# The TikZ-Extensions Package

Manual for version 0.4

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

Qrrbrbirlbel

October 9, 2022

## Contents

### I Introduction

1. Usage 4
2. Why do we need it? 4
3. Having problems? 4

### II TikZ Libraries

4. Calendar 5
   4.1 Value-keys and nestable if key 6
   4.2 pgfmath functions 6
   4.3 Week numbering (ISO 8601) 6

5. Node Families 7
   5.1 Text Box 7
   5.2 Minimum Width/Height 8
   5.3 More shapes that support the keys width and height 9
Part I
Introduction

1 Usage

This package is called \texttt{tikz-ext}, however, one can’t load it via \texttt{\usepackage}. Instead, this package consists of multiple PGF and TikZ libraries which are loaded by either \texttt{\usepgflibrary} or \texttt{\usetikzlibrary}.

2 Why do we need it?

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

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

3 Having problems?

Don’t hesitate to open an issue on GitHub.

\footnote{Except for \texttt{calendar-ext} of section 19.}
Part II

TikZ Libraries

These libraries only work with TikZ.
4 Calendar

TikZ Library ext.calendar-plus

\usetikzlibrary{ext.calendar-plus} % \LaTeX{} and plain \TeX{}
\usetikzlibrary{ext.calendar-plus} % Con\TeX{}

This library extends the TikZ library calendar.

Q & A: [6, 7, 3] & [16, 26, 25]

4.1 Value-keys and nestable if key

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

/tikz/day xshift
(initially 3ex)

/tikz/day yshift
(initially 3.5ex)

/tikz/month xshift
(initially 9ex)

/tikz/month yshift
(initially 9ex)

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

/tikz/if=((conditions))(code or options)\texttt{else}(else code or options) (no default)

4.2 pgfmath functions

\pgfmathweeksinnmonthofyear(first weekday, month, year)
\pgfmathweeksinnmonthofyear(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.

\pgfmathlastdayinnmonthofyear(month, year)
\pgfmathlastdayinnmonthofyear(month){year}

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

4.3 Week numbering (ISO 8601)

The actual week number algorithm is implemented by the \pgfcalendar-ext package/module in section 19.2.

\tikz\texttt{week code=}(code) (no default)

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

\tikz\texttt{week text=}(text) (no default)

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

\tikz\texttt{every week} (style, no value)

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

\tikz\texttt{week label left} (style, no value)

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

\begin{tabular}{|c|c|c|c|c|c|}
\hline
\hline
July & & & July & 26 & 27 & 28 & 29 & 30 & 31 \\
\hline
\end{tabular}

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

TikZ Library \texttt{ext.node-families}
\begin{verbatim}
\usetikzlibrary{ext.node-families} % \LaTeX \ and plain \TeX
\usetikzlibrary{ext.node-families} % \ConTeXt
\end{verbatim}

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 \texttt{aux} file. Unfortunately, this does not work with the \texttt{external} library.\footnote{First of all, I can’t figure out how to use the \texttt{aux} file during externalization since it gets written to the \texttt{log} instead. And then there’s the question about how \texttt{external} would notice the need to export the picture again until it’s stable.}

Q \& A: [8] \& [18]

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

\begin{verbatim}
/tikz/node family/prefix=⟨prefix⟩
\end{verbatim}

The family names are prefixed with the value of \texttt{/tikz/node family/prefix}.

5.1 Text Box

The following keys – when setup, see below – work with every shape with one single node part.\footnote{Technically, it will also work with shapes with multiple node parts but it will only affect the main node part.} Initially though, only \texttt{circle} and \texttt{rectangle} are set up that way.

\begin{verbatim}
/tikz/node family/text height=⟨name⟩
\end{verbatim}

Nodes with the same \texttt{⟨name⟩} will have the same text height. An empty \texttt{⟨name⟩} disables the evaluation by the library.

\begin{verbatim}
/tikz/node family/text depth=⟨name⟩
\end{verbatim}

Nodes with the same \texttt{⟨name⟩} will have the same text depth. An empty \texttt{⟨name⟩} disables the evaluation by the library.

\begin{verbatim}
/tikz/node family/text width=⟨name⟩
\end{verbatim}

Nodes with the same \texttt{⟨name⟩} will have the same text width. An empty \texttt{⟨name⟩} disables the evaluation by the library.

\begin{verbatim}
/tikz/node family/text=⟨name⟩
\end{verbatim}

Sets text height, text depth and text width.

Since the width of the node’s content’s box is setup much earlier, the previous key only extends the width of that box which would make the text seem as if it where aligned to the left. With \texttt{text width family align} this can changed.
\tikz/node family/text width align=(alignment)

(alignment) is one of left, center or right.

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

\tikz/node family/setup shape=(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.

5.2 Minimum Width/Height

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

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

\tikz/node family/width=(name)

Nodes with the same (name) will have the same /pgf/minimum width. An empty (name) disables the evaluation by the library.

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

\tikz/node family/height=(name)

Nodes with the same (name) will have the same /pgf/minimum height. An empty (name) disables the evaluation by the library.

\tikz/node family/size=(name)

Sets both height and width.

(0 default, initially {\empty})
5.3 More shapes that support the keys width and height

\textbf{TikZ Library} \texttt{ext.node-families.shapes.geometric}
\begin{verbatim}
\usetikzlibrary{ext.node-families.shapes.geometric} % \LaTeX{} and plain \TeX{}
\usetikzlibrary{ext.node-families.shapes.geometric} % Con\TeX{}
\end{verbatim}

This library adds support for the keys \texttt{/tikz/node family/width} and \texttt{/tikz/node family/height} for the shapes of the PGF library \texttt{shapes.geometric}.

Q: [12]

The shapes are also setup for the keys from subsection 5.1.

\begin{tikzpicture}
\foreach \cnt [count=\Cnt] in {a,...,h}
\node[draw, diamond, node family/text=aTOh] (\cnt) at (right:\Cnt) {\cnt};
\draw[help lines] (a.south) -- (h.south) (a.north) -- (h.north) (a.base-|a.west) -- (h.base-|h.east);
\end{tikzpicture}
6 Arc to a point

TikZ Library `ext.paths.arcto`

\usetikzlibrary{ext.paths.arcto} \% \LaTeX \ and \ plain \ \TeX
\usetikzlibrary{ext.paths.arcto} \% Con\TeX

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

\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}

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 \((options)\).

\begin{example}
\texttt{/tikz/arc to/clockwise}
\end{example}

\begin{example}
\texttt{\path ... \textbf{arc to}[(options)](coordinate or cycle) ...;}
\end{example}
This constructs an arc that goes clockwise.

/tikz/arc to/counter clockwise

This constructs an arc that goes counter clockwise.

This is the default.

/tikz/arc to/large

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

/tikz/arc to/small

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

/tikz/arc to/rotate=⟨degree⟩

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

/tikz/arc to/x radius=⟨value⟩

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

/tikz/arc to/y radius=⟨value⟩

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

/tikz/arc to/radius=⟨value⟩

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

/tikz/every arc to

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

It should be noted that this uses \pgfpatharcto for which the 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 ⟨target coordinate⟩, but may be off by up to several points. A more precise positioning is currently infeasible due to \TeX’s numerical weaknesses. The only case it works quite nicely is when the resulting angle is a multiple of 90°.*

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

TikZ Library `ext.paths.ortho`

\usetikzlibrary{ext.paths.ortho} % \LaTeX{} and plain \TeX{}
\usetikzlibrary{ext.paths.ortho} % Con\TeX

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

7.1 Zig-Zag

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

\path ... |-|\begin{itemize}
\item \textit{(options)} \item \textit{(coordinate or cycle)} \end{itemize} ...;

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

\path ... -|-\begin{itemize}
\item \textit{(options)} \item \textit{(coordinate or cycle)} \end{itemize} ...;

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

/tikz/ortho/ratio=\begin{itemize}
\item \textit{(ratio)} \end{itemize}

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

For values \textit{(ratio)} < 0 and \textit{(ratio)} > 1 the Zig-Zag lines will look more like Zig-Zig lines.

/tikz/ortho/distance=\begin{itemize}
\item \textit{(distance)} \end{itemize}

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

For values \textit{(distance)} < 0 the distance will be used for the target coordinate.
/tikz/ortho/from center=(true or false)  
(default true)
When nodes get connected the placement of the middle part of the Zig-Zag and the Zig-Zig (see below) connections will be calculated from the border of these nodes. The middle part of the connections can be calculated from the nodes’ center if this key is set to true.

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

/tikz/ortho/spacing=(number)  
(no default, initially 4)
Unless (number) = 0 is set
- pos = 0 will be at the start,
- pos = 1 will be at the end,
- pos = \frac{1}{\text{(number)}} will be at the first kink,
- pos = \frac{\text{(number)}-1}{\text{(number)}} will be at the second kink and
- pos = .5 will be in the middle of the middle part of the connection.
If \(<\text{number}\) = 0 then

- \(\text{pos} = -1\) will be at the start,
- \(\text{pos} = 2\) will be at the end,
- \(\text{pos} = 0\) will be at the first kink,
- \(\text{pos} = 1\) will be at the second kink and
- \(\text{pos} = .5\) will still be in the middle of the middle part of the connection.

\texttt{/tikz/ortho/middle 0 to 1} \\
This is an alias for \texttt{spacing = 0}.

### 7.2 Zig-Zig

\texttt{\path ... r-ud[\langle options\rangle](coordinate or cycle) ... ;}

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

\texttt{/tikz/ortho/ud distance=\langle length\rangle}

This sets the distance between the start and the horizontal line to \(<\text{length}\). 

\texttt{\path ... r-du[\langle options\rangle](coordinate or cycle) ... ;}

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

\texttt{/tikz/ortho/du distance=\langle length\rangle}

This sets the distance between the start and the horizontal line to \(<\text{length}\).

\texttt{\path ... r-lr[\langle options\rangle](coordinate or cycle) ... ;}

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

\texttt{/tikz/ortho/lr distance=\langle length\rangle}

This sets the distance between the start and the vertical line to \(<\text{length}\).

\texttt{\path ... r-rl[\langle options\rangle](coordinate or cycle) ... ;}

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

\texttt{/tikz/ortho/rl distance=\langle length\rangle}

This sets the distance between the start and the vertical line to \(<\text{length}\).

All distances can be set with on key.

\texttt{/tikz/ortho/udlr distance=\langle length\rangle}

Sets all the previous distances to the same value \(<\text{length}\).
7.3 Even more Horizontal and Vertical Lines

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

/tikz/horizontal vertical (style, no value)
This installs \path = -| (\tikztotarget) \tikztonodes that can be used with the path operations to or edge.

/tikz/vertical horizontal (style, no value)
This installs \path = |- (\tikztotarget) \tikztonodes that can be used with the path operations to or edge.

/tikz/horizontal vertical horizontal (style, no value)
This installs \path = -|- (\tikztotarget) \tikztonodes that can be used with the path operations to or edge.

/tikz/vertical horizontal vertical (style, no value)
This installs \path = |-| (\tikztotarget) \tikztonodes that can be used with the path operations to or edge.

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

/tikz/only vertical second=⟨length⟩ (style, default 0pt)
This draws a vertical line from the start point to the target point so that it connects to the target point in the center (or at its border in case it is a node).
The optional ⟨length⟩ can be used to shift the line orthogonally to its direction.

/tikz/only horizontal second=⟨length⟩ (style, default 0pt)
This draws a horizontal line from the start point to the target point so that it connects to the target point in the center (or at its border in case it is a node).
The optional ⟨length⟩ can be used to shift the line orthogonally to its direction.

/tikz/only vertical first=⟨length⟩ (style, default 0pt)
This draws a vertical line from the start point to the target point so that it connects to the start point in the center (or at its border in case it is a node).
The optional ⟨length⟩ can be used to shift the line orthogonally to its direction.

/tikz/only horizontal first=⟨length⟩ (style, default 0pt)
This draws a horizontal line from the start point to the target point so that it connects to the start point in the center (or at its border in case it is a node).
The optional ⟨length⟩ can be used to shift the line orthogonally to its direction.
Since all previous key are rather cumbersome, one can install shortcuts for these.

/\texttt{tikz/ortho/install shortcuts} (style, no value)

Installs the following shortcuts:

- \texttt{|-} → vertical horizontal
- \texttt{-|} → horizontal vertical
- \texttt{--} → horizontal vertical horizontal
- \texttt{|--} → vertical horizontal vertical
- \texttt{|*} → only vertical first
- \texttt{*|} → only vertical second
- \texttt{|*} → only vertical first
- \texttt{*|} → only vertical second
8 Extending the Path Timers

TikZ Library \texttt{ext.paths.timer}
\begin{verbatim}
\usetikzlibrary{ext.paths.timer} % \LaTeX\ and plain \TeX\n\usetikzlibrary{ext.paths.timer} % Con\TeXt
\end{verbatim}

This library adds timers to the path specifications \texttt{rectangle}, \texttt{parabola}, \texttt{sin} and \texttt{cos}.

Q \& A: \[5, 4\] \& \[22, 28\]

In TikZ, the path specification \texttt{rectangle}, \texttt{parabola}, \texttt{sin} and \texttt{cos} do not provide their own timer, i.e. a node placing algorithm that is dependent on the actual path. For \texttt{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.

8.1 Rectangle

For the \texttt{rectangle} path operator, the timer starts with \texttt{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.

\begin{verbatim}
\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}
\end{verbatim}

8.2 Parabola

For the \texttt{parabola} path operator the timer is similar to the \texttt{.. controls ..} operator.
The position 0.5 will lie at the bend.

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

8.3 Sine/Cosine

The \texttt{sin} and \texttt{cos} path operators also allow placing of nodes along their paths.
9 Using Images as a Pattern

TikZ Library `ext.patterns.images`

\usetikzlibrary{ext.patterns.images} \% L\TeX{} and \LaTeX{}
\usetikzlibrary{ext.patterns.images} \% Con\TeX{}

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

Q & A: [10] & [27]

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”.

```latex
\pgfsetupimageaspattern[\{options\}]{\{name\}]{\{image\}}
```

/tikz/image as pattern=(options) \hfill (default {})

/tikz/image as pattern/name=(name) \hfill (no default)

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

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

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

/tikz/image as pattern/options=(style) \hfill (style, no default)

 Appends style /tikz/image as pattern/option.
10 Positioning Plus

\textbf{TikZ Library} \texttt{ext.positioning-plus}

\begin{verbatim}
\usetikzlibrary{ext.positioning-plus} % \LaTeX and \texttt{plain \LaTeX}
\usetikzlibrary[ext.positioning-plus] % \texttt{Con\TeXt}
\end{verbatim}

With the help of the positioning and the \texttt{fit} library this extends the placement of nodes.

10.1 Useful corner anchors

The anchors \texttt{corner north east}, \texttt{corner north west}, \texttt{corner south west} and \texttt{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.

\begin{verbatim}
\tikzcorneraboveleft=⟨specification⟩
\end{verbatim}  \hspace{1cm} (style, default 0pt)

Similar as \texttt{/tikz/above left} of the TikZ library positioning but uses the \texttt{corner north west} anchor.

\begin{verbatim}
\tikzcornernbelowleft=⟨specification⟩
\end{verbatim}  \hspace{1cm} (style, default 0pt)

Similar as \texttt{/tikz/below left} of the TikZ library positioning but uses the \texttt{corner south west} anchor.

\begin{verbatim}
\tikzcorneraboveright=⟨specification⟩
\end{verbatim}  \hspace{1cm} (style, default 0pt)

Similar as \texttt{/tikz/above right} of the TikZ library positioning but uses the \texttt{corner north east} anchor.

\begin{verbatim}
\tikzcornernbellowright=⟨specification⟩
\end{verbatim}  \hspace{1cm} (style, default 0pt)

Similar as \texttt{/tikz/below right} of the TikZ library positioning but uses the \texttt{corner south east} anchor.
10.2 Useful placement keys for vertical and horizontal alignment

`tikz/north left=(specification)`

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

This is basically the same as `left=of reference.north west, anchor=north east`. 
/tikz/north right=(specification)
Like \texttt{/tikz/right} but aligns the nodes at their north border.
This is basically the same as \texttt{left=of reference.north east, anchor=north west}.

/tikz/south left=(specification)
Like \texttt{/tikz/left} but aligns the nodes at their south border.
This is basically the same as \texttt{left=of reference.south west, anchor=south east}.

/tikz/south right=(specification)
Like \texttt{/tikz/right} but aligns the nodes at their south border.
This is basically the same as \texttt{left=of reference.south east, anchor=south west}.

/tikz/west above=(specification)
Like \texttt{/tikz/above} but aligns the nodes at their west border.
This is basically the same as \texttt{left=of reference.north west, anchor=south west}.

/tikz/west below=(specification)
Like \texttt{/tikz/below} but aligns the nodes at their west border.
This is basically the same as \texttt{left=of reference.south west, anchor=north west}.

/tikz/east above=(specification)
Like \texttt{/tikz/above} but aligns the nodes at their east border.
This is basically the same as \texttt{left=of reference.north east, anchor=south east}.

/tikz/east below=(specification)
Like \texttt{/tikz/below} but aligns the nodes at their east border.
This is basically the same as \texttt{left=of reference.south east, anchor=north east}.
The same exist for the recently introduces corner anchors, too.

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

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

A
BCD

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.

This functionality is also available without the placement:
/tikz/fit bounding box=(list of coordinates)  
Creates a rectangular node with the name fit bounding box that encompasses the (list of coordinates).  

/tikz/span vertical=(list of coordinates)  
Creates a rectangular node with the name fit bounding box that encompasses the (list of coordinates) and sets the /pgfminimum height to the height of this bounding box.  

/tikz/span horizontal=(list of coordinates)  
Creates a rectangular node with the name fit bounding box that encompasses the (list of coordinates) and sets the /pgfminimum width to the width of this bounding box.  

/tikz/span=(list of coordinates)  
Is a combination of /tikz/span vertical and /tikz/span horizontal.  

As you maybe noticed in the example above, the (specification) also allows a prefix delimited by : which the node distance will be multiplied to with for the placement.\footnote{This is probably more useful when /tikz/on grid is used.}
11 Arrows through Three Points

TikZ Library \texttt{ext.topaths.arcthrough}

\begin{tikzpicture}
\coordinate [label=\textit{above right:$A$}] (A) at ( 3, 1);
\coordinate [label=\textit{above:$B$}] (B) at ( 1, 2);
\coordinate [label=\textit{below left:$C$}] (C) at (-2,-2);
\draw[ultra thick, draw=\textcolor{green}, fill=\textcolor{green!50}]
\path (B) to\textit{arc through}={clockwise,(A)} (C)
\draw[ultra thick, draw=\textcolor{blue}, fill=\textcolor{blue!50}]
\path (B) to\textit{arc through}=\textit{A} (C)
\foreach \p in {A,B,C, arc through center} \draw[red] (\p) circle [radius=2pt];
\end{tikzpicture}

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

\begin{itemize}
\item \texttt{/tikz/arc through/through=(coordinate)} (no default, initially \texttt{(0,0)})
  The coordinate on the circle that defines – together with the starting and target point – a circle.
\item \texttt{/tikz/arc through/center suffix=(suffix)} (no default, initially \texttt{)}
  The \texttt{arc through} will define a coordinate named \texttt{arc through center\texttt{(suffix)}} so that it can be referenced later.
\item \texttt{/tikz/arc through/clockwise} (no value)
  The resulting arc will go clockwise from the starting point to the target point. This will not necessarily go through the through point.
\item \texttt{/tikz/arc through/counter clockwise} (no value)
  The resulting arc will go counter clockwise from the starting point to the target point. This will not necessarily go through the through point.
\item \texttt{/tikz/arc through=\textit{key-value}} (no default)
  This key should be used with \texttt{to} or \texttt{edge}. A parameter other than \texttt{center suffix}, \texttt{clockwise} or \texttt{counter clockwise} will be assumed to be the through coordinate.
\end{itemize}
12 Mirror, Mirror on the Wall

TikZ Library \texttt{ext.transformations.mirror}

\begin{verbatim}
\usetikzlibrary{ext.transformations.mirror} % \LaTeX{} and plain \TeX{}
\usetikzlibrary{ext.transformations.mirror} % Con\TeX{}t
\end{verbatim}

This library adds more transformations to TikZ.

As explained in section 13, 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.

12.1 Using the reflection matrix

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

/tikz/xmirror=(value or coordinate)  \hspace*{\textwidth}

Sets up a transformation that mirrors along a horizontal line that goes through point \texttt{(value,0)} or \texttt{(coordinate)}.

\begin{verbatim}
\usetikzlibrary{ext.transformations.mirror}
\begin{tikzpicture}
  \draw[help lines] (-0.25, -.25) grid (3.25, 1.25);
  \draw[->] (0,0) .. controls (.5,1) .. (1,1);
  \draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);
  \draw[xmirror=(m),-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
\end{verbatim}
Sets up a transformation that mirrors along a vertical line that goes through point \((0, \langle \text{value} \rangle)\) or \(\langle \text{coordinate} \rangle\).

Similar to /tikz/xmirror, this however uses the \texttt{xyz} coordinate system instead of the \texttt{canvas} system.

Similar to /tikz/ymirror, this however uses the \texttt{xyz} coordinate system instead of the \texttt{canvas} system.

Sets up a transformation that mirrors along a line that goes through \(\langle \text{point A} \rangle\) and \(\langle \text{point B} \rangle\). When only \(\langle \text{point A} \rangle\) is given that line goes through \(\langle \text{point A} \rangle\) and the origin.

### 12.2 Using built-in transformations
\texttt{/tikz/xMirror}=(\textit{value or coordinate}) 
Sets up a transformation that mirrors along a horizontal line that goes through point \((\textit{value},0)\) or \((\textit{coordinate})\).

\begin{tikzpicture}
\draw[help lines] (-0.25, -.25) grid (3.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);
\draw[xMirror=(m),-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}

\texttt{/tikz/yMirror}=(\textit{value or coordinate}) 
Sets up a transformation that mirrors along a vertical line that goes through point \((0,\textit{value})\) or \((\textit{coordinate})\).

\texttt{/tikz/Mirror x=}\textit{(coordinate)}
Similar to \texttt{/tikz/xMirror}, this however uses the \textit{xyz} coordinate system instead of the \textit{canvas} system.

\begin{tikzpicture}[x=.5cm, y=(45:1cm)]
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);
\draw[xMirror=(m),-latex, red, dotted] (0,0) .. controls (.5,1) .. (1,1);
\draw[Mirror x=(m),-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}

\texttt{/tikz/Mirror y=}\textit{(coordinate)}
Similar to \texttt{/tikz/yMirror}, this however uses the \textit{xyz} coordinate system instead of the \textit{canvas} system.

\texttt{/tikz/Mirror=}\textit{(point A)}--\textit{(point B)}
Sets up a transformation that mirrors along a line that goes through \(\textit{point A}\) and \(\textit{point B}\).
When only \(\textit{point A}\) is given that line goes through \(\textit{point A}\) and the origin.
Part III

PGF Libraries

These libraries (should) work with both PGF and TikZ.
13 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 \texttt{pgf}.

Two approaches to mirror transformation exist:

1. Using the reflection matrix (see left column).
   - This depends on \texttt{\pgfpointnormalised} which involves the sine and the cosine functions of \texttt{pgfmath}.

2. Using built-in transformations (see right column).
   - This depends on \texttt{\pgfmathanglebetweenpoints} which involves the arctangent (\texttt{atan2}) function of \texttt{pgfmath}.

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

### 13.1 Using the reflection matrix

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

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

\texttt{\pgftransformxmirror{⟨value⟩}}

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

### 13.2 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 \texttt{m} in \texttt{mirror} is capitalized.

\texttt{\pgftransformxMirror{(value)}}

Sets up a transformation that mirrors along a vertical line that goes through point \((⟨value⟩, 0)\).
\pgftransformymirror{(value)}
Sets up a transformation that mirrors along a horizontal line that goes through point (0, (value)).

\pgftransformmirror{(point A)}{(point B)}
Sets up a transformation that mirrors along the line that goes through (point A) and (point B).

\pgftransformyMirror{(value)}
Sets up a transformation that mirrors along a horizontal line that goes through point (0, (value)).

\pgftransformMirror{(point A)}{(point B)}
Sets up a transformation that mirrors along the line that goes through (point A) and (point B).

\pgfqtransformmirror{(point A)}
Sets up a transformation that mirrors along the line that goes through the origin and (point A).
14 Shape: Circle Arrow

TikZ Library \texttt{ext.shapes.circlearrow}

\usepgflibrary{ext.shapes.circlearrow} % \LaTeX{} and plain \TeX{} and pure pgf
\usepgflibrary{ext.shapes.circlearrow} % Con\TeX{} and pure pgf
\usetikzlibrary{ext.shapes.circlearrow} % \LaTeX{} and plain \TeX{} when using TikZ
\usetikzlibrary{ext.shapes.circlearrow} % Con\TeX{} when using TikZ

A circular shape named \texttt{circle arrow} that has an arc as its background path that can have an arrow tip.

Q & A: [14] & [23]

Shape \texttt{circle arrow}

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

\texttt{/pgf/circle arrow start angle=(start angle)} \hfill (no default, initially \{\})

Sets the start angle.

\texttt{/pgf/circle arrow end angle=(end angle)} \hfill (no default, initially \{\})

Sets the end angle.

\texttt{/pgf/circle arrow delta angle=(delta angle)} \hfill (no default, initially \{\})

Sets the delta angle.

\texttt{/pgf/circle arrow arrows=(start arrow tip specification)-(end arrow tip specification)} \hfill (no default, initially -)

The specification will be forwarded to \texttt{\pgfsetarrows}.

A few handful styles are pre-defined.

\texttt{/pgf/circle arrow turn left north} \hfill (no value)

Sets \texttt{circle arrow start angle = 100}, \texttt{circle arrow delta angle = 340} and \texttt{circle arrow arrows = ->}.

\texttt{/pgf/circle arrow turn left east} \hfill (no value)

As above but \texttt{circle arrow start angle = 10}.

\texttt{/pgf/circle arrow turn left west} \hfill (no value)

As above but \texttt{circle arrow start angle = 280}.

\texttt{/pgf/circle arrow turn left south} \hfill (no value)

As above but \texttt{circle arrow start angle = 190}.
/pgf/circle arrow turn right north
Sets circle arrow start angle = 100, circle arrow delta angle = 340 and circle arrow arrows = <-. (no value)

/pgf/circle arrow turn right east
As above but circle arrow start angle = 10. (no value)

/pgf/circle arrow turn right west
As above but circle arrow start angle = 280. (no value)

/pgf/circle arrow turn right south
As above but circle arrow start angle = 190. (no value)
\usetikzlibrary{ext.shapes.circlearrow}
\begin{tikzpicture}
\Huge
\node[name=s, shape=circle arrow, circle arrow turn left west, shape example]
{(Circle Arrow)\vrule width 1pt height 2cm};
\foreach \anchor/\placement in 
{north west/above left, north/above, north east/above right, west/left, center/above, east/right, mid west/right, mid/above, mid east/left, base west/left, base/below, base east/right, south west/below left, south/below, south east/below right, text/left, 10/right, 130/above}
\draw[shift=(s.\anchor)] plot[mark=x] coordinates{(0,0)}
node[\placement] \scriptsize{(s.\anchor)};
\end{tikzpicture}
15 Shape: Circle Cross Split

TikZ Library ext.shapes.circlecrosssplit

\usepgflibrary{ext.shapes.circlecrosssplit} % \LaTeX{} and plain \TeX{} and pure pgf
\usepgflibrary{ext.shapes.circlecrosssplit} % Con\TeX{} and pure pgf
\usetikzlibrary{ext.shapes.circlecrosssplit} % \LaTeX{} and plain \TeX{} when using TikZ
\usetikzlibrary{ext.shapes.circlecrosssplit} % Con\TeX{} when using TikZ

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


Shape circle cross split

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

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

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

/pgf/circle cross split uses custom fill={boolean} (default true)

This enables the use of a custom fill for each of the node parts (including the area covered by the inner sep). The background path for the shape should not be filled (e.g., in TikZ, the fill option for the node must be implicitly or explicitly set to none). Internally, this key sets the \LaTeX{-}if\pgfcirclecrosssplitcustomfill appropriately.
text
two
three
text two
three four
(s.north west)
(s.north)
(s.north east)
(s.west) (s.center) (s.east)
(s.mid west) (s.mid) ... mid east)
(s.south west)
(s.south)
(s.south east)
(s.lower base west)
(s.lower mid west)
(s.lower mid)
(s.lower mid east)
(s.lower base west)
(s.lower base east)
(s.lower base)
(s.south west)
(s.south)
(s.south east)
(s.south west)
\usepgflibrary{ext.shapes.circlecrosssplit}
\begin{tikzpicture}
\Huge
\node [name=s, shape=circle cross split, shape example, inner xsep=1.5cm, fill=none, circle cross split part fill={green,blue,red,yellow!90!black}]
  \nodepart{text}
  text
  \nodepart{two}
  two
  \nodepart{three}
  three
  \nodepart{four}
  four;
\foreach \anchor/\placement in
  {north west/above left, north/above, north east/above right, west/left, center/left, east/right, mid west/right, mid/left, mid east/left, base west/left, base/left, base east/right, lower base west/left, lower base/below, lower base east/right, lower mid west/left, lower mid/above, lower mid east/right, south west/below left, south/below, south east/below right, text/below, 10/right, 130/above, two/left, three/left, four/left}
  \draw[shift={s.\anchor}] plot[mark=x] coordinates{(0,0)}
  \node[\placement] \scriptsize \texttt{(s.\anchor)};
\end{tikzpicture}
16 Shape: Heatmark

TikZ Library `ext.shapes.heatmark`

\usepgflibrary{ext.shapes.heatmark} % \LaTeX{} and plain \TeX{} and pure pgf
\usepgflibrary[ext.shapes.heatmark] % Con\TeX{} and pure pgf
\usetikzlibrary{ext.shapes.heatmark} % \LaTeX{} and plain \TeX{} when using Ti\textsc{k}Z
\usetikzlibrary[ext.shapes.heatmark] % Con\TeX{} when using Ti\textsc{k}Z

A circular shape that has customizable rings around it.


Shape heatmark

\texttt{/pgf/heatmark arcs\texttt{=}(arcs num)} \hfill (no default, initially 3)

Sets the number of arc around the circle to \texttt{(arcs num)}.

\texttt{/pgf/heatmark arc width\texttt{=}(arc width)} \hfill (no default, initially 4pt)

Sets the width of the rings around the circle to \texttt{(arc width)}.

\texttt{/pgf/heatmark arc sep\texttt{=}(sep length)} \hfill (no default, initially 1pt)

Sets the whitespace between the rings to \texttt{(sep length)}.

\texttt{/pgf/heatmark arc rings\texttt{=}(rings num)} \hfill (no default, initially 3)

Sets the number of rings around the circle to \texttt{(rings num)}.

\texttt{/pgf/heatmark arc sep angle\texttt{=}(sep angle)} \hfill (no default, initially 20)

Sets the whitespace angle between the arcs in one ring to \texttt{(sep angle)}.

\texttt{/pgf/heatmark inner opacity\texttt{=}(inner opacity)} \hfill (no default, initially 0.8)

Sets the opacity of the inner ring to \texttt{(inner opacity)}.

\texttt{/pgf/heatmark outer opacity\texttt{=}(low opacity)} \hfill (no default, initially 0.2)

Sets the opacity of the outer ring to \texttt{(outer opacity)}.

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

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

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

\texttt{/pgf/heatmark ring \texttt{(ring number)}} \hfill (style, no value)
The `pgf` shape is setup in a way that even TikZ styles can be used with a little bit work:

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

It is best to use this shape with no actual border (\texttt{draw = none}) and the \texttt{outer sep} set to zero.
17 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] % Con\TeX{} and pure pgf
\usetikzlibrary{ext.shapes.rectangleroundedcorners} % \LaTeX{} and plain \TeX{} when using TikZ
\usetikzlibrary[ext.shapes.rectangleroundedcorners] % Con\TeX{} when using TikZ

A rectangle with rounded corners.

Shape `rectangle with rounded corners`

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

\begin{itemize}
  \item `/pgf/rectangle with rounded corners north west radius=(dimen)`
    Sets the north west radius to `(dimen).` (no default, initially `.5\pgflinewidth`)
  \item `/pgf/rectangle with rounded corners north east radius=(dimen)`
    Sets the north east radius to `(dimen).` (no default, initially `.5\pgflinewidth`)
  \item `/pgf/rectangle with rounded corners south west radius=(dimen)`
    Sets the south west radius to `(dimen).` (no default, initially `.5\pgflinewidth`)
  \item `/pgf/rectangle with rounded corners south east radius=(dimen)`
    Sets the south east radius to `(dimen).` (no default, initially `.5\pgflinewidth`)
  \item `/pgf/rectangle with rounded corners radius=(dimen)`
    Sets all radii to `(dimen).` (no default)
\end{itemize}
\usepgflibrary{ext.shapes.rectangleroundedcorners}
\begin{tikzpicture}
\Huge
\node[name=s, shape=rectangle with rounded corners, shape example, rectangle with rounded corners north west radius=10pt, rectangle with rounded corners north east radius=20pt, rectangle with rounded corners south west radius=30pt, rectangle with rounded corners south east radius=40pt] {Rectangle with rounded corners\vrule width 1pt height 2cm};
\foreach \anchor/\placement in {north west/above left, north/above, north east/above right, west/left, center/above, east/right, mid west/right, mid/above, mid east/left, base west/left, base/below, base east/right, south west/below left, south/below, south east/below right, text/below, 10/right, 130/above, north west center/below right, north east center/left, south west center/above right, south east center/left, below north west/left, above south west/left, above south east/right, below north east/right, right north west/above, right south west/below, left south east/below, left north east/above}
\draw[shift={s.\anchor}] plot[mark=x] coordinates{(0,0)} node[\placement] \{\scriptsize\texttt{(s.\anchor)}\};
\end{tikzpicture}
18 Shape: Superellipse

TikZ Library `ext.shapes.superellipse`

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

Shape in the form of a “superellipse”.

Q & A: [29] & [17]

**Shape superellipse**

This shape is defined by formula

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

and will be plotted by

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

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

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

/pgf/superellipse x exponent=(x exponent) \hfill (no default, initially 2.5)

This sets $m$.

/pgf/superellipse y exponent=(y exponent) \hfill (no default, initially 2.5)

This sets $n$.

/pgf/superellipse step=(step) \hfill (no default, initially 5)

This specifies the step of the underlying plot handler. The smaller \textit{(step)} is, the slower computation will be. Sensible values for \textit{(step)} are integer dividers of 90, i.e. 2, 3, 5, 6, 9, 10, 15, 18, 30 and 45.

/pgf/superellipse exponent=(exponent) \hfill (no default)

Sets both superellipse \textit{x exponent} and superellipse \textit{y exponent} to \textit{(exponent)}. 

43
Notes on Implementation  For implementing this shape, additional mathematical functions were declared.

\begin{verbatim}
\text{superellipsex}(t, 2/m, r_x) \pgfmathsuperellipsex{t}{2/m}{r_x}
\end{verbatim}

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

\begin{equation}
x = r_x \cos^{2/m} t
\end{equation}

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

\begin{verbatim}
\text{superellipsey}(t, 2/n, r_y) \pgfmathsuperellipsey{t}{2/n}{r_y}
\end{verbatim}

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

\begin{equation}
y = r_y \cos^{2/n} t
\end{equation}

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

Both \texttt{PGFmath} functions can be used at once with the following macro.

\begin{verbatim}
\pgfmathsuperellipseXY{(i)}{(2/m)}{(2/n)}{(a)}{(b)}
\end{verbatim}

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

\begin{align*}
x &= a \cos^{2/m} t \\
y &= b \cos^{2/n} t
\end{align*}

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

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

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

\begin{verbatim}
\pgfutil@prefix@macrotomacro{(macro 1)}{(macro 2)}
\end{verbatim}

Adds the once-expansion of (\texttt{macro 2}) in front of (\texttt{macro 1}).
\usetikzlibrary{ext.shapes.superellipse}
\begin{tikzpicture}
\node[name=s,shape=superellipse,shape example] (Superellipse) {Superellipse\rule width lpt 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}
\foreach \xe/\ye[count=i] in {.5/.5, 1/1, 2/2, 3/3, .5/5} 
{\node[draw, superellipse, superellipse x exponent=\xe, superellipse y exponent=\ye] at (1.5*i,0) {}};
\end{tikzpicture}
Part IV
Utilities

\usetikzlibrary{ext.misc}
\begin{tikzpicture}
define function={bigR(n)=smallR+.05*n;},
define constant={smallR=1; segments=20;},
full arc=segments
\foreach \n in [evaluate={\endRadius=bigR(\n+1);}, use int=0 to segments-1]
  \fill\draw[fill=gray!50]\(\\bigcirc\)\text{ arc }[radius=\endRadius, start angle=\n \cdot R, delta angle=1R] -- (\n R+1R:smallR)
  \text{ arc }[radius=smallR, end angle=\n \cdot R, delta angle=-1R] -- cycle;
\node {$\phi^2$};
\node at (north west:{sqrt 2 * bigR(segments/2)}) {$\{\Omega\}_{i=1}^n$};
\node at (-.5R: bigR segments) {$\partial \varphi$};
\end{tikzpicture}
19 Calendar: Weeknumbers and more conditionals

\usepackage{calendar-ext} % \LaTeX
\input{calendar-ext.tex} % plain \TeX
\usemodule{calendar-ext} % Con\TeX t

This package adds week numbers and more conditionals to the PGF package \texttt{pgfcalendar}. (Despite the code example above, this package is not set up to work with Con\TeX t.)

Q & A: \[6, 7, 9\] & \[16, 26, 19\]

19.1 Extensions

The following tests are added.

- \texttt{Jan} This test is passed by all dates that are in the month of January.
- \texttt{Feb} as above.
- \texttt{Mar} as above.
- \texttt{Apr} as above.
- \texttt{May} as above.
- \texttt{Jun} as above.
- \texttt{Jul} as above.
- \texttt{Aug} as above.
- \texttt{Sep} as above.
- \texttt{Oct} as above.
- \texttt{Nov} as above.
- \texttt{Dec} as above.
- \texttt{leap year=(year)} This test checks whether the given year is a leap year. If \langle year\rangle is omitted, it checks the year of the current date.
- \texttt{and\{\langle tests\rangle\}} This test passes when all \langle tests\rangle pass.
- \texttt{not\{\langle tests\rangle\}} This test passes when \langle tests\rangle do not pass.

- \texttt{week of month\{\langle num\rangle\}} This test passes when the date is in \langle num\rangle th week of the month. The first week of the month start at day 1 and ends with day 7.
- \texttt{week of month\'\{\langle num\rangle\}} As above but counts from the last day of the month. For a month with 31 days, this means the “1st” week starts at day 25 and ends with day 31.
- \texttt{calendar week of month\{\langle num\rangle\}} This test passes when the date is in \langle num\rangle th calendar week of the month. The first week starts at the first day of the month and ends at the next Sunday.
- \texttt{calendar week of month\'\{\langle num\rangle\}} As above but counts from the last day of the month.
- \texttt{yesterday\{\langle tests\rangle\}} This test passes when the previous day passes \langle tests\rangle.
- \texttt{week\{\langle num\rangle\}} This test passes when the current week of the year equals \langle num\rangle.

The shorthands for \texttt{d-} and \texttt{m-} are slightly changed so that they are expandable. This makes it possible to use these shorthands inside of PGF\texttt{math}. The shorthands for the week (see section 19.2) are added. These are
• \( n \) (shortest numerical representation),
• \( n= \) (shortest but added horizontal space) and
• \( n0 \) (leading zero when below 10).

19.2 Week numbering (ISO 8601)

\( \texttt{\textbackslash pgfcalendar\textbackslash julianyeartoweek\{(Julian day\}\{(year\}\{(week counter\}\}} \)

This command calculates the week for the \( \text{\{Julian day\}} \) of \( \text{\{year\}} \). The \( \text{\{week counter\}} \) must be a \( \text{\TeX} \) counter.

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

Inside of \( \text{\texttt{pgfcalendar}} \) the command \( \text{\texttt{pgfcalendar\textbackslash currentweek}} \) will be available.

\( \text{\texttt{\pgfcalendarcurrentweek}} \)

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

Inside of \( \text{\texttt{ifdate}} \) the command \( \text{\texttt{pgfcalendar\textbackslash ifdate\textbackslash week}} \) will be available.

\( \text{\texttt{\pgfcalendarifdate\textbackslash week}} \)

This command returns the week number (always two digits).
20 And a little bit more

TikZ Library `ext.misc`

```latex
\usetikzlibrary{ext.misc} % \LaTeX and plain \TeX
\usetikzlibrary{ext.misc} % Con\TeX
```

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

Q & A: [13, 1] & [15, 21]

20.1 PGFmath

20.1.1 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.

```
\tikz[full arc](num) % (default)
```

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

The given value `(num)` is evaluated when the key is used and doesn’t change when `(num)` contains variables that change.

The R operator can then be used.

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

Multiplies `x` with `\frac{360}{(num)}`.

20.1.2 Functions

```
strrepeat("Text", x)
```

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

```
foofoofoofoofoo
```

\pgfmathstrrepeat{"Text"}{x}

```
isEmpty("Text")
```

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

```
0 and 1
```

```
\pgfmathisEmpty{"Text"}
```

```
strcat("Text A", "Text B", ...)
```

```
\pgfmathstrcat{"Text A"}{"Text B"}{...}
```

Returns the concatenation of all given parameters.

```
blue!21!green
```

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

```
isEmpty("Text")
```

\pgfmathisEmpty{"Text"}

atanXY(x, y)

\pgfmathresult
\pgfmathatanXY{x}\{y\}

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

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

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

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

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

\subsection*{20.1.3 Functions: using coordinates}

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

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

\subsection{20.2.1 Conditionals}

\texttt{/utils/if=(\textit{cond})(true)(false)} \hspace{1cm} (no default)

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

As a side effect on how \texttt{PGFkeys} parses argument, the \texttt{\langle false \rangle} argument is actually optional.

The following keys use \texttt{\TeX} macros \texttt{\if, \ifx, \ifnum} and \texttt{\ifdim} for faster executions.

\texttt{/utils/TeX/if=(\textit{token A})(\textit{token B})(true)(false)} \hspace{1cm} (no default)

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

As a side effect on how \texttt{PGFkeys} parses argument, the \texttt{\langle false \rangle} argument is actually optional.

\texttt{/utils/TeX/ifx=(\textit{token A})(\textit{token B})(true)(false)} \hspace{1cm} (no default)

As above.

\texttt{/utils/TeX/ifnum=(\textit{num cond})(true)} \hspace{1cm} \texttt{opt(false)} \hspace{1cm} (no default)

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

As a side effect on how \texttt{PGFkeys} parses argument, the \texttt{\langle false \rangle} argument is actually optional.

\texttt{/utils/TeX/ifdim=(\textit{dim cond})(true)(false)} \hspace{1cm} (no default)

As above.

\texttt{/utils/TeX/ifempty=(\textit{Text})(true)(false)} \hspace{1cm} (no default)

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

\subsection{20.2.2 Handlers}

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

\textbf{Key handler} (key)/.\texttt{pgfmath=(eval)}

This handler evaluates \texttt{\langle eval \rangle} before it is handed to the key.

\textbf{Key handler} (key)/.\texttt{pgfmath int=(eval)}

As above but truncates the result.

\textbf{Key handler} (key)/.\texttt{pgfmath strcat=(eval)}

As above but uses the \texttt{strcat} function.

In the example below, one could have used the \texttt{/pgf/foreach/evaluate} key from \texttt{\foreach}.

\begin{verbatim}
\usepackage{tikz}
\usetikzlibrary{misc}
\tikz\foreach \i in {0,10,...,100}
\draw[line width=+.2cm, color/.pgfmath strcat="red!",sqrt(\i)*10,"!blue"]
[0,\i/50] -- +(right:3);
\end{verbatim}

\textbf{Key handler} (key)/.\texttt{List=(\textit{e1}, \textit{e2}, \ldots, \textit{en})}

This handler evaluates the given list with \texttt{\foreach} and concatenates the element and the result is then given to the used key.
20.3 \texttt{PGFfor}

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

\texttt{/pgf/foreach/use int=(start)to(end)step(delta)} \hspace{1cm} \texttt{(no default)}

The values \texttt{(start)}, \texttt{(end)} and \texttt{(delta)} are evaluated by \texttt{PGFmath} at initialization. The part \texttt{step (delta)} is optional (\texttt{(delta) = 1}).

\texttt{/pgf/foreach/use float=(start)to(end)step(delta)} \hspace{1cm} \texttt{(no default)}

Same as above, however the results are not truncated.
Part V
Changelog, Index & References

Changelog

Version 0.3.1/0.4

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

Version 0.3

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

Version 0.2

• Added TiKZ library `ext.positioning-plus`.
• Added TiKZ library `ext.node-families`.

Version 0.1

• Added TiKZ library `ext.calendar-plus`.
• Added TiKZ library `ext.misc`.
• Added TiKZ library `ext.paths.arcto`.
• Added TiKZ library `ext.paths.ortho`.
• Added TiKZ library `ext.paths.timer`.
• Added TiKZ library `ext.patterns.images`.
• Added TiKZ library `ext.topaths.arcthrough`.
• 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 \texttt{PGF/TikZ}.

\begin{itemize}
\item path operation, 12
\item path operation, 12
\item path operation, 12
\item path operation, 12
\item path operation, 12
\item above key, 22
\item above left key, 20
\item above right key, 20
\item and date test, 47
\item angle between math function, 50
\item Apr date test, 47
\item arc path operation, 32
\item arc through key, 25
\item arc to path operation, 10
\item \texttt{atan2} math function, 30, 50
\item \texttt{atanXY} math function, 49
\item \texttt{atanYX} math function, 50
\item Aug date test, 47
\item below key, 22
\item below left key, 20
\item below right key, 20
\item \texttt{math function}, 50
\item \texttt{calendar library}, 6
\item \texttt{calendar week of month} date test, 47
\item \texttt{calendar week of month'} date test, 47
\item \texttt{calendar-ext} package, 47
\item center suffix key, 25
\item circle shape, 7
\item circle arrow shape, 32
\item circle arrow arrows key, 32
\item circle arrow delta angle key, 32
\item circle arrow end angle key, 32
\item circle arrow start angle key, 32
\item circle arrow turn left east key, 32
\item circle arrow turn left north key, 32
\item circle arrow turn left south key, 32
\item circle arrow turn left west key, 32
\item circle arrow turn right east key, 33
\item circle arrow turn right north key, 33
\item circle arrow turn right south key, 33
\item circle arrow turn right west key, 33
\item circle cross split shape, 35
\item circle cross split part fill key, 35
\item circle cross split uses custom fill key, 35
\item clockwise key, 10, 25
\item corner above left key, 20
\item corner above right key, 20
\item corner below left key, 20
\item corner below right key, 20
\item corner east above key, 23
\item corner east below key, 23
\item corner north left key, 23
\item corner north right key, 23
\item corner south left key, 23
\item corner south right key, 23
\item corner west above key, 23
\item corner west below key, 23
\item cos math function, 30
\item cos path operation, 18
\item counter clockwise key, 11, 25
\item Date tests
\item and, 47
\item Apr, 47
\item Aug, 47
\item calendar week of month, 47
\item calendar week of month', 47
\item Dec, 47
\item Feb, 47
\item Jan, 47
\end{itemize}
Key handlers

.List, \texttt{51}
\texttt{.pgfmath}, \texttt{51}
\texttt{.pgfmath int}, \texttt{51}
\texttt{.pgfmath strcat}, \texttt{51}

large key, \texttt{11}
lastdayinmonthofyear math function, \texttt{6}
leap year date test, \texttt{47}
left key, \texttt{21, 22}

Libraries

calc, \texttt{50}
calendar, \texttt{6}
ext.calendar-plus, \texttt{6}
ext.misc, \texttt{49}
ext.node-families, \texttt{7}
ext.node-families.shapes.geometric, \texttt{9}
ext.paths.arcto, \texttt{10}
ext.paths.ortho, \texttt{12}
ext.paths.timer, \texttt{17}
ext.patterns.images, \texttt{19}
ext.positioning-plus, \texttt{20}
ext.shapes.circlearrow, \texttt{32}
ext.shapes.circlexcrosssplit, \texttt{35}
ext.shapes.heatmark, \texttt{38}
ext.shapes.rectangleroundedcorners, \texttt{41}
ext.shapes.superellipse, \texttt{43}
ext.topaths.arcthrough, \texttt{25}
ext.transformations.mirror, \texttt{26, 30}
external, \texttt{7}
f, \texttt{20}
fit, \texttt{20}
positioning, \texttt{20}
shapes.geometric, \texttt{9}

.List handler, \texttt{51}
lr distance key, \texttt{14}

Math functions

anglebetween, \texttt{50}
atan2, \texttt{30, 50}
atanXY, \texttt{49}
atanXY, \texttt{50}
cos, \texttt{30}
distancebetween, \texttt{50}
isEmpty, \texttt{49}
isInString, \texttt{49}
lastdayinmonthofyear, \texttt{6}
qanglebetween, \texttt{50}
qdistancebetween, \texttt{50}
sin, \texttt{30}
strcat, \texttt{49}
strrepeat, \texttt{49}
superellipsex, \texttt{44}
superellipsey, \texttt{44}
weeksinmonthofyear, \texttt{6}

Math operators

\texttt{R, 49}

May date test, \texttt{47}
middle 0 to 1 key, \texttt{14}
minimum height key, \texttt{8}
minimum width key, \texttt{8}
Mirror key, \texttt{28}
mirror key, \texttt{27}
mirror x key, \texttt{28}
mirror x key, \texttt{27}
mirror y key, \texttt{28}
mirror y key, \texttt{27}
month code key, \texttt{6}
month text key, \texttt{6}
month xshift key, \texttt{6}
month yshift key, \texttt{6}

name key, \texttt{19}
north left key, \texttt{21}
north right key, \texttt{22}
not date test, \texttt{47}
Nov date test, \texttt{47}

Oct date test, \texttt{47}
on grid key, \texttt{24}
only horizontal first key, \texttt{15}
only horizontal second key, \texttt{15}
only vertical first key, 15
only vertical second key, 15
option key, 19
options key, 19

Packages and files
    calendar-ext, 47
parabola path operation, 17
Path operations
    |-|, 12
    -|-, 12
    -|, 12
    |-, 12
    arc, 32
    arc to, 10
cos, 18
dege, 25
parabola, 17
r-du, 14
r-lr, 14
r-r1, 14
rectangle, 17
sin, 18
to, 25

\pgf/
circle arrow arrows, 32
circle arrow delta angle, 32
circle arrow end angle, 32
circle arrow start angle, 32
circle arrow turn left east, 32
circle arrow turn left north, 32
circle arrow turn left south, 32
circle arrow turn left west, 32
circle arrow turn right east, 33
circle arrow turn right north, 33
circle arrow turn right south, 33
circle arrow turn right west, 33
circle cross split part fill, 35
circle cross split uses custom fill, 35
foreach/
use float, 52
use int, 52
heatmark arc (arc number), 39
heatmark arc rings, 38
heatmark arc sep, 38
heatmark arc sep angle, 38
heatmark arc width, 38
heatmark arcs, 38
heatmark inner opacity, 38
heatmark outer opacity, 38
heatmark ring (ring number), 38
heatmark ring (ring number) arc (arc number), 39
minimum height, 8
minimum width, 8
rectangle with rounded corners north east radius, 41
rectangle with rounded corners north west radius, 41
rectangle with rounded corners radius, 41
rectangle with rounded corners south east radius, 41
rectangle with rounded corners south west radius, 41
shape border rotate, 38
superellipse exponent, 43
superellipse step, 43
superellipse x exponent, 43
superellipse y exponent, 43
text, 19
\pgfcalendar, 48
\pgfcalendarcurrentweek, 48
\pgfcalendarifydateweek, 48
\pgfcalendarjulianyartoweek, 48
\pgfmath handler, 51
\pgfmath int handler, 51
\pgfmath strcat handler, 51
\pgfmathanglebetween, 50
\pgfmathanglebetweenpoints, 30
\pgfmathatanXY, 50
\pgfmathatanX, 50
\pgfmathdistancebetween, 50
\pgfmathisInString, 49
\pgfmaththisEmpty, 49
\pgfmaththisInString, 49
\pgfmathlastdayinmonthofyear, 6
\pgfmathanglebetween, 50
text width align, 8
width, 8, 9
north left, 21
north right, 22
on grid, 24
only horizontal first, 15
only horizontal second, 15
only vertical first, 15
only vertical second, 15
ortho/
  distance, 12
du distance, 14
from center, 13
install shortcuts, 16
lr distance, 14
middle 0 to 1, 14
ratio, 12
rl distance, 14
spacing, 13
ud distance, 14
udlr distance, 14
pos, 13, 17
right, 22
south left, 22
south right, 22
span, 24
span horizontal, 24
span vertical, 24
to path, 15
vertical horizontal, 15
vertical horizontal vertical, 15
week code, 6
week date test, 47
week label left key, 6
week label right key, 6
week of month date test, 47
week of month' date test, 47
week text key, 6
weeks in month of year math function, 6
west above key, 22
west below key, 22
width key, 8, 9
x radius key, 11
x radius key, 11
xMirror key, 28
xmirror key, 26
y radius key, 11
yMirror key, 28
yMirror key, 27
ymirror, 27
to path operation, 25
to path key, 15
ud distance key, 14
udlr distance key, 14
use float key, 52
use int key, 52
/track/
if, 51
TeX/
if, 51
ifdim, 51
ifempty, 51
ifnum, 51
ifeq, 51
vertical horizontal key, 15
vertical horizontal vertical key, 15
yesterday date test, 47
yMirror key, 28
ymirror key, 27
References


