The TikZ-Extensions Package
Manual for version 0.4.2
https://github.com/Qrrbrbirlbel/tikz-extensions

Qrrbrbirlbel
October 30, 2022

Contents

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

II TikZ Libraries 5
  4 Calendar 6
    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
18 Shape: Rectangle with Rounded Corners
19 Shape: Superellipse
20 Shape: Uncentered Rectangle

IV Utilities
21 Calendar: Weeknumbers and more conditionals
  21.1 Extensions
  21.2 Week numbering (ISO 8601)
22 And a little bit more
  22.1 pgfmath
     22.1.1 Postfix operator \( R \)
     22.1.2 Functions
     22.1.3 Functions: using coordinates
  22.2 pgf\for
  22.3 pgf\keys
     22.3.1 Conditionals
     22.3.2 Handlers

V Changelog, Index & References
Change log
Index
References
Part I

Introduction

1 Usage

This package is called tikz-ext, however, one can’t load it via \usepackage. Instead, this package consists of multiple 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 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.

---

1Except for calendar-ext of section 21.
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} % \ConTeXt{}

This library extends the TikZ library calendar.

Q & A: [6, 7, 3] & [18, 30, 29]

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

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

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

4.2 pgfmath functions

\pgfmathweeksinmonthofyear{first weekday, month, year}
\pgfmathweeksinmonthofyear{first weekday}{month}{year}

Returns the number of (partial) weeks in the month \textit{month} of year \textit{year} when this month begins on a \textit{first weekday}.

\pgfmathlastdayinmonthofyear{month, year}
\pgfmathlastdayinmonthofyear{month}{year}

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

4.3 Week numbering (ISO 8601)

The actual week number algorithm is implemented by the \pgfcalendar-ext package/module in section 21.2.

/tikz/week code=(code) \quad (no default)
\quad Works like /tikz/day code or /tikz/month code, only for weeks.

/tikz/week text=(text) \quad (no default)
\quad Works like /tikz/day text or /tikz/month text, only for weeks.

/tikz/every week \quad (style, no value)
\quad Works like /tikz/every day or /tikz/every month, only for weeks.

/tikz/week label left \quad (style, no value)
\quad Places the week label to the left of the first day of the month. (For \textit{week} list and \textit{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.)

5 Node Families

TikZ Library ext.node-families

\usetikzlibrary{ext.node-families} % \LaTeX{} and plain \TeX{}
\usetikzlibrary{ext.node-families} % Con\TeX

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

Unfortunately, this does not work with the external library.²

Q & A: [8] & [20]

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

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

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

5.1 Text Box

The following keys – when setup, see below – work with every shape with one single node part.³ Initially though, only circle and rectangle are set up that way.

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

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

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

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

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

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

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

Sets text height, text depth and text width.

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

²First of all, I can’t figure out how to use the aux file during externalization since it gets written to the log instead. And then there’s the question about how external would notice the need to export the picture again until it’s stable …

³Technically, it will also work with shapes with multiple node parts but it will only affect the main node part.
/tikz/node family/text width align=(alignment)

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

/foo

/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 only be 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.

/foo

/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.
5.3 More shapes that support the keys width and height

TikZ Library ext.node-families.shapes.geometric

\usetikzlibrary{ext.node-families.shapes.geometric} % \LaTeX{} and plain \TeX{}
\usetikzlibrary{ext.node-families.shapes.geometric} % Con\TeX

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

Q: [13]

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

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

TikZ Library ext.paths.arcto

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

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

When this operation is used, the path gets extended by an arc that goes through the current point and \texttt{(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 \texttt{(options)}.

\begin{tikzpicture}[ultra thick,dot/.style={label=\textcolor{white}{#1}}]
\coordinate[dot=below left:$a$] (a) at (0,0);
\coordinate[dot=above right:$b$] (b) at (2,3);
\begin{scope}[
odes={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}
This constructs an arc that goes clockwise.

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

This constructs an arc that goes counter clockwise.
This is the default.

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

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

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

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

/tikz/arc to/rotate=(degree) (no default)

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

/tikz/arc to/x radius=(value) (no default)

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

/tikz/arc to/y radius=(value) (no default)

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

/tikz/arc to/radius=(value) (no default)

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

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

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

It should be noted that this uses \pgfpatharcto for which the 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 $90^\circ$.

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

TikZ Library `ext.paths.ortho`

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

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 ... `-|([options])`\texttt{coordinate or cycle} ...;

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

\path ... `-|([options])`\texttt{coordinate or cycle} ...;

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

\texttt{/tikz/ortho/ratio=}\texttt{(ratio)}

(no default, initially 0.5)

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

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

\texttt{/tikz/ortho/distance=}\texttt{(distance)}

(no default)

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

For values \texttt{(distance)} < 0 the distance will be used for the target coordinate.
\usetikzlibrary{ext.paths.ortho}
\begin{tikzpicture}[very thick,-latex]
\draw[help lines,-] (-.25, -.25) grid (5.25, 3.25);
\draw (0, 0) -|-[distance=.5cm] ++(2, 1);
\draw (0, 2) -|-[distance=-.5cm] ++(2, 1);
\tikzset{xshift=3cm}
\draw (2, 1) -|-[distance=.5cm] ++(-2, -1);
\draw (2, 3) -|-[distance=-.5cm] ++(-2, -1);
\end{tikzpicture}

\tikz/ortho/from center=(true or false)

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

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

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

\tikz/ortho/spacing=(number)

Unless \texttt{(number) = 0} is set

- pos = 0 will be at the start,
- pos = 1 will be at the end,
- \( \frac{1}{\texttt{(number)}} \) will be at the first kink,
- \( \frac{\texttt{(number)}-1}{\texttt{(number)}} \) 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.

\(/\text{tikz/ortho/middle 0 to 1}\)

This is an alias for \(\text{spacing} = 0\).

### 7.2 Zig-Zig

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

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

\(/\text{tikz/ortho/ud distance}=(\text{length})\)

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

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

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

\(/\text{tikz/ortho/du distance}=(\text{length})\)

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

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

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

\(/\text{tikz/ortho/lr distance}=(\text{length})\)

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

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

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

\(/\text{tikz/ortho/rl distance}=(\text{length})\)

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

All distances can be set with one key.

\(/\text{tikz/ortho/udlr distance}=(\text{length})\)

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

/tikz/vertical horizontal
This installs \texttt{path = |- (\tikztotarget) \tikztonodes} that can be used with the path operations \texttt{to} or \texttt{edge}.

/tikz/horizontal vertical horizontal
This installs \texttt{path = -|- (\tikztotarget) \tikztonodes} that can be used with the path operations \texttt{to} or \texttt{edge}.

/tikz/vertical horizontal vertical
This installs \texttt{path = |-| (\tikztotarget) \tikztonodes} that can be used with the path operations \texttt{to} or \texttt{edge}.

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

/tikz/only vertical second=⟨length⟩
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⟩
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⟩
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⟩
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}

\texttt{(style, no value)}

Installs the following shortcuts:
\begin{verbatim}
| -   \rightarrow vertical horizontal
-|   \rightarrow horizontal vertical
-|-  \rightarrow horizontal vertical horizontal
|-|  \rightarrow vertical horizontal vertical
|*   \rightarrow only vertical first
*|   \rightarrow only vertical second
-*  \rightarrow only horizontal first
*-  \rightarrow only horizontal second
\end{verbatim}
8 Extending the Path Timers

TikZ Library \texttt{ext.paths.timer}

\begin{verbatim}
\usetikzlibrary{ext.paths.timer} % L ATEX and plain \TeX
\usetikzlibrary{ext.paths.timer} % Con\TeX

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

Q \& A: [5, 4] \& [26, 32]

In \texttt{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{tikzpicture}
\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}

8.2 Parabola

For the \texttt{parabola} path operator the timer is similar to the \texttt{.. controls ..} operator.

\begin{tikzpicture}
\begin{scope}[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{scope}
\end{tikzpicture}
The position 0.5 will lie at the bend.

\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.

\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}

\section{8.3 Sine/Cosine}
The \texttt{sin} and \texttt{cos} path operators also allow placing of nodes along their paths.

\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,-1) sin (1,0) [mark nodes on line];
\draw[shift={(0:1)}] (-2,-1) cos (1,0) [mark nodes on line];
\end{tikzpicture}
9 Using Images as a Pattern

TikZ Library `ext.patterns.images`

\usetikzlibrary{ext.patterns.images} % \LaTeX and plain \TeX
\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] & [31]

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

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

\pgfsetupimageaspattern{((options))}{(name)}{(image)}

/tikz/image as pattern={options}  \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

TikZ Library `ext.positioning-plus`

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

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

10.1 Useful corner anchors

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

/tikz/corner above left=(\textit{specification})

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

/tikz/corner below left=(\textit{specification})

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

/tikz/corner above right=(\textit{specification})

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

/tikz/corner below right=(\textit{specification})

Similar as `/tikz/below right` of the TikZ library positioning but uses the `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 /tikz/right but aligns the nodes at their north border.
This is basically the same as left=of reference.north east, anchor=north west.

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

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

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

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

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

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

\begin{itemize}
\item \texttt{/tikz/corner north left=\{specification\}} \hfill (style, default 0pt)
\begin{quote}
The same as \texttt{/tikz/north left} but uses the new corner anchors.
\end{quote}
\item \texttt{/tikz/corner north right=\{specification\}} \hfill (style, default 0pt)
\begin{quote}
The same as \texttt{/tikz/north right} but uses the new corner anchors.
\end{quote}
\item \texttt{/tikz/corner south left=\{specification\}} \hfill (style, default 0pt)
\begin{quote}
The same as \texttt{/tikz/south left} but uses the new corner anchors.
\end{quote}
\item \texttt{/tikz/corner south right=\{specification\}} \hfill (style, default 0pt)
\begin{quote}
The same as \texttt{/tikz/south right} but uses the new corner anchors.
\end{quote}
\item \texttt{/tikz/corner west above=\{specification\}} \hfill (style, default 0pt)
\begin{quote}
The same as \texttt{/tikz/west above} but uses the new corner anchors.
\end{quote}
\item \texttt{/tikz/corner west below=\{specification\}} \hfill (style, default 0pt)
\begin{quote}
The same as \texttt{/tikz/west below} but uses the new corner anchors.
\end{quote}
\item \texttt{/tikz/corner east above=\{specification\}} \hfill (style, default 0pt)
\begin{quote}
The same as \texttt{/tikz/east above} but uses the new corner anchors.
\end{quote}
\item \texttt{/tikz/corner east below=\{specification\}} \hfill (style, default 0pt)
\begin{quote}
The same as \texttt{/tikz/east below} but uses the new corner anchors.
\end{quote}
\end{itemize}

While the \texttt{(specification)} of all these keys still accept the same form as with Ti\textsc{k}Z, the \texttt{ext.positioning-plus} library extends this even more.

The specification after of can contain a list of coordinates (like the \texttt{fit} key of the \texttt{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 \texttt{| -} or \texttt{+}, the new node will also have the same height (\texttt{|}), the same width (\texttt{-}) or both as this bounding box.

\begin{itemize}
\item BCD
\end{itemize}

This functionality is also available without the placement:
/tikz/fit bounding box=(list of coordinates) (style, no default)

Creates a rectangular node with the name fit bounding box that encompasses the (list of coordinates).

/tikz/span vertical=(list of coordinates) (style, no default)

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

/tikz/span horizontal=(list of coordinates) (style, no default)

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

/tikz/span=(list of coordinates) (style, no default)

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

As you maybe noticed in the example above, the (specification) also allows a prefix delimited by : which the node distance will be multiplied to with for the placement.4

4This is probably more useful when /tikz/on grid is used.
11 Scaling Pictures to a Specific Size

TikZ Library `ext.scalepicture`

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

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

If one of the keys below are used on a TikZ picture, i.e. as an option to \tikzpicture or \begin{tikzpicture} the size of the picture\(^5\) will be measured and written to the aux file so that it will be available at the next compilation run and an appropriate scaling for the picture can be installed.

\tikzextpicturewidth
   Returns the last measured width of the picture.
   This will expand to 0pt if the picture hasn’t been measured before.

\tikzextpictureheight
   Returns the last measured height of the picture.
   This will expand to 0pt if the picture hasn’t been measured before.

\tikz/save picture size
   (style, no value)
   This key is usually used by the keys provided by this library. Normally, this is not needed to be explicitly given.

11.1 Keeping the aspect ratio

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

\tikz/picture width=(dimension)  \quad (no default)
   Scales the picture so that the width of the picture will be (dimension). This will keep the aspect ratio the same.

\tikz/minimum picture width=(dimension)  \quad (no default)
   As above but will not change the size of the picture if its width is greater than (dimension).

\tikz/maximum picture width=(dimension)  \quad (no default)
   As above but will not change the size of the picture if its width is less than (dimension).

\tikz/picture height=(dimension)  \quad (no default)
   Scales the picture so that the height of the picture will be (dimension). This will keep the aspect ratio the same.

\tikz/minimum picture height=(dimension)  \quad (no default)
   As above but will not change the size of the picture if its height is greater than (dimension).

\tikz/maximum picture height=(dimension)  \quad (no default)
   As above but will not change the size of the picture if its height is less than (dimension).

\tikz/minmum picture size={⟨width⟩}{⟨height⟩}  \quad (no default)
   Scales the picture so that its height will be at least ⟨width⟩ and its height will be at least ⟨height⟩.

\tikz/maximum picture size={⟨width⟩}{⟨height⟩}  \quad (no default)
   Scales the picture so that its height will be at most ⟨width⟩ and its height will be at most ⟨height⟩.

11.2 Changing the aspect ratio.

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

\tikz/picture width*=⟨dimension⟩  \quad (no default)
   Scales the picture so that the width of the picture will be (dimension). This will only scale the x axis.

\(^5\)This is the size of the pseudo-node current bounding box.
/tikz/minimum picture width*=(dimension) (no default)
As above but will not change the size of the picture if its width is greater than (dimension).

/tikz/maximum picture width*=(dimension) (no default)
As above but will not change the size of the picture if its width is less than (dimension).

/tikz/picture height*=(dimension) (no default)
Scales the picture so that the height of the picture will be (dimension). This will only scale the y axis.

/tikz/picture size*={(width)}{(height)} (no default)
Scales the picture so that its width will be (width) and its height will be (height). This will scale both axes but independent from each other.
12 Arrows through Three Points

TikZ Library \texttt{ext.topaths.arcthrough}

\begin{itemize}
\item \texttt{\usetikzlibrary{ext.topaths.arcthrough}} \% \LaTeX{} and plain \TeX{}
\item \texttt{\usetikzlibrary{ext.topaths.arcthrough}} \% ConTeXt
\end{itemize}

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

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

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

\begin{itemize}
\item /tikz/arc through/\texttt{through}=\texttt{(coordinate)} \begin{itemize}
\item (no default, initially \(0,0\))
\end{itemize}
The coordinate on the circle that defines – together with the starting and target point – a circle.

\item /tikz/arc through/\texttt{center suffix}=\texttt{(suffix)} \begin{itemize}
\item (no default, initially )
\end{itemize}
The arc through will define a coordinate named \texttt{arc through center\{suffix\}} so that it can be referenced later.

\item /tikz/arc through\texttt{=\{key-value\}} \begin{itemize}
\item (no value)
\end{itemize}
The resulting arc will go clockwise from the starting point to the target point.

\item /tikz/arc through/\texttt{counter clockwise} \begin{itemize}
\item (no value)
\end{itemize}
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 /tikz/arc through\texttt{=\{key-value\}} \begin{itemize}
\item (no default)
\end{itemize}
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.
\end{itemize}

27
13 Mirror, Mirror on the Wall

TikZ Library ext.transformations.mirror

\usetikzlibrary{ext.transformations.mirror} % \LaTeX and plain \TeX
\usetikzlibrary[ext.transformations.mirror] % Con\TeXt

This library adds more transformations to TikZ.

As explained in section 14, there are two approaches to setting a mirror transformation. As with the commands in PGF, we'll be using a lowercase \texttt{m} for the reflection matrix and an uppercase \texttt{M} for the built-in approach.

13.1 Using the reflection matrix

\begin{tikzpicture}
  \node[reg poly=5, minimum size=2cm, draw, very thick] (a) {};
  \foreach \i [evaluate={\col=(\i-1)/.04}] in {1,...,5}
    \node[mirror=(a.corner \i)--(a.side \i), transform shape, reg poly=5, minimum size=2cm, draw=red!\col!blue] {};
\end{tikzpicture}

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

Sets up a transformation that mirrors along a horizontal line that goes through point ⟨(value),0⟩ or ⟨coordinate⟩.
/tikz/ymirror=(value or coordinate) (default 0pt)

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

/tikz/mirror x=(coordinate) (default (0,0))

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

/tikz/mirror y=(coordinate) (default (0,0))

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

/tikz/mirror=(point A)--(point B)

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.

13.2 Using built-in transformations
/tikz/xMirror=(value or coordinate)  
Sets up a transformation that mirrors along a horizontal line that goes through point ((value), 0) or (coordinate).

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

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

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

/tikz/Mirror=(point A)--(point B)  
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.
14 Transformations: Mirroring

PGF Library `ext.transformations.mirror`
\usepgflibrary{ext.transformations.mirror} % \LaTeX{} and plain \TeX{}
\usepgflibrary{ext.transformations.mirror} % Con\TeX{}t

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.

14.1 Using the reflection matrix

The following commands use the reflection matrix that sets the transformation matrix following
\[ A = \frac{1}{\|\vec{l}\|^2} \begin{bmatrix} l_x^2 - l_y^2 & 2l_xl_y \\ 2l_xl_y & l_x^2 - l_y^2 \end{bmatrix}. \]

\texttt{\textbackslash pgftransformxmirror\{(value)\}}

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

14.2 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{\textbackslash pgftransformxMirror\{(value)\}}

Sets up a transformation that mirrors along a vertical line that goes through point \((\texttt{value}, 0)\).
\texttt{\textbackslash pgftransformmirror\{\textit{value}\}}

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

\texttt{\textbackslash pgftransformmirror\{\textit{point A}\}\{\textit{point B}\}}

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

\texttt{\usepgflibrary\{transformations.mirror\}}
\begin{tikzpicture}
\draw[help lines] (-.25, -2.25) grid (2.5, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\draw[dashed] (0, -1) -- (2, 0);
\pgftransformmirror\{\pgfpointxy{0}{-1}\}\{\pgfpointxy{2}{0}\}
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}

\texttt{\textbackslash pgfqtransformmirror\{\textit{point A}\}}

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

\texttt{\usepgflibrary\{transformations.mirror\}}
\begin{tikzpicture}
\draw[help lines] (-.25, -.25) grid (2.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\draw[dashed] (0, 0) -- (2, 1);
\pgfqtransformmirror\{\pgfpointxy{2}{1}\}
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
A circular shape named *circle arrow* that has an arc as its background path that can have an arrow tip.

**Q & A: [16] & [27]**

**Shape circle arrow**

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

```
\pgfdeclarearrow{start angle=}{end angle}{delta angle} {start arrow tip specification}{end arrow tip specification}
```

A few handful styles are pre-defined.

```
/pgf/circle arrow turn left north  
  Sets `circle arrow start angle = 100`, `circle arrow delta angle = 340` and `circle arrow arrows = ->`.  

/pgf/circle arrow turn left east  
  As above but `circle arrow start angle = 10`.  

/pgf/circle arrow turn left west  
  As above but `circle arrow start angle = 280`.  

/pgf/circle arrow turn left south  
  As above but `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,matrix}
\begin{tikzpicture}
\matrix[matrix of nodes, draw=none, row sep=1em, column sep=1em, every node/.style={draw=gray, shape=circle arrow, ultra thick, inner sep=1em}]
(m) {
    ![circle arrow turn left north] & ![circle arrow turn left east] \\
    ![circle arrow turn left west] & ![circle arrow turn left south] \\
    ![circle arrow turn right north] & ![circle arrow turn right east] \\
    ![circle arrow turn right west] & ![circle arrow turn right south] \\
};
\end{tikzpicture}
35
\usetikzlibrary{ext.shapes.circlearrow}
\begin{tikzpicture}
\node[name=s, shape=circle arrow, circle arrow turn left west, shape example]
{Circle Arrow} \vrule width 1pt height 2cm;
\foreach \anchor/\placement in
{north west/above left, north/above, north east/above right, west/left, center/above, east/right, mid west/right, mid/above, mid east/left, base west/left, base/below, base east/right, south west/below left, south/below, south east/below right, text/left, 10/right, 130/above}
\draw[shift=s.\anchor] plot[mark=x] coordinates{(0,0)} node[\placement] {\scriptsize\texttt{(s.\anchor)}};
\end{tikzpicture}
16 Shape: Circle Cross Split

\textbf{TikZ Library} \texttt{ext.shapes.circlecrosssplit}
\begin{verbatim}
\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
\end{verbatim}

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


\textbf{Shape circle cross split}

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

\texttt{/pgf/circle cross split part fill=\langle list\rangle}
(no default, initially \texttt{none})

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 \texttt{/pgf/circle cross split uses custom fill}.

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

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

TikZ Library `ext.shapes.heatmark`

\usepgflibrary{ext.shapes.heatmark} % \LaTeX{} and plain \TeX{} and pure pgf
\usetikzlibrary{ext.shapes.heatmark} % ConTExt and pure pgf
\usetikzlibrary{ext.shapes.heatmark} % \LaTeX{} and plain \TeX{} when using Ti\textit{k}Z
\usetikzlibrary{ext.shapes.heatmark} % ConTExt when using Ti\textit{k}Z

A circular shape that has customizable rings around it.

Q&A: [2] & [23]

Shape heatmark

 `/pgf/heatmark arcs=(arcs num)`
Sets the number of arc around the circle to `(arcs num)`. (no default, initially 3)

 `/pgf/heatmark arc width=(arc width)`
Sets the width of the rings around the circle to `(arc width)`. (no default, initially 4pt)

 `/pgf/heatmark arc sep=(sep length)`
Sets the whitespace between the rings to `(sep length)`.

 `/pgf/heatmark arc rings=(rings num)`
Sets the number of rings around the circle to `(rings num)` (no default, initially 3)

 `/pgf/heatmark arc sep angle=(sep angle)`
Sets the whitespace angle between the arcs in one ring to `(sep angle)`.

 `/pgf/heatmark inner opacity=(inner opacity)`
Sets the opacity of the inner ring to `(inner opacity)`.

 `/pgf/heatmark outer opacity=(low opacity)`
Sets the opacity of the outer ring to `(low opacity)`.

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

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

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

 `/pgf/heatmark ring (ring number)`
(style, no value)
The \texttt{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.
18 Shape: Rectangle with Rounded Corners

TikZ Library \texttt{ext.shapes.rectangleroundedcorners}

\begin{itemize}
  \item \texttt{\usepgflibrary{ext.shapes.rectangleroundedcorners}} % \LaTeX{} and plain \TeX{} and pure pgf
  \item \texttt{\usepgflibrary[ext.shapes.rectangleroundedcorners]} % Con\TeX{}t and pure pgf
  \item \texttt{\usetikzlibrary{ext.shapes.rectangleroundedcorners}} % \LaTeX{} and plain \TeX{} when using TikZ
  \item \texttt{\usetikzlibrary[ext.shapes.rectangleroundedcorners]} % Con\TeX{}t when using TikZ
\end{itemize}

A rectangle with rounded corners.

Shape \texttt{rectangle with rounded corners}

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

\begin{itemize}
  \item \texttt{/pgf/rectangle with rounded corners north west radius=(dimen)}
    \hspace{1cm} (no default, initially \texttt{.5\pgflinewidth})
    \hspace{1cm} Sets the north west radius to \texttt{(dimen)}.
  \item \texttt{/pgf/rectangle with rounded corners north east radius=(dimen)}
    \hspace{1cm} (no default, initially \texttt{.5\pgflinewidth})
    \hspace{1cm} Sets the north east radius to \texttt{(dimen)}.
  \item \texttt{/pgf/rectangle with rounded corners south west radius=(dimen)}
    \hspace{1cm} (no default, initially \texttt{.5\pgflinewidth})
    \hspace{1cm} Sets the south west radius to \texttt{(dimen)}.
  \item \texttt{/pgf/rectangle with rounded corners south east radius=(dimen)}
    \hspace{1cm} (no default, initially \texttt{.5\pgflinewidth})
    \hspace{1cm} Sets the south east radius to \texttt{(dimen)}.
  \item \texttt{/pgf/rectangle with rounded corners radius=(dimen)}
    \hspace{1cm} (no default)
    \hspace{1cm} Sets all radii to \texttt{(dimen)}.\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}
19 Shape: Superellipse

TikZ Library \texttt{ext.shapes.superellipse}

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

Shape in the form of a "superellipse''.

Q & A: [35] & [19]

Shape \texttt{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

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

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

\texttt{/pgf/superellipse x exponent=⟨x exponent⟩} \hspace{1cm} (no default, initially 2.5)

This sets \(m\).

\texttt{/pgf/superellipse y exponent=⟨y exponent⟩} \hspace{1cm} (no default, initially 2.5)

This sets \(n\).

\texttt{/pgf/superellipse step=⟨step⟩} \hspace{1cm} (no default, initially 5)

This specifies the step of the underlying plot handler. The smaller \(⟨\text{step}⟩\) is, the slower computation will be.

Sensible values for \(⟨\text{step}⟩\) are integer dividers of 90, i.e. 2, 3, 5, 6, 9, 10, 15, 18, 30 and 45.

\texttt{/pgf/superellipse exponent=⟨exponent⟩} \hspace{1cm} (no default)

Sets both \texttt{superellipse x exponent} and \texttt{superellipse y exponent} to \(⟨\text{exponent}⟩\).
Notes on Implementation  For implementing this shape, additional mathematical functions were declared.

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

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 $0 \leq t \leq 90$.

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

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

\begin{align*}
\text{\texttt{pgfmathsuperellipseXY}}{(t)}{(2/m)}{(2/n)}{(a)}{(b)}
\end{align*}

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*}
\text{\texttt{pgfmathresultX}} &= a \cos^{2/m} t \\
\text{\texttt{pgfmathresultY}} &= 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{align*}
\text{\texttt{\pgfutil@prefix@macrotomacro}}{(macro 1)}{(macro 2)}
\end{align*}

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

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

TikZ Library \texttt{ext.shapes.uncenteredrectangle}

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

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

Q & A: [34, 15] & [24, 21]

\textbf{Shape uncentered rectangle}

For some alignment problems, this shape could be useful.

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

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

\begin{verbatim}
\pgf/uncentered rectangle center=(\textit{left}) or (\textit{text}) or (\textit{right}) or (\textit{real}) \ (no default, initially text)
\end{verbatim}

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, these are marked red.

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

\begin{verbatim}
\pgf/uncentered rectangle center yshift=(\textit{dimension}) \ (no default, initially {})\end{verbatim}

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

If (\textit{dimension}) is empty, the real center will be used which is the default.

For use with \texttt{cd}, set this to \texttt{axis_height}. Due to a bug with \texttt{/tikz/execute at end node} this needs a lot of fixing to be able to use in a commutative diagram, though.
\usetikzlibrary{cd, ext.shapes.uncenteredrectangle}
\makeatletter
\tikzcdset{
  every diagram/.append style={(n)\pgf/uncentered rectangle center=text,\pgf/uncentered rectangle center yshift=axis_height,\commutative diagrams\math mode=false,\tikz/math align/.style={\math align,\tikz/math align right/.style={\math align,\tikz/matrix of math nodes/.style={(\tikzcdset{install C shortcut/.code=\newcommand\C[1]{C_{\%_{##1}}}})(\begin{tikzcd}
install C shortcut, sep=tiny, arrows={-, gray}, cells={font=\strut, inner xsep=.2ex, inner ysep=.1ex}
\C(1) \drar & \& \C(2) \urar & \\C_1 \& m_r = \C_2 - \C_1 \& \C_1 \& m_r = \C_2 - \C_1
\end{tikzcd)}}}
\usepgflibrary{ext.shapes.uncenteredrectangle}
\begin{tikzpicture}[style north/.style=red, style south/.style=red, style center/.style=red, style base/.style=red, style mid/.style=red]
  \Huge
  \node[shape example, name=n, uncentered rectangle]{centered
    \nodepart{left} Un \nodepart{right} \space Rectangle\vrule width 1pt height 2cm}
  \foreach \anchor/\pos in {
    north west/above left, north/below, north east/above right, real north/above, left north/above, right north/above, text north/above,
    west/\left, center/above, east/\right, real center/above, left center/above, right center/above, text center/below,
    mid west/\left, mid/\left, mid east/\right, real mid/above, left mid/above, right mid/above, text mid/above,
    base west/\left, base/right, base east/\right, real base/\below, left base/\below, right base/\below, text base/\below,
    south west/below left, south/\above, south east/\below right, real south/\below, left south/\below, right south/\below, text south/\below,
    10/\right, 130/\below, left/\left, right/\right, text/\right}{
    plot[mark=x, only marks] coordinates {(n.\anchor)}
  }
\end{tikzpicture}
Part IV
Utilities

\usetikzlibrary{ext.misc}
\begin{tikzpicture}
\begin{scope}[radius=0.3cm, xshift=1cm, declare function={bigR(\n)=smallR+.05*\n;}, declare constant={smallR=1; segments=20;}, full arc=\segments]
\foreach \i in {\segments} {evaluate={\endRadius=bigR(\i+1);}} {use int=0 to \segments-1}
\filldraw[fill=gray!50] (\i R:smallR) arc [radius=smallR, start angle=\i R, delta angle=-1R] -- (\i R+1R:smallR) arc [radius=smallR, end angle=\i R, delta angle=+1R] -- cycle;
\node {$\phi^2$};
\node at (north west:{sqrt 2 * bigR(\segments/2)}) {$\{\Omega\}_{i=1}^n$};
\node[rotate=-.5R, right] at (-.5R: bigR \segments) {$\partial \varphi$};
\end{tikzpicture}

\usetikzlibrary{ext.misc}
\begin{tikzpicture}
\begin{scope}[radius=0.3cm, xshift=1cm, declare function={bigR(\n)=smallR+.05*\n;}, declare constant={smallR=1; segments=20;}, full arc=\segments]
\foreach \i in {\segments} {evaluate={\endRadius=bigR(\i+1);}} {use int=0 to \segments-1}
\filldraw[fill=gray!50] (right:smallR) \foreach \i in {\segments} {\input{\i R:bigR(\segments/2)}} arc [radius=\endRadius, start angle=\i R, delta angle=+1R] -- (\i R+1R:smallR) arc [radius=smallR, end angle=\i R, delta angle=-1R] -- cycle;
\node {$\phi^2$};
\node at (north west:{sqrt 2 * bigR(\segments/2)}) {$\{\Omega\}_{i=1}^n$};
\node[rotate=-.5R, right] at (-.5R: bigR \segments) {$\partial \varphi$};
\end{tikzpicture}
21 Calendar: Weeknumbers and more conditionals

\usepackage{calendar-ext} % TeX
\input calendar-ext.tex % plain TeX
\usemodule[calendar-ext] % ConTeXt

This package adds week numbers and more conditionals to the PGF package pgfcalendar. (Despite the code example above, this package is not set up to work with ConTeXt.)

Q & A: [6, 7, 9] & [18, 30, 22]

21.1 Extensions

The following tests are added.

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

The shorthands for d- and m- are slightly changed so that they are expandable. This makes it possible to use these shorthands inside of PGFmath. The shorthands for the week (see section 21.2) are added. These are
• \(n\) (shortest numerical representation),
• \(n\#\) (shortest but added horizontal space) and
• \(n0\) (leading zero when below 10).

### 21.2 Week numbering (ISO 8601)

\[\text{\texttt{\textbackslash pgfcalendarjulianyeartoweek\{\textit{Julian day}\}\{\textit{year}\}\{\textit{week counter}\}}}\]

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

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

Inside of \texttt{\textbackslash pgfcalendar} the command \texttt{\textbackslash pgfcalendarcurrentweek} will be available.

\texttt{\textbackslash pgfcalendarcurrentweek}

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

Inside of \texttt{\textbackslash ifdate} the command \texttt{\textbackslash pgfcalendarifdateweek} will be available.

\texttt{\textbackslash pgfcalendarifdateweek}

This command returns the week number (always two digits).
22 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: [14, 1] & [17, 25]

22.1 PGFmath

22.1.1 Postfix operator \texttt{R}

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

\texttt{/tikz/full arc=<num>}
\begin{itemize}
  \item The number \texttt{(num)} of segments will be set up. Using \texttt{full arc} with an empty value disables the segmentation and \texttt{1R} equals 1°.
  \item The given value \texttt{(num)} is evaluated when the key is used and doesn’t change when \texttt{(num)} contains variables that change.
\end{itemize}

The \texttt{R} operator can then be used.
\begin{itemize}
  \item \texttt{xR} \begin{itemize}
    \item Multiplying \texttt{x} with \texttt{\frac{360}{\texttt{(num)}}}.
  \end{itemize}
\end{itemize}

22.1.2 Functions

\texttt{strrepeat("Text", x)}
\texttt{\pgfmathstrrepeat{"Text"}{x}}
\begin{itemize}
  \item Returns a string with \texttt{Text} repeated \texttt{x} times.
\end{itemize}

\texttt{isInString("String", "Text")}

\begin{itemize}
  \item Returns 1 (true) if \texttt{Text} contains \texttt{String}, otherwise 0 (false).
\end{itemize}

\begin{itemize}
  \item \texttt{\pgfmathisInString("String")["Text"]}
\end{itemize}

\texttt{strcat("Text A", "Text B", \ldots)}
\texttt{\pgfmathstrcat{"Text A"}{"Text B"}{\ldots}}
\begin{itemize}
  \item Returns the concatenation of all given parameters.
\end{itemize}

\texttt{isEmpty("Text")}
\texttt{\pgfmathisEmpty{"Text"}}
\begin{itemize}
  \item Returns 1 (true) if \texttt{Text} is empty, otherwise 0 (false).
\end{itemize}

\begin{itemize}
  \item \texttt{\pgfmathisEmpty{"foo"}} \texttt{\pgfmathresult} \texttt{\pgfmathisEmpty{"bar"}} \texttt{\pgfmathresult} \texttt{\pgfmathisEmpty{"foobar"}} \texttt{\pgfmathresult}
\end{itemize}

\texttt{atanXY(x, y)}
\begin{itemize}
  \item \texttt{\pgfmathparse{atanXY(int(7*3), "green")}} \texttt{\pgfmathresult}
\end{itemize}
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{atanXY} commands of the \texttt{calc} library.

\pgfmathatanXY{x}{y}

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

\pgfmathatanYX{y}{x}

22.1.3 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 ".

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

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

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

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

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

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

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

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

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

22.2 \texttt{PGFfor}

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

/\texttt{pgf/foreach/use int=(start)to(end)step(delta)}

(no default)
The values \((start)\), \((end)\) and \((delta)\) are evaluated by \texttt{pgfmath} at initialization. The part \texttt{step} \((delta)\) is optional \((delta = 1)\).

\begin{verbatim}
/pgf/foreach/use float=(start)to (end) step (delta)
\end{verbatim}

Same as above, however the results are not truncated.

### 22.3 \texttt{PGFkeys}

\texttt{pgfkeys Library \texttt{ext.pgfkeys-plus}}

\begin{verbatim}
\usepgfkeyslibrary\{ext.pgfkeys-plus\} % \LaTeX and plain \TeX
\usepgfkeyslibrary\[ext.pgfkeys-plus\] % Con\TeXt
\end{verbatim}

This extends \texttt{pgfkeys} and adds helpful /\texttt{utils} keys as well as handlers. This library gets loaded by the \texttt{ext.misc} library.\(^6\)

#### 22.3.1 Conditionals

\begin{verbatim}
/util\texttt{s/\TeX/\if=(dim cond)(true)(false)}
\end{verbatim}

As above but with \texttt{\ifdim}.

\begin{verbatim}
/util\texttt{s/\TeX/\ifempty=(Text)(true)(false)}
\end{verbatim}

This checks whether \texttt{\TeX} is empty and applies styles \texttt{(true)} if true, otherwise \texttt{(false)}.

\begin{verbatim}
/util\texttt{s/\TeX/\ifxempty=(Text)(true)(false)}
\end{verbatim}

This checks whether fully expanded \texttt{\TeX} is empty and applies styles \texttt{(true)} if true, otherwise \texttt{(false)}.

#### 22.3.2 Handlers

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

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

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

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

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

As above but truncates the result.

\(^6\)\texttt{\usepgfkeyslibrary} is an upcoming feature of \texttt{PGF/Ti\LaTeX}. For now, you need to load \texttt{ext.misc} or manually \texttt{\input} the file \texttt{pgfkeyslibraryext.pgfkeys-plus.code.tex} with \texttt{@} being a letter.
Key handler (key) /.pgfmath wrap={{(wrapper)}{(eval)}}
This feeds the result of (eval) as #1 to (wrapper).
In the example below, one could have used the /pgf/foreach/evaluate key from the \foreach loop.

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

Key handler (key) /.pgfmath if={{(cond)}{(true)}{(false)}}
Evaluates (cond) with PGFMath and returns (true) or (false) to the used key respectively.

Key handler (key) /.if={(token A)(token B)}{(true)}{(false)}
Checks via \if if (token A) matches (token B) and applies the value (true) if it does, otherwise (false).

Key handler (key) /.ifnum={{(ifnum cond)}{(true)}{(false)}}
As above but via \ifnum.

Key handler (key) /.ifdim={{(ifdim cond)}{(true)}{(false)}}
As above but via \ifdim.

Key handler (key) /.ifxempty={{Text} {(true)}{(false)}}
Checks whether a fully expanded (Text) is empty and applies the value (true) if it does, otherwise (false).

Key handler (key) /.ifempty={{Text} {(true)}{(false)}}
Checks whether (Text) is empty and applies the value (true) if it does, otherwise (false).

Key handler (key) /.List={{(e1), (e2), ..., (en)}}
This handler evaluates the given list with \foreach and concatenates the element and the result is then given to the used key.
Part V
Changelog, Index & References

Changelog

Version 0.4.2
• Added TiKZ library ext.scalepicture.
• Bugfixes to shapes.uncenteredrectangle, paths.ortho, positioning-plus
  and pgfcalendar-ext.

Version 0.4.1
• Cleaned up directory structure of documentary.
• Added pgfkeys library ext.pgfkeys-plus.
• Added shape uncentered rectangle
  (PGF library ext.shapes.uncenteredrectangle).
• Fixed ext.paths.arcto – again [12].

Version 0.4
• CTAN version of 0.3.1

Version 0.3
• Added TiKZ library ext.node-families
  (PGF library ext.node-families).
• 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.

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 PGF/TiKZ.

| |- | path operation, 12
- |- | path operation, 12
- | | path operation, 12
- | - | path operation, 12
above key, 22
above left key, 20
above right key, 20
and date test, 52
angle between math function, 55
Apr date test, 52
arc path operation, 34
arc through key, 27
arc to path operation, 10
atan2 math function, 32, 55
atanXY math function, 54
atanYX math function, 55
Aug date test, 52
below key, 22
below left key, 20
below right key, 20
calc library, 55
calendar library, 6
calendar week of month date test, 52
calendar week of month' date test, 52
calendar-ext package, 52
cd library, 48
center suffix key, 27
circle shape, 7
circle arrow shape, 34
circle arrow arrows key, 34
circle arrow delta angle key, 34
circle arrow end angle key, 34
circle arrow start angle key, 34
circle arrow turn left east key, 34
circle arrow turn left north key, 34
circle arrow turn left south key, 34
circle arrow turn left west key, 34
circle arrow turn right east key, 35
circle arrow turn right north key, 35
circle arrow turn right south key, 35
circle arrow turn right west key, 35
circle cross split shape, 37
circle cross split part fill key, 37
circle cross split uses custom fill key, 37
clockwise key, 10, 27
corner above left key, 20
corner above right key, 20
corner below left key, 20
corner below right key, 20
corner east above key, 23
corner east below key, 23
corner north left key, 23
corner north right key, 23
corner south left key, 23
corner south right key, 23
corner west above key, 23
corner west below key, 23
cos math function, 32
cos path operation, 18
counter clockwise key, 11, 27
Date tests
    and, 52
    Apr, 52
    Aug, 52
    calendar week of month, 52
    calendar week of month', 52
    Dec, 52
    Feb, 52
ifempty key, 56
.ifnum handler, 57
.ifnum key, 56
.ifx handler, 57
.ifx key, 56
.ifxempty handler, 57
.ifxempty key, 56
image as pattern key, 19
isEmpty math function, 54
isInString math function, 54

Jan date test, 52
Jul date test, 52
Jun date test, 52

Key handlers
  .List, 57
  .evaluated, 56
  .if, 57
  .ifdef, 57
  .ifndef, 57
  .ifnum, 57
  .ifx, 57
  .ifndefx, 57
  .pgfmath, 56
  .pgfmath if, 57
  .pgfmath int, 56
  .pgfmath wrap, 57

large key, 11
lastdayinmonthofyear math function, 6
leap year date test, 52
left key, 21, 22

Libraries
  calc, 55
  calendar, 6
  cd, 48
  ext.calendar-plus, 6
  ext.misc, 54
  ext.node-families, 7
  ext.node-families.shapes.geometric, 9
  ext.paths.arcto, 10
  ext.paths.ortho, 12
  ext.paths.timer, 17
  ext.patterns.images, 19
  ext.positioning-plus, 20
  ext.scalepicture, 25
  ext.shapes.circlearrow, 34
  ext.shapes.circlecrosssplit, 37
  ext.shapes.heatmark, 40
  ext.shapes.rectangleroundedcorners, 43
  ext.shapes.superellipse, 45
  ext.shapes.uncenteredrectangle, 48
  ext.topaths.arcthrough, 27
  ext.transformations.mirror, 28, 32
  external, 7
  fit, 20
  positioning, 20
  shapes.geometric, 9
  .List handler, 57
  lr distance key, 14

Mar date test, 52

Math functions
  anglebetween, 55
  atan2, 32, 55
  atanXY, 54
  atanYX, 55
  cos, 32
  distancebetween, 55
  isEmpty, 54
  isInString, 54
  lastdayinmonthofyear, 6
  qanglebetween, 55
  qdistancebetween, 55
  sin, 32
  strcat, 54
  strrepeat, 54
  superellipsex, 46
  superellipsy, 46
  weeksinmonthofyear, 6
Math operators

\texttt{R}, 54
maximum picture height key, 25
maximum picture size key, 25
maximum picture width key, 25
maximum picture width key*, 26
May date test, 52
middle 0 to 1 key, 14
minimum height key, 8
minimum picture height key, 25
minimum picture size key, 25
minimum picture width key, 25
minimum picture width key*, 26
minimum width key, 8
Mirror key, 30
mirror key, 29
Mirror x key, 30
mirror x key, 29
Mirror y key, 30
mirror y key, 29
month code key, 6
month text key, 6
month xshift key, 6
month yshift key, 6
name key, 19
north left key, 21
north right key, 22
not date test, 52
Nov date test, 52
Oct date test, 52
on grid key, 24
only horizontal first key, 15
only horizontal second key, 15
only vertical first key, 15
only vertical second key, 15
option key, 19
options key, 19
Packages and files

calendar-ext, 52
parabola path operation, 17
Path operations

| - | - | - | - | 12
| - | - | - | - | 12
| - | - | - | - | 12
| - | - | - | - | 12
| arc | 34
arc to, 10
cos, 18
dge, 27
parabola, 17
r-du, 14
r-lr, 14
r-rt, 14
r-ud, 14
rectangle, 17
sin, 18
to, 27
/pgf/
circle arrow arrows, 34
circle arrow delta angle, 34
circle arrow end angle, 34
circle arrow start angle, 34
circle arrow turn left east, 34
circle arrow turn left north, 34
circle arrow turn left south, 34
circle arrow turn left west, 34
circle arrow turn right east, 35
circle arrow turn right north, 35
circle arrow turn right south, 35
circle arrow turn right west, 35
circle cross split part fill, 37
circle cross split uses custom fill, 37
foreach/
evaluate, 57
use float, 56
use int, 55
heatmark arc (arc number), 41
heatmark arc rings, 40
heatmark arc sep, 40
heatmark arc sep angle, 40
heatmark arc width, 40
heatmark arcs, 40
heatmark inner opacity, 40
heatmark outer opacity, 40
heatmark ring (ring number), 40
heatmark ring (ring number) arc (arc number), 41
minimum height, 8
minimum width, 8
rectangle with rounded corners north east radius, 43
rectangle with rounded corners north west radius, 43
rectangle with rounded corners radius, 43
rectangle with rounded corners south east radius, 43
rectangle with rounded corners south west radius, 43
shape border rotate, 40
superellipse exponent, 45
superellipse step, 45
superellipse x exponent, 45
superellipse y exponent, 45
text, 19
uncentered rectangle center, 48
uncentered rectangle center yshift, 48
\pgfcalendar, 53
\pgfcalendarcurrentweek, 53
\pgfcalendariifdateweek, 53
\pgfcalendarjulianyearweek, 53
\pgfkeys Libraries
- ext.pgfkeys-plus, 56
- .pgfmath handler, 56
- .pgfmath if handler, 57
- .pgfmath int handler, 56
- .pgfmath wrap handler, 57
- \pgfmathanglebetween, 55
- \pgfmathanglebetweenpoints, 32
- \pgfmathatanXY, 55
- \pgfmathatanYX, 55
- \pgfmathdistancetowith, 55
- \pgfmathisempty, 54
- \pgfmathisinstring, 54
- \pgfmathlastdayinmonthofyear, 6
- \pgfmathanglebetween, 55
- \pgfmathqdistancebetween, 55
- \pgfmathstrcat, 55
- \pgfmathstrrepeat, 55
- \pgfmathsuperellipsex, 46
- \pgfmathsuperellipseXY, 46
- \pgfmathsuperellipsy, 46
- \pgfmathweeksinnmonthofyear, 6
- \pgfminimumheight, 24
- \pgfminimumheightkey, 24
- \pgfminimumwidth, 24
- \pgfminimumwidthkey, 24
- \pgfpatharcto, 11
- \pgfpointnormalised, 32
- \pgftransformmirror, 33
- \pgftransformxmirror, 32
- \pgftransformyMirror, 33
- \pgfutil@prefix@macrotomacro, 46
- \pgftransformmirror, 33
- \pgftransformxmirror, 32
- \pgftransformymirror, 33
- \pgftransformymirror, 33
- \pgfutil@prefix@macrotomacro, 46
- picture height key, 25
- picture height* key, 26
- picture size* key, 26
- picture width key, 25
- picture width* key, 25
- pos key, 13, 17
- positioning library, 20
- prefix key, 7
- qanglebetween math function, 55
- qdistancebetween math function, 55
R postfix math operator, 54
r-du path operation, 14
r-lr path operation, 14
r-r1 path operation, 14
r-ud path operation, 14
radius key, 11
dratio key, 12
rectangle path operation, 17
rectangle shape, 7
rectangle with rounded corners shape, 43
rectangle with rounded corners north east radius key, 43
rectangle with rounded corners north west radius key, 43
rectangle with rounded corners radius key, 43
rectangle with rounded corners south east radius key, 43
rectangle with rounded corners south west radius key, 43
right key, 22
rl distance key, 14
rotate key, 11
save picture size key, 25
Sep date test, 52
setup shape key, 8
shape border rotate key, 40
Shapes
  circle, 7
circle arrow, 34
circle cross split, 37
heatmark, 40
rectangle, 7
rectangle with rounded corners, 43
superellipse, 45
uncentered rectangle, 48
shapes.geometric library, 9
sin math function, 32
sin path operation, 18
size key, 8
small key, 11
south left key, 22
south right key, 22
spacing key, 13
span key, 24
span horizontal key, 24
span vertical key, 24
strcat math function, 54
strrepeat math function, 54
superellipse shape, 45
superellipse exponent key, 45
superellipse step key, 45
superellipse x exponent key, 45
superellipse y exponent key, 45
superellipsex math function, 46
superellipsey math function, 46
text key, 19
text key, 7
text depth key, 7
text height key, 7
text width, 7
text width align key, 8
through key, 27
/tikz/
  above, 22
  above left, 20
  above right, 20
  arc through/
    center suffix, 27
    clockwise, 27
    counter clockwise, 27
    through, 27
  arc through, 27
  arc to/
    clockwise, 10
    counter clockwise, 11
    large, 11
    radius, 11
    rotate, 11
    small, 11
    x radius, 11
    y radius, 11
  below, 22
  below left, 20
  below right, 20
  corner above left, 20
  corner above right, 20
  corner below left, 20
  corner below right, 20

65
corner east above, 23
corner east below, 23
corner north left, 23
corner north right, 23
corner south left, 23
corner south right, 23
corner west above, 23
corner west below, 23
day code, 6
day text, 6
day xshift, 6
day yshift, 6
east above, 22
east below, 22
every arc to, 11
every day, 6
every month, 6
every week, 6
execute at end node, 48
execute at end picture, 7
fit bounding box, 24
full arc, 54
horizontal vertical, 15
horizontal vertical horizontal, 15
if, 6
image as pattern/
   name, 19
   option, 19
   options, 19
image as pattern, 19
left, 21, 22
maximum picture height, 25
maximum picture size, 25
maximum picture width, 25
maximum picture width*, 26
minimum picture height, 25
minimum picture size, 25
minimum picture width, 25
minimum picture width*, 26
Mirror, 30
mirror, 29
Mirror x, 30
mirror x, 29
Mirror y, 30
mirror y, 29
month code, 6
month text, 6
month xshift, 6
month yshift, 6
node family/
   height, 8, 9
   prefix, 7
   setup shape, 8
   size, 8
   text, 7
   text depth, 7
   text height, 7
   text width, 7
   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
picture height, 25
picture height*, 26
picture size*, 26
picture width, 25
picture width*, 25
pos, 13, 17
right, 22
save picture size, 25
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 label left, 6
week text, 6
west above, 22
west below, 22
x radius, 11
xMirror, 30
xmirror, 28
y radius, 11
yMirror, 30
ymirror, 29
\tikzextpictureheight, 25
\tikzextpicturewidth, 25
to path operation, 27
to path key, 15

ud distance key, 14
udlr distance key, 14
uncentered rectangle shape, 48
uncentered rectangle center key, 48
uncentered rectangle center yshift key, 48
use float key, 56
use int key, 55
/utils/
  if, 56
TeX/
    if, 56
    ifdim, 56
ifempty, 56
ifnum, 56
ifx, 56
ifxempty, 56
vertical horizontal key, 15
vertical horizontal vertical key, 15
week date test, 52
week code key, 6
week label left key, 6
week of month date test, 52
week of month' date test, 52
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, 30
xmirror key, 28
y radius key, 11
y radius key, 11
yesterday date test, 52
yMirror key, 30
ymirror key, 29

67
References


