key, 41
- base right key, 41
- beamer function key, 14
- beamer shortcuts key, 15
- below key, 41
- below left key, 41
- below right key, 41
- bend left key, 46
- better beamer shortcut key, 17
- brace decoration, 28
- Bracket arrow tip, 54, 55
- `calc` library, 84
- calendar library, 18
- `cd` library, 75, 76
- center suffix key, 45
- Circle arrow tip, 54, 55
- circle path operation, 10, 88
- circle shape, 20
- circle arrow arrows key, 60
- circle arrow delta angle key, 60
- circle arrow end angle key, 60

- circle arrow start angle key, 60
- circle arrow turn left east key, 60
- circle arrow turn left north key, 60
- circle arrow turn left south key, 61
- circle arrow turn left west key, 61
- circle arrow turn right east key, 61
- circle arrow turn right north key, 61
- circle arrow turn right south key, 61
- circle arrow turn right west key, 61
- circle cross split part fill key, 64
- circle cross split uses custom fill key, 64
- clip rule key, 88
- clockwise key, 30, 45
- code key, 19
- compat key, 7
- corner above left key, 42
- corner above right key, 42
- corner below left key, 42
- corner below right key, 42
- corner east above key, 42
- corner east below key, 42
- corner north left key, 42
- corner north right key, 42
- corner south left key, 42
- corner south right key, 42
- corner west above key, 42
- corner west below key, 42
- `cos` math function, 58
- `cos` path operation, 10, 37
- counter clockwise key, 30, 45
- cover key, 14

Date tests

- ext/and, 79
- ext/Apr, 79
- ext/Aug, 79
- ext/calendar week of month, 79
- ext/calendar week of month', 79
- ext/Dec, 79
- ext/Feb, 79
- ext/Jan, 79

- ext/Jul, 79
- ext/Jun, 79
- ext/leap year, 79
- ext/Mar, 79
- ext/May, 79
- ext/not, 79
- ext/Nov, 79
- ext/Oct, 79
- ext/Sep, 79
- ext/week, 79
- ext/week of month, 79
- ext/week of month', 79
- ext/yesterday, 79
- day code key, 18
- day text key, 18
- day xshift key, 18
- day yshift key, 18
- Decorations
 - brace, 28
 - markings, 9
- decorations.markings library, 11
- decorations.pathreplacing library, 28
- Diamond arrow tip, 54
- \dimexpr, 43
- distance key, 33, 34
- down horizontal up key, 35
- du distance key, 34
- east above key, 41
- east below key, 41
- edge path operation, 45
- edge on layer key, 19
- Ellipse arrow tip, 54, 55
- ellipse path operation, 10
- enable first char key, 16
- enable first char < key, 16
- enable handler key, 17
- Environments
 - pgfonlayer, 19
 - tikzcd, 77
- evaluate key, 86

- .evaluated handler, 85
- even odd rule key, 88
- every arc arrows key, 11
- every arc to key, 31
- every brace node key, 28
- every day key, 18
- every diagram key, 76
- every month key, 18
- every node picture key, 25
- every softpath arrows key, 11
- every week key, 18
- execute at end node key, 77
- execute at end picture key, 20
- ext.arrows library, 9, 53
- ext.arrows-plus library, 9
- ext.beamer library, 13
- ext.calendar-plus library, 18
- ext.layers library, 19
- ext.misc library, 81–83
- ext.node-families library, 20
- ext.node-families.shapes.geometric library, 23
- ext.nodes library, 24
- ext.paths.arcto library, 30
- ext.paths.ortho library, 32
- ext.paths.timer library, 10, 36
- ext.patterns.images library, 39
- ext.pgfkeys-plus pgfkeys library, 85
- ext.positioning-plus library, 40
- ext.scalepicture library, 43
- ext.shapes.circlearrow library, 60
- ext.shapes.circlecrosssplit library, 64
- ext.shapes.heatmark library, 67
- ext.shapes.rectangleroundedcorners library, 70
- ext.shapes.superellipse library, 72
- ext.shapes.uncenteredrectangle library, 75, 76
- ext.topaths.arctthrough library, 45
- ext.topaths.autobend library, 46
- ext.transformations.mirror library, 48, 58
- ext/and date test, 79
- ext/Apr date test, 79
- ext/arrow pic type, 10

- ext/Aug date test, [79](#)
- ext/calendar week of month date test, [79](#)
- ext/calendar week of month' date test, [79](#)
- ext/Dec date test, [79](#)
- ext/Feb date test, [79](#)
- ext/Jan date test, [79](#)
- ext/Jul date test, [79](#)
- ext/Jun date test, [79](#)
- ext/leap year date test, [79](#)
- ext/Mar date test, [79](#)
- ext/May date test, [79](#)
- ext/not date test, [79](#)
- ext/Nov date test, [79](#)
- ext/Oct date test, [79](#)
- ext/Sep date test, [79](#)
- ext/softpath arrow pic type, [10](#)
- ext/softpath arrows pic type, [10](#)
- ext/week date test, [79](#)
- ext/week of month date test, [79](#)
- ext/week of month' date test, [79](#)
- ext/yesterday date test, [79](#)
- .ext_< handler, [17](#)
- ext_Centered Arc Barb arrow tip, [54, 55](#)
- ext_Centered Bar arrow tip, [54](#)
- ext_Centered Bracket arrow tip, [54](#)
- ext_Centered Circle arrow tip, [54](#)
- ext_Centered Diamond arrow tip, [54](#)
- ext_Centered Ellipse arrow tip, [54](#)
- ext_Centered Hooks arrow tip, [54](#)
- ext_Centered Kite arrow tip, [54](#)
- ext_Centered Parenthesis arrow tip, [54](#)
- ext_Centered Rays arrow tip, [55](#)
- ext_Centered Rectangle arrow tip, [54](#)
- ext_Centered Square arrow tip, [54](#)
- ext_Centered Stealth arrow tip, [54](#)
- ext_Centered Straight Barb arrow tip, [54](#)
- ext_Centered Tee Barb arrow tip, [54](#)
- ext_Centered Triangle arrow tip, [54](#)
- ext_Centered Turned Square arrow tip, [55](#)
- ext_circle arrow shape, [60](#)
- ext_circle cross split shape, [64](#)

- ext_Double Cap arrow tip, [56](#)
- ext_Double Stealth arrow tip, [56](#)
- ext_Double Triangle arrow tip, [56](#)
- ext_heatmark shape, [67](#)
- ext_Hug Cap arrow tip, [56](#)
- .ext_if handler, [86](#)
- .ext_ifdim handler, [86](#)
- .ext_ifempty handler, [86](#)
- .ext_ifnum handler, [86](#)
- .ext_ifx handler, [86](#)
- .ext_ifxempty handler, [86](#)
- ext_lastdayinmonthofyear math function, [18](#)
- .ext_List handler, [86](#)
- .ext_list xparse handler, [82](#)
- ext_Loop arrow tip, [56](#)
- .ext_pgfmath handler, [85](#)
- .ext_pgfmath if handler, [86](#)
- .ext_pgfmath int handler, [85](#)
- .ext_pgfmath wrap handler, [86](#)
- ext_rectangle with rounded corners shape, [70](#)
- ext_superellipse shape, [72](#)
- ext_superellipsex math function, [73](#)
- ext_superellipsey math function, [73](#)
- ext_uncentered rectangle shape, [75](#)
- ext_Untipped Bar arrow tip, [55](#)
- ext_Untipped Bracket arrow tip, [55](#)
- ext_Untipped Circle arrow tip, [55](#)
- ext_Untipped Ellipse arrow tip, [55](#)
- ext_Untipped Parenthesis arrow tip, [55](#)
- ext_Untipped Tee Barb arrow tip, [55](#)
- ext_weeksinmonthofyear math function, [18](#)
- extanglebetween math function, [84](#)
- extatanXY math function, [84](#)
- extatanYX math function, [84](#)
- extdistancebetween math function, [84](#)
- external library, [13, 20, 43](#)
- extisEmpty math function, [83](#)
- extisInString math function, [83](#)
- extqanglebetween math function, [84](#)
- extqdistancebetween math function, [84](#)
- extstrcat math function, [83](#)

extstrrepeat math function, 83

first char key, 16

fit library, 40

fit bounding box key, 41

foreground code key, 19

from center key, 33

full arc key, 83

graphs library, 16

grid path operation, 10

heatmark arc *(arc number)* key, 68

heatmark arc rings key, 67

heatmark arc sep key, 67

heatmark arc sep angle key, 67

heatmark arc width key, 67

heatmark arcs key, 67

heatmark inner opacity key, 67

heatmark outer opacity key, 67

heatmark ring *(ring number)* key, 68

heatmark ring *(ring number)* arc *(arc number)* key, 68

height key, 21, 23

Hooks arrow tip, 54

horizontal vertical key, 35

horizontal vertical horizontal key, 35

hvh distance key, 33

hvh ratio key, 33

hvvh distance key, 33

hvvh ratio key, 33

if key, 18, 85

\ifdate, 80

ifdim key, 85

ifempty key, 85

ifnum key, 85

ifx key, 85

ifxempty key, 85

ignore line width key, 13

image as pattern key, 39

install auto key, 28

install auto offset for brace decoration key, 28

install shortcuts key, 35

install uncentered rectangle in column key, 77

intersections library, 26

invisible key, 14

Key handlers

.<, 17

.evaluated, 85

.ext_<, 17

.ext_List, 86

.ext_if, 86

.ext_ifdim, 86

.ext_ifempty, 86

.ext_ifnum, 86

.ext_ifx, 86

.ext_ifxempty, 86

.ext_list xparse, 82

.ext_pgfmath, 85

.ext_pgfmath if, 86

.ext_pgfmath int, 85

.ext_pgfmath wrap, 86

Kite arrow tip, 54

large key, 31

layers key, 7

left key, 41

left vertical right key, 35

length key, 56

Libraries

arrows.meta, 53

calc, 84

calendar, 18

cd, 75, 76

decorations.markings, 11

decorations.pathreplacing, 28

ext.arrows, 9, 53

ext.arrows-plus, 9

ext.beamer, 13

ext.calendar-plus, 18

ext.layers, 19

ext.misc, 81–83

- ext.node-families, 20
- ext.node-families.shapes.geometric, 23
- ext.nodes, 24
- ext.paths.arcto, 30
- ext.paths.ortho, 32
- ext.paths.timer, 10, 36
- ext.patterns.images, 39
- ext.positioning-plus, 40
- ext.scalepicture, 43
- ext.shapes.circlearrow, 60
- ext.shapes.circlecrosssplit, 64
- ext.shapes.heatmark, 67
- ext.shapes.rectangleroundedcorners, 70
- ext.shapes.superellipse, 72
- ext.shapes.uncenteredrectangle, 75, 76
- ext.topaths.arctthrough, 45
- ext.topaths.autobend, 46
- ext.transformations.mirror, 48, 58
- external, 13, 20, 43
- fit, 40
- graphs, 16
- intersections, 26
- markings, 26
- positioning, 40
- shapes.geometric, 23
- spath3, 26

lr distance key, 34

mark connection node key, 26

markings decoration, 9

markings library, 26

Math functions

- atan2, 58, 84
- cos, 58
- ext_lastdayinmonthofyear, 18
- ext_superellipsex, 73
- ext_superellipsey, 73
- ext_weeksinmonthofyear, 18
- extanglebetween, 84
- extatanXY, 84
- extatanYX, 84

- extdistancebetween, 84
- extisEmpty, 83
- extisInString, 83
- extqanglebetween, 84
- extqdistancebetween, 84
- extstrcat, 83
- extstrrepeat, 83
- sin, 58

math mode key, 77

Math operators

- R, 83

matrix of math nodes key, 77

matrix on layer key, 19

max bounding box key, 13

maximum picture height key, 43

maximum picture height* key, 44

maximum picture size key, 44

maximum picture width key, 43

maximum picture width* key, 44

memoize package, 13, 20, 43

mid left key, 41

mid right key, 41

middle 0 to 1 key, 32

minimum height key, 22, 41

minimum picture height key, 43

minimum picture height* key, 44

minimum picture size key, 43

minimum picture width key, 43

minimum picture width* key, 44

minimum width key, 21, 41

Mirror key, 51

mirror key, 49

Mirror x key, 50

mirror x key, 49

Mirror y key, 51

mirror y key, 49

month code key, 18

month text key, 18

month xshift key, 18

month yshift key, 18

- name key, 39
- name prefix key, 25
- name prefix .. key, 25
- no separator key, 81
- node on layer key, 19
- node on line key, 26
- node picture key, 25
- node-families key, 7
- \nodepart, 24
- nodes key, 7
- nodes on curve key, 26
- nodes on curve' key, 26
- nodes on line key, 26
- nonzero rule key, 88
- normal list key, 82
- north left key, 41
- north right key, 41

- on grid key, 41
- only key, 15, 16
- only horizontal first key, 35
- only horizontal second key, 35
- only vertical first key, 35
- only vertical second key, 35
- \onslide, 13
- option key, 39
- options key, 39
- outer sep key, 26

- Packages and files
 - aobs-tikz, 13
 - memoize, 13, 20, 43
 - pgfcalendar-ext, 79
 - pgffor, 81
 - pgffor-ext, 81, 85
 - xparse, 82
- parabola path operation, 10, 37
- Parenthesis arrow tip, 54, 55
- patch key, 19
- Path operations
 - , 26

- |-, 33
- |-, 33
- , 33
- |-, 33
- arc, 10, 60
- arc to, 30
- circle, 10, 88
- cos, 10, 37
- edge, 45
- ellipse, 10
- grid, 10
- parabola, 10, 37
- plot, 10
- r-du, 34
- r-lr, 34
- r-rl, 34
- r-ud, 34
- rectangle, 10, 36
- sin, 10, 37
- to, 45
- paths.arcto key, 7
- paths.ortho key, 7
- paths.timer key, 7
- patterns.images key, 7
- /pgf/
 - arrow keys/
 - length, 56
 - decoration/
 - mark connection node, 26
 - raise, 28
 - foreach/
 - evaluate, 86
 - var, 82
 - minimum height, 22, 41
 - minimum width, 21, 41
 - outer sep, 26
 - shape border rotate, 68
 - text, 39
- /pgf/ext/
 - circle arrow arrows, 60
 - circle arrow delta angle, 60

- circle arrow end angle, [60](#)
- circle arrow start angle, [60](#)
- circle arrow turn left east, [60](#)
- circle arrow turn left north, [60](#)
- circle arrow turn left south, [61](#)
- circle arrow turn left west, [61](#)
- circle arrow turn right east, [61](#)
- circle arrow turn right north, [61](#)
- circle arrow turn right south, [61](#)
- circle arrow turn right west, [61](#)
- circle cross split part fill, [64](#)
- circle cross split uses custom fill, [64](#)
- full arc, [83](#)
- heatmark arc *<arc number>*, [68](#)
- heatmark arc rings, [67](#)
- heatmark arc sep, [67](#)
- heatmark arc sep angle, [67](#)
- heatmark arc width, [67](#)
- heatmark arcs, [67](#)
- heatmark inner opacity, [67](#)
- heatmark outer opacity, [67](#)
- heatmark ring *<ring number>*, [68](#)
- heatmark ring *<ring number>* arc *<arc number>*, [68](#)
- rectangle with rounded corners north east radius, [70](#)
- rectangle with rounded corners north west radius, [70](#)
- rectangle with rounded corners radius, [70](#)
- rectangle with rounded corners south east radius, [70](#)
- rectangle with rounded corners south west radius, [70](#)
- superellipse exponent, [72](#)
- superellipse step, [72](#)
- superellipse x exponent, [72](#)
- superellipse y exponent, [72](#)
- uncentered rectangle center, [75](#), [77](#)
- uncentered rectangle center yshift, [75](#)
- uncentered rectangle use saved center, [75](#)
- /pgf/foreach/ext/
 - no separator, [81](#)
 - normal list, [82](#)
 - use float, [81](#)
 - use int, [81](#)
 - xparser, [82](#)

- xparser Om, [82](#)
- \pgfcalendar, [80](#)
- pgfcalendar-ext key, [7](#)
- pgfcalendar-ext package, [79](#)
- \pgfextcalendarcurrentweek, [80](#)
- \pgfextcalendarifdateweek, [80](#)
- \pgfextcalendarjulianyeartoweek, [80](#)
- \pgfextmathsuperellipseXY, [73](#)
- \pgfextqttransformMirror, [59](#)
- \pgfextqttransformmirror, [59](#)
- \pgfextset, [6](#)
- \pgfextttransformMirror, [59](#)
- \pgfextttransformmirror, [59](#)
- \pgfextttransformxMirror, [58](#)
- \pgfextttransformxmirror, [58](#)
- \pgfextttransformyMirror, [59](#)
- \pgfextttransformymirror, [59](#)
- pgffor package, [81](#)
- pgffor-ext key, [7](#)
- pgffor-ext package, [81](#), [85](#)
- pgfkeys Libraries
 - ext.pgfkeys-plus, [85](#)
- pgfkeys-plus key, [7](#)
- pgfmath if key, [85](#)
- \pgfmathanglebetweenpoints, [58](#)
- \pgfmathextanglebetween, [84](#)
- \pgfmathextatanXY, [84](#)
- \pgfmathextatanYX, [84](#)
- \pgfmathextdistancebetween, [84](#)
- \pgfmathextisEmpty, [83](#)
- \pgfmathextisInString, [83](#)
- \pgfmathextlastdayinmonthofyear, [18](#)
- \pgfmathextqanglebetween, [84](#)
- \pgfmathextqdistancebetween, [84](#)
- \pgfmathextstrcat, [83](#)
- \pgfmathextstrrepeat, [83](#)
- \pgfmathextsuperellipseX, [73](#)
- \pgfmathextsuperellipseY, [73](#)
- \pgfmathextweeksInmonthofyear, [18](#)
- pgfonlayer environment, [19](#)
- \pgfpatharcto, [31](#)

- `\pgfpointnormalised`, 58
- `\pgfsetarrows`, 60
- `\pgfseteorule`, 88
- `\pgfsetnonzerorule`, 88
- `\pgftext`, 39
- `pic` key, 24
- `pic on layer` key, 19
- Pic Types
 - `ext/arrow`, 10
 - `ext/softpath arrow`, 10
 - `ext/softpath arrows`, 10
- `picture height` key, 43
- `picture height*` key, 44
- `picture size*` key, 44
- `picture width` key, 43
- `picture width*` key, 44
- `plot path` operation, 10
- `pos` key, 32, 36
- `pos` key, 9
- `pos <` key, 9
- `pos < angle` key, 9
- `pos >` key, 9
- `pos > angle` key, 9
- `positioning` library, 40
- `positioning-plus` key, 7
- `precise auto angle` key, 28
- `prefix` key, 20
- R postfix math operator, 83
- `r-du` path operation, 34
- `r-lr` path operation, 34
- `r-rl` path operation, 34
- `r-ud` path operation, 34
- `radius` key, 31
- `raise` key, 28
- `ratio` key, 33
- Rays arrow tip, 55
- Rectangle arrow tip, 54
- `rectangle` path operation, 10, 36
- `rectangle` shape, 20
- `rectangle timer` key, 36

- `rectangle with rounded corners north east radius` key, 70
- `rectangle with rounded corners north west radius` key, 70
- `rectangle with rounded corners radius` key, 70
- `rectangle with rounded corners south east radius` key, 70
- `rectangle with rounded corners south west radius` key, 70
- `reset graphic state` key, 25
- `reverse clip` key, 88
- `right` key, 41
- `right vertical left` key, 35
- `rl distance` key, 34
- `rotate` key, 31
- `save picture size` key, 43
- `scalepicture` key, 7
- `setup shape` key, 21
- `shape border rotate` key, 68
- Shapes
 - `circle`, 20
 - `ext_circle arrow`, 60
 - `ext_circle cross split`, 64
 - `ext_heatmark`, 67
 - `ext_rectangle with rounded corners`, 70
 - `ext_superellipse`, 72
 - `ext_uncentered rectangle`, 75
 - `rectangle`, 20
- `shapes` key, 7
- `shapes.geometric` library, 23
- `sin` math function, 58
- `sin path` operation, 10, 37
- `size` key, 22
- `sloped` key, 27, 28
- `small` key, 31
- `softpath arrows` key, 11
- `south left` key, 41
- `south right` key, 41
- `spacing` key, 32
- `span` key, 41
- `span horizontal` key, 41
- `span vertical` key, 41
- `spath3` library, 26
- Square arrow tip, 54

- Stealth arrow tip, [54, 56](#)
- Straight Barb arrow tip, [54](#)
- superellipse exponent key, [72](#)
- superellipse step key, [72](#)
- superellipse x exponent key, [72](#)
- superellipse y exponent key, [72](#)
- swap key, [28](#)
- sync bounding box key, [14](#)

- Tee Barb arrow tip, [54, 55](#)

- temporal key, [15, 16](#)

- text key, [39](#)

- text key, [20](#)

- text depth key, [20](#)

- text height key, [20](#)

- text width key, [20](#)

- text width align key, [21](#)

- through key, [45](#)

- /tikz/

- alt, [16](#)

- auto, [27, 28](#)

- bend left, [46](#)

- commutative diagrams/

- every diagram, [76](#)

- install uncentered rectangle in column, [77](#)

- math mode, [77](#)

- day code, [18](#)

- day text, [18](#)

- day xshift, [18](#)

- day yshift, [18](#)

- even odd rule, [88](#)

- every day, [18](#)

- every month, [18](#)

- execute at end node, [77](#)

- execute at end picture, [20](#)

- if, [18](#)

- matrix of math nodes, [77](#)

- month code, [18](#)

- month text, [18](#)

- month xshift, [18](#)

- month yshift, [18](#)

- name prefix, [25](#)

- nonzero rule, [88](#)

- on grid, [41](#)

- only, [16](#)

- pics/

- background code, [19](#)

- code, [19](#)

- foreground code, [19](#)

- pos, [32, 36](#)

- pos, [9](#)

- sloped, [27, 28](#)

- swap, [28](#)

- temporal, [16](#)

- to path, [26, 35](#)

- unrec, [77](#)

- unrec math mode, [77](#)

- x radius, [31](#)

- y radius, [31](#)

- tikz-cd fix key, [76](#)

- /tikz/ext/

- above, [41](#)

- above left, [41](#)

- above right, [41](#)

- aobs invisible, [15](#)

- aobs visible, [15](#)

- arc arrows, [10, 11](#)

- arc through/

- center suffix, [45](#)

- clockwise, [45](#)

- counter clockwise, [45](#)

- through, [45](#)

- arc through, [45](#)

- arc to/

- clockwise, [30](#)

- counter clockwise, [30](#)

- large, [31](#)

- radius, [31](#)

- rotate, [31](#)

- small, [31](#)

- x radius, [31](#)

- y radius, [31](#)

- arrow shift factor, [10](#)
- arrow shift factor end, [10](#)
- arrow shift factor start, [10](#)
- arrow shift mode, [9](#)
- auto offset, [28](#)
- auto offset for brace decoration, [28](#)
- auto with offset, [28](#)
- autobend down, [46](#)
- autobend east, [46](#)
- autobend left, [46](#)
- autobend north, [46](#)
- autobend right, [46](#)
- autobend south, [46](#)
- autobend up, [46](#)
- autobend west, [46](#)
- base left, [41](#)
- base right, [41](#)
- 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, [41](#)
- below left, [41](#)
- below right, [41](#)
- clip rule, [88](#)
- 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, [42](#)
- corner above right, [42](#)
- corner below left, [42](#)
- corner below right, [42](#)
- corner east above, [42](#)
- corner east below, [42](#)
- corner north left, [42](#)
- corner north right, [42](#)
- corner south left, [42](#)
- corner south right, [42](#)
- corner west above, [42](#)
- corner west below, [42](#)
- cover, [14](#)
- down horizontal up, [35](#)
- east above, [41](#)
- east below, [41](#)
- edge on layer, [19](#)
- every arc arrows, [11](#)
- every arc to, [31](#)
- every brace node, [28](#)
- every node picture, [25](#)
- every softpath arrows, [11](#)
- every week, [18](#)
- fit bounding box, [41](#)
- horizontal vertical, [35](#)
- horizontal vertical horizontal, [35](#)
- ignore line width, [13](#)
- image as pattern/
 - name, [39](#)
 - option, [39](#)
 - options, [39](#)
- image as pattern, [39](#)
- invisible, [14](#)
- layers/
 - patch, [19](#)

- left, [41](#)
- left vertical right, [35](#)
- matrix on layer, [19](#)
- max bounding box, [13](#)
- maximum picture height, [43](#)
- maximum picture height*, [44](#)
- maximum picture size, [44](#)
- maximum picture width, [43](#)
- maximum picture width*, [44](#)
- mid left, [41](#)
- mid right, [41](#)
- minimum picture height, [43](#)
- minimum picture height*, [44](#)
- minimum picture size, [43](#)
- minimum picture width, [43](#)
- minimum picture width*, [44](#)
- Mirror, [51](#)
- mirror, [49](#)
- Mirror x, [50](#)
- mirror x, [49](#)
- Mirror y, [51](#)
- mirror y, [49](#)
- name prefix ..., [25](#)
- 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, [26](#)
- node picture/
 - reset graphic state, [25](#)
- node picture, [25](#)
- nodes/
 - install auto, [28](#)

- install auto offset for brace decoration, [28](#)
- nodes on curve, [26](#)
- nodes on curve', [26](#)
- nodes on line, [26](#)
- north left, [41](#)
- north right, [41](#)
- only horizontal first, [35](#)
- only horizontal second, [35](#)
- only vertical first, [35](#)
- only vertical second, [35](#)
- ortho/
 - distance, [33, 34](#)
 - du distance, [34](#)
 - from center, [33](#)
 - hvh distance, [33](#)
 - hvh ratio, [33](#)
 - hvvh distance, [33](#)
 - hvvh ratio, [33](#)
 - install shortcuts, [35](#)
 - lr distance, [34](#)
 - middle 0 to 1, [32](#)
 - ratio, [33](#)
 - rl distance, [34](#)
 - spacing, [32](#)
 - ud distance, [34](#)
 - udlr distance, [34](#)
 - vhv distance, [33](#)
 - vhv ratio, [33](#)
- pic, [24](#)
- pic on layer, [19](#)
- picture height, [43](#)
- picture height*, [44](#)
- picture size*, [44](#)
- picture width, [43](#)
- picture width*, [44](#)
- pos <, [9](#)
- pos < angle, [9](#)
- pos >, [9](#)
- pos > angle, [9](#)
- precise auto angle, [28](#)
- rectangle timer, [36](#)

- reverse clip, 88
- right, 41
- right vertical left, 35
- save picture size, 43
- softpath arrows, 11
- south left, 41
- south right, 41
- span, 41
- span horizontal, 41
- span vertical, 41
- sync bounding box, 14
- tikz-cd fix, 76
- uncover, 14
- up horizontal down, 35
- vertical horizontal, 35
- vertical horizontal vertical, 35
- visible, 14
- week code, 18
- week label left, 18
- week text, 18
- west above, 41
- west below, 41
- xMirror, 50
- xmirror, 48
- yMirror, 50
- ymirror, 49
- /tikz/graphs/ext/
 - better beamer shortcut, 17
- tikzcd environment, 77
- \tikzextpictureheight, 43
- \tikzextpicturewidth, 43
- \tikzextset, 6
- \tikzextsetupimageaspattern, 39
- \tikzextversion, 6
- \tikzextversionnumber, 6
- to path operation, 45
- to path key, 26, 35
- topaths.arctthrough key, 7
- transformations.mirror key, 7
- Triangle arrow tip, 54, 56
- Turned Square arrow tip, 55

- ud distance key, 34
- udlr distance key, 34
- uncentered rectangle center key, 75, 77
- uncentered rectangle center yshift key, 75
- uncentered rectangle use saved center key, 75
- uncover key, 14
- \uncrec, 77
- uncrec key, 77
- uncrec math mode key, 77
- up horizontal down key, 35
- use float key, 81
- use int key, 81
- /utils/ext/
 - alt, 15
 - if, 85
 - ifdim, 85
 - ifempty, 85
 - ifnum, 85
 - ifx, 85
 - ifxempty, 85
 - only, 15
 - pgfmath if, 85
 - temporal, 15
- var key, 82
- vertical horizontal key, 35
- vertical horizontal vertical key, 35
- vhv distance key, 33
- vhv ratio key, 33
- visible key, 14
- week code key, 18
- week label left key, 18
- week text key, 18
- west above key, 41
- west below key, 41
- width key, 21, 23
- x radius key, 31
- x radius key, 31
- xMirror key, 50

xmirror key, [48](#)
xparse package, [82](#)
xparser key, [82](#)
xparser Om key, [82](#)

y radius key, [31](#)
y radius key, [31](#)
yMirror key, [50](#)
ymirror key, [49](#)

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. 81).
- [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. 67).
- [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. 53).
- [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. 36).
- [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. 36).
- [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. 81).
- [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. 53).
- [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, 79).
- [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, 79).
- [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. 53).
- [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. 53).
- [16] jd6. *Full weeks in Tikz Calendar*. TeX - LaTeX Stack Exchange. Dec. 2020. URL: <https://tex.stackexchange.com/q/576673> (visited on 10/09/2022) (cit. on p. 79).
- [17] knut. *TikZ: Define pattern with reference to external picture*. Mar. 2013. URL: <https://tex.stackexchange.com/q/103980> (visited on 09/24/2022) (cit. on p. 39).
- [18] Ben Liblit. *path with both mark connection node and arrow tip*. TeX - LaTeX Stack Exchange. Feb. 2013. URL: <https://tex.stackexchange.com/q/99945> (visited on 12/12/2022) (cit. on p. 24).
- [19] 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. 53).
- [20] Marco. *TikZ - Four Colored Circle Split*. Apr. 2017. URL: <https://tex.stackexchange.com/q/121686> (visited on 09/24/2022) (cit. on p. 64).
- [21] marmotghost. *clockwise/counter clockwise does not seem to work reliably*. Oct. 2022. URL: <https://github.com/Qrrbrlbel/tikz-extensions/issues/2> (visited on 10/23/2022) (cit. on p. 89).

- [22] 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. 89).
- [23] 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. 46).
- [24] 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).
- [25] 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. 83).
- [26] 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. 75).
- [27] 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. 60).
- [28] 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. 83).
- [29] 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. 53).
- [30] 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, 79).
- [31] 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. 72).
- [32] 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. 53).
- [33] 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).
- [34] 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. 46).
- [35] 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).
- [36] 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. 75).
- [37] 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. 79).
- [38] 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. 67).
- [39] 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. 75).
- [40] 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. 81).

- [41] 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. 36).
- [42] 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. 81).
- [43] 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. 53).
- [44] 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).
- [45] 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. 53).
- [46] 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. 81).
- [47] 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. 60).
- [48] 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. 64).
- [49] 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).
- [50] 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. 53).
- [51] 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, 79).
- [52] 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. 39).
- [53] 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. 36).
- [54] 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. 25).
- [55] 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. 89).
- [56] 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. 72).
- [57] 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. 76).

- [58] 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. 26).
- [59] 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. 75).
- [60] 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. 81).
- [61] 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, 43).