`tzplot.sty`
Plot Graphs with Ti*k*Z Abbreviations

In-Sung Cho
`ischo <at> ktug.org`

Economics, Kongju National University

2021/03/20   version 1.0.1

Abstract

This is a LaTeX package that provides Ti*k*Z based macros to make it easy to draw graphs. The macros provided in the `tzplot` package are just *abbreviations* for Ti*k*Z codes, which can be complicated, but using the package, hopefully, makes drawing easier, especially when drawing repeatedly. The macros were chosen and developed with an emphasis on drawing graphs in *economics*.

Keywords: lines, dots, curves, axes, functions, projection, ticks, intersections, tangent lines

Contents

Part I
Getting Ready

1 Introduction

1.1 About tzplot.sty

The `tzplot` package is just a collection of macros based on Ti*k*Z to save you time typing Ti*k*Z code.

In `pstricks`, a line connecting two points `(A)` and `(B)` is drawn by `\psline(A)(B)`. With the package `tzplot`, you can do it by `\tzline(A)(B)`.

```
\tzline(A)(B)                    % is an abbreviation of:
  \draw (A) -- (B);
```

```
\tzline[blue](A)(B){my line}[r]  % is an abbreviation of:
  \draw [blue] (A) -- (B) node [right] {my line};
```

Some macros in this package represent one or a few lines of code, but some represent dozens of lines of Ti*k*Z code.

All of the drawing macros of `tzplot` are prefixed by `\tz`. Of course, it means Ti*k*Z. The syntax of the `tzplot` macros comes from `tikz` and `pstricks`. However, the input mode is more like `pstricks`.

To use the `tzplot` package you have to load the package in the preamble of your document as follows:

```
\usepackage{tzplot}
```

The package depends on the packages `tikz`, `xparse`, and `expl3`. And it uses the following Ti*k*Z libraries:

```
calc,backgrounds,positioning,intersections,arrows,shapes,patterns,plotmarks,
decorations.pathreplacing,calligraphy
```

This package sets the basic arrow style to `stealth`. If you don't like this, as an alternative, you can set the style like `\tikzset{>=to}` after the `tzplot` package is loaded.

This package was originally motivated by drawing graphs in economics. Therefore, the macros in this package have been selected and developed for drawing graphs efficiently in economics. However, this package will do a good job of drawing basic graphics in any fields.

Finally, note that this is far from a Ti*k*Z tutorial. To make good use of this package, you need to familiarize yourself with Ti*k*Z.

1.2 Preoccupied style names

This package does not provide any environment. Since all the drawing macros prefixed by `\tz` are just abbreviations of Ti*k*Z code, you can use the macros in the `tikzpicture` environment together with any Ti*k*Z commands.

However, there are some preoccupied style names that you should not overwrite. Those are as follows:

```
tzdot        tzmark         tzdotted
tzdashed     tzhelplines    tznode
tzshorten    tzextend       tzshowcontrols
```

This package also predefined *abbreviations* of Ti*k*Z's basic placement options as follows:

```
% preoccupied (alias) styles
\tikzset{%
  a/.style={above=#1},
  b/.style={below=#1},
  c/.style={centered=#1},
  l/.style={left=#1},
  r/.style={right=#1},
  al/.style={above left=#1},
  ar/.style={above right=#1},
  bl/.style={below left=#1},
  br/.style={below right=#1},
}
```

You can use these abbreviations to place Ti*k*Z's *main nodes* but not *label nodes*. To place label nodes, use non-abbreviated options or angles instead.

The `tzplot` package also defines graphic layers as follows:

```
\pgfdeclarelayer{background}
\pgfdeclarelayer{behind}
\pgfdeclarelayer{above}
\pgfdeclarelayer{foreground}

\pgfsetlayers{background,behind,main,above,foreground}
```

Therefore, you can select the graphic layers in sequence: `background`, `behind`, `main`, `above`, and `foreground`. For example, you can change the layer of a straight line from `main` (default) to `background` as follows:

```
\begin{tikzpicture}
\tzhelplines(4,3)
  <tzplot macros>
  <tikz macros>
\begin{pgfonlayer}{background}
  \tzline[blue](0,0)(3,1)
\end{pgfonlayer}
\end{tikzpicture}
```

1.3 How to read this document

In drawing graphs, too many factors are involved: line style, color, fill, label, positioning, shift, and so on. Almost all macros of this package have many arguments that control these factors. Some are mandatory and some are optional. Optional arguments are hidden when not used.

The document has three essential parts: Part II, Part III, and Part IV. Part II introduces essential macros with only frequently used options. There are many options hidden in the macros introduced in Part II. Some macros are not introduced in Part II. Part III and IV describe all the features of all macros.

You must get started with Part II. Part II is sufficient for drawing needs in most cases.

Unless you are an experienced user of Ti*k*Z, it is recommended to move on to Part III and Part IV once you become familiar with Part II. In the meantime, use Part III and Part IV for reference only. Use the list of contents and the index efficiently to find macros you need.

Part II

Getting Started

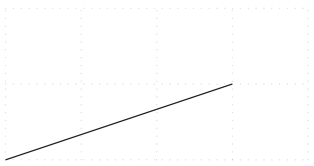
2 An Intuitive Introduction I: Basics

All drawing macros provided in this package work within `tikzpicture` environment, just like any other Ti*k*Z commands.

2.1 Lines: Basics: `\tzline(0,0)(3,1)`

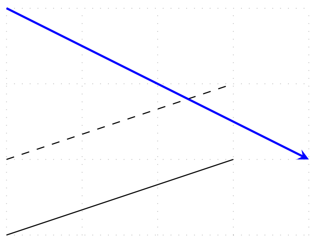
To draw a line from $(0,0)$ to $(3,1)$, just do `\tzline(0,0)(3,1)`.

```
% \tzline
\begin{tikzpicture}
\tzhelplines(4,2)
\tzline(0,0)(3,1)
\end{tikzpicture}
```



You can use Ti*k*Z options to change the style of a line.

```
% \tzline: change line style
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline(0,0)(3,1)
\tzline[dashed](0,1)(3,2)
\tzline[->,blue,thick](0,3)(4,1)
\end{tikzpicture}
```



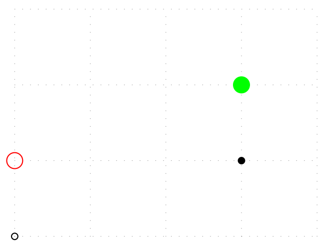
```
\tzline[dashed](0,1)(3,2) % works like:
  \draw [dashed] (0,1) -- (3,2);
```

2.2 Dots: Basics

2.2.1 A circle dot: `\tzcdot(0,0)`

`\tzcdot(0,0)` prints a 'circle dot' ∘, with the *radius* `1.2pt` by default, at the point `(0,0)`. The starred version `\tzcdot*` prints a filled dot •.

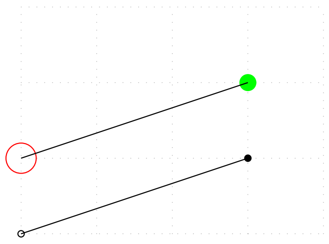
```
% \tzcdot: circle dot
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdot(0,0)
\tzcdot*(3,1)
\tzcdot [red]  (0,1)(3pt) % radius=3pt
\tzcdot*[green](3,2)(3pt) % radius=3pt
\end{tikzpicture}
```



You can change of the size of a dot by specifying the *radius* of the circle, like, for example, `\tzcdot(0,0)(3pt)`.

```
\tzcdot*[green](3,2)(3pt) % is an abbreviation for:
  \draw [fill,green] (0,0) circle (3pt);
```

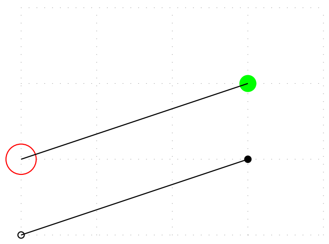
```
% \tzcdot
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdot(0,0)               \tzcdot*(3,1)
\tzcdot[red](0,1)(2mm)     % radius=2mm
\tzcdot*[green](3,2)(3pt)  % radius=3pt
\tzline(0,0)(3,1)
\tzline(0,1)(3,2)
\end{tikzpicture}
```



2.2.2 A circle node dot: `\tzdot(0,0)`

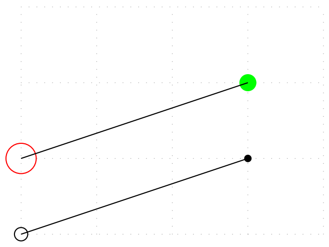
`\tzdot` draws a 'circle node dot' at a specified coordinate. The starred version `\tzdot*` prints a filled dot. The default size (*diameter* or `minimum size`) is `2.4pt`.

```
% \tzdot: node dot
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdot(0,0)                \tzdot*(3,1)
\tzdot[red](0,1)(4mm)      % minimum size=4mm
\tzdot*[green](3,2)(6pt)   % minimum size=6pt
\tzline(0,0)(3,1)
\tzline(0,1)(3,2)
\end{tikzpicture}
```



The size (*diameter* or `minimum size`) of a node dot can be changed by the second (or the last) parenthesis option, like `(6pt)`.

```
% \tzcdot
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdot(0,0)(5pt)          \tzdot*(3,1)
\tzdot[red](0,1)(4mm)     \tzdot*[green](3,2)(6pt)
\tzline(0,0)(3,1)
\tzline(0,1)(3,2)
\end{tikzpicture}
```

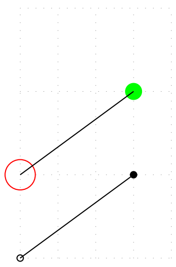


```
\tzdot[green](0,0)(6pt) % works like:
  \draw [green] (0,0) node [tzdot,minimum size=6pt] {};
  % tzdot style is predefined
```

2.2.3 Difference between `\tzcdot` and `\tzdot`

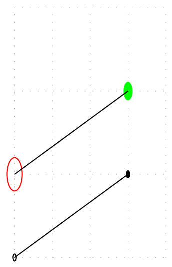
A 'circle dot' drawn by `\tzcdot` is affected by `xscale` or `yscale` in Ti*k*Z, but a 'circle node dot' drawn by `\tzdot` is not. Note also that `\tzcdot` controls the *radius* of a circle dot (following Ti*k*Z practice), while `\tzdot` controls the *diameter* of a circle node dot.

```
% \tzdot: size not changed by scaling
\begin{tikzpicture}[xscale=.5,yscale=1.1] %%
\tzhelplines(4,3)
\tzdot(0,0)                 \tzdot*(3,1)
\tzdot[red](0,1)(4mm)       \tzdot*[green](3,2)(6pt)
\tzline(0,0)(3,1)
\tzline(0,1)(3,2)
\end{tikzpicture}
```



Remark: The circle dots drawn by `\tzcdot` are affected by Ti*k*Z `scale` factors. It gets bigger or smaller by `scale`. Let us see what happens to circle dots, especially when `xscale` and `yscale` are not symmetric.

```
% \tzcdot: distorted
\begin{tikzpicture}[xscale=.5,yscale=1.1] %%
\tzhelplines(4,3)
\tzcdot(0,0)                 \tzcdot*(3,1)
\tzcdot[red](0,1)(2mm)       \tzcdot*[green](3,2)(3pt)
\tzline(0,0)(3,1)
\tzline(0,1)(3,2)
\end{tikzpicture}
```

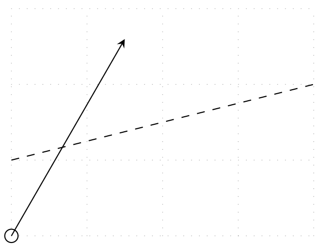


2.3 Coordinates: Basics: `\tzcoor(0,0)(A)`

To define a coordinate, use `\tzcoor` with a coordinate followed by its name in parentheses.

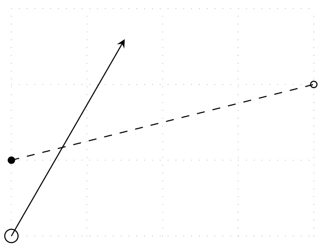
```
\tzcoor(0,0)(A) % works like:
  \path (0,0) coordinate (A);
  % or
  \coordinate (A) at (0,0);
```

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(0,0)(A)       \tzcoor(60:3cm)(B)
\tzline[->](A)(B)
\tzdot(A)(5pt)
\tzcoor(0,1)(C)       \tzcoor(4,2)(D)
\tzline[dashed](C)(D)
\end{tikzpicture}
```



The starred version `\tzcoor*` prints a filled node dot at a specified coordinate.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(0,0)(A)       \tzcoor(60:3cm)(B)
\tzline[->](A)(B)
\tzdot(A)(5pt)
\tzcoor*(0,1)(C)      \tzcoor*[fill=none](4,2)(D)
\tzline[dashed](C)(D)
\end{tikzpicture}
```

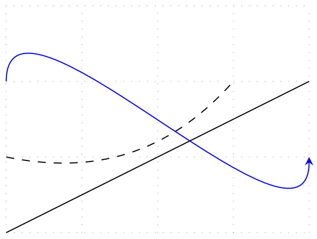


2.4 Curves: Basics

2.4.1 \tzto(0,0)(4,2)

\tzto connects two points with a line or a curve.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzto(0,0)(4,2)
\tzto[bend right,dashed](0,1)(3,2)
\tzto[out=90,in=-90,->,blue](0,2)(4,1)
\end{tikzpicture}
```

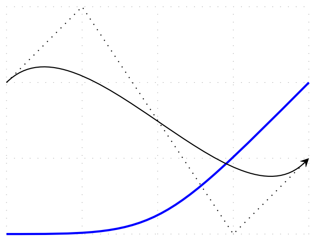


```
\tzto[bend right](0,1)(3,2) % works like:
  \draw [bend right] (0,1) to (3,2);
```

2.4.2 \tzbezier

\tzbezier draws a Bézier curve with one or two control points. The style tzshowcontrols, which is predefined in the package, reveals the control point(s).

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzbezier[blue,thick](0,0)(2,0)(4,2)
\tzbezier[->,tzshowcontrols](0,2)(1,3)(3,0)(4,1)
\end{tikzpicture}
```



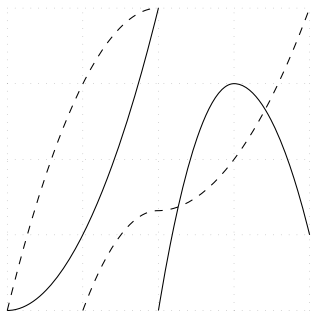
```
\tzbezier[blue](0,0)(2,0)(4,2)        % works like:
  \draw [blue] (0,0) .. controls (2,0) .. (4,2);
\tzbezier(0,2)(1,3)(3,0)(4,1)         % works like:
  \draw (0,2) .. controls (1,3) and (3,0) .. (4,1);
```

2.4.3 \tzparabola

\tzparabola draws a parabola controlled by several options of Ti*k*Z's parabola operation.

```
\tzparabola(0,0)(2,4)       % works like:
  \draw (0,0) parabola (2,4);
\tzparabola(2,0)(3,3)(4,1) % works like:
  \draw (2,0) parabola bend (3,3) (4,1);
```

```
\begin{tikzpicture}
\tzhelplines(4,4)
\tzparabola(0,0)(2,4)
\tzparabola[bend at end,dashed](0,0)(2,4)
\tzparabola(2,0)(3,3)(4,1)
\tzparabola[bend pos=.33,dashed](1,0)(4,4)
\end{tikzpicture}
```

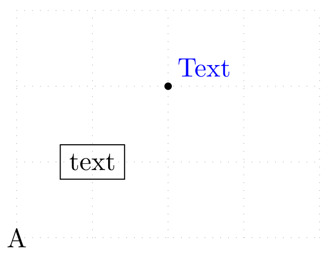


2.5 Adding text: Nodes and placement

2.5.1 \tznode(3,1){text}[right]

With `\tznode(<coor>){<text>}[<node opt>]`, you can put some text at a specified position. The starred version `\tznode*` draws the node perimeter, which is a `rectangle` by default.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tznode(0,0){A}
\tznode*(1,1){text}
\tzdot*(2,2)
\tznode(2,2){Text}[above right,blue]
\end{tikzpicture}
```



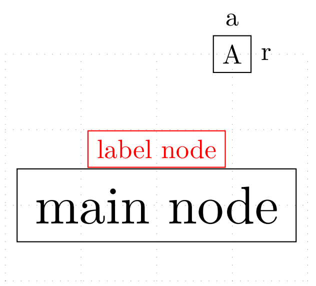
```
\tznode(0,0){A}                       % works like:
  \node at (0,0) {A};
\tznode(2,2){Text}[above right,blue]  % works like:
  \draw (2,2) node [above right,blue] {Text};
```

2.5.2 Review: Main nodes and label nodes in Ti*k*Z

In Ti*k*Z, there are two kinds of nodes: main nodes and label nodes.

When a *main node* with text in it is placed at a specific point, its *label node* with a label in it is optionally placed in the *direction* of a designated *angle* relative to the main node.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tznode*(2,1){main node}
       [scale=2,label={[draw,red]90:label node}]
\tznode*(3,3){A}[label={above:a},label={0:r}]
\end{tikzpicture}
```



The angles for label nodes can be replaced by the corresponding placement words. For example, the angle 90 degree for a label node can be replaced by `above`, `0` by `right`, `-135` by `below left`, and the like. Note that the angle expression *cannot* be used for placing a main node.

Remark:

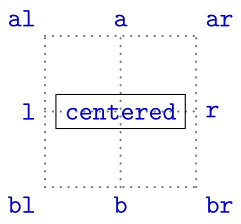
- In this package, macros related to 'dots' or 'coordinates' (like `\tzcdot`, `tzdot`, and `\tzcoor`) can optionally have 'label nodes,' while macros related to 'lines' and 'curves' (like `\tzline`, `\tzto`, `\tzbezier` and `\tzparabola`) optionally have 'main nodes' or 'text nodes.'

- There is one exception: `\tzshoworigin`. `\tzshoworigin` can have a label, but its location is controlled by positional words such as `below left` but not by `<angle>`. (See Section 17.3 on page 102 for more details.)

2.5.3 Abbreviations of Ti*k*Z basic placement options: `a`, `b`, `l`, `r`, `ar`, `bl`, etc.

You can use abbreviations `a` for `above`, `r` for `right`, `bl` for `below left`, and so on to place *main nodes*.

```
\begin{tikzpicture}[font=\ttfamily,text=blue]
\tzhelplines[thick](2,2)
\tznode(1,1){centered}[draw]  % default: centered
\tznode(1,2){a} [a]     \tznode(1,0){b} [b]
\tznode(0,1){l} [l]     \tznode(2,1){r} [r]
\tznode(0,2){al}[al]    \tznode(2,2){ar}[ar]
\tznode(0,0){bl}[bl]    \tznode(2,0){br}[br]
\end{tikzpicture}
```



Warning: The abbreviations of Ti*k*Z basic placement options *cannot* be used to place *labels*. To place labels, use *angles* or the unabridged placement options of Ti*k*Z such as `above` and `below left`.

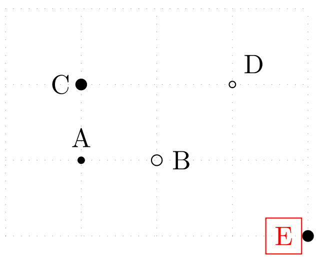
2.6 Labeling dots and coordinates

2.6.1 `\tzdot`, `\tzcdot`

To add a label to a dot generated by `\tzdot` or `\tzcdot`, you should specify, *right after a coordinate*, `{<label>}` followed by `[<angle>]` (90 degree or `above` by default in Ti*k*Z).

REMEMBER that the order of the arguments is `(<coor>){<label>}[<angle>]`. To change the size of a dot, you need to specify the last option `(<dimension>)` after all the other arguments.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdot*(1,1){A}  % default: 90 or above
\tzdot(2,1){B}[0](4pt)
\tzcdot*(1,2){C}[180](2pt)
\tzcdot(3,2){D}[45]
\tzdot*(4,0){E}[[red,draw]180](4pt)
\end{tikzpicture}
```



```
\tzcdot*(1,2){C}[180](2pt) % works like:
  \draw[fill] (1,2) circle (2pt) node [label={180:C}] {};
```

2.6.2 `\tzcoor`

`\tzcoor` can add a label to a coordinate. Just append the optional arguments `{<label>}` and `[<angle>]` after the two mandatory parenthesis arguments.

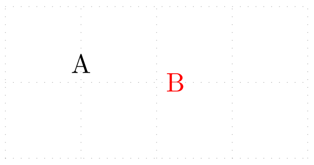
Remark:

- REMEMBER the order of arguments is `(<coor>)(<name>){<label>}[<angle>]`.
- It cannot be emphasized that `{<label>}` is placed by `[<angle>]` (by default `90` or `above`) and you *cannot* use the abbreviations such as `a`, `l`, `ar`, and so forth to place labels for coordinates and dots.

```
% syntax: simplified
\tzcoor(<coor>)(<coor name>){<label>}[[<label opt>]<angle>]
% defaults
  (<m>)(<m>){}[]
% <m> means `mandatory'
```

You can see the full syntax of `\tzcoor` in Section 8.1 on page 47.

```
\begin{tikzpicture}
\tzhelplines(4,2)
\tzcoor(1,1)(A){A}  % default: 90 or above
\tzcoor(2,1)(B){B}[[red]0]
\end{tikzpicture}
```



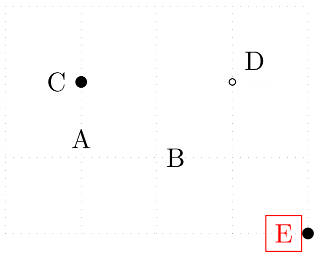
```
\tzcoor(1,1)(B){B}[0]  % works like:
  \draw (1,1) coordinate (B) node [label={0:B}] {};
\tzcoor(1,1)(B){B}[[red]0]  % works like:
  \draw (1,1) coordinate (B) node [label={[red]0:B}] {};
```

2.6.3 `\tzcoor*`

The starred version `\tzcoor*` designates a coordinate and prints a node dot with a label around the designated point like `\tzdot*` does.

```
% syntax: simflified
\tzcoor*[<dot opt>](<coor>)(<coor name>){<label>}[[<label opt>]<angle>](<dot size>)
```

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(1,1)(A){A}  % default: 90 or above
\tzcoor(2,1)(B){B}[0]
\tzcoor*(1,2)(C){C}[180](4pt)
\tzcoor*[fill=none](3,2)(D){D}[45]
\tzcoor*(4,0)(E){E}[[red,draw]180](4pt)
\end{tikzpicture}
```



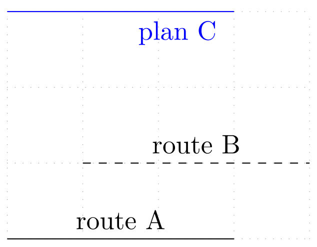
```
\tzcoor*(1,2)(C){C}[180](4pt)  % works as follows:
  \tzcoor(1,2)(C)
  \tzdot*(C){C}[180](4pt)
```

2.7 Adding text next to lines or curves

2.7.1 `\tzline`

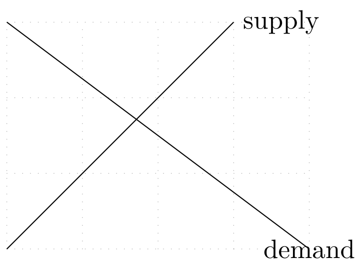
`\tzline` accepts two mandatory coordinates. To add text to a line segment, just specify the optional arguments `{<text>}` and `[<node opt>]` *in-between* the two coordinates. The `[<node opt>]` is `[above,midway]`, by default.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline(0,0){route A}(3,0)
\tzline[dashed](1,1){route B}(4,1)
\tzline[blue]
       (0,3) {plan C} [below,near end] (3,3)
\end{tikzpicture}
```



The optional argument `{<text>}` *following the second coordinate* can also be used as a name of the graph. By default, it is placed at the second coordinate.

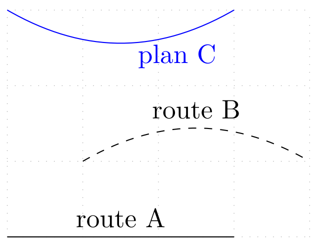
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline(0,3)(4,0){demand}
\tzline(0,0)(3,3){supply}[r]
\end{tikzpicture}
```



2.7.2 `\tzto`

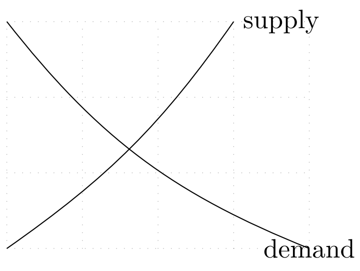
To add text to a line or a curve drawn by `\tzto`, just specify the optional arguments `{<text>}` and `[<node opt>]` *in-between* the two coordinates. By default, the `[<node opt>]` is `[above,midway]`.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzto(0,0){route A}(3,0)
\tzto[dashed,bend left](1,1){route B}(4,1)
\tzto[blue,bend right]
     (0,3) {plan C} [below,near end] (3,3)
\end{tikzpicture}
```



The optional argument `{<text>}` *following the second coordinate* can also be used as a name of the graph. By default, it is placed at the second coordinate.

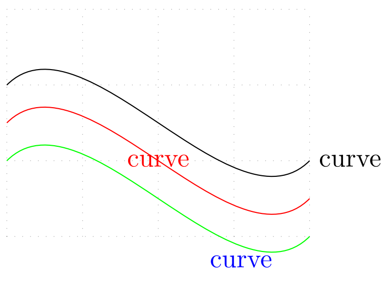
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzto[bend right=15](0,3)(4,0){demand}
\tzto[bend right=10](0,0)(3,3){supply}[r]
\end{tikzpicture}
```



2.7.3 `\tzbezier`

`\tzbezier` accepts three or four coordinates as arguments. You can add text to the curve drawn by `\tzbezier` using the optional arguments `{<text>}` and `[<node opt>]` after the last coordinate.

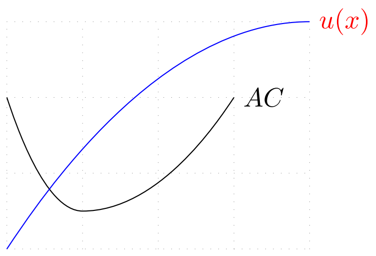
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzbezier(0,2)(1,3)(3,0)(4,1){curve}[r]
\tzbezier[red,yshift=-5mm]
         (0,2)(1,3)(3,0)(4,1){curve}[midway]
\tzbezier[green,text=blue,yshift=-10mm]
         (0,2)(1,3)(3,0)(4,1){curve}[b,near end]
\end{tikzpicture}
```



2.7.4 `\tzparabola`

`\tzparabola` accepts two or three coordinate arguments. You can add text to a parabola drawn by `\tzparabola` using the optional arguments `{<text>}` and `[<node opt>]` *following the last coordinate*. The text is placed at (by default) or around the last coordinate according to `[<node opt>]`.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola(0,2)(1,.5)(3,2){$AC$}[r]
\tzparabola[bend at end,blue](0,0)(4,3){$u(x)$}[r,red]
\end{tikzpicture}
```



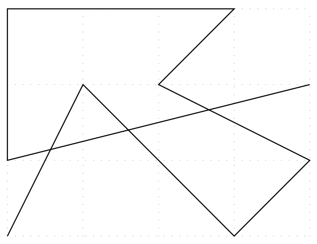
3 An Intuitive Introduction II: Repetition of Coordinates

3.1 Linking many coordinates: Semicolon versions

3.1.1 `\tzlines`: Connected line segments

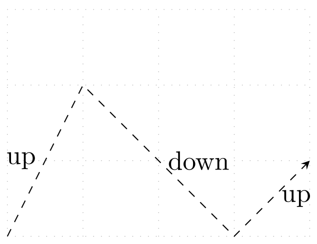
`\tzlines` connects with line segments an arbitrary number of coordinates. The coordinate iteration must end with semicolon `;`. Here, the *semicolon* '`;`' indicates *the end of repetition* of coordinates. Let us call this kind of macro a *semicolon version* macro.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines(0,0)(1,2)(3,0)(4,1)(2,2)(3,3)(0,3)(0,1)(4,2);
\end{tikzpicture}
```



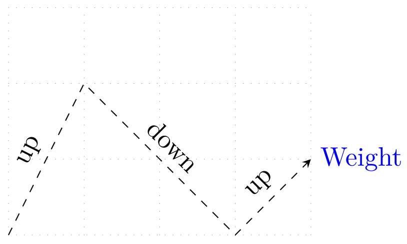
With the optional argument `{<text>}` followed by `[<node opt>]` *in-between two coordinates*, you can print `{<text>}` at or around the middle point of the corresponding line segment in accordance with `[<node opt>]` (by default `[midway]`).

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines[dashed,->](0,0){up}[l]
                  (1,2){down}[r]
                  (3,0){up}[r]
                  (4,1);
\end{tikzpicture}
```



The optional argument `{<text>}` *following the last coordinate* can be used as a name of the whole connected line segments. The `{<text>}` is placed at (by default) or around the last coordinate according to `[<node opt>]`.

```
\begin{tikzpicture}[sloped,auto]
\tzhelplines(4,3)
\tzlines[dashed,->](0,0){up}
                   (1,2){down}
                   (3,0){up}
                   (4,1){Weight}[r,blue] ;
\end{tikzpicture}
```



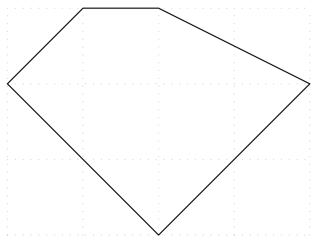
REMEMBER the repeating pattern is the triple `(<coor>){<text>}[<node opt>]` in that order. DO NOT FORGET to indicate when the repetition ends by typing a semicolon. So it will look like `(){}[]..repeated..(){}[];`.

3.1.2 `\tzpolygon`, `\tzpolygon*`: Closed paths

`\tzpolygon` draws closed line segments. `\tzpolygon` is also one of semicolon versions, meaning that it has to end with a semicolon `;`. In fact, `\tzpolygon` is a closed version of `\tzlines`.

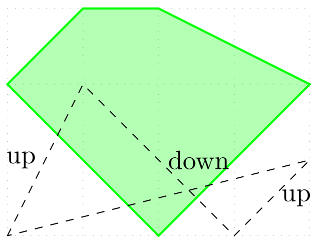
The starred version `\tzpolygon*` does the same thing as `\tzpolgon` except for one thing. `\tzplygon*`, by default, fills the interior of the polygon with `black!50` with `fill opacity=.3` but with `text opacity=1`. (Changing the `fill opacity` is not an issue in this introduction. See Section 15.3 on page 88 for more details.)

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpolygon(1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\end{tikzpicture}
```



The optional arguments `{<text>}` and `[<node opt>]` *in-between* two coordinates prints `<text>` according to `[<node opt>]` (by default `[midway]`) around the middle point of the corresponding line segment.

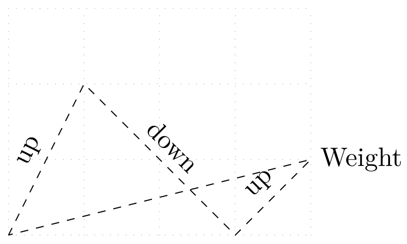
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpolygon*[green,thick](1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\tzpolygon[dashed]
  (0,0){up}[l] (1,2){down}[r] (3,0){up}[r] (4,1);
\end{tikzpicture}
```



The options `{<text>}` and `[<node opt>]` following the last coordinate can be used as a name of the connected line segments.

The entire repetition will look like `(<coor>){<text>}[<node opt>] ..repeated.. (){}[];`. DO NOT FORGET to indicate when the repetition ends by typing a semicolon.

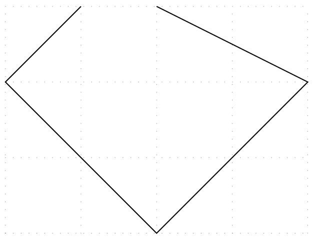
```
\begin{tikzpicture}[sloped,auto]
\tzhelplines(4,3)
\tzpolygon[dashed]
  (0,0){up} (1,2){down} (3,0){up} (4,1){Weight}[r] ;
\end{tikzpicture}
```



3.1.3 \tzpath*: Filling area

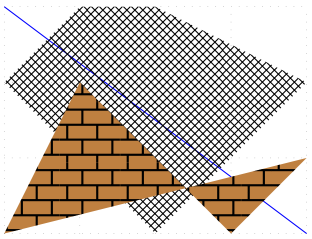
\tzpath accepts an arbitrary number of coordinates to form a path, like \tzlines does, but the path is invisible. This is a *semicolon version* macro, so the coordinate iteration must be ended by a semicolon ;. With [draw] option you can visualize the invisible path.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpath[draw](1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\end{tikzpicture}
```

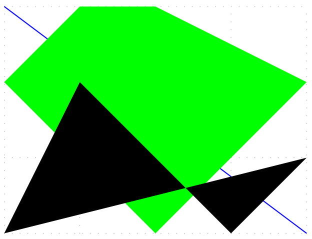


You can fill the interior of a path formed by \tzpath (after being closed) with color or pattern, in usual Ti*k*Z way.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue](0,3)(4,0)
\tzpath[pattern=crosshatch]
       (1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\tzpath[pattern=bricks,preaction={fill=brown}]
       (0,0) (1,2) (3,0) (4,1);
\end{tikzpicture}
```

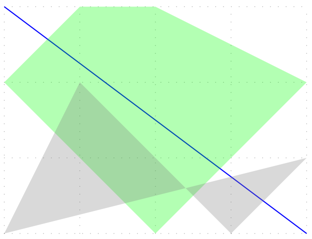


```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue](0,3)(4,0)
\tzpath[fill,green](1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\tzpath[fill](0,0) (1,2) (3,0) (4,1);
\end{tikzpicture}
```



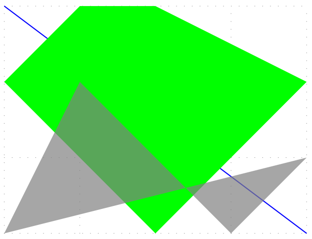
The starred version \tzpath* takes the default options fill=black!50, fill opacity=.3 to fill the area.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue](0,3)(4,0)
\tzpath*[green](1,3)(0,2)(2,0)(3,1)(4,2)(2,3);
\tzpath*(0,0) (1,2) (3,0) (4,1);
\end{tikzpicture}
```



How to change the fill opacity with \tzpath* is not discussed in this introduction, but one example is given below. (See Section 13.2 on page 80 for more details.)

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue](0,3)(4,0)
\tzpath*[green](1,3)(0,2)(2,0)(3,1)(4,2)(2,3);{1} %%
\tzpath*(0,0) (1,2) (3,0) (4,1);{.7}             %%
\end{tikzpicture}
```

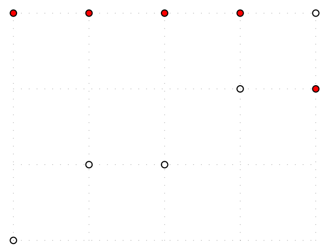


3.2 Many dots: Semicolon versions

3.2.1 `\tzcdots(*)`

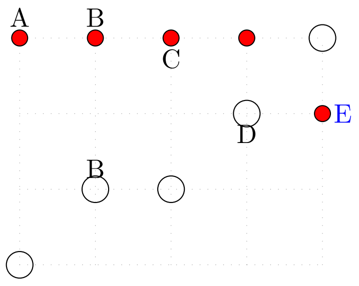
`\tzcdots` accepts an arbitrary number of coordinates to print circle dots, but the coordinate repetition must be ended by `;` (semicolon version). `\tzcdots*` prints filled circle dots.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdots(0,0)(1,1)(2,1)(3,2)(4,3);
\tzcdots*[fill=red](0,3)(1,3)(2,3)(3,3)(4,2);
\end{tikzpicture}
```



Each coordinate can be labeled by specifying the optional argument `{<label>}` followed by `[<angle>]`. You can also change the size (*radius*) of the dots by specifying the *last* parenthesis option `(<dot radius>)` *after* the semicolon.

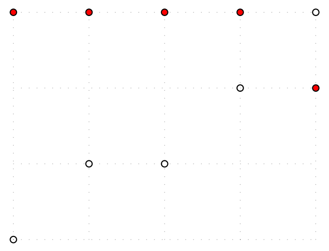
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdots(0,0)(1,1){B}(2,1)(3,2){D}[-90](4,3);(5pt)
\tzcdots*[fill=red](0,3){A}
                    (1,3){B}
                    (2,3){C}[-90]
                    (3,3)
                    (4,2){E}[[blue]0];(3pt) % radius
\end{tikzpicture}
```



3.2.2 `\tzdots(*)`

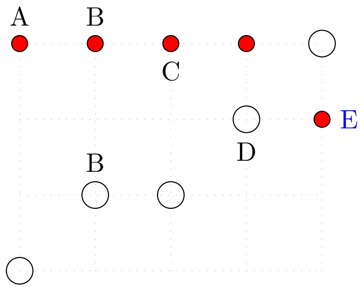
`\tzdots` accepts an arbitrary number of coordinates to print circle node dots, but the repetition must be ended by `;` (semicolon version). `\tzdots*` prints filled circle node dots.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdots(0,0)(1,1)(2,1)(3,2)(4,3);
\tzdots*[fill=red](0,3)(1,3)(2,3)(3,3)(4,2);
\end{tikzpicture}
```



Each coordinate can be labeled by specifying the optional argument `{<label>}` followed by `[<angle>]`. You can also change the size (*diameter*) of the dots by specifying the *last* parenthesis argument `(<dot size>)` *after* the semicolon.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdots(0,0)(1,1){B}(2,1)(3,2){D}[-90](4,3);(10pt)
\tzdots*[fill=red](0,3){A}
                  (1,3){B}
                  (2,3){C}[-90]
                  (3,3)
                  (4,2){E}[[blue]0];(6pt) % diameter
\end{tikzpicture}
```

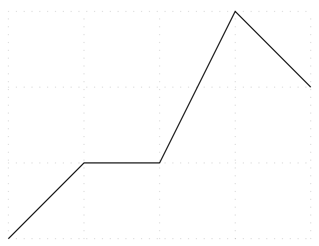


3.3 Many coordinates: Semicolon versions

3.3.1 `\tzcoors`, `\tzcoors*`

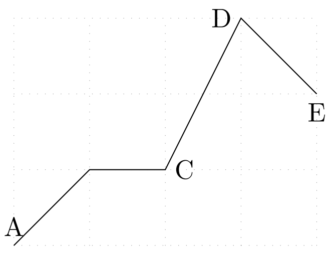
`\tzcoor` accepts *a pair* of mandatory arguments in parentheses: `(<coor>)(<name>)`. `\tzcoors` accepts *an arbitrary number of the pairs* to define multiple coordinates. It is a semicolon version, meaning that a semicolon is necessary to indicate when the coordinate repetition ends.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoors(0,0)(A)
        (1,1)(B)
        (2,1)(C)
        (3,3)(D)
        (4,2)(E);
\tzlines(A)(B)(C)(D)(E);
\end{tikzpicture}
```



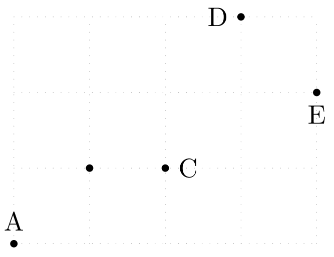
The options `{<label>}` and `[<angle>]` following each pair of `(<coor>)` and `(<name>)` allow you to put `<label>` in the direction of `<angle>` around the coordinate. Here, the repeating pattern is the quadruple `(<coor>)(<name>){<label>}[<angle>]`. The first two parenthesis arguments are mandatory and others are optional. The pattern is repeated until `\tzcoors` meets a semicolon `;`.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoors(0,0)(A){A}
         (1,1)(B)
         (2,1)(C){C}[0]
         (3,3)(D){D}[180]
         (4,2)(E){E}[-90];
\tzlines(A)(B)(C)(D)(E);
\end{tikzpicture}
```



The starred version `\tzcoors*` does one more thing: prints node dots.

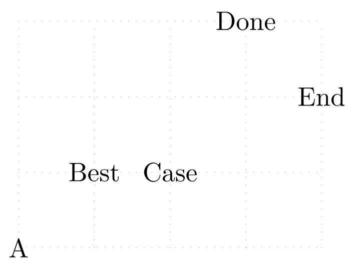
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoors*(0,0)(A){A}
         (1,1)(B)
         (2,1)(C){C}[0]
         (3,3)(D){D}[180]
         (4,2)(E){E}[-90];
\end{tikzpicture}
```



3.3.2 `\tzcoorsquick`

`\tzcoorsquick` is just to see the array of many coordinates at a glance. `\tzcoorsquick` works like `\tzcoors`, but it automatically prints the name of each coordinate as its label, right at the point, by default.

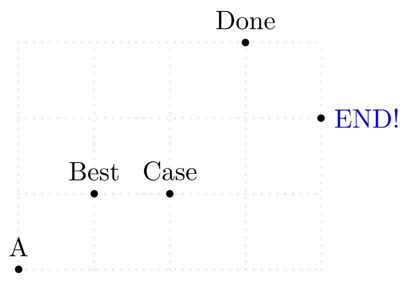
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoorsquick(0,0)(A)
             (1,1)(Best)
             (2,1)(Case)
             (3,3)(Done)
             (4,2)(End);
\end{tikzpicture}
```



3.3.3 \tzcoorsquick*

The starred version `\tzcoorsquick*` prints node dots and automatically puts the labels `above` (in the direction of `90` degree from) them, by default.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoorsquick*(0,0)(A)
              (1,1)(Best)
              (2,1)(Case)
              (3,3)(Done)
              (4,2)(End){END!}[[blue]0];
\end{tikzpicture}
```

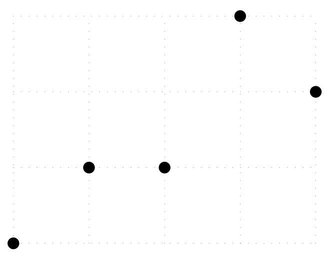


3.4 plot coordinates: Semicolon versions

3.4.1 \tzplot*: Mark dots with [mark=*]

`\tzplot*` accepts an arbitrary number of coordinates to print bullets with the *radius* (`mark size` in TikZ) of `2pt`, which is the initial value in TikZ. The repetition of coordinates must be ended by `;` (semicolon version).

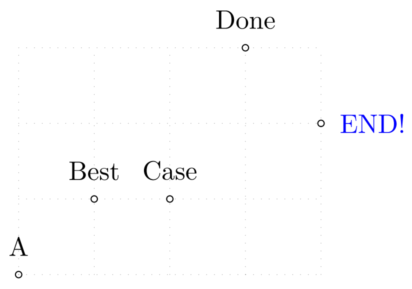
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplot*(0,0)
        (1,1)
        (2,1)
        (3,3)
        (4,2);
\end{tikzpicture}
```



```
\tzplot*(0,0)(1,1)(2,1); % works like:
  \draw [mark=*] plot coordinates {(0,0)(1,1)(2,1)};
```

Each coordinate can be labeled by specifying the optional argument `{<text>}` followed by `[<angle>]`. With the option `[mark=o]` you can print hollow dots. You can also change the *radius* of the marks by specifying the *last* parenthesis argument `(<mark size>)` *after* the semicolon.

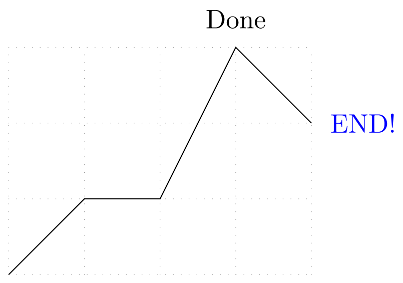
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplot*[mark=o](0,0){A}
                (1,1){Best}
                (2,1){Case}
                (3,3){Done}
                (4,2){END!}[[blue]0];(1.2pt)
\end{tikzpicture}
```



3.4.2 \tzplot: Lines with [tension=0]

`\tzplot` accepts an arbitrary number of coordinates and draws line segments connecting them. The repetition of coordinates must be ended by `;` (semicolon version).

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplot(0,0)
       (1,1)
       (2,1)
       (3,3){Done}
       (4,2){END!}[[blue]0];
\end{tikzpicture}
```

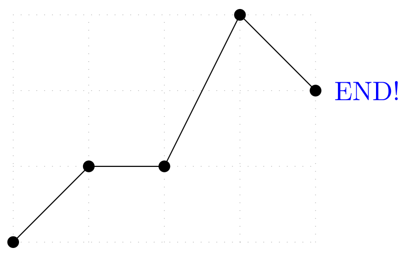


```
\tzplot(0,0)(1,1)(2,1); % works like:
  \draw plot coordinates {(0,0)(1,1)(2,1)};
```

3.4.3 \tzplot*[draw]: Lines with dots

`\tzplot*[draw]` prints bullets and draws line segments connecting specified coordinates. The repetition of coordinates must be ended by `;` (semicolon version).

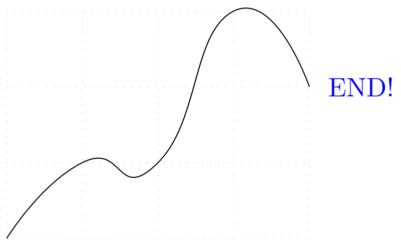
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplot*[draw](0,0)
        (1,1)
        (2,1)
        (3,3)
        (4,2){END!}[[blue]0];
\end{tikzpicture}
```



3.4.4 \tzplotcurve: Curves with [smooth,tension=1]

`\tzplotcurve` plots coordinates with the option `[smooth,tension=1]`, resulting in a curve connecting an arbitrary number of coordinates. The repetition of coordinates must be ended by `;` (semicolon version).

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplotcurve(0,0)
            (1,1)
            (2,1)
            (3,3)
            (4,2){END!}[[blue]0];
\end{tikzpicture}
```

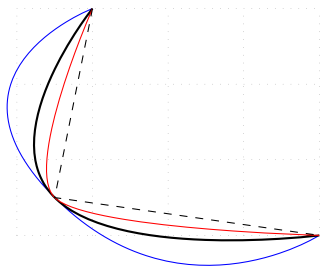


```
% syntax: simplified
\tzplotcurve[<opt>]{<tension>}"<path name>"
            (<coor>){<label>}[<angle>]..repeated..(<coor>){<label>}[<angle>];
% defaults
  [smooth,tension=1]{1}""(<m>){}[]..repeated..(){}[];
% <m> means mandatory
```

```
\tzplotcurve(0,0)(1,1)(2,1); % works like:
  \draw [smooth,tension=1] plot coordinates {(0,0)(1,1)(2,1)};
```

You can change the tension value by specifying the optional argument `{<tension>}`, before the first coordinate.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(.5,.5)(A)
\tzplotcurve[blue]{2}(1,3)(A)(4,0);
\tzplotcurve[thick](1,3)(A)(4,0); % default: tension=1
\tzplotcurve[red]{.5}(1,3)(A)(4,0);
\tzplotcurve[dashed]{0}(1,3)(A)(4,0);
\end{tikzpicture}
```



4 An Intuitive Introduction III: Plotting Functions

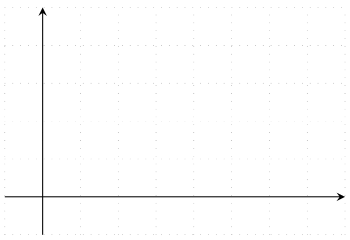
4.1 Axes

4.1.1 \tzaxes

```
% syntax: simplified
\tzaxes[<opt>]<x-shift,y-shift>(<x1,y1>)(<x2,y2>)
                                  {<x-text>}[<x-opt>]{<y-text>}[<y-opt>]
% defaults
  [->]<0,0>(0,0)(<m>){}[right]{}[above]
% <m> means mandatory
```

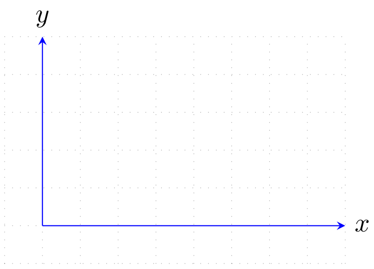
`\tzaxes` draws the x-axis from `<x1>` to `<x2>` and the y-axis from `<y1>` to `<y2>`.

```
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzaxes(-1,-1)(8,5) % basics
\end{tikzpicture}
```



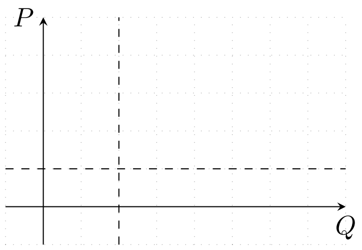
If (<x1,y1>) is omitted, it is regarded as (0,0). And optionally the names of x-axis and y-axis can be printed at a specified place (by default, [right] for x-axis and [above] for y-axis).

```
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzaxes[draw=blue](8,5){$x$}{$y$} %%
\end{tikzpicture}
```



With the optional argument <x-shift,y-shift>, the axes are shifted accordingly. Two axes intersect at (<x-shift,y-shift>), by default (0,0).

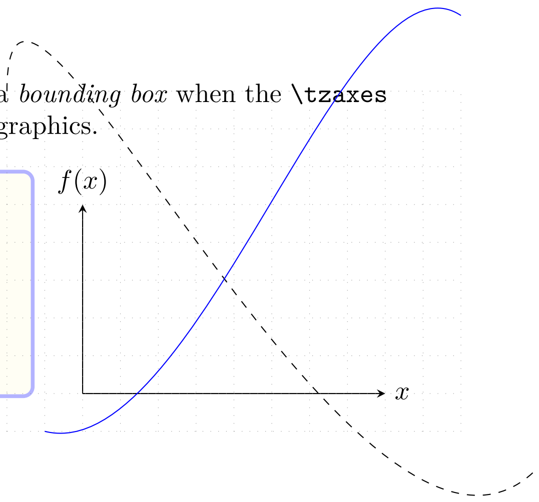
```
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzaxes(-1,-1)(8,5){$Q$}[b]{$P$}[l]
\tzaxes[dashed,-]<2,1>(-1,-1)(8,5) % shift
\end{tikzpicture}
```



4.1.2 \tzaxes*

The starred version \tzaxes* is just to set the current state to a *bounding box* when the \tzaxes macro execution is completed. Use \tzaxes* before any larger graphics.

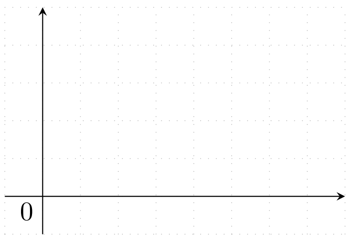
```
\begin{tikzpicture}[scale=.5]
\tzaxes*(8,5){$x$}{$f(x)$} % bounding box
\tzhelplines(-2,-1)(10,8)
\tzto[out=90,in=-135,dashed](-2,8)(12,-2)
\tzbezier[blue](-1,-1)(3,-2)(7,12)(10,10)
\end{tikzpicture}
```



4.1.3 \tzshoworigin, \tzshoworigin*

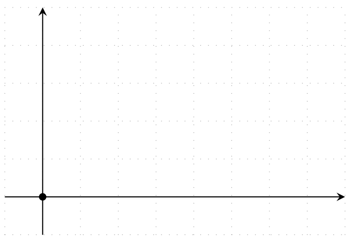
\tzshoworigin prints '0' (roughly) at the bottom left of the origin.

```
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzshoworigin
\tzaxes(-1,-1)(8,5)
\end{tikzpicture}
```



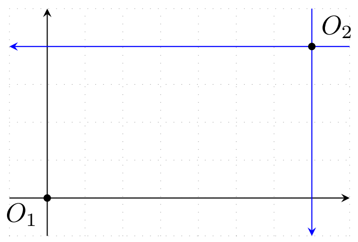
\tzshoworigin* prints a node dot with the size of 2.4pt (by default) at the origin.

```
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzshoworigin*
\tzaxes(-1,-1)(8,5)
\end{tikzpicture}
```



\tzshoworigin*(<coor>){<text>}[<node opt>] prints a node dot and text around (<coor>), by default (0,0). (Notice that the place where the <text> is printed at is not by <angle>. Instead, you can use the *abbreviations* of TikZ basic placement options such as a, l, br, ect. See Section 1.2, for more details.)

```
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzshoworigin*{$O_1$}
\tzaxes(-1,-1)(8,5)
\tzaxes[blue]<7,4>(8,5)(-1,-1)
\tzshoworigin*(7,4){$O_2$}[ar]
\end{tikzpicture}
```



Notice that, in the above example, the two axes intersect at (7,4) by the shift option <7,4>.

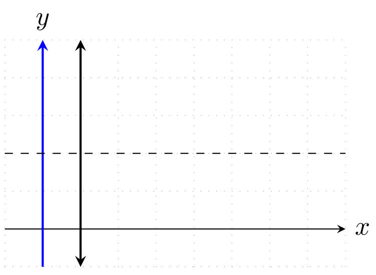
4.1.4 \tzaxisx, \tzaxisy

\tzaxisx and \tzaxisy draw the x-axis and the y-axis, respectively.

```
% syntax: simplified
\tzaxisx[<opt>]<y-shift>{<x1>}{<x2>}{<text>}[<node opt>]
% defaults
  [->,>=stealth]<0>{<m>}{<m>}{}[right]
```

\tzaxisy works similarly for the y-axis.

```
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzaxisx{-1}{8}{$x$}
\tzaxisy[draw=blue,thick]{-1}{5}{$y$}
\tzaxisx[dashed,-]<2>{-1}{8}
\tzaxisy[thick,<->]<1>{-1}{5}
\end{tikzpicture}
```

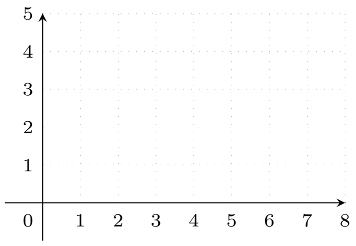


4.2 Ticks

4.2.1 \tzticks

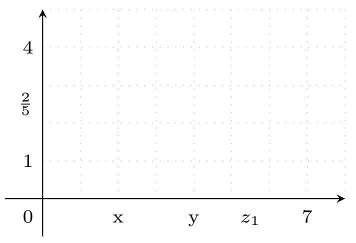
\tzticks{<x-tick places>}{<y-tick places>} prints tick labels for x- and y-axis at specified places, which are comma separated. By default, tick labels are the numbers specified.

```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(8,5)
\tzshoworigin
\tzaxes(-1,-1)(8,5)
\tzticks{1,2,...,8} % x-ticks
        {1,2,...,5} % y-ticks
\end{tikzpicture}
```



You can change the numbered labels, for example `{2,4,7}`, to any other form, by doing like, for example, `{2/mylabel,4,7}`. (Internally, `\tzticks` uses the `foreach` operation of Ti*k*Z.)

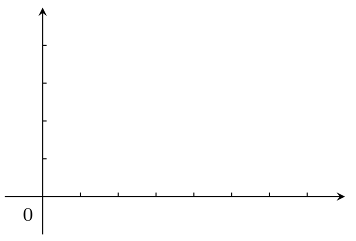
```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(8,5)
\tzshoworigin
\tzaxes(-1,-1)(8,5)
\tzticks{2/x,4/y,5.5/$z_1$,7}
        {1,2.5/$\frac25$,4}
\end{tikzpicture}
```



4.2.2 \tzticks*

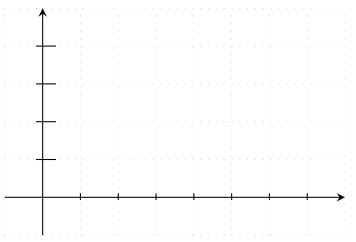
The starred version `\tzticks*` prints tick marks from `0pt` to `3pt` by default, without printing tick labels.

```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(8,5)
\tzshoworigin
\tzaxes(-1,-1)(8,5)
\tzticks*{1,2,...,7}
         {1,2,3,4}
\end{tikzpicture}
```



You can change the length of tick marks, for example, like `(-2pt:3pt)` and `(-5pt:10pt)` as shown in the following example.

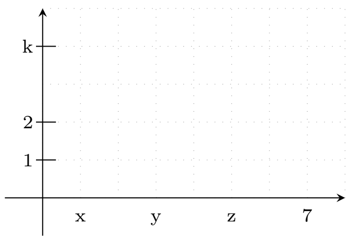
```
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(8,5)
\tzaxes(-1,-1)(8,5)
\tzticks*(-2pt:3pt){1,2,...,7}
         (-5pt:10pt){1,2,3,4}
\end{tikzpicture}
```



4.2.3 \tzticksx(*), \tzticksy(*)

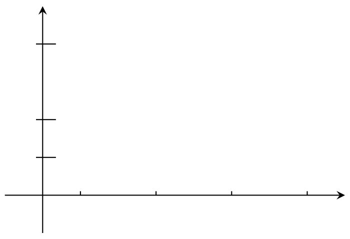
`\tzticksx` and `\tzticksy` prints x-tick labels and y-tick labels, respectively.

```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(8,5)
\tzaxes(-1,-1)(8,5)
\tzticksx{1/x,3/y,5/z,7}
\tzticksy(-5pt:10pt){1,2,4/k}
\end{tikzpicture}
```



`\tzticksx*` and `\tzticksy*` suppress tick labels for their corresponding axes, like `\tzticks*`.

```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(8,5)
\tzaxes(-1,-1)(8,5)
\tzticksx*{1/x,3/y,5/z,7}
\tzticksy*(-5pt:10pt){1,2,4/k}
\end{tikzpicture}
```



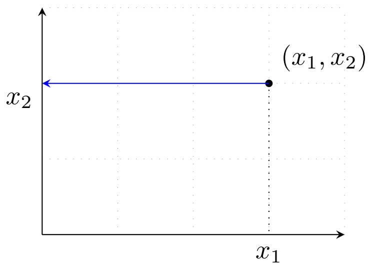
4.3 Projections on the axes

4.3.1 `\tzprojx(*)`, `\tzprojy(*)`

`\tzprojx` draws a dotted (by default) line from a specified coordinate to its projection point on the x-axis and prints text around (`[below]` by default) the projection point.

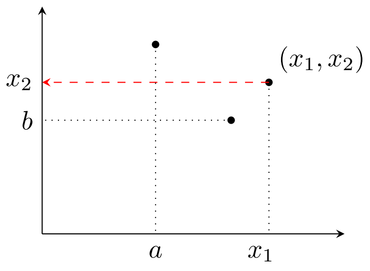
`\tzprojy` works similarly but for the projection point on the y-axis.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzaxes(4,3)
\tzdot*(3,2){$(x_1,x_2)$}[45]
\tzprojx(3,2){$x_1$}
\tzprojy[solid,->,draw=blue](3,2){$x_2$}[bl]
\end{tikzpicture}
```



`\tzprojx*` does one more thing. It prints a node dot at a specified coordinate. `\tzprojy*` works similarly but for the projection point on the y-axis.

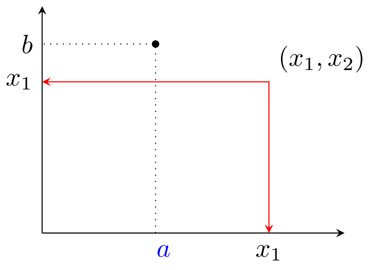
```
\begin{tikzpicture}
%\tzhelplines(4,3)
\tzaxes(4,3)
\tznode(3,2){$(x_1,x_2)$}[ar]
\tzprojx*(3,2){$x_1$}[xshift=-3pt]
\tzprojy[->,dashed,draw=red](3,2){$x_2$}
\tzprojx*(1.5,2.5){$a$}
\tzprojy*(2.5,1.5){$b$}
\end{tikzpicture}
```



4.3.2 `\tzproj(*)`

`\tzproj` combines `\tzprojx` and `\tzprojy`. And `\tzproj*` combines `\tzprojx*` and `\tzprojy*`, so it suppresses tick labels.

```
\begin{tikzpicture}
%\tzhelplines(4,3)
\tzaxes(4,3)
\tznode(3,2){$(x_1,x_2)$}[ar]
\tzproj[<->,solid,draw=red](3,2){$x_1$}{$x_1$}
\tzproj*(1.5,2.5){$a$}[xshift=3pt,text=blue]{$b$}
\end{tikzpicture}
```

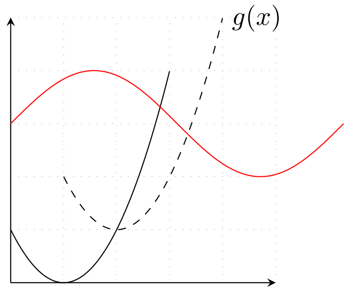


4.4 Plot functions

4.4.1 \tzfn

`\tzfn{<fn of \x>}[<a:b>]` plots a function of x over the specified domain $[a : b]$, which means that $a \leq x \leq b$. Optionally, you can add `{<text>}` with `[<node opt>]` as shown in the following example.

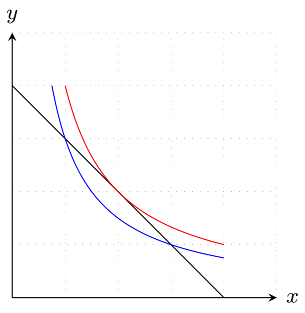
```
\begin{tikzpicture}[scale=.7]
\tzhelplines(5,5)
\tzaxes(5,5)
\tzfn{(\x-1)^2}[0:3]              % y=(x-1)^2
\def\Gx{(\x-2)^2+1}
\tzfn[dashed]{\Gx}[1:4]{$g(x)$}[r] % g(x)=(x-2)^2+1
\tzfn[red]{sin(\x r)+3}[0:2*pi]    % y=sin x
\end{tikzpicture}
```



You can name a path formed by `\tzfn` by specifying the optional argument `"<path name>"` right before the mandatory curly brace argument `{<fn of \x>}`. The name of a path is used to find intersection points.

```
% syntax: simplified
\tzfn[<opt>]"<path name>"{<fn of \x>}[<a:b>]{<text>}[<node dot>]
% defaults
  []""{<m>}[<m>]{}[]
% <m>: mandatory
```

```
\begin{tikzpicture}[scale=.7,font=\footnotesize]
\tzhelplines(5,5)
\tzaxes(5,5){$x$}{$y$}
\def\bgt{4-\x}
\def\IC{3/\x}
\def\ICa{4/\x}
\tzfn"bgt"{\bgt}[0:4]       % name path = bgt
\tzfn[blue]"IC"{\IC}[.75:4] % name path = IC
\tzfn[red]{\ICa}[1:4] % name path = ICa (automatically)
\end{tikzpicture}
```



Remark: If the curly brace mandatory argument consists of *only a macro name* like `{\Foo}`, the macro name `Foo` (without the backslash) is automatically assigned to the *name of the path*. (See Section 20.1.3 on page 111, for more details.)

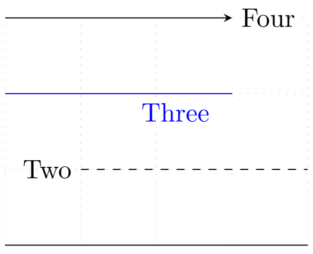
4.4.2 \tzhfnat, \tzhfn: Horizontal lines

`\tzhfnat` draws a horizontal line (the graph of a constant function) at y, by default, from left to right of the `current bounding box`.

```
% syntax
\tzhfnat[<opt>]"<path name>"{<y-val>}[<from:to>]{<text>}[<node opt>]
% defaults
  []""{<m>}[west:east (of current bounding box)]{}[]
```

`\tzhfn(<coor>)` accepts a coordinate, instead of the value of y, to draw a horizontal line at the value of y coordinate of `(<coor>)`, ignoring the value of x coordinate.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzhfnat{0}
\tzhfnat[dashed]{1}[1:4]{Two}[l,at start]
\tzcoors(0,2)(A)(0,3)(B);
\tzhfn[blue](A)[0:3]{Three}[b,near end]
\tzhfn[->](B)[0:3]{Four}[r]
\end{tikzpicture}
```



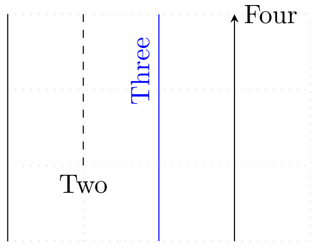
4.4.3 \tzvfnat, \tzvfn: Vertical lines

\tzvfnat draws a vertical line at x from bottom to top of the `current bounding box` by default.

```
% syntax: simplified
\tzvfnat[<opt>]"<path name>"{<x-vale>}[<from:to>]{<text>}[<pos>]
% defaults
  []""{<m>}[south:north (of current bounding box)]{}[]
```

\tzvfn(<coor>) accepts a coordinate, instead of the value of x. It draws a vertical line at the value of x coordinate of (<coor>), ignoring the value of y coordinate.

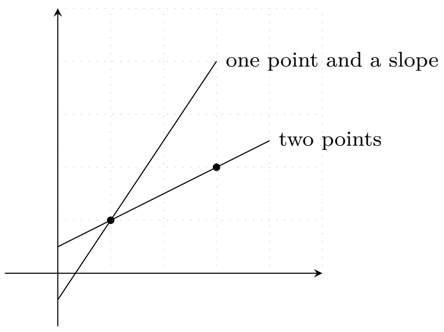
```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzvfnat{0}
\tzvfnat[dashed]{1}[1:3]{Two}[b,at start]
\tzvfn[blue](2,0)[0:3]{Three}[a,near end,sloped]
\tzvfn[->](3,0)[0:3]{Four}[r]
\end{tikzpicture}
```



4.4.4 \tzLFn: Linear functions

\tzLFn(<coor1>)(<coor2>)... draws a *linear function* passing through two points: (<coor1>) and (<coor2>). \tzLFn(<coor1>){<slope>}... draws a linear function passing through one point, (<coor1>), with the slope of <slope>. If both of the two coordinates are specified the option {<slope>} is ignored. The domain in the form of [a:b] is also a required argument.

```
\begin{tikzpicture}[scale=.7,font=\footnotesize]
\tzhelplines(5,5)
\tzaxes(-1,-1)(5,5)
\tzdots*(1,1)(3,2);
\tzLFn(1,1)(3,2)[0:4]{two points}[r]
\tzLFn(1,1){1.5}[0:3]{one point and a slope}[r]
\end{tikzpicture}
```



4.5 Intersection points

4.5.1 Naming paths

In Ti*k*Z, you can find *intersection* points when two *named paths* intersect. The name of a path is usually given by the option [name path=<path name>] in Ti*k*Z.

With the package tzplot, you can name a path by specifying an optional argument within quotation marks such as `"<path name>"`. (Of course, you can also name a path in usual Ti*k*Z way, like `[name path=<path name>]`.)

In this package, all macros (with a few exceptions) related to lines and curves accept this quote optional argument to name paths as follows:

```
\tzline[<opt>]"<path name>"(<coor>)...
\tzlines...    "<path name>"(<coor>)...
\tzpolygon...  "<path name>"(<coor>)...
\tzpath...     "<path name>"(<coor>)...
\tzto...       "<path name>"(<coor>)...
\tzparabola..."<path name>"(<coor>)...
\tzbezier...   "<path name>"(<coor>)...
\tzfn...       "<path name>"{<fn of \x>}...
\tzLFn...      "<path name>"(<coor>)...
and more...
```

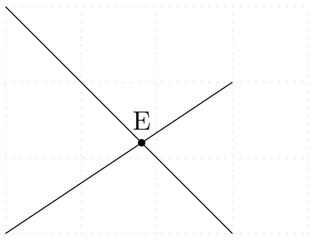
```
\tzline[dashed]"foo"(1,1)(3,3) % works like
  \draw [dashed,name path=foo](1,1) -- (3,3);
```

In most cases, the quote optional arguments for naming paths are placed *immediately before the first mandatory argument* of the `tzplot` macros.

4.5.2 `\tzXpoint(*)`: Intersection points of two paths

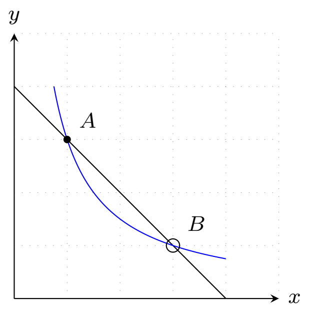
For example, `\tzXpoint{path1}{path2}(A)` finds intersection points of `path1` and `path2` and names the first intersection point `(A)`. This intersection point can be referred to as `(A)` or `(A-1)`. If there are two or more intersection points found, they are called `(A)=(A-1)`, `(A-2)`, `(A-3)`, and so on.

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline"AA"(0,0)(3,2)
\tzline"BB"(0,3)(3,0)
\tzXpoint{AA}{BB}(X)
\tzdot*(X){E}
\end{tikzpicture}
```



`\tzXpoint*` prints a node dot at the first intersection point. You can label the point by specifying `{<text>}` and `[<angle>]` after the specified intersection name.

```
\begin{tikzpicture}[scale=.7,font=\footnotesize]
\tzhelplines(5,5)
\tzaxes(5,5){$x$}{$y$}
\def\bgt{4-\x}
\def\IC{3/\x}
\tzfn"bgt"{\bgt}[0:4]           % name path = bgt
\tzfn[blue]"IC"{\IC}[.75:4]     % name path = IC
\tzXpoint*{bgt}{IC}(E){$A$}[45] % first intersection
\tzdot(E-2){$B$}[45](5pt)       % second intersection
\end{tikzpicture}
```



Remark: You have to expect Ti*k*Z to take a few seconds (or less) to find intersection points.

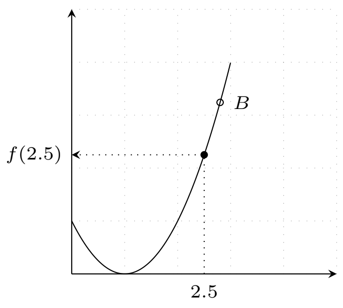
4.5.3 `\tzvXpointat(*)`, `\tzvXpoint(*)`: Vertical intersection points

To find vertical intersection points at x to a curve, you should specify a path name and either the value of x or the coordinate (x, y). Here the y coordinate is ignored.

`\tzvXpointat{<path>}{<x>}(A)` finds vertical intersection points of `<path>` at $x =$ `<x>` and names it `(A)`. The starred version `\tzvXpointat*` additionally prints a node dot at the (first) intersection point.

The macro `\tzvXpoint` uses `(<coor>)`, while `\tzvXpointat` uses the value of x. Here the y coordinate of `(<coor>)` is not important. `\tzvXpoint` is useful when you do not know the exact value of x coordinate of `(<coor>)`. The starred version `\tzvXpoint*` additionally prints a node dot at the (first) intersection point.

```
\begin{tikzpicture}[scale=.7,font=\scriptsize]
\tzhelplines(5,5)
\tzaxes(5,5)
\def\Fx{(\x-1)^2}
\tzfn\Fx[0:3] % name path = Fx (automatically)
\tzvXpointat{Fx}{2.5}(A)
\tzvXpoint{Fx}(2.8,1)(B) % y=1 is ignored
\tzproj*[->](A){$2.5$}{$f(2.5)$}
\tzdot(B){$B$}[0]
\end{tikzpicture}
```

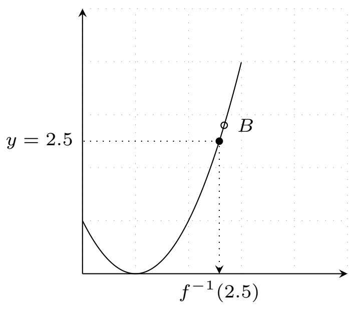


4.5.4 `\tzhXpointat(*)`, `\tzhXpoint(*)`: Horizontal intersection points

`\tzhXpointat{<path>}{<y>}(A)` works like `\tzvXpointat{<path>}{<x>}(A)`, but it uses the value of y instead of x. The starred version `\tzhXpointat*` additionally prints a node dot at the (first) intersection point.

`\tzhXpoint` uses `(<coor>)`, while `\tzhXpointat` uses the value of y. Here the x coordinate of `(<coor>)` is ignored. The starred version `\tzhXpoint*` prints a node dot at the (first) intersection point.

```
\begin{tikzpicture}[scale=.7,font=\scriptsize]
\tzhelplines(5,5)
\tzaxes(5,5)
\def\Fx{(\x-1)^2}
\tzfn\Fx[0:3] % name path = Fx (automatically)
\tzhXpointat{Fx}{2.5}(A)
\tzhXpoint{Fx}(1,2.8)(B) % x=1 is ignored
\tzproj*[<-](A){$f^{-1}(2.5)$}{$y=2.5$}
\tzdot(B){$B$}[0]
\end{tikzpicture}
```



4.6 Tangent lines

4.6.1 `\tztangentat`

`\tztangentat{<path>}{<x>}[<a:b>]` draws a tangent line to `<path>` at $x =$ `<x>` over $x \in [a, b]$. The domain is a mandatory argument and should be of the form `[<from:to>]`.

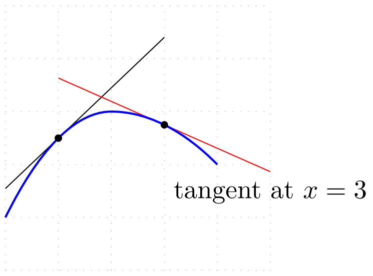
```
% syntax: simplified
\tztangentat{<path>}{<x>}[<domain>]{<text>}[<node opt>]
% defaults
  []{<m>}{<m>}[<m>]{}[]
% <m> means mandatory
```

Remark: The slope of a tangent line drawn by `\tztangentat` is just approximate.

The line is drawn on the `behind` layer, by default.

You can also add some text next to the line by specifying the optional arguments `{<text>}` and `[<node opt>]`, after the domain.

```
\begin{tikzpicture}[scale=.7]
\tzhelplines(5,5)
\tzparabola[thick,blue]"curve"(0,1)(2,3)(4,2)
\tzvXpointat*{curve}{1}
\tzvXpointat*{curve}{3}
\tztangentat{curve}{1}[0:3]
\tztangentat[red]{curve}{3}[1:5]{tangent at $x=3$}[b]
\end{tikzpicture}
```



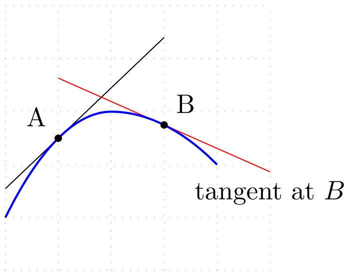
4.6.2 `\tztangent`

`\tztangent` works like `\tztangentat`, but it accepts a coordinate instead of the value of x.

`\tztangent{<path>}(<coor>)` draws a tangent line to `<path>` at the x coordinate of `(<coor>)`. Here, the y coordinate of `(<coor>)` is ignored. The line is drawn on the `behind` layer, by default.

```
% syntax: simplified
\tztangent{<path>}(<coor>)[<from:to>]{<text>}[<node opt>]
% defaults
  []{<m>}(<m>)[<m>]{}[]
```

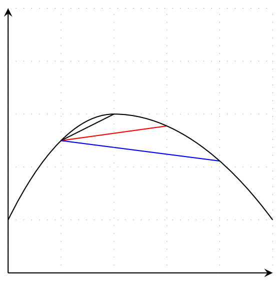
```
\begin{tikzpicture}[scale=.7]
\tzhelplines(5,5)
\tzparabola[thick,blue]"curve"(0,1)(2,3)(4,2)
\tzvXpoint*{curve}(1,0)(A){A}[135]
\tzvXpoint*{curve}(3,0)(B){B}[45]
\tztangent{curve}(A)[0:3]
\tztangent[red]{curve}(B)[1:5]{tangent at $B$}[b]
\end{tikzpicture}
```



4.6.3 `\tzsecantat`, `\tzsecant`

`\tzsecantat{<path>}{<x1>}{<x2>}` draws a secant line segment of `<path>` from x_1 = `<x1>` to x_2 = `<x2>` on the `behind` layer, by default.

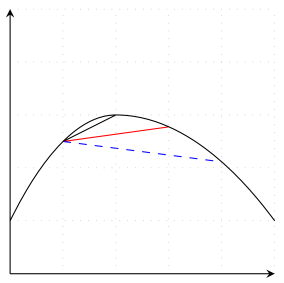
```
\begin{tikzpicture}[scale=.7]
\tzhelplines(5,5)
\tzaxes*(5,5)
\tzparabola"curve"(0,1)(2,3)(5,1)
\tzsecantat{curve}{1}{2}
\tzsecantat[red]{curve}{1}{3}
\tzsecantat[blue]{curve}{1}{4}
\end{tikzpicture}
```



`\tzsecant` works like `\tzsecantat`, but it accepts two coordinates instead of two values of x.

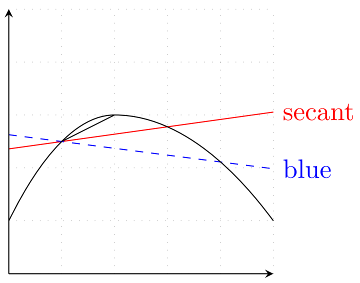
`\tzsecant{<path>}(<coor1>)(<coor2>)` draws a secant line segment of `<path>` from the x coordinate of `(<coor1>)` to the x coordinate of `(<coor2>)`, ignoring y values of the coordinates, on the `behind` layer by default.

```
\begin{tikzpicture}[scale=.7]
\tzhelplines(5,5)
\tzaxes(5,5)
\tzparabola"curve"(0,1)(2,3)(5,1)
\tzsecant{curve}(1,0)(2,0)
\tzsecant[red]{curve}(1,0)(3,0)
\tzsecant[blue,dashed]{curve}(1,0)(4,0)
\end{tikzpicture}
```



You can extend or shorten a secant line by specifying the domain `[<from:to>]`, which is an optional argument. If you specify the domain, `\tzsecant` draws a secant line over the domain. You can also add some text next to the line by specifying the optional arguments `{<text>}` and `[<node opt>]`.

```
\begin{tikzpicture}[scale=.7]
\tzhelplines[use as bounding box](5,5)
\tzaxes(5,5)
\tzparabola"curve"(0,1)(2,3)(5,1)
\tzsecant{curve}(1,0)(2,0)
\tzsecant[red]{curve}(1,0)(3,0)[0:5]{secant}[r]
\tzsecantat[blue,dashed]{curve}{1}{4}[0:5]{blue}[r]
\end{tikzpicture}
```



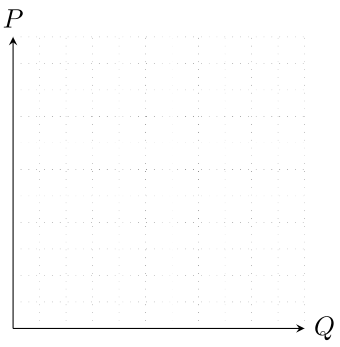
5 Examples: Economics

5.1 Markets

5.1.1 Market equilibrium: step by step

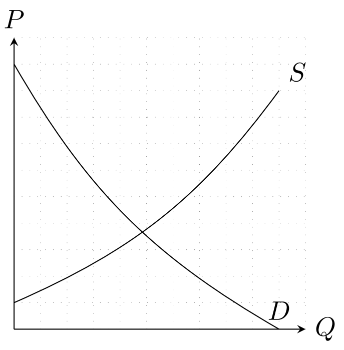
Step 1 Determine the size of the graph.

```
\begin{tikzpicture}[x=0.05cm,y=0.05cm,scale=.7]
% to avoid `dimension too large' error: x/y=0.05cm
\tzhelplines[step=.5cm](110,110)
\tzaxes(110,110){$Q$}{$P$}
%\tzto[bend right=15]"dem"(0,100)(100,0){$D$}[a]
%\tzto[bend right=15]"supp"(0,10)(100,90){$S$}[ar]
%\tzXpoint*{dem}{supp}(eqm){$E$}
%\tzproj(eqm){$Q^*$}{$P^*$}
\end{tikzpicture}
```



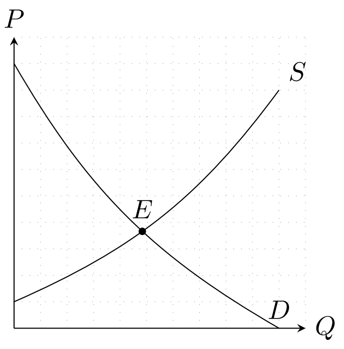
Step 2 Draw the demand and supply curves. Here, we are using `\tzto`.

```
\begin{tikzpicture}[x=0.05cm,y=0.05cm,scale=.7]
% to avoid `dimension too large' error: x/y=0.05cm
\tzhelplines[step=.5cm](110,110)
\tzaxes(110,110){$Q$}{$P$}
\tzto[bend right=15]"dem"(0,100)(100,0){$D$}[a]
\tzto[bend right=15]"supp"(0,10)(100,90){$S$}[ar]
%\tzXpoint*{dem}{supp}(eqm){$E$}
%\tzproj(eqm){$Q^*$}{$P^*$}
\end{tikzpicture}
```



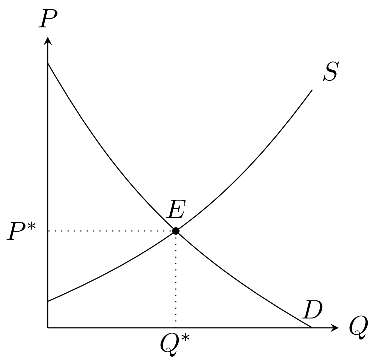
Step 3 Find an equilibrium point and name it. Use the starred version `\tzXpoint*` to print a dot and then label the point.

```
\begin{tikzpicture}[x=0.05cm,y=0.05cm,scale=.7]
% to avoid `dimension too large' error: x/y=0.05cm
\tzhelplines[step=.5cm](110,110)
\tzaxes(110,110){$Q$}{$P$}
\tzto[bend right=15]"dem"(0,100)(100,0){$D$}[a]
\tzto[bend right=15]"supp"(0,10)(100,90){$S$}[ar]
\tzXpoint*{dem}{supp}(eqm){$E$}
%\tzproj(eqm){$Q^*$}{$P^*$}
\end{tikzpicture}
```



Step 4 If necessary, use `\tzproj` to draw projection lines with text around the projection points.

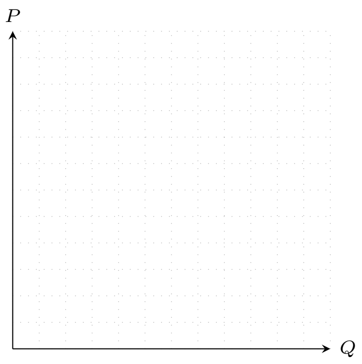
```
\begin{tikzpicture}[x=0.05cm,y=0.05cm,scale=.7]
% to avoid `dimension too large' error: x/y=0.05cm
%\tzhelplines[step=.5cm](110,110)
\tzaxes(110,110){$Q$}{$P$}
\tzto[bend right=15]"dem"(0,100)(100,0){$D$}[a]
\tzto[bend right=15]"supp"(0,10)(100,90){$S$}[ar]
\tzXpoint*{dem}{supp}(eqm){$E$}
\tzproj(eqm){$Q^*$}{$P^*$}
\end{tikzpicture}
```



5.1.2 Tax incidence: step by step

Step 1: Determine the size of the graph.

```
\begin{tikzpicture}[scale=.035,font=\scriptsize]
\tzhelplines[step=10cm](120,120)
\tzaxes(120,120){$Q$}{$P$}
\end{tikzpicture}
```



Step 2: Define functions and plot them. And then, find an intersection point.

Step 3: Draw the shifted supply curve and find new equilibrium point. And then, project the point on each axis.

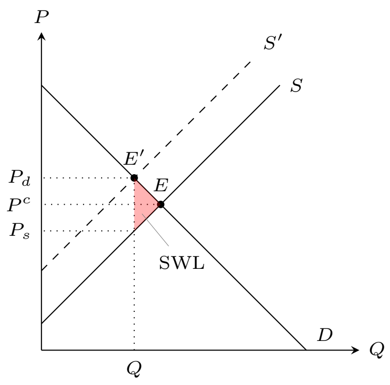
Step 4: To illustrate the social welfare loss (SWL), find a vertical intersection point of the original supply curve using new equilibrium point.

Step 5: Project the old equilibrium point and the vertical intersection point onto the y axis and add text.

Step 6: Fill the area of the social welfare loss with color.

Step 7: Add text ‘SWL’ at the appropriate place.

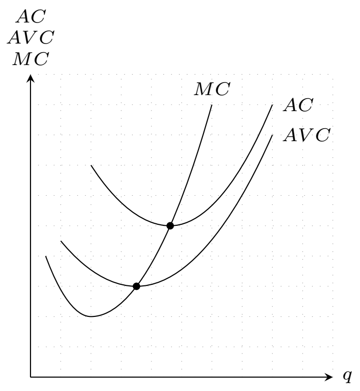
```
\begin{tikzpicture}[scale=.035,font=\scriptsize]
%\tzhelplines[step=10cm](120,120)
\tzaxes(120,120){$Q$}{$P$}
% step 2
\def\demA{100-\x}
\def\suppA{10+\x}
\tzfn\demA[0:100]{$D$}[ar]        % name path = \demA
\tzfn\suppA[0:90]{$S$}[r]         % name path = \suppA
\tzXpoint*{demA}{suppA}(E){$E$}
% step 3
\tzfn[dashed]"suppB"{\suppA+20}[0:80]{$S'$}[ar]
\tzXpoint*{demA}{suppB}(newE){$E'$}
\tzproj(newE){$Q$}{$P_d$}
% step 4
\tzvXpoint{suppA}(newE)(vX)
% step 5
\tzprojy(vX){$P_s$}
\tzprojy(E){$P^c$}
% step 6
\tzpath*[red](E)(newE)(vX);
% step 7
\tznode($(E)!10cm!(E-|0,0)$){}[pin={-70:SWL}]
\end{tikzpicture}
```



5.2 Firms

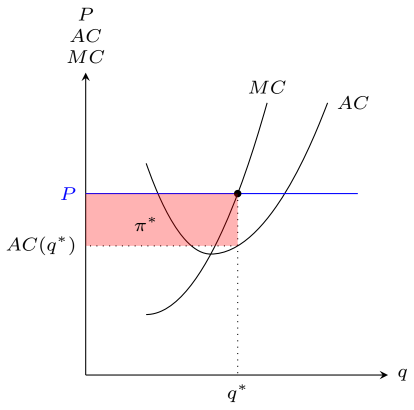
5.2.1 Cost curves

```
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\tzhelplines(10,10)
\tzaxes(10,10){$q$}{$AC$\\$AVC$\\$MC$}[align=center]
\tzparabola"MC"(.5,4)(2,2)(6,9){$MC$}[a]
\tzhXpointat{MC}{5}(A)
\tzhXpointat{MC}{3}(B) % (B-2) will be used!
\tzdots*(A)(B-2);
\tzparabola(2,7)(A)(8,9){$AC$}[r]
\tzparabola(1,4.5)(B-2)(8,8){$AVC$}[r]
\end{tikzpicture}
```



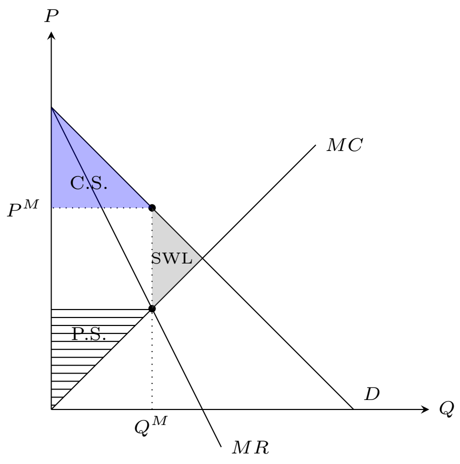
5.2.2 Equilibrium of a competitive firm

```
\begin{tikzpicture}[scale=.4,font=\scriptsize]
%\tzhelplines(10,10)
\tzaxes(10,10){$q$}{$P$\\$AC$\\$MC$}[align=center]
\tzparabola"MC"(2,2)(6,9){$MC$}[a]
\tzhXpointat{MC}{4}(A) % point (A) on MC at q=4
\tzparabola"AC"(2,7)(A)(8,9){$AC$}[r] % (A): minAC
\tzhfnat[blue]"price"{6}[0:9]{$P$}[l,at start]
\tzXpoint*{price}{MC}(E)
\tzprojx(E){$q^*$}
\tzvXpoint{AC}(E)(ACeqm) % point on AC in eqmuilibrium
\tzprojy(ACeqm){$AC(q^*)$}
\tzpath*[red](E-|0,0)(E)(ACeqm)(ACeqm-|0,0);
\tznode(2,5){$\pi^*$}
\end{tikzpicture}
```



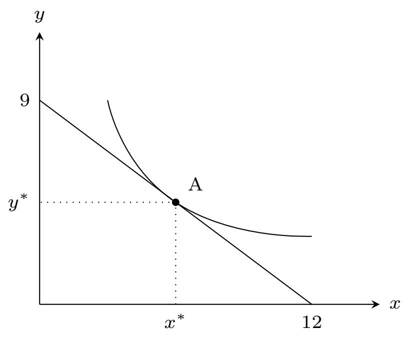
5.2.3 Monopoly equilibrium

```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(10,10)
\tzaxes(10,10){$Q$}{$P$}
\def\DD{8-\x}
\def\MR{8-2*\x}
\def\MC{\x}
\tzfn\DD[0:8]{$D$}[ar]
\tzfn\MR[0:4.5]{$MR$}[r]
\tzfn\MC[0:7]{$MC$}[r]
\tzXpoint*{MR}{MC}(E)
\tzvXpoint*{DD}(E)(EE)
\tzproj(EE){$Q^M$}{$P^M$}
\tzXpoint{DD}{MC}(C)
\tzpath*(EE)(C)(E);
\tznode(C){\tiny SWL}[l]
\tzpath*[blue](0,8)(EE)(EE-|0,0);
\tznode(1,6){C.S.}
\tzpath[pattern=horizontal lines](0,0)(E)(E-|0,0);
\tznode(1,2){P.S.}
\end{tikzpicture}
```

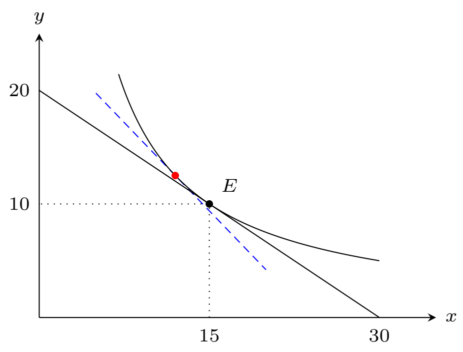


5.3 Consumers: Budget lines and indifference curves

```
\begin{tikzpicture}[scale=.3,font=\scriptsize]
%\tzhelplines(15,12)
\tzaxes(15,12){$x$}{$y$}
\def\bgt{-3/4*\x+9} % 3x+4y=36
\tzfn\bgt[0:12]
\tzvXpoint*{bgt}(6,0)(A){A}[45]
\tzplotcurve"ICC"(3,9)(A)(12,3); % trial and error
\tzproj(A){$x^*$}{$y^*$}
\tzticks{12}{9}
\end{tikzpicture}
```

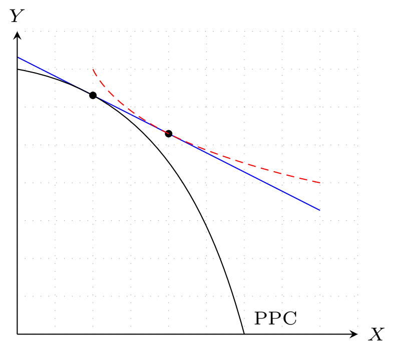


```
\begin{tikzpicture}[scale=.15,font=\scriptsize]
%\tzhelplines(35,25)
\tzaxes(35,25){$x$}{$y$}
\def\bgt{-2/3*\x+20} % 2x+3y=60
\def\IC{150/\x}       % u(x,y)=xy
\tzfn\bgt[0:30]
\tzfn\IC[7:30]
\tzcoor*(15,10)(E){$E$}[45]
\tzproj(E)
\tzticks{15,30}{10,20}
\tzvXpointat*[red]{IC}{12}(A)
\tztangent[blue,densely dashed]{IC}(A)[5:20]
\end{tikzpicture}
```



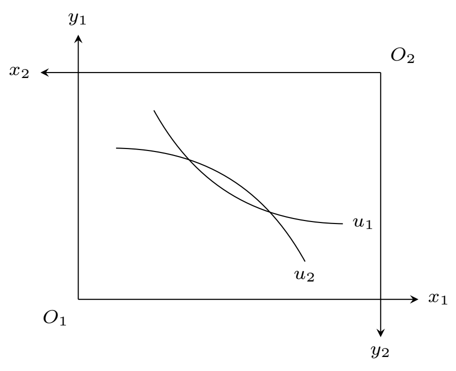
5.4 Production Possibility Curves

```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(9,8)
\tzaxes(9,8){$X$}{$Y$}
\tzto[out=-10,in=105]"PPC"(0,7)(6,0){PPC}[ar]
\tzvXpointat*{PPC}{2}(E)
\tztangent[blue]"tan"{PPC}(E)[0:8]
\tzvXpointat*{tan}{4}(F)
\tzplotcurve[densely dashed,red](2,7)(F)(8,4);
\end{tikzpicture}
```



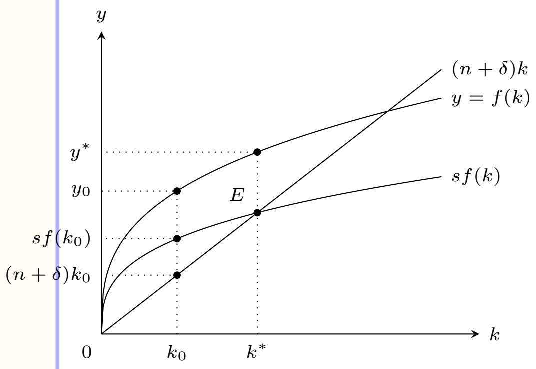
5.5 Edgeworth box

```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(9,7)
\tzaxes(9,7){$x_1$}{$y_1$}
\tzaxes<8,6>(8,6)(-1,-1){$x_2$}[l]{$y_2$}[b]
\tzshoworigin{$O_1$}
\tzshoworigin(8,6){$O_2$}[ar]
\tzto[bend right](2,5)(7,2){$u_1$}[r]
\tzto[bend left](1,4)(6,1){$u_2$}[b]
\end{tikzpicture}
```



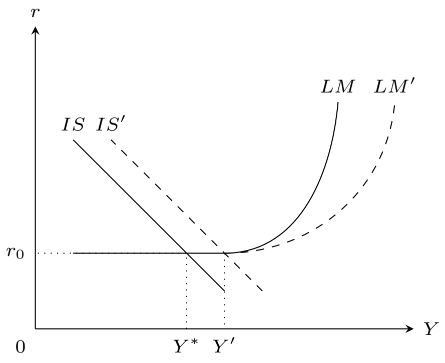
5.6 Growth

```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(10,8)
\tzshoworigin
\tzaxes*(10,8){$k$}{$y$} % bounding box
\def\Fk{3*(\x)^(1/3)}
\def\sFk{2*(\x)^(1/3)}
\def\ndk{7/9*\x}
\tzfn\Fk[0:9]{$y=f(k)$}[r]        % path name=Fk
\tzfn\sFk[0:9]{$sf(k)$}[r]        % path name=sFk
\tzfn\ndk[0:9]{$(n+\delta)k$}[r]  % path name=ndk
\tzXpoint*{sFk}{ndk}(E)[2]{$E$}[135] % 2nd X point
\tzvXpoint*{Fk}(E-2)(YE) % 2nd X point
\tzproj(YE){$k^*$}{$y^*$}
\tzvXpointat*{Fk}{2}(A)
\tzvXpointat*{sFk}{2}(B)
\tzvXpointat*{ndk}{2}(C)
\tzproj(A){$k_0$}{$y_0$}
\tzprojy(B){$sf(k_0)$}
\tzprojy(C){$(n+\delta)k_0$}
\end{tikzpicture}
```



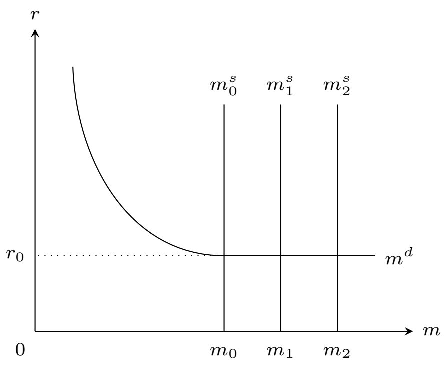
5.7 Liquidity trap

```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(10,8)
\tzshoworigin
\tzaxes(10,8){$Y$}{$r$}
\tzline"LM"(1,2)(5,2)
\tzto[out=0,in=-95](5,2)(8,6){$LM$}[a]
\tzto[dashed,out=0,in=-95](5,2)(9.5,6){$LM'$}[a]
\tzLFn"IS"(4,2){-1}[5:1]{$IS$}[a]
\tzLFn[dashed]<1,0>"ISa"(4,2){-1}[5:1]{$IS'$}[a] %%
\tzXpoint{IS}{LM}(E)
\tzXpoint{ISa}{LM}(E1)
\tzproj(E){$Y^*$}{$r_0$}
\tzprojx(E1){$Y'$}
\end{tikzpicture}
```



LM curves are drawn with two paths. To shift the *IS* curve, `<shift coor>` is used. See Section 20.2 on page 113 for more details.

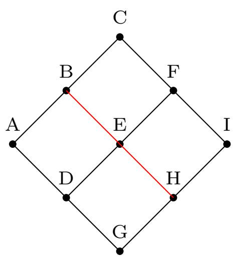
```
\begin{tikzpicture}[scale=.5,font=\scriptsize]
%\tzhelplines(10,8)
\tzshoworigin
\tzaxes(10,8){$m$}{$r$}
\tzto[out=-88,in=180](1,7)(5,2)
     <--(9,2) node [r] {$m^d$}> % code.append
\tzvfnat{5}[0:6]{$m^s_0$}[a]
\tzvfnat{6.5}[0:6]{$m^s_1$}[a]
\tzvfnat{8}[0:6]{$m^s_2$}[a]
\tzprojy(5,2){$r_0$}
\tzticksx{5/{$m_0$},6.5/{$m_1$},8/{$m_2$}}
\end{tikzpicture}
```



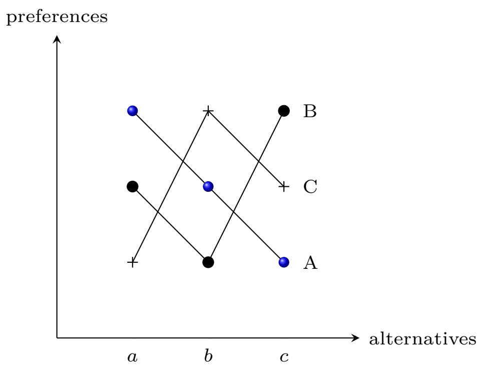
The money demand (m_d) curve is drawn with one path. To do this, `<code.append>` is used. See Section 12.1 on page 74 for more details.

5.8 Miscellany

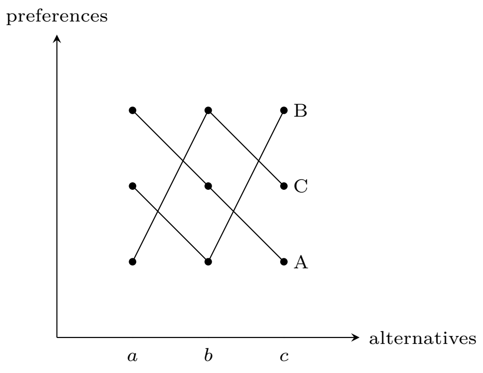
```
\begin{tikzpicture}[font=\scriptsize]
\def\z{1}
\tzcoors
  (0,0)(A)
  ($(A) + (45:\z)$)(B)
  ($(B) + (45:\z)$)(C)
  ($(A) + (-45:\z)$)(D)
  ($(D) + (45:\z)$)(E)
  ($(E) + (45:\z)$)(F)
  ($(D) + (-45:\z)$)(G)
  ($(G) + (45:\z)$)(H)
  ($(H) + (45:\z)$)(I);

\foreach \a in {A,...,I}
  { \tzdot*(\a){\a}
  }

\tzlines(D)(A)(B)(C)(F)(E)(D)(G)(H)(I)(F);
\tzlines[red](B)(E)(H);
\end{tikzpicture}
```



```
\begin{tikzpicture}[scale=1,font=\scriptsize]
\tzaxes(4,4){alternatives}{preferences}
\tzticksx{1/$a$,2/$b$,3/$c$}
\tzplot[mark=ball](1,3)(2,2)(3,1){A}[0];
\tzplot[mark=*]   (1,2)(2,1)(3,3){B}[0];
\tzplot[mark=+]   (1,1)(2,3)(3,2){C}[0];
\end{tikzpicture}
```



```
\begin{tikzpicture}[scale=1,font=\scriptsize]
\tzaxes(4,4){alternatives}{preferences}
\tzticksx{1/$a$,2/$b$,3/$c$}
\foreach \x in {1,2,3}
  {\foreach \y in {1,2,3}
    {\tzdot*(\x,\y)}
  }
\tzlines(1,3)(2,2)(3,1){A}[r];
\tzlines(1,2)(2,1)(3,3){B}[r];
\tzlines(1,1)(2,3)(3,2){C}[r];
\end{tikzpicture}
```



Part III
Points, Lines, and Curves

6 Getting Ready

6.1 Styles: tzdotted, tzdashed, tzhelplines

The styles `tzdotted`, `tzdashed`, and `tzhelplines` are defined as follows:

```
% styles: tzdotted, tzdashed, tzhelplines
\tikzset{%
  tzdotted/.style={line cap=round,dash pattern=on 0pt off 1cm/(#1)},
  tzdotted/.default=10
}
\tikzset{%
  tzdashed/.style={dashed=none,dash pattern=on 5mm/(#1) off 5mm/(#1)},
  tzdashed/.default=10
}
\tikzset{%
  tzhelplines/.style={help lines,-,tzdotted}
}
```

The styles `tzdotted` and `tzdashed` print 10 dots and 10 dashes per 1cm, respectively, by default. The style `tzhelplines` uses `tzdotted` by default.

6.2 \tzhelplines, \tzhelplines*

`\tzhelplines` draws `grid` from the first coordinate to the second coordinate. If only one coordinate is specified, then the first coordinate is regarded as `(0,0)`.

The starred version `\tzhelplines*` uses the grid as a *bounding box*.

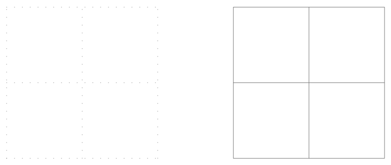
```
% syntax: minimum
\tzhelplines(<coor>)
% syntax: full
\tzhelplines[<opt>](<coor1>)(<coor2>)
% defaults
  [help lines,tzdotted=10](<m>)()
% (<m>): mandatory argument
```

Here, `(<m>)` stands for a *mandatory* argument.

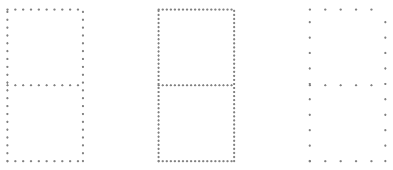
```
\tzhelplines(4,3)      % works similarly to:
  \draw [help lines] (0,0) grid (4,3);

\tzhelplines(1,1)(4,3) % works similarly to:
  \draw [help lines] (1,1) grid (4,3);
```

By default, `\tzhelplines` prints `grid` with `10` dots per `1cm`. `\tzhelplines` with the option value `[tzdotted=<n>]` prints `<n>` dots per `1cm`. (That is, the default value is `tzdotted=10`.)

```
% \tzhelplines
\begin{tikzpicture}
\tzhelplines(2,2)
\draw [help lines] (3,0) grid (5,2);
\end{tikzpicture}
```

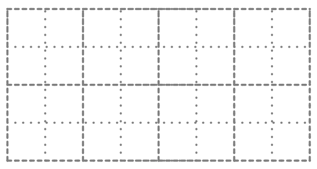


```
% tzdotted: (default: 10 dots per 1cm)
\begin{tikzpicture}
\tzhelplines[thick](1,2)                    % 10 dots
\tzhelplines[thick,tzdotted=20](2,0)(3,2) % 20 dots
\tzhelplines[thick,tzdotted=5](4,0)(5,2)  %  5 dots
\end{tikzpicture}
```

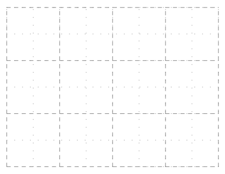


With the option value, `[tzdotted=<n>/<d>]`, `\tzhelplines` prints `<n>` dots per `<d>`cm. Similarly for `tzdashed`.

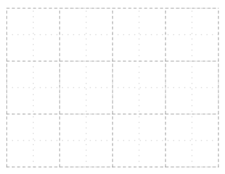
```
% tzdotted, tzdashed
\begin{tikzpicture}
\tzhelplines[thick,tzdashed](4,2)
\tzhelplines[thick,step=.5](4,2)
\end{tikzpicture}
```



```
% scaled: 7 dots per .7cm (10 dots per hard 1cm)
\begin{tikzpicture}[scale=.7]
\tzhelplines[tzdashed](4,3)
\tzhelplines[step=.5](4,3)
\end{tikzpicture}
```



```
% scaled: 10/.7 means 10 dots per .7cm
\begin{tikzpicture}[scale=.7]
\tzhelplines[tzdashed=10/.7](4,3)          %%
\tzhelplines[step=.5,tzdotted=10/.7](4,3) %%
\end{tikzpicture}
```



6.3 \tzbbox: A bounding box

`\tzbbox` sets a bounding box.

```
% syntax
\tzbbox(<coor1>)(<coor2>)
% defaults
  (0,0)(<m>)
```

```
\tzbbox(-1,-1)(4,3) % is an abbreviation of:
  \path [ use as bounding box ] (-1,-1) rectangle (4,3);
```

If only one coordinate is specified, the first coordinate is regarded as `(0,0)`.

7 Dots

7.1 \tzcdot(*): A small circle

A dot is usually expressed by a small circle. `\tzcdot` prints a circle dot ∘.

The starred version `\tzcdot*` prints a filled circle dot •. The *radius* of the circle is `1.2pt`, by default.

```
% syntax: minimum
\tzcdot(<coor>)
% syntax: medium
\tzcdot*(<coor>){<label>}[<angle>](<radius>)
% syntax: full
\tzcdot*[<opt>]<shift coor>(<coor>){<label>}[<[label opt]angle>](<radius>)
% defaults
 *[ solid,thin,tzcdot=1.2pt ]<>(<m>){}[](1.2pt)
% tzcdot is a predefined key (in this package).
% (<m>): mandatory
```

Here, `(<m>)` stands for a *mandatory* argument. All others are optional arguments.

How to change the size There are three ways to change the *radius* of a circle dot drawn by `\tzcdot`.

1. The simplest way is to use the last parenthesis option, like `\tzcdot(0,0)(3pt)`.

```
\tzcdot(0,0)        % is an abbreviation of:
  \draw (0,0) circle (1.2pt); % default radius=1.2pt
```

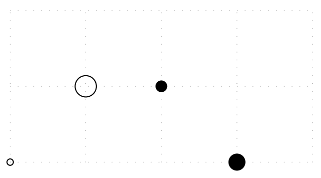
```
\tzcdot*(0,0)(3pt) % is an abbreviation of:
  \draw [fill] (0,0) circle (3pt);
```

2. You can use the key-value option `[tzcdot=<dim>]`, like `\tzcdot[tzcdot=3pt](0,0)`, to change the *radius* of a circle dot. The `tzcdot` key is defined in the package. If both the `tzcdot` key-value and the last parenthesis option are used, the former wins.

```
\tzcdot*(1,1) % works like:
  \draw [fill] (1,1) circle [radius=1.2pt]; % default radius=1.2pt
```

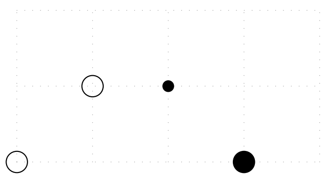
```
\tzcdot*[tzcdot=3pt] % works like:
  \draw [fill] (0,0) circle [radius=3pt];
```

```
% \tzcdot(*)
\begin{tikzpicture}
\tzhelplines(4,2)
\tzcdot(0,0)         \tzcdot[tzcdot=4pt](1,1)
\tzcdot*(2,1)(2pt)   \tzcdot*(3,0)(3pt)
\end{tikzpicture}
```



3. Another way to change the radius is to use a macro, like `\settzcdotradius{3pt}`, It is effective within the `tizpicture` environment unless changed by `\settzcdotradius` again.

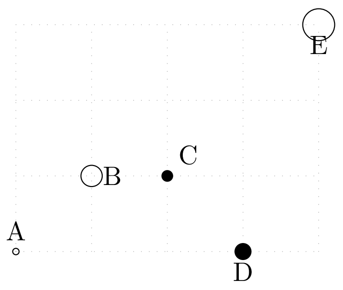
```
% \settzcdotradius
\begin{tikzpicture}
\tzhelplines(4,2)
\settzcdotradius{4pt}
\tzcdot(0,0)        \tzcdot(1,1)
\tzcdot*(2,1)(2pt)  \tzcdot*(3,0)
\end{tikzpicture}
```



How to label You can add a label to a specified coordinate by adding the optional argument `{<label>}` immediately after `(<coor>)`. You can also change the `<label>` position by the option `[<angle>]`. (Note that you *cannot* use the abbreviations such as `a`, `r`, `bl`, etc. to place label nodes.)

```
\tzcdot(0,0){A} % is an abbreviation of:
  \draw (0,0) circle (1.2pt) node [label={:A}] {};

\tzcdot(0,0)(2pt){A}[[blue]0] % is an abbreviation of:
  \draw (0,0) circle (2pt) node [label={[blue]0:A}] {};
```

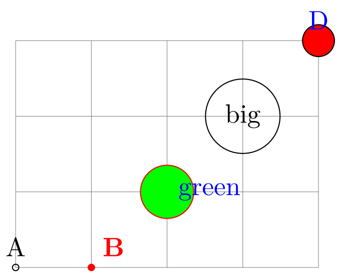
```
% \tzcdot: labels
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdot(0,0){A} % default position: 90 or above
\tzcdot[tzcdot=4pt](1,1){B}[0]
\tzcdot*(2,1){C}[45](2pt)
\tzcdot*(3,0){D}[-90](3pt)
\tzcdot(4,3){E}[-90](6pt)
\end{tikzpicture}
```



The label position does not depend on the size of circle dots.

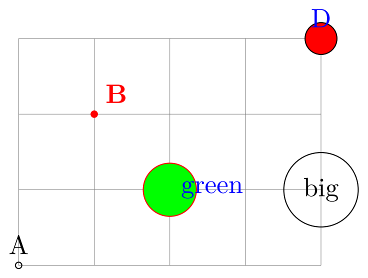
How to change colors With the first optional argument `[<opt>]`, you can change the color of a dot. You can also change the color of a label, as shown in the following example.

```
% \tzcdot(*)
\begin{tikzpicture}
\draw [help lines] (0,0) grid (4,3);
\tzcdot(0,0){A}
\tzcdot*[red](1,0){\textbf{B}}[45]
\tzcdot*[red,fill=green](2,1){green}[[blue]0](10pt)
\tzcdot[tzcdot=2*7pt](3,2){big}[center]
\tzcdot*[fill=red,text=blue](4,3){D}(2*3pt)
\end{tikzpicture}
```

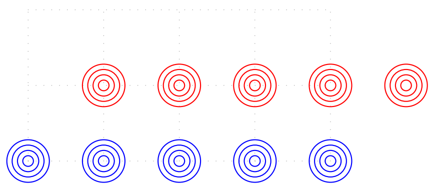


Shift Dots can be shifted by specifying the optional argument `<shift coor>` immediately before `(<coor>)`. The empty shift option `<>` is not allowed.

```
% \tzcdot: shift
\begin{tikzpicture}
\draw [help lines] (0,0) grid (4,3);
\tzcdot(0,0){A}
\tzcdot*[red]<0,2>(1,0){\textbf{B}}[45] % shift
\tzcdot*[red,fill=green](2,1){green}[[blue]0](10pt)
\tzcdot[tzcdot=2*7pt]<1,-1>(3,2){big}[center] % shift
\tzcdot*[fill=red,text=blue](4,3){D}(2*3pt)
\end{tikzpicture}
```



```
% \tzcdot: repeated
\begin{tikzpicture}
\tzhelplines(4,2)
\foreach \x in {0,...,4}
  { \foreach \A in {2,4,6,8}
      { \tzcdot[blue]    (\x,0)(\A pt) } }
\foreach \x in {0,...,4}
  { \foreach \A in {2,4,6,8}
      { \tzcdot[red]<1,1>(\x,0)(\A pt) } } % shift
\end{tikzpicture}
```



7.2 \tzcdots(*): Multiple circle dots

The macro `\tzcdots` takes an arbitrary number of coordinates as arguments to print multiple circle dots with the radius `1.2pt`, by default. You need to indicate when the iteration of an arbitrary number of coordinates ends, by typing a *semicolon* `;`. Let us call this kind of macro a *semicolon version* macro.

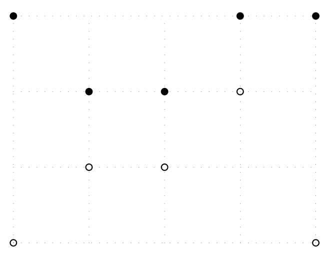
Remark:

- DO NOT FORGET to enter '`;`' at the end of iteration.
  - Tokens such as '`!`' and '`#`' other than the open parenthesis '`(`' work like the semicolon '`;`' to indicate the end of iteration. But it is highly recommended to use '`;`' for consistency.
- Without the semicolon `;`, the macro does nothing or a compile error occurs.

The starred version `\tzcdots*` prints multiple filled dots.

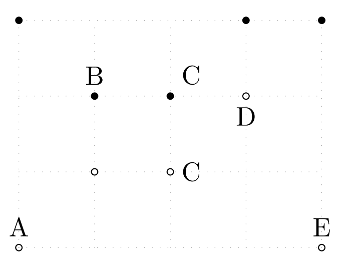
```
% syntax: minimum
\tzcdots*(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: full
\tzcdots*[<opt>]<shift coor> (<coor>){<label>}[<[label opt]angle>]
                              ..repeated.. (){}[] ; (<dot radius>)
% defaults
 *[tzcdot=1.2pt]<>(<m>){}[]..repeated..(){}[] ; (1.2pt)
```

```
% \tzcdots(*)
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdots(0,0)(1,1)(2,1)(3,2)(4,0);
\tzcdots*(0,3)(1,2)(2,2)(3,3)(4,3);
\end{tikzpicture}
```



How to label Each coordinate can be followed by the optional arguments `{<label>}` and `[<angle>]` to label dots. So the repeating pattern is the triple `(<coor>){<label>}[<angle>]`.

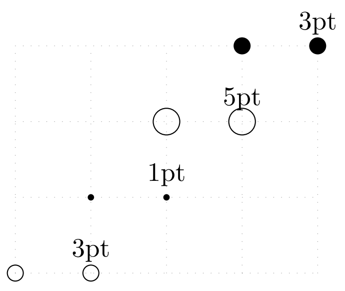
```
% \tzcdots: label
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdots(0,0){A}
        (1,1)
        (2,1){C}[0]
        (3,2){D}[-90]
        (4,0){E};
\tzcdots*(0,3)(1,2){B}(2,2){C}[45](3,3)(4,3);
\end{tikzpicture}
```



How to change the size of dots There are three ways of changing the *radius* of dots.

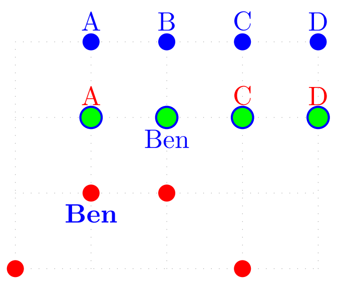
1. The simplest way is to use the *last* parenthesis optional argument, *after the semicolon.*
2. Another way is to use the `tzcdot` key, like `\tzcdots[tzcdot=3pt]...`. If both options are used the key-value option wins.
3. You can also use the macro `\settzcdotradius`. The effect remains within the `tikzpicture` environment unless it is changed again.

```
% \tzcdots: size (radius)
\begin{tikzpicture}
\tzhelplines(4,3)
\settzcdotradius{3pt}
\tzcdots(0,0)(1,0){3pt};
\tzcdots*(1,1)(2,1){1pt};(1pt) % simplest
\tzcdots[tzcdot=5pt](2,2)(3,2){5pt};
\tzcdots*(3,3)(4,3){3pt};
\end{tikzpicture}
```



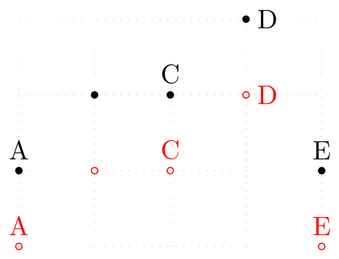
How to change colors With the first optional argument `[<opt>]`, you can change the color of dots. You can also change the color of all labels at once using the first optional argument, like `\tzcdots[text=red]...` as shown in the following example.

```
% \tzcdots: color
\begin{tikzpicture}
\tzhelplines(4,3)
\settzcdotradius{3pt}
\tzcdots*[red]
  (0,0)(1,1){\textbf{Ben}}[[blue]-90](2,1)(3,0);
\tzcdots*[thick,blue,fill=green,text=red]
  (1,2){A}(2,2){Ben}[[blue]-90](3,2){C}(4,2){D};(4pt)
\tzcdots*[blue]
  (1,3){A}(2,3){B}(3,3){C}(4,3){D};
\end{tikzpicture}
```



Shift You can move the coordinates of dots by specifying `<shift coor>` option immediately before the first coordinate.

```
% \tzcdots: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcdots[red] (0,0){A}(1,1)(2,1){C}(3,2){D}[0](4,0){E};
\tzcdots*<0,1>(0,0){A}(1,1)(2,1){C}(3,2){D}[0](4,0){E};
\end{tikzpicture}
```



7.3 \tzdot(*): A single node dot

The macro `\tzdot` prints a small circle node ∘, as a dot, with the *diameter* (or `minimum size`) of `2.4pt`, by default.

The starred version `\tzdot*` prints a filled dot •.

```
% syntax: minimum
\tzdot(coor)
% syntax: medium
\tzdot*(<coor>){<label>}[<angle>](<dot size>)
% syntax: full
\tzdot*[<node opt>]<shift coor>(<coor>){<label>}[<[label opt]angle>](<dot size>)
% defaults
 *[ tzdot=2.4pt ]<>(<m>){}[](2.4pt)
% the style tzdot is predefined
% (<m>): mandatory
```

`\tzdot(*)` accepts one mandatory argument, denoted by `(<m>)`. All others are optional.

The style `tzdot` is defined as follows:

```
% style: tzdot
\tikzset{
  tzdot/.style={draw,circle,solid,thin,inner sep=0pt,minimum size=#1},
  tzdot/.default=2.4pt
}
```

7.3.1 Three Ways to change the size of node dots

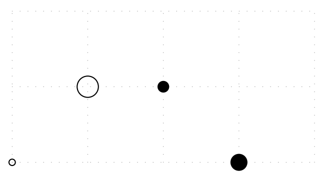
There are Three Ways to change the *diameter* (or `minimum size`) of a node dot drawn by `\tzdot(*)`.

1. Use the predefined style `tzdot` in the first optional agrument, like `\tzdot[tzdot=5pt](0,0)`, which gives the same result as `\tzdot[minimum size=5pt](0,0)`.

   ```
   \tzdot(0,0) % works like:
     \path (0,0) node [tzdot=2.4pt] {}; % default size
     % or equivalently
     \node [tzdot,minimum size=2.4pt] at (0,0) {};
   ```

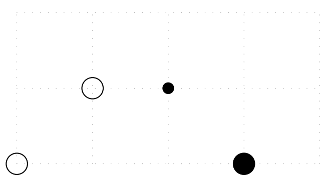
2. The simplest way is to use the *last* parenthesis optional argument, like `\tzdot(0,0)(5pt)`, which yields the same result as in `\tzdot[tzdot=5pt](0,0)`. If both options are used, the `tzdot` (or `minimum size`) option overwrites the last parenthesis option.

```
% \tzdot(*)
\begin{tikzpicture}
\tzhelplines(4,2)
\tzdot(0,0)         \tzdot[tzdot=8pt](1,1)
\tzdot*(2,1)(4pt)   \tzdot*(3,0)(6pt)
\end{tikzpicture}
```



3. To change the size of all node dots drawn by `\tzdot*`, you can use the macro `\settzdotsize`. Its effect remains valid until the end of `tikzpicture` environment unless changed again.

```
% \settzdotsize
\begin{tikzpicture}
\tzhelplines(4,2)
\settzdotsize{8pt}
\tzdot(0,0)         \tzdot(1,1)
\tzdot*(2,1)(4pt)   \tzdot*(3,0)
\end{tikzpicture}
```



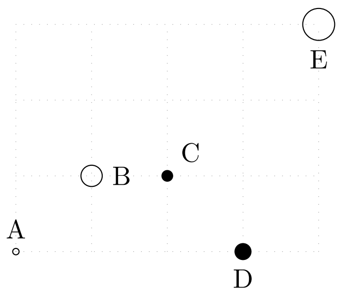
7.3.2 How to label

You can add a label to a specified coordinate by specifying the optional argument `{<label>}` immediately after the coordinate `(<coor>)`.

You can also change the label position by the option `[<angle>]` following `{<label>}`.

```
\tzdot(0,0){A}(3pt)          % works like:
  \path (0,0) node [tzdot=3pt,label={:A}] {};

\tzdot*(0,0){A}[[red]0](3pt) % works like:
  \path (0,0) node [fill,tzdot=3pt,label={[red]0:A}] {};
```

```
% \tzdot: labels
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdot(0,0){A}
\tzdot(1,1){B}[0](8pt)
\tzdot*(2,1){C}[45](4pt)
\tzdot*(3,0){D}[-90](6pt)
\tzdot(4,3){E}[-90](12pt)
\end{tikzpicture}
```



Unlike `\tzcdot`, the `\tzdot`'s label position depends on the size of a circle node. In Ti*k*Z jargon, `{<label>}` is in a *label node* for a *main node* that is a circle node with no text in it, so `<label>` moves accordingly as the main node dot gets bigger or smaller.

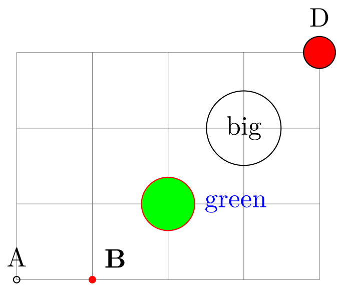
7.3.3 How to change colors and shapes

With the first optional argument `[<node opt>]`, you can change of the color or shape of dots. You can also change the label color using `[<label opt>]` as shown in the following example.

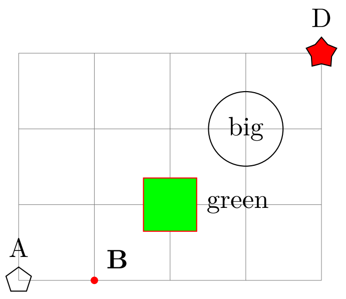
Remark:

- `[<node opt>]` is for options of *main nodes*, `[<label opt>]` is for options of *label nodes*.
- `[<label opt>]` is used in the form of `[<[label opt]angle>]`.

```
% \tzdot: color
\begin{tikzpicture}
\draw [help lines] (0,0) grid (4,3);
\tzdot(0,0){A}
\tzdot*[red](1,0){\textbf{B}}[45]
\tzdot*[red,fill=green](2,1){green}[[blue]0](2*10pt)
\tzdot[tzdot=4*7pt](3,2){big}[center]
\tzdot*[fill=red](4,3){D}(4*3pt)
\end{tikzpicture}
```



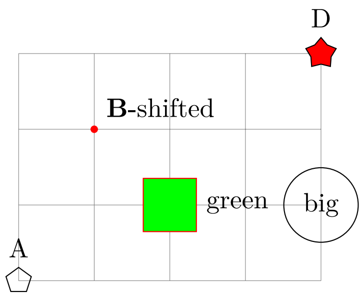
```
% \tzdot: shape
\begin{tikzpicture}
\draw [help lines] (0,0) grid (4,3);
\tzdot[regular polygon](0,0){A}(10pt)
\tzdot*[red](1,0){\textbf{B}}[45]
\tzdot*[red,fill=green,rectangle](2,1){green}[0](2*10pt)
\tzdot[tzdot=4*7pt](3,2){big}[center]
\tzdot*[fill=red,star](4,3){D}(4*3pt)
\end{tikzpicture}
```



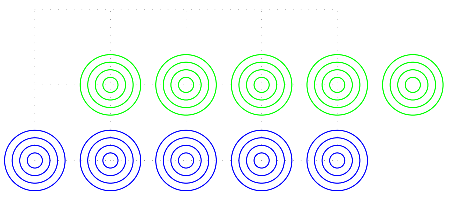
7.3.4 How to move: shift

Dots can be shifted by specifying the optional argument `<shift coor>` immediately before `(<coor>)`. The empty shift option `<>` is not allowed.

```
% \tzdot: shift
\begin{tikzpicture}
\draw [help lines] (0,0) grid (4,3);
\tzdot[regular polygon](0,0){A}(10pt)
\tzdot*[red]<0,2>(1,0){\textbf{B}-shifted}[45] % shift
\tzdot*[red,fill=green,rectangle](2,1){green}[0](2*10pt)
\tzdot[tzdot=4*7pt]<1,-1>(3,2){big}[center]     % shift
\tzdot*[fill=red,star](4,3){D}(4*3pt)
\end{tikzpicture}
```



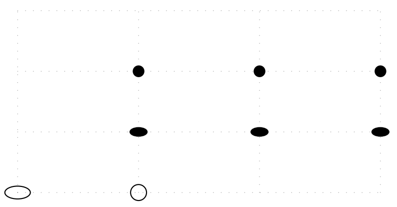
```
% \tzdot: repeated
\begin{tikzpicture}
\tzhelplines(4,2)
\foreach \x in {0,...,4}
  { \foreach \A in {1,2,3,4}
      { \tzdot[blue]       (\x,0)(2*\A mm) } }
\foreach \x in {0,...,4}
  { \foreach \A in {1,2,3,4}
      { \tzdot[green]<1,1>(\x,0)(2*\A mm) } }  % shift
\end{tikzpicture}
```



7.3.5 Comparison: `\tzdot` and `\tzcdot`

The most important difference between `\tzcdot` and `\tzdot` is that `\tzcdot` is affected by Ti*k*Z's scaling factor, but `\tzdot` is not. This is critical when `xscale` is not equal to `yscale`.

```
\begin{tikzpicture}[xscale=1.6,yscale=.8]
\tzhelplines(3,3)
\tzcdot(0,0)(3pt)              % dostorted
\tzdot(1,0)(6pt)
\tzcdots*(1,1)(2,1)(3,1);(2pt) % distorted
\tzdots*(1,2)(2,2)(3,2);(4pt)
\end{tikzpicture}
```



The following table further shows the differences between them.

```
% concept        % single  % multi    % size control      % [key=default size]
node [circle]    \tzdot    \tzdots    \settzdotsize       [tzdot=2.4pt]  % diameter
circle           \tzcdot   \tzcdots   \settzcdotradius    [tzcdot=1.2pt] % radius
```

Remark:

- In Ti*k*Z, a 'node' is 'not' affected by 'scaling' unless the Ti*k*Z option `transform shape` is used together. `\tzdot` is also useful for labelling a large dot.
  - In `\tzdot`, `<label>` is a label in a *label node* for a node dot (as a *main node*). So if a main node dot gets larger or smaller, its label moves accordingly. (Unlike, the labels with `\tzcdot` or `\tzcdots`.)
  - The position of `<label>` in `\tzcdot` does not depend on the size of dots.
- The package `tzplot` takes `\tzdot` as a *standard dot*, not `\tzcdot`. So, you can apply the THREE WAYS (on page 41) to change the size of any standard dots.

7.4 `\tzdots(*)`: Multiple node dots

`\tzdots` takes an arbitrary number of coordinates as arguments to print multiple circle node dots with the *diameter* (or `minimum size`) of `2.4pt`, by default.

This is a *semicolon version* macro, with the repeating pattern `(<coor>){<label>}[<angle>]`, which means that you need to type a *semicolon* '`;`' at the end of the coordinate repetition. The *semicolon* says, "*The repetition ends here.*"

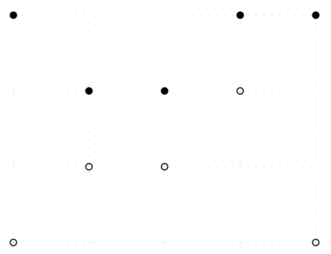
Remark:

- DO NOT FORGET to enter '`;`' at the end of iteration.
  - Tokens such as '`!`' and '`#`' other than the open parenthesis '`(`' work like the semicolon '`;`' to indicate the end of iteration. But it is highly recommended to use '`;`' for consistency.
- Without the semicolon `;`, the macro does nothing or a compile error occurs.

The starred version `\tzdots*` prints multiple filled node dots.

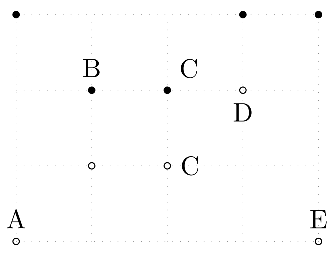
```
% syntax: minimum
\tzdots*(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: full
\tzdots*[<node opt>]<shift coor> (<coor>){<label>}[<[label opt]angle>]
                                   ..repeated.. (){}[] ; (<dot size>)
% defaults
 *[tzdot=2.4pt]<> (<m>){}[] ..repeated.. (){}[] ; (2.4pt)
```

```
% \tzdots(*)
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdots(0,0)(1,1)(2,1)(3,2)(4,0);
\tzdots*(0,3)(1,2)(2,2)(3,3)(4,3);
\end{tikzpicture}
```



How to label Each coordinate can be followed by the optional arguments `{<label>}` and `[<angle>]` to label dots. So the triple `(<coor>){<label>}[<angle>]` is the whole repeating pattern.

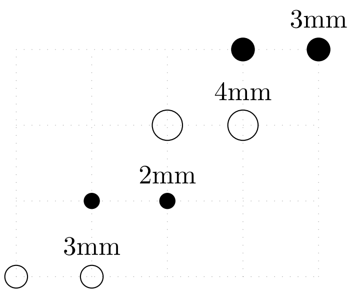
```
% \tzdots: label
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdots(0,0){A}
       (1,1)
       (2,1){C}[0]
       (3,2){D}[-90]
       (4,0){E};
\tzdots*(0,3)(1,2){B}(2,2){C}[45](3,3)(4,3);
\end{tikzpicture}
```



How to change the size of dots There are THREE WAYS of changing the *diameter* of node dots, as discussed in Section 7.3.1 on page 41.

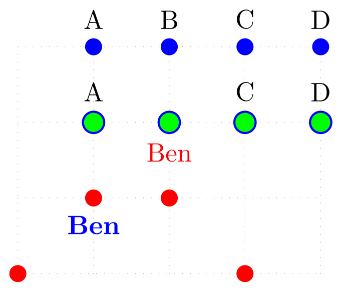
1. The simplest way is to use the *last* parenthesis optional argument, *after the semicolon.*
2. Another way is to use the style `tzdot`, like `\tzdots[tzdot=3pt]....` If both options are used the `tzdot` option style wins.
3. You can also use the macro `\settzdotsize`. The effect remains within the `tikzpicture` environment unless it is changed again.

```
% \tzdots: size (diameter)
\begin{tikzpicture}
\tzhelplines(4,3)
\settzdotsize{3mm}
\tzdots(0,0)(1,0){3mm};
\tzdots*(1,1)(2,1){2mm};(2mm) % simplest
\tzdots[tzdot=4mm](2,2)(3,2){4mm};
\tzdots*(3,3)(4,3){3mm};
\end{tikzpicture}
```



How to change colors With the first optional argument `[<node opt>]` you can change the color of node dots. You can also change the color of each label by `[<label opt>]`.

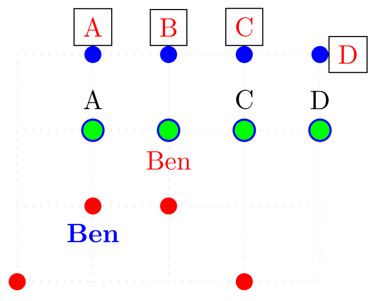
```
% \tzdots: color
\begin{tikzpicture}[->]
\tzhelplines(4,3)
\settzdotsize{6pt}
\tzdots*[red]
  (0,0)(1,1){\textbf{Ben}}[[blue]-90](2,1)(3,0);
\tzdots*[thick,blue,fill=green]
  (1,2){A}(2,2){Ben}[[red]-90](3,2){C}(4,2){D};(8pt)
\tzdots*[blue]
  (1,3){A}(2,3){B}(3,3){C}(4,3){D};
\end{tikzpicture}
```



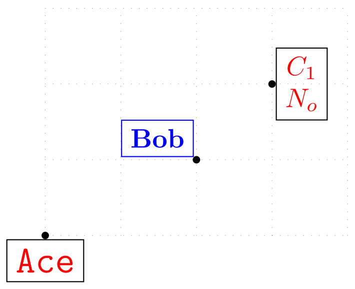
Remark:

- `[<node opt>]` is the option of a *main node* and `[<label opt>]` is the option of a *label node*.
- `[<label opt>]` is used in the form of `[<[label opt]angle>]`, like `[[red]90]`.
- You can control all labels together using `every label/.style` as in the following examples:

```
% \tzdots: every label/.style
\begin{tikzpicture}
\tzhelplines(4,3)
\settzdotsize{6pt}
\tzdots*[red]
  (0,0)(1,1){\textbf{Ben}}[[blue]-90](2,1)(3,0);
\tzdots*[thick,blue,fill=green]
  (1,2){A}(2,2){Ben}[[red]-90](3,2){C}(4,2){D};(8pt)
\tikzset{every label/.style={draw,text=red}} %%
\tzdots*[blue]
  (1,3){A}(2,3){B}(3,3){C}(4,3){D}[0];
\end{tikzpicture}
```

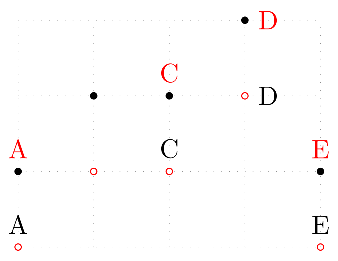


```
% every label/.style
\begin{tikzpicture}[every label/.style={draw,text=red}]
\tzhelplines(4,3)
\tzdots*
  (0,0){Ace}[[font=\LARGE\ttfamily]-90]
  (2,1){\textbf{Bob}}[[blue]135]
  (3,2){$C_1$\\$N_o$}[[align=center]0];
\end{tikzpicture}
```



Shift You can move the coordinates of dots by specifying `<shift coor>` option immediately before the first coordinate. The empty shift option `<>` is not allowed.

```
% \tzdots: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdots[red] (0,0){A}(1,1)(2,1){C}(3,2){D}[0](4,0){E};
\tikzset{every label/.style={red}}
\tzdots*<0,1>(0,0){A}(1,1)(2,1){C}(3,2){D}[0](4,0){E};
\end{tikzpicture}
```



8 Coordinates

8.1 \tzcoor and \tzcoor*

8.1.1 \tzcoor

For example, `\tzcoor(0,0)(A)` means that the coordinate `(0,0)` is named `(A)`.

```
\tzcoor(0,0)(A) % is an abbreviation of:
  \path (0,0) coordinate (A);
  % or
  \coordinate (A) at (0,0);
```

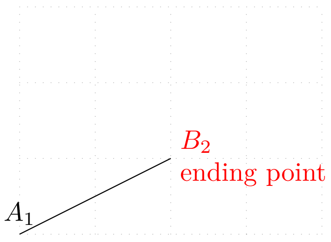
```
% syntax: minimum
\tzcoor(<coor>)(<name>)
% syntax: medium
\tzcoor(<coor>)(<name>){<label>}[<angle>]
% syntax: full
\tzcoor<shift coor>(<coor>)(<name>){<label>}[<[label opt]angle>]
% defaults
  <>(<m>)(<m>){}[]
```

Here, `<m>` stands for 'mandatory.' `\tzcoor` takes two mandatory arguments in parenthesis.

How to label You can put a label to a coordinate by specifying the optional arguments `{<label>}` and `[<angle>]` immediately after `(<name>)`.

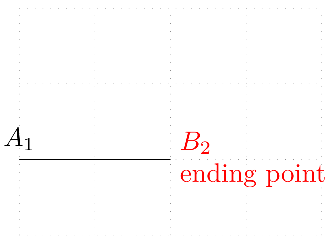
```
\tzcoor(0,0)(A){$A$}[0] % works like:
  \path (0,0) coordinate [label={0:$A$}] (A);
```

```
% \tzcoor
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(0,0)(A){$A_1$} % TikZ default: 90 or above
\tzcoor(2,1)(B){$B_2$\\ending point}[[align=left,red]0]
\draw (A) -- (B);
\end{tikzpicture}
```



Shift You can move the coordinate by specifying the optional argument `<shift coor>` before `(<coor>)`. The empty shift option `<>` is not allowed.

```
% \tzcoor: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor<0,1>(0,0)(A){$A_1$} %%
\tzcoor(2,1)(B){$B_2$\\ending point}[[align=left,red]0]
\tzline(A)(B)
\end{tikzpicture}
```



8.1.2 `\tzcoor*`

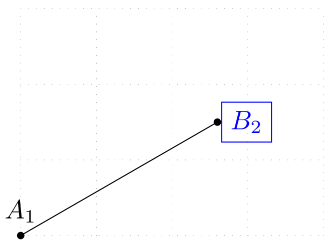
The starred version `\tzcoor*` works like `\tzcoor` with one exception. It prints a 'node dot' of the size `2.4pt`, by default, at a specified coordinate.

```
% syntax: minimum
\tzcoor*(<coor>)(<coor name>)
% syntax: medium
\tzcoor*(<coor>)(<coor name>){<label>}[<angle>]
% syntax: full
\tzcoor*[<dot opt>]<shift coor>
        (<coor>)(<name>){<label>}[[<label opt>]<angle>](<dot size>)
% defaults
 *[]<>(<m>)(<m>){}[](2.4pt)
```

```
\tzcoor*(0,0)(A) % works like:
  \path (0,0) coordinate (A);
  \tzdot*(0,0)
```

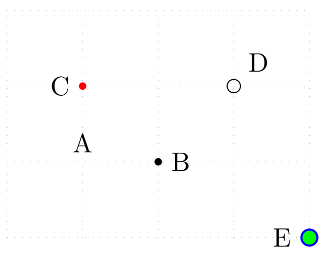
```
\tzcoor(0,0)(A){$A$}[right] % works like:
  \path (0,0) coordinate (A);
  \tzdot*(0,0){$A$}[right]
```

```
% \tzcoor*
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor*(0,0)(A){$A_1$} % TikZ default: 90 or above
\tzcoor*(30:3cm)(B){$B_2$}[[draw,blue]0]
\draw (A) -- (B);
\end{tikzpicture}
```



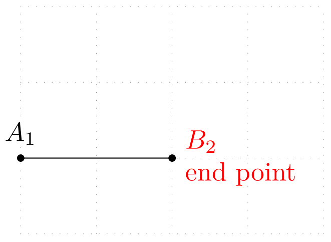
Changing the color and size of a dot You can change the color of a dot by specifying `[<dot opt>]`, which is, in fact, Ti*k*Z's `node` option. To change the size of dots, you can apply the Three Ways (see Subsection 7.3.1 on page 41).

```
% \tzcoor*: color, size
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(1,1)(A){A}
\tzcoor*(2,1)(B){B}[0]
\tzcoor*[red](1,2)(C){C}[180]
\tzcoor*[fill=none,tzdot=5pt](3,2)(D){D}[45]
\tzcoor*[blue,thick,fill=green](4,0)(E){E}[180](6pt)
\end{tikzpicture}
```



Shift The optional argument `<shift coor>` works just like in `\tzcoor`. The empty shift option `<>` is not allowed.

```
% \tzcoor*: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor*<0,1>(0,0)(A){$A_1$} %%
\tzcoor*(2,1)(B){$B_2$\\end point}[[align=left,red]0]
\tzline(A)(B)
\end{tikzpicture}
```



8.2 \tzcoors and \tzcoors*: Semicolon versions

8.2.1 \tzcoors

The macro `\tzcoors` takes an arbitrary number of pairs of coordinates and their names as arguments. For example, `\tzcoors(0,0)(A) (1,1)(B) (2,2)(C);` means that the coordinate `(0,0)` is represented by the name `(A)`, `(1,1)` by `(B)`, and `(2,2)` by `(C)`.

```
% syntax: minimum
\tzcoors(<coor>)(<name>)..repeated..(<coor>)(<name>) ;
% syntax: full
\tzcoors <shift coor>(<coor>)(<name>){<label>}[<[label opt]angle>]
                     ..repeated.. ()(){}[] ;
% defaults
  <> (<m>)(<m>){}[] ..repeated.. ()(){}[] ;
```

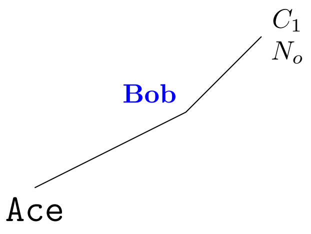
This is a *semicolon version* macro. The quadruple `(<coor>)(<name>){<label>}[<angle>]` is the whole repeating pattern. It is required to type a *semicolon* '`;`' to indicate when the coordinate repetition ends.

```
\tzcoors (0,0)(A) (1,1)(B) (2,1)(C) (3,0)(D) ; % works like:
  \path (0,0) coordinate (A)
        (1,1) coordinate (B)
        (2,1) coordinate (C)
        (3,0) coordinate (D);
```

```
\tzcoors (0,0)(A) (1,1)(B) (2,1)(C){C}[0] (3,0)(D){D}[90]; % works like:
  \path (0,0) coordinate                (A)
        (1,1) coordinate                (B)
        (2,1) coordinate [label={0:C}]  (C)
        (3,0) coordinate [label={90:D}] (D);
```

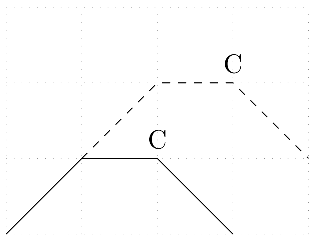
You can add a label to each specified coordinate by adding the optional arguments `{<label>}` and `[<angle>]` immediately after `(<name>)`.

```
% \tzcoors
\begin{tikzpicture}
\tzcoors (0,0)(A){Ace}[[font=\LARGE\ttfamily]-90]
         (2,1)(B){\textbf{Bob}}[[blue]135]
         (3,2)(C){$C_1$\\$N_o$}[[align=center]0];
\draw (A) -- (B) -- (C);
\end{tikzpicture}
```



By the option `<shift coor>`, all specified coordinates are shifted. The empty shift option `<>` is not allowed.

```
% \tzcoors
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoors (0,0)(A) (1,1)(B) (2,1)(C){C} (3,0)(D);
\tzlines(A)(B)(C)(D);
% shift
\tzcoors <1,1> (0,0)(A) (1,1)(B) (2,1)(C){C} (3,0)(D);
\tzlines[dashed](A)(B)(C)(D);
\end{tikzpicture}
```



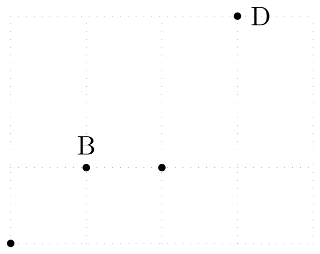
8.2.2 \tzcoors*

The starred version `\tzcoors*` takes an arbitrary number of coordinates as arguments to print node dots at the coordinates. Full repeating pattern is `(<coor>)(<name>){<label>}[<angle>]`. It is required to type a *semicolon* '`;`' to indicate when the iteration of coordinates ends.

```
% syntax: minimum
\tzcoors*(<coor>)(<name>)..repeated..(<coor>)(<name>) ;
% syntax: full
\tzcoors*[<dot opt>]<shift coor>(<coor>)(<name>){<label>}[<[label opt]angle>]
                                  ..repeated.. ()(){}[] ; (<dot size>)
% defaults
 *[]<> (<m>)(<m>){}[] ..repeated.. ()(){}[] ; (2.4pt)
```

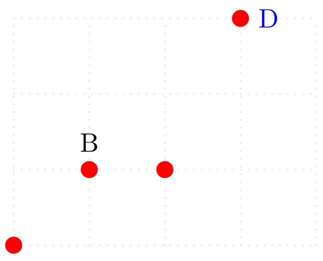
You can label each dot by specifying the optional arguments `{<label>}` and `[<angle>]` after the pair `(<coor>)(<name>)`.

```
% \tzcoors*
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoors*(0,0)(A)
         (1,1)(B){B}
         (2,1)(C)
         (3,3)(D){D}[0] ;
\end{tikzpicture}
```



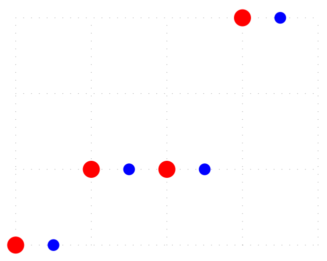
You can change the dot color by `[<dot opt>]` and the label color by `[<label opt>]`. You can apply the THREE WAYS (on page 41) to change the dot size. The simplest way of changing the dot size is to specify the *last* (even after the semicolon) parenthesis option `(<dot size>)`.

```
% \tzcoors*
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoors*[red](0,0)(A)
         (1,1)(B){B}
         (2,1)(C)
         (3,3)(D){D}[[blue]0] ; (6pt)
\end{tikzpicture}
```



By specifying the optional argument `<shift coor>` immediately before the first coordinate, you can move all specified coordinates. The empty shift option `<>` is not allowed.

```
% \tzcoors*: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\settzdotsize{6pt}
\tzcoors*[red]       (0,0)(A)(1,1)(B)(2,1)(C)(3,3)(D);
\settzdotsize{4pt}
\tzcoors*[blue]<.5,0>(0,0)(A)(1,1)(B)(2,1)(C)(3,3)(D);
\end{tikzpicture}
```



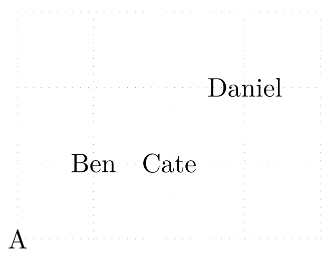
8.3 \tzcoorsquick and \tzcoorsquick*

8.3.1 \tzcoorsquick

You can see the coordinate array at a glance using `\tzcoorsquick`, which displays specified names as text at the `center` of the coordinates, by default.

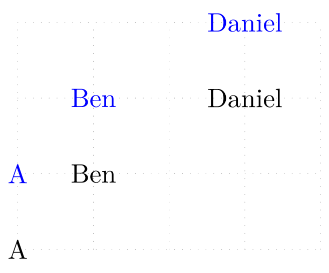
```
% syntax: minimum
\tzcoorsquick(<coor>)(<name>)..repeated..(<coor>)(<name>) ;
% syntax: full
\tzcoorsquick<shift coor>(<coor>)(<name>){<label>}[<[label opt]angle>]
                                    ..repeated.. ()(){}[] ;
% defaults
  <> (<m>)(<m>){}[center] ..repeated.. ()(){}[] ;
```

```
% \tzcoorsquick
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoorsquick(0,0)(A)
             (1,1)(Ben)
             (2,1)(Cate)
             (3,2)(Daniel);
\end{tikzpicture}
```



A label can be suppressed by the empty braces `{}`. You can move the coordinates by specifying `<shift coor>` immediately before the first coordinate. The empty option `<>` is not allowed.

```
% \tzcoorsquick: shift, color
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoorsquick
  (0,0)(A) (1,1)(Ben) (2,1)(Cate){} (3,2)(Daniel);
\tikzset{every label/.style={blue}}
\tzcoorsquick <0,1>
  (0,0)(A) (1,1)(Ben) (2,1)(Cate){} (3,2)(Daniel);
\end{tikzpicture}
```

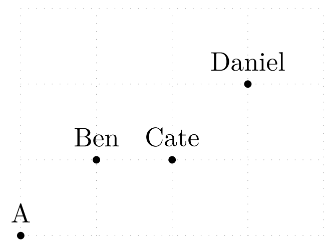


8.3.2 \tzcoorsquick*

The starred version `\tzcoorsquick*` prints node dots on the coordinates and displays the names `above` (or 90 degree from) the dots, by default.

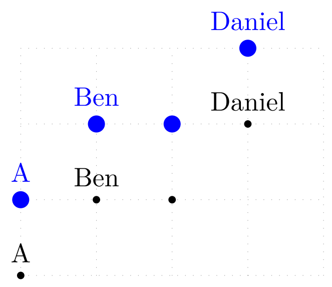
```
% syntax: minimum
\tzcoorsquick*(<coor>)(<name>)..repeated..(<coor>)(<name>) ;
% syntax: full
\tzcoorsquick*[<dot opt>]<shift coor>(<coor>)(<name>){<label>}[<[label opt]angle>]
                                        ..repeated.. ()(){}[] ;
% defaults
 *[ tzdot=1.2pt ]<> (<m>)(<m>){}[] ..repeated.. ()(){}[] ;
```

```
% \tzcoorsquick*
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoorsquick*(0,0)(A)    (1,1)(Ben)
              (2,1)(Cate) (3,2)(Daniel);
\end{tikzpicture}
```



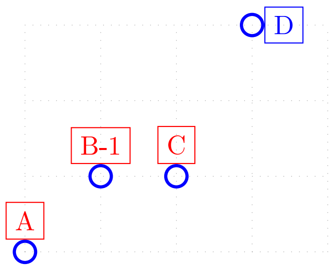
A label can be suppressed by the empty braces {}. You can change the dot size using the THREE WAYS (on page 41). You can shift the coordinate by specifying <shift coor> immediately before the first coordinate. The empty shift option <> is not allowed.

```
% \tzcoorsquick*: size, color, shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoorsquick*
  (0,0)(A) (1,1)(Ben) (2,1)(Cate){} (3,2)(Daniel);
\tikzset{every label/.style={blue}}
\tzcoorsquick*[blue]<0,1>
  (0,0)(A) (1,1)(Ben) (2,1)(Cate){} (3,2)(Daniel);(6pt)
\end{tikzpicture}
```



Remark: The first optional argument of \tzcoorsquick* is for only dots. You can use the Ti*k*Z option every label/.style={...} to control all the labels together. You can also control each labels using [<label opt>] for each coordinate.

```
% \tzcoorsquick*
\begin{tikzpicture}[every label/.style={draw,red}]
\tzhelplines(4,3)
\tzcoorsquick*[fill=none,blue,very thick]
  (0,0)(A)(1,1)(B-1)(2,1)(C)(3,3)(D)[[blue]0];(8pt)
\end{tikzpicture}
```

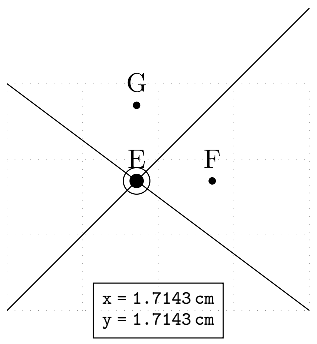


8.4 \tzgetxyval

\tzgetxyval extracts the values of x-coordinate and the y-coordinate *in the unit of centimeter* from a specified coordinate and saves the values in the user-defined macros, so that you can use them later. For example, \tzgetxyval(3,2){\xval}{\yval} results in \xval=3 and \yval=2.

```
% syntax
\tzgetxyval(<coor>){<\macroXval>}{<\macroYval>}
% default
(<m>){<m>}{<m>}
```

```
\begin{tikzpicture}
\tzhelplines(4,3)
\tzfn"dem"{3-(3/4)*\x}[0:4]
\tzfn"sup"{\x}[0:4]
% intersection point: (E)
\tzXpoint*{dem}{sup}(E)(5pt)
% extract x/y-coordinate from (E)
\tzgetxyval(E){\Ex}{\Ey}
\tzdots*(\Ex,\Ey){E}(\Ex+1,\Ey){F}(\Ex,\Ey+1){G};
\tzdot(E)(10pt)
\tznodeframe(2,0){x\;=\;\Ex\,cm\\ y\;=\;\Ey\,cm}
  [font=\ttfamily\scriptsize,align=left]
\end{tikzpicture}
```



9 Plot Coordinates: \tzplot: Semicolon Versions

9.1 \tzplot and \tzplot*: Syntax

\tzplot takes an arbitrary number of coordinates as arguments. Each (<coor>) can be followed by the optional arguments {<label>} and [<angle>] to label the coordinate. This is a *semicolon version* and the whole repeating pattern is the triple (<coor>){<label>}[<angle>]. It is required to type a semicolon ';' to indicate when the coordinate iteration ends.

The macro \tzplot draws connected line segments that link specified coordinates.

```
% syntax: minimum
\tzplot(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: medium
\tzplot(<coor>){<label>}[<angle>]..repeated..(<coor>){<label>}[<angle>] ;
% syntax:full
\tzplot[<opt>]{<tension>} [<plot opt>]<shift coor>"<path name>"
       (<coor>){<label>}[<[label opt]angle>]
       ..repeated..
       (){}[] ; (<mark size>) <code.append>
% defaults
  [tzmark=2pt]{0}[smooth]  <>"" (<m>){}[] ..repeated.. (){}[] ; (2pt) <>
```

The style tzmark is defined as follows:

```
% style: tzmark
\tikzset{
  tzmark/.style=
    {mark options={solid,thin},mark size=#1},
  tzmark/.default=\tzmarksize
}
```

\tzmarksize is the *radius* of a mark and the default is 2pt as in Ti*k*Z. The value of \tzmarksize can be changed by the macro \settzmarksize, like \settzmarksize{3pt}.

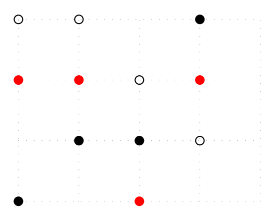
The starred version \tzplot* prints dot marks at specified coordinates, without drawing line segments connecting the coordinates, by default.

9.2 \tzplot*: Dots and marks

The starred version \tzplot* prints Ti*k*Z marks (* by default) at specified coordinates. You can change the mark color and mark style using the first bracket optional argument.

```
\tzplot*(0,0)(1,2)(2,2)(3,3); % works like:
  \draw [draw=none,mark=*] plot coordinates { (0,0)(1,1)(2,2)(3,3) } ;
```

```
% \tzplot*
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot*(0,0)(1,1)(2,1)(3,3);
\tzplot*[mark=o](0,3)(1,3)(2,2)(3,1);
\tzplot*[red](0,2)(1,2)(2,0)(3,2);
\end{tikzpicture}
```



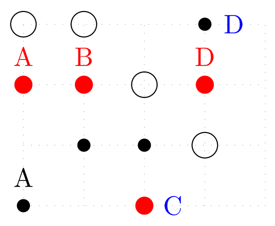
Labels, marks, and mark size You can also add labels to specified coordinates with the optional arguments `{<label>}` and `[<angle>]` immediately after each `(<coor>)`.

```
\tzplot*(0,0){A}[90](1,2)(2,2)(3,3){D}[0]; % works like:
  \draw [draw=none,mark=*] plot coordinates { (0,0)(1,1)(2,2)(3,3) }
                           node (0,0) [label={90:D}] {}
                           node (3,3) [label={0:D}] {} ;
```

There are three ways to change the mark size.

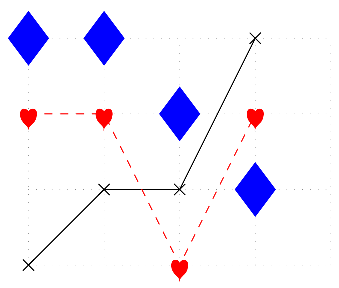
1. The simplest way is to use the parenthesis optional argument `(<mark size>)`, *immediately after the semicolon.*
2. You can use the style `tzmark`, like `tzmark=3pt`.
3. You can also use the macro `\settzmarksize`, which is effective until the end of `tikzpicture` environment.

```
% \tzplot*: label, size
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot*(0,0){A}(1,1)(2,1)(3,3){D}[[blue]0];(1mm)
\tzplot*[mark=o,tzmark=6pt](0,3)(1,3)(2,2)(3,1);
\settzmarksize{4pt}
\tzplot*[red](0,2){A}(1,2){B}(2,0){C}[[blue]0](3,2){D};
\end{tikzpicture}
```



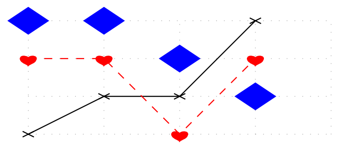
With `\tzplot*`, you can draw line segments by giving the Ti*k*Z's option `draw` in the first bracket optional argument, like `\tzplot*[draw]`.

```
% \tzplot*: line, more marks, size
\begin{tikzpicture}[scale=1]
\tzhelplines(4,3)
\settzmarksize{3pt}
\tzplot*[draw,mark=x](0,0)(1,1)(2,1)(3,3);
\tzplot*[blue,mark=diamond*](0,3)(1,3)(2,2)(3,1);(10pt)
\tzplot*[draw,dashed,red,mark=heart](0,2)(1,2)(2,0)(3,2);
\end{tikzpicture}
```



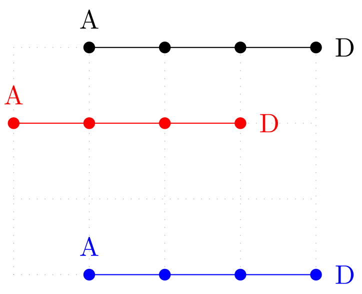
Remark: In Ti*k*Z, the `mark` shapes are affected by `scale`, `xscale`, and `yscale`.

```
% \tzplot*: marks: distorted
\begin{tikzpicture}[yscale=.5]
\tzhelplines(4,3)
\settzmarksize{3pt}
\tzplot*[draw,mark=x](0,0)(1,1)(2,1)(3,3);
\tzplot*[blue,mark=diamond*](0,3)(1,3)(2,2)(3,1);(10pt)
\tzplot*[draw,dashed,red,mark=heart](0,2)(1,2)(2,0)(3,2);
\end{tikzpicture}
```



Shift You can move specified coordinates using the option `<shift coor>` before the first coordinate (to be precise, immediately before the option `"<path name>"` if it exists). The empty shift option `<>` is not allowed.

```
% \tzplot*: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplot*[draw]           (1,3){A}(2,3)(3,3)(4,3){D}[0];
\tzplot*[draw,red]<-1,-1>(1,3){A}(2,3)(3,3)(4,3){D}[0];
\tzplot*[draw,blue]<0,-3>(1,3){A}(2,3)(3,3)(4,3){D}[0];
\end{tikzpicture}
```



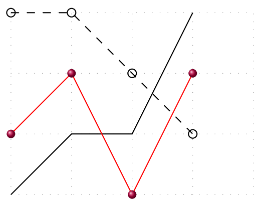
9.3 \tzplot: Lines

`\tzplot` draws connected line segments connecting specified coordinates. (By default, `tension=0`.)

```
\tzplot(0,0)(1,2)(2,2)(3,3); % works like:
  \draw [tension=0] plot [smooth] coordinates { (0,0)(1,1)(2,2)(3,3) } ;
```

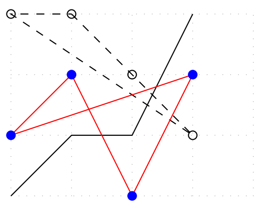
Options: draw, mark, mark options, etc. You can use the optional argument `[<opt>]` to change the style of lines and marks.

```
% \tzplot: line
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot        (0,0)(1,1)(2,1)(3,3);
\tzplot[dashed,mark=o](0,3)(1,3)(2,2)(3,1);
\tzplot[red,mark=ball,mark options={ball color=purple}]
               (0,1)(1,2)(2,0)(3,2);
\end{tikzpicture}
```



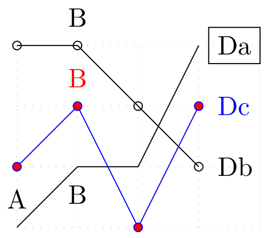
To close the path of `\tzplot` you can use the option `smooth cycle` in the first bracket optional argument `[<opt>]` or in the second bracket optional argument `[<plot opt>]`.

```
% \tzplot: smooth cycle
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot        (0,0)(1,1)(2,1)(3,3);
\tzplot[dashed,mark=o,smooth cycle]
        (0,3)(1,3)(2,2)(3,1);
\tzplot[red,mark=*,mark options={blue}]
        [smooth cycle] %%
        (0,1)(1,2)(2,0)(3,2);
\end{tikzpicture}
```



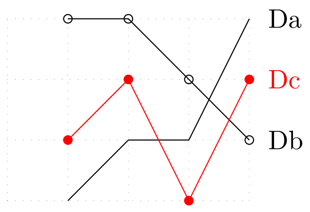
Labels You can label specified coordinates with the options `{<label>}` and `[<angle>]` immediately after each `(<coor>)`.

```
% \tzplot: label
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot(0,0){A}(1,1){B}[-90](2,1)(3,3){Da}[[draw]0];
\tzplot[mark=o](0,3)(1,3){B}(2,2)(3,1){Db}[0];
\tzplot[red,mark=*,draw=blue]
  (0,1)(1,2){B}(2,0)(3,2){Dc}[[blue]0];
\end{tikzpicture}
```



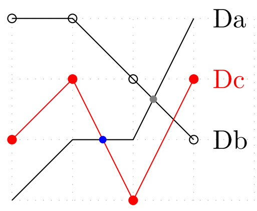
Shift You can also move the line segments by specifying the option `<shift coor>` before the first coordinate (to be precise, immediately before the option `"<path name>"` if it exists). The empty shift option `<>` is not allowed.

```
% \tzplot: shift
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot          <1,0>(0,0)(1,1)(2,1)(3,3){Da}[0];
\tzplot[mark=o]    <1,0>(0,3)(1,3)(2,2)(3,1){Db}[0];
\tzplot*[draw,red]<1,0>(0,1)(1,2)(2,0)(3,2){Dc}[0];
\end{tikzpicture}
```



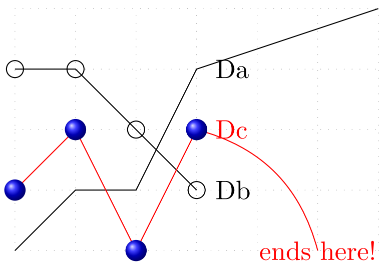
`name path` for intersections To find the intersection points of two lines, you may want to name the paths first, like `[name path=<path name>]` in Ti*k*Z. With `\tzplot`, you can do it by specifying the quote optional argument `"<path name>"` immediately before the first coordinate.

```
% \tzplot: "path name"
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot[name path=Da] (0,0)(1,1)(2,1)(3,3){Da}[0];
\tzplot[mark=o]    "Db"(0,3)(1,3)(2,2)(3,1){Db}[0];
\tzplot*[draw,red]"Dc"(0,1)(1,2)(2,0)(3,2){Dc}[0];
% intersection points
\tzXpoint*[gray]{Da}{Db}
\tzXpoint*[blue]{Da}{Dc}
\end{tikzpicture}
```



Extending the path In order to *extend a path*, formed by `\tzplot`, *from the last coordinate*, you can write Ti*k*Z code in the very last optional argument `<code.append>`, *after the semicolon*.

```
% \tzplot: "path name"
\begin{tikzpicture}[scale=.8]
\tzhelplines(6,4)
\tzplot(0,0)(1,1)(2,1)(3,3){Da}[0];< -- (6,4) >
\tzplot[mark=o](0,3)(1,3)(2,2)(3,1){Db}[0];(4pt)
\tzplot[mark=ball,red](0,1)(1,2)(2,0)(3,2){Dc}[0];
  (5pt)< to [bend left] ++(2,-2) node {ends here!} >
\end{tikzpicture}
```

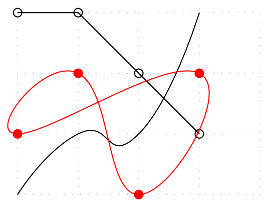


9.4 \tzplot: Curves

With `\tzplot`, the default value of `tension` is `0`. You can draw a curve with `\tzplot`, by specifying the optional argument `{<tension>}` before the coordinates or between the first and second bracket options (if they exist).

```
\tzplot[blue,smooth cycle]{1}(0,0)(1,2)(2,2)(3,3); % works like:
  \draw [blue,tension=1] plot [smooth cycle] coordinates { (0,0)(1,1)(2,2)(3,3) } ;
```

```
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot[tension=1](0,0)(1,1)(2,1)(3,3);
\tzplot[mark=o]    (0,3)(1,3)(2,2)(3,1); % tension=0
\tzplot*[draw,red,smooth cycle]{1}
                   (0,1)(1,2)(2,0)(3,2);
\end{tikzpicture}
```

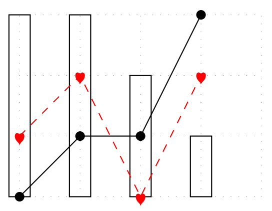


To plot curves, the macro `\tzplotcurve` is provided. Basically, `\tzplotcurve` is the `tension=1` version of `\tzplot`. (See Section 9.6 on page 58.)

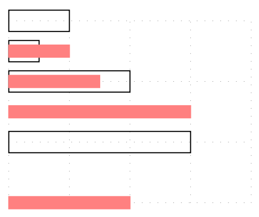
9.5 \tzplot: Bars and combs

With `\tzplot`, you can draw bars or combs, using the Ti*k*Z options `ybar`, `xbar`, `ycomb`, and `xcomb`.

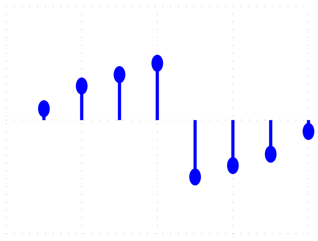
```
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot*[draw](0,0)(1,1)(2,1)(3,3);
\tzplot[ybar](0,3)(1,3)(2,2)(3,1);
\tzplot[dashed,red,mark=heart](0,1)(1,2)(2,0)(3,2);%(5pt)
\end{tikzpicture}
```



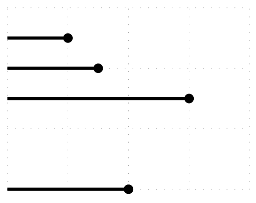
```
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot[xbar](.5,2.5)(1,3)(2,2)(3,1);
\tzplot[xbar,bar width=2mm,fill,red!50]
       (1,2.5)(1.5,2)(2,0)(3,1.5);(3pt)
\end{tikzpicture}
```



```
\begin{tikzpicture}[yscale=1.5]
\tzhelplines(0,-1)(4,1)
\tzplot[ycomb,mark=*,very thick,blue]
        (.5,.1)(1,.3)(1.5,.4)(2,.5)
        (4,-.1)(3.5,-.3)(3,-.4)(2.5,-.5);
\end{tikzpicture}
```



```
\begin{tikzpicture}[scale=.8]
\tzhelplines(4,3)
\tzplot[xcomb,mark=*,very thick]
        (1,2.5)(1.5,2)(2,0)(3,1.5);
\end{tikzpicture}
```



Remark:

- Do not use `<shift coor>` for plotting bars or combs to avoid getting unexpected results. It gives you wrong bars because `<shift coor>` moves coordinates but not bars.
- It can be a mess when using the Ti*k*Z option `shift={(coor)}` with the type of mixed coordinates: native and named coordinates.

9.6 `\tzplotcurve(*)`

`\tzplotcurve` draws a curve connecting specified coordinates with `tension=1`, by default. Basically, it is equivalent to `\tzplot` with `[tension=1]`.

The starred version `\tzplotcurve*` draws a curve and displays marks `*`, by default. Basically, this is equivalent to `\tzpolot*[draw,tension=1]`.

```
% syntax: minimum
\tzplotcurve(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: full
\tzplotcurve*[<opt>]{<tension>} [<plot opt>]<shift coor>"<path name>"
              (<coor>){<label>}[<[label opt]angle>]
              ..repeated..
              (){}[] ; (<mark size>) <code.append>
% defaults
 *[tzmark=2pt]{1}[smooth] <>"" (<m>){}[] ..repeated.. (){}[] ; (2pt) <>
```

```
\tzplotcurve(0,0)(1,2)(2,2)(3,3); % works like:
  \draw [tension=1] plot [smooth] coordinates { (0,0)(1,1)(2,2)(3,3) } ;
```

```
\tzplotcurve[blue,smooth cycle]{2}(0,0)(1,2)(2,2)(3,3); % works like:
  \draw [blue,tension=2] plot [smooth cycle] coordinates { (0,0)(1,1)(2,2)(3,3) } ;
```

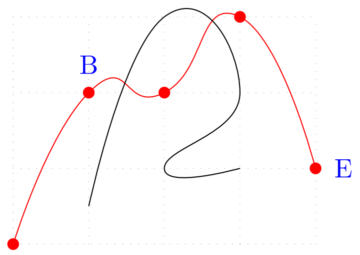
Since `\tzplotcurve` is a *semicolon version*, you need to enter a semicolon to indicate when the coordinate iteration ends. In repeating coordinates, each mandatory coordinate can have a label. So the whole repeating pattern is the triple `(<coor>){<text>}[<pos>]`. For example, `(A){here}[above]` represents `(A) node [above] {here}` in Ti*k*Z.

Options: lines, labels, colors, smooth cycle Use the first bracket option to control the colors of lines or labels.

```
\tzplotcurve(0,0)(1,2)(2,2){A}[below](3,3){B}[right]; % works like:
  \draw [tension=1] plot [smooth] coordinates { (0,0)(1,1)(2,2)(3,3) }
        (2,2) node [below] {A}
        (3,3) node [right] {B} ;
```

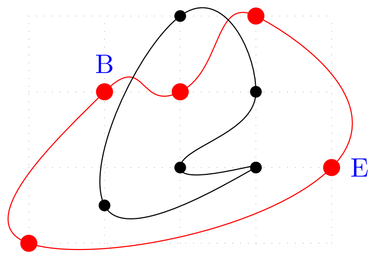
You can change the color of all labels together by adding `[text=<color>]` to the first bracket option list.

```
% \tzplotcurve(*)
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplotcurve*[red,text=blue]
  (0,0)(1,2){B}(2,2)(3,3)(4,1){E}[0];
\tzplotcurve(1,.5)(2,3)(3,2)(2,1)(3,1);
\end{tikzpicture}
```



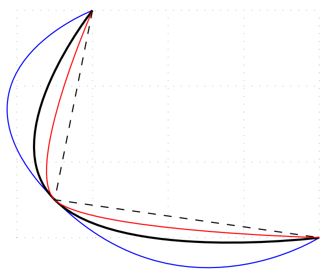
The simplest way to change the mark size is to specify `(<mark size>)` immediately after the semicolon `;`. To close the path of `\tzplotcurve`, you can use the Ti*k*Z option `smooth cycle` in the first bracket option or in the second bracket option.

```
% \tzplotcurve(*): smooth cycle, mark size
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplotcurve*[red,text=blue][smooth cycle]
  (0,0)(1,2){B}(2,2)(3,3)(4,1){E}[0];(3pt) %%
\tzplotcurve*[smooth cycle](1,.5)(2,3)(3,2)(2,1)(3,1);
\end{tikzpicture}
```



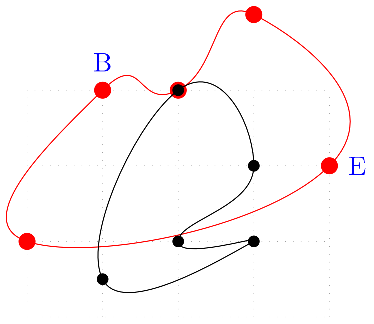
Tension You can change the value of `tension` (`tension=1` by default) by specifying the option `{<tension>}` before the coordinates or between the two bracket options if they exist.

```
% \tzplotcurve: tension
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcoor(.5,.5)(A)
\tzplotcurve[blue]{2}(1,3)(A)(4,0);
\tzplotcurve[thick](1,3)(A)(4,0); % default: tension=1
\tzplotcurve[red]{.5}(1,3)(A)(4,0);
\tzplotcurve[dashed]{0}(1,3)(A)(4,0);
\end{tikzpicture}
```



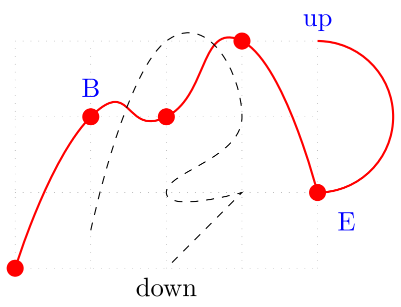
Shift Use the optional argument `<shift coor>` before the first coordinate (to be precise, immediately before `"<path name>"`, if exists). The empty shift option `<>` is not allowed.

```
% \tzplotcurve(*): shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplotcurve*[red,text=blue,smooth cycle]
  <0,1>(0,0)(1,2){B}(2,2)(3,3)(4,1){E}[0];(3pt) %%
\tzplotcurve*[smooth cycle]
  (1,.5)(2,3)(3,2)(2,1)(3,1);
\end{tikzpicture}
```



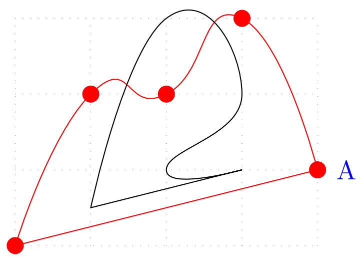
Extending the path In order to extend the path created by `\tzplotcurve(*)` from the last coordinate, you can directly write Ti*k*Z code in the very last optional argument `<code.append>`, after the semicolon.

```
% \tzplotcurve(*): <code.append>
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplotcurve*[red,text=blue,thick]
  (0,0)(1,2){B}(2,2)(3,3)(4,1){E}[-45];(3pt)
  < arc (-90:90:1cm) node [above] {up} > %%
\tzplotcurve[dashed]
  (1,.5)(2,3)(3,2)(2,1)(3,1);
  < -- ++(-1,-1) node [below] {down} >   %%
\end{tikzpicture}
```



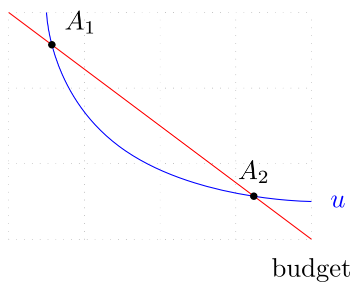
You can also use `<--cycle>` to *close* the path *with a straight line* from the last coordinate to the first coordinate.

```
% \tzplotcurve(*): <--cycle>
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplotcurve*[red,text=blue]
  (0,0)(1,2)(2,2)(3,3)(4,1){A}[right] ; (3pt)<--cycle>
\tzplotcurve
  (1,.5)(2,3)(3,2)(2,1)(3,1) ; <--cycle>
\end{tikzpicture}
```



`name path` for intersection points You can name the path of `\tzplotcurve` by specifying the option `"<path name>"` immediately before the first coordinate.

```
% \tzplotcurve(*): name path
\begin{tikzpicture}
\tzhelplines(4,3)
\tzplotcurve[blue]
  "IC"(.5,3)(1.5,1.2)(4,.5){$u$}[0];
\tzplot[draw=red]
  "bgt"(0,3)(4,0){budget}[-90];
% intersection points
\tzXpoint{IC}{bgt}(K)
\tzdots*(K){$A_1$}[30](K-2){$A_2$};
\end{tikzpicture}
```



10 Nodes

10.1 \tznode and \tznode*

The macro `\tznode` allows you to put text at specified coordinates. You can also optionally name a node so you can refer to the *node coordinate* later.

The starred version `\tznode*` is equivalent to `\tznode[draw]`, which draws the perimeter of the specified node. The default node shape is a `rectangle`.

```
% syntax: minimal
\tznode(<coor>){<text>}
% syntax: full
\tznode[<opt>]<shift coor>(<coor>)(<node name>){<text>}[<node opt>]
% defaults
  []<>(<m>)(){}[]
```

```
\tznode(0,0){text} % works like:
  \path (0,0) node {text};
  % or
  \node at (0,0) {text};
```

```
\tznode[draw] (0,0)(A){text}[above right] % works like:
  \node [draw] (A) at (0,0) [above right] {text};
```

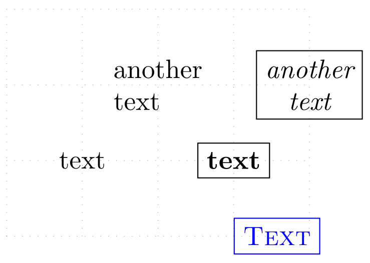
`\tznode*` prints the perimeter of a node, which is a `rectangular` by default.

```
% syntax: minimal
\tznode*(<coor>){<text>}
% syntax: full
\tznode*[<opt>]<shift coor>(<coor>)(<name>){<text>}[<node opt>]<code.append>
% defaults
 *[draw]<>(<m>)(){}[]<>
```

```
\tznode* (0,0)(A){text}[above right] % works like:
  \node [draw] (A) at (0,0) [above right] {text};
```

Putting text You can use Ti*k*Z options in the first bracket optional argument `[<opt>]` or the second bracket option `[<node opt>]` to put text with different colors, fonts, and so on.

```
% \tznode(*)
\begin{tikzpicture}
\tzhelplines(4,3)
\tznode(1,1){text}
\tznode(2,2){another\\text}[align=left]
\tznode*(3,1){\textbf{text}}
\tznode*[blue](3,0){\scshape Text}[r] % right
\tznode*(4,2){another\\text}[align=center,font=\itshape]
\end{tikzpicture}
```

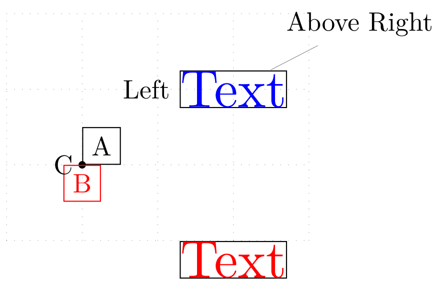


Abbreviations You can use *abbreviations* such as `a` for `above`, `l` for `left`, `ar` for `above right`, `bl` for `below left`, and so on to indicate where the *text* of a *main node* is placed. (See also Section 1.2 on page 1.)

Warning: A *main node* is placed by the placement words, while a *label node* is placed by the placement words or the `<angle>` expression.

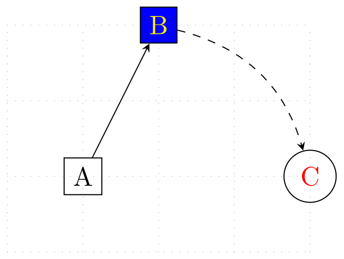
- You CANNOT use the abbreviations to place a *label node* for *points* or *dots* or *coordinates*.
- Instead, use `<angle>` or the original (unabridged) placement words such as `right` and `below left` to place `<label>`.
- Note that, on the other hand, you CANNOT use the `<angle>` option to place a *main node*.

```
% \tznode(*): main node, label node
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdot*(1,1)
\tznode*(1,1){A}[ar]
\tznode*[red](1,1){B}[b]
\tznode(1,1){C}[l]
\tznode*[inner sep=0pt,text=blue,scale=2]
  (3,2){Text}[label=180:Left,pin=45:Above Right]
\tznode*[inner sep=0pt,text=red,scale=2]
  (3,2){Text}[b=2cm] %% below=2cm
\end{tikzpicture}
```



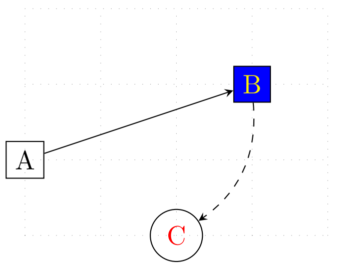
Naming nodes You can name a node at a specified coordinate (`<coor>`) by specifying (`<name>`) immediately after the coordinate. You can use the node name as a *node coordinate*.

```
% \tznode(*): name nodes
\begin{tikzpicture}
\tzhelplines(4,3)
\tznode*(1,1)(A){A}
\tznode*[fill=blue,text=yellow](2,3)(B){B}
\tznode*[circle,text=red](4,1)(C){C}
\tzline[->](A)(B)
\tzto[->,bend left,dashed](B)(C)
\end{tikzpicture}
```



Shift You can move the coordinates by specifying the option `<shift coor>` immediately before the coordinate (`<coor>`). The empty shift option `<>` is not allowed.

```
% \tznode(*): shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tznode*<-1,0>(1,1)(A){A}
\tznode*[fill=blue,text=yellow]<1,-1>(2,3)(B){B}
\tznode*[circle,text=red]<-2,-1>(4,1)(C){C}
\tzline[->](A)(B)
\tzto[->,bend left,dashed](B)(C)
\end{tikzpicture}
```



10.2 `\tznodedot(*)`

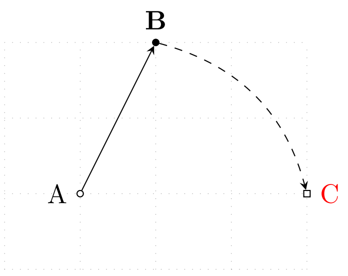
`\tznodedot` names a node and prints a circle node dot. `\tznodedot` is basically the same as `\tzdot`, except for one thing. `\tznodedot` names a node.

The starred version `\tznodedot*` prints a filled circle node dot (of the size `2.4pt`, by default), just like `\tzdot*`. But it optionally names a node.

```
% syntax: medium
\tznodedot(<coor>)(<node name>){<label>}[<angle>]
% syntax: full
\tznodedot*[<opt>]<shift coor>
           (<coor>)(<node name>){<label>}[<[label opt]angle>] (<dot size>)
% defaults
 *[]<>(<m>)(){}[](2.4pt)
```

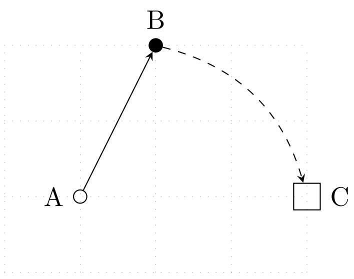
Since `\tznodedot(*)` prints a node dot, its `<label>` is placed by `<angle>`.

```
% \tznodedot(*): label, color
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodedot(1,1)(A){A}[180]
\tznodedot*(2,3)(B){\textbf{B}}
\tznodedot[rectangle](4,1)(C){C}[[red]0]
\tzline[->](A)(B)
\tzto[->,bend left,dashed](B)(C)
\end{tikzpicture}
```



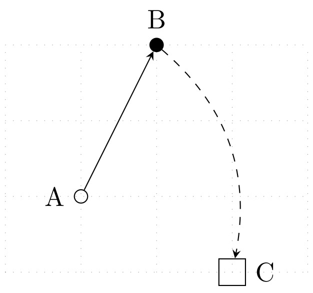
You can apply the Three Ways (on page 41) to change the size of node dots. The simplest way is to use the last parenthesis option (`<dot size>`).

```
% \tznodedot(*): size
\begin{tikzpicture}
\tzhelplines(4,3)
\settzdotsize{5pt} %%
\tznodedot(1,1)(A){A}[180]
\tznodedot*(2,3)(B){B}
\tznodedot[rectangle](4,1)(C){C}[0](10pt) %%
\tzline[->](A)(B)
\tzto[->,bend left,dashed](B)(C)
\end{tikzpicture}
```



You can move the coordinates of dots by specifying the `<shift coor>` option immediately before the coordinate. The empty shift option `<>` is not allowed.

```
% \tznodedot(*): shift
\begin{tikzpicture}
\tzhelplines(4,3)
\settzdotsize{5pt}
\tznodedot(1,1)(A){A}[180]
\tznodedot*(2,3)(B){B}
\tznodedot[rectangle]<-1,-1>(4,1)(C){C}[0](10pt) %%
\tzline[->](A)(B)
\tzto[->,bend left,dashed](B)(C)
\end{tikzpicture}
```

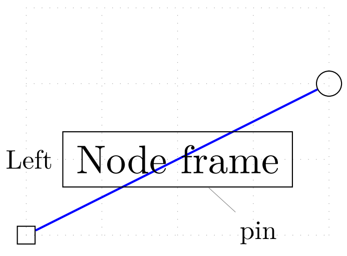


10.3 \tznodeframe and \tznodeframe*

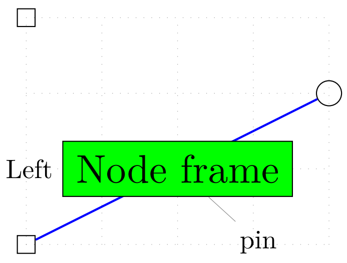
\tznodeframe draws and names a rectangle node with text in it.

```
% syntax:
\tznodeframe[<opt>](<coor>)(<node name>){<text>}[<node opt>]
% defaults
 []<>(<m>)(){}[text=black]
```

```
% \tznodeframe
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodeframe(0,0)(A)
\tznodeframe(4,2)(B)[circle]
\tzline[blue,thick](A)(B)
\tznodeframe[scale=1.5](2,1){Node frame}
            [label=180:Left, pin=-45:pin]
\end{tikzpicture}
```



```
% \tznodeframe: fill (opacity=1)
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodeframe(0,3)
\tznodeframe(0,0)(A)
\tznodeframe(4,2)(B)[circle]
\tzline[blue,thick](A)(B)
\tznodeframe[fill=green,scale=1.5](2,1){Node frame}
            [label=180:Left, pin=-45:pin]
\end{tikzpicture}
```

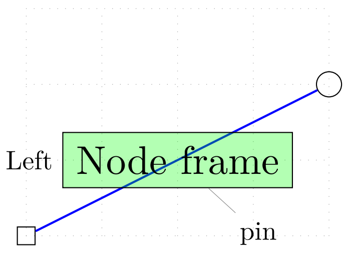


The starred version \tznodeframe* fills the rectangle with color (black!50 by default) with fill opacity=.3 but with text opacity=1, by default.

```
% syntax:
\tznodeframe*[<opt>]<shift coor>
             (<coor>)(<node name>){<text>}[<node opt>]{<fill opacity>}
% defaults
 *[fill=black!50,fill opacity=.3,text opacity=1] <>
  (<m>)(){}[draw=black,text=black]{.3}
```

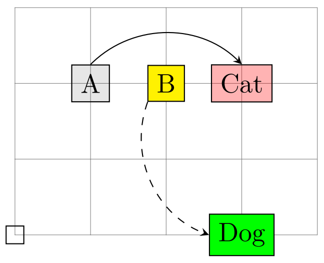
Remark: \tznodeframe works very similar to \tznode, but their 'starred versions' work differently. While \tznode* draws the perimeter of a node, \tznodeframe* fills a node with color.

```
% \tznodeframe*: opacity=.3 (by default)
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodeframe(0,0)(A)
\tznodeframe(4,2)(B)[circle]
\tzline[blue,thick](A)(B)
\tznodeframe*[green,scale=1.5](2,1){Node frame}
             [label=180:Left, pin=-45:pin]
\end{tikzpicture}
```



You can change the `fill opacity` of `\tznodeframe*` by specifying the last curly brace option `{<fill opacity>}`. You can also move the node by specifying the `<shift coor>` option immediately before the coordinate. (The empty shift coor `<>` is not allowed.)

```
% \tznodeframe(*): opacity, shift
\begin{tikzpicture}
\tzhelplines[solid](4,3)
\tznodebox(0,0)
\tznodeframe*(1,2)(A){A}{.1} % opacity
\tznodeframe[fill=yellow](2,2)(B){B}
\tznodeframe*[red](3,2)(C){Cat}
\tznodeframe[fill=green]<-1,-2>(4,2)(D){Dog} % shift
\tzto[->,bend left=45](A.north)(C.90)
\tzto[->,bend right=45,dashed](B.-135)(D.west)
\end{tikzpicture}
```



Remark: In addition to using the option `{<fill opacity>}`, you can use a macro to change the default value.

- The default fill opacity can be changed by the macro `\settzfillopacity`.
- The default fill color is `black!50`. You can use the macro `\settzfillcolor` to change the default fill color.
- With `\tznodeframe`, you can change the color of the perimeter and text with the second bracket option `[<node opt>]`.

10.4 `\tznodecircle` and `\tznodecircle*`

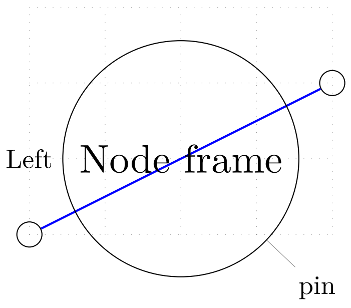
`\tznodecircle` works just like `\tznodeframe` but with a circle node.

```
% syntax
\tznodecircle[<opt>]<shift coor>(<coor>)(<node name>){<text>}[<node opt>]
% defaults
  [circle]<>(<m>)(){}[text=black]
```

`\tznodecircle*` works just like `\tznodeframe*` but with a circle node.

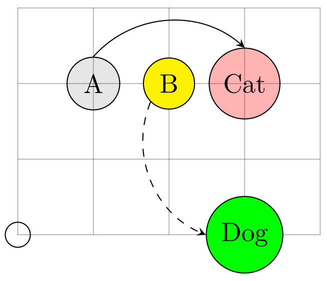
```
% syntax
\tznodecircle*[<opt>]<shift coor>
               (<coor>)(<node name>){<text>}[<node opt>]{<fill opacity>}
% defaults
 *[circle,fill=black!50,fill opacity=.3,text opacity=1] <>
  (<m>)(){}[draw=black,text=black]{.3}
```

```
% \tznodecircle
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodecircle(0,0)(A)
\tznodecircle(4,2)(B)[circle]
\tzline[blue,thick](A)(B)
\tznodecircle[scale=1.5](2,1){Node frame}
             [label=180:Left, pin=-45:pin]
\end{tikzpicture}
```



You can change the `fill opacity` of `\tznodecircle*` by specifying the last curly brace option `{<fill opacity>}` or using `\settzfillopacity`. You can also move the node by specifying the `<shift coor>` option immediately before the coordinate. (The empty shift coor `<>` is not allowed.)

```
% opacity, shift
\begin{tikzpicture}
\tzhelplines[solid](4,3)
\tznodecircle(0,0)
\tznodecircle*(1,2)(A){A}{.1} % opacity
\tznodecircle[fill=yellow](2,2)(B){B}
\tznodecircle*[red](3,2)(C){Cat}
\tznodecircle[fill=green]<-1,-2>(4,2)(D){Dog} % shift
\tzto[->,bend left=45](A.north)(C.90)
\tzto[->,bend right=45,dashed](B.-135)(D.west)
\end{tikzpicture}
```



10.5 `\tznodeellipse` and `\tznodeellipse*`

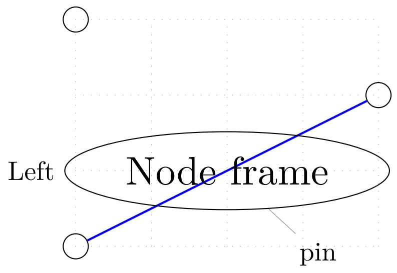
`\tznodeellipse` works just like `\tznodeframe` but with an ellipse node.

```
% syntax:
\tznodeellipse[<opt>]<shift coor>(<coor>)(<node name>){<text>}[<node opt>]
% defaults
  [ellipse]<>(<m>)(){}[text=black]
```

The starred version `\tznodeellipse*` works just like `\tznodeframe*` but with an ellipse node.

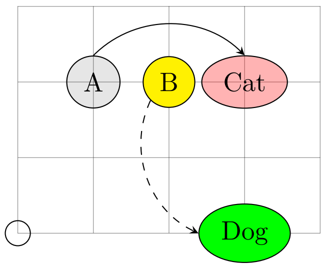
```
% syntax:
\tznodeellipse*[<opt>]<shift coor>
               (<coor>)(<node name>){<text>}[<node opt>]{<fill opacity>}
% defaults
 *[ellipse,fill=black!50,fill opacity=.3,text opacity=1] <>
  (<m>)(){}[draw=black,text=black]{.3}
```

```
% \tznodeellipse
\begin{tikzpicture}
\tzhelplines(4,3)
\tznodeellipse(0,3)
\tznodeellipse(0,0)(A)
\tznodeellipse(4,2)(B)[circle]
\tzline[blue,thick](A)(B)
\tznodeellipse[scale=1.5](2,1){Node frame}
            [label=180:Left, pin=-45:pin]
\end{tikzpicture}
```



You can change the `fill opacity` of `\tznodeellipse*` by specifying the last curly brace option `{<fill opacity>}`. You can also move the node by specifying the `<shift coor>` option immediately before the coordinate. (The empty shift coor `<>` is not allowed.)

```
% opacity, shift
\begin{tikzpicture}
\tzhelplines[solid](4,3)
\tznodeellipse(0,0)
\tznodeellipse*(1,2)(A){A}{.1} % opacity
\tznodeellipse[fill=yellow](2,2)(B){B}
\tznodeellipse*[red](3,2)(C){Cat}
\tznodeellipse[fill=green]<-1,-2>(4,2)(D){Dog} % shift
\tzto[->,bend left=45](A.north)(C.90)
\tzto[->,bend right=45,dashed](B.-135)(D.west)
\end{tikzpicture}
```



11 Lines

11.1 \tzline: Connecting two points

`\tzline` connects two points with a straight line.

```
% syntax: miminum
\tzline(<coor1>)(<coor2>)
% syntax: medium
\tzline[<opt>](<coor1>){<text1>}[<node opt1>]
              (<coor2>){<text2>}[<node opt2>]
% syntax: full
\tzline[<opt>]<shift coor>"<path name>"
       (<coor1>){<text1>}[<node opt1>]
       (<coor2>){<text2>}[<node opt2>]<code.append>
% defaults
  []<>""(<m>){}[above](<m>){}[]<>
```

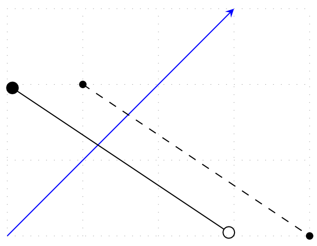
```
\tzline(0,1)(2,1) % works like:
  \draw (0,1) -- (2,1);
```

11.1.1 Line styles

You can use the first optional argument `[<opt>]` to control the line styles.

```
\tzline[blue](0,1)(2,1) % works like:
  \draw [blue] (0,1) -- (2,1);
```

```
% \tzline
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue,->](0,0)(3,3)
\tzline[*-o](0,2)(3,0)
\tzline[dashed](1,2)(4,0)
\tzdots*(1,2)(4,0);
\end{tikzpicture}
```



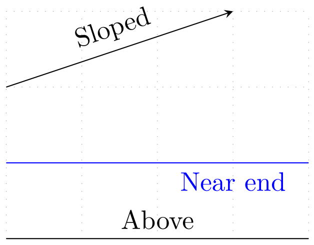
11.1.2 Adding text

You can add text by specifying the optional arguments `{<text>}` and `[<node opt>]`.

Text next to the line With the options `{<text1>}[<node opt1>]` *in-between* two coordinates, you can add text next to the line, with the option `[above,midway]` by default.

```
\tzline[blue](0,1){my line}[sloped](2,1) % works like:
  \draw [blue] (0,1) -- node [above,sloped] {my line} (2,1) ;
```

```
% \tzline: text next to line
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline(0,0){Above}(4,0)                % default [above]
\tzline[blue](0,1){Near end}[b,near end](4,1) % below
\tzline[->](0,2){Sloped}[sloped](3,3) % default [above]
\end{tikzpicture}
```

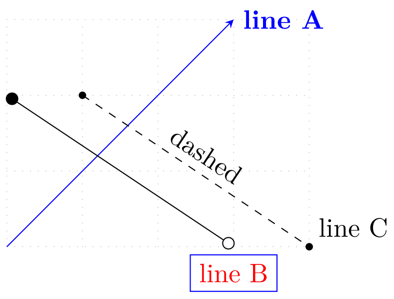


You can use the *abbreviations* of Ti*k*Z basic placement options such as `a` for `above`, `bl` for `below left`, and so on. (For more details, see page 2.)

Text at or around the last coordinate You can also add text at (by default) or around the second coordinate by specifying `{<text2>}` and `[<node opt2>]` immediately *after* the second coordinate.

```
\tzline[blue](0,1)(2,1){my line}[right] % works like:
  \draw [blue] (0,1) -- (2,1) node [right] {my line};
```

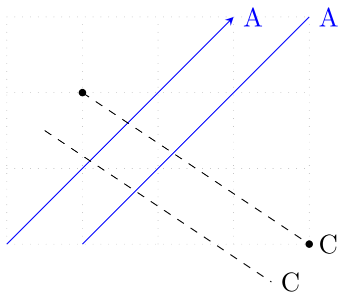
```
% \tzline
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue,->](0,0)(3,3){\textbf{line A}}[r]
\tzline[*-o](0,2)(3,0){line B}[red,draw=blue,b=1mm]
\tzline[dashed](1,2){dashed}[sloped](4,0){line C}[ar]
\tzdots*(1,2)(4,0);
\end{tikzpicture}
```



11.1.3 Moving the line: shift

You can move the line generated by `\tzline` by specifying the option `<shift coor>` before the first coordinate (to be precise, immediately before the option `"<path name>"`, which is put immediately before the first coordinate, if it exists). The empty shift coor `<>` is not allowed.

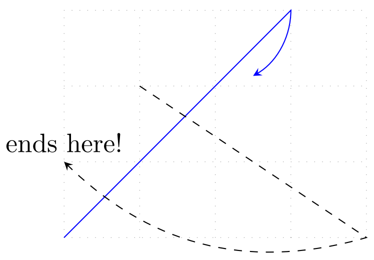
```
% \tzline: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue,->]     (0,0)(3,3){A}[r]
\tzline[blue]    <1,0>(0,0)(3,3){A}[r]
\tzline[dashed]         (1,2)(4,0){C}[r]
\tzline[dashed]<-.5,-.5>(1,2)(4,0){C}[r]
\tzdots*(1,2)(4,0);
\end{tikzpicture}
```



11.1.4 Extending the path

You can extend the path of `\tzline` from the last coordinate, by writing Ti*k*Z code in the last optional argument `<code.append>`.

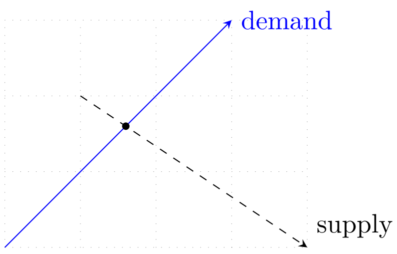
```
% \tzline: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue,->](0,0)(3,3)<arc(0:-60:1cm)>
\tzline[dashed,->](1,2)(4,0)
       <to[bend left] ++(-4,1) node [a] {ends here!}>
\end{tikzpicture}
```



11.1.5 Naming the path: Intersection points

When you specify the option `"<path name>"` immediately before the first coordinate, it works like `[name path=<path name>]` in the option list of `[<opt>]`. You can use this name of path to find intersection points.

```
% \tzline+: name path
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[blue,->]  "dem" (0,0)(3,3){demand}[r]
\tzline[dashed,->]"supp"(1,2)(4,0){supply}[ar]
% intersection point
\tzXpoint*{dem}{supp}
\end{tikzpicture}
```



11.2 `\tzline+`: Relative coordinates

The *plus version* `\tzline+` takes the second coordinate (`<coor2>`) relative (with `++`) to the first coordinate (`<coor1>`).

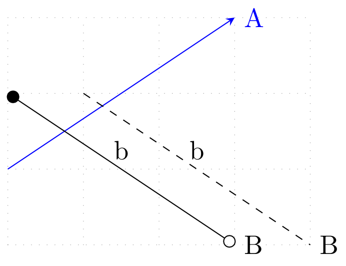
Everything else is the same as in `\tzline`.

```
% syntax: full
\tzline+[<opt>]<shift coor>"<path name>"
        (<coor1>){<text1>}[<node opt1>]
        (<coor2>){<text2>}[<node opt2>]<code.append>
% defaults
 +[]<>""(<m>){}[above](<m>){}[]<>
```

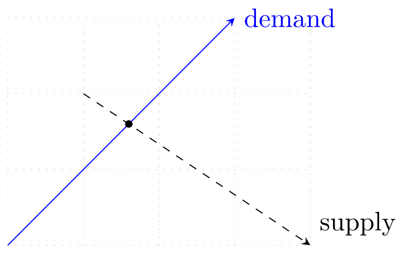
```
\tzline+(0,1)(2,1) % works like:
  \draw (0,1) -- ++ (2,1);
```

```
\tzline+[dashed]"AA"(0,1){A}[red](2,1){B}[right,blue] % works like:
  \draw [dashed,name path=AA]
        (0,1) -- node [red] {A} ++ (2,1) node [right,blue] {B};
```

```
% \tzline+
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline+[blue,->](0,1)(3,2) {A}[r]
\tzline+[*-o]        (0,2){b}(3,-2){B}[r]
\tzline+[dashed]<1,0>(0,2){b}(3,-2){B}[r]
\end{tikzpicture}
```



```
% \tzline+: name path
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline+[blue,->]  "dem" (0,0)(3,3){demand}[r]
\tzline+[dashed,->]"supp"(1,2)(3,-2){supply}[ar]
% intersection point
\tzXpoint*{dem}{supp}
\end{tikzpicture}
```



11.3 Styles: tzshorten and tzextend

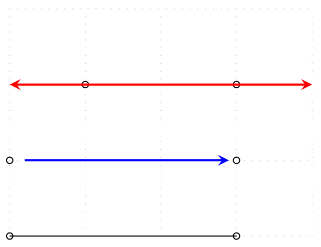
The styles `tzshorten` and `tzextend` are defined as follows.

```
% tzshorten
\tikzset{%
  tzshorten/.style 2 args ={shorten <=#1, shorten >=#2},
  tzshoretn/.default={2pt}{2pt}
}

% tzextend (negative tzshorten)
\tikzset{%
  tzextend/.style 2 args ={shorten <=-#1, shorten >=-#2},
  tzextend/.default={2pt}{2pt}
}
```

For example, `[tzshorten={2mm}{1mm}]` is equivalent to `[shorten <=2mm,shorten >=1mm]` in Ti*k*Z. Simple `[tzshorten]` means that `[tzshorten={2pt}{2pt}]` by default.

The style `tzextend` is a negative `tzshorten`. For example, `tzextend{2mm}{1mm}` is equivalent to `tzshorten{-2mm}{-1mm}`.

```
% tzshorten, tzextend
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdots(0,0)(3,0);  \tzline(0,0)(3,0)
\tzdots(0,1)(3,1);
\tzline[->,thick,blue,tzshorten={2mm}{1mm}](0,1)(3,1)
\tzdots(1,2)(3,2);
\tzline[<->,thick,red,tzextend={1cm}{1cm}](1,2)(3,2)
\end{tikzpicture}
```



11.4 \tzlines: Connecting multiple points: Semicolon version

`\tzlines` connects two or more points with connected line segments. Since `\tzlines` takes an arbitrary number of coordinates as arguments, it is a *semicolon version*. So, you need to enter a semicolon ‘;’ to indicate when the coordinate iteration ends.

```
% syntax: minimum
\tzlines(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: medium
\tzlines(<coor>){<text>}[<node opt>]..repeatd..(<coor>){<text>}[<node opt>] ;
% syntax: full
\tzlines[<opt>]<shift coor>"<path name>"
        (<coor>){<text>}[<node opt>]
        ..repeated.. (){}[] ; <code.append>
% defaults
  []<>"" (<m>){}[] ..repeated.. (){}[] ; <>
```

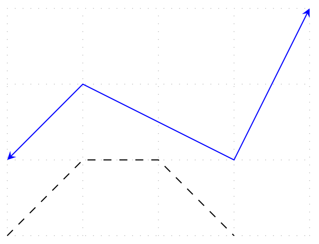
```
\tzlines(1,1)(2,2)(3,1)(4,3); % works like:
  \draw (1,1) -- (2,2) -- (3,1) -- (4,3);
```

The whole repeating pattern in `\tzlines` is the triple `(<coor>){<text>}[<node opt>]`.

```
\tzlines(1,1)(2,2)(3,1){C}(4,3){D}[r]; % works like:
  \draw (1,1) -- (2,2) -- (3,1) -- node {C} (4,3) node [right] {D};
```

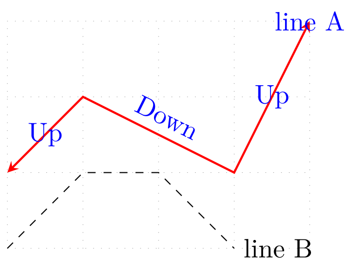
Line styles Use the first optional argument `[<opt>]` to control the style of the connected line drawn by `\tzlines`.

```
% \tzlines: line style
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines[blue,<->](0,1)(1,2)(3,1)(4,3);
\tzlines[dashed](0,0)(1,1)(2,1)(3,0);
\end{tikzpicture}
```



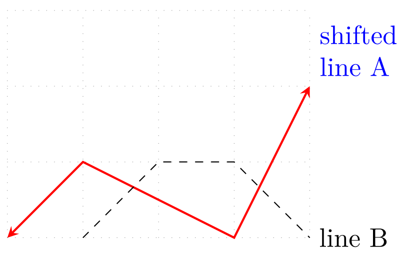
Adding text You can add text in the `midway`, by default, of each line segment by specifying the options `{<text>}` and `[<node opt>]` immediately after each coordinate (except the last one). The options `{<text>}` and `[<node opt>]` after the last coordinate put `<text>` at or around the last coordinate.

```
% \tzlines: adding text
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines[red,thick,<->,text=blue]
        (0,1){Up}
        (1,2){Down}[a,sloped]
        (3,1){Up}
        (4,3){line A} ;
\tzlines[dashed](0,0)(1,1)(2,1)(3,0){line B}[r];
\end{tikzpicture}
```



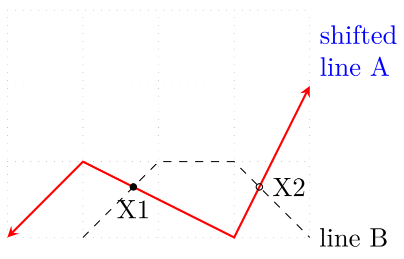
Shift You can move the connected line by specifying `<shift coor>` before the first coordinate or immediately before the option `"<path name>"` if it exists. (The empty shift option `<>` is not allowed.)

```
% \tzlines: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines[red,thick,<->,text=blue]
  <0,-1>(0,1)
        (1,2)
        (3,1)
        (4,3){shifted\\line A}[align=left,ar] ;
\tzlines[dashed]<1,0>(0,0)(1,1)(2,1)(3,0){line B}[r] ;
\end{tikzpicture}
```



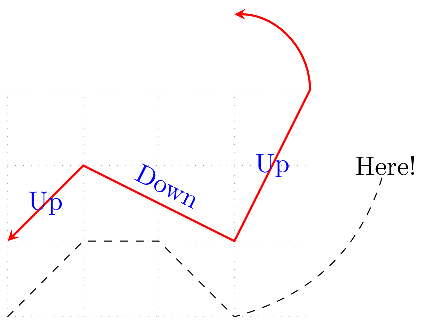
Naming paths You can name the path of `\tzlines` by specifying the option `"<path name>"` immediately before the first coordinate.

```
% \tzlines: shift, intersection
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines[red,thick,<->,text=blue]
  <0,-1>"AA"(0,1)
  (1,2)(3,1)(4,3){shifted\\line A}[align=left,ar] ;
\tzlines[dashed]
  <1,0>"BB"(0,0)(1,1)(2,1)(3,0){line B}[r] ;
% intersection points
\tzXpoint*{AA}{BB}(X){X1}[-90] % [<angle>] for point
\tzdot(X-2){X2}[0]             % [<angle>] for dot
\end{tikzpicture}
```



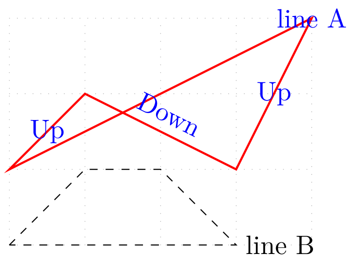
Extending the path You can extend the path of `\tzlines` by writing Ti*k*Z code in the last (even *after the semicolon*) optional argument `<code.append>`.

```
% \tzlines: <code.append>
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines[red,thick,<->,text=blue]
        (0,1){Up}
        (1,2){Down}[a,sloped]
        (3,1){Up}
        (4,3) ;
        <arc(0:90:1cm)>
\tzlines[dashed]
        (0,0)(1,1)(2,1)(3,0) ;
        <to[bend right] ++(2,2) node{Here!}>
\end{tikzpicture}
```



You can close the path with a straight line by `<--cycle>`.

```
% \tzlines: closed path
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines[red,thick,<->,text=blue]
        (0,1){Up}
        (1,2){Down}[a,sloped]
        (3,1){Up}
        (4,3){line A} ; <--cycle>
\tzlines[dashed]
  (0,0)(1,1)(2,1)(3,0){line B}[r] ; <--cycle>
\end{tikzpicture}
```



11.5 \tzlines+: Relative coordinates: Semicolon version

The *plus version* \tzlines+ connects two or more points with connected line segments, but each coordinate (except the first one) is *relative* to the previous coordinate.

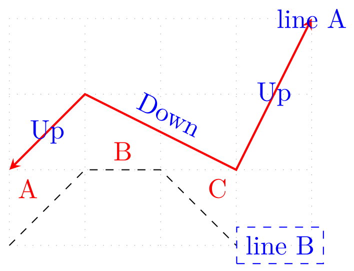
Everything else is the same as in \tzlines. It is also required to enter a *semicolon* to indicate when the coordinate iteration ends.

```
% syntax: full
\tzlines+[<opt>]<shift coor>"<path name>"
         (<coor>){<text>}[<node opt>]
         ..repeated.. (){}[] ; <code.append>
% defaults
  []<>"" (<m>){}[] ..repeated.. (){}[] ; <>
```

```
\tzlines+(0,1)(1,1)(2,-1)(1,2); % works like:
  \draw (0,1) -- ++(1,1) -- ++(2,-1) -- ++(1,2);
```

```
\tzlines+[dashed]"AA"(0,1){A}(1,1){B}(2,-1){C}(1,2){D}[right]; % works like:
  \draw [dashed,name path=AA]
      (0,1) -- node {A} ++(1,1)
            -- node {B} ++(2,-1)
            -- node {C} ++(1,2)  node [right] {D} ;
```

```
% \tzlines+
\begin{tikzpicture}
\tzhelplines(4,3)
\tzlines+[red,thick,<->,text=blue]
         (0,1){Up}
         (1,1){Down}[a,sloped]
         (2,-1){Up}
         (1,2){line A} ;
\tzlines+[dashed,auto,text=red]
         (0,0){A}
         (1,1){B}
         (1,0){C}
         (1,-1){line B}[blue,draw,r] ;
\end{tikzpicture}
```



12 Connecting Points

12.1 \tzto: Two points

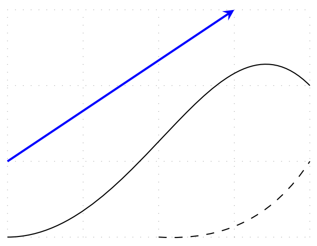
\tzto connects two points with a straight or curved line, using TikZ's to operation. So \tzto is more general than \tzline, which connects points only with a straight line.

```
% syntax: minimum
\tzto(<coor>)(<coor>)
% syntax: medium
\tzto[<opt>](<coor>){<text>}[<node opt>](<coor>){<text>}[<node opt>]
% syntax: full
\tzto[<opt>]<shift coor>"<path name>"
     (<coor>){<text>}[<node opt>]
     (<coor>){<text>}[<node opt>]<code.append>
% defaults
  []<>""(<m>){}[above](<m>){}[]<>
```

```
\tzto[dashed](1,1)(3,2) % works like:
  \draw [dashed] (1,1) to (3,2);
```

Line styles You can control line styles with the first optional argument [<opt>].

```
% \tzto
\begin{tikzpicture}
\tzhelplines(4,3)
\tzto[blue,thick,->](0,1)(3,3)
\tzto[out=0](0,0)(4,2)
\tzto[bend right,dashed](2,0)(4,1)
\end{tikzpicture}
```

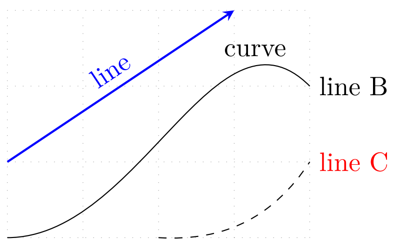


Adding text You can add text *next to the line* ([midway,above], by default) by specifying the options {<text>} and [<node opt>] *in-between* the two coordinates.

```
\tzto[->,bend right](1,1){A}[near start](3,2) % works like:
  \draw [->,bend right] (1,1) to node [near start] {A} (3,2);

\tzto[->,bend right](1,1){A}[near start](3,2){B}[right] % works like:
  \draw [->,bend right] (1,1) to node [near start] {A}
                        (3,2)    node [right]      {B};
```

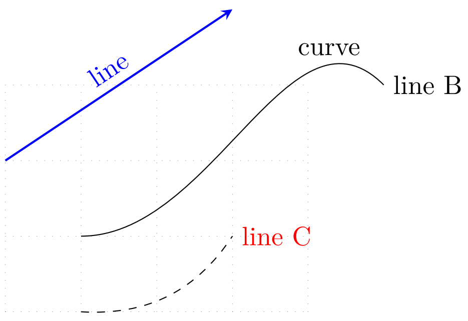
You can also add text *at or around* the last coordinate by the options {<text>} and [<node opt>] *immediately after* the last coordinate.

```
% \tzto: adding text
\begin{tikzpicture}
\tzhelplines(4,3)
\tzto[blue,thick,->](0,1){line}[sloped](3,3)
\tzto[out=0](0,0){curve}[pos=.8](4,2){line B}[r]
\tzto[bend right,dashed](2,0)(4,1){line C}[red,r]
\end{tikzpicture}
```



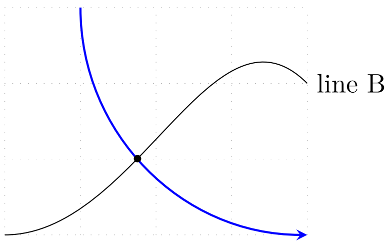
Shift You can move the line by specifying the option `<shift coor>` before the first coordinate or immediately before the option `"<path name>"` if it exists. (The empty shift option `<>` is not allowed.)

```
% \tzto: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzto[blue,thick,->]<0,1>(0,1){line}[sloped](3,3)
\tzto[out=0]<1,1>(0,0){curve}[pos=.8](4,2){line B}[r]
\tzto[bend right,dashed]<-1,0>(2,0)(4,1){line C}[red,r]
\end{tikzpicture}
```



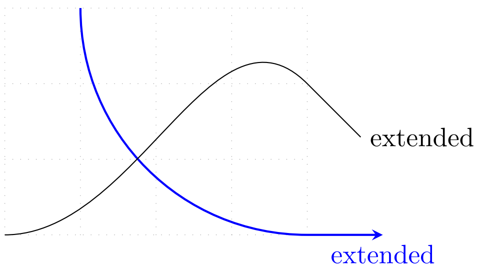
Naming paths: Intersections You can name the path of `\tzto` by specifying the option `"<path name>"` immediately before the first coordinate.

```
% \tzto: name path
\begin{tikzpicture}
\tzhelplines(4,3)
\tzto[bend right=45,blue,thick,->]"curveA"(1,3)(4,0)
\tzto[out=0]"curveB"(0,0)(4,2){line B}[r]
% intersection point
\tzXpoint*{curveA}{curveB}
\end{tikzpicture}
```



Extending the path You can extend the path of `\tzto` by writing Ti*k*Z code in the last optional argument `<code.append>`.

```
% \tzto: extending path
\begin{tikzpicture}
\tzhelplines(4,3)
\tzto[bend right=45,blue,thick,->]
  (1,3)(4,0)<--([turn]0:1cm) node [b] {extended}>
\tzto[out=0]
  (0,0)(4,2)<--([turn]0:1cm) node [r] {extended}>
\end{tikzpicture}
```



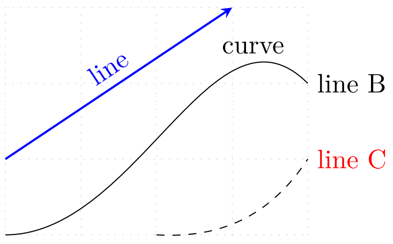
12.2 \tzto+: Relative coordinates

The *plus version* `\tzto+` uses the second coordinate relative to the first coordinate.

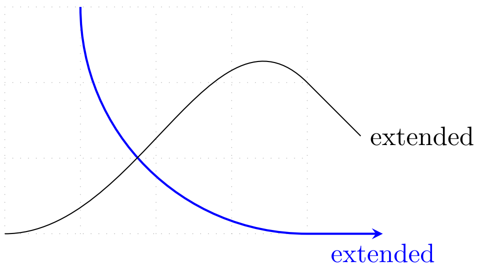
Everything else is the same as in `\tzto`.

```
\tzto+(1,1)(3,2) % works like:
  \draw (1,1) to ++(3,2);
```

```
% \tzto+
\begin{tikzpicture}
\tzhelplines(4,3)
\tzto+[blue,thick,->](0,1){line}[sloped](3,2)
\tzto+[out=0](0,0){curve}[pos=.8](4,2){line B}[r]
\tzto+[bend right,dashed](2,0)(2,1){line C}[red,r]
\end{tikzpicture}
```



```
% \tzto+
\begin{tikzpicture}
\tzhelplines(4,3)
\tzto+[bend right=45,blue,thick,->]
  (1,3)(3,-3)<--([turn]0:1cm) node [b] {extended}>
\tzto+[out=0]
  (0,0)(4,2)<--([turn]0:1cm) node [r] {extended}>
\end{tikzpicture}
```



12.3 \tztos: Multiple points: Semicolon version

\tztos takes an arbitrary number of coordinates as arguments to connect them by TikZ's to operation. Since this is a *semicolon version*, you need to enter a *semicolon* to indicate when the coordinate iteration ends.

```
% syntax
\tztos[<opt>]<shift coor>"<path name>"
      (<coor1>)[<to opt>]{<text>}[<node opt>]..repeated..()[]{}[] ; <code.append>
% defaults
  []<>""(<m>)[]{}[]..repeated..()[]{}[]<>
```

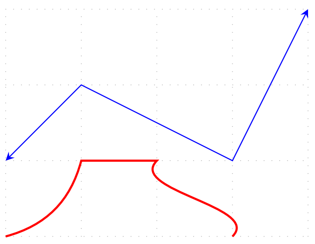
The quadruple (<coor>)[<to opt>]{<text>}[<node opt>] is the whole repeating pattern. Here, [<to opt>] is for the options of TikZ's to operation such as [bend right], [bend left], [bend left=<angle>], [out=<angle>,in=<angle>] and so on.

```
\tztos(0,0)(1,2)(3,1); % works like:
  \draw (0,0) to (1,2) to (3,1);
```

```
\tztos[blue]  (0,0)    [bend right] (1,2)    [out=-60,in=45] (3,1); % works like:
  \draw[blue] (0,0) to [bend right] (1,2) to [out=-60,in=45] (3,1);
```

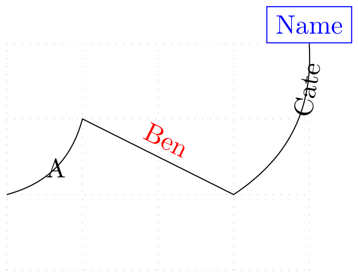
How to connect coordinates You can use the options of TikZ's to operation to connect the coordinates with different types of curves.

```
% \tztos
\begin{tikzpicture}
\tzhelplines(4,3)
\tztos[blue,<->](0,1)(1,2)(3,1)(4,3);
\tztos[red,thick]
      (0,0)[bend right]
      (1,1)
      (2,1)[out=-135,in=45]
      (3,0);
\end{tikzpicture}
```



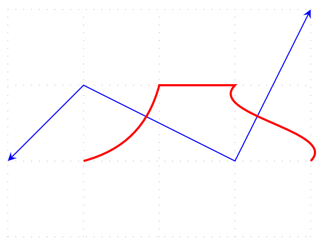
Adding text You can add text next to lines or curves by specifying the options `{<text>}` and `[<node opt>]` *in-between* coordinates or *after* the option `[<to opt>]`, if it exists. You can also add text at or around the last coordinate by the last options `{<text>}` and `[<node opt>]`.

```
% \tztos: adding text
\begin{tikzpicture}
\tzhelplines(4,3)
\tztos(0,1)[bend right]{A}
      (1,2)            {Ben}[red,sloped,a]
      (3,1)[bend right]{Cate}[sloped,near end]
      (4,3)            {Name}[draw,blue,a] ;
\end{tikzpicture}
```



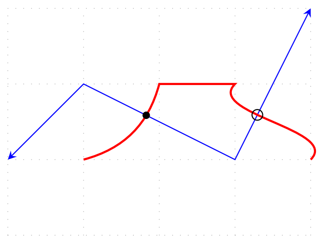
Shift You can move the line or curve of `\tztos` using the option `<shift coor>` before the first coordinate or immediately before the option `"<path name">`, if any. (The empty shift option `<>` is not allowed.)

```
% \tztos: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tztos[blue,<->](0,1)(1,2)(3,1)(4,3);
\tztos[red,thick]
  <1,1>(0,0)[bend right]
       (1,1)
       (2,1)[out=-135,in=45]
       (3,0);
\end{tikzpicture}
```



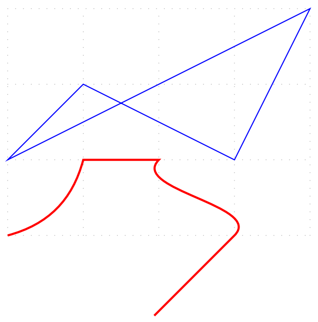
`name path` for intersections You can name the path of `\tztos` by specifying `"<path name>"` immediately before the first mandatory coordinate.

```
% \tztos: name path, intersection points
\begin{tikzpicture}
\tzhelplines(4,3)
\tztos[blue,<->]"AA"(0,1)(1,2)(3,1)(4,3);
\tztos[red,thick]
  <1,1>"BB"(0,0)[bend right]
       (1,1)
       (2,1)[out=-135,in=45]
       (3,0);
% intersection points
\tzXpoint*{AA}{BB}(X)
\tzdot(X-2)(4pt)
\end{tikzpicture}
```



Extending paths You can extend the path of `\tztos` from the last coordinate, by writing Ti*k*Z code in the last (after the semicolon) optional argument `<code.append>`.

```
% \tztos
\begin{tikzpicture}
\tzhelplines(4,3)
\tztos[blue,<->](0,1)(1,2)(3,1)(4,3);<--cycle>
\tztos[red,thick]
      (0,0)[bend right]
      (1,1)
      (2,1)[out=-135,in=45]
      (3,0); < to ([turn]0:1.5cm) >
\end{tikzpicture}
```

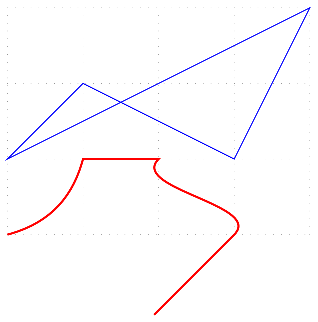


12.4 `\tztos+`: Relative coordinates: Semicolon version

The *plus version* `\tztos+` takes each coordinate (except the first coordinate) relative (with `++`) to the previous coordinate.

Everything else is the same as in `\tztos`.

```
% \tztos+
\begin{tikzpicture}
\tzhelplines(4,3)
\tztos+[blue,<->](0,1)(1,1)(2,-1)(1,2);<--cycle>
\tztos+[red,thick]
      (0,0)[bend right]
      (1,1)
      (1,0)[out=-135,in=45]
      (1,-1); < to ([turn]0:1.5cm) >
\end{tikzpicture}
```



13 Filling Area

13.1 `\tzpath`: Semicolon version

`\tzpath` creates a path connecting an arbitrary number of coordinates, but it does not stroke the path. You can visualize the path with `\tzpath[draw]`. Since `\tzpath` is a *semicolon version* macro, you need to enter a *semicolon* ‘;’ to indicate where the coordinate iteration ends.

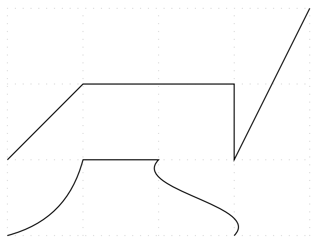
```
% syntax: minimal
\tzpath (<coor>)(<coor>) ..repeated.. (<coor>) ;
% syntax: full
\tzpath{<path style>}[<opt>]<shift coor>"<path name>"
       (<coor>)[<path style>]{<text>}[<node opt>]
       ..repeated.. ()[]{}[] ; {<fill opacity>} <code.append>
% defaults: \tzpath
  {to}[]<>"" (<m>)[]{}[] ..repeated.. ()[]{}[] ; {}<>
```

The main purpose of `\tzpath` is to fill an enclosed area with color or pattern. You can use `\tzpath[fill]` or `\tzpath[pattern=<...>]` to do it.

Path construction operation (If you are not an experienced user of Ti*k*Z just skip this part.)

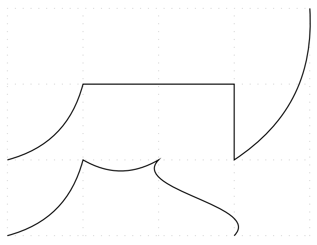
`\tzpath` is similar to `\tztos`, but it is more flexible in constructing paths. `\tzpath` allows you to choose how to construct a path using the `[<path style>]` option *in-between* coordinates. *Path extension operation* can be selected from '`--`', '`to`', '`|-`', '`-|`', etc. You can use *the first brace option* `{<path style>}` to change all the `<path style>` in-between coordinates.

```
% \tzpath: path construction
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpath[draw](0,1)(1,2)[-|](3,1)(4,3);
\tzpath[draw]
      (0,0)[to[bend right]]
      (1,1)
      (2,1)[to[out=-135,in=45]]
      (3,0);
\end{tikzpicture}
```



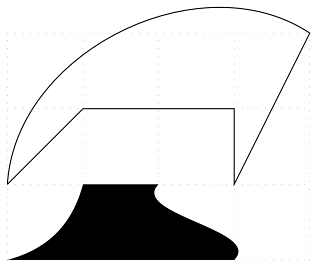
The default path style is '`to`' and can also be changed by `\settzpathstyle`, like, for example, `\settzpathstyle{--}`. The effect remains valid until the end of `tikzpicture` environment unless changed again.

```
% \tzpath: \settzpathstyle
\begin{tikzpicture}
\tzhelplines(4,3)
\settzpathstyle{to[bend right]}
\tzpath[draw](0,1)(1,2)[-|](3,1)(4,3);
\tzpath[draw]
      (0,0)[to[bend right]]
      (1,1)
      (2,1)[to[out=-135,in=45]]
      (3,0);
\end{tikzpicture}
```



You can extend the path of `\tzpath` by writing Ti*k*Z code in the last (after the semicolon) optional argument `<code.append>`. So `<--cycle>` closes the path with a straight line.

```
% \tzpath: <code.append>
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpath[draw](0,1)(1,2)[-|](3,1)(4,3);
            < to [bend right=60] (0,1) >
\tzpath[fill]
      (0,0)[to[bend right]]
      (1,1)
      (2,1)[to[out=-135,in=45]]
      (3,0); <--cycle>
\end{tikzpicture}
```



13.2 \tzpath*: Semicolon version

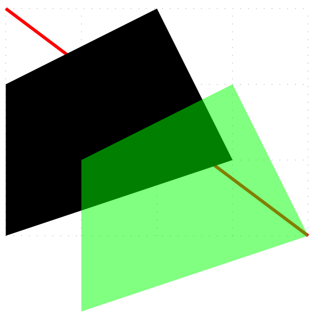
The starred version `\tzpath*` additionally fills the interior of `\tzpath` with `black!50` with `fill opacity=.3` and `text opacity=1`, by default.

`\tzpath*` works like `\tzpath[fill=black!50,fill opacity=.3,text opacity=1]`. You can change the defaults by `\settzfillcolor` and `\settzfillopacity`.

```
% syntax: minimal
\tzpath*(<coor>)(<coor>) ..repeated.. (<coor>) ; {<fill opacity>}
% syntax: full
\tzpath*{<path style>}[<opt>]<shift coor>"<path name>"
        (<coor>)[<path style>]{<text>}[<node opt>]
        ..repeated.. ()[]{}[] ; {<fill opacity>} <code.append>
% defaults: \tzpath*
 *{to}[fill=black!50,fill opacity=.3,text opacity=1]<>""
  (<m>)[]{}[] ..repeated.. ()[]{}[] ; {.3}<>
```

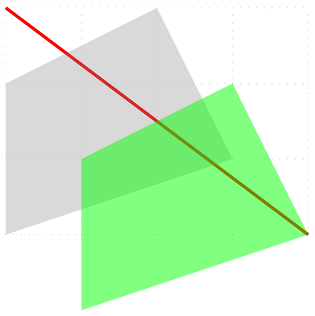
Filling the interior You can optionally change the opacity of fill color using the Ti*k*Z option `fill opacity`.

```
% \tzpath: fill, fill opacity
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[red,very thick](0,3)(4,0)
\tzpath[fill](0,0)(3,1)(2,3)(0,2); % default opacity=1
\tzpath[fill,green,fill opacity=.5]
  <1,-1>(0,0)(3,1)(2,3)(0,2);
\end{tikzpicture}
```



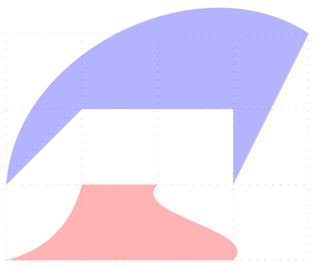
You can also change the fill opacity by specifying the *last* curly brace optional argument `{<fill opacity>}`, *after the semicolon.*

```
% \tzpath*
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[red,very thick](0,3)(4,0)
\tzpath*(0,0)(3,1)(2,3)(0,2);
\tzpath*[fill=green]
  <1,-1>(0,0)(3,1)(2,3)(0,2); {.5} %%
\end{tikzpicture}
```



You can also use the macros \settzfillcolor and \settzfillopacity to change the defaults. The effect remains valid until the end of the tikzpicture environment, unless changed again.

```
% \tzpath*: \settzfillcolor
\begin{tikzpicture}
\tzhelplines(4,3)
\settzfillcolor{red}
\tzpath*[blue](0,1)(1,2)[-|](3,1)(4,3);
              < to [bend right=60] (0,1) >
\tzpath*
      (0,0)[to[bend right]]
      (1,1)
      (2,1)[to[out=-135,in=45]]
      (3,0); <--cycle>
\end{tikzpicture}
```



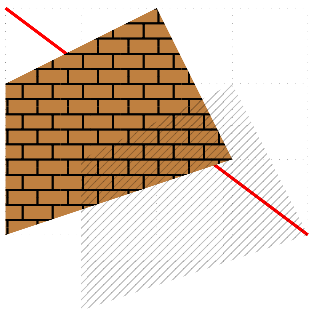
13.3 \tzpath+ and \tzpath*+: Relative coordinates: Semicolon versions

The *plus version* \tzpath+ uses each coordinate (except for the first coordinate) relative (with ++) to the previous coordinate.

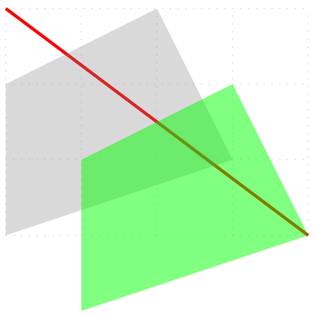
Everything else is the same as in \tzpath.

\tzpath*+ is simply the plus version of \tzpath*.

```
% \tzpath+
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[red,very thick](0,3)(4,0)
\tzpath+[pattern=bricks,preaction={fill=brown}]
        (0,0)(3,1)(-1,2)(-2,-1);
\tzpath+[pattern=north east lines,opacity=.5]
  <1,-1>(0,0)(3,1)(-1,2)(-2,-1);
\end{tikzpicture}
```



```
% \tzpath*+
\begin{tikzpicture}
\tzhelplines(4,3)
\tzline[red,very thick](0,3)(4,0)
\tzpath*+(0,0)(3,1)(-1,2)(-2,-1);
\tzpath*+[fill=green]
   <1,-1>(0,0)(3,1)(-1,2)(-2,-1); {.5} %%
\end{tikzpicture}
```



14 Curves

There are many ways to draw curves.

14.1 Bézier curves

\tzbezier draws a Bézier curve with one or two control points.

14.1.1 \tzbezier

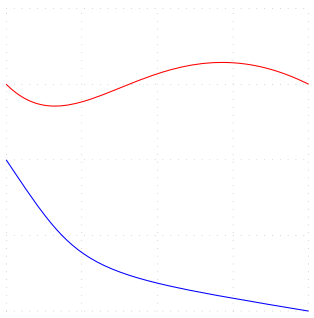
```
% syntax: minimal
\tzbezier(<start-coor>)(cntl-coor>)(<last-coor>)
\tzbezier(<start-coor>)(cntl-coor>)(<cntl-coor>)(<last-coor>)
% syntax: full
\tzbezier[<draw opt>]<shift coor>"<name path>"
         (<start-coor>)(cntl-coor>)(<cntl-coor>)(<last-coor>)
         {<text>}[<node opt>]<code.append>
% defaults
  []<>""(<m>)(<m>)()(<m>){}[]<>
```

Control points You can specify one or two control points, (<cntl-coor>). So \tzbezier accepts three or four coordinates as arguments.

```
% three coordinates: one control point
\tzbezier(0,1)(1,0)(4,3) % works like:
  \draw (0,0) ..controls (1,0).. (4,3);
```

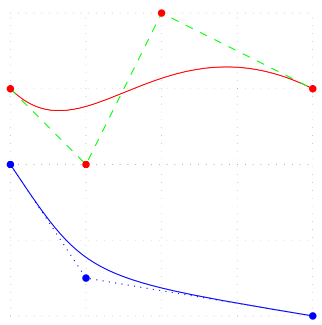
```
% four coordinates: two control points
\tzbezier(0,1)(1,0)(2,4)(4,3) % works like:
  \draw (0,0) ..controls (1,0) and (2,4).. (4,3);
```

```
% \tzbezier
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier[red](0,3)(1,2)(2,4)(4,3)
\tzbezier[blue](0,2)(1,.5)(4,0)
\end{tikzpicture}
```



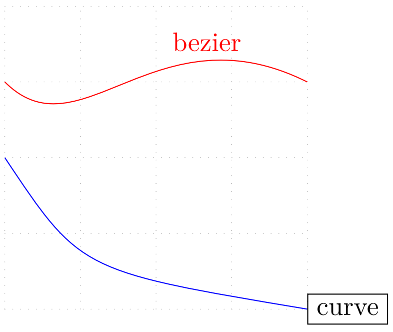
The style tzshowcontrols displays the control points by drawing dotted lines, by default. You can also change the dotted line style, like tzshowcontrols={dashed,green}.

```
% \tzbezier: tzshowcontrols
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier[blue,tzshowcontrols](0,2)(1,.5)(4,0)
\tzdots*[blue](0,2)(1,.5)(4,0);
\tzbezier[red,tzshowcontrols={green,dashed}]
         (0,3)(1,2)(2,4)(4,3)
\tzdots*[red](0,3)(1,2)(2,4)(4,3);
\end{tikzpicture}
```



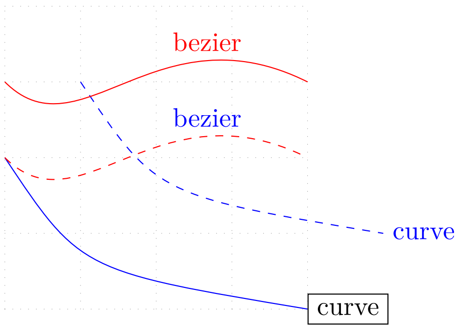
Adding text You can add text next to the curve or at the last coordinate by specifying the options {<text>} and [<node opt>] immediately after the last coordinate.

```
% \tzbezier: adding text
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier[blue](0,2)(1,.5)(4,0){curve}[draw,black,r]
\tzbezier[red](0,3)(1,2)(2,4)(4,3){bezier}[near end,a]
\end{tikzpicture}
```



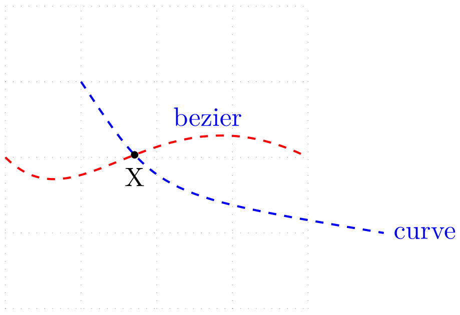
Shift You can move the curve of `\tzbezier` by specifying the option `<shift coor>` before the first coordinate or immediately before the option `"<path name>"`, if any. The empty shift option `<>` is not allowed.

```
% \tzbezier: shift
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier[blue]
         (0,2)(1,.5)(4,0){curve}[draw,black,r]
\tzbezier[blue,dashed]
  <1,1>(0,2)(1,.5)(4,0){curve}[r]
\tzbezier[red]
         (0,3)(1,2)(2,4)(4,3){bezier}[near end,a]
\tzbezier[red,dashed]
  <0,-1>(0,3)(1,2)(2,4)(4,3){bezier}[blue,near end,a]
\end{tikzpicture}
```



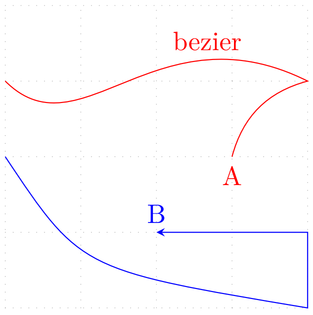
`name path` You can name the path of `\tzbezier` by specifying the option `"<path name>"` immediately before the first coordinate.

```
% \tzbezier: name path, intersection
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier[blue,dashed,thick]<1,1>"curve"
  (0,2)(1,.5)(4,0){curve}[r]
\tzbezier[red,dashed,thick]<0,-1>"bezier"
  (0,3)(1,2)(2,4)(4,3){bezier}[blue,near end,a]
% intersection point
\tzXpoint*{curve}{bezier}(X){X}[-90] % <angle>
\end{tikzpicture}
```



Extending paths You can extend the path of `\tzbezier` from the last coordinate, by writing Ti*k*Z code in the last optional argument `<code.append>`.

```
% \tzbezier: extending path
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier[red](0,3)(1,2)(2,4)(4,3){bezier}[near end,a]
  < to [bend right] ++(-1,-1) node [below] {A} >
\tzbezier[blue,->](0,2)(1,.5)(4,0)
  < |- ++(-2,1) node [above] {B} >
\end{tikzpicture}
```



14.1.2 \tzbezier+: Relative coordinates

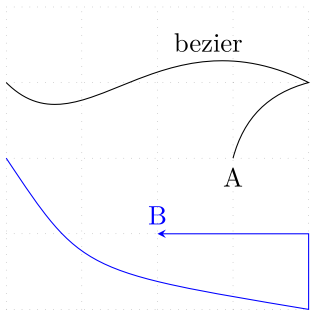
For the *plus version* `\tzbezier+(A)(B)(C)`, `(B)` and `(C)` are relative to `(A)`, as in Ti*k*Z.

For `\tzbezier+(A)(B)(C)(D)`, `(B)` and `(D)` are relative to `(A)`, and `(C)` is relative to `(D)`, as in Ti*k*Z.

```
% three coordinates
\tzbezier+(0,1)(1,-1)(4,3) % works like:
  \draw (0,1) ..controls +(1,-1).. ($(0,1)+(4,3)$);
```

```
% four coordinates
\tzbezier+(0,1)(1,-1)(-2,1)(4,3) % works like:
  \draw (0,1) ..controls +(1,-1) and +(-2,1).. ($(0,1)+(4,3)$);
```

```
% \tzbezier+
\begin{tikzpicture}
\tzhelplines(4,4)
\tzbezier+(0,3)(1,-1)(-2,1)(4,0){bezier}[near end,a]
  < to [bend right] ++(-1,-1) node [below] {A} >
\tzbezier+[blue,->](0,2)(1,-1.5)(4,-2)
  < |- ++(-2,1) node [above] {B} >
\end{tikzpicture}
```



14.2 Parabolas

14.2.1 \tzparabola

`\tzparabola` draws a parabola going through specified coordinates. It accepts two or three coordinates. In the case of three coordinates, the parabola bends at the second coordinate.

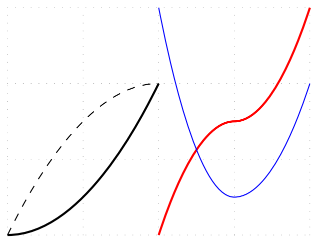
```
% two coordinates
\tzparabola(0,0)(3,2) % works like:
  \draw (0,0) parabola (3,2);
```

```
% three coordinates
\tzparabola(0,0)(1,1)(3,2) % works like:
  \draw (0,0) parabola bend (1,1) (3,2);
```

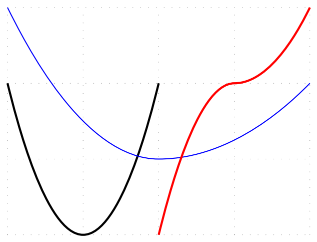
```
% syntax: minimal
\tzparabola(<coor>)(<coor>)
\tzparabola(<coor>)(<coor>)(<coor>)
% syntax: full
\tzparabola[<opt1>]<shift coor>"<path name>"
           (<coor>)(<coor>)(<coor>){<text>}[<node opt>]<code.append>
% default
  []<>""(<m>)()(<m>){}[]<>
```

Parabolas `\tzplot` draws the graph of a quadratic function $f(x) = ax^2 + bx + c$ for appropriate values of a, b, and c.

```
% \tzparabola: two coordinates
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola[thick](0,0)(2,2)
\tzparabola[bend at end,dashed](0,0)(2,2)
\tzparabola[red,thick,bend pos=.5](2,0)(4,3)
\tzparabola[blue,parabola height=-2cm](2,3)(4,2)
\end{tikzpicture}
```

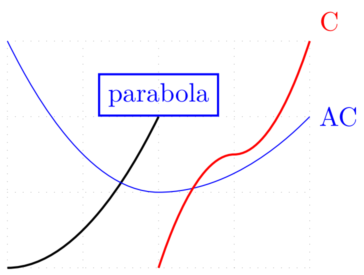


```
% \tzparabola: three coordinates
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola[thick](0,2)(1,0)(2,2)
\tzparabola[blue](0,3)(2,1)(4,2)
\tzparabola[red,thick](2,0)(3,2)(4,3)
\end{tikzpicture}
```



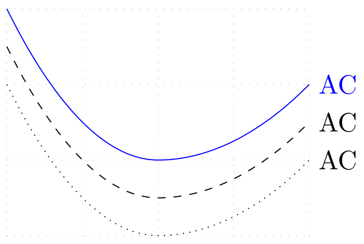
Adding text You can add text at or around the last coordinate by specifying the options `{<text>}` and `[<node opt>]` immediately after the last coordinate.

```
% \tzparabola: adding text
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola[thick](0,0)(2,2){parabola}[blue,draw,a]
\tzparabola[blue](0,3)(2,1)(4,2){AC}[r]
\tzparabola[red,thick,bend pos=.5](2,0)(4,3){C}[ar]
\end{tikzpicture}
```



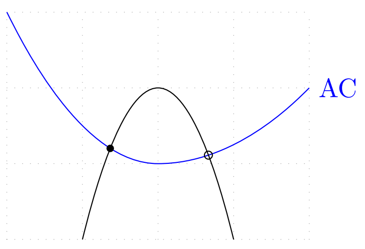
Shift You can move the parabola by specifying the option `<shift coor>` before the first coordinate or immediately before the option `"<path name>"`, if any. The empty shift option `<>` is not allowed.

```
% \tzparabola: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola[blue]         (0,3)(2,1)(4,2){AC}[r]
\tzparabola[dashed]<0,-.5>(0,3)(2,1)(4,2){AC}[r]
\tzparabola[dotted]<0,-1> (0,3)(2,1)(4,2){AC}[r]
\end{tikzpicture}
```



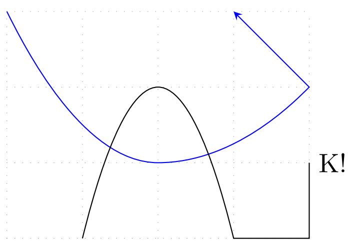
`name path` You can name the path of `\tzparabola` by specifying the option `"<path name>"` immediately before the first coordinate.

```
% \tzparabola: name path, intersection
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola[blue]"AC"(0,3)(2,1)(4,2){AC}[r]
\tzparabola"BB"(1,0)(2,2)(3,0)
% intersection points
\tzXpoint*{AC}{BB}(X)
\tzdot(X-2)(3pt)
\end{tikzpicture}
```



Extending paths You can extend the path of `\tzparabola` from the last coordinate by writing Ti*k*Z code in the last optional argument `<code.append>`.

```
% \tzparabola: extending path
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola[blue,->]"AC"(0,3)(2,1)(4,2)
  < -- ++ (-1,1) >
\tzparabola"BB"(1,0)(2,2)(3,0)
  < -| ++ (1,1) node [right] {K!} >
\end{tikzpicture}
```



14.2.2 `\tzparabola+`: Relative coordinates

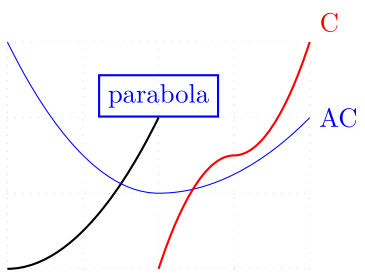
For the *plus version* `\tzparabola+`, the second and the third coordinates are relative to the first coordinate.

Everything else is the same as in `\tzparabola`.

```
% two coordinates
\tzparabola+(0,1)(3,2) % works like:
  \draw (0,1) parabola ($(0,1)+(1,2)$);
```

```
% three coordinates
\tzparabola+(0,1)(1,1)(3,-1) % works like:
  \draw (0,1) parabola bend +(1,1) ($(0,1)+(3,-1)$);
```

```
% \tzparabola+
\begin{tikzpicture}
\tzhelplines(4,3)
\tzparabola+[thick](0,0)(2,2){parabola}[blue,draw,a]
\tzparabola+[blue](0,3)(2,-2)(4,-1){AC}[r]
\tzparabola+[red,thick,bend pos=.5](2,0)(2,3){C}[ar]
\end{tikzpicture}
```



14.3 `\tzplotcurve`

See Section 9.6 on page 58, for more details on `\tzplotcurve`.

14.4 \tzto, \tztos

See Section 12.1 on page 74, for more details on `\tzto`.

See Section 12.3 on page 76, for more details on `\tztos`.

14.5 \tzfn

With `\tzfn`, you can plot functions such as $f(x) = \frac{1}{3}(x-1)^3 + 1$, $g(x) = \sin x$, $h(x) = \sqrt{x-1}$, and so on. See Section 20.1 on page 110, for more details on `\tzfn`.

15 Polygons and Circles

15.1 \tzpolygon: Semicolon verion

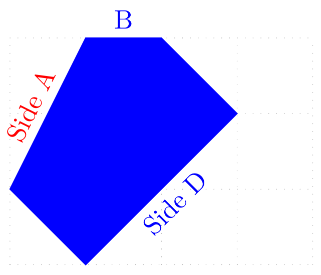
`\tzpolygon` connects an arbitrary number of coordinates to draw a polygon, a closed figure. `\tzpolygon` is equivalent to a closed `\tzlines`. Since `\tzpolygon` is a *semicolon version*, you need to enter a *semicolon* to indicate when the coordinate repetition ends.

```
% syntax: minimum
\tzpolygon(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: medium
\tzpolygon (<coor>){<text>}[<node opt>]..repeated..(<coor>){<text>}[<node opt>] ;
% syntax: full
\tzpolygon[<opt>]<shift coor>"<path name>"
          (<coor>){<text>}[<node opt>]
          ..repeated.. (){}[] ; <code.append>
% defaults
  []<>""(<m>){}[] ..repeated.. (){}[] ; <>
```

```
\tzpolygon(1,1)(2,2)(3,1)(4,3); % works like:
  \draw (1,1) -- (2,2) -- (3,1) -- (4,3) -- cycle;
```

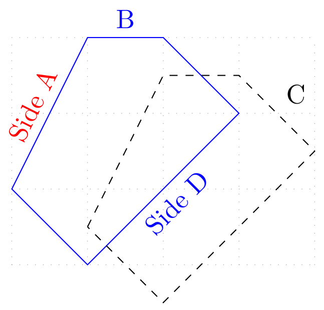
You can add text next to lines by specifying the options `{<text>}` and `[<node opt>]` *in-between* coordinates.

```
% \tzpolygon
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpolygon[fill,blue,auto]
          (0,1){Side A}[sloped,red]
          (1,3){B}
          (2,3)
          (3,2){Side D}[swap,sloped]
          (1,0);
\end{tikzpicture}
```



You can also move the polygon by specifying the option `<shift coor>` before the first coordinate. The empty shift option `<>` is not allowed.

```
% \tzpolygon: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpolygon[blue,auto]
          (0,1){Side A}[sloped,red]
          (1,3){B}
          (2,3)
          (3,2){Side D}[swap,sloped]
          (1,0);
\tzpolygon[auto,dashed]<1,-.5>
          (0,1)(1,3)(2,3){C}(3,2)(1,0);
\end{tikzpicture}
```



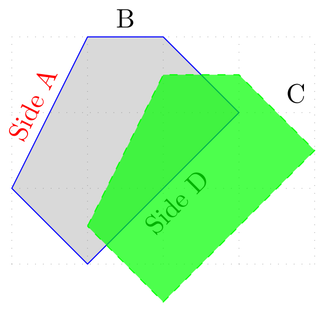
15.2 \tzpolygon*: Semicolon version

\tzpolygon* paints the interior of the polygon with black!50 with fill opacity=.3 but with text opacity=1, by default.

```
% syntax: minimum
\tzpolygon*(<coor>)(<coor>)..repeated..(<coor>) ;
% syntax: medium
\tzpolygon*(<coor>){<text>}[<node opt>]..repeated..(<coor>){<text>}[<node opt] ;
% syntax: full
\tzpolygon*[<opt>]<shift coor>"<path name>"
           (<coor>){<text>}[<node opt>]
           ..repeated.. (){}[] ; {<fill opacity>} <code.append>
% defaults
 *[fill=black!50,fill opacity=.3,text opacity=1]<>""
  (<m>){}[] .. repeated.. (){}[] ; {} <>
```

You can change the fill opacity by specifying the the last curly brace option {<fill opacity>} immediately *after the semicolon.*

```
% \tzpolygon: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpolygon*[draw=blue,auto]
          (0,1){Side A}[sloped,red]
          (1,3){B}
          (2,3)
          (3,2){Side D}[swap,sloped]
          (1,0);
\tzpolygon*[green,auto,dashed,text=black]<1,-.5>
          (0,1)(1,3)(2,3){C}(3,2)(1,0); {.7}
\end{tikzpicture}
```



You can also change the defaults using \settzfillcolor and \sttzfillopacity.

15.3 \tzpolygon+ and \tzpolygon*+: Relative coordinates: Semicolon verion

The plus version \tzpolygon+ uses each coordinate (except the first one) relative (with ++) to the previous coordinate.

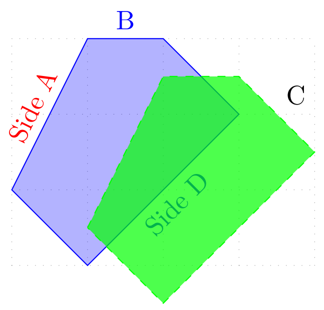
Everything else is the same as in \tzpolygon.

\tzpolygon*+ is just a plus version of \tzpolygon*.

```
\tzpolygon+(1,1)(2,2)(3,1)(4,3); % works like:
  \draw (1,1) -- ++(2,2) -- ++(3,1) -- ++(4,3) -- cycle;
```

```
\tzpolygon+[dashed]"AA"(1,1)(2,2){A}(3,1){B}[below]; % works like:
  \draw [dashed,name path=AA] (1,1)
            -- ++(2,2)
            -- ++(3,1) node         {A}
            -- ++(4,3) node [below] {B}
            -- cycle;
```

```
% \tzpolygon: shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzpolygon*+[blue,auto]
          (0,1){Side A}[sloped,red]
          (1,2){B}
          (1,0)
          (1,-1){Side D}[swap,sloped]
          (-2,-2);
\tzpolygon*+[green,auto,dashed,text=black]<1,-.5>
          (0,1)(1,2)(1,0){C}(1,-1)(-2,-2); {.7}
\end{tikzpicture}
```



15.4 \tzframe and \tzframe*

`\tzframe` accepts two coordinates draws a rectangle.

```
% syntax: minimum
\tzframe(<coor>)(<coor>)
% syntax: full
\tzframe[<opt>]<shift coor>"<path name>"(<coor1>)(<coor2>)<code.append>
% defaults
  []<>""(<m>)(<m>)<>
```

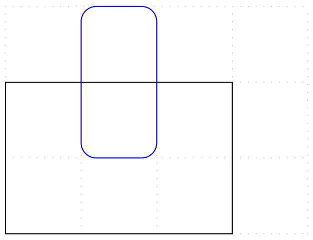
`\tzrectangle` is an alias of `\tzframe`.

```
\tzframe(0,1)(3,2) % works like:
  \draw (0,1) rectangle (3,2);
```

The plus version `\tzframe+` uses the second coordinate as the coordinate relative (with `++`) to the first. `\tzrectangle+` is an alias of `\tzframe+`.

```
\tzframe+(0,1)(3,2) % works like:
  \draw (0,1) rectangle ++(3,2);
```

```
% \tzframe, \tzframe+
\begin{tikzpicture}
\tzhelplines(4,3)
\tzframe(0,0)(3,2)
\tzframe+[blue,rounded corners=2mm](1,3)(1,-2)
\end{tikzpicture}
```



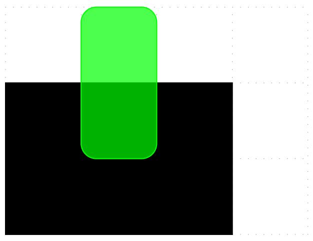
The starred version `\tzframe*` fills the interior with `black!50` with `fill opacity=.3`, by default. (`\tzrectangle*` is an alias of `\tzframe*`.)

`\tzframe+` has also its starred version `\tzframe*+`. (`\tzrectangle*+` is an alias of `\tzframe*+`.)

```
% syntax
\tzframe*[<opt>]<shift coor>"<path name>"
         (<coor1>)(<coor2>){<fill opacity>}<code.append>
% defaults
 *[fill=black!50,fill opacity=.3]<>""(<m>)(<m>){.3}<>
```

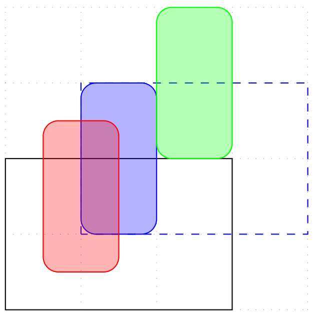
With the starred versions, you can change the fill opacity using the last option `{<fill opacity>}`.

```
% \tzframe*(+)
\begin{tikzpicture}
\tzhelplines(4,3)
\tzframe[fill](0,0)(3,2)
\tzframe*+[green,rounded corners=2mm](1,3)(1,-2){.7} %%
\end{tikzpicture}
```



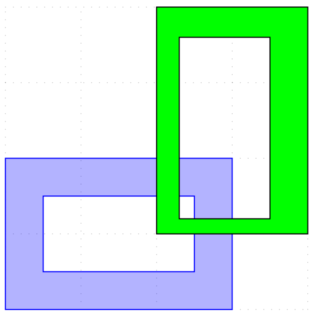
You can move `\tzframe` and its variants by specifying the option `<shift coor>` immediately before the first mandatory coordinate. The empty shift option `<>` is not allowed.

```
% \tzframe(*)(+): shift
\begin{tikzpicture}
\tzhelplines*(4,4)
\tzframe(0,0)(3,2)
\tzframe[blue,dashed]<1,1>(0,0)(3,2)
\tzcoors(1,3)(A)(1,-2)(B);
\tzframe*+[blue,rounded corners=2mm](A)(B)
\tzframe*+[red,rounded corners=2mm]<-.5,-.5>(A)(B)
\tzframe*+[green,rounded corners=2mm]<1,1>(A)(B)
\end{tikzpicture}
```



You can extend the path by the option `<code.append>`.

```
% \tzframe(*), \tzrectangle(*): <code.append>
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzrectangle*[blue,even odd rule](0,0)(3,2)
         <(.5,.5) rectangle (2.5,1.5)>
\tzframe[fill=green,even odd rule](2,1)(4,4)
         <(2.3,1.2) rectangle (3.5,3.6)>
\end{tikzpicture}
```



15.5 \tzcircle and \tzcircle*

`\tzcircle` draws a circle around a specified coordinate with a specified radius. The coordinate and the radius are mandatory.

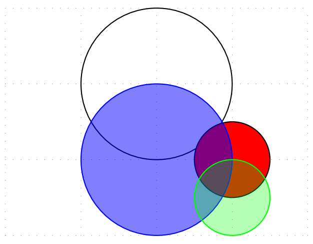
```
% syntax
\tzcircle[<opt>]<shift coor>"<path name>"(<coor>)(<radius>)<code.append>
% defaults
  []<>""(<m>)(<m>)<>
```

```
\tzcircle(0,0)(1cm) % works like:
  \draw (0,0) circle (1cm);
```

The starred version `\tzcircle*` fills the interior with `black!50` with `fill opacity=.3`, by default. You can change the fill opacity using the last option `{<fill opacity>}`.

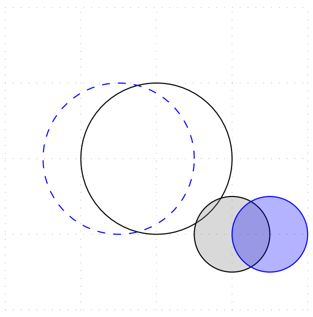
```
% syntax
\tzcircle*[<opt>]<shift coor>"<path name>"
          (<coor>)(<radius>){<fill opacity>}<code.append>
% defaults
 *[fill=black!50,fill opacity=.3]<>""(<m>)(<m>){.3}<>
```

```
% \tzcircle(*)
\begin{tikzpicture}
\tzhelplines(4,3)
\tzcircle(2,2)(1)
\tzcircle[fill=red](3,1)(.5cm)
\tzcircle*[blue](2,1)(1){.5}
\tzcircle*[green](3,.5)(.5cm)
\end{tikzpicture}
```



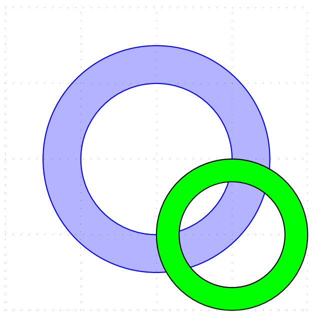
You can move the circle by specifying the option `<shift coor>` immediately before the center coordinate. The empty shift option `<>` is not allowed.

```
% \tzcircle(*): shift
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzcircle(2,2)(1)
\tzcircle[blue,dashed]<-.5,0>(A)(1)
\tzcircle*(3,1)(.5cm)
\tzcircle*[blue]<.5,0>(B)(.5cm)
\end{tikzpicture}
```



You can extend the path by the option `<code.append>`.

```
% \tzcircle(*): <code.append>
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzcircle*[blue,even odd rule](2,2)(1)
        <(2,2) circle (1.5)>
\tzcircle[fill=green,even odd rule](3,1)(1)
        <(3,1) circle (.7)>
\end{tikzpicture}
```



15.6 \tzellipse and \tzellipse*

`\tzellipse` draws an ellipse around a specified coordinate with the specified x-radius and y-radius.

```
% syntax
\tzellipse[<opt>]<shift coor>"<path name>"(<coor>)(<x and y radius>)<code.append>
% defaults
  []<>""(<m>)(<m>)<>
```

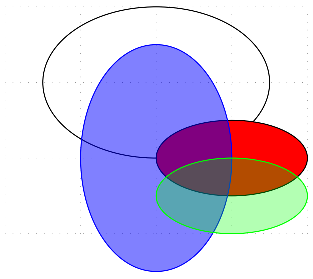
`\tzoval` is an alias of `\tzellipse`.

```
\tzellipse(0,0)(1 and .5) % works like:
  \draw (0,0) ellipse (1 and .5);
```

The starred version `\tzellipse*` fills the interior with `black!50` with `fill opacity=.3`, by default. You can change the fill opacity using the last option `{<fill opacity>}`.

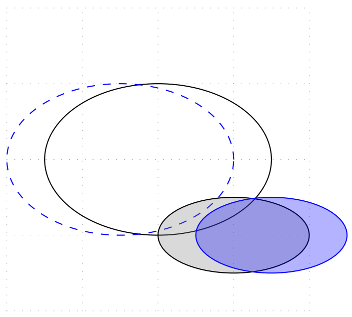
```
% syntax
\tzellipse*[<opt>]<shift coor>"<path name>"
          (<coor>)(<x and y radius>){<fill opacity>}<code.append>
% defaults
 *[fill=black!50,fill opacity=.3]<>""(<m>)(<m>){.3}<>
```

```
% \tzellipse(*)
\begin{tikzpicture}
\tzhelplines(4,3)
\tzellipse(2,2)(1.5 and 1)
\tzellipse[fill=red](3,1)(1cm and .5cm)
\tzellipse*[blue](2,1)(1 and 1.5){.5}
\tzellipse*[green](3,.5)(1cm and .5cm)
\end{tikzpicture}
```



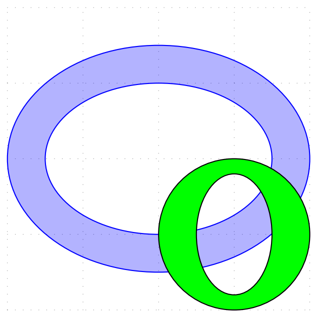
You can move the ellipse by specifying the option `<shift coor>` immediately before the mandatory coordinate. The empty shift option `<>` is not allowed.

```
% \tzellipse(*): shift
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzellipse(2,2)(1.5 and 1)
\tzellipse[blue,dashed]<-.5,0>(A)(1.5 and 1)
\tzellipse*(3,1)(1cm and .5cm)
\tzellipse*[blue]<.5,0>(B)(1cm and .5cm)
\end{tikzpicture}
```



You can extend the path by the option `<code.append>`. Note also that `\tzoval*` is an alias of `\tzellipse*`.

```
% \tzellipse(*), \tzoval(*): <code.append>
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors(2,2)(A)(3,1)(B);
\tzoval*[blue,even odd rule](2,2)(1.5 and 1)
         <(2,2) ellipse (2 and 1.5)>
\tzellipse[fill=green,even odd rule](3,1)(1 and 1)
         <(3,1) ellipse (.5 and .8)>
\end{tikzpicture}
```



16 Arcs and Wedges

16.1 `\tzarc(')`: Centered arcs

16.1.1 Arcs

`\tzarc` draws an arc around a specified center coordinate.

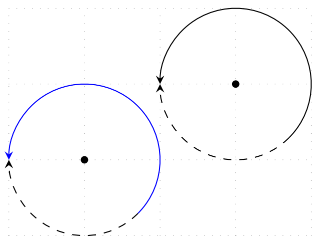
```
% syntax: minimum
\tzarc(<coor>)(<angA:angB:radius>)
% syntax: full
\tzarc[<opt>]<shift coor>"<path name>"
      (<coor>)(<angA:angB:radius>){<text>}[<node opt>]<code.append>
% defaults
  []<>""(<m>)(<m>){}[]<>
```

```
\tzarc(1,1)(30:120:1) % works like:
  \draw (1,1) ++(30:1) arc (30:120:1);
```

The *swap version* `\tzarc'` switches its drawing direction from `counterclockwise` to `clockwise` and vice versa.

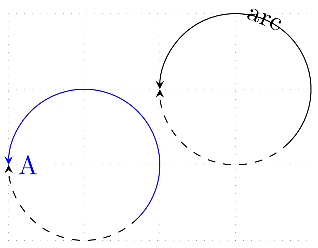
```
\tzarc'(1,1)(30:120:1) % works like:
  \draw (1,1) ++(30:1) arc (30:120-360:1);
```

```
% \tzarc, \tzarc'
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdots*(1,1)(3,2);
\tzarc [blue,->]   (1,1)(-45:180:1)
\tzarc'[dashed,->] (1,1)(-45:180:1)
\draw [->](3,2) ++(-45:1) arc (-45:180:1);
\draw [dashed,->](3,2) ++(-45:1) arc (-45:180-360:1);
\end{tikzpicture}
```



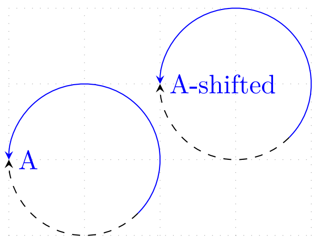
You can add text along the arc by specifying the options `{<text>}` and `[<node opt>]` immediately after the two mandatory arguments.

```
% \tzarc('): adding text
\begin{tikzpicture}
\tzhelplines(4,3)
\tzarc [blue,->]  (1,1)(-45:180:1){A}[r]
\tzarc'[dashed,->](1,1)(-45:180:1)
\tzarc[->]        (3,2)(-45:180:1){arc}[midway,sloped]
\tzarc'[dashed,->](3,2)(-45:180:1)
\end{tikzpicture}
```



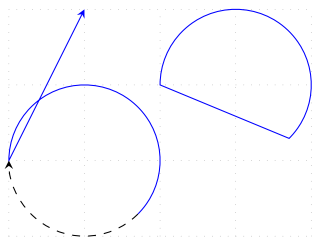
You can move arcs by specifying the option `<shift coor>` just before the center coordinate.

```
% \tzarc('): shift
\begin{tikzpicture}
\tzhelplines(4,3)
\tzarc [blue,->]       (1,1)(-45:180:1){A}[r]
\tzarc'[dashed,->]     (1,1)(-45:180:1)
\tzarc [blue,->]  <2,1>(1,1)(-45:180:1){A-shifted}[r]
\tzarc'[dashed,->]<2,1>(1,1)(-45:180:1)
\end{tikzpicture}
```



You can also extend the path of `\tzarc` by specifying the last option `<code.append>` with Ti*k*Z code written in it. For example, `<--cycle>` makes a closed path.

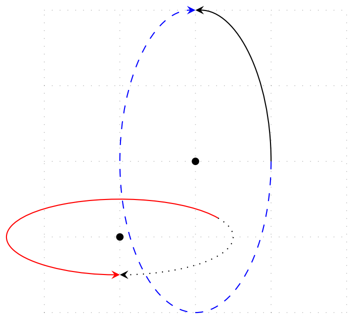
```
% \tzarc('): <code.append>
\begin{tikzpicture}
\tzhelplines(4,3)
\tzarc [blue,->](1,1)(-45:180:1)<--++(1,2)>
\tzarc'[dashed,->]     (1,1)(-45:180:1)
\tzarc [blue,->]  <2,1>(1,1)(-45:180:1)<--cycle>
\end{tikzpicture}
```



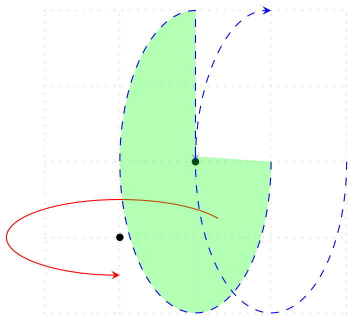
16.1.2 Elliptical arcs

`\tzarc` draws an elliptical arc if you specify `x-radius` and `y-radius`.

```
% \tzarc: elliptical
\begin{tikzpicture}
\tzhelplines(4,4)
\tzarc[->,red](1,1)(30:270:1.5 and 0.5)
\tzarc'[->,dotted](1,1)(30:270:1.5 and 0.5)
\tzdots*(1,1)(2,2);
\tzarc[->,blue,dashed](2,2)(0:-270:1 and 2)
\tzarc'[->](2,2)(0:-270:1 and 2)
\end{tikzpicture}
```



```
% \tzarc: elliptical
\begin{tikzpicture}
\tzhelplines(4,4)
\tzarc[->,red](1,1)(30:270:1.5 and 0.5)
\tzdots*(1,1)(2,2);
\tzarc[->,blue,dashed,fill=green,fill opacity=.3]
       (2,2)(0:-270:1 and 2)<--(2,2)> % code.append
\tzarc[->,blue,dashed]
  <1,0>(2,2)(0:-270:1 and 2) % shift
\end{tikzpicture}
```



16.2 \tzarcfrom('): Arcs as in Ti*k*Z

\tzarcfrom draws an arc starting from a specified point, like Ti*k*Z does.

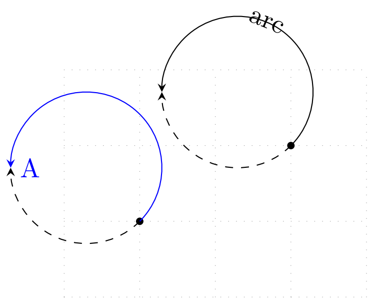
```
% syntax: minimum
\tzarcfrom(<coor>)(<angA:angB:radius>)
% syntax: full
\tzarcfrom[<opt>]<shift coor>"<path name>"
      (<coor>)(<angA:angB:radius>){<text>}[<node opt>]<code.append>
% defaults
  []<>""(<m>)(<m>){}[]<>
```

```
\tzarcfrom(1,1)(30:120:1) % works like:
  \draw (1,1) arc (30:120:1);
```

Everything else is the same as in \tzarc.

\tzarcfrom' is the *swap version* of \tzarcfrom.

```
% \tzarcfrom
\begin{tikzpicture}
\tzhelplines(4,3)
\tzdots*(1,1)(3,2);
\tzarcfrom [blue,->]  (1,1)(-45:180:1){A}[r]
\tzarcfrom'[dashed,->](1,1)(-45:180:1)
\tzarcfrom[->](3,2)(-45:180:1){arc}[midway,sloped]
\tzarcfrom'[dashed,->](3,2)(-45:180:1)
\end{tikzpicture}
```

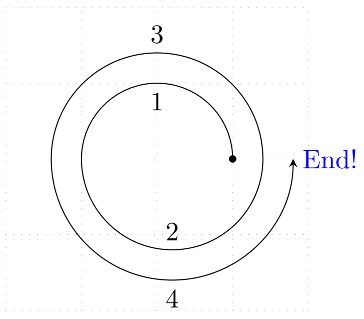


16.3 \tzarcsfrom: Connected arcs: Semicolon version

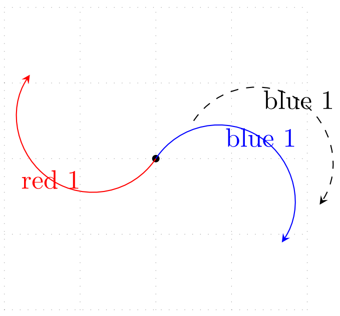
The macro `\tzarcsfrom` (i.e. `\tzarcs + from`) accepts an arbitrary number of parenthesis arguments in the form of `(<angA:AngB:radius>)` following the start coordinate. You need to enter a *semicolon* to indicate when the repetition ends.

```
% syntax: minimum
\tzarcsfrom(<start coor>)
           (<angA:angB:radius>)..repeated..(<angA:angB:radius>) ;
% syntax: full
\tzarcsfrom[<opt>]<shift coor>"<path name>"
           (<start coor>)(<angA:angB:radius>){<text>}[<node opt>]
                         ..repeated..(){}[] ; <code.append>
% defaults
  []<>""(<m>)(<m>){}[]..repeated..(){}[] ; <>
```

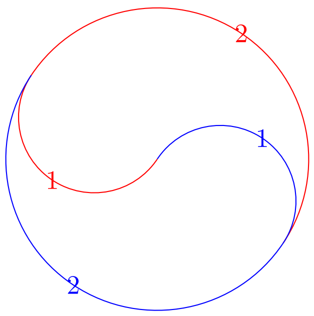
```
% \tzarcfrom: adding text, <code.append>
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoor*(3,2)(A)
\tzarcsfrom[->,auto](A)
      (0:180:1){1}[midway]
      (180:360:1.2){2}[midway]
      (0:180:1.4){3}[midway,swap]
      (180:360:1.6){4}[midway,swap];
      < node [right,blue] {End!} >
\end{tikzpicture}
```



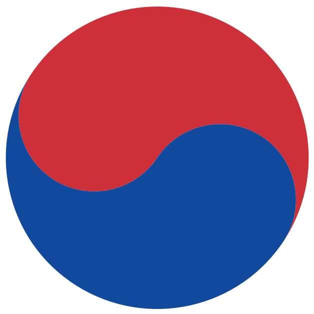
```
% \tzarcfrom: shift
% flag: step 1
\begin{tikzpicture}
\tzhelplines(-2,-2)(2,2)
\tzdot*(0,0)
\edef\x{atan(2/3)}
\tzarcsfrom[red,->](0,0)
      (-\x:-\x-180:1){red 1}[midway];
\tzarcsfrom[blue,->](0,0)
      (180-\x:-\x:1){blue 1}[midway];
\tzarcsfrom[dashed,->]<.5,.5>(0,0)
      (180-\x:-\x:1){blue 1}[midway];
\end{tikzpicture}
```



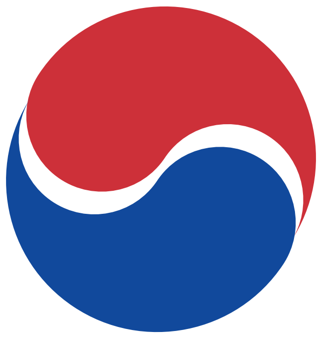
```
% flag: step 2
\begin{tikzpicture}
\edef\x{atan(2/3)}
\tzarcsfrom[red](0,0)
      (-\x:-\x-180:1){1}[midway]
      (-\x+180:-\x:2){2}[midway];
\tzarcsfrom[blue](0,0)
      (180-\x:-\x:1){1}[midway]
      (-\x:-\x-180:2){2}[midway];
\end{tikzpicture}
```



```
% flag: step 3
\definecolor{flagred}{RGB}{205,48,57}
\definecolor{flagblue}{RGB}{17,73,156}
\begin{tikzpicture}[scale=1]
\edef\x{atan(2/3)}
\tzarcsfrom[draw=none,fill=flagred](0,0)
      (-\x:-\x-180:1)
      (-\x+180:-\x:2)
      (-\x:-\x+180:1);<-- cycle>
\tzarcsfrom[draw=none,fill=flagblue](0,0)
      (180-\x:-\x:1)
      (-\x:-\x-180:2)
      (-\x+180:-\x+360:1);<-- cycle>
\end{tikzpicture}
```



```
% flag: shift
\definecolor{flagred}{RGB}{205,48,57}
\definecolor{flagblue}{RGB}{17,73,156}
\begin{tikzpicture}[scale=1]
\edef\x{atan(2/3)}
\tzarcsfrom[draw=none,fill=flagred]<.1,.3>(0,0)
      (-\x:-\x-180:1)
      (-\x+180:-\x:2)
      (-\x:-\x+180:1);<-- cycle>
\tzarcsfrom[draw=none,fill=flagblue](0,0)
      (180-\x:-\x:1)
      (-\x:-\x-180:2)
      (-\x+180:-\x+360:1);<-- cycle>
\end{tikzpicture}
```



16.4 Wedges

16.4.1 \tzwedge(')

\tzwedge draws a wedge around a specified center coordinate.

\tzwedge works similarly to \tzarc, but it forms a closed path from the center coordinate.

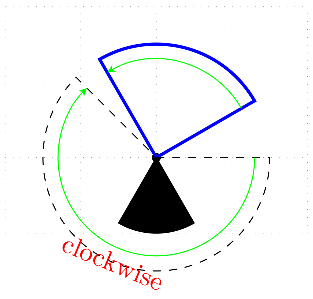
```
% syntax
\tzwedge[<opt>]<shift coor>"<path name>"
        (<coor>)(<angA:angB:radius>){<text>}[<node opt>]
% defaults
  []<>""(<m>)(<m>){}[midway]
```

```
\tzwedge(1,1)(30:120:1) % works like:
  \draw (1,1) -- ++(30:1) arc (30:120:1) -- cycle;
```

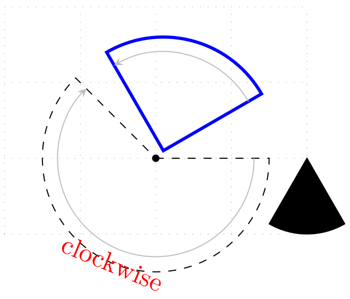
The macro \tzwedge' is the *swap version* of \tzwedge. It switches the drawing direction from counterclockwise to clockwise and vice versa.

```
\tzwedge'(1,1)(30:120:1) % works like:
  \draw (1,1) -- ++(30:1) arc (30:120-360:1) -- cycle;
```

```
% \tzwedge, \tzwedge'
\begin{tikzpicture}
\tzcoor*(2,1)(A)(3pt)
\tzhelplines(4,3)
\tzwedge[blue,very thick](A)(30:120:1.5)
\tzarc[->,green]         (A)(30:120:1.3)
\tzwedge'[dashed](A)(0:135:1.5){clockwise}[sloped,red]
\tzarc'[->,green](A)(0:135:1.3)
\tzwedge'[fill](A)(-60:240:1)
\end{tikzpicture}
```



```
% \tzwedge('): shift
\begin{tikzpicture}
\tzcoor*(2,1)(A)
\tzhelplines(4,3)
\tzwedge[blue,very thick]<.1,.1>(A)(30:120:1.5) % shift
\tzarc[->,lightgray]<.1,.1>(A)(30:120:1.3)      % shift
\tzwedge'[dashed](A)(0:135:1.5){clockwise}[sloped,red]
\tzarc'[->,lightgray](A)(0:135:1.3)
\tzwedge'[fill]<2,0>(A)(-60:240:1)              % shift
\end{tikzpicture}
```



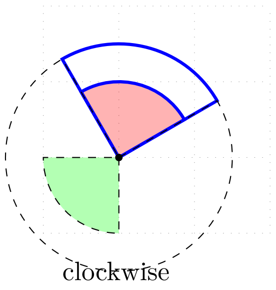
16.4.2 `\tzwedge*(')`

The starred version `\tzwedge*` works just like `\tzwedge`, but it fills wedges with `black!50` with `fill opacity=.3` and `fill opacity=1`, by default. You can change the defaults with `\settzfillcolor` and `\settzfillopacity`. You can also change the fill opacity by specifying the last optional argument `{<fill opacity>}`.

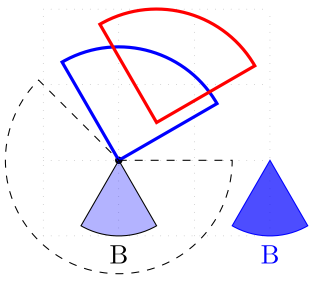
```
% syntax
\tzwedge*[<opt>]<shift coor>"<path name>"
         (<coor>)(<angA:angB:radius>){<text>}[<node opt>]{<fill opacity>}
% defaults
  [fill=black!50,fill opacity=.3,text opacity=1]<>""(<m>)(<m>){}[midway]{.3}
```

`\tzwedge*'` is the *swap version* of `\tzwedge*`.

```
% \tzwedge*, \tzwedge*'
\begin{tikzpicture}[->,>=stealth]
\tzhelplines(3,3)
\tzwedge[very thick,blue](1,1)(30:120:1.5)
\tzwedge*[-,very thick,blue,fill=red](1,1)(30:120:1)
\tzwedge'[dashed](1,1)(30:120:1.5){clockwise}[pos=.45]
\tzwedge*'[dashed,fill=green](1,1)(-90:180:1)
\tzdot*(1,1)
\end{tikzpicture}
```



```
% \tzwedge*('): shift, fill opacity
\begin{tikzpicture}
\tzdot*(1,1)
\tzhelplines(3,3)
\tzwedge[blue,very thick](1,1)(30:120:1.5)
\tzwedge[red,very thick]<.5,.5>(1,1)(30:120:1.5)
\tzwedge'[dashed](1,1)(0:135:1.5)
\tzwedge*[fill=blue](1,1)(300:240:1){B}[b]
\tzwedge*[blue]<2,0>(1,1)(300:240:1){B}[b]{.7}
\end{tikzpicture}
```



Part IV

Plotting Graphs

17 Axes

17.1 Draw axes

17.1.1 \tzaxes

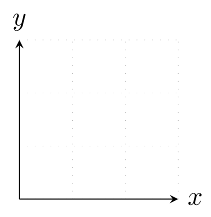
Basically, `\tzaxes(<x1,y1>)(<x2,y2>)` draws the $x$ axis from `<x1>` to `<x2>` and the $y$ axis from `<y1>` to `<y2>`. The coordinate `(<x1,y1>)` represents the origin and `(<x2,y2>)` represents the opposite corner of the rectangle formed by the two coordinates.

`\tzaxes` takes only one coordinate `(<x2,y2>)` as a mandatory argument, in which case the coordinate `(<x1,y1>)` is considered as `(0,0)`.

```
% syntax: minimal
\tzaxes(<x2,y2>){<x-axis name>}{<y-axis name>}
% syntax: full
\tzaxes[<opt>]<x-shift,y-shift>(<x1,y1>)(<x2,y2>)
        {<x-axis name>}[<node opt>]{<y-axis name>}[<node opt>]
% defaults
  [->]<0,0>(0,0)(<m>){}[right]{}[above]
% arguments
  [#1]: line style, arrow type (for x-axis & y-axis)
  <#2>: axes shift coor     %% axes intersect at (#2)
  (#3): (x1,y1)             %% origin: if omitted, regarded as (0,0)
  (#4): (x2,y2)             %% opposite corner: mandatory
  {#5}: x-axis name
  [#6]: x-axis name option %% node option
  {#7}: y-axis name
  [#8]: y-axis name option %% node option
```

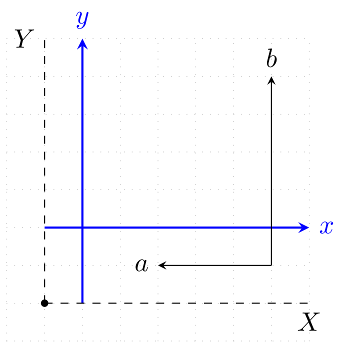
Here, `(<m>)` stands for a mandatory argument.

```
% tzaxes
\begin{tikzpicture}[scale=.7]
\tzhelplines(3,3)
\tzaxes(3,3){$x$}{$y$}
\end{tikzpicture}
```



Shift By default, the $x$ and $y$ axes intersect at $(0,0)$. Specifying the option `<x-shift,y-shift>` moves the axes to intersect at (`<x-shift,y-shift>`).

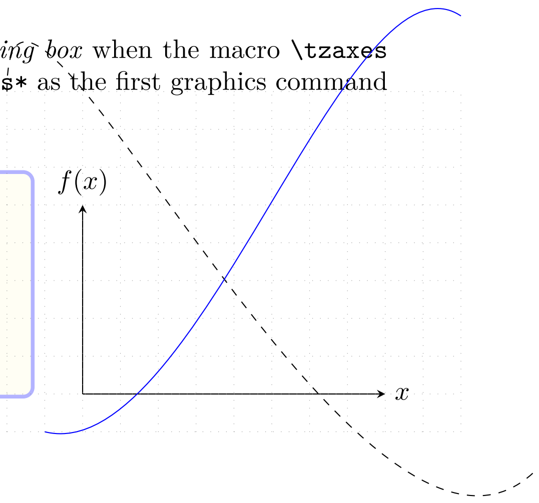
```
% \tzaxes: shift
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(7,7)
\tzshoworigin*
\tzaxes[thick,blue]<1,2>(7,7){$x$}{$y$}     %%
\tzaxes[-,dashed](7,7){$X$}[below]{$Y$}[left]
\tzaxes<6,1>(6,1)(3,6){$a$}[left]{$b$}      %%
\end{tikzpicture}
```



17.1.2 \tzaxes*

The starred version `\tzaxes*` sets the current state to a *bounding box* when the macro `\tzaxes` execution is complete. It is recommended for you to use `\tzaxes*` as the first graphics command in `tikzpicture` environment or before any larger graphics.

```
\begin{tikzpicture}[scale=.5]
\tzaxes*(8,5){$x$}{$f(x)$} % bounding box
\tzhelplines(-2,-1)(10,8)
\tzto[out=90,in=-135,dashed](-2,8)(12,-2)
\tzbezier[blue](-1,-1)(3,-2)(7,12)(10,10)
\end{tikzpicture}
```



17.2 \tzaxisx and \tzaxisy

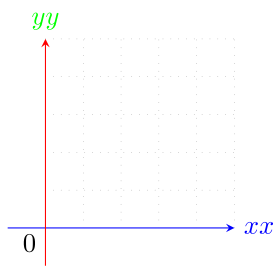
`\tzaxisx` draws only the x axis.

```
% syntax
\tzaxisx[<opt>]<y-shift>{<from>}{<to>}{<x-axis name>}[<node opt>]
% defaults
  [->]<0>{<m>}{<m>}{}[right]
% arguments:
  [#1]: line style, arrow type (for x-axis)
  <#2>: y-shift of x-axis
  {#3}: x-axis starts from %% mandatory
  {#4}: x-axis runs to     %% mandatory
  {#5}: x-axis name
  [#6]: x-axis name option %% node option
```

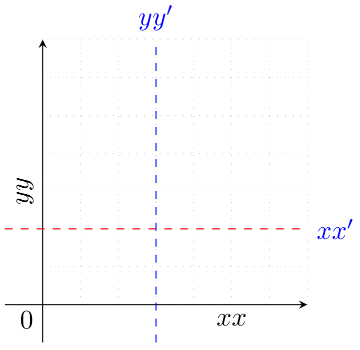
`\tzaxisy` draws only the y axis.

```
% syntax
\tzaxisy[<opt>]<x-shift>{<from>}{<to>}{<y-axis name>}[<node opt>]
% defaults
  [->]<0>{<m>}{<m>}{}[above]
% arguments:
  [#1]: line style, arrow type (for y-axis)
  <#2>: x-shift of y axis
  {#3}: y-axis starts from %% mandatory
  {#4}: y-axis runs to     %% mandatory
  {#5}: y-axis name
  [#6]: y-axis name option %% node option
```

```
% \tzaxisx, \tzaxisy
\begin{tikzpicture}[scale=.5]
\tzshoworigin
\tzhelplines(5,5)
\tzaxisx[blue]{-1}{5}{$xx$}
\tzaxisy[red] {-1}{5}{$yy$}[green]
\end{tikzpicture}
```



```
% \tzaxisx, \tzaxisy: shift
\begin{tikzpicture}[scale=.5]
\tzhelplines(7,7)
\tzshoworigin
\tzaxisx{-1}{7}{$xx$}[near end,below]
\tzaxisy{-1}{7}{$yy$}[midway,sloped,auto]
\tzaxisx[-,dashed,red]<2>{-1}{7}{$xx'$}[blue]
\tzaxisy[-,dashed,blue]<3>{-1}{7}{$yy'$}
\end{tikzpicture}
```



17.3 Display the origin

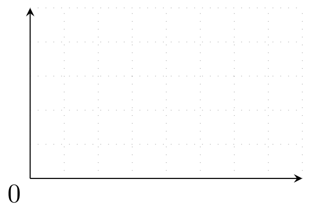
17.3.1 \tzshoworigin

`\tzshoworigin` prints '0' (approximately) at the bottom left of the origin `(0,0)`, by default.

```
% syntax
\tzshoworigin<shift coor>(<origin>){<text>}[<node opt>]
% default
  <>(0,0){0}[below left,text height=1.25ex,text depth=.25ex]
```

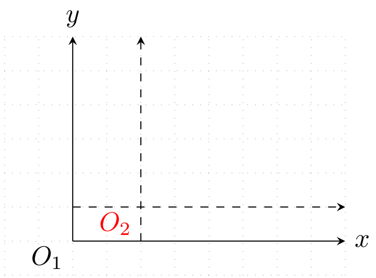
All arguments of `\tzshoworigin` are optional.

```
% \tzshoworigin
\begin{tikzpicture}[scale=.45]
\tzhelplines(8,5)
\tzshoworigin
\tzaxes(8,5)
\end{tikzpicture}
```



You can change the text by specifying the curly brace option `{<text>}`, like, for example, `\tzshoworigin{$O$}`. You can also change the coordinate of origin by the option `(<origin>)`. Specifying the option `<shift coor>` also moves the origin.

```
% \tzshoworigin: shift
\begin{tikzpicture}[scale=.45]
\tzhelplines(-2,-1)(8,6)
\tzshoworigin{$O_1$}            %%
\tzaxes(8,6){$x$}{$y$}
\tzshoworigin<2,1>{$O_2$}[red]  %%
\tzaxes[dashed]<2,1>(8,6)
\end{tikzpicture}
```

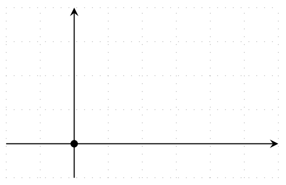


17.3.2 \tzshoworigin*

`\tzshoworigin*` prints a node dot at the origin with no text by default. Internally the dot is processed by `\tzdot*`. All arguments are optional.

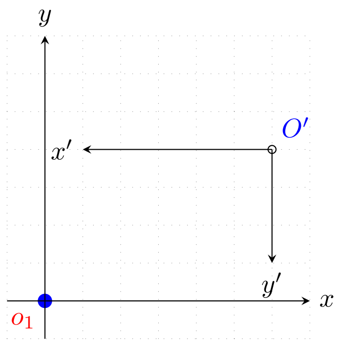
```
% syntax
\tzshoworigin*[<dot opt>]<shift coor>(<origin>){<text>}[<node opt>](<dot size>)
% default
 *[]<>(0,0){}[below left,text height=1.25ex,text depth=.25ex](2.4pt)
```

```
% \tzshoworigin*
\begin{tikzpicture}[scale=.45]
\tzhelplines(-2,-1)(6,4)
\tzshoworigin*
\tzaxes(-2,-1)(6,4)
\end{tikzpicture}
```



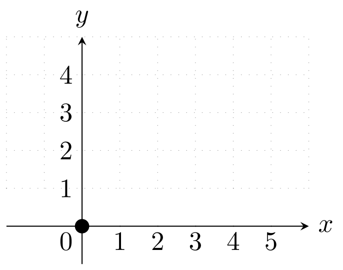
You can add text with the option `{<text>}`. The default size of the dot is `2.4pt`, and it can be changed with the last option `(<dot size>)`. You can change the dot style using the first optional argument `[<dot opt>]`. You can also move the dot by specifying the option `<shift coor>`.

```
% \tzshoworigin*
\begin{tikzpicture}[scale=.5]
\tzhelplines(-1,-1)(7,7)
\tzshoworigin*[blue]{$o_1$}[red](5pt)
\tzaxes(-1,-1)(7,7){$x$}{$y$}
\tzaxes<6,4>(6,4)(1,1){$x'$}[left]{$y'$}[below]
\tzshoworigin*[fill=none]<6,4>{$O'$}[blue,ar](3pt)
\end{tikzpicture}
```



Remark: For `\tzshoworigin*`, text for the origin and the dot are placed independently. In other words, the position of node text does not depend on the size of a node dot. (In fact, the node text for the origin should look good with the 'ticks labels', so it was not designed as a `label` for the node dot. This also means that the origin text cannot be positioned by an `<angle>`.)

```
% \tzshoworigin* (with tick labels)
\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,1)(6,5)
\tzshoworigin*{0}(5pt)
\tzaxes(-2,-1)(6,5){$x$}{$y$}
\tzticks{1,...,5}{1,...,4}
\end{tikzpicture}
```



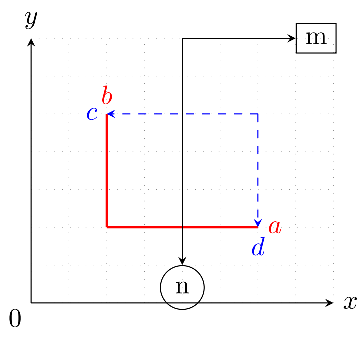
17.4 \tzaxesL('): L-type axes

`\tzaxesL` is similar to `\tzaxes`, but it draws only the 'L' type axes with `(<x1,y1>)` as the origin and `(<x2,y2>)` as the opposite corner of the rectangle. Those two coordinates are mandatory.

```
% syntax
\tzaxesL[<opt>]<shift coor>(<x1,y1>)(<x2,y2>)
        {<x-axis name>}[<node opt>]{<y-axis name>}[<node opt>]
% defaults
  []<>(<m>)(<m>){}[right]{}[above]
% arguments:
  [#1]: line style, arrow type
  <#2>: shift coordinate
  (#3): (x1,y1)            %% mandatory
  (#4): (x2,y2)            %% mandatory
  {#5}: x-axis name
  [#6]: x-axis name option %% node option
  {#7}: y-axis name
  [#8]: y-axis name option %% node option
```

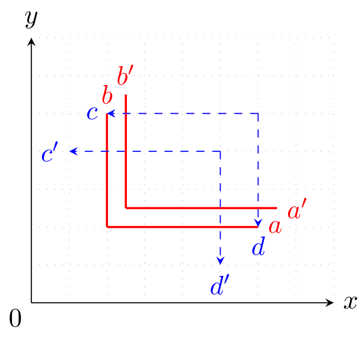
The *swap version* `\tzaxesL'` swaps `(<x1,y1>)` and `(<x2,y2>)`. That is, `\tzaxesL'(A)(B)` is equivalent to `\tzaxesL(B)(A)`.

```
% \tzaxesL, \tzaxesL'
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,7)
\tzshoworigin
\tzaxes(8,7){$x$}{$y$}
\tzaxesL[red,thick](2,2)(6,5){$a$}{$b$}
\tzaxesL'[blue,dashed,->](2,2)(6,5){$c$}{$d$}
\tzaxesL'[->](7,1)(4,7){m}[draw,r]{n}[draw,circle]
\end{tikzpicture}
```



The option `<shift coor>` moves the whole L-type axes. The empty option `<>` is not allowed.

```
% \tzaxesL, \tzaxesL': shift
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,7)
\tzshoworigin
\tzaxes(8,7){$x$}{$y$}
\tzaxesL[red,thick](2,2)(6,5){$a$}{$b$}
\tzaxesL[red,thick]<.5,.5>(2,2)(6,5){$a'$}{$b'$}
\tzaxesL'[blue,dashed,->](2,2)(6,5){$c$}{$d$}
\tzaxesL'[blue,dashed,->]<-1,-1>(2,2)(6,5){$c'$}{$d'$}
\end{tikzpicture}
```



18 Ticks

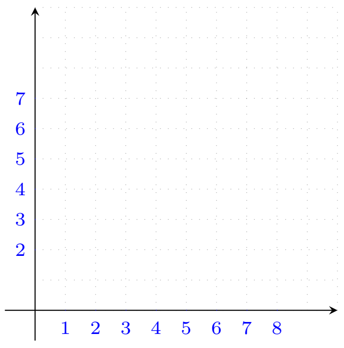
18.1 \tzticks: Tick labels

By default, `\tzticks` prints tick labels and draws zero length tick marks, i.e. from `(0pt)` to `(0pt)`.

```
% syntax: minimal
\tzticks{<x-ticks pos>}{<y-ticks pos>}
% syntax: medium
\tzticks(<x-from:x-to>){<x-ticks pos/labels>}[<node opt>]
        (<y-from:y-to>){<y-ticks pos/labels>}[<node opt>]
% syntax: full
\tzticks[<opt>]<x-shift,y-shift>
        (<x-from:x-to>){<x-ticks pos/labels>}[<node opt>]
        (<y-from:y-to>){<y-ticks pos/labels>}[<node opt>]
% defaults
  []<0,0>(0pt:0pt){<m>}[text height=1.25ex,text depth=.25ex,below](0pt:0pt){}[left]
```

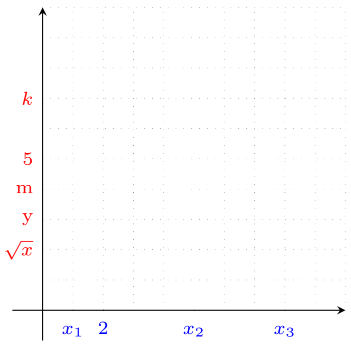
Tick labels Internally, `\tzticks` uses Ti*k*Z's `foreach` operation. So you need to provide comma separated lists to print tick labels. If only one comma separated list is specified, it is for x tick labels.

```
% \tzticks
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\tzhelplines(10,10)
\tzaxes(-1,-1)(10,10)
\tzticks[blue]{1,...,8}{2,...,7}
\end{tikzpicture}
```



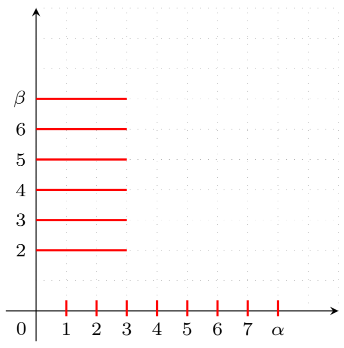
You can change the numbered labels to a different format with slashes and other text, as follows: `<number>/<other text>`.

```
% \tzticks: tick labels
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\tzhelplines(10,10)
\tzaxes(-1,-1)(10,10)
\tzticks{1/$x_1$,2,5/$x_2$,8/$x_3$}[blue]
        {2/$\sqrt{x}$,3/y,4/m,5,7/$k$}[red]
\end{tikzpicture}
```



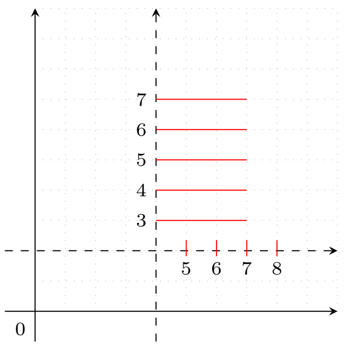
Tick marks To prints tick marks you need to specify `(<x-from:x-to>)` for x ticks and/or `(<y-from:y-to>)` for y ticks. (The default is `(0pt:0pt)`.)

```
% \tzticks: tick marks with (<a pt:b pt>)
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\tzhelplines(10,10)
\tzshoworigin
\tzaxes(-1,-1)(10,10)
\tzticks[draw=red,thick]
        (-5pt:10pt){1,...,7,8/$\alpha$}
        (0pt:3cm)  {2,...,6,7/$\beta$}
\end{tikzpicture}
```



Shift The position of tick labels does not depend on the length of the tick marks. To change the position, use the option `<x-shift,y-shift>`, where `<x-shift>` is for `y-ticks` and `<y-shift>` is for `x-ticks`.

```
% \tzticks: shift
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\tzhelplines(10,10)
\tzshoworigin
\tzaxes(-1,-1)(10,10)
\tzaxes[dashed]<4,2>(-1,-1)(10,10)
\tzticks[draw=red]<4,2>
        (-5pt:10pt){5,...,8}
        (0pt:3cm)  {3,...,7}
\end{tikzpicture}
```

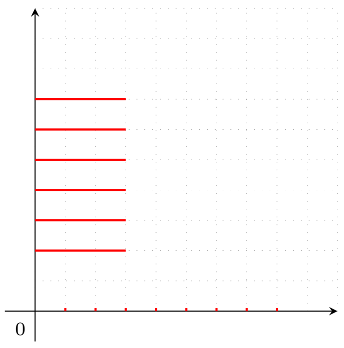


18.2 \tzticks*: Tick marks

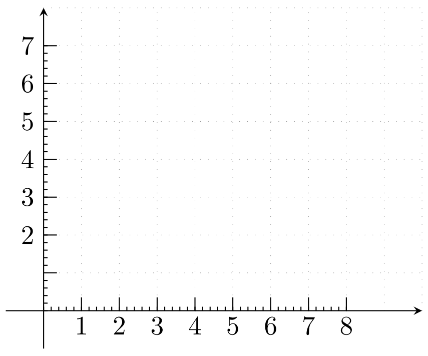
The starred version `\tzticks*` always ignores all tick labels and draws tick marks from `0pt` to `3pt`, by default.

```
% syntax: minimal
\tzticks*{<x-ticks pos>}{<y-ticks pos>}
% syntax: medium
\tzticks*(<x-from:x-to>){<x-ticks pos>}(<y-from:y-to>){<y-ticks pos>}
% syntax: full
\tzticks*[<opt>]<x-shift,y-shift>
         (<x-from:x-to>){<x-ticks pos>}(<y-from:y-to>){<y-ticks pos>}
% defaults
  []<0,0>(0pt:3pt){<m>}(0pt:3pt){}
% starred(*) version always suppresses tick labels
```

```
% \tzticks*
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\tzhelplines(10,10)
\tzshoworigin
\tzaxes(-1,-1)(10,10)
\tzticks*[draw=red,thick]
          {1,...,7,8/$\alpha$} % label ignored
 (0pt:3cm){2,...,6,7/$\beta$}  % label ignored
\end{tikzpicture}
```



```
% \tzticks(*)
\begin{tikzpicture}[scale=.5]
\tzhelplines(10,8)
\tzaxes(-1,-1)(10,8)
\tzticks*
  {0,0.2,...,8}
  {0,0.2,...,7} % default (0pt:3pt)
\tzticks*
  (0pt:10pt){1,...,8}
  (0pt:10pt){1,...,7}
\tzticks
  {1,...,8}
  {2,...,7}      % default (0pt:0pt)
\end{tikzpicture}
```



18.3 \tzticksx(*) and \tzticksy(*)

You can handle x ticks and y ticks independently.

X ticks `\tzticksx` only prints x-tick labels but not tick marks, by default. To prints tick marks you need to specify `(<x-from>:<x-to>)`.

```
% syntax
\tzticksx[<opt>]<y-shift>(<from>:<to>){<x-tick pos/labels>}[<node opt>]
% defaults
  []<>(0pt:0pt){<m>}[text height=1.25ex,text depth=.25ex,below]
```

`\tzticksx*` only prints x-tick marks from `0pt` to `3pt`, by default, suppressing tick labels.

```
% syntax:
\tzticksx*[<opt>]<y-shift>(<from>:<to>){<xtick pos>}
% defaults
 *[]<>(0pt:3pt){<m>}
% starred(*) version always suppresses tick labels
```

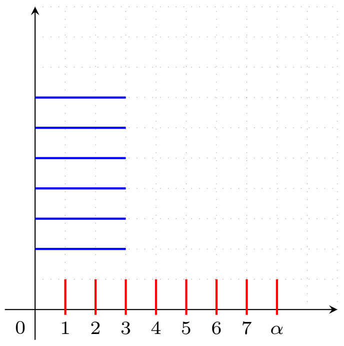
Y ticks `\tzticksy` only prints y-tick labels but not tick marks, by default. To prints tick marks you need to specify `(<x-from>:<x-to>)`.

`\tzticksy*` only prints y-ticks from `0pt` to `3pt` by default, suppressing tick labels.

```
% syntax:
\tzticksy[<opt>]<x-shift>(<from:to>){<y-ticks pos/labels>}[<node opt>]
% defaults
  []<>(0pt:0pt){<m>}[]
```

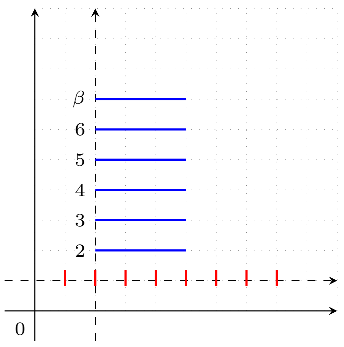
```
% syntax
\tzticksy*[<opt>]<x-shift>(<from:to>){<yticks pos>}
% defaults
  []<>(0pt:0pt){<m>}
% starred(*) version suppresses tick labels
```

```
% \tztickx(*), \tzticky(*)
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\tzhelplines(10,10)
\tzshoworigin
\tzaxes(-1,-1)(10,10)
\tzticksx [draw=red,thick]
          (-5pt:1cm){1,...,7,8/$\alpha$}
\tzticksy*[draw=blue,thick]
          (0pt:3cm){2,...,6,7/$\beta$} % labels ignored
\end{tikzpicture}
```



Shift The options `<y-shift>` and `<x-shift>` move x-ticks and y-ticks, respectively.

```
% \tztickx(*), \tzticky(*): shift
\begin{tikzpicture}[scale=.4,font=\scriptsize]
\tzhelplines(10,10)
\tzshoworigin
\tzaxes(-1,-1)(10,10)
\tzaxes[dashed]<2,1>(-1,-1)(10,10)
\tzticksx*[draw=red,thick]
  <1>(-5pt:10pt){1,...,7,8/$\alpha$} % labels ignored
\tzticksy [draw=blue,thick]
  <2>(0pt:3cm){2,...,6,7/$\beta$}
\end{tikzpicture}
```



19 Projections

19.1 \tzproj(*)

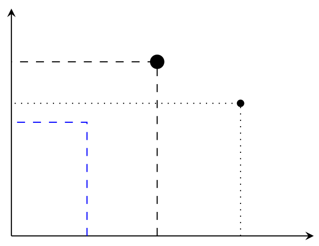
\tzproj accepts a mandatory coordinate and draws perpendicular lines onto each axis from the coordinate. The lines are dotted, by default.

```
% syntax: minimum
  \tzproj(<coor>)
% syntax: medium
  \tzproj*(<coor>){<x-text>}[<node opt>]{<y-text>}[<node opt>]
% syntax:
  \tzproj*[<opt>]<x-shift,y-shift>(<coor>)
         {<x-text>}[<node opt>]{<y-text>}[<node opt>](<dot size>)
% defaults
 *[dotted]<0,0>(<m>){}[text height=1.25ex,text depth=.25ex,below]{}[left](2.4pt)
```

\tzproj* additionally prints a 'black node dot' of the size 2.4pt, by default. Internally, the node dot is processed by \tzdot*. The first option <opt> does not control the node dot.

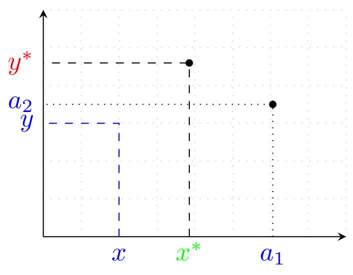
Dot size You can only control the size of dots by the last optional argument (<dot size>) or by the Three Ways on page 41. If you want to control fill or color of dots, use \tzdot* separately.

```
%\tzproj(*)
\begin{tikzpicture}[scale=.5]
\tzaxes(8,6)
\tzproj[dashed,blue](2,3)
\tzcoors(30:7)(A)(50:6)(B);
\tzproj*(A)
\tzproj*[dashed,text=blue](B)(5pt)
\end{tikzpicture}
```



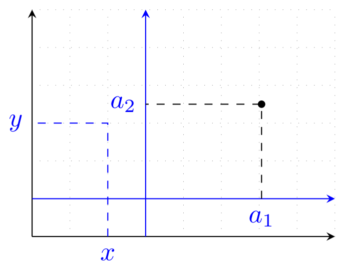
Adding text You can also add text around the projection point on each axis by the option {<text>}. The position and color of the text is controlled by the option [<node option>]. The default position is (approximately) [below] for the x axis and [left] for the y axis.

```
% \tzproj(*): adding text
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(8,6)
\tzproj[dashed,blue](2,3){$x$}{$y$}
\tzcoors(30:7)(A)(50:6)(B);
\tzproj*[text=blue](A){$a_1$}{$a_2$}
\tzproj*[dashed](B){$x^*$}[green]{$y^*$}[red]
\end{tikzpicture}
```



Projection shift Specifying the option <x-shift,y-shift> moves the projection point and text on each axis.

```
% \tzproj(*): projection shift
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(8,6)
\tzproj[dashed,blue](2,3){$x$}{$y$}
\tzcoors(30:7)(A)(50:6)(B);
\tzaxes[blue]<3,1>(8,6)
\tzproj*[dashed,text=blue]<3,1>(A){$a_1$}{$a_2$} %%
\end{tikzpicture}
```



19.2 \tzprojx(*) and \tzprojy(*)

\tzprojx draws a dotted line, which is perpendicular to the x axis.

\tzprojx* additionally prints a 'black node dot' of the size 2.4pt, by default.

```
% syntax:
  \tzprojx[<opt>]<x-shift,y-shift>(<coor>){<x-text>}[<node opt>](<dot size>)
% defaults
  []<0,0>(<m>){}[text height=1.25ex,text depth=.25ex,below](2.4pt)
```

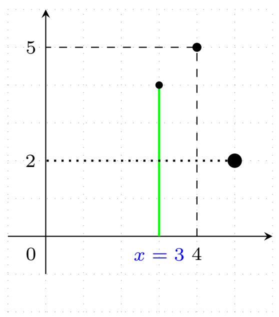
\tzprojy draws a dotted line, which is perpendicular to the y axis.

\tzprojy* additionally prints a 'black node dot' of the size 2.4pt, by default.

```
% syntax:
  \tzprojy[<opt>]<x-shift,y-shift>(<coor>){<y-text>}[<node opt>](<dot size>)
% defaults
  []<0,0>(<m>){}[left](2.4pt)
```

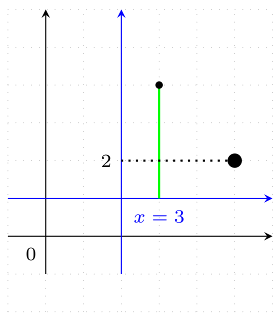
You can only control the size of dots by the last option (<dot size>). If you want to control fill or color of dots, use \tzdot* separately. You can also add text around the projection point on each axis by specifying the option {<x-text>} or {<y-text>} followed by the option [<node option>].

```
% \tzprojx(*), \tzprojy(*)
\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(-1,-2)(6,6)
\tzshoworigin
\tzaxes(-1,-1)(6,6)
\tzproj*[dashed](4,5){4}{5}(3pt)
\tzprojx*[green,thick,solid](3,4){$x=3$}[blue]
\tzprojy*[thick](5,2){2}(5pt)
\end{tikzpicture}
```



Specifying the option <x-shift,y-shift> with \tzprojx(*) and \tzprojy(*) moves the projection point and text accordingly.

```
% \tzprojx(*), \tzprojy(*): shift
\begin{tikzpicture}[scale=.5,font=\scriptsize]
\tzhelplines(-1,-2)(6,6)
\tzshoworigin
\tzaxes(-1,-1)(6,6)
\tzaxes[blue]<2,1>(-1,-1)(6,6)
\tzprojx*[green,thick,solid]<2,1>(3,4){$x=3$}[blue]
\tzprojy*[thick]<2,1>(5,2){2}(5pt)
\end{tikzpicture}
```



20 Plot Functions

20.1 \tzfn: Plot functions

20.1.1 Syntax

`\tzfn` plots a function of `\x`.

```
% syntax: minimum
\tzfn{<fn of \x>}[<domain>]
% syntax: medium
\tzfn{<fn of \x>}[<domain>]{<text>}[<pos>]
% syntax: full
\tzfn[<opt>]<shift coor>"<path name>"
     {<fn of \x>}[<domain>]{<text>}[<node opt>]<code.append>
% [<domain>] should be of the form [<from num:to num>]
% defaults
  [samples=200]<>""{<m>}[<m>]{}[]<>
```

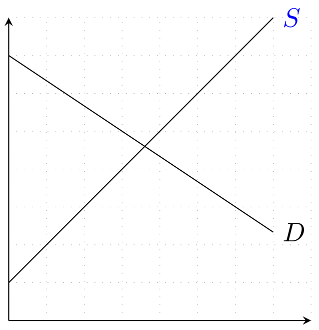
`\tzfn` takes two mandatory arguments: `{<fn of \x>}` and `[<domain>]`. The domain should be of the form `[<from num:to num>]`, like `[1:5]`.

```
\tzfn{2*\x+1}[1:5] % works like:
  \draw [samples=200,domain=1:5] plot (\x,{2*\x+1});
```

20.1.2 Define and name functions

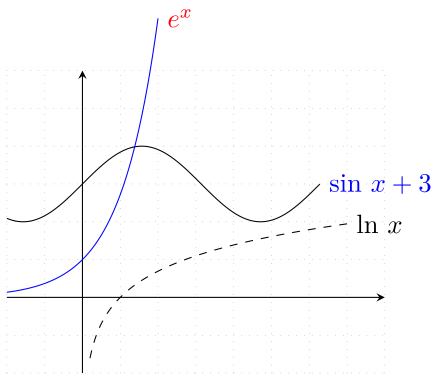
To use `\tzfn` you need to express a function as a function of `\x`. You can add text (as a function name) at the end of the graph, by specifying `{<text>}` and `[<node opt>]` immediately after the domain.

```
% \tzfn
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,8)
\tzaxes(8,8)
\def\Dx{7-2/3*\x}
\tzfn{\Dx}[0:7]{$D$}[r]
\tzfn{1+\x}[0:7]{$S$}[blue,r]
\end{tikzpicture}
```



You can also use the predefined functions of Ti*k*Z such as `sin`, `cos`, `ln`, `log10`, `log2`, `exp`, `sqrt`, and so on. (See Ti*k*Z manual.)

```
% \tzfn
\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,-2)(8,6)
\tzaxes*(-2,-2)(8,6)
\def\Fx{sin(\x r)+3}
\def\Gx{exp(\x)}
\def\Hx{ln(\x)}
\tzfn\Fx[-2:2*pi]{$\sin\,x+3$}[blue,r]
\tzfn[blue]\Gx[-2:2]{$e^x$}[red,r]
\tzfn[dashed]\Hx[.2:7]{$\ln\,x$}[r]
\end{tikzpicture}
```



20.1.3 Name paths: `name path`

You can name the path of `\tzfn` by specifying the option `"<path name>"` immediately before the mandatory argument `{<fn of \x>}`. You can use the path name to find intersection points.

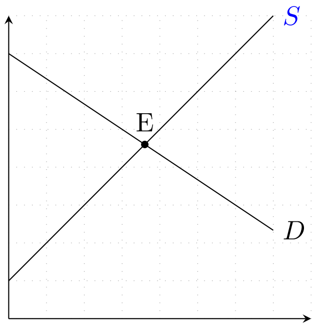
```
\tzfn"mypath"\Fx[1:5] % works like:
  \draw [samples=200,,domain=1:5,name path=mypath] plot (\x,{\Fx});
```

Remark: Suppose that the function's expression `<fn of \x>` consists *only of a macro name*, say `\Fx`. Then

- The macro name `Fx` (*without the backslash*) is *automatically assigned* to `<path name>`, unless you give another name.
- That is, `\tzfn\Fx` is equivalent to `\tzfn"Fx"\Fx`. (*You don't need to type the same thing twice.*)

```
\tzfn\Fx[1:5] % works like:
  \draw [samples=200,domain=1:5,name path=Fx] plot (\x,{\Fx});
```

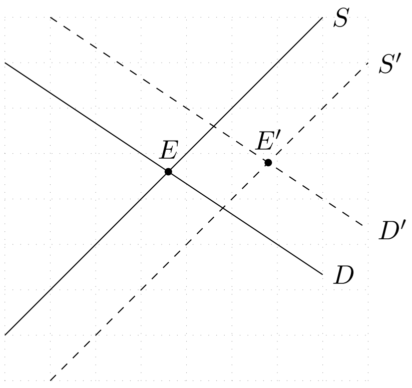
```
% \tzfn: name path: intersection point
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,8)
\tzaxes(8,8)
\def\Dx{7-2/3*\x}
\def\Sx{1+\x}
\tzfn"Dx"\Dx[0:7]{$D$}[r]       % name path = Dx
\tzfn    \Sx[0:7]{$S$}[blue,r] % name path = Sx
\tzXpoint*{Dx}{Sx}(E){E}
\end{tikzpicture}
```



20.1.4 Move graphs: `shift`

You can move the graph of `\tzfn` by specifying the option `<shift coor>` *before* the mandatory argument `{<fn of \x>}` or *immediately before* the option `"<path name>"`, if it exists. The empty shift option `<>` is not allowed.

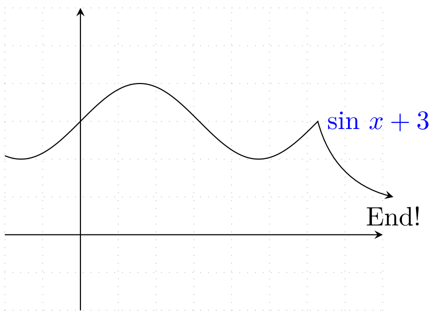
```
% \tzfn: shift
\begin{tikzpicture}[scale=.6]
\tzhelplines(8,8)
\def\Dx{7-2/3*\x}
\def\Sx{1+\x}
\tzfn        \Dx[0:7]{$D$}[right] % name path = Dx
\tzfn"supply"\Sx[0:7]{$S$}[right] % name path = supply
\tzXpoint*{Dx}{supply}(E){$E$}
\tzfn[dashed]<1,1>"demandA"\Dx[0:7]{$D'$}[right]
\tzfn[dashed]<1,-1>"supplyA"\Sx[0:7]{$S'$}[right]
\tzXpoint*{demandA}{supplyA}(E1){$E'$}
\end{tikzpicture}
```



20.1.5 Extend paths: `<code.append>`, `\tzfnAtBegin`, `\tzfnAtEnd`

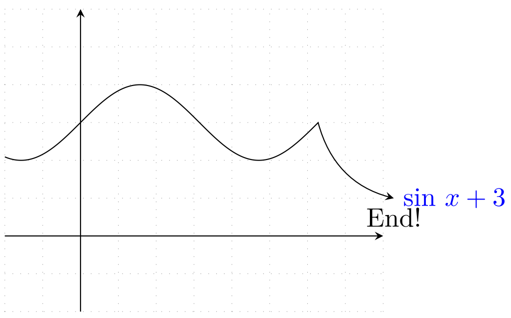
`<code.append>` You can extend the path created by `\tzfn` from the end of the graph, by writing Ti*k*Z code in the *last optional argument* `<code.append>`. Internally it adds the Ti*k*Z code to the path *after* the options `{<text>}` and `[<node opt>]`.

```
% \tzfn: <code.append>
\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,-2)(8,6)
\tzaxes*(-2,-2)(8,6)
\def\Fx{sin(\x r)+3}
\tzfn[->]\Fx[-2:2*pi]{$\sin\,x+3$}[blue,r]
        < to [bend right] ++(2,-2) node [b] {End!} >
\end{tikzpicture}
```



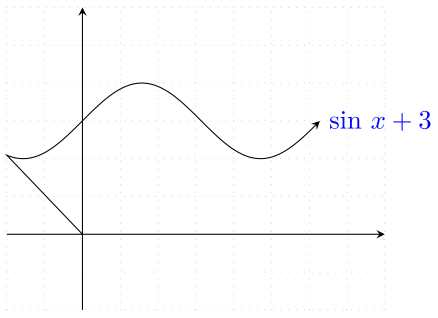
`\tzfnAtEnd` You can also extend the path with `\tzfnAtEnd`. Internally it adds Ti*k*Z code immediately *before* the options `{<text>}` and `[<node opt>]`. But you have to use `\tzfnAtEnd` (immediately) *before each* `\tzfn`.

```
% \tzfnAtEnd (before \tzfn)
\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,-2)(8,6)
\tzaxes*(-2,-2)(8,6)
\def\Fx{sin(\x r)+3}
\tzfnAtEnd{to [bend right] ++(2,-2) node [b] {End!}}
\tzfn[->]\Fx[-2:2*pi]{$\sin\,x+3$}[blue,r]
%        < to [bend right] ++(2,-2) node [b] {End!} >
\end{tikzpicture}
```



`\tzfnAtBegin` You can use `\tzfnAtBegin` (immediately) *before each* `\tzfn` to insert Ti*k*Z code before the path of `\tzfn`.

```
% \tzfnAtBegin (before \tzfn)
\begin{tikzpicture}[scale=.5]
\tzhelplines(-2,-2)(8,6)
\tzaxes*(-2,-2)(8,6)
\def\Fx{sin(\x r)+3}
\tzfnAtBegin{ (0,0) -- }
\tzfn[->]\Fx[-2:2*pi]{$\sin\,x+3$}[blue,r]
\end{tikzpicture}
```



20.2 \tzLFn: Plot linear functions

\tzLFn draws a linear function: $y = ax + b$.

- When you know two coordinates on a line, `\tzLFn(<coor1>)(<coor2>)` draws the line.
- When you know one coordinate and the slope of a line, `\tzLFn(<coor1>){<slope>}` draws the line.
- If you specify all the three arguments `(<coor1>)(<coor2>){<slope>}`, then the slope is ignored.

```
% syntax
\tzLFn[<opt>]<shift coor>"<path name>"
      (<coor1>)(<coor2>){<slope>}[<domain>]{<text>}[<node opt>]<code.append>
% defaults
[]<>""(<m>)(){1}[<m>]{}[]<>
```

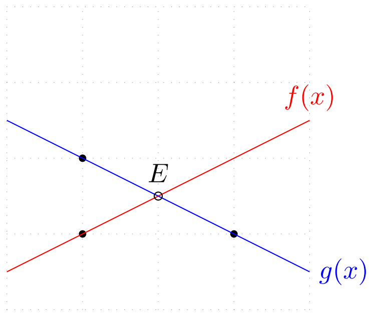
\tzLFn accepts two mandatory arguments: `(<coor1>)` and `[<domain>]`.

- The domain should be of the form `[<from num:to num>]`.
- If just one coordinate is specified without a slope, the slope is regarded as `1`, by default.

For example, `\tzLFn(1,1)(2,3)[0:4]` draws a line passing through two points: $(1, 1)$ and $(2, 3)$, over $0 \le x \le 4$. `\tzLFn(1,1){.5}[0:4]` draws a line passing through a point $(1, 1)$ with the slope `.5`, over $0 \le x \le 4$.

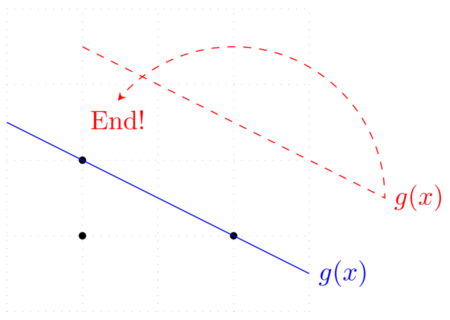
You can add text at the end of the line of \tzLFn by the options `{<text>}` and `[<node opt>]`. You can also name the path of \tzLFn by specifying the option `"<path name>"` immediately before the mandatory coordinate.

```
% \tzLFn
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors*(1,1)(A)(1,2)(B)(3,1)(C);
\tzLFn[red]"Fx"(A){.5}[0:4]{$f(x)$}[a]
\tzLFn[blue]"Gx"(B)(C)[0:4]{$g(x)$}[r]
\tzXpoint*[fill=none]{Fx}{Gx}(E){$E$}(3pt)
\end{tikzpicture}
```



You can move the line of \tzLFn by specifying the option `<shift coor>` immediately before the option `"<path name>"`. (The empty shift option `<>` is not allowed.) You can also expand the path of \tzLFn by writing Ti*k*Z code in the last optional argument `<code.append>`.

```
% \tzLFn: shift, expanding path
\begin{tikzpicture}
\tzhelplines(4,4)
\tzcoors*(1,1)(A)(1,2)(B)(3,1)(C);
\tzLFn[blue]"Gx"(B)(C)[0:4]{$g(x)$}[r]
\tzLFn[dashed,red,->]<1,1>"Gx"(B)(C)[0:4]{$g(x)$}[r]
      < arc (0:140:2) node [below] {End!} >
\end{tikzpicture}
```



20.3 Horizontal lines

20.3.1 \tzhfnat

\tzhfnat draws a horizontal line at a specified value of y.

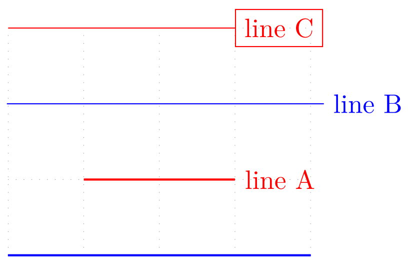
```
% syntax: minimal
\tzhfnat{<y-val>}[<domain>]
% syntax: full
\tzhfnat[<opt>]<shift coor>"<path name>"
        {<y-val>}[<domain>]{<text>}[<node opt>]<code.append>
% defaults
  []<>""{<m>}[west:east (of current bounding box)]{}[]<>
```

\tzhfnat accepts only one mandatory argument `{<y-val>}`. The domain is optional and should be of the form `[<from num:to num>]`. The default domain is from left to right of the `current bounding box`.

Remark: Internally, the default domain of \tzhfnat depends on the `current bounding box`.

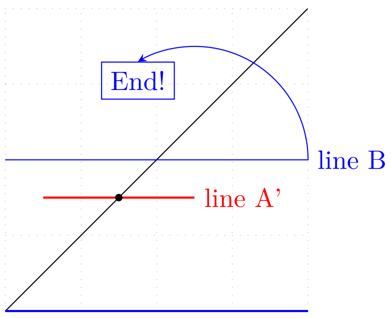
- Each \tzhfnat may draw a line with a (slightly) different length.
- If an appropriate `current bounding box` is not formed before \tzhfnat is executed, you will probably get an unexpected result.
- In that case, you can fix a `bounding box` in the beginning of the `tikzpicture` environment using macros such as \tzbbox, \tzhelplines*, \tzaxes*, or Ti*k*Z's \useasboundingbox.

```
% \tzhfnat
\begin{tikzpicture}
\tzhelplines(4,3)
\tzhfnat[blue,thick]{0}
\tzhfnat[red,thick]{1}[1:3]{line A}[r]
\tzhfnat[blue]{2}{line B}[r]
\tzhfnat[red]{3}[0:3]{line C}[draw=blue,red,r]
\end{tikzpicture}
```



You can name the path of \tzhfnat by the option `"<path name>"`. You can move the line by the option `<shift coor>`. You can also expand the path from the end of the line by writing Ti*k*Z code in the last optional argument `<code.append>`.

```
% \tzhfnat: shift, <code.append>
\begin{tikzpicture}
\tzhelplines*(4,4) % bounding box
\tzfn"XX"{\x}[0:4]
\tzhfnat[blue,thick]{0}
\tzhfnat[red,thick]<-.5,.5>"AA"{1}[1:3]{line A'}[r]
\tzXpoint*{AA}{XX}
\tzhfnat[blue,->]{2}{line B}[r]
        < arc (0:120:1.5) node [b,draw] {End!} >
\end{tikzpicture}
```



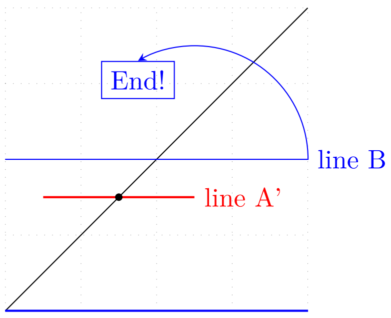
20.3.2 `\tzhfn`

`\tzhfn` accepts a coordinate as a mandatory argument and draws a horizontal line at the y value of the coordinate. For example, `\tzhfn(<x>,3)`, ignoring `<x>`, is equivalent to `\tzhfnat{3}`.

Everything else is the same as in `\tzhfnat`.

```
% syntax: minimal
\tzhfn(<coor>)[<domain>]
% syntax: full
  \tzhfn[<opt>]<shift coor>"<path name>"
        (<coor>)[<domain>]{<text>}[<node opt>]<code.append>
% defaults
  []<>""(<m>)[west:east (of current bounding box)]{}[]<>
```

```
% \tzhfn: shift, <code.append>
\begin{tikzpicture}
\tzhelplines*(4,4) % bounding box
\tzcoors(0,1)(A)(0,2)(B);
\tzfn"XX"{\x}[0:4]
\tzhfn[blue,thick](5,0) % x=5 ignored
\tzhfn[red,thick]<-.5,.5>"AA"(A)[1:3]{line A'}[r]
\tzXpoint*{AA}{XX}
\tzhfn[blue,->](B){line B}[r]
        < arc (0:120:1.5) node [b,draw] {End!} >
\end{tikzpicture}
```



20.4 Vertical lines

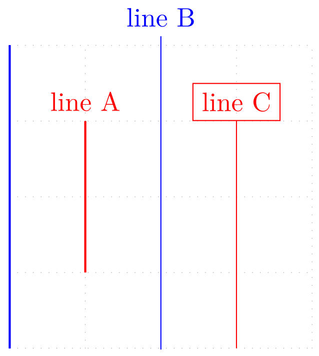
20.4.1 `\tzvfnat`

`\tzvfnat` draws a vertical line at a specified value of x.

```
% syntax: minimal
\tzvfnat{<x-val>}[<domain>]
% syntax: full
\tzvfnat[<opt>]<shift coor>"<path name>"
        {<x-val>}[<domain>]{<text>}[<node opt>]<code.append>
% defaults
  []<>""{<m>}[south:north (of current bounding box)]{}[]<>
```

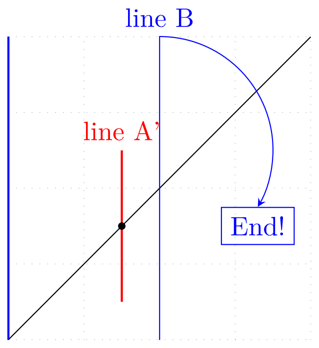
`\tzvfnat` accepts only one mandatory argument `{<x-val>}`. The domain is optional and should be of the form `[<from num:to num>]`. The default domain is from bottom to top of the `current bounding box`.

```
% \tzvfnat
\begin{tikzpicture}
\tzhelplines(4,4)
\tzvfnat[blue,thick]{0}
\tzvfnat[red,thick]{1}[1:3]{line A}[a]
\tzvfnat[blue]{2}{line B}[a]
\tzvfnat[red]{3}[0:3]{line C}[draw=blue,red,a]
\end{tikzpicture}
```



You can name the path of `\tzvfnat` by the option `"<path name>"`. You can move the line by the option `<shift coor>`. You can also expand the path from the end of the line by writing Ti*k*Z code in the last optional argument `<code.append>`.

```
% \tzvfnat: shift, <code.append>
\begin{tikzpicture}
\tzhelplines*(4,4) % bounding box
\tzfn"XX"{\x}[0:4]
\tzvfnat[blue,thick]{0}
\tzvfnat[red,thick]<.5,-.5>"AA"{1}[1:3]{line A'}[a]
\tzXpoint*{AA}{XX}
\tzvfnat[blue,->]{2}{line B}[a]
        < arc (90:-30:1.5) node [b,draw] {End!} >
\end{tikzpicture}
```



In the previous example, `\tzhelplines*` is used to fix a bounding box. (See Section 6.2 on page 35, for more details.)

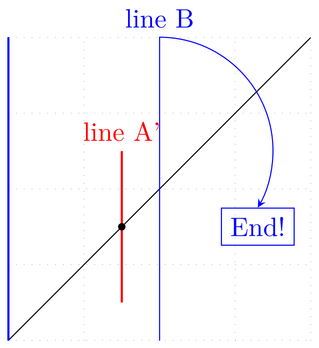
20.4.2 `\tzvfn`

`\tzvfn` accepts a coordinate as a mandatory argument and draws a horizontal line at the x value of the coordinate. For example, `\tzvfn(3,<y>)`, ignoring `<y>`, is equivalent to `\tzvfnat{3}`.

Everything else is the same as in `\tzvfnat`.

```
% syntax: minimal
\tzvfn(<coor>)[<domain>]
% syntax: full
\tzvfn[<opt>]<shift coor>"<path name>"
      (<coor>)[<domain>]{<text>}[<node opt>]<code.append>
% defaults
  []<>""(<m>)[south:north (of current bounding box)]{}[]<>
```

```
% \tzvfn: shift, <code.append>
\begin{tikzpicture}
\tzhelplines*(4,4) % bounding box
\tzcoors(1,0)(A)(2,0)(B);
\tzfn"XX"{\x}[0:4]
\tzvfn[blue,thick](0,5) % y=5 ignored
\tzvfn[red,thick]<.5,-.5>"AA"(A)[1:3]{line A'}[a]
\tzXpoint*{AA}{XX}
\tzvfn[blue,->](B){line B}[a]
        < arc (90:-30:1.5) node [b,draw] {End!} >
\end{tikzpicture}
```



21 Intersections

21.1 \tzXpoint(*): Intersection points

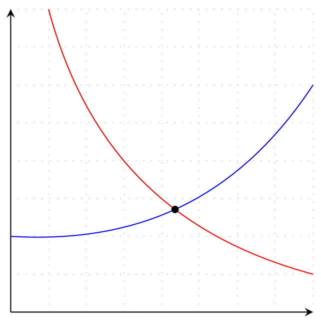
`\tzXpoint` finds intersection points of two paths and saves them as coordinate names for later use.

```
% syntax: minimal
\tzXpoint{<path>}{<path>}(<coor name>)
% syntax: medium
\tzXpoint{<path>}{<path>}(<coor name>){<label>}[<angle>]
% syntax: full
\tzXpoint[<opt>]{<path>}{<path>}(<coor name>)[<nth>]{<label>}[<[label opt]angle>]
% defaults
  []{<m>}{<m>}(intersection)[1]{}[]
```

For example, `\tzXpoint{path1}{path2}(A)` determines an intersection of the two paths and names the point `(A)` or `(A-1)`. (By default, the name is `(intersection)` as in Ti*k*Z.) If there are two or more intersection points, they are named as follows: `(A)=(A-1)`, `(A-2)`, `(A-3)`, etc.

You can determine which intersection point is named directly by specifying the option `[<nth>]`. If you select the second intersection point to be named `(A)` out of multiple intersections, they are named as follows: `(A-1)`, `(A)=(A-2)`, `(A-3)`, `(A-4)`, etc. You can label intersection points by specifying the option `{<label>}` and `[<angle>]`.

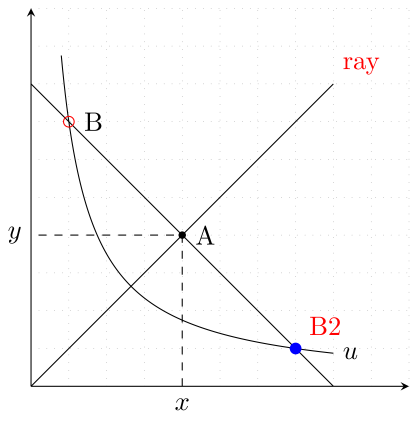
```
% \tzXpoint
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,8)
\tzaxes(8,8)
\tzto[red,bend right]"AA"(1,8)(8,1)
\tzto[blue,bend right]"BB"(0,2)(8,6)
\tzXpoint{AA}{BB}(A)
\tzdot*(A)
\end{tikzpicture}
```



`\tzXpoint*` The starred version `\tzXpoint*` simply adds a node dot to `\tzXpoint`. The default dot size is `2.4pt` and it can be changed by the last option `(<dot size>)` or the Three Ways (on page 41).

```
% syntax: minimum
\tzXpoint*{<path>}{<path>}
% syntax: medium
\tzXpoint*{<path>}{<path>}(<coor name>){<label>}[<angle>]
% syntax:
\tzXpoint*[<opt>]{<path>}{<path>}
          (<coor name>)[<nth>]{<label>}[<[label opt]angle>](<dot size>)
% defaults
  [tzdot=2.4pt]{<m>}{<m>}(intersection)[1]{}[](2.4pt)
```

```
% \tzXpoint*
\begin{tikzpicture}[scale=.5]
\tzhelplines(10,10)
\tzaxes(10,10)
\def\bgt{8-\x}
\def\Fx{\x}
\def\IC{7/\x}
\tzfn\bgt[0:8]                 % name path=bgt
\tzfn\Fx[0:8]{ray}[red,ar]     % name path=Fx
\tzfn"IC"\IC[.8:8]{$u$}[r]     % name path=IC
\tzXpoint*{bgt}{Fx}(A){A}[0]  % <angle>
\tzproj[dashed](A){$x$}{$y$}
\tzXpoint*[fill=none,red]{bgt}{IC}(B){B}[0](4pt)
\tzdot*[blue](B-2){B2}[[red]45](4pt)
\end{tikzpicture}
```



21.2 Vertical intersection points

21.2.1 \tzvXpointat(*)

\tzvXpointat determines vertical intersection points of a path at a specified value of x. So it takes {<path>} and {<x-val>} as mandatory arguments.

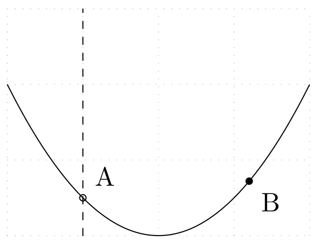
Remark: Internally, \tzvXpointat depends on the current bounding box, which generally does not cause a problem because it is used after paths to be intersected are formed. In case of any problem of no intersection point, you may want to fix a bounding box using \tzbbox or \tzaxes* or Ti*k*Z's \useasboundingbox.

```
% syntax: minimal
\tzvXpointat{<path>}{<x-val>}(<coor name>)
% syntax: medium
\tzvXpointat{<path>}{<x-val>}(<coor name>){<label>}[<angle>]
% syntax: full
\tzvXpointat[<opt>]{<path>}{<x-val>}(<coor name>)[<n>]
            {<label>}[<[label opt]angle>]
% defaults
  []{<m>}{<m>}(intersection)[1]{}[]
```

The starred version \tzvXpointat* additionally prints a node dot of the size 2.4pt, by default, at the (first) intersection point.

```
% syntax: minimum
\tzvXpointat*{<path>}{<x-val>}
% syntax: medium
\tzvXpointat*{<path>}{<x-val>}(<coor name>){<label>}[<angle>]
% syntax: full
\tzvXpointat*[<opt>]{<path>}{<x-val>}(<coor name>)[<n>]
             {<label>}[<[label opt]angle>](<dot size>)
% defaults
  [tzdot=2.4pt]{<m>}{<m>}(intersection)[1]{}[](2.4pt)
```

```
% \tzvXpointat(*)
\begin{tikzpicture}
\tzhelplines(4,3)
\def\Fx{.5*(\x-2)^2}
\tzfn\Fx[0:4] % name path=Fx
\tzvfnat[dashed]{1}
\tzvXpointat{Fx}{1}(A)
\tzdot(A){A}[45]
\tzvXpointat*{Fx}{3.2}{B}[-45]
\end{tikzpicture}
```



21.2.2 `\tzvXpoint(*)`

`\tzvXpoint` accepts `{<path>}` and `(<coor>)` as mandatory arguments to find vertical intersection points of a path at the x value of the coordinate, ignoring the y value. For example, `\tzvXpoint{mypath}(3,<y>)`, ignoring `<y>`, is equivalent to `\tzvXpointat{mypath}{3}`.

Everything else is the same as in `\tzvXpointat`.

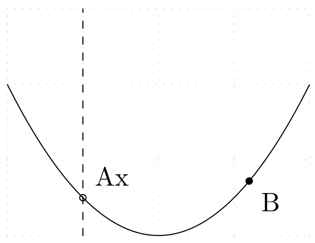
```
% syntax
\tzvXpoint[<opt>]{<path>}(<coor>)(<coor name>)[<n>]
          {<label>}[<[label opt]angle>]
% defaults
  []{<m>}(<m>)(intersection)[1]{}[]
```

The starred version `\tzvXpoint*` just adds a node dot to `\tzvXpoint`.

```
% syntax
\tzvXpoint*[<opt>]{<path>}(<coor>)(<coor name>)[<n>]
           {<label>}[<[<opt>]angle>](<dot size>)
% defaults
  [tzdot=2.4pt]{<m>}(<m>)(intersection)[1]{}[](2.4pt)
```

`\tzvXpoint*` prints, at the first intersection point, a node dot of the size 2.4pt by default.

```
% \tzvXpoint(*)
\begin{tikzpicture}
\tzhelplines(4,3)
\def\Fx{.5*(\x-2)^2}
\tzfn\Fx[0:4] % name path=Fx
\tzcoors(1,0)(A)(3.2,0)(B);
\tzvfn[dashed](1,0)
\tzvXpoint{Fx}(A)(Ax)
\tzdot(Ax){Ax}[45]
\tzvXpoint*{Fx}(B){B}[-45]
\end{tikzpicture}
```



21.3 Horizontal intersection points

21.3.1 `\tzhXpointat(*)`

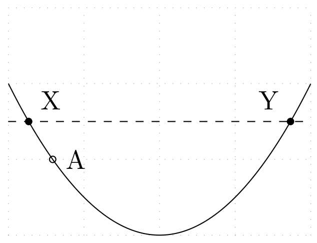
`\tzhXpointat` determines horizontal intersection points of a path at a specified value of y. So it takes `{<path>}` and `{<y-val>}` as mandatory arguments.

```
% syntax: minimal
\tzhXpointat{<path>}{<y-val>}(<coor name>)
% syntax: medium
\tzhXpointat{<path>}{<y-val>}(<coor name>){<label>}[<angle>]
% syntax: full
\tzhXpointat[<opt>]{<path>}{<y-val>}(<coor name>)[<n>]{<label>}[<[label opt]angle>]
% defaults
  []{<m>}{<m>}(intersection)[1]{}[]
```

The starred version `\tzhXpointat*` additionally prints a node dot of the size 2.4pt, by default, at the intersection point.

```
% syntax: minimum
\tzhXpointat*{<path>}{<y-val>}
% syntax: medium
\tzhXpointat*{<path>}{<y-val>}(<coor name>){<label>}[<angle>]
% syntax: full
\tzhXpointat*[<opt>]{<path>}{<y-val>}(<coor name>)[<n>]
             {<label>}[<[label opt]angle>](<dot size>)
% defaults
  [tzdot=2.4pt]{<m>}{<m>}(intersection)[1]{}[](2.4pt)
```

```
% \tzhXpointat(*)
\begin{tikzpicture}
\tzhelplines*(4,3)
\def\Fx{.5*(\x-2)^2}
\tzfn\Fx[0:4] % name path=Fx
\tzhXpointat{Fx}{1}(A)
\tzdot(A){A}[0]
\tzhfnat[dashed]{1.5}
\tzhXpointat*{Fx}{1.5}(X){X}[45]
\tzdot*(X-2){Y}[135]
\end{tikzpicture}
```



21.3.2 `\tzhXpoint(*)`

`\tzhXpoint` accepts `{<path>}` and `(<coor>)` as mandatory arguments to find horizontal intersection points of a path at the y value of the coordinate, ignoring the x value. For example, `\tzhXpoint{mypath}(<x>,3)`, ignoring `<x>`, is equivalent to `\tzhXpointat{mypath}{3}`.

Everything else is the same as in `\tzhXpointat`.

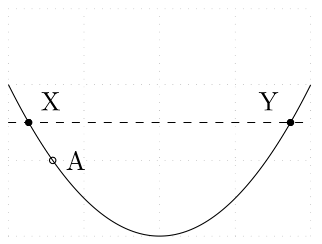
```
% syntax
\tzhXpoint[<opt>]{<path>}(<coor>)(<coor name>)[<n>]{<label>}[<[label opt]angle>]
% defaults
  []{<m>}(<m>)(intersection)[1]{}[]
```

The starred version `\tzhXpoint*` just adds a node dot to `\tzhXpoint`.

```
% syntax
\tzhXpoint*[<opt>]{<path>}(<coor>)(<coor name>)[<n>]
           {<label>}[<[label opt]angle>](<dot size>)
% defaults
  [tzdot=2.4pt]{<m>}(<m>)(intersection)[1]{}[](2.4pt)
```

\tzhXpoint* prints, at the (first) intersection point, a node dot of the size 2.4pt, by default.

```
% \tzhXpoint(*)
\begin{tikzpicture}
\tzhelplines*(4,3)
\def\Fx{.5*(\x-2)^2}
\tzfn\Fx[0:4] % name path=Fx
\tzcoors(0,1)(A)(0,1.5)(B);
\tzhXpoint{Fx}(A)(A)
\tzdot(A){A}[0]
\tzhfn[dashed](0,1.5)
\tzhXpoint*{Fx}(X){X}[45]
\tzdot*(X-2){Y}[135]
\end{tikzpicture}
```



22 Secant and Tangent Lines

22.1 Secant lines

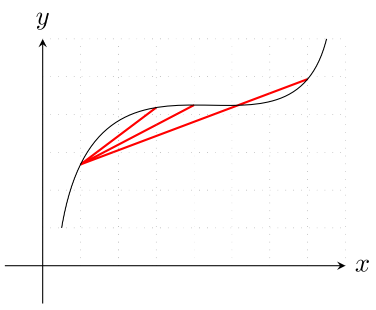
22.1.1 \tzsecantat

\tzsecantat draws a line segment or a secant line of a curve, on the behind layer by default. You need to specify a path name and two values of $x$. With \settzsecantlayer, you can change the layer, like \settzsecantlayer{main}.

```
% syntax: minium
\tzsecantat{<path>}{<from-x>}{<to-x>}
% syntax: medium
\tzsecantat{<path>}{<from-x>}{<to-x>}[<domain>]{<text>}[<node opt>]
% syntax: full
\tzsecantat[<opt>]<shift coor>"path name"
           {<path>}{<from-x>}{<to-x>}[<domain>]{<text>}[<node opt>]<code.append>
% defaults
  []<>""{<m>}{<m>}{<m>}[]{}[]<>
```

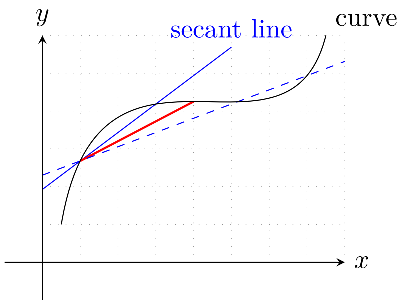
Domain The domain should be of the form [<from num:to num>]. Without specifying the optional domain, \tzsecantat draws a line segment connecting two points on the (curved) path.

```
% \tzsecantat
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier+"curve"(.5,1)(1,6)(-1,-4)(7,5)
\tzsecantat[thick,red]{curve}{1}{3}
\tzsecantat[thick,red]{curve}{1}{4}
\tzsecantat[thick,red]{curve}{1}{7}
\end{tikzpicture}
```



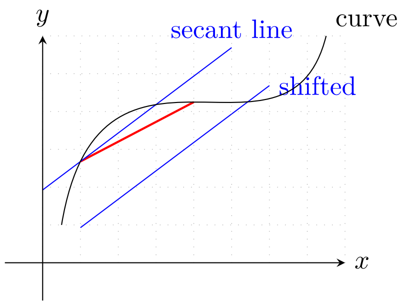
You can optionally specify the [<domain>].

```
% \tzsecantat: domain
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier+"curve"(.5,1)(1,6)(-1,-4)(7,5){curve}[ar]
\tzsecantat[blue]{curve}{1}{3}[0:5]{secant line}[a]
\tzsecantat[thick,red]{curve}{1}{4}
\tzsecantat[blue,dashed]{curve}{1}{7}[0:8]
\end{tikzpicture}
```



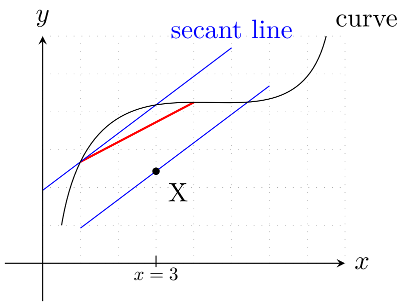
Shift You can move `\tzsecantat` by specifying the option `<shift coor>`.

```
% \tzsecantat: domain
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier+"curve"(.5,1)(1,6)(-1,-4)(7,5){curve}[ar]
\tzsecantat[thick,red]{curve}{1}{4}
\tzsecantat[blue]{curve}{1}{3}[0:5]{secant line}[a]
\tzsecantat[blue]<1,-1>{curve}{1}{3}[0:5]{shifted}[r]
\end{tikzpicture}
```



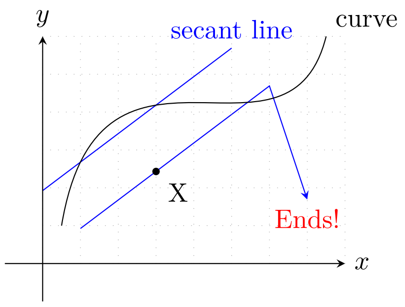
Naming paths By specifying the option `"<path name>"` you can name a path of `\tzsecantat`.

```
% \tzsecantat: shift, name path (intersection)
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier+"curve"(.5,1)(1,6)(-1,-4)(7,5){curve}[ar]
\tzsecantat[thick,red]{curve}{1}{4}
\tzsecantat[blue]{curve}{1}{3}[0:5]{secant line}[a]
\tzsecantat[blue]<1,-1>"shift"{curve}{1}{3}[0:5] %%
\tzvXpointat*{shift}{3}{X}[-45]
\tzticksx(-1mm:2mm){3/$x=3$}[scale=.7]
\end{tikzpicture}
```



`<code.append>` You can extend the path of `\tzsecantat` by writing Ti*k*Z code in the last optional argument `<code.append>`.

```
% \tzsecantat: <code.append>
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier+"curve"(.5,1)(1,6)(-1,-4)(7,5){curve}[ar]
\tzsecantat[blue]{curve}{1}{3}[0:5]{secant line}[a]
\tzsecantat[blue,->]<1,-1>"shift"{curve}{1}{3}[0:5] %%
                  < --++(1,-3) node [red,b] {Ends!} >
\tzvXpointat*{shift}{3}{X}[-45]
\end{tikzpicture}
```



22.1.2 `\tzsecant`

`\tzsecant` uses two *coordinates instead of two values of $x$* to draw a line segment or a secant line of a curve, on the `behind` layer by default. You need to specify a path name and two coordinates, then `\tzsecant` uses the $x$ values of the two coordinates.

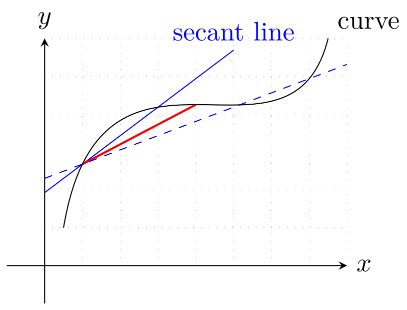
Everything else is the same as in `\tzsecantat`.

You can change the layer with `\settzsecantlayer`.

```
% syntax: minium
\tzsecant{<path>}(<coor>)(<coor>)
% syntax: medium
\tzsecant{<path>}(<coor>)(<coor>)[<domain>]{<text>}[<node opt>]
% syntax: full
\tzsecant[<opt>]<shift coor>"path name"
            {<path>}(<coor>)(<coor>)[<domain>]{<text>}[<node opt>]<code.append>
% defaults
  []<>""{<m>}(<m>)(<m>)[]{}[]<>
```

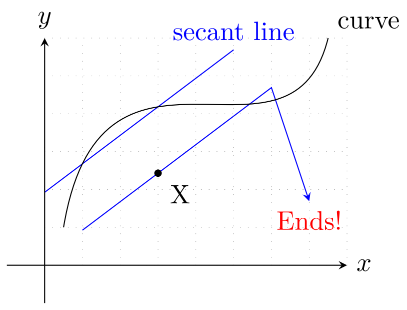
The domain should be of the form `[<from num:to num>]`. Without specifying the optional domain, `\tzsecant` draws a line segment connecting two points on the (curved) path.

```
% \tzsecant: domain
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier+"curve"(.5,1)(1,6)(-1,-4)(7,5){curve}[ar]
\tzcoor(1,0)(K)
\tzsecant[blue]{curve}(K)(3,0)[0:5]{secant line}[a]
\tzsecant[thick,red]{curve}(K)(4,0)
\tzsecant[blue,dashed]{curve}(K)(7,0)[0:8]
\end{tikzpicture}
```



You can move the secant line and extend the path.

```
% \tzsecant: shift, <code.append>
\begin{tikzpicture}[scale=.5]
\tzhelplines(8,6)
\tzaxes(-1,-1)(8,6){$x$}{$y$}
\tzbezier+"curve"(.5,1)(1,6)(-1,-4)(7,5){curve}[ar]
\tzcoor(1,0)(K)
\tzsecant[blue]{curve}(K)(3,0)[0:5]{secant line}[a]
\tzsecant[blue,->]<1,-1>"shift"{curve}(K)(3,0)[0:5] %%
                < --++(1,-3) node [red,b] {Ends!} >
\tzvXpointat*{shift}{3}{X}[-45]
\end{tikzpicture}
```



22.2 Tangent lines

22.2.1 `\tztangentat`

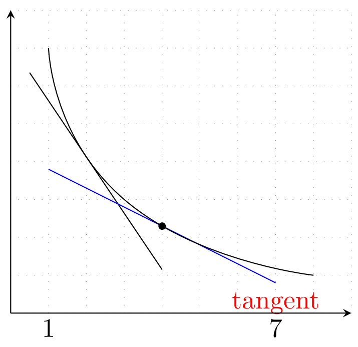
`\tztangentat` draws a tangent line to a curve at a specified value of $x$. By default, the tangent line is drawn on the `behind` layer, which can be changed by `\settztangentlayer`, like `\settztangentlayer{main}`.

Remark: To calculate the slope at $x$, $x$ varies over the interval $(x-\varepsilon_1, x+\varepsilon_2)$ and $\varepsilon_1 = \varepsilon_2 = 0.01$, by default. So the slope of tangent line is only *approximate*.

```
% syntax: minimum
\tztangentat{<path>}{<x-val>}[<domain>]
% syntax: medium
\tztangentat{<path>}{<x-val>}[<domain>]{<text>}[<node opt>]
% syntax: full
\tztangentat[<opt>]<shift coor>"<path name>"
            {<path>}{<x-val>}(<epsilon1>,<epsilon2>)[<domain>]
            {<text>}[<node opt>]<code.append>
% The domain should be of the form [<from:to>]
% defaults
  []<>""{<m>}{<m>}(.01,.01)[<m>]{}[]<>
```

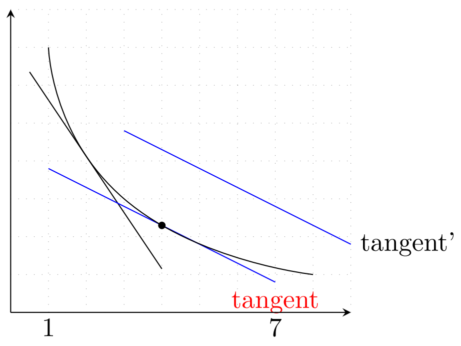
Domain `\tztangentat` takes three mandatory arguments: `{<path>}`, `{<x-val>}`, and `[<domain>]`. The mandatory argument `[<domain>]` should be of the form `[<from num:to num>]`.

```
% \tztangentat
\begin{tikzpicture}[scale=.5]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\tztangentat{AA}{2}[.5:4]
\tztangentat[blue]{AA}{4}[1:7]{tangent}[red,b]
\tzvXpointat*{AA}{4}
\tzticksx{1,7}
\end{tikzpicture}
```



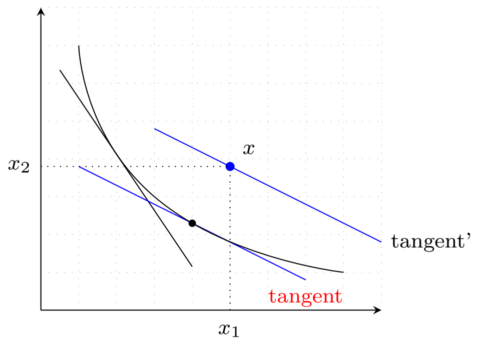
Shift You can move the tangent line by specifying the option `<shift coor>`.

```
% \tztangentat: shift
\begin{tikzpicture}[scale=.5]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\tztangentat{AA}{2}[.5:4]
\tztangentat[blue]{AA}{4}[1:7]{tangent}[red,b]
\tztangentat[blue]<2,1>{AA}{4}[1:7]{tangent'}[r]
\tzvXpointat*{AA}{4}
\tzticksx{1,7}
\end{tikzpicture}
```



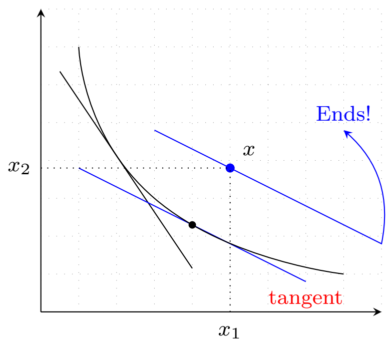
Naming paths By specifying the option `"<path name>"`, you can name the path of `\tztangentat`.

```
% \tztangentat: name path (intersection)
\begin{tikzpicture}[scale=.5,font=\footnotesize]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\tztangentat{AA}{2}[.5:4]
\tztangentat[blue]{AA}{4}[1:7]{tangent}[red,b]
\tzvXpointat*{AA}{4}
\tztangentat[blue]
            <2,1>"BB"{AA}{4}[1:7]{tangent'}[r]
\tzvXpointat*[blue]{BB}{5}(X){$x$}[45](3pt)
\tzproj(X){$x_1$}{$x_2$}
\end{tikzpicture}
```



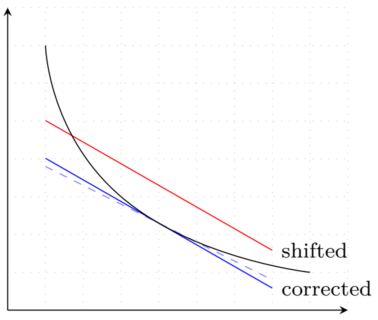
`<code.append>` You can extend the path of `\tztangentat` by writing Ti*k*Z code in the last optional argument `<code.append>`.

```
% \tztangentat: <code.append>
\begin{tikzpicture}[scale=.5,font=\footnotesize]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\tztangentat{AA}{2}[.5:4]
\tztangentat[blue]{AA}{4}[1:7]{tangent}[red,b]
\tzvXpointat*{AA}{4}
\tztangentat[blue,->]<2,1>"BB"{AA}{4}[1:7]
  < to [bend right] ++(-1,3) node [a] {Ends!} >
\tzvXpointat*[blue]{BB}{5}(X){$x$}[45](3pt)
\tzproj(X){$x_1$}{$x_2$}
\end{tikzpicture}
```



Variations The slope of the tangent line is approximate, so sometimes you may want to change the variation interval to get better results. You can change ε_1 and ε_2 by specifying the option `(<epsilon1,epsilon2>)` immediately after the mandatory argument `{<x-val>}`. Or you can change the variations by the macro `\settztangentepsilon`, like `\settztangentepsilon{`ε_1`}{`ε_2`}`. The effect remains until the end of `tikzpicture` environment unless changed again.

```
% \tztangentat: variations, shift
\begin{tikzpicture}[scale=.5,font=\footnotesize]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\tztangentat[blue!50,dashed]{AA}{4}[1:7]
\settztangentepsilon{0.005}{0.01}           %%
\tztangentat[blue]{AA}{4}[1:7]{corrected}[r]
\tztangentat[red]<0,1>{AA}{4}[1:7]{shifted}[r] %%
\end{tikzpicture}
```



22.2.2 `\tztangent`

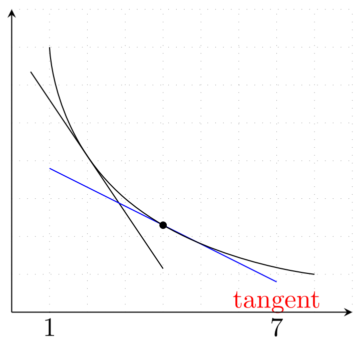
`\tztangent` uses a *coordinate instead of a value* of $x$ to draw a tangent line. For example, `\tztangent{curve}(4,<y>)` is equivalent to `\tztangentat{curve}{4}` for any `<y>`.

Everything else is the same as in `\tztangentat`.

```
% syntax: minimum
\tztangent{<path>}(<coor>)[<domain>]
% syntax: medium
\tztangent{<path>}(<coor>)[<domain>]{<text>}[<node opt>]
% syntax: full
\tztangent[<opt>]<shift coor>"<path name>"
          {<path>}(<coor>)(<epsilon1>,<epsilon2>)[<domain>]
          {<text>}[<node opt>]<code.append>
% The domain should be of the form [<from:to>]
% defaults
  []<>""{<m>}(<m>)(.01,.01)[<m>]{}[]<>
```

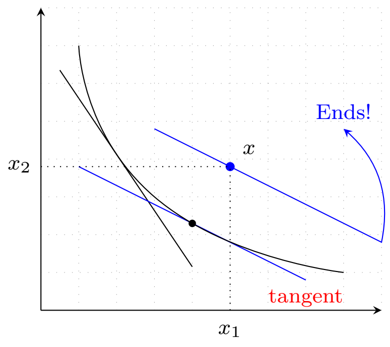
`\tztangent` accepts three mandatory arguments: `{<path>}`, `(<coor>)`, and `<domain>`.

```
% \tztangent
\begin{tikzpicture}[scale=.5]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\tzcoors(2,0)(K2)(4,0)(K4);
\tztangent{AA}(K2)[.5:4]
\tztangent[blue]{AA}(K4)[1:7]{tangent}[red,b]
\tzvXpoint*{AA}(K4)
\tzticksx{1,7}
\end{tikzpicture}
```



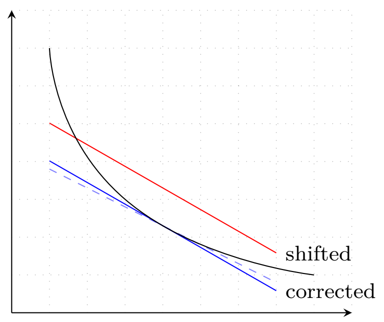
You can shift the tangent line and extend its path.

```
% \tztangent: shift, <code.append>
\begin{tikzpicture}[scale=.5,font=\footnotesize]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\tzcoors(2,0)(K2)(4,0)(K4);
\tztangent{AA}(K2)[.5:4]
\tztangent[blue]{AA}(K4)[1:7]{tangent}[red,b]
\tzvXpoint*{AA}(K4)
\tztangent[blue,->]<2,1>"BB"{AA}(K4)[1:7]
  < to [bend right] ++(-1,3) node [a] {Ends!} >
\tzvXpoint*[blue]{BB}(5,0)(X){$x$}[45](3pt)
\tzproj(X){$x_1$}{$x_2$}
\end{tikzpicture}
```



You can control the interval of variations of $x$ by the option `(<epsilon1,epsion2>)` or the marcro `\settztangentepsilon`.

```
% \tztangent: variations, shift
\begin{tikzpicture}[scale=.5,font=\footnotesize]
\tzhelplines(9,8)
\tzaxes(9,8)
\tzplotcurve"AA"(1,7)(3,3)(8,1);
\tztangent[blue!50,dashed]{AA}(4,0)[1:7]
\settztangentepsilon{0.005}{0.01}                %%
\tztangent[blue]{AA}(4,0)[1:7]{corrected}[r]
\tztangent[red]<0,1>{AA}(4,0)[1:7]{shifted}[r]  %%
\end{tikzpicture}
```



23 Miscellany

23.1 \tzbrace(')

\tzbrace takes two coordinates as mandatory arguments to draw a *calligraphic brace* connecting them.

```
% syntax: minimum
\tzbrace(<coor>)(<coor>)
% syntax: medium
\tzbrace(<coor>)(<coor>){<text>}[<node opt>]
% syntax: full
\tzbrace[<draw opt>]{<raise>}[<decoration opt>]<shift coor>
        (<coor>)(<coor>){<text>}[<node opt>]
% defaults
  []{5pt}[amplitude=5pt]<>(<m>)(<m>){}[]
```

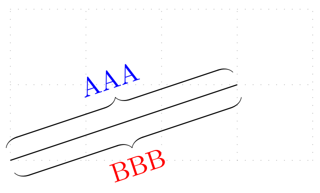
The raise value of a brace is 5pt by default and the value can be changed by the first curly brace optional argument {<raise>}.

The amplitude of a brace is 5pt by default. You can control the amplitude by writing the option amplitude=<dim> in the second bracket option [<decoration opt>].

```
\tzbrace[thick](0,0)(3,1) % works like:
  \draw [thick,decorate,decoration={calligraphic brace, amplitude=5pt, raise=5pt}]
        (0,0) to (3,1);
```

The *wap version* \tzbrace' swaps the coordinates. So it prints a mirror image of \tzbrace. For example, \tzbrace'(0,0)(3,1) is equivalent to \tzbrace(3,1)(0,0).

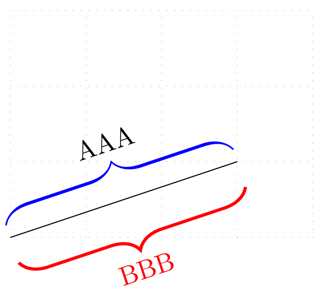
```
% \tzbrace(')
\begin{tikzpicture}[sloped]
\tzhelplines(4,2)
\tzline(0,0)(3,1)
\tzbrace(0,0)(3,1){AAA}[above=10pt,blue]
\tzbrace'[red](0,0)(3,1){BBB}[below=10pt]
\end{tikzpicture}
```



You can change the style of the decorating brace by the second bracket optional argument [<decoration opt>].

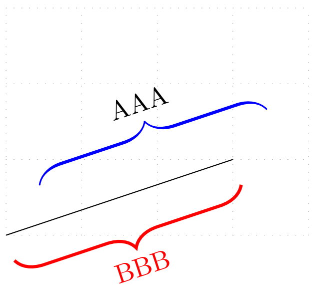
The color of the calligraphic brace can be changed by the option pen colour in the list of [<draw option>].

```
% \tzbrace('): decoration options
\begin{tikzpicture}[sloped]
\tzhelplines(4,3)
\tzline(0,0)(3,1)
\tzbrace [very thick,pen colour=blue]
        [amplitude=10pt]
        (0,0)(3,1){AAA}[a=15pt]
\tzbrace'[red,very thick]{10pt}
        [brace,amplitude=10pt]
        (0,0)(3,1){BBB}[b=20pt]
\end{tikzpicture}
```



You can also move a brace by specifying the option <shift coor> immediately before the the first mandatory coordinate. The empty shift option <> is not allowed.

```
% \tzbrace('): shift
\begin{tikzpicture}[sloped]
\tzhelplines(4,3)
\tzline(0,0)(3,1)
\tzbrace [very thick,pen colour=blue][amplitude=10pt]
         <.5,.5>(0,0)(3,1){AAA}[a=15pt]
\tzbrace'[red,very thick]{10pt}[brace,amplitude=10pt]
         (0,0)(3,1){BBB}[b=20pt]
\end{tikzpicture}
```



Version history

- v1.0.1(2021/03/20)
  - revised the document with typo corrections
  - added the option `<code.append>` to `\tzframe`, `\tzcircle`, and `\tzellipse`
  - added aliases: `\let\tzrectangle\tzframe` and `\let\tzoval\tzellipse`
- v1.0 (2021/02/28) uploaded to CTAN
- v0.999a (2021/02/27)
  - writing document
  - some `\tz<...>AtBegin` and `\tz<...>AtEnd` not documented
- v0.999 (2021/02/24)
  - fixed the title. "Plot Graphs with Ti*k*Z Abbreviations"
  - changed the default `\tzpathstyle` and `\tzlinkstyle` from '`--`' to '`to`' (no harm)
  - added `\tzpathlayer` and `\settzpathlayer`, for later use. (currently not used)
  - removed explanation of the option `<+or++>` in the document (considering to remove later)
  - changed the delimiter `<path style>` to `[<path style>]` in `\tzpath` and `\tzlink(s)` (critical change)
  - added `text opactiy=1` everywhere `fill opacity=.3` is used

Acknowledgement

I am grateful to Kangsoo Kim of KTUG (Korean TeX Users Group) who has written many packages including `oblivoir.cls` for helping to implement the idea of the semicolon version with `expl3`.

References

Casteleyn, Jean Pierre (2016), "Visual PSTricks," version 2.30.
____________________ (2018), "Visual TikZ," version 0.66.
Tantau, Till (2020), "Ti*k*Z and PGF: Manual for version 3.1.8b," `http://sourceforge.net/projects/pgf`.

Index