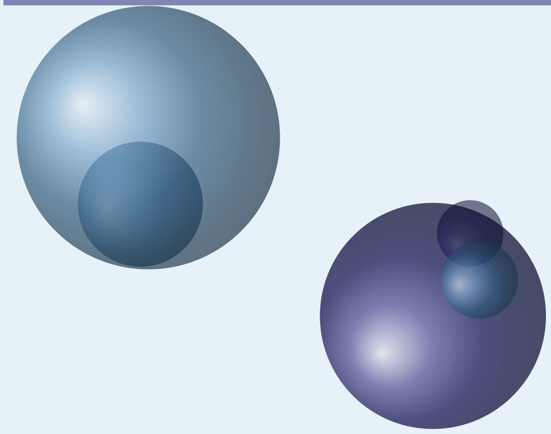
tkz-base 4.00 b

AlterMundus

tkz-base



Alain Matthes

January 4, 2022 Documentation V.4.00 b

http://altermundus.fr

tkz-base

AlterMundus

Alain Matthes

`tkz-base` is a package based on Ti*k*Z to make graphics as simple as possible. It is the basis on which a series of packages will be built, having as a common point, the creation of drawings useful in the teaching of mathematics. The main function of `tkz-base` is to provide an orthogonal coordinate system, and to let the user choose the graphical units. This package requires version 3 or higher of Ti*k*Z.

I'd like to thank **Till Tantau** for creating the wonderful tool Ti*k*Z.

I thank **Yves Combe** for sharing his work on the protractor and the compass constructions. I would also like to thank, **David Arnold** who corrected a lot of errors and tested many examples, **Wolfgang Büchel** who also corrected errors and built great scripts to get the example files, **John Kitzmiller** and **Dimitri Kapetas** for their examples, **Gaétan Marris** for his remarks and corrections, and finally **Laurent Van Deik** for all his corrections, remarks and questions.

You will find many examples on my site: altermundus.fr.

You can send your remarks, and reports on errors you find, to the following address: Alain Matthes.

Contents

1 News and presentation

This package was the foundation of the tkz-euclide and tkz-fct in particular. Now tkz-euclide is independent of tkz-base. **tkz-euclide** should be used only for Euclidean geometry. The package has been modified and object transfers between **tkz-base** and **tkz-euclide** have been performed.

tkz-base provides a Cartesian system that will be defined by the macro `\tkzInit`. The big difference now between **tkznameofpack** and tkz-euclide is the role of the units. The unit in tkz-euclide is the cm and is fixed. Ce n'est pas le cas **tkz-base**

The main novelty is the recent replacement of the fp package by xfp. The appearance of this one is a step towards version 3 of LaTeX. The next step will be the creation of a new package.

Here are some of the changes. The **tkz-euclide** package brings more new features. **tkz-euclide** is used for some examples in this documentation.

- Code Improvement;
- Bug correction;
- The bounding box is now controlled in each macro (hopefully) to avoid the use of `\tkzInit` followed by `\tkzClip`;
- Logically most macros accept Ti*k*Z options. So I removed the "duplicate" options;
- Removing the option "label options";
- Random points are now in **tkz-euclide** and the macro `\tkzGetRandPointOn` is replaced by `\tkzDefRandPointOn`. For homogeneity reasons, the points must be retrieved with `\tkzGetPoint`;
- The options **end** and **start** which allowed to give a label to a line are removed. You must now use the macro `\tkzLabelLine`;
- Introduction of the libraries quotes and angles they allows to give a label to a point.even if I am not in favour of this practice;
- Appearance of the macro `\usetkztool`, which allows to load new "tools".

2 Installation

tkz-base is now on the server of the CTAN[1]. If you want to test a beta version, just put the following files in a texmf folder that your system can find. You will have to check several points:

– The tkz-base folder must be located on a path recognized by latex.

– The tkz-base uses xfp.

– This documentation and all examples were obtained with lualatex but pdflatex or xelatex should be suitable.

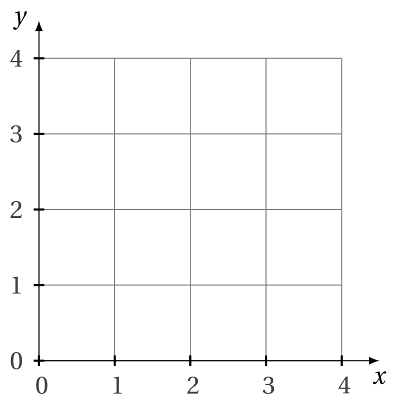
1 tkz-base is part of TeXLive and tlmgr allows you to install them. This package is also part of MiKTeX under Windows.

3 Compilation of examples

3.1 Installation test

The code below allows you to test your installation of **tkz-base**. Please note that xfp as well as numprint must be present as well as version 3.01 (or higher) of pgf. All examples and this documentation have been compiled using LuaLaTeX.

```
\documentclass{standalone}
\usepackage{tkz-base}
\begin{document}
\begin{tikzpicture}
 \tkzInit[xmax=4,ymax=4]
 \tkzGrid
 \tkzAxeXY
\end{tikzpicture}
\end{document}
```



Notes on this test

1. The compilation of this document and examples is obtained with **luaLaTeX**.
2. tkz-base loads numprint with the option autolanguage, xfp and of course Ti*k*Z.
3. Ti*k*Z seems that version 3 of pgf has fixed those problems. In case of difficulty, it is recommended to load the babel library with `\usetikzlabry{babel}`. Another possibility is to compile with LuaLaTeX.

3.2 xfp and numprint

xfp now replaces fp in this package. One of the advantages for the user is a simplified syntax. It allows to manage calculations on large or very small numbers with precision. This slows down the compilation a bit, so it is better not to overuse it. xfp is used above all, to obtain correct graduations. numprint was present when I started to write this series of packages, since siunitx has grown and I can understand that some people prefer it. In a future version, I plan to leave the choice of the package for displaying numbers.

4 Presentation of tkz-base

4.1 Example that poses a problem

The following code gives an error

```
\begin{tikzpicture}
  \draw (0,0)--(600,0);
\end{tikzpicture}
```

Latex Error: ... Dimension too large.

Indeed, the default unit is a centimeter but TEX cannot store a dimension greater than 575 cm, which leads to an error. TEX however, can store integers up to $2^{31} - 1$, so it is possible to work on integers first and then define the dimensions.

```
\begin{tikzpicture}[x=0.01 cm]
  \draw (0,0)--(600 cm,0);
\end{tikzpicture}
```

Latex Error: ... Dimension too large.

The previous code still makes an error. Indeed, 600 cm is a dimension and does not take into account the change of unit. The correct version is:

```
\begin{tikzpicture}[x=0.01 cm]
  \draw (0,0)--(600,0);
\end{tikzpicture}
```

This time, the stored dimension is 6 cm which is acceptable. It is possible with TEX to handle large whole numbers, but, on the other hand, the dimensions cannot exceed 16,384 pt or approximately 5.75 m.

With TEX, it's also possible to work with the **xfp** package. This allows him to work at longer intervals, but at the cost of a certain slowness. This is the method I have preferred for some sensitive calculations that require good precision, such as calculations to measure angles or segment length, but it is necessary once a number has been found to assign it to a dimension. We always find the same constraints.

4.2 The role of tkz-base

The following code gives an error not because 6,000,000 is too large, but because 0.000,001 cm is too small.

Latex Error:

```
\begin{tikzpicture}[x=0.000001 cm]
 \coordinate (x) at (6000000,0);
  \draw (0,0)--(x);
\end{tikzpicture}
```

With **tkz-base**, it will be possible to work with any coordinates, but it will be necessary to use the macros of the package.

tkz-base simplifies the use of different value ranges. This package is used by tkz-fct which allows to draw graphical representations of functions using **gnuplot**.

First of all, you should know that it is not necessary to deal with Ti*k*Z with the size of the support (bounding box); however it is sometimes necessary, either to draw a grid, or to draw axes, or to work with a different unit than the centimeter, or finally to control the size of what will be displayed. To do this, you must have prepared the frame in which you are going to work, this is the role of tkz-base and its main macro **\tkzInit**. For example, if you want to work on a 10 cm square, but such that the unit is the dm then you will have to use.

```
\tkzInit[xmax=1,ymax=1,xstep=0.1,ystep=0.1]
```

`xstep=0.1` means that 1cm represents the 0.1 graduation so the 1 graduation is at 10 cm from the origin.

On the other hand, for values of x between 0 and 10,000 and values of y between 0 and 100,000, it will be necessary to write

```
\tkzInit[xmax=10000,ymax=100000,xstep=1000,ystep=10000]
```

The result is always a 10 cm square.

All this makes little sense for Euclidean geometry, and in this case it is recommended to leave the graphic unit equal to 1 cm. I have not tested whether all macros for Euclidean geometry accept other values than `xstep=1` and `ystep=1`. On the other hand, for some drawings, it is interesting to fix the extreme values and to ”clip” the definition rectangle in order to control the size of the figure as well as possible.

4.3 Syntax of tkz-base

I tried to generalize the following syntax:

- The syntax is close to that of LaTeX, there's no need for ”;” with **tkz-base**.
- all the macros have names beginning with **tkz**;
- braces are used to pass a parameter that will be the reference of an object created by the macro;
- parentheses are used to refer to an object that has already been created or to a coordinate pair;
- square brackets are necessary to pass optional arguments or options, some choices are sometimes mandatory. The use of the comma even in a Math mode requires to be protected in a TeX group;
- blanks (space) are prohibited between [...] and (...), [...] and {...}, as well as between (...) and {...}, but it is possible to put spaces between passed in optional arguments [...].

5 Initialization \tkzInit

5.1 The main macro \tkzInit

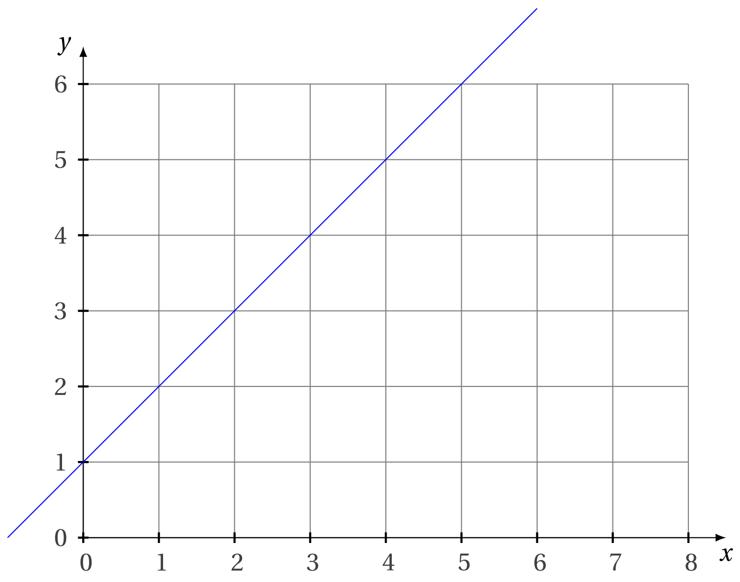
`\tkzInit[⟨local options⟩]`

options	default	definition
xmin	0	minimum value of the abscissae in cm
xmax	10	maximum value of the abscissae in cm
xstep	1	difference between two graduations in x
ymin	0	minimum y-axis value in cm
ymax	10	maximum y-axis value in cm
ystep	1	difference between two graduations in y

The role of **tkzInit** is to define a orthogonal coordinates system and a rectangular part of the plane in which you will place your drawings using Cartesian coordinates. The coordinates system does not have to be normalized. This macro allows you to define your working environment as with a calculator.

5.1.1 Changing the drawing size with \tkzInit

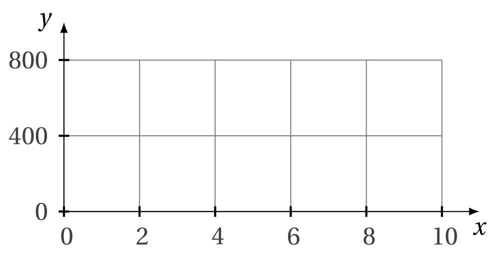
This macro sets the stage and defines several constants. It is quite possible to make a figure larger than the predefined rectangle. Moreover, as you can see, it is possible to use the commands of Ti*k*Z in the middle of those of `tkz` but attention to the units! This possibility must be reserved for exceptional cases only.



```
\begin{tikzpicture}
   \tkzInit[xmax=8,ymax=6]
   \tkzGrid
   \tkzAxeXY
   \draw[blue](-1,0)--(6,7);
\end{tikzpicture}
```

5.1.2 Role of xstep , ystep

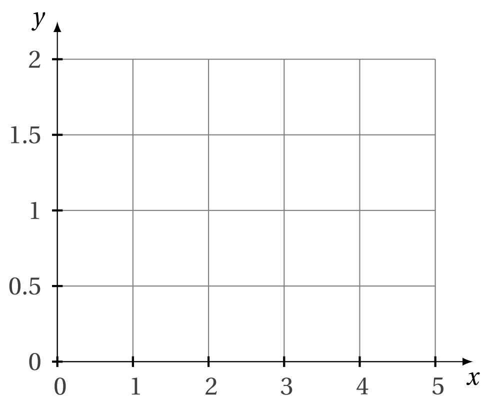
Warning, a graduation is represented by 1 cm, unless you resize the figure with the **`scale`** option. In the example below **`xstep`** = 2 corresponds to 1 cm, so between 0 and 10, we will need 5 cm. Similarly **`ystep`**=400, so between 0 and 800 there are 2 cm. It is not possible to use the options of Ti*k*Z, **`x=...`** and **`y=...`**.



```
\begin{tikzpicture}
  \tkzInit[xmax=10,xstep=2,ymax=800,ystep=400]
  \tkzGrid
  \tkzAxeXY

\end{tikzpicture}
```

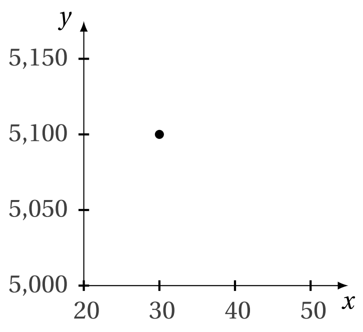
5.2 Another example with xstep and ystep



```
\begin{tikzpicture}
  \tkzInit[xmax=5,xstep=1,ymax=2,ystep=.5]
    \tkzGrid
    \tkzAxeXY
\end{tikzpicture}
```

5.2.1 Customized origin.

It is important to note that you can place a point without calculating anything.



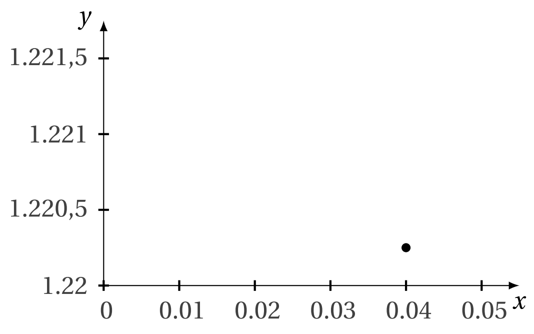
```
\begin{tikzpicture}
  \tkzInit[xmin=20,
           xmax=50,
           xstep=10,
           ymin=5000,
           ymax=5150,
           ystep=50]
  \tkzAxeXY
  \tkzDefPoint(30,5100){A}
  \tkzDrawPoint(A)
\end{tikzpicture}
```

5.2.2 Use of decimals

It is preferable to write the different arguments relating to an axis with the same number of decimals. **numprint** is used to display the graduations correctly.

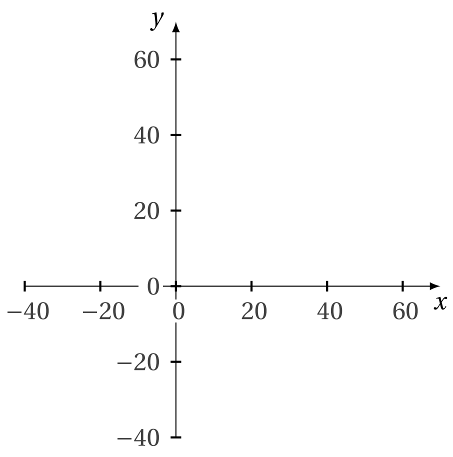
In the following example, **numprint** uses the English conventions for writing numbers because I used:

`\usepackage[english]{babel}`



```
\begin{tikzpicture}
  \tkzInit[xmin=0.00,  xmax=0.05,
           ymin=1.2200,ymax=1.2215,
           xstep=0.01, ystep=0.0005]
  \tkzAxeXY
  \tkzDefPoint(.04,1.22025){I}
  \tkzDrawPoint(I)
\end{tikzpicture}
```

5.2.3 Negative values



```
\begin{tikzpicture}
  \tkzInit[xmin  = -40,
           xmax  =  60,
           ymin  = -40,
           ymax  =  60,
           xstep =  20,
           ystep =  20]
  \tkzAxeXY
\end{tikzpicture}
```

6 Macros for the axes

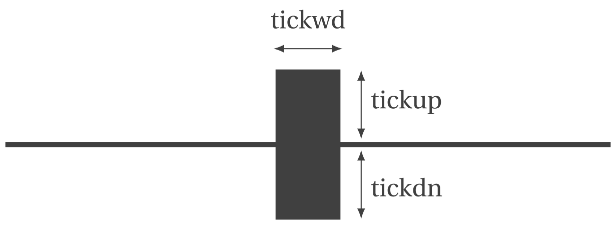
Careful, these macros have been modified. It's now easier to use the styles of Ti*k*Z. **`\tkzDrawX`** allows to draw an axis, **`\tkzLabelX`** places graduations and finally in simple cases **`\tkzAxeX`** traces and graduations. The options of Ti*k*Z are accessible. Fractions can be used for graduations.

6.1 \tkzDrawX

`\tkzDrawX[⟨local options⟩]`

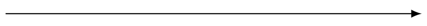
This macro allows you to draw the abscissa axis with default ticks. The options are those of Ti*k*Z plus the following ones:

options	default	definition
color	black	Axis and ticks
noticks	false	no ticks on axis
right space	0.5 cm	axis extended right
left space	0 cm	extension of the axis to the left
label	x	label name
trig	0	if <>0 graduations are multiples of pi/trig" "trig is an integer"
tickwd	0.8pt	tick thickness
tickup	1pt	tick over axis
tickdn	1pt	tick depth over axis



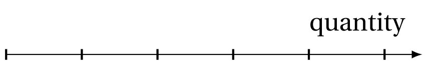
This macro is used to draw the abscissa axis. The most important thing is to test all the options. Above, you have the values that define a tick. Otherwise the options of Ti*k*Z apply and in particular `text`, `color`, `fill` and `font`.

6.1.1 No tick, no label



```
\begin{tikzpicture}
 \tkzInit[xmax=5]
 \tkzDrawX[label={},noticks]
\end{tikzpicture}
```

6.1.2 Label placement

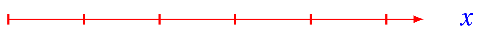


```
\begin{tikzpicture}
 \tkzInit[xmax=5]
 \tkzDrawX[label      = quantity,
          above left = 8pt]
\end{tikzpicture}
```

6.1.3 Label and Axis Colour

The color of the label is obtained with the option `text`, that of the axis with the option `color`.

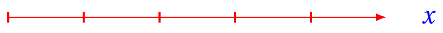
The option `right=12pt` shifts the label x by 12 pt.



```
\begin{tikzpicture}
  \tkzInit[xmax=5]
  \tkzDrawX[text=blue,color=red,right=12pt]
\end{tikzpicture}
```

6.1.4 Option right space

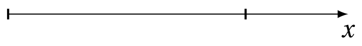
It adds a little space after the last tick.



```
\begin{tikzpicture}
\tkzInit[xmax=0.4,xstep=0.1]
\tkzDrawX[text=blue,color=red,right=12pt,right space=1]
\end{tikzpicture}
```

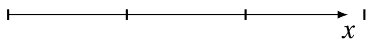
6.1.5 Trigonometric axis with the option trig=n

If $number = 0$ then the axis is graduated from cm to cm, otherwise the axis is graduated using multiples of $\frac{\pi}{number}$.



```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=4,ymin=-1,ymax=1]
  \tkzDrawX[trig=1]
\end{tikzpicture}
```

6.1.6 Trigonometric axis with the option trig=2



```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=4,ymin=-1,ymax=1]
  \tkzDrawX[trig=2]
\end{tikzpicture}
```

6.2 \tkzLabelX

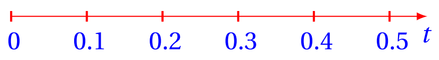
`\tkzLabelX[⟨local options⟩]`

This macro allows you to place graduations. The option `orig` can be used again, but its behavior is reversed. By default, the original value is placed. The options are those of TikZ, plus the following ones:

options	default	definition
frac	0	if <>0 graduations are multiples num/frac "frac is an integer"
trig	0	if <>0 graduations are multiples *pi*/trig "trig is an integer"
font	\textstyle	scale size.
color	black	graduation color
step	1	interval between graduations
np off	false	numprint deactivation
orig	true	displays the origin graduation

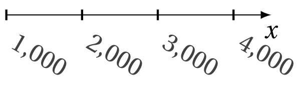
`frac` and `trig` are integers that can be changed to fractional or trigonometric writing.

6.2.1 Position of the graduations



```
\begin{tikzpicture}
\tkzInit[xmax=.5,xstep=0.1]
\tkzDrawX[label=$t$,text=blue,color=red]
\tkzLabelX[text=blue,below = 3pt]
\end{tikzpicture}
```

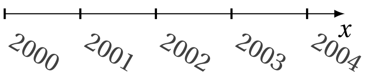
6.2.2 Position of the graduations with xlabel style



```
\begin{tikzpicture}
 \tkzInit[xmin=1000,xmax=4000,xstep=1000]
 \tkzDrawX
 \tikzset{xlabel style/.append style={rotate=-30}}
 \tkzLabelX[below right=3 pt,inner sep = 1pt]
\end{tikzpicture}
```

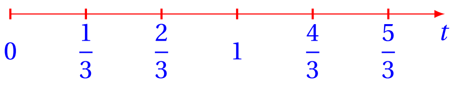
6.2.3 Dates with np off

For dates, you have to deactivate numprint.



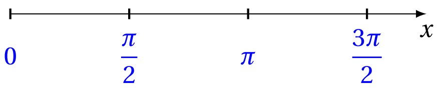
```
\begin{tikzpicture}
 \tkzInit[xmin=2000,xmax=2004]
 \tkzDrawX
 \tikzset{xlabel style/.append style={rotate=-30}}
 \tkzLabelX[np off,below right=3 pt,inner sep =1pt]
\end{tikzpicture}
```

6.2.4 frac



```
\begin{tikzpicture}
\tkzInit[xmax=1.75,xstep=0.33333]
\tkzDrawX[label=$t$,text=blue,color=red]
\tkzLabelX[frac=3,text=blue,below = 6pt]
\end{tikzpicture}
```

6.2.5 trig

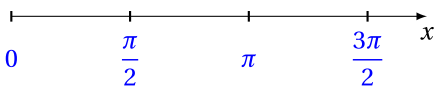


```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=5,ymin=-1,ymax=1]
  \tkzDrawX[trig=2]
  \tkzLabelX[trig=2,text=blue,below = 8pt]
\end{tikzpicture}
```

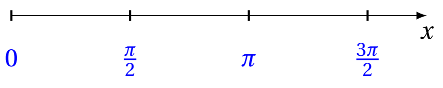
6.2.6 Graduations size

Two possibilities. It is possible to define the default style used for the math mode:

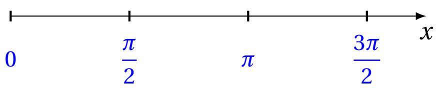
```
\let\tkzmathstyle\textstyle
```



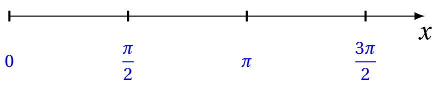
```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=5,ymin=-1,ymax=1]
  \tkzDrawX[trig=2]
  \tkzLabelX[trig=2,text=blue,below = 8pt]
\end{tikzpicture}
```



```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=5,ymin=-1,ymax=1]
  \let\tkzmathstyle\textstyle
  \tkzDrawX[trig=2]
  \tkzLabelX[trig=2,text=blue,
            below = 8pt]
\end{tikzpicture}
```



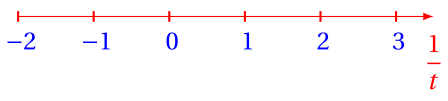
```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=5,ymin=-1,ymax=1]
  \tkzDrawX[trig=2]
  \tkzLabelX[trig=2,text=blue,
           below = 8pt,node font=\small]
\end{tikzpicture}
```



```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=5,ymin=-1,ymax=1]
  \tkzDrawX[trig=2]
  \tkzLabelX[trig=2,text=blue,
      below = 8pt,node font=\scriptsize]
\end{tikzpicture}
```

6.2.7 Colour of the graduations

The key here is to use the color, text, and text options correctly.

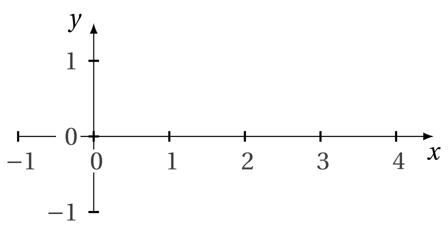


```
\begin{tikzpicture}
  \tkzInit[xmin = -2,xmax = 3,
           ymin = -2,ymax = 2]
 \tkzDrawX[color = red,
           label = $\displaystyle\frac{1}{t}$,
           below = 6pt]
 \tkzLabelX[text=blue]
\end{tikzpicture}
```

6.2.8 Axis drawings before the graduation

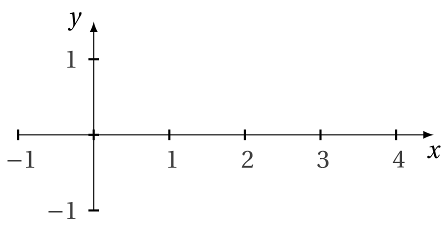
In some cases, it is preferable to place `\tkzDrawXY` after `\tkzLabelX` and `\tkzLabelY`.

This prevents display problems.



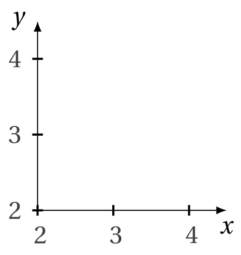
```
\begin{tikzpicture}
\tkzInit[xmin = -1,xmax = 4,
         ymin = -1,ymax = 1]
\tkzDrawXY \tkzLabelX  \tkzLabelY
\end{tikzpicture}
```

6.2.9 Graduations (except originally) prior to tracings



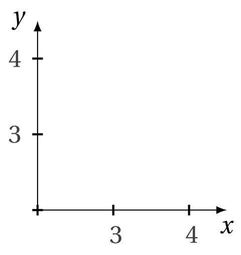
```
\begin{tikzpicture}
   \tkzInit[xmin = -1,xmax = 4,
            ymin = -1,ymax = 1]
   \tkzLabelX[orig=false]
   \tkzLabelY[orig=false]
   \tkzDrawXY
\end{tikzpicture}
```

6.2.10 Only positive graduations before drawings



```
\begin{tikzpicture}
  \tkzInit[xmin=2,ymin=2,xmax=4,ymax=4]
  \tkzLabelX \tkzLabelY
  \tkzDrawXY
\end{tikzpicture}
```

6.2.11 No graduations at the origin



```
\begin{tikzpicture}
  \tkzInit[xmin=2,ymin=2,xmax=4,ymax=4]
  \tkzLabelX[orig]    \tkzLabelY[orig]
  \tkzDrawXY
\end{tikzpicture}
```

6.3 \tkzAxeX

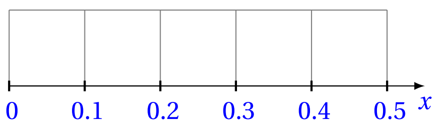
\tkzAxeX[⟨local options⟩]

This macro allows you to draw the abscissa axis with default ticks as well as the graduations. It combines the two macros **\tkzDrawX** and **\tkzLabelX**. It should only be used in simple cases.

options	default	definition
label	x	label name
trig	0	if <>0, graduations are multiples of pi/trig
frac	0	if <>0, graduations are multiples of 1/frac
swap	false	allows you to run **\tkzLabelX** before **\tkzDrawX**

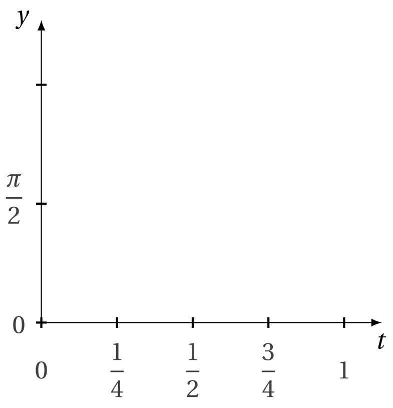
The option `text` defines the color of the graduations.

6.3.1 Example with \tkzAxeX



```
\begin{tikzpicture}
   \tkzInit[xmax=0.5,xstep=0.1,ymax=1]
   \tkzGrid
   \tkzAxeX[text=blue]
\end{tikzpicture}
```

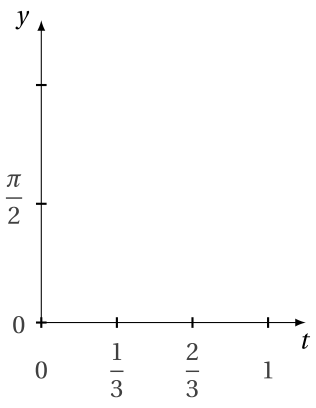
6.3.2 Use of pi and \tkzAxeX



```
\begin{tikzpicture}
  \tkzInit[xmax=4,ymax=3.5]
  \let\tkzmathstyle\displaystyle
  \tkzLabelX[orig  = false, frac  = 4,below = 10pt]
  \tkzDrawX[label = $t$]
  \tkzAxeY[trig=2]
\end{tikzpicture}
```

6.3.3 Option frac and trig

In this example, we position the t label as well as the graduations. `\below=10pt` is used to place the graduations underneath.



```
\begin{tikzpicture}
  \tkzInit[xmax=9,xstep=3,ymax=3.5]
  \tkzLabelX[below=10pt,orig=false,frac=3]
  \tkzDrawX[label = $t$]
  \tkzAxeY[trig=2]
\end{tikzpicture}
```

6.4 \tkzDrawY

```
\tkzDrawY[⟨local options⟩]
```

This macro allows you to draw the ordinate axis with default ticks. The options are those of TikZ plus the following ones:

options	default	definition
color	black	color of axis and ticks
noticks	false	no ticks on the axis
up space	0.5 cm	top axis extension
down space	0 cm	axis extension down
label	x	label name
trig	0	if <>0, graduations are multiples of pi/trig "trig is an integer"
tickwd	0.8pt	tick's thickness
ticklt	1pt	height of the tick above the axis
tickrt	1pt	above-axis tick depth

6.5 \tkzLabelY

```
\tkzLabelY[⟨local options⟩]
```

This macro allows you to draw the abscissa axis with default ticks. The options are those of Ti*k*Z plus the following ones:

options	default	definition
color	black	graduation color
frac	0	if <>0, graduations are multiples of 1/frac "frac is an integer"
font	\textstyle	graduation size.
step	1	interval between graduations

`frac` is a integer that can be changed to fractional or trigonometric writing.

6.6 \tkzAxeY

```
\tkzAxeY[⟨local options⟩]
```

This macro combines the two macros: `\tkzDrawY` `\tkzLabelY` See `\tkzAxeX` for options.

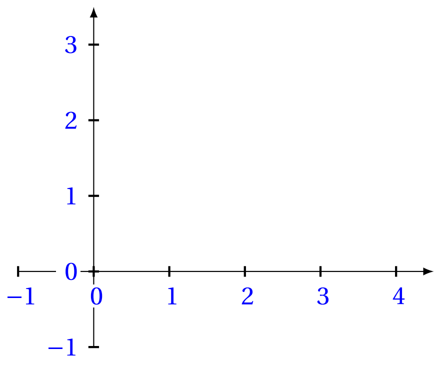
6.7 \tkzAxeXY

```
\tkzAxeXY[⟨local options⟩]
```

This macro combines the four macros: `\tkzDrawX\tkzDrawY` `\tkzLabelX\tkzLabelY`

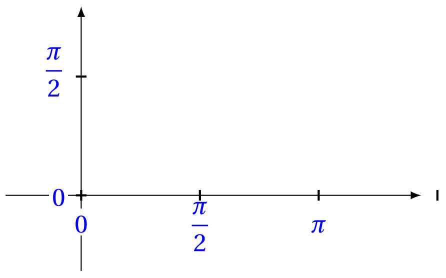
It is necessary to use common options as in the example below, but this means that the same options are applied to both macros. Thus it is not possible to change `label`.

6.7.1 Colour of axes, graduations



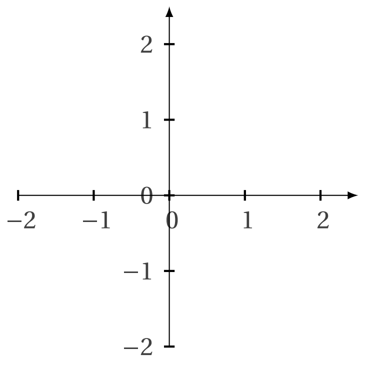
```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=4,ymin=-1,ymax=3]
  \tkzAxeXY[label={},text=blue]
\end{tikzpicture}
```

6.7.2 Option label={}



```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=4,ymin=-1,ymax=2]
  \tkzAxeXY[label={},text=blue,trig=2]
\end{tikzpicture}
```

6.7.3 Option swap



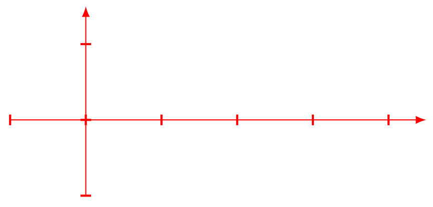
```
\begin{tikzpicture}
\tkzInit[xmin=-2,xmax=2,ymin=-2,ymax=2]
\tkzAxeXY[label={},swap]
\end{tikzpicture}
```

6.8 \tkzDrawXY

`\tkzDrawXY[⟨local options⟩]`

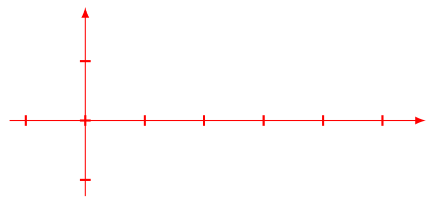
This macro combines the two macros: `\tkzDrawX\tkzDrawY`. It is necessary to use common options as in the example below.

6.8.1 Common colour and empty labels



```
\begin{tikzpicture}
   \tkzInit[xmin=-1,xmax=4,ymin=-1,ymax=1]
   \tkzDrawXY[label={},color=red]
\end{tikzpicture}
```

6.8.2 Two trigonometric axes



```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=4,ymin=-1,ymax=1]
  \tkzDrawXY[label={},color=red,trig=4]
\end{tikzpicture}
```

6.9 \tkzLabelXY

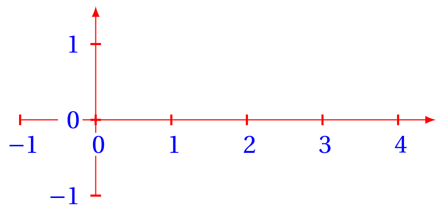
`\tkzLabelXY[⟨local options⟩]`

This macro combines the two macros:

`\tkzLabelX\tkzLabelY`

It is necessary to use common options as in the example below.

6.9.1



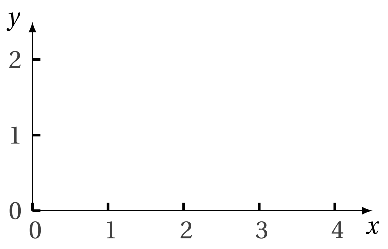
```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=4,ymin=-1,ymax=1]
  \tkzDrawXY[label={},color=red]
  \tkzLabelXY[text=blue]
\end{tikzpicture}
```

6.10 Changing values by axis default

`\tkzSetUpAxis[⟨local options⟩]`

options	default	definition
line width	0.4pt	line width defines the width of the line
tickwd	0.8pt	tick thickness
ticka	1pt	right side or above the tick
tickb	1pt	left side or below the tick
font	\textstyle	graduation size.

6.10.1 Changing the default axes



```
\begin{tikzpicture}[scale=1]
 \tkzInit[ymax=2,xmax=4]
 \tkzSetUpAxis[line width=1pt,tickwd=1pt,ticka=3pt, tickb=0pt]
 \tkzAxeXY
\end{tikzpicture}
```

You have to run **`\tkzSetUpAxis`** again to retrieve the default values.

```
\tkzSetUpAxis[line width=1pt,tickwd=1pt,ticka=2pt,tickb=2pt]
```

7 Use of \tkzGrid

`\tkzGrid[⟨local options⟩](⟨xA ; yA⟩) (⟨xB ; yB⟩)`

A few changes for this macro. First of all, to simplify currently the color of the thinnest grid is determined automatically from the main grid, same process for the thickness. This behavior can be modified using styles.

options	default	definition
(⟨x_A ; y_A⟩) (⟨x_B ; y_B⟩)	(xmin,ymin)(xmax,ymax)	grid pattern

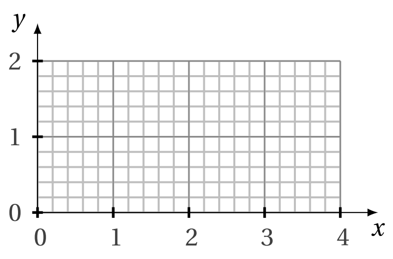
options	default	definition
sub	true	asks for a sub-grid
color	darkgray	main grid color
subxstep	0.2	the step of the subgraduations for the abscissa axis
subystep	0.2	the step of the subgraduations for the ordinate axis
line width	0.4pt	main grid line thickness

Default values can be changed in the configuration file or by macros. The color of the second grid is the same as the main grid, but less intense (by default `gray!50`).

Same behavior for the line thickness (by default `0.75 of line width`). See the examples to change this behavior.

7.0.1 \tkzGrid and the option sub

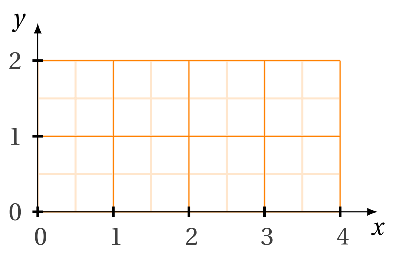
The option **sub** allows you to display a finer secondary grid. It is preferable to run **\tkzGrid** first, to prevent the grid from being overlapped with other elements.



```
\begin{tikzpicture}
 \tkzInit[xmax=4, ymax=2]
 \tkzGrid[sub]
 \tkzAxeXY
\end{tikzpicture}
```

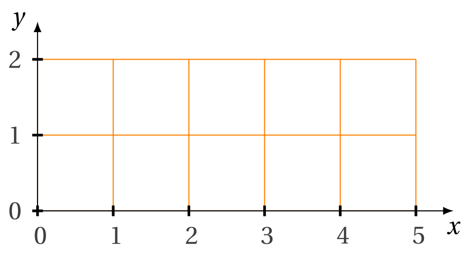
7.0.2 Option sub

The option **sub** allows to display a finer secondary grid. Some parameters are modifiable.



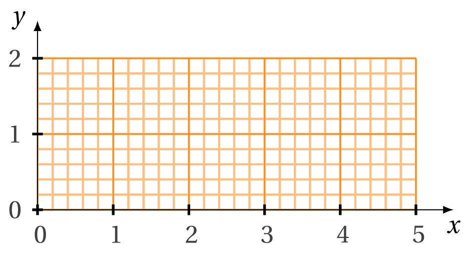
```
\def\tkzCoeffSubColor{20}% instead of 50
\def\tkzCoeffSubLw{0.2}% instead of 0.75
\begin{tikzpicture}
 \tkzInit[xmax=4, ymax=2]
 % we can change the step for the second grid
 \tkzGrid[sub,color=orange,
         subxstep=.5,subystep=.5]
  \tkzAxeXY
\end{tikzpicture}
```

7.0.3 Almost Default



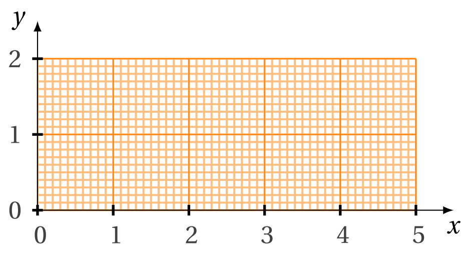
```
\begin{tikzpicture}
  \tkzInit[xmax=5,ymax=2]
  \tkzGrid[color=orange]
  \tkzAxeXY
\end{tikzpicture}
```

7.0.4 Under the grid, too, option sub



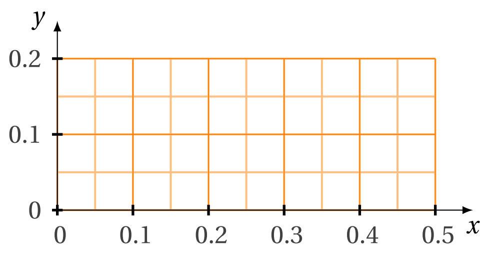
```
\begin{tikzpicture}
 \tkzInit[xmax=5,ymax=2]
 \tkzGrid[sub,color=orange]
 \tkzGrid[color=orange]
 \tkzAxeXY
\end{tikzpicture}
```

7.0.5 Grid change



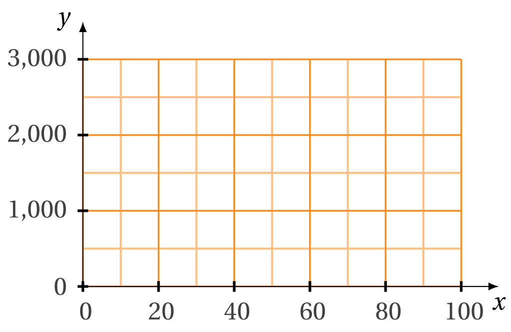
```
\begin{tikzpicture}
  \tkzInit[xmax=5,ymax=2]
  \tkzGrid[color    = orange,
           sub,
           subxstep = 0.1,
           subystep = 0.1]
   \tkzAxeXY
\end{tikzpicture}
```

7.0.6 Option xstep, xstep, subxstep and subystep



```
\begin{tikzpicture}
\tkzInit[xmax=.5,xstep=.1,
         ymax=.2,ystep=.1]
\tkzGrid[sub,
         subxstep = 0.05,
         subystep = 0.05,
         color=orange]
\tkzAxeXY
\end{tikzpicture}
```

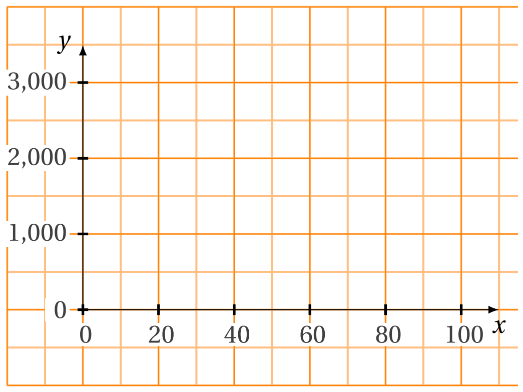
7.0.7 With large intervals



```
\begin{tikzpicture}
   \tkzInit[xmax=100,xstep=20,
            ymax=3000,ystep=1000]
   \tkzGrid[sub,subxstep=10,
                subystep=500,
                color=orange]
   \tkzAxeXY
\end{tikzpicture}
```

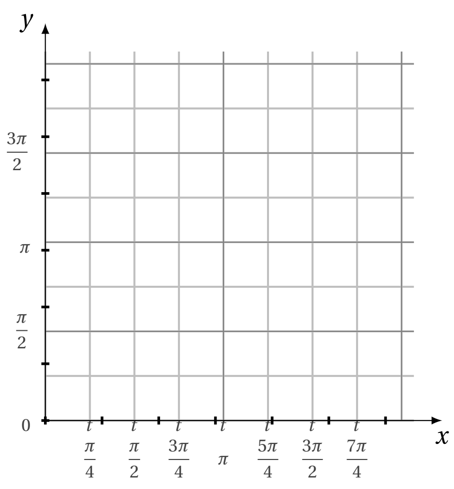
7.0.8 \tkzGrid and the arguments

The grid can be any size.



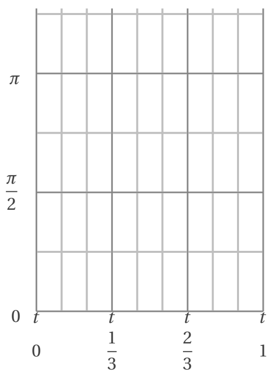
```
\begin{tikzpicture}
   \tkzInit[xmax=100,xstep=20,
            ymax=3000,ystep=1000]
   \tkzGrid[sub,subxstep=10,
            subystep=500,
            color=orange]
            (-20,-1000)(115,4000)%
  \tkzAxeXY
\end{tikzpicture}
```

7.0.9 Use of pi with \tkzGrid



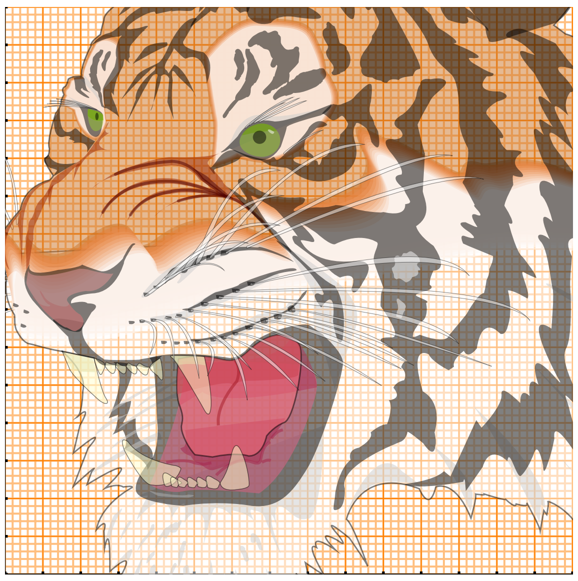
```
\begin{tikzpicture}[scale=.75]
   \tkzInit[xmax=6.5,ymax=6.5]
   \tkzGrid[xstep=pi,ystep=pi/2,sub,
            subxstep=pi/4,subystep=pi/4]
   \tkzLabelX[label=$t$,orig=false,trig=4,
            below=6pt,font=\scriptsize]
   \tkzLabelY[trig=2,font=\scriptsize]
   \tkzDrawXY
\end{tikzpicture}
```

7.0.10 Options frac and trig with \tkzGrid



```
\begin{tikzpicture}
   \tkzInit[xmax=9,xstep=3,ymax=4]
   \tkzGrid[xstep=1,ystep=pi/2,sub,
            subxstep=1,subystep=pi/4]
   \tkzLabelX[label=$t$,orig=false,frac=3,
            below=6pt,font=\scriptsize]
   \tkzLabelY[trig=2,font=\scriptsize]
\end{tikzpicture}
```

7.0.11 Use of a repetition grid



```
\begin{tikzpicture}[scale=.5]
 % \tikzset{xaxe style/.style ={-}}
  \tkzInit[xmax=15,ymax=15]
  \tkzClip
  \tkzGrid[sub,color=orange]
  \tkzLabelX[label= ]   \tkzLabelY[label= ]
  \tkzDrawXY
  \node[opacity=.5] at (8,6){%
    \includegraphics[scale=.5]{tiger}};
\end{tikzpicture}
```

8 The points

I made a distinction between the point used in Euclidean geometry and the point used to represent an element of a statistical cloud. In the first case, I use as object a `node`, which means that the representation of the point cannot be modified by a `scale`; in the second case, I use as object a `plot mark`. The latter can be scaled and have more varied forms than the node.

The new macro is `\tkzDefPoint`, it allows to use Ti*k*Z-specific options as a shift and the values are processed with tkz-base. Moreover, if calculations are needed then the `xfp` package takes care of them. You can use Cartesian or polar coordinates.

8.1 Defining a point in Cartesian coordinates: \tkzDefPoint

```
\tkzDefPoint[⟨local options⟩](⟨x,y⟩){⟨name⟩} or (⟨a:r⟩){⟨name⟩}
```

arguments	default	definition
x,y	no default	x and y are two dimensions, by default in cm.
a:r	no default	a is an angle in degrees, r is a dimension

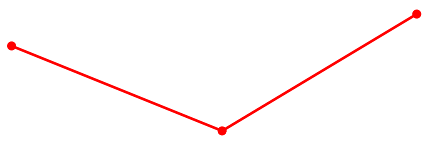
The mandatory arguments of this macro are two dimensions expressed with decimals, in the first case they are two measures of length, in the second case they are a measure of length and the measure of an angle in degrees.

options	default	definition
shift	(0,0)	value spacing

All the options of Ti*k*Z that we can apply to `coordinate`, are applicable (well I hope!) as for example the option `label` defined with the library `quotes`.

8.1.1 Use of shift

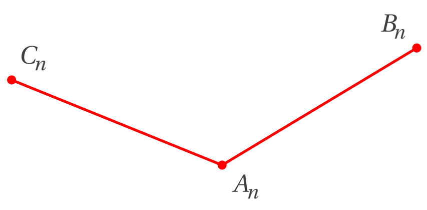
`shift` allows the points to be placed in relation to each other.



```
\begin{tikzpicture}[trim left=-1cm]
 \tkzDefPoint(2,3){A}
 \tkzDefPoint[shift={(2,3)}](31:3){B}
 \tkzDefPoint[shift={(2,3)}](158:3){C}
 \tkzDrawSegments[color=red,line width=1pt](A,B A,C)
 \tkzDrawPoints[color=red](A,B,C)
\end{tikzpicture}
```

8.2 Placing a label with the library quotes

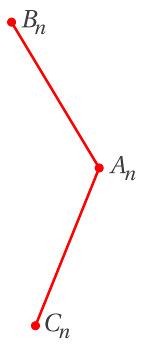
I prefer not to mix operations and use `\tkzLabelPoint` to place labels. See the section ”The Quotes Syntax” in the Ti*k*Z manual.



```
\begin{tikzpicture}[trim left=-1cm]
 \tkzDefPoint["-60:$A_n$" ](2,3){A}
 \tkzDefPoint[shift={(2,3)},%
    "$B_n$" above left](31:3){B}
 \tkzDefPoint[shift={(2,3)},%
     "$C_n$" above right](158:3){C}
 \tkzDrawSegments[color=red,%
          line width=1pt](A,B A,C)
 \tkzDrawPoints[color=red](A,B,C)
\end{tikzpicture}
```

8.2.1 Rotation with shift and scope

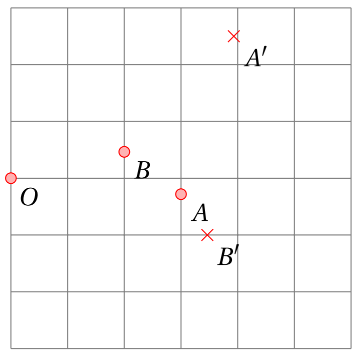
Preferable to rotate is to use a `scope` environment.



```
\begin{tikzpicture}[scale=.75,rotate=90]
 \tkzDefPoint[label=right:$A_n$](2,3){A}
 \begin{scope}[shift={(A)}]
   \tkzDefPoint[label= right:$B_n$](31:3){B}
   \tkzDefPoint[label= right:$C_n$](158:3){C}
 \end{scope}
  \tkzDrawSegments[color=red,%
           line width=1pt](A,B A,C)
  \tkzDrawPoints[color=red](A,B,C)
 \end{tikzpicture}
```

8.2.2 Forms and coordinates

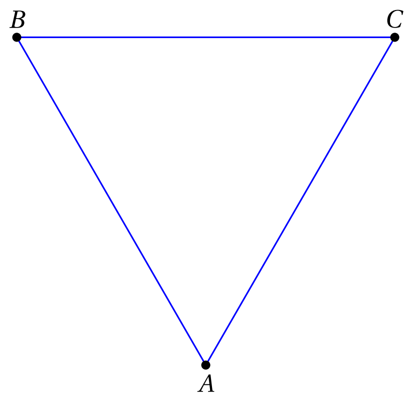
Here we must follow the syntax of `xfp`. It is always possible to go through `pgfmath` but in this case, the coordinates must be calculated before using the macro `\tkzDefPoint`.



```
\begin{tikzpicture}[scale=.75]
  \tkzInit[xmax=6,ymax=6]
  \tkzGrid
  \tkzSetUpPoint[shape = circle,color = red,%
                 size = 4,fill = red!30]
  \tkzDefPoint(-1+1,-1+4){O}
  \tkzDefPoint({3*ln(exp(1))},{exp(1)}){A}
  \tkzDefPoint({4*sin(pi/6)},{4*cos(pi/6)}){B}
  \tkzDefPoint({4*sin(pi/3)},{4*cos(pi/3)}){B'}
  \tkzDefPoint[shift={(1,3)}](30:3){A'}
  \tkzDrawPoints(O,A,B)
  \tkzDrawPoints[color=red,shape=cross out](B',A')
  \tkzLabelPoints(A,O,B,B',A')
\end{tikzpicture}
```

8.2.3 Scope and \tkzDefPoint

First, we can use the `scope` of Ti*k*Z. In the following example, we have a way to define an isosceles triangle.



```
\begin{tikzpicture}[scale=1]
 \begin{scope}[rotate=30]
  \tkzDefPoint(2,3){A}
  \begin{scope}[shift=(A)]
     \tkzDefPoint(90:5){B}
     \tkzDefPoint(30:5){C}
  \end{scope}
 \end{scope}
\tkzDrawSegments[color=blue](A,B B,C C,A)
\tkzDrawPoints(A,B,C)
\tkzLabelPoints[above](B,C)
\tkzLabelPoints[below](A)
\end{tikzpicture}
```

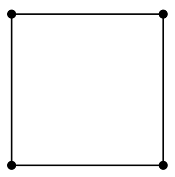
8.3 Definition of points in Cartesian coordinates: \tkzDefPoints

`\tkzDefPoints[⟨local options⟩]{⟨x1/y1/n1,x2/y2/n2, ...⟩}`

x_1 and y_1 are the coordinates of a referenced point n_1

arguments	example
$x_i/y_i/n_i$	`\tkzDefPoints{0/0/O,2/2/A}`

8.3.1 Definition of points



```
\begin{tikzpicture}[scale=1]
 \tkzDefPoints{0/0/A,2/0/B,2/2/C,0/2/D}
 \tkzDrawSegments(D,A A,B B,C C,D)
 %with tkz-euclide \tkzDrawPolygon(A,...,D)
 \tkzDrawPoints(A,B,C,D)
\end{tikzpicture}
```

8.4 Point relative to another: \tkzDefShiftPoint

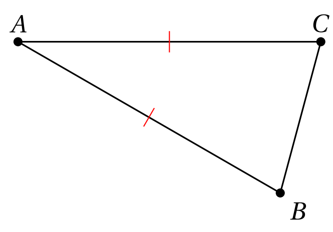
`\tkzDefShiftPoint[⟨Point⟩](⟨x,y⟩){⟨name⟩} ou (⟨a:r⟩){⟨name⟩}`

arguments	default	definition
(x,y)	no default	x and y are two dimensions, by default in cm.
(a:r)	no default	a is an angle in degrees, r is a dimension
point	no default	`\tkzDefShiftPoint[A](0:4){B}`

No options. The name of the point is mandatory.

8.4.1 Example with \tkzDefShiftPoint

This macro allows you to place one point relative to another. This is equivalent to a translation. Here is how to construct an isosceles triangle with main vertex A and angle at vertex of 30°.



```
\begin{tikzpicture}[rotate=-30]
 \tkzDefPoint(2,3){A}
 \tkzDefShiftPoint[A](0:4){B}
 \tkzDefShiftPoint[A](30:4){C}
 \tkzDrawSegments(A,B B,C C,A)
 \tkzMarkSegments[mark=|,color=red](A,B A,C)
 \tkzDrawPoints(A,B,C)
 \tkzLabelPoints[above](A,C)
 \tkzLabelPoints(B)
\end{tikzpicture}
```

8.5 Point relative to another: \tkzDefShiftPointCoord

`\tkzDefShiftPointCoord[⟨a,b⟩](⟨x,y⟩){⟨name⟩} or (⟨a:r⟩){⟨name⟩}`

This involves performing a (a,b) vector translation at the defined point relative to the origin.

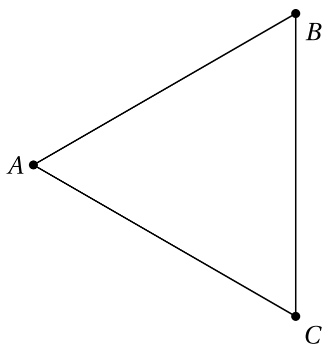
arguments	default	definition
(x,y)	no default	x and y are two dimensions, by default in cm.
(a:r)	no default	a is an angle in degrees, r is a dimension

options	default	example
a,b	no default	\tkzDefShiftPointCoord[2,3](0:4){B}

The option is mandatory

8.5.1 Equilateral triangle with \tkzDefShiftPointCoord

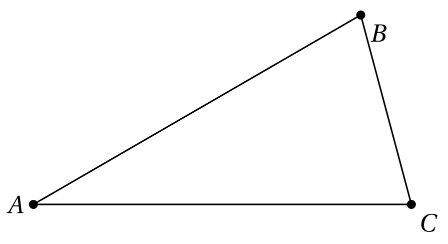
Let's see how to get an equilateral triangle (there is much simpler)



```
\begin{tikzpicture}[scale=1]
 \tkzDefPoint(2,3){A}
 \tkzDefShiftPointCoord[2,3](30:4){B}
 \tkzDefShiftPointCoord[2,3](-30:4){C}
 \tkzDrawSegments(A,B B,C C,A)
 % or \tkzDrawPolygon
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(B,C)
  \tkzLabelPoint[left](A){$A$}
\end{tikzpicture}
```

8.5.2 Isosceles triangle with \tkzDefShiftPointCoord

Let's see how to obtain an isosceles triangle with a principal angle of 30 degrees. Rotation is possible. $AB = AC = 5$ and $\widehat{BAC}$



```
\begin{tikzpicture}[rotate=15]
 \tkzDefPoint(2,3){A}
 \tkzDefShiftPointCoord[2,3](15:5){B}
 \tkzDefShiftPointCoord[2,3](-15:5){C}
 \tkzDrawSegments(A,B B,C C,A)
 \tkzDrawPoints(A,B,C)
 \tkzLabelPoints(B,C)
 \tkzLabelPoint[left](A){$A$}
\end{tikzpicture}
```

8.6 Drawing a point \tkzDrawPoint

`\tkzDrawPoint[⟨local options⟩](⟨point⟩)`

arguments	default	definition
point	no default	a name or reference is requested

The argument is mandatory, but it is not necessary (although recommended) to use a reference; a pair of coordinates placed between braces is accepted. The disk takes the color of the circle, but 50% lighter. It is possible to modify everything. The point is a node and is therefore invariant if the drawing is modified by scaling..

options	default	definition
shape	circle	Possible **cross** or **cross out**
size	2 pt	disk size
color	black	the default color can be changed

We can create other forms such as `cross`

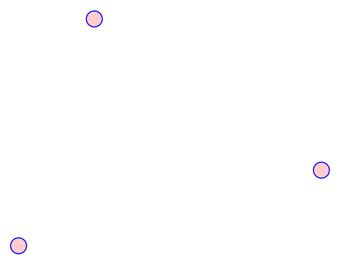
8.6.1 Default stitch style

```
\begin{tikzpicture}
 \tkzDefPoint(1,3){A}
 \tkzDrawPoint(A)
\end{tikzpicture}
```

8.6.2 Changing the style

The default definition is in the file **tkz-base.cfg**

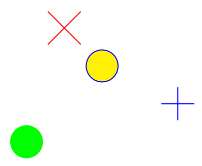
```
\tikzset{point style/.style={draw         = \tkz@euc@pointcolor,
                             inner sep    = 0pt,
                             shape        = \tkz@euc@pointshape,
                             minimum size = \tkz@euc@pointsize,
                             fill         = \tkz@euc@pointcolor!50}}
```



```
\begin{tikzpicture}
 \tikzset{point style/.style={%
   draw         = blue,
   inner sep    = 0pt,
   shape        = circle,
   minimum size = 6pt,
   fill         = red!20}}
 \tkzDefPoint(1,3){A}
 \tkzDefPoint(4,1){B}
 \tkzDefPoint(0,0){O}
 \tkzDrawPoint(A)
 \tkzDrawPoint(B)
 \tkzDrawPoint(O)
\end{tikzpicture}
```

8.6.3 Example of point plots

Note that `scale` does not affect the shape of the dots. Which is normal. Most of the time, we are satisfied with a single point shape that we can define from the beginning, either with a macro or by modifying a configuration file.



```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoint(1,3){A}
 \tkzDefPoint(4,1){B}
 \tkzDefPoint(0,0){O}
 \tkzDrawPoint[shape=cross out,size=12,color=red](A)
 \tkzDrawPoint[shape=cross,size=12,color=blue](B)
 \tkzDrawPoint[size=12,color=green](O)
 \tkzDrawPoint[size=12,color=blue,fill=yellow]({2,2})
\end{tikzpicture}
```

It is possible to draw several points at once, but this macro is a little slower than the previous one. Moreover, we have to make do with the same options for all the points.

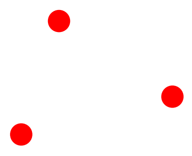
8.7 Drawing points \tkzDrawPoints

`\tkzDrawPoints[⟨local options⟩](⟨liste⟩)`

arguments	default	definition
`points list`	`no default`	`example \tkzDrawPoints(A,B,C)`

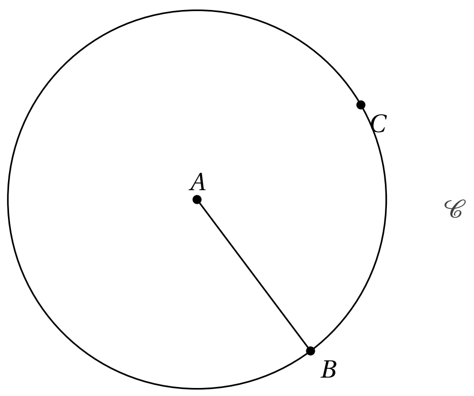
Warning at the final "s", an oversight leads to cascading errors if you attempt to plot multiple points. The options are the same as for the previous macro.

8.7.1 Example with \tkzDefPoint and \tkzDrawPoints



```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoint(1,3){A}
 \tkzDefPoint(4,1){B}
 \tkzDefPoint(0,0){O}
 \tkzDrawPoints[size=8,color=red](A,B,O)
\end{tikzpicture}
```

8.7.2 More complex example



```
\begin{tikzpicture}[scale=.5]
 \tkzDefPoint(2,3){A}  \tkzDefPoint(5,-1){B}
 \tkzDefPoint[label=below:$\mathcal{C}$,
              shift={(2,3)}](-30:5.5){E}
 \begin{scope}[shift=(A)]
   \tkzDefPoint(30:5){C}
 \end{scope}
 \tkzCalcLength[cm](A,B)\tkzGetLength{rAB}
 \tkzDrawCircle[R](A,\rAB)
 \tkzDrawSegment(A,B)
 \tkzDrawPoints(A,B,C)
 \tkzLabelPoints(B,C)
 \tkzLabelPoints[above](A)
\end{tikzpicture}
```

8.8 Add a label to a point \tkzLabelPoint

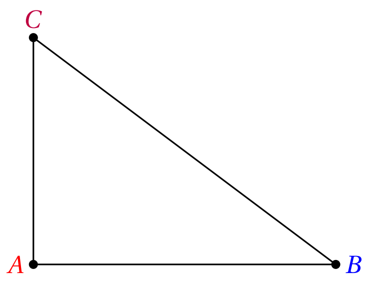
It is possible to add several labels at the same point by using this macro several times.

\tkzLabelPoint[⟨local options⟩](⟨point⟩){⟨label⟩}

arguments	example	
point	\tkzLabelPoint(A){A_1}	
options	default	definition
TikZ options		colour, position etc.

Optionally, we can use any style of Ti*k*Z, especially placement with above, right, dots...

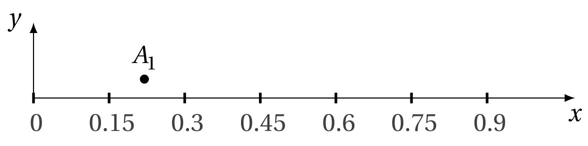
8.8.1 Example with \tkzLabelPoint



```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(4,0){B}
  \tkzDefPoint(0,3){C}
  \tkzDrawSegments(A,B B,C C,A)
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoint[left,red](A){$A$}
  \tkzLabelPoint[right,blue](B){$B$}
  \tkzLabelPoint[above,purple](C){$C$}
\end{tikzpicture}
```

8.8.2 Label and reference

The reference of a point is the object that allows to use the point, the label is the name of the point that will be displayed.



```
\begin{tikzpicture}
   \tkzInit[xmax=1,xstep=0.15,ymax=.5]
   \tkzAxeX \tkzDrawY[noticks]
   \tkzDefPoint(0.22,0.25){A}
   \tkzDrawPoint(A)
   \tkzLabelPoint[above](A){$A_1$}
 \end{tikzpicture}
```

8.9 Add labels to points \tkzLabelPoints

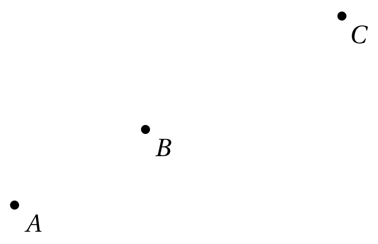
It is possible to place several labels quickly when the point references are identical to the labels and when the labels are placed in the same way in relation to the points. By default, `below right` is chosen.

`\tkzLabelPoints[⟨local options⟩](⟨$A_1, A_2, ...$⟩)`

arguments	example	result
list of points	`\tkzLabelPoints(A,B,C)`	Display of A, B and C

This macro reduces the number of lines of code, but it is not obvious that all points need the same label positioning.

8.9.1 Example with \tkzLabelPoints



```
\begin{tikzpicture}
  \tkzDefPoint(2,3){A}
  \tkzDefShiftPoint[A](30:2){B}
  \tkzDefShiftPoint[A](30:5){C}
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(A,B,C)
\end{tikzpicture}
```

8.10 Automatic position of labels \tkzAutoLabelPoints

The label of a point is placed in a direction defined by a `center` and a point . The distance to the point is determined by a percentage of the distance between the center and the point. This percentage is given by `dist`.

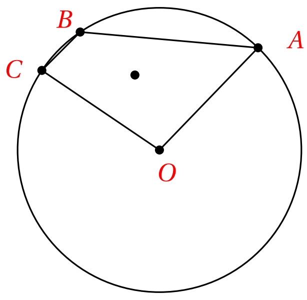
`\tkzLabelPoints[⟨local options⟩](⟨$A_1, A_2, ...$⟩)`

arguments	example	result
list of points	`\tkzLabelPoint(A,B,C)`	Display of A, B and C

options	default	definition
center	no default	you need to deisgn a center
dist	0.15	percentage change in the distance between the center and the points

8.10.1 Example 1 with \tkzAutoLabelPoints

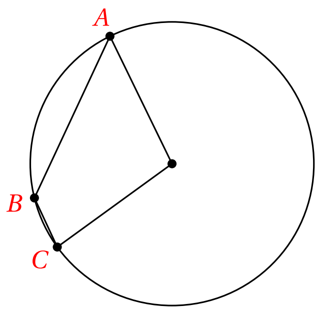
Here the points are positioned relative to the center of gravity of A, B, C et O.



```
\begin{tikzpicture}[scale=1.25]
  \tkzDefPoint(2,1){O}
  \tkzDefRandPointOn[circle=center O radius 1.5]
  \tkzGetPoint{A}
  \tkzDrawCircle(O,A)
  \tkzDefPointBy[rotation=center O angle 100](A)
  \tkzGetPoint{C}
  \tkzDefPointBy[rotation=center O angle 78](A)
  \tkzGetPoint{B}
  \tkzDrawPoints(O,A,B,C)
  \tkzDrawSegments(C,B B,A A,O O,C)
  \tkzDefCentroid(A,B,C,O)
  \tkzDrawPoint(tkzPointResult)
  \tkzAutoLabelPoints[center=tkzPointResult,
                     dist=.3,red](O,A,B,C)
\end{tikzpicture}
```

8.10.2 Example 2 with \tkzAutoLabelPoints

This time the reference is O and the distance is by default 0.15.



```
\begin{tikzpicture}[scale=1.25]
 \tkzDefPoint(2,1){O}
 \tkzDefRandPointOn[circle=center O radius 1.5]
 \tkzGetPoint{A}
 \tkzDrawCircle(O,A)
 \tkzDefPointBy[rotation=center O angle 100](A)
 \tkzGetPoint{C}
 \tkzDefPointBy[rotation=center O angle 78](A)
 \tkzGetPoint{B}
 \tkzDrawPoints(O,A,B,C)
 \tkzDrawSegments(C,B B,A A,O O,C)
 \tkzAutoLabelPoints[center=O,red](A,B,C)
\end{tikzpicture}
```

8.11 Point style with \tkzSetUpPoint

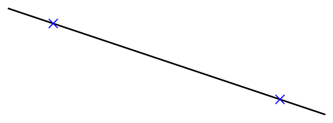
It is important to understand that the size of a dot depends on the size of a line.

\tkzSetUpPoint[⟨local options⟩]

options	default	definition
shape	circle	possible: circle, cross, cross out
size	current	the size of the point is size * line width
color	current	
fill	current!50	

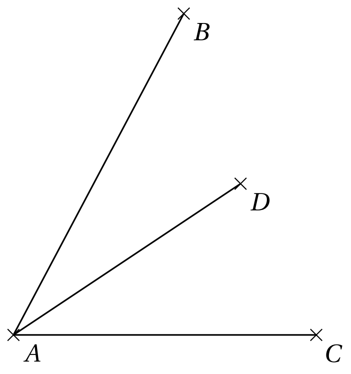
This is a macro for choosing a style for points.

8.11.1 Simple example with \tkzSetUpPoint



```
\begin{tikzpicture}
 \tkzSetUpPoint[shape = cross out,
                  color=blue]
 \tkzInit[xmax=100,xstep=20,ymax=.5]
 \tkzDefPoint(20,1){A}
 \tkzDefPoint(80,0){B}
 \tkzDrawLine(A,B)
 \tkzDrawPoints(A,B)
\end{tikzpicture}
```

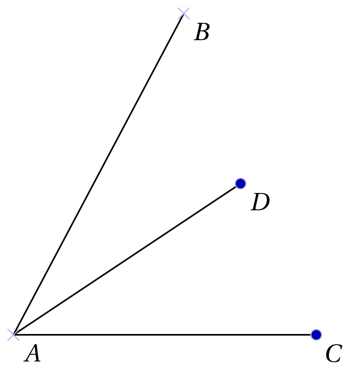
8.11.2 Second example with \tkzSetUpPoint



```
\begin{tikzpicture}
  \tkzInit[ymin=-0.5,ymax=3,xmin=-0.5,xmax=7]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(02.25,04.25){B}
  \tkzDefPoint(4,0){C}
  \tkzDefPoint(3,2){D}
  \tkzDrawSegments(A,B A,C A,D)
  \tkzSetUpPoint[shape=cross out,size=4,]
  \tkzDrawPoints(A,B,C,D)
  \tkzLabelPoints(A,B,C,D)
\end{tikzpicture}
```

8.11.3 Using \tkzSetUpPoint in a group

Only the points in the group are affected by the changes.



```
\begin{tikzpicture}
  \tkzInit[ymin=-0.5,ymax=3,xmin=-0.5,xmax=7]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(02.25,04.25){B}
  \tkzDefPoint(4,0){C}
  \tkzDefPoint(3,2){D}
  \tkzDrawSegments(A,B A,C A,D)
{\tkzSetUpPoint[shape=cross out,
            fill= blue!70!black!!50,
            size=4,color=blue!70!black!30]
  \tkzDrawPoints(A,B)}
  \tkzSetUpPoint[fill= blue!70!black!!50,size=4,
               color=blue!70!black!30]
   \tkzDrawPoints(C,D)
  \tkzLabelPoints(A,B,C,D)
\end{tikzpicture}
```

8.12 Show point coordinates

This macro allows you to display the coordinates of a point and to draw arrows to specify the abscissa and ordinate. The point is given by its reference (its name). It is possible to give a couple of coordinates.

`\tkzPointShowCoord[⟨local options⟩](⟨point⟩)`

argument	example	explanation
(⟨ref⟩)	`\tkzPointShowCoord(A)`	shows the coordinates of point A

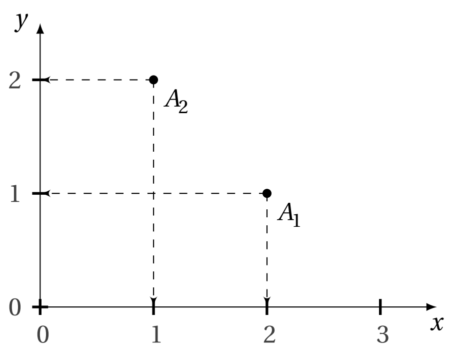
option	default	explication
xlabel	empty	label abscissa
xstyle	empty	style for the abscissa label node example text=red
noxdraw	false	boolean for not draw an arrow to the X-axis $(x'x)$
ylabel	empty	idem
ystyle	empty	idem
noydraw	false	idem

8.12.1 Default styles

Here are some of the main styles:

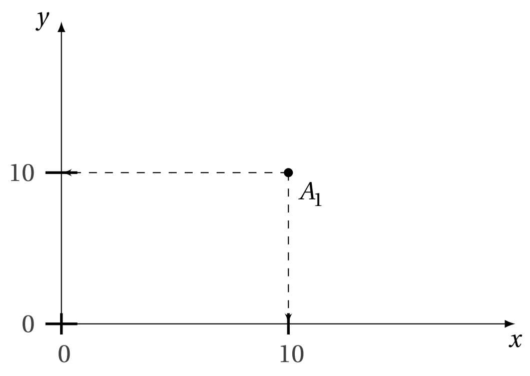
```
\tikzset{arrow coord style/.style={dashed,
                              \tkz@euc@linecolor,
                              >=latex',
                              ->}}
\tikzset{xcoord style/.style={\tkz@euc@labelcolor,
                            font=\normalsize,text height=1ex,
                            inner sep = 0pt,
                            outer sep = 0pt,
                            fill=\tkz@fillcolor,
                            below=3pt}}
\tikzset{ycoord style/.style={\tkz@euc@labelcolor,
                            font=\normalsize,text height=1ex,
                            inner sep = 0pt,
                            outer sep = 0pt,
                            fill=\tkz@fillcolor,
                            left=3pt}}
```

8.12.2 Example with \tkzPointShowCoord



```
\begin{tikzpicture}[scale=1.5]
 \tkzInit[xmax=3,ymax=2]
 \tkzAxeXY
 \tkzDefPoint(2,1){a}
 \tkzPointShowCoord(a)
 \tkzDrawPoint(a)
 \tkzLabelPoint(a){$A_1$}
 \tkzPointShowCoord({1,2})
 \tkzDrawPoint({1,2})
 \tkzLabelPoint({1,2}){$A_2$}
\end{tikzpicture}
```

8.12.3 Example with \tkzPointShowCoord and xstep



```
\begin{tikzpicture}[xscale=3,yscale=2]
 \tkzInit[xmax=15,ymax=15,
         xstep=10,ystep=10]
 \tkzAxeXY
 \tkzDefPoint(10,10){a} \tkzDrawPoint(a)
 \tkzPointShowCoord(a)
 \tkzLabelPoint(a){$A_1$}
\end{tikzpicture}
```

8.13 \tkzDefSetOfPoints

It was already possible to create a scatter plot with the macro **\tkzDefPoints**, but this requires making a reference (a name) to each point, which is sometimes tedious. The macro **\tkzSetOfPoints** allows to define points **tkzPt1**, **tkzPt2**, etc.

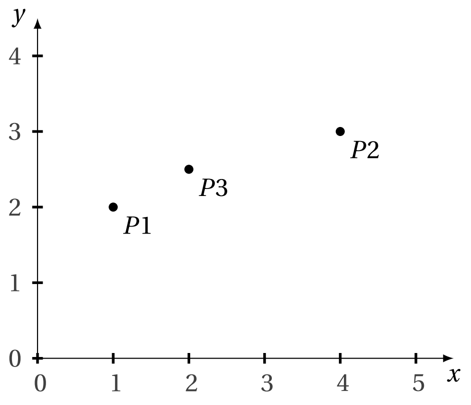
This is frequently referred to as ”scatter plot”. The difference from the macro **\tkzDefPoints** is that the reference to the points is given by a prefix (default **tkzPt**) and the point number. The points are not drawn.

\tkzDefSetOfPoints[⟨local options⟩]{⟨$x_1/y_1,x_2/y_2,\dots,x_n/y_n$⟩}

arguments	default	definition
x_n/y_n	no default	List of couples x_n/y_n separated by commas

options	default	definition
prefix	tkzPt	prefix for point names

8.13.1 Creating a scatter plot with \tkzDefSetOfPoints



```
\begin{tikzpicture}
  \tkzInit[ymax=4,xmax=5]
  \tkzAxeXY
  \tkzDefSetOfPoints[prefix=P]%
           {1/2,4/3,2/2.5}
  \tkzDrawPoints(P1,P2,P3)
  \tkzLabelPoints(P1,P2,P3)
\end{tikzpicture}
```

9 Style Use

9.1 Modification of tkz-base

tkz-base.sty has a default configuration file. Its existence is not mandatory, but if it exists, you can modify it to get different default styles. I only give a quick description of this file, as it may evolve soon.

In **tkz-base.cfg**, you can set the axes, the reference (if used), the grid, etc. as well as the styles which are linked to these objects. It is possible to modify the styles of the points and segments.

It is also possible to define the dimensions of a drawing by default by modifying **xmin**, **xmax**, **ymin** and **ymax**.

```
\def\tkz@xa{0}
\def\tkz@xb{10}
\def\tkz@ya{0}
\def\tkz@yb{10}
```

These lines are used to define the values of **xmin**, **xmax**, etc.

You can change them, for example:

```
\def\tkz@xa{-5}
\def\tkz@xb{-5}
\def\tkz@ya{5}
\def\tkz@yb{5}
```

Here's a list of used styles you'll find in **tkz-base.cfg**

- xlabel style
- xaxe style
- ylabel style
- yaxe style
- rep style
- line style
- point style
- mark style
- compass style
- vector style
- arrow coord style
- xcoord style
- ycoord style

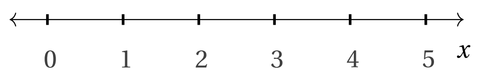
9.2 Use \tikzset

It's better to use **\tikzset** now rather than **\tikzstyle** and it's possible to use **tkz-base.cfg**.

If you want to change the appearance of the axes of the orthogonal coordinate system, for example place arrows at each end or remove them. This can be done in **tkz-base.cfg** or in your code.

```
\tikzset{xaxe style/.style ={>=latex,<->}}
```

The transformation will be valid for the entire document. Note that **`xmin`** has been modified, in fact the arrow and the line corresponding to the graduation merge.



```
\tikzset{xaxe style/.style = {<->}}
\tikzset{xlabel style/.style={below=6pt}}
\begin{tikzpicture}
  \tkzInit[xmin=-0.5,xmax=5]
  \tkzDrawX
  \tkzLabelX
\end{tikzpicture}
```

10 Controlling Bounding Box

From the **PgfManual** :"When you add the clip option, the current path is used for clipping subsequent drawings. Clipping never enlarges the clipping area. Thus, when you clip against a certain path and then clip again against another path, you clip against the intersection of both. The only way to enlarge the clipping path is to end the pgfscope in which the clipping was done. At the end of a pgfscope the clipping path that was in force at the beginning of the scope is reinstalled."

First of all, you don't have to deal with Ti*k*Z the size of the bounding box. Early versions of `tkz-euclide` did not control the size of the bounding box, now with `tkz-base` 4 the size of the bounding box is limited.

The initial bounding box after using the macro `\tkzInit` is defined by the rectangle based on the points $(0,0)$ and $(10,10)$. The `\tkzInit` macro allows this initial bounding box to be modified using the arguments (**`xmin`**, **`xmax`**, **`ymin`**, and **`ymax`**). Of course any external trace modifies the bounding box. Ti*k*Z maintains that bounding box. It is possible to influence this behavior either directly with commands or options in Ti*k*Z such as a command like `\useasboundingbox` or the option **`use as bounding box`**. A possible consequence is to reserve a box for a figure but the figure may overflow the box and spread over the main text. The following command `\pgfresetboundingbox` clears a bounding box and establishes a new one.

10.1 Utility of \tkzInit

However, it is sometimes necessary to control the size of what will be displayed. To do this, you need to have prepared the bounding box you are going to work in, this is the role of the macro `\tkzInit`. For some drawings, it is interesting to fix the extreme values (xmin,xmax,ymin and ymax) and to "clip" the definition rectangle in order to control the size of the figure as well as possible.

The two macros that are useful for controlling the bounding box:

- `\tkzInit`
- `\tkzClip`

To this, I added macros directly linked to the bounding box. You can now view it, backup it, restore it (see the section Bounding Box).

10.2 \tkzInit

```
\tkzInit[⟨local options⟩]
```

options	default	definition
xmin	0	minimum value of the abscissae in cm
xmax	10	maximum value of the abscissae in cm
xstep	1	difference between two graduations in x
ymin	0	minimum y-axis value in cm
ymax	10	maximum y-axis value in cm
ystep	1	difference between two graduations in y

The role of `\tkzInit` is to define a orthogonal coordinates system and a rectangular part of the plane in which you will place your drawings using Cartesian coordinates. This macro allows you to define your working environment as with a calculator. With **`tkz-base`** 4 `\xstep` and `\ystep` are always 1. Logically it is no longer useful to use `\tkzInit`, except for an action like "Clipping Out".

10.3 \tkzClip

10.4 tkzClip

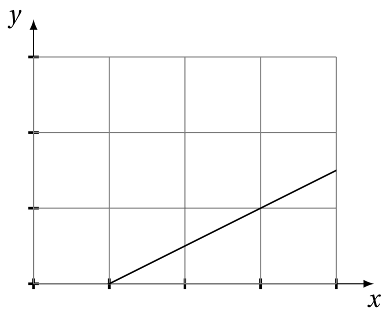
```
\tkzClip[⟨local options⟩]
```

The role of this macro is to make invisible what is outside the rectangle defined by (xmin ; ymin) and (xmax ; ymax).

options	default	definition
space	1	added value on the right, left, bottom and top of the background

The role of the **space** option is to enlarge the visible part of the drawing. This part becomes the rectangle defined by (xmin-space ; ymin-space) and (xmax+space ; ymax+space). **space** can be negative! The unit is cm and should not be specified.

The role of this macro is to "clip" the initial rectangle so that only the paths contained in this rectangle are drawn.



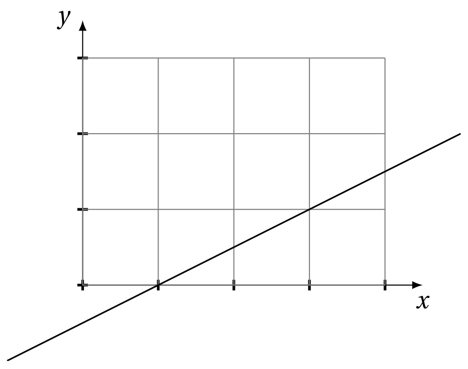
```
\begin{tikzpicture}
 \tkzInit[xmax=4, ymax=3]
 \tkzDefPoints{-1/-1/A,5/2/B}
 \tkzDrawX \tkzDrawY
 \tkzGrid
 \tkzClip
 \tkzDrawSegment(A,B)
\end{tikzpicture}
```

It is possible to add a bit of space

```
\tkzClip[space=1]
```

10.5 \tkzClip and the option space

This option allows you to add some space around the "clipped" rectangle.



```
\begin{tikzpicture}
 \tkzInit[xmax=4, ymax=3]
 \tkzDefPoints{-1/-1/A,5/2/B}
 \tkzDrawX \tkzDrawY
 \tkzGrid
 \tkzClip[space=1]
 \tkzDrawSegment(A,B)
\end{tikzpicture}
```

The dimensions of the "clipped" rectangle are `xmin-1`, `ymin-1`, `xmax+1` and `ymax+1`.

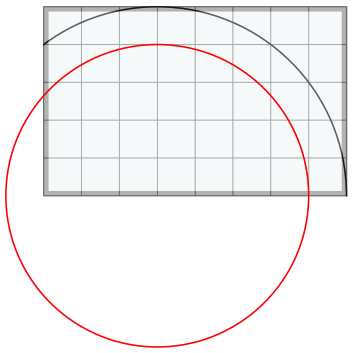
10.6 tkzShowBB

The simplest macro.

```
\tkzShowBB[⟨local options⟩]
```

This macro displays the bounding box. A rectangular frame surrounds the bounding box. This macro accepts Ti*k*Z options.

10.6.1 Example with \tkzShowBB



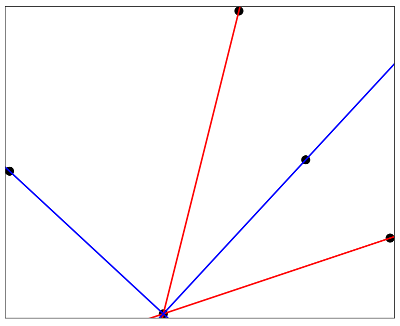
```
\begin{tikzpicture}[scale=.5]
  \tkzInit[ymax=5,xmax=8]
  \tkzGrid
  \tkzDefPoint(3,0){A}
   \begin{scope}
    \tkzClipBB
    \tkzDrawCircle[R](A,5)
     \tkzShowBB[line width = 4pt,fill=teal!10,opacity=.4]
   \end{scope}
\tkzDrawCircle[R,red](A,4)
\end{tikzpicture}
```

10.7 tkzClipBB

```
\tkzClipBB
```

The idea is to limit future constructions to the current bounding box.

10.7.1 Example with \tkzClipBB and the bisectors



```
\begin{tikzpicture}
\tkzInit[xmin=-3,xmax=6, ymin=-1,ymax=6]
\tkzDefPoint(0,0){O}\tkzDefPoint(3,1){I}
\tkzDefPoint(1,4){J}
\tkzDefLine[bisector](I,O,J) \tkzGetPoint{i}
\tkzDefLine[bisector out](I,O,J) \tkzGetPoint{j}
\tkzDrawPoints(O,I,J,i,j)
\tkzClipBB
\tkzDrawLines[add = 1 and 2,color=red](O,I O,J)
\tkzDrawLines[add = 1 and 2,color=blue](O,i O,j)
\tkzShowBB
\end{tikzpicture}
```

11 Use Additional Objects or Tools

These complementary objects can be particular points, straight lines, circles, arcs, etc. Now **tkz-base** has been minimized. If you want to use particular objects you must use **tkz-euclide**.

12 Using an orthogonal coordinate system

12.1 Coordinate system with \tkzRep

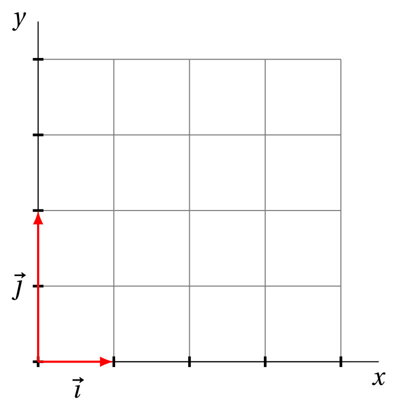
```
\tkzRep[⟨local options⟩]
```

options	default	definition
line width	0.8pt	line width defines the width of the line
xlabel	$\vec{\imath}$	label for the abscissa axis
ylabel	$\vec{\jmath}$	label for the ordinate axis
posxlabel	below=2pt	Label position
posylabel	left=2pt	Label position
xnorm	1	norm of the x-vector
ynorm	1	vector norm in y
color	black	line colour
colorlabel	black	label color

12.1.1 Some modifiable styles

```
\tikzset{xlabel style/.style          =  {below     =  3 pt,
                                            inner sep =  1pt,
                                            outer sep =  0pt}}
\tikzset{ylabel style/.style          =  {left      =  3 pt,
                                            inner sep =  1pt,
                                            outer sep =  0pt}}
\tikzset{xaxe style/.style            =  {>         =  latex,  ->}}
\tikzset{yaxe style/.style            =  {>         =  latex,  ->}}
```

12.1.2 Example of use



```
\begin{tikzpicture}
  \tikzset{xaxe style/.style={-}}
  \tikzset{yaxe style/.style={-}}
  \tkzInit[xmax=4,ymax=4]
  \tkzGrid
  \tkzDrawX
  \tkzDrawY
  \tkzRep[color=red,ynorm=2]
\end{tikzpicture}
```

For those who use **french** with **babel**, in case of problems with version 3 of pgf, just load the **babel** library. Ti*k*Z was indeed sometimes allergic to the active characters.

13 Lines parallel to the axes

13.1 Draw a horizontal line with \tkzHLine

☞💣 The syntax is that of **xfp**!

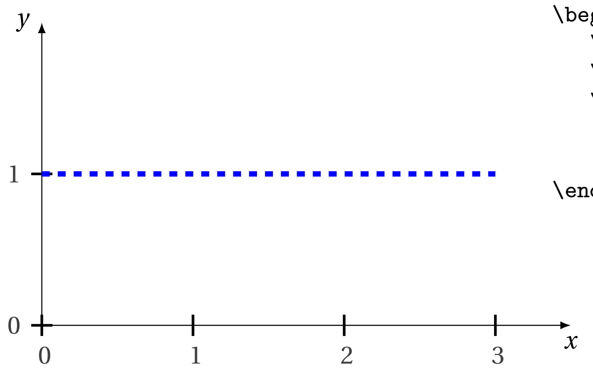
```
\tkzHLine[⟨local options⟩]{⟨decimal number⟩}
```

arguments	example	definition
decimal number	\tkzHLine{1}	Draw the straight line $y = 1$

options	default	definition
color	black	line colour
line width	0.6pt	point thickness
style	solid	line style

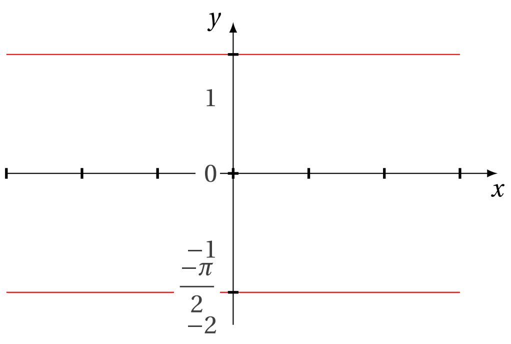
see the lines options in Ti*kZ*

13.1.1 Horizontal line



```
\begin{tikzpicture}[scale=2]
   \tkzInit[xmax=3,ymax=1.5]
   \tkzAxeXY
   \tkzHLine[color      = blue,
             style      = dashed,
             line width = 2pt]{1}
\end{tikzpicture}
```

13.1.2 Horizontal line and value calculated by xfp



```
\begin{tikzpicture}
  \tkzInit[xmin=-3,xmax=3,ymin=-2,ymax=1.5]
  \foreach\v in {-1,1}
  {\tkzHLine[color=red]{\v*pi/2}}
  \tkzDrawX
  \tkzAxeY[trig=2]
  \tkzLabelY
\end{tikzpicture}
```

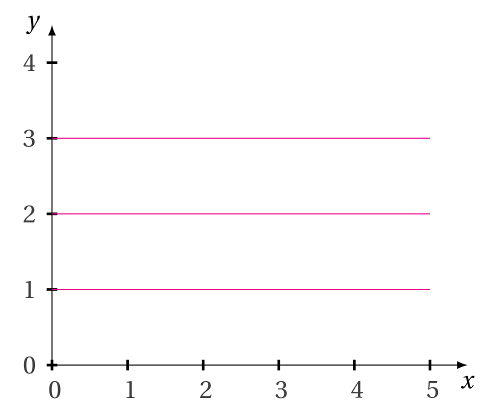
13.2 Horizontal lines with \tkzHLines

The syntax is that of **xfp**!

```
\tkzHLines[⟨local options⟩]{⟨list of values⟩}
```

arguments	example	definition
list of values	\tkzHLines{1,4}	draws the lines $y=1$ and $y=4$

13.2.1 Horizontal lines



```
\begin{tikzpicture}
  \tkzInit[xmax=5,ymax=4]
  \tkzAxeXY
 \tkzHLines[color = magenta]{1,...,3}
\end{tikzpicture}
```

13.3 Draw a vertical line with \tkzVLine

The syntax is that of **xfp**!

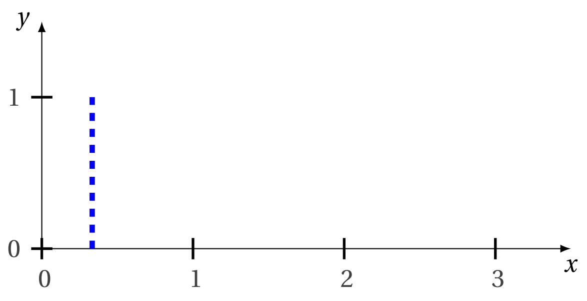
```
\tkzVLine[⟨local options⟩]{⟨decimal number⟩}
```

arguments	example	definition
decimal number	\tkzVLine{1}	Draw the line $x=1$

options	default	definition
color	black	line colour
line width	0.6pt	point thickness
style	solid	line style

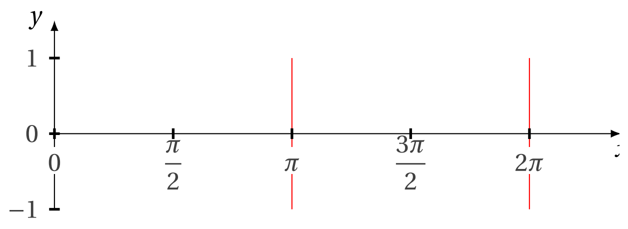
See options the lines in Ti*k*Z.

13.3.1 Vertical line



```
\begin{tikzpicture}[scale=2]
   \tkzInit[xmax=3,ymax=1]
   \tkzAxeXY
   \tkzVLine[color      = blue,
             style      = dashed,
             line width = 2pt]{1/3}
\end{tikzpicture}
```

13.3.2 Vertical line and value calculated by xfp



```
\begin{tikzpicture}
  \tkzInit[xmax=7,ymin=-1,ymax=1]
  \foreach\v in {1,2}
  {\tkzVLine[color=red]{\v*pi}}
  \tkzDrawY
  \tkzAxeX[trig=2]
  \tkzLabelY
\end{tikzpicture}
```

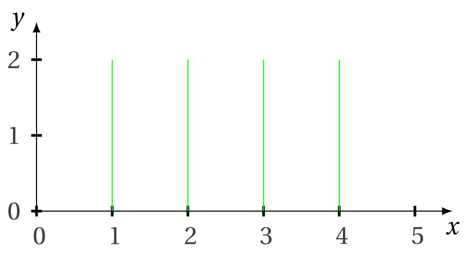
13.4 Vertical lines with \tkzVLines

☞ The syntax is that of **xfp**!

`\tkzVLines[⟨local options⟩]{⟨list of values⟩}`

arguments	example	definition
list of values	\tkzVLines{1,4}	Trace the lines $x=1$ and $x=4$

13.4.1 Vertical lines



```
\begin{tikzpicture}
 \tkzInit[xmax=5,ymax=2]
 \tkzAxeXY
 \tkzVLines[color = green]{1,2,...,4}
\end{tikzpicture}
```

14 Ticks on the axes

14.1 Drawing one tick on the abscissa axis \tkzHTick

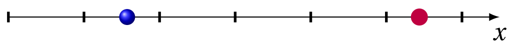
```
\tkzHTick[⟨local options⟩]{⟨decimal number⟩}
```

arguments	example	definition
decimal number	\tkzHTick{1}	the abscissa of the tick is 1

options	default	definition
mark	*	full disk
mark size	3 pt	symbol size
mark options	empty	allows you to use color for example

See options for Ti*k*Z.

14.1.1 Example



```
\begin{tikzpicture}
  \tkzInit[xmax=6]
  \tkzDrawX
  \tkzHTick[mark=ball,mark size=3pt]{pi/2}
  \tkzHTick[mark=*,
     mark options={color=purple}]{2*exp(1)}
\end{tikzpicture}
```

14.2 Drawing ticks on the abscissa axis \tkzHTicks

```
\tkzHTicks[⟨local options⟩]{⟨list of numbers⟩}
```

arguments	example	definition
decimal number	\tkzHTicks{1}	the abscissa of the tick is 1

See options for Ti*k*Z.

14.3 Drawing one tick on the ordinate axis \tkzVTick

```
\tkzVTick[⟨local options⟩]{⟨decimal number⟩}
```

arguments	example	definition
decimal number	\tkzVTick{1}	the ordinate of the tick is 1

See options for Ti*k*Z.

14.4 Drawing ticks on the ordinate axis \tkzVTicks

```
\tkzVTicks[⟨local options⟩]{⟨decimal number⟩}
```

arguments	example	definition
`decimal number`	`\tkzVTicks{1,3}`	`the ordinates of the ticks are 1 and 3`

See options for Ti*k*Z.

15 Marks or symbols

I distinguished between the points used in Euclidean geometry and the ”marks or symbols” that can be found in statistics.

To position the symbol, we use the macro `\tkzDefPoint` to correctly define a point, then the macro `\tkzDrawMark` to draw the symbol.

It is common to have to draw a scatter plot, so I created a macro that allows you to define several points quickly.

A ”mark” symbol can be scaled, which is sometimes useful, but, on the other hand, if you change the abscissa and ordinates differently then the ”marks” are distorted.

Reminder: it was already possible to create a cloud of points with the macro `\tkzDefPoints`, but this requires to give a reference (a name) to each point, which is sometimes tedious. The macro `\tkzSetOfPoints` allows to define points **tkzPt1**, **tkzPt2**, etc.

This is frequently referred to as the ”scatter plot”. The difference from the macro `\tkzDefPoints` is that the reference to the points is given by a prefix (default `tkzPt`) and the point number.

The points are not drawn.

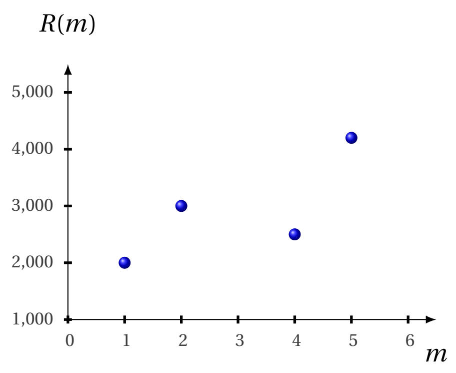
15.1 \tkzDrawSetOfPoints

`\tkzDrawSetOfPoints[⟨local options⟩]`

Allows you to place symbols on the points defined by `\tkzDefSetOfPoints`.

options	default	definition
prefix	tkzPt	point name prefix

15.1.1 Drawing of a scatter plot with \tkzDrawSetOfPoints



```
\begin{tikzpicture}[scale=0.75]
\tkzInit[xmax=6,ymin=1000,ymax=5000,ystep=1000]
\tkzDrawX[label=$m$,below=10pt]
\tkzDrawY[label=$R(m)$,above=10pt]
\tkzLabelX[font=\scriptsize]
\tkzLabelY[font=\scriptsize]
\tkzDefSetOfPoints[show]{1/2000,2/3000,4/2500,5/4200}
\tkzDrawSetOfPoints[mark=ball,mark size=3pt]
\end{tikzpicture}
```

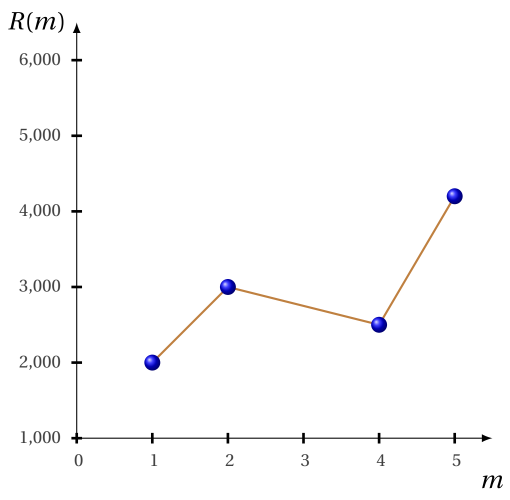
15.2 \tkzJoinSetOfPoints

`\tkzJoinSetOfPoints[⟨local options⟩]`

Allows the symbols to be joined by line segments. Of course, it is possible to use all the options of Ti*k*Z.

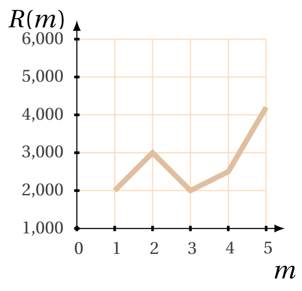
options	default	definition
prefix	tkzPt	point name prefix

15.2.1 Link the points of a scatter plot with \tkzJoinSetOfPoints



```
\begin{tikzpicture}[scale=1]
\tkzInit[xmax=5,ymin=1000,ymax=6000,ystep=1000]
\tkzDrawX[label=$m$,below=13pt]
\tkzDrawY[label=$R(m)$]
\tkzLabelX[font=\scriptsize]
\tkzLabelY[font=\scriptsize]
\tkzDefSetOfPoints{%
   1/2000,2/3000,4/2500,5/4200}
\tkzJoinSetOfPoints[%
      thick, color=brown]
\tkzDrawSetOfPoints[%
      mark=ball, mark size=3pt]
\end{tikzpicture}
```

15.2.2 Using the points of a scatter plot



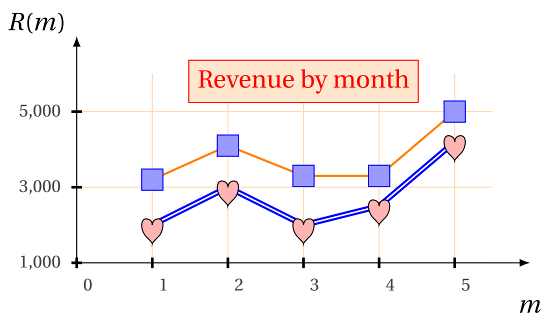
```
\begin{tikzpicture}[scale=.5]
\tkzInit[xmax=5,ymin=1000,
         ymax=6000,ystep=1000]
\tkzGrid[color=orange!30]
\tkzDrawX[label=$m$,below=13pt]
\tkzDrawY[label=$R(m)$]
\tkzLabelX[font=\scriptsize]
\tkzLabelY[font=\scriptsize]
\tkzDefSetOfPoints[prefix=P]{%
   1/2000,2/3000,3/2000,4/2500,5/4200}
\tkzDrawPolySeg[%
     color=brown!50,
     line width=2pt](P1,P2,P3,P4,P5)
\end{tikzpicture}
```

15.3 \tkzSetUpMark

`\tkzSetUpMark[⟨local options⟩]`

options	default	example
mark	no default	\tkzSetUpMark[mark=heart]

15.3.1 Two scatter plots



```
\begin{tikzpicture}
\tkzInit[xmax=5.5,ymin=1000,%
         ymax=6000,ystep=2000]
\tkzGrid[color=orange!30]
\tkzDrawX[label=$m$,below=13pt]
\tkzDrawY[above left,label=$R(m)$]
\tkzLabelX[below right,font=\scriptsize]
\tkzLabelY[font=\scriptsize]
\tkzDefSetOfPoints{1/2000,2/3000,3/2000,
    4/2500,5/4200}
\tkzDefSetOfPoints[prefix=P]{1/3200,2/4100,
    3/3300,4/3300,5/5000}
\tkzSetUpMark[mark=heart,color=black,
   fill=red!30,size=4pt]
\tkzJoinSetOfPoints[thick,color=blue,double]
\tkzDrawSetOfPoints
\tkzJoinSetOfPoints[prefix=P,thick,color=orange]
\tkzDrawSetOfPoints[prefix=P,mark=square*,
     mark size=4pt,
     mark options={color=blue,fill=blue!40}]
\tkzText[draw,color = red,
        fill  = orange!20](3,5800)%
        {Revenue by month}
  \end{tikzpicture}
```

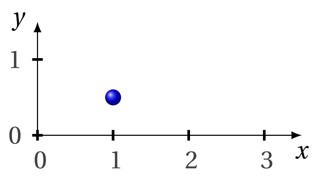
15.4 \tkzDrawMark

```
\tkzDrawMark[⟨local options⟩](⟨point⟩)
```

Place a symbol. More efficient than the next to place a single symbol.

options	default	definition
prefix	tkzPt	point name prefix

15.4.1 Ball; use of \tkzDrawMarks



```
\begin{tikzpicture}
\tkzInit[xmax=3,ymax=1]
\tkzAxeXY
\tkzDrawMark[mark=ball](1,.5)
\end{tikzpicture}
```

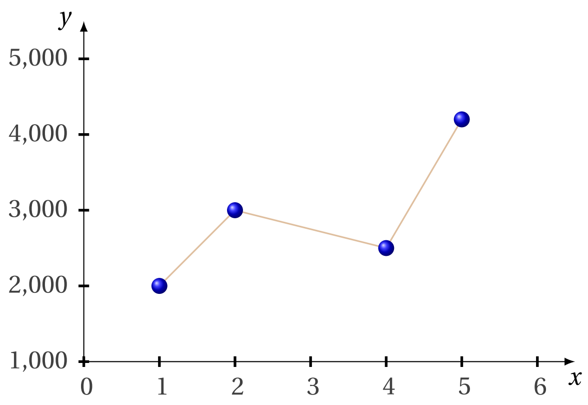
15.5 \tkzDrawMarks

```
\tkzDrawMarks[⟨local options⟩](⟨list of points⟩)
```

Allows you to place a series of marks.

options	default	definition
prefix	tkzPt	point name prefix

15.5.1 Mark and plot; use of \tkzDrawMarks



```
  \begin{tikzpicture}
  \tkzInit[xmax=6,ymin=1000,
          ymax=5000,ystep=1000]
  \tkzAxeXY
  \tkzDefSetOfPoints[prefix=P]{%
        1/2000,
        2/3000,
        4/2500,
        5/4200}
  \tkzDrawSegments[color=brown!50]%
(P1,P2 P2,P3 P3,P4)
  \tkzDrawMarks[mark=ball](P1,P2,P3,P4)
  \end{tikzpicture}
```

16 Texts and Legends

16.1 Placing a title

Of course you can use Ti*k*Z, but the macro I propose to allow you to place the text using the units chosen for the drawing.

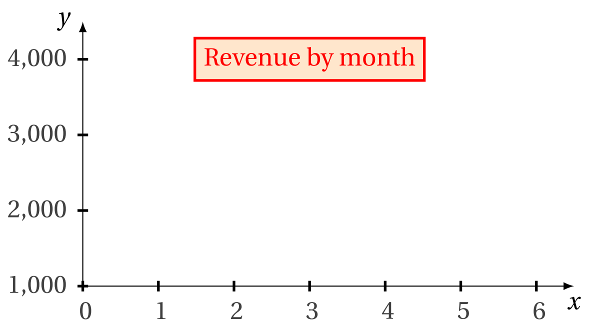
The options are always those of Ti*k*Z, in particular the following ones:

```
\tkzText[⟨local options⟩](⟨dot⟩){⟨text⟩}
```

The point can either be given by its coordinates or by its name.

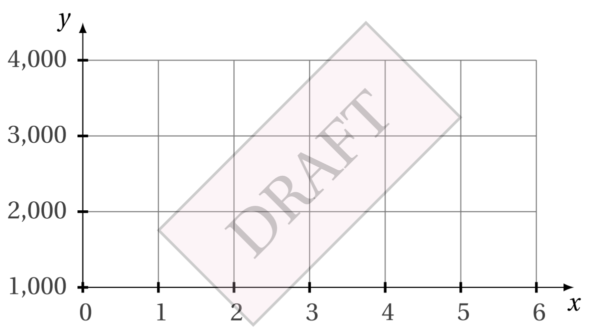
options	default	definition
color	black	current colour
text	black	text colour
fill	white	background colour
opacity	1	opacity

16.1.1 A title



```
\begin{tikzpicture}
  \tkzInit[xmax  = 6,   ymin  = 1000,%
           ymax  = 4000,ystep = 1000]
  \tkzAxeXY
  \tkzText[draw,
           line width = 1pt,%
           color    = red,%
           fill = orange!20](3,4000)%
           {Revenue by month}
\end{tikzpicture}
```

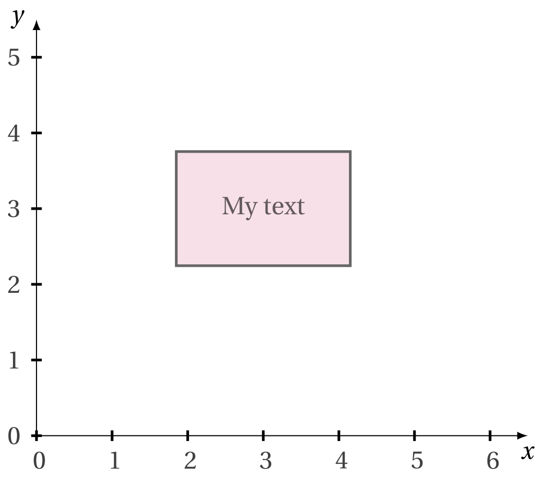
16.1.2 Draft



```
\begin{tikzpicture}
  \tkzInit[xmax  = 6,   ymin  = 1000,%
           ymax  = 4000,ystep = 1000]
  \tkzGrid   \tkzAxeXY
  \tkzText[draw,opacity=.2,
           rotate=45,inner sep=.6 cm,
           line width = 1pt,
           color = black,
           fill = purple!20](3,2500)
           {\Huge DRAFT}
\end{tikzpicture}
```

16.1.3 Text with a point

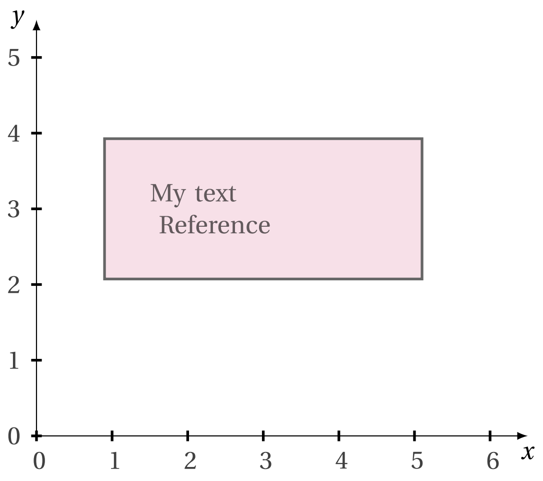
It is possible to give the reference of a point instead of its coordinates.



```
\begin{tikzpicture}
  \tkzInit[ymax=5,xmax=6]
  \tkzAxeXY
  \tkzDefPoint(3,3){A}
  \tkzText[draw,opacity=.6,
           inner sep=.6 cm,
           line width = 1pt,
           color    = black,
           fill = purple!20](A)
           {My text}
\end{tikzpicture}
```

16.1.4 Text format

The option **`text width`** is interesting, see the pgfmanual for more information.



```
\begin{tikzpicture}
 \tkzInit[ymax=5,xmax=6]
 \tkzAxeXY
 \tkzText[draw,opacity=.6,
          inner sep=.6 cm,
          line width = 1pt,
          color    = black,
          fill = purple!20,
          text width=3cm](3,3)
          {My text\\ Reference}
\end{tikzpicture}
```

16.2 Placing legends

There are two ways to use this macro. Either you can place legends for curves. Then you can represent lines with their own style, or you can differentiate symbols (mark).

`\tkzLegend[⟨local options⟩]{⟨mark/color/size/text⟩}`

The arguments differ according to the boolean `line`.

options	default	definition
`line`	`false`	`Boolean: line or symbol`

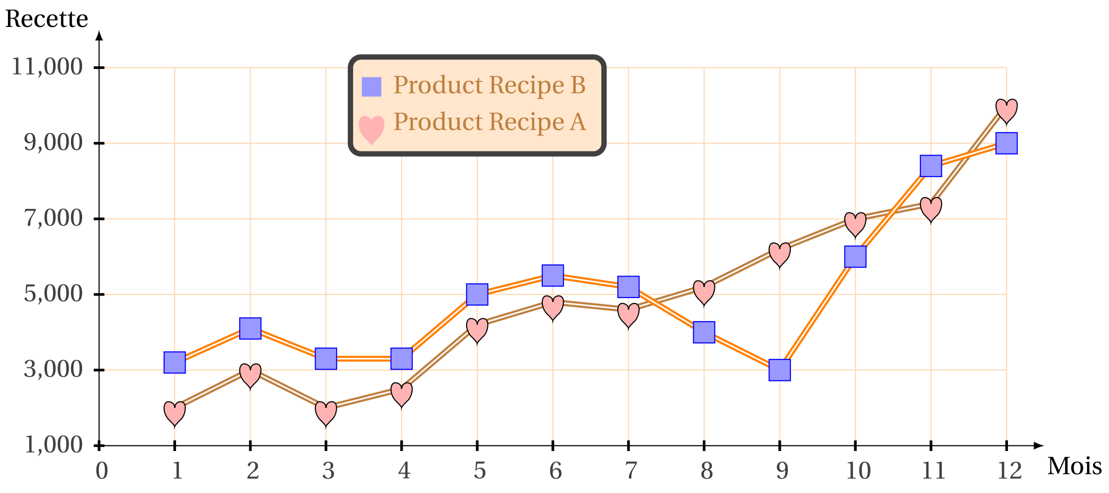
With `line=true`

arguments	default	example
`style/line width/color/text`	`no default`	`dashed/1pt/red/Product Recipe B`

With `line=false`

arguments	default	example
`mark/mark size/color/text`	no default	`heart/1ex/red!30/Product Recipe A`

16.2.1 Legends with symbols



```
\begin{tikzpicture}
\tkzInit[xmax=12,ymin=1000,ymax=11000,ystep=2000]
\tkzGrid[color=orange!30]
\tkzDrawX[below right,label=Mois]
\tkzDrawY[above left,label=Recette]
\tkzLabelX
\tkzLabelY
\tkzDefSetOfPoints{1/2000,2/3000,3/2000,4/2500,5/4200,6/4800,7/4600,
                   8/5200,9/6200,10/7000,11/7400,12/10000}
\tkzDefSetOfPoints[prefix=P]{1/3200,2/4100,3/3300,4/3300,5/5000,6/5500,7/5200,8/4000,
         9/3000,10/6000,11/8400,12/9000}
\tkzSetUpMark[mark=heart,color=black,fill=red!30,size=4pt]
\tkzJoinSetOfPoints[thick,color=brown,double]
\tkzDrawSetOfPoints
\tkzJoinSetOfPoints[prefix=P,thick,color=orange,double]
\tkzDrawSetOfPoints[prefix=P,mark=square*,mark size=4pt,
                    mark options={color=blue,fill=blue!40}]
\tkzLegend[draw,rounded corners,fill=orange!20,text=brown,
          line width=2pt](5,10000){heart/1ex/red!30/Product Recipe A,%
                                   square*/0.75ex/blue!40/Product Recipe B}
\end{tikzpicture}
```

17 FAQ

17.1 General Questions

- **Why `tkz-base` ?** As a Mathematics teacher, I needed tools that would allow me to write my lessons and exercises quickly. Ti*k*Z was perfect for that, but I was wasting too much time on details. I wanted to create a syntax that was both close to that of LaTeXand math so I could memorize better. So I created a module for each branch of mathematics I taught. `tkz-base` is the common part of all these modules. `tkz-euclide` and `tkz-berge` are the ones I invested the most in.

- **Relationship with Ti*k*Z?** Ti*k*Z is a great package for describing drawings. My packages are based on it. That said, it is in no way comparable. My packages are only useful for people who want to create mathematical figures.

17.2 Most common errors

- **Error unknown option: ”label options”**. This option is no longer available. You can now directly use the options in Ti*k*Z.

- **Error with `\tkzDrawPoint` or `\tkzDefPoint` `\tkzDrawPoint(A,B)`** when you need `\tkzDrawPoints`. This is true with all macros that allow you to define multiple objects. The singular form allows you to use custom options. On the other hand, it is possible to use the plural form for a single object.

- **Propagation of a style** It is possible to restrict the propagation of a style by placing a piece of code in a group or in a **scope** environment or between parentheses.

- **The use of the comma** even in a Mathematical mode 2.5 needs to be protected in a TeX group, for example {$2,5$}.

- `\tkzDrawSegments{B,B' C,C'}` is a mistake. Only macros that define an object use braces.

- If an error occurs in a calculation when passing parameters, then it is better to make these calculations before calling the macro.

- Do not mix the syntax of `pgfmath` and `xfp`.