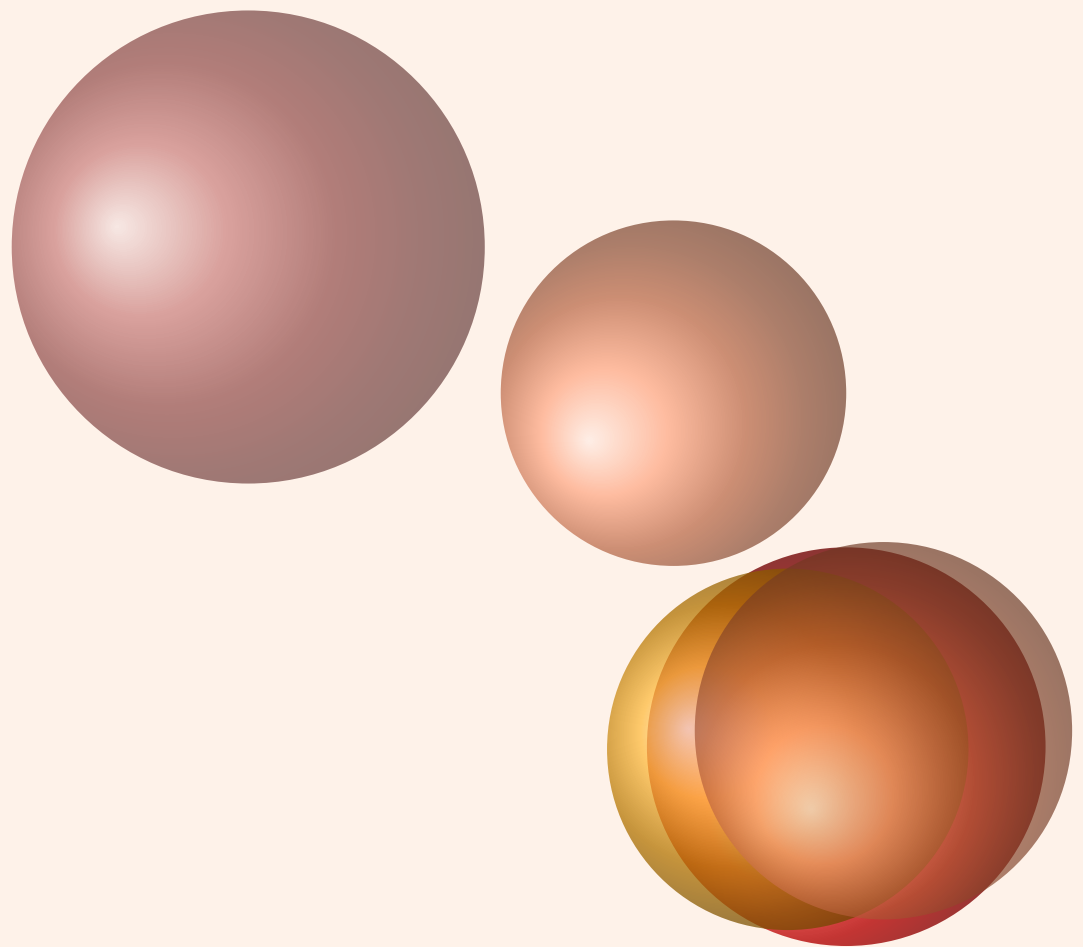


tkz-base 1.16 c

AlterMundus



Alain Matthes

3 juin 2011

<http://altermundus.fr> <http://altermundus.com>

tkz-base

Alain Matthes

tkz-base.sty est un module (package) pour créer à l'aide de **TikZ** des graphiques le plus simplement possible. Il dépend de **TikZ** et est la base sur laquelle sera construite une série de modules ayant comme point commun, la création de dessins utiles dans l'enseignement des mathématiques. Le rôle de **tkz-base.sty** est essentiellement de fournir une macro permettant de définir un repère orthogonal, et de laisser le choix à l'utilisateur des unités graphiques. Ce package existait déjà, et était disponible sur mon site internet. La version « officielle » a pour premier numéro de version 1.13 c (c pour **CTAN**), de plus, la syntaxe a évolué et certaines macros ont commencé une mutation qui permettra de rendre l'ensemble de mes packages plus homogène. Ce package nécessite la version 2.1 de **TikZ**.

☞ Je souhaite remercier **Till Tantau** pour avoir créé le merveilleux outil **TikZ**, ainsi que **Michel Bovani** pour **fourier**, dont l'association avec **utopia** est excellente.

☞ Je remercie **Yve Combe** pour avoir partagé son travail sur le rapporteur et les constructions à l'aide du compas. Je souhaite remercier également, **David Arnold** qui a corrigé un grand nombre d'erreurs et qui a testé de nombreux exemples, **Wolfgang Büchel** qui a corrigé également des erreurs et a construit de superbes scripts pour obtenir les fichiers d'exemples, **John Kitzmiller** et **Dimitri Kapetas** pour leurs exemples, et enfin **Gaétan Marris** pour ses remarques et corrections.

☞ Vous trouverez de nombreux exemples sur mes sites : altermundus.fr ou altermundus.com

Vous pouvez envoyer vos remarques, et les rapports sur des erreurs que vous aurez constatées à l'adresse suivante : [Alain Matthes](mailto:Alain.Matthes@univ-lille.fr).

This file can be redistributed and/or modified under the terms of the LATEX Project Public License Distributed from [CTAN](http://www.ctan.org) archives.

Installation

Lorsque vous lirez ce document, il est possible que **tkz-base** soit présent sur le serveur du **CTAN**¹ alors **tlmgr** vous permettra de l'installer. Si **tkz-base** ne fait pas encore partie de votre distribution, cette section vous montre comment l'installer, elle est aussi nécessaire si vous avez envie d'installer une version beta ou personnalisée de **tkz-base**. Si le package est présent sur le serveur du **CTAN** et que vous n'utilisez pas **tlmgr**, je vous conseille de la télécharger à partir de ce serveur, sinon vous le trouverez sur mon site. Pour distinguer les anciennes versions de la nouvelle, j'ai repris la numérotation à 1.00 et j'ai ajouté « c »². Vous allez donc installer la version **1.121 c**.

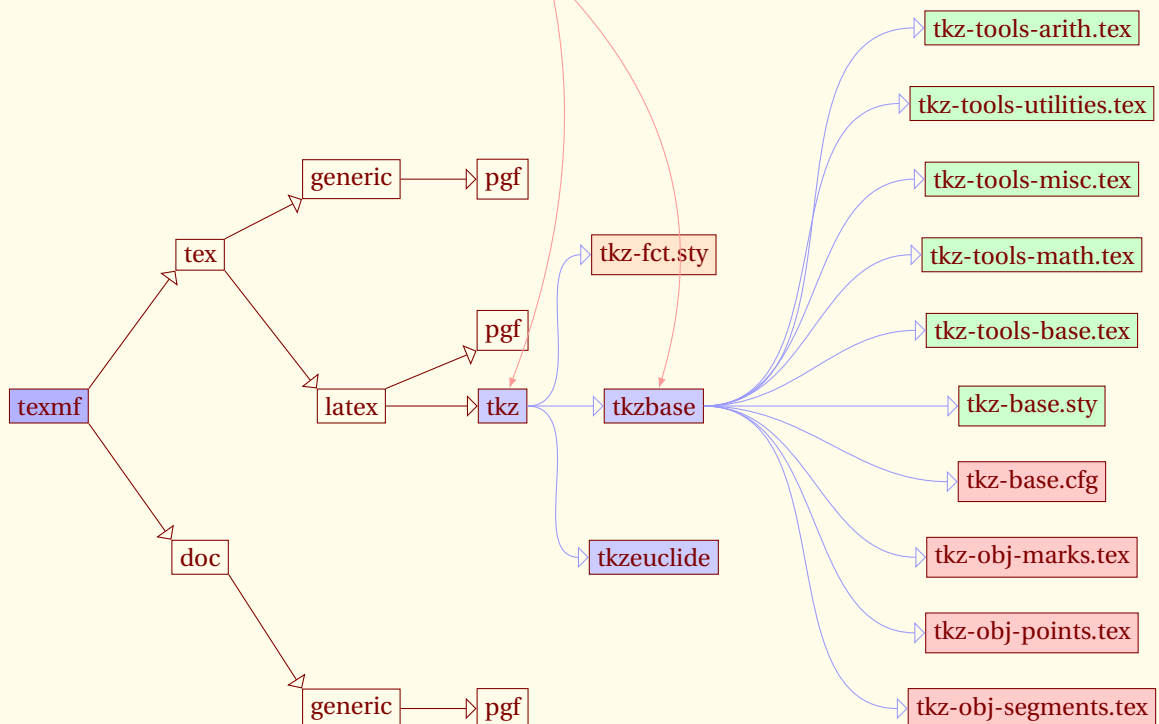
Le plus simple est de créer un dossier **tkz**³ avec comme chemin : **texmf/tex/latex/tkz**. Voici les chemins de ce dossier sur mes deux ordinateurs :

- sous OS X **/Users/ego/Library/texmf** ;
- sous Ubuntu **/home/ego/texmf**.

Je suppose que si vous mettez vos packages ailleurs, vous savez pourquoi!

L'installation que je propose n'est valable que pour un utilisateur.

1. Après l'avoir décompressé, placez le dossier **tkzbase** dans le dossier **tkz**.



1. **tkz-base** ne fait pas encore partie de **TeXLive**
2. pour CTAN
3. ou bien un autre nom

2. Ouvrir un terminal, puis faire `sudo texhash` si nécessaire.
3. Vérifier que **fp**, **numprint** et **tikz 2.10** sont installés car ils sont obligatoires, pour le bon fonctionnement de **tkz-base**.

Reamarque : Installation de tkz-base avec MikTeX sous Windows XP.

Je ne connais pas grand-chose à ce système, mais un utilisateur de mes packages **Wolfgang Buechel** a eu la gentillesse de me faire parvenir ce qui suit :

Pour ajouter **tkzbase** à MiKTeX⁴ :

- ajouter un dossier **tkz** dans le dossier `[MiKTeX-dir]/tex/latex`
- copier **tkzbase** et tous les fichiers présents dans le dossier **tkz**,
- mettre à jour MiKTeX, pour cela dans shell DOS lancer la commande `mktexlsr -u` ou bien encore, choisir `Start/Programs/Miktex/Settings/General` puis appuyer sur le bouton `Refresh FNDB`.

1.1 Fichiers installés

Avant de tester l'installation, vous pouvez vérifier que le dossier **tkzbase** contient les fichiers suivants :

- **tkz-base.cfg**
- **tkz-base.sty**
- **tkz-obj-marks.tex**
- **tkz-obj-points.tex**
- **tkz-obj-segments.tex**
- **tkz-tools-arith.tex**
- **tkz-tools-base.tex**
- **tkz-tools-math.tex**
- **tkz-tools-misc.tex**
- **tkz-tools-utilities.tex**

Celui qui est contient les principales macros est **tkz-tools-base.tex**, il est appelé par **tkz-base.sty** qui gère l'ensemble des fichiers. Les différents outils sont dans les fichiers commençant par **tkz-tools**, les objets mathématiques créés le sont dans des fichiers dont le nom a pour préfixe **tkz-obj**. Enfin **tkz-base.cfg** dont la présence n'est pas obligatoire permet de modifier beaucoup de valeurs par défaut.

Une remarque sur **tkz-tools-arith.tex** qui contient des fonctions mathématiques qui sont dans la version cvs de **TikZ**. J'en ai tenu compte, et logiquement cela doit fonctionner sans problème, enfin je l'espère.

De plus, **TikZ** est chargé avec les bibliothèques suivantes :

```
\usetikzlibrary{calc,
                arrows,
                plotmarks,
                positioning,
                shapes.misc,
                decorations,
                decorations.markings,
                decorations.pathreplacing,
                patterns}
```

4. Essai réalisé avec la version 2.7

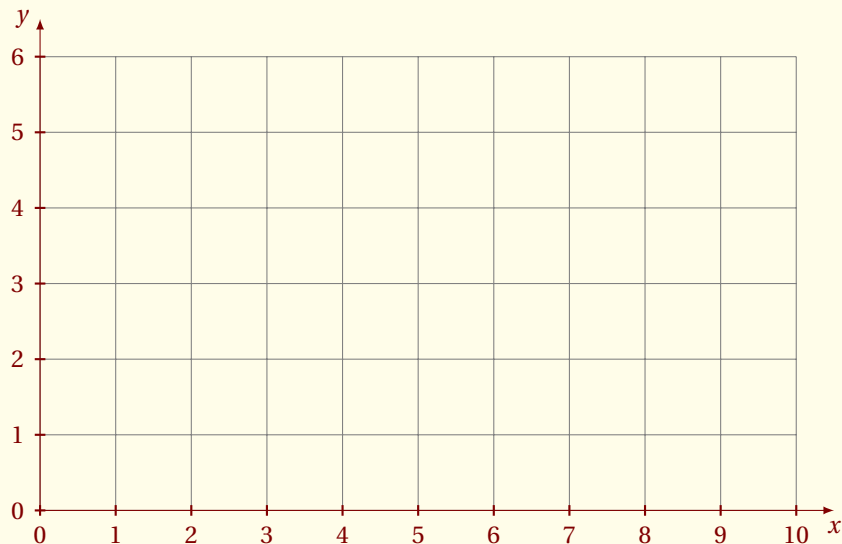
SECTION 2

Compilation des exemples

2.1 Test de l'installation

Le code ci-dessous permet de tester votre installation de **tkz-base**. Je vous rappelle que **fp.sty**, tout comme **numprint.sty** doit être présent ainsi que la version 2.10 de **pgf**.

```
\documentclass{article}
\usepackage{tkz-base}
\begin{document}
\begin{tikzpicture}
  \tkzInit[ymax=6]
  \tkzGrid
  \tkzAxeXY
\end{tikzpicture}
\end{document}
```



Remarques sur ce test

1. ☞ En principe, **tkz-base** n'est pas chargé par l'utilisateur, il sera chargé par un autre package comme **tkz-euclide** ou **tkz-fct** et **tkz-base** charge **numprint.sty** avec l'option **autolanguage**, **fp.sty**, **etex.sty** et bien sûr **TikZ**.
2. ☞ Vous remarquerez que **TikZ** est parfois allergique aux caractères actifs, aussi j'ai créé deux macros **\tkzActivOff** et **\tkzActivOn** pour désactiver et activer « :! ». Il semblerait que la version 2.1 de **pgf** est réglé certains problèmes liés aux caractères actifs.

```
\tkzActivoff
\begin{tikzpicture}
  \dots
\end{tikzpicture}
\tkzActivon
```

2.2 Test des exemples

Sur le site <http://altermundus.fr> et bientôt sur <http://altermundus.com>, vous trouverez des exemples. Ces exemples utilisent un préambule **tkz preamble.ltx** qui se trouve dans le dossier des exemples.

Son code est le suivant :

```

1 \documentclass{scrartcl}
2 \usepackage[T1]{fontenc}
3 \usepackage[utf8x]{inputenc}% utf8x
4 \usepackage{lmodern} % fourier
5 \usepackage{fullpage}
6 \usepackage{amsmath,amssymb,amsfonts}
7 % with fourier, only amsmath
8 \usepackage[usenames,dvipsnames,svgnames]{xcolor}
9 % before tikz or tkz
10 \usepackage{tkz-tab,tkz-euclide,tkz-fct}
11 \usetkzobj{all} % all the objects
12 % some colors
13 \definecolor{bistre}{rgb}{.75,.50,.30}
14 \definecolor{Maroon}{rgb}{0.5,0.0,0.0}
15 \definecolor{fondpaille}{cmymk}{0,0,0.1,0}
16 \pagecolor{fondpaille}
17 \color{Maroon}
18 \tkzSetUpColors[background=fondpaille,text=Maroon]
```

Un **Makefile** est aussi donné pour ceux qui veulent tout compiler avec une seule commande. Pour cela, décompressez le dossier exemple **base-ex.zip**, puis dans un terminal, placez-vous dans ce dossier et lancez la commande **make** :

```

$ cd path vers le dossier
$ make
```

*Remarque : Pour ceux qui ne souhaitent pas charger **tkz-tab**, **tkz-euclide** et **tkz-fct**, mais charger seulement **tkz-base**, il faudra faire attention à placer parfois la commande **\usetkzobj{polygons, lines, circles}** dans le préambule, après **tkz-base**, pour utiliser les objets correspondants.*

2.3 Pourquoi **fp.sty** et **numprint.sty**

Pour le moment, seul **fp.sty** permet de gérer des calculs sur des grands nombres ou des très petits avec précision. Cela ralentit la compilation, aussi il est préférable de ne pas en abuser. Ici c'est le cas, **fp.sty** est avant tout utilisé, pour obtenir des graduations correctes. Je vais essayer de faire une version qui permet de ne travailler qu'avec **pgfmath.sty** en se passant de **fp.sty**, mais surtout je vais faire une version pour **lua¹l²at³ex** et les calculs seront effectués par l'intermédiaire de **lua**.

numprint.sty était présent quand j'ai commencé à écrire cette série de packages, depuis **siunitx.sty** s'est développé et je peux comprendre que certains le préfèrent. Dans une prochaine version, j'ai prévu de laisser le choix du package pour l'affichage des nombres.

Les macros

Le package vous fournit les macros essentielles suivantes, qui sont données avec leurs principales options et valeurs par défaut :

1. Macros générales

- `\usetkzobj`{*all*} ou {*circles,lines,polygons,etc.*}
- `\tkzInit`[xmin=0,xmax=10,xstep=1,ymin=0,ymax=10,ystep=1]
- `\tkzGrid`[sub,color=darkgray,line width=.4pt]
- `\tkzClip`[space=1]
- `\tkzRep`[xlabel= \vec{i} ,ylabel= \vec{j}]
- `\tkzText`[color=black,text=black,fill=white] (*point*) (*un texte*)
- `\tkzLegend`[options]{*mark/couleur/size/texte*}

2. Macros sur les axes

- `\tkzAxeX`[label=*x*,color=black,trig,frac]
- `\tkzDrawX`[noticks,label=*x*]
- `\tkzLabelX`[trig,frac,label options={...},np off]
- `\tkzAxeY`[label=*y*,color=black]
- `\tkzDrawY`[noticks,label=*y*]
- `\tkzLabelY`[label options={...}]
- `\tkzAxeXY`[label={},color=black,frac]
- `\tkzDrawXY`[label={},color=black,frac]
- `\tkzLabelXY`[text=black,frac,trig]

3. Macros sur les points

- `\tkzDefPoint` (*x*, *y*) {*name*} ou bien (*a* : *r*) {*name*}
- `\tkzDefPoints` {*x*₁/*y*₁/*name*,*x*₂/*y*₂/*name*}
- `\tkzDefShiftPoint`[point] (*x*, *y* ou *a* : *r*) {*name*}
- `\tkzDrawPoint`[options] (*name*)
- `\tkzDrawPoints`[options] (*n*₁,*n*₂,...)
- `\tkzLabelPoint`[options] (*name*) {*label*}
- `\tkzLabelPoints` (*n*₁,*n*₂,...)
- `\tkzLabelPoints` (*n*₁,*n*₂,...)
- `\tkzPointShowCoord`[options] (*point*)

4. Macros sur les segments

- `\tkzDrawSegment`[options] (*name*,*name*)
- `\tkzDrawSegments`[options] (*p*₁, *p*₂ *p*₃, *p*₄)
- `\tkzDrawPolySeg`[options] (*p*₁, *p*₂, *p*₃, ... ,...)

- `\tkzLabelSegment`[options]($\langle name, name \rangle$){ $\langle label \rangle$ }
- `\tkzLabelSegments`[options]($\langle n1, n2 n3, n4 \dots \rangle$)
- `\tkzMarkSegment`[mark=none, pos=.5, size=4pt]($\langle name, name \rangle$)
- `\tkzMarkSegments`[options]($\langle n1, n2 n3, n4 \dots \rangle$)

5. Autres macros


- `\tkzHLine`[options]{ $\langle v \rangle$ }
- `\tkzHLines`[options]{ $\langle v_1, v_2, \dots \rangle$ }
- `\tkzVLine`[options]{ $\langle v \rangle$ }
- `\tkzVLines`[options]{ $\langle v_1, v_2, \dots \rangle$ }
- `\tkzHTick`[options]{ $\langle v \rangle$ }
- `\tkzHTicks`[options]{ $\langle v_1, v_2, \dots \rangle$ }
- `\tkzVTick`[options]{ $\langle v \rangle$ }
- `\tkzVTicks`[options]{ $\langle v_1, v_2, \dots \rangle$ }

SECTION 4

Présentation de **tkz-base****4.1 Exemple qui pose un problème**


Le code suivant donne une erreur

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

 **Latex Error : ... Dimension too large.**

En effet, l'unité par défaut est le cm or \TeX ne peut pas stocker une dimension supérieure à 575 cm, c'est ce qui entraîne une erreur. \TeX cependant, peut stocker des entiers allant jusqu'à $2^{31} - 1$, aussi il est possible de travailler en premier sur des entiers puis de définir les dimensions.

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

 **Latex Error : ... Dimension too large.**

Le code précédent donne encore une erreur. En effet, 600 cm est une dimension et ne tient pas compte du changement d'unité. Correct est :

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

Cette fois, la dimension stockée est 6 cm ce qui est acceptable. Il est possible avec \TeX de manipuler de grands nombres entiers, mais en revanche les dimensions ne peuvent excéder 16 384 pt soit 5,75 m environ.

Avec \TeX , il est aussi possible de travailler avec le package `fp.sty`, qui lui permet de travailler sur des intervalles plus importants, mais au prix d'une certaine lenteur. C'est la méthode que j'ai privilégiée pour certains calculs sensibles qui requiert une bonne précision comme des calculs de mesure d'angles ou de longueur de segment, mais il est nécessaire une fois un nombre trouvé, de l'attribuer à une dimension. On retrouve toujours les mêmes contraintes.

4.2 Le rôle de **tkz-base**

Le code suivant donne une erreur n'ont parce que 6 000 000 est un trop grand nombre, mais parce que 0,000 001 cm est une trop petite dimension.

 **Latex Error :**

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

Avec **tkz-base**, il sera possible de travailler avec des coordonnées quelconques, mais il faudra pour cela utiliser les macros du package.

```
\begin{tikzpicture}
  \tkzInit[xmax=10000000,xstep=1000000]
  \tkzDrawX
```

```
\tkzLabelX[label options={text = red,
                             below right = 6pt,
                             rotate = -45}]
\end{tikzpicture}
```

tkz-base permet de simplifier l'utilisation d'intervalles de valeurs divers. Ce package est utilisé par plusieurs de mes packages comme **tkz-tukey**, un package pour dessiner les représentations graphiques en statistiques élémentaires, **tkz-fct** qui permet de dessiner les représentations graphiques des fonctions à l'aide du logiciel **gnuplot**, ainsi qu'avec **tkz-euclide** pour la géométrie euclidienne.

Premièrement, il faut savoir qu'il n'est pas nécessaire de s'occuper avec **TikZ** de la taille du support (bounding box), cependant il est parfois nécessaire, soit de tracer une grille, soit de tracer des axes, soit de travailler avec une unité différente que le centimètre, soit finalement de contrôler la taille de ce qui sera affiché. Pour cela, il faut avoir préparé le repère dans lequel vous allez travailler, c'est le rôle de **tkz-base** et de sa macro principale **\tkzInit**. Par exemple, si l'on veut travailler sur un carré de 10 cm de côté, mais tel que l'unité soit le dm alors il faudra utiliser.

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

xstep=0.1 signifie que 1cm représente la graduation 0.1 ainsi la graduation 1 se trouve à 10 cm de l'origine.

En revanche pour des valeurs de x comprises entre 0 et 10 000 et des valeurs de y comprises entre 0 et 100 000, il faudra écrire

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

Le résultat est toujours un carré de 10 cm de côté.

Tout cela a peu de sens pour faire de la géométrie euclidienne, et dans ce cas, il est recommandé de laisser l'unité graphique égale à 1 cm. Je n'ai d'ailleurs pas testé si toutes les macros destinées à la géométrie euclidienne acceptaient d'autres valeurs que **xstep=1** et **ystep=1**. En revanche pour certains dessins, il est intéressant de fixer les valeurs extrêmes et de « clipper » le rectangle de définition afin de contrôler au mieux la taille de la figure.

4.3 Syntaxe de **tkz-base**

J'ai essayé de généraliser la syntaxe suivante :

- la syntaxe est proche de celle de \LaTeX , pas besoin « ; » ;
- toutes les macros ont un nom commençant par **tkz** ;
- les accolades sont utilisées pour passer un paramètre qui sera la référence d'un objet créé par la macro ;
- les parenthèses sont utilisées pour faire référence à un objet déjà créé ou bien pour un couple de coordonnées ;
- les crochets sont nécessaires pour faire passer des arguments optionnels ou bien encore des options, certains choix sont parfois obligatoires. L'emploi de la virgule même dans un mode Math nécessite d'être protégé dans un groupe TeX ;
- les blancs (espace) sont interdits entre [...] et (...), [...] et {...}, ainsi qu'entre (...) et {...} mais il est possible de mettre des espaces entre les arguments optionnels passés [...].

SECTION 5

Initialisation \tkzInit

5.1 La macro principale \tkzInit

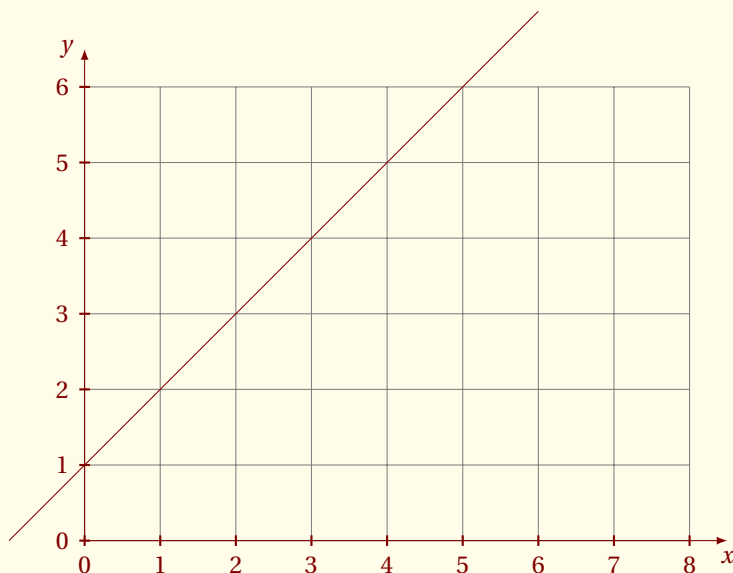
`\tkzInit[local options]`

options	défaut	définition
xmin	0	valeur minimum des abscisses en cm
xmax	10	valeur maximum des abscisses en cm
xstep	1	différence entre deux graduations en x
ymin	0	valeur minimum des ordonnées en cm
ymax	10	valeur maximum des ordonnées en cm
ystep	1	différence entre deux graduations en y

Le rôle de **tkzInit** est de définir un repère *orthogonal* et une partie rectangulaire du plan dans laquelle vous aller placer vos dessins à l'aide de coordonnées cartésiennes. Le repère n'est pas obligatoirement normé. Cette macro permet de définir votre environnement de travail comme avec une calculatrice.

5.2 Modification de la taille du dessin avec \tkzInit

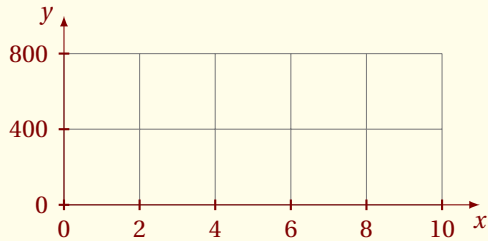
Cette macro prépare le terrain et définit plusieurs constantes. Il est tout à fait possible de faire une figure plus grande que le rectangle prédéfini. De plus, comme vous pouvez le constater, il est possible d'utiliser les commandes de **TikZ** au milieu de celles de **tkz**.



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

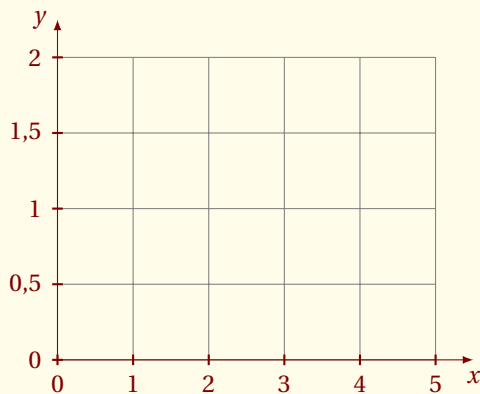
5.3 Rôle de `xstep`, `ystep`

Attention, une graduation est représentée par 1 cm, sauf si vous redimensionner la figure avec l'option **scale**. Dans l'exemple ci-dessous `xstep = 2` correspond à 1 cm, donc entre 0 et 10, il nous faudra 5 cm. De même `ystep=400`, il y a donc 2 cm entre 0 et 800. Il n'est pas possible d'utiliser les options de **TikZ**, `x=...` et `y=...`.



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

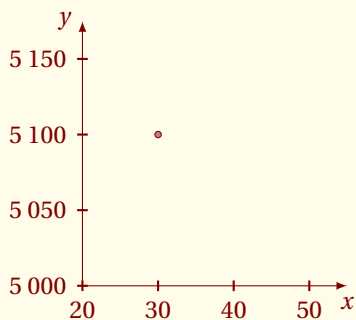
5.4 Autre exemple avec `xstep` et `ystep`



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

5.5 Origine personnalisée.

Il est important de remarquer que l'on peut placer un point sans rien calculer.

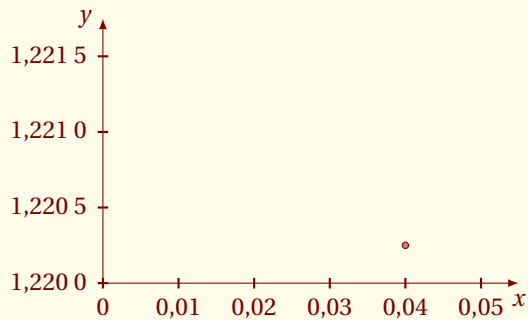


```
\begin{tikzpicture}
  \tkzInit[xmin=20,
           xmax=50,
           xstep=10,
           ymin=5000,
           ymax=5150,
           ystep=50]
  \tkzAxeXY
  \tkzDefPoint(30,5100){A}
  \tkzDrawPoint(A)
\end{tikzpicture}
```

5.6 Utilisation des décimaux

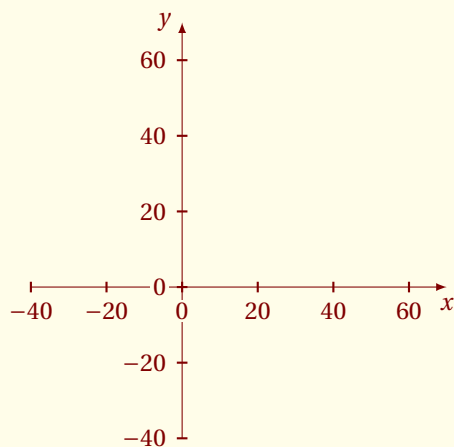
Il est préférable d'écrire les différents arguments relatifs à un axe avec le même nombre de décimales.

`numprint.sty` est utilisé pour afficher les graduations correctement. Dans l'exemple suivant, `numprint.sty` utilise les conventions françaises pour l'écriture des nombres car j'ai utilisé : `\usepackage[frenchb]{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}
```

5.7 Valeurs négatives



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

SECTION 6

Macros pour les axes

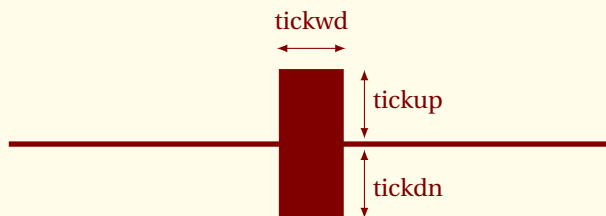
Je viens d'ajouter des nouvelles macros qui respectent davantage l'esprit dans lequel sont construits les nouveaux packages. Il s'agit de pouvoir utiliser les options de **TikZ**. Ces macros remplacent `\tkzX` et `\tkzY`. Ainsi pour tracer l'axe des abscisses, on peut utiliser `\tkzDrawX`, pour placer des graduations `\tkzLabelX` et enfin dans les cas simples, il est possible de n'utiliser que `\tkzAxeX`. La syntaxe est plus homogène et on peut utiliser les options de **TikZ**. Pour les graduations, il est possible d'utiliser des fractions.

6.1 `\tkzDrawX`

`\tkzDrawX[local options]`

Cette macro permet de tracer l'axe des abscisses avec des ticks par défaut. Les options sont celles de **TikZ** plus les suivantes :

options	défaut	définition
color	black	couleur de l'axe et des ticks
noticks	false	pas de ticks sur l'axe
right space	0,5 cm	prolongement de l'axe à droite
left space	0 cm	prolongement de l'axe à gauche
label	x	nom attribué au label
trig	0	si $\langle 0 \pi / \text{trig}$ est l'unité
tickwd	0.8pt	épaisseur du tick
tickup	1pt	hauteur du tick au dessus de l'axe
tickdn	1pt	profondeur du tick en dessous de l'axe



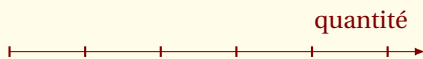
Cette macro permet de tracer l'axe des abscisses. Le plus important est de tester l'ensemble des options. Ci-dessus, vous avez les valeurs qui définissent un tick. Sinon les options de **TikZ** s'appliquent et en particulier **text**, **color**, **fill** et **font**.

6.1.1 Sans tick, ni label



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

6.1.2 Placement du label

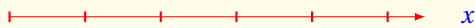


```
\begin{tikzpicture}
\tkzInit[xmax=5]
\tkzDrawX[label      = quantité,
           above left = 8pt]
\end{tikzpicture}
```

6.1.3 Couleur du label et de l'axe

La couleur du label est obtenue avec l'option **text**, celle de l'axe avec l'option **color**.

L'option **right=12pt** décale le label x de 12 pt.



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

6.1.4 option right space

Cela ajoute un peu d'espace après le dernier tick.



```
\begin{tikzpicture}
\tkzInit[xmax=0.5,xstep=0.1]
\tkzDrawX[text=blue,color=red,
           right=12pt,right space=1]
\end{tikzpicture}
```

6.1.5 Axe trigonométrique avec l'option trig=1

Si $number = 0$ alors l'axe est gradué de cm en cm, sinon l'axe est gradué à l'aide des multiples de $\frac{\pi}{number}$



```
\begin{tikzpicture}
\tkzInit[xmin=0,xmax=7,ymin=-1,ymax=1]
\tkzDrawX[trig=1]
\end{tikzpicture}
```

6.1.6 Axe trigonométrique avec l'option trig=2



```
\begin{tikzpicture}
\tkzInit[xmin=0,xmax=7,ymin=-1,ymax=1]
\tkzDrawX[trig=2]
\end{tikzpicture}
```

6.2 \tkzLabelX

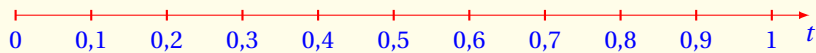
`\tkzLabelX[local options]`

Cette macro permet de placer des graduations. L'option **orig** peut de nouveau être utilisée, mais son comportement est inversée. Par défaut, la valeur à l'origine est placée. Les options sont celles de **TikZ**, plus les suivantes :

options	défaut	définition
frac	0	si $\langle > 0$ graduations = num/frac "frac est un entier"
trig	0	si $\langle > 0$ pi/trig "trig est un entier"
font	\textstyle	taille de la graduation.
label options	empty	option de position des graduations
color	black	couleur des graduations
step	1	intervalle entre deux graduations
np off	false	désactivation de numprint
orig	true	affiche la graduation de l'origine

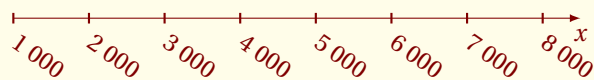
frac et trig sont des nombres entiers permettant de passer à une écriture fractionnaire ou trigonométrique.

6.2.1 Position des graduations avec label options



```
\begin{tikzpicture}
\tkzInit[xmax=1,xstep=0.1]
\tkzDrawX[label=$t$,text=blue,color=red]
\tkzLabelX[label options={text=blue,below = 3pt}]
\end{tikzpicture}
```

6.2.2 Position des graduations avec label options



```
\begin{tikzpicture}
\tkzInit[xmin=1000,xmax=8000,xstep=1000]
\tkzDrawX
\tkzLabelX[label options={below right=3 pt,inner sep = 1pt,rotate=-35}]
\end{tikzpicture}
```

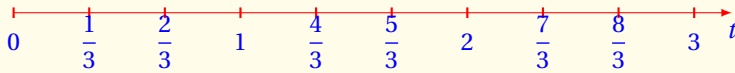
6.2.3 Dates avec np off

Pour les dates, il faut désactiver numprint.



```
\begin{tikzpicture}
\tkzInit[xmin=2000,xmax=2010]
\tkzDrawX
\tkzLabelX[np off,label options={below right=3 pt,inner sep =1pt,rotate=-35}]
\end{tikzpicture}
```

6.2.4 frac



```
\begin{tikzpicture}
\tkzInit[xmax=3,xstep=0.33333]
\tkzDrawX[label=$t$,text=blue,color=red]
\tkzLabelX[frac=3,text=blue,below = 9pt]
\end{tikzpicture}
```

6.2.5 trig



```
\begin{tikzpicture}
\tkzInit[xmin=0,xmax=7,ymin=-1,ymax=1]
\tkzDrawX[trig=2]
\tkzLabelX[trig=2]
\end{tikzpicture}
```

6.2.6 Taille des graduations

Deux possibilités. Il est possible de définir le style employé par défaut pour le mode math, il s'agit de `\tkzmathstyle` qui équivaut à `\textstyle`. Il est possible de faire

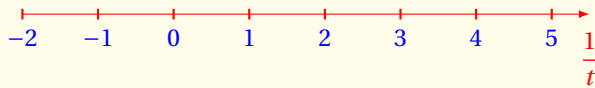
```
\let\tkzmathstyle\textstyle
```



```
\begin{tikzpicture}
\tkzInit[xmin=0,xmax=7,ymin=-1,ymax=1]
\tkzDrawX[trig=2]
\tkzLabelX[trig=2,below=8pt]
\end{tikzpicture}
```

6.2.7 Couleur des graduations

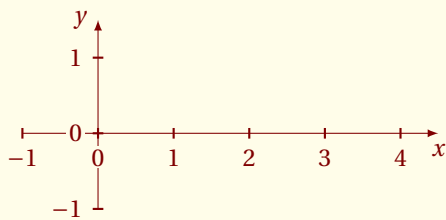
Il s'agit ici de bien utiliser les options `color`, `text` et `fill`



```
\begin{tikzpicture}
  \tkzInit[xmin = -2,xmax = 5,
    ymin = -2,ymax = 5]
  \tkzDrawX[color = red,
    label =  $\displaystyle\frac{1}{t}$ ,
    below = 6pt]
  \tkzLabelX[color=red,text=blue]
\end{tikzpicture}
```

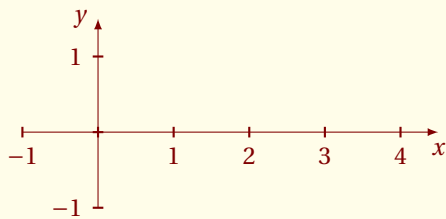
6.2.8 Tracés des axes avant la graduation

Dans certains cas, il est préférable de placer `\tkzDrawXY` après `\tkzLabelX` et `\tkzLabelY`. Cela permet d'éviter des problèmes d'affichage.



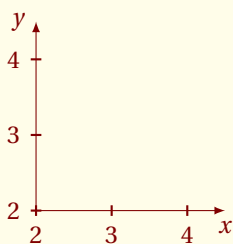
```
\begin{tikzpicture}
  \tkzInit[xmin = -1,xmax = 4,
    ymin = -1,ymax = 1]
  \tkzDrawXY \tkzLabelX \tkzLabelY
\end{tikzpicture}
```

6.2.9 Graduations (exceptées à l'origine) avant les tracés



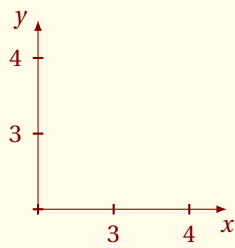
```
\begin{tikzpicture}
  \tkzInit[xmin = -1,xmax = 4,
    ymin = -1,ymax = 1]
  \tkzLabelX[orig=false] \tkzLabelY[orig=false]
  \tkzDrawXY
\end{tikzpicture}
```

6.2.10 Graduations uniquement positives avant les tracés



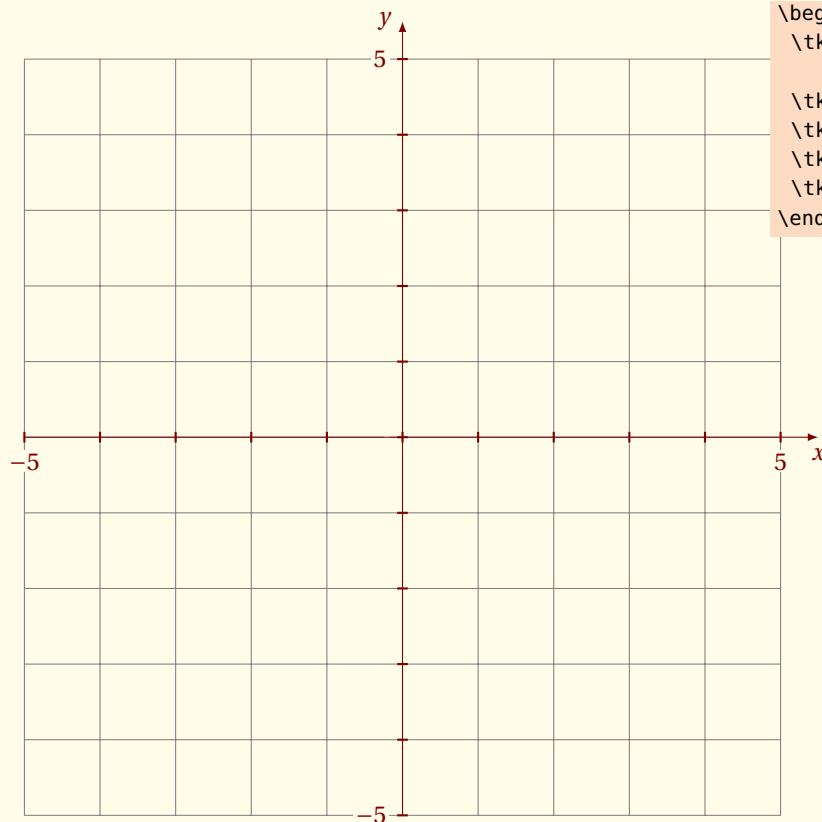
```
\begin{tikzpicture}
  \tkzInit[xmin=2,ymin=2,xmax=4,ymax=4]
  \tkzLabelX \tkzLabelY
  \tkzDrawXY
\end{tikzpicture}
```

6.2.11 Pas de graduations à l'origine



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

6.2.12 Graduations quelconques (exceptées à l'origine)



```
\begin{tikzpicture}
  \tkzInit[xmin = -5,xmax = 5,
           ymin = -5,ymax = 5]
  \tkzGrid
  \tkzLabelX[orig=false,step=5]
  \tkzLabelY[orig=false,step=5]
  \tkzDrawXY
\end{tikzpicture}
```

6.3 \tkzDrawY**\tkzDrawY[*local options*]**

Cette macro permet de tracer l'axe des ordonnées avec des ticks par défaut. Les options sont celles de **TikZ** plus les suivantes :

options	défaut	définition
color	black	couleur de l'axe et des ticks
noticks	false	pas de ticks sur l'axe
up space	0,5 cm	prolongement de l'axe en haut
down space	0 cm	prolongement de l'axe en bas
label	<i>x</i>	nom attribué au label
trig	0	si $\langle \rangle 0$ π/trig est l'unité
tickwd	0.8pt	épaisseur du tick
ticklt	1pt	hauteur du tick au dessus de l'axe
tickrt	1pt	profondeur du tick en dessus de l'axe

6.4 \tkzLabelY**\tkzLabelY[*local options*]**

Cette macro permet de tracer l'axe des abscisses avec des ticks par défaut. Les options sont celles de **TikZ** plus les suivantes :

options	défaut	définition
color	black	couleur des graduations
frac	0	si $\langle \rangle 0$ les graduations sont des fractions dénominateur=frac
font	\textstyle	taille de la graduation.
step	1	intervalle entre deux graduations

frac et **trig** sont des nombres entiers permettant de passer à une écriture fractionnaire ou trigonométrique.

6.5 \tkzAxeX

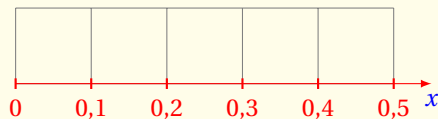
`\tkzAxeX[(local options)]`

Cette macro permet de tracer l'axe des abscisses avec des ticks par défaut ainsi que les graduations. Elle combine les deux macros `\tkzDrawX` et `\tkzLabelX`. Elle doit être utilisée dans les cas simples. Il faut éviter la rotation des labels pour la graduation.

options	défaut	définition
label	x	nom attribué au label
trig	0	graduation fraction de π
frac	0	graduation fractionnaire, de dénominateur « frac »
label options	{}	positionnement des graduations
orig	true	affichage de la graduation à l'origine
swap	false	permet de lancer <code>\tkzLabelX</code> avant <code>\tkzDrawX</code>

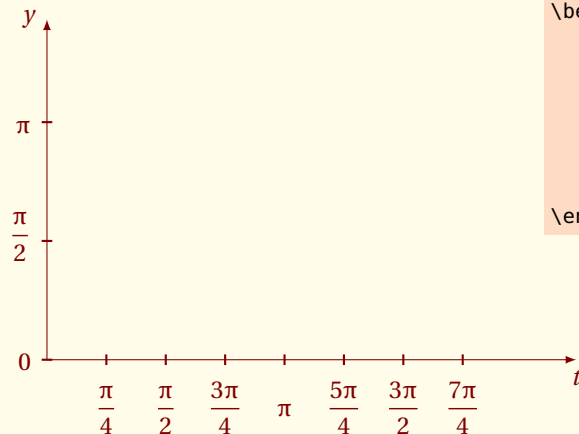
L'option **color** définit la couleur de l'axe alors que **text** définit la couleur des graduations, également possible est l'usage de **font**

6.5.1 exemple avec \tkzAxeX



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

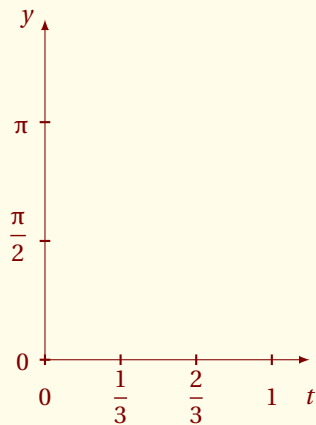
6.5.2 Usage de pi et \tkzAxeX



```
\begin{tikzpicture}
  \tkzInit[xmax=6.5,ymax=4]
  \let\tkzmathstyle\displaystyle
  \tkzAxeX[label = $$,orig = false,
    trig = 4,
    label options={below = 10pt}]
  \tkzAxeY[trig=2]
\end{tikzpicture}
```

6.5.3 Option frac et trig

Dans cet exemple on positionne le label t ainsi que les graduations. `\label options={below=6pt}` sert à placer les graduations



```
\begin{tikzpicture}
\tkzInit[xmax=9,xstep=3,ymax=4]
\tkzAxeX[label=$t$,below=10pt,orig=false,
frac=3,label options={below=6pt}]
\tkzAxeY[trig=2]
\end{tikzpicture}
```

6.6 \tkzAxeY

`\tkzAxeY[local options]`

Cette macro combine les deux macros : `\tkzDrawY` `\tkzLabelY` Voir `\tkzAxeX` pour les options

6.7 \tkzAxeXY

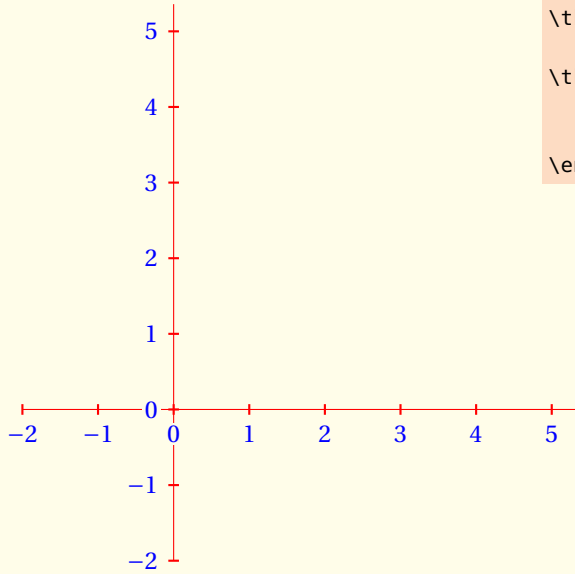
`\tkzAxeXY[local options]`

Cette macro combine les quatre macros : `\tkzDrawX` `\tkzDrawY` `\tkzLabelX` `\tkzLabelY`

*Il est nécessaire d'utiliser des options communes comme dans l'exemple ci-dessous, mais cela signifie que les mêmes options sont appliquées aux deux macros. Ainsi il n'est pas possible de modifier **label***

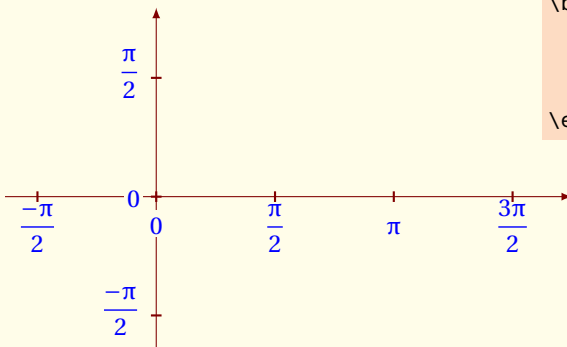
6.7.1 Couleur des axes, des graduations

Attention ici **fill=fondpaille** est obligatoire sinon le fond est rouge.



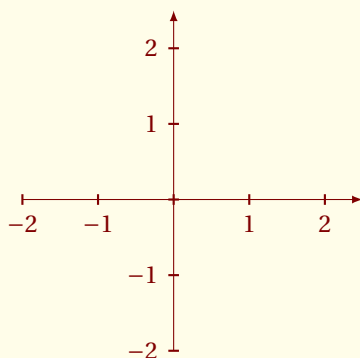
```
\begin{tikzpicture}
\tkzInit[xmin = -2,xmax = 5,
         ymin = -2,ymax = 5]
\tkzAxeXY[label={},color=red,
          text=blue,
          fill=fondpaille]
\end{tikzpicture}
```

6.7.2 Option {label=}



```
\begin{tikzpicture}
\tkzInit[xmin = -2,xmax = 5,
         ymin = -2,ymax = 2]
\tkzAxeXY[label={},text=blue,trig=2]
\end{tikzpicture}
```

6.7.3 Option orig

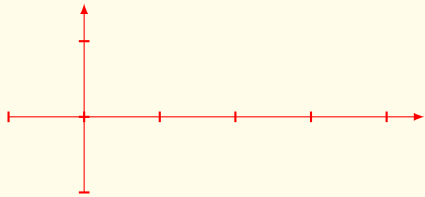


```
\begin{tikzpicture}
\tkzInit[xmin = -2,xmax = 2,
         ymin = -2,ymax = 2]
\tkzAxeXY[orig=false,label={},swap]
\end{tikzpicture}
```

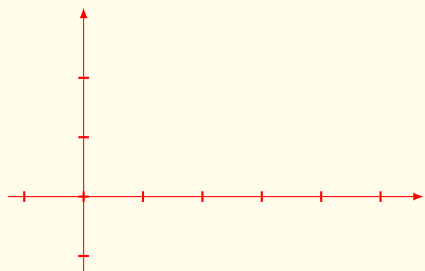
6.8 \tkzDrawXY**\tkzDrawXY[*local options*]**

Cette macro combine les deux macros : **\tkzDrawX**\tkzDrawY

Il est nécessaire d'utiliser des options communes comme dans l'exemple ci-dessous

6.8.1 Couleur commune et labels vides

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

6.8.2 Deux axes trigonométriques

```
\begin{tikzpicture}
  \tkzInit[xmin = -1,xmax = 4,
           ymin = -1,ymax = 2]
  \tkzDrawXY[label={},color=red,trig=4]
\end{tikzpicture}
```

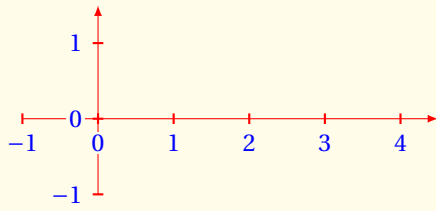
6.9 \tkzLabelXY**\tkzLabelXY[*local options*]**

Cette macro combine les deux macros :

\tkzLabelX\tkzLabelY

Il est nécessaire d'utiliser des options communes comme dans l'exemple ci-dessous

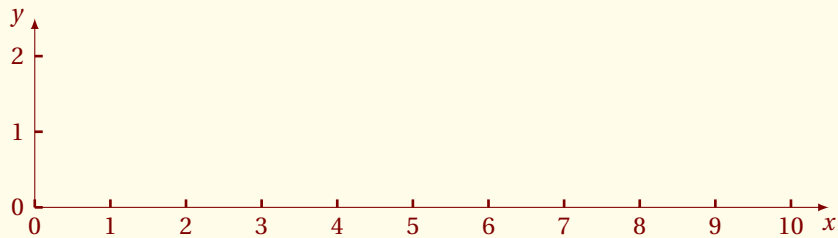
6.9.1



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

6.10 Modifier les valeurs par des défauts des axes`\tkzSetUpAxis[⟨local options⟩]`

options	défaut	définition
line width	0.4pt	line width définit la largeur du trait
tickwd	0.8pt	épaisseur du tick
ticka	1pt	partie droite ou au dessus du tick
tickb	1pt	partie gauche ou en dessous du tick
font	<code>\textstyle</code>	taille de la graduation.

6.11 Modification des axes par défaut

```

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

```

Il faut lancer de nouveau `\tkzSetUpAxis` pour récupérer les valeurs par défaut.

```

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

```

SECTION 7

Utilisation de `\tkzGrid`

```
\tkzGrid[local options]( $\langle x_A ; y_A \rangle$ ) ( $\langle x_B ; y_B \rangle$ )
```

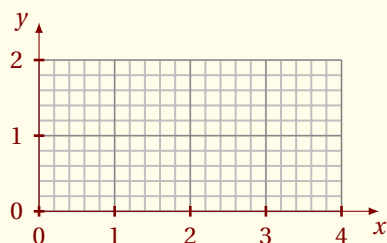
Quelques changements pour cette macro. Tout d'abord, pour simplifier actuellement la couleur de la grille la plus fine est déterminée automatiquement à partir de la grille principale, même processus pour l'épaisseur. Ce comportement pourra être modifié à l'aide de styles.

options	défaut	définition
$\langle x_A ; y_A \rangle$ ($\langle x_B ; y_B \rangle$)	(xmin,ymin)(xmax,ymax)	trace une grille
options	défaut	définition
sub	true	demande une sous grille
color	darkgray	couleur de la grille principale
subxstep	0.2	le pas des sous-graduations pour l'axe des abscisses
subystep	0.2	le pas des sous-graduations pour l'axe des ordonnées
line width	0.4pt	épaisseur des traits de la grille principale

Les valeurs par défaut peuvent être changées dans le fichier de configuration ou encore par des macros. La couleur de la seconde grille est celle de la grille principale, mais moins intense. Même comportement pour l'épaisseur du trait. Voir les exemples pour modifier ce comportement.

7.1 `\tkzGrid` et l'option **sub**

L'option **sub** permet d'afficher une grille secondaire plus fine. ☞ Il est préférable de lancer `\tkzGrid` en premier, pour éviter que la grille se superpose à d'autres éléments.

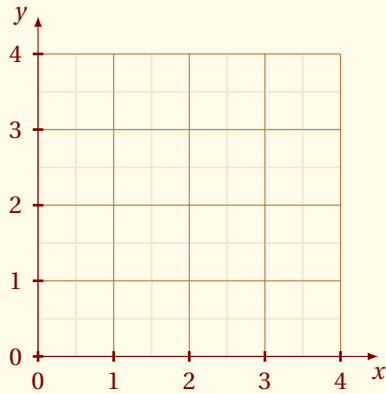


```
\begin{tikzpicture}
  \tkzInit[xmax=4, ymax=2]
  \tkzGrid[sub]
  \tkzAxeXY
\end{tikzpicture}
```

7.2 Option sub

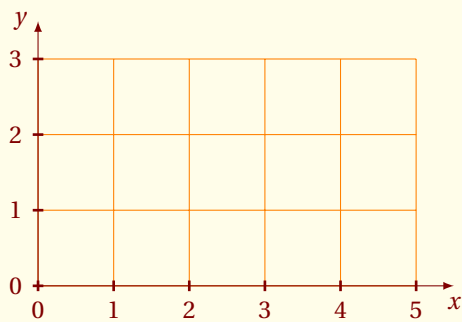
L'option `sub` permet d'afficher un grille secondaire plus fine avec comme paramètres

```
\definecolor{bistre}{rgb}{.75,.50,.30} % on définit une couleur
\providecolor{bistre}{rgb}{.75,.50,.30}
\def\tkzCoeffSubColor{50} % 50 % de la couleur principale
\def\tkzCoeffSubLw{0.6} % 60 % de l'épaisseur du trait
```



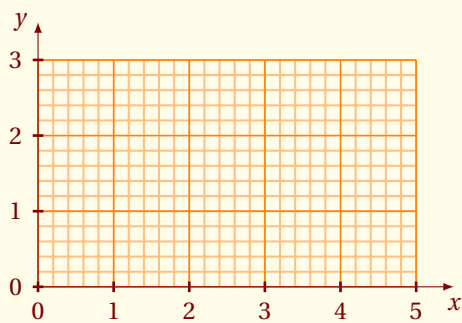
```
\def\tkzCoeffSubColor{20}
\def\tkzCoeffSubLw{0.2}
\begin{tikzpicture}
\tkzInit[xmax=4, ymax=4]
% on peut modifier le pas pour la seconde grille
\tkzGrid[sub,color=bistre,
subxstep=.5,substep=.5]
\tkzAxeXY
\end{tikzpicture}
```

7.3 Presque par défaut

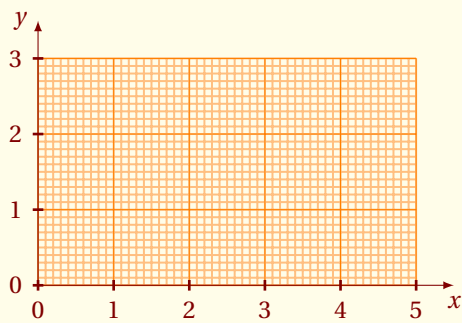


```
\begin{tikzpicture}
\tkzInit[xmax=5,ymax=3]
\tkzGrid[color=orange]
\tkzAxeXY
\end{tikzpicture}
```

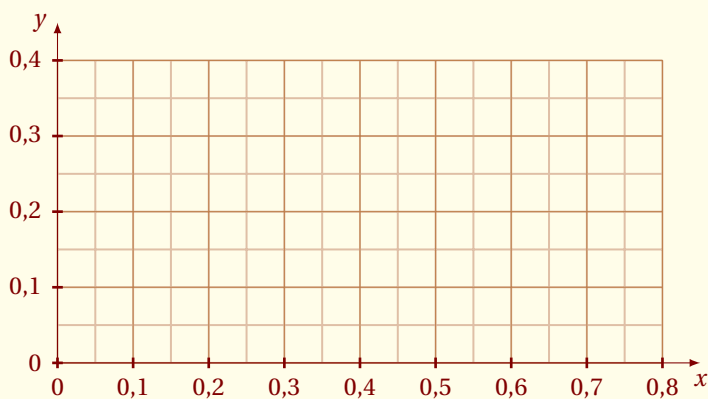
7.4 Sous grille en plus, option sub



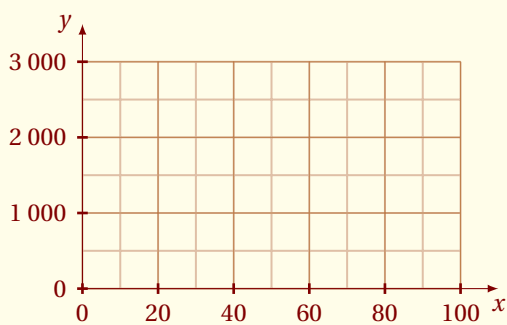
```
\begin{tikzpicture}
\tkzInit[xmax=5,ymax=3]
\tkzGrid[sub,color=orange]
\tkzGrid[color=orange]
\tkzAxeXY
\end{tikzpicture}
```

7.5 Changement de maille

```
\begin{tikzpicture}
  \tkzInit[xmax=5,ymax=3]
  \tkzGrid[color = orange,
    sub,
    subxstep = 0.1,
    subystep = 0.1]
  \tkzAxeXY
\end{tikzpicture}
```

7.6 Option xstep, xstep, subxstep et subystep

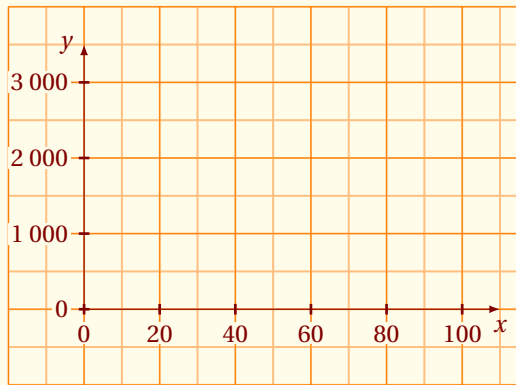
```
\begin{tikzpicture}
  \tkzInit[xmax=.8,
    xstep=.1,
    ymax=.4,
    ystep=.1]
  \tkzGrid[sub,
    subxstep = 0.05,
    subystep = 0.05,
    color=bistre]
  \tkzAxeXY
\end{tikzpicture}
```

7.7 Avec des intervalles importants

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

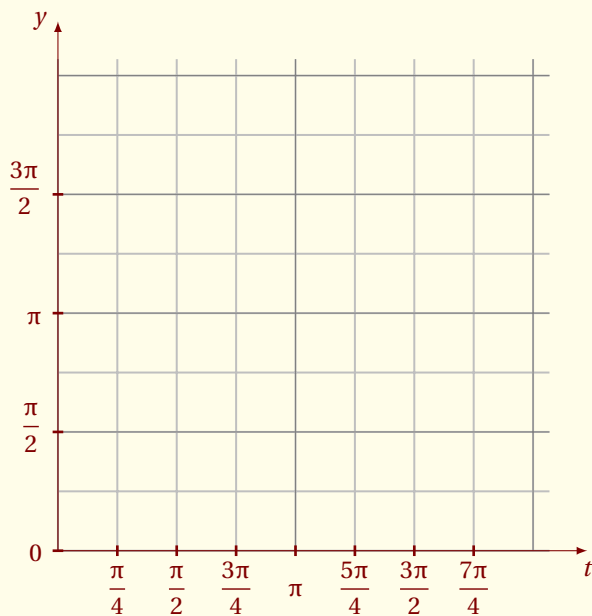
7.8 \tkzGrid et les arguments

La grille peut avoir une taille quelconque.



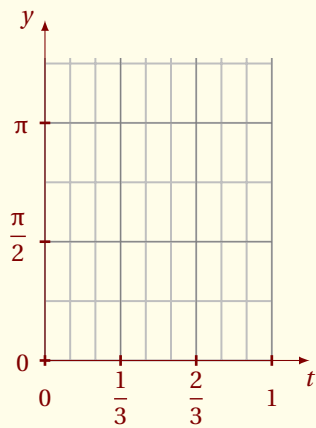
```
\begin{tikzpicture}
  \tkzInit[xmax=100,xstep=20,
           ymax=3000,ystep=1000]
  \tkzGrid[sub,subxstep=10,
           subystep=500,
           color=orange]
  \tkzAxeXY
\end{tikzpicture}
```

7.8.1 Usage de pi avec \tkzGrid



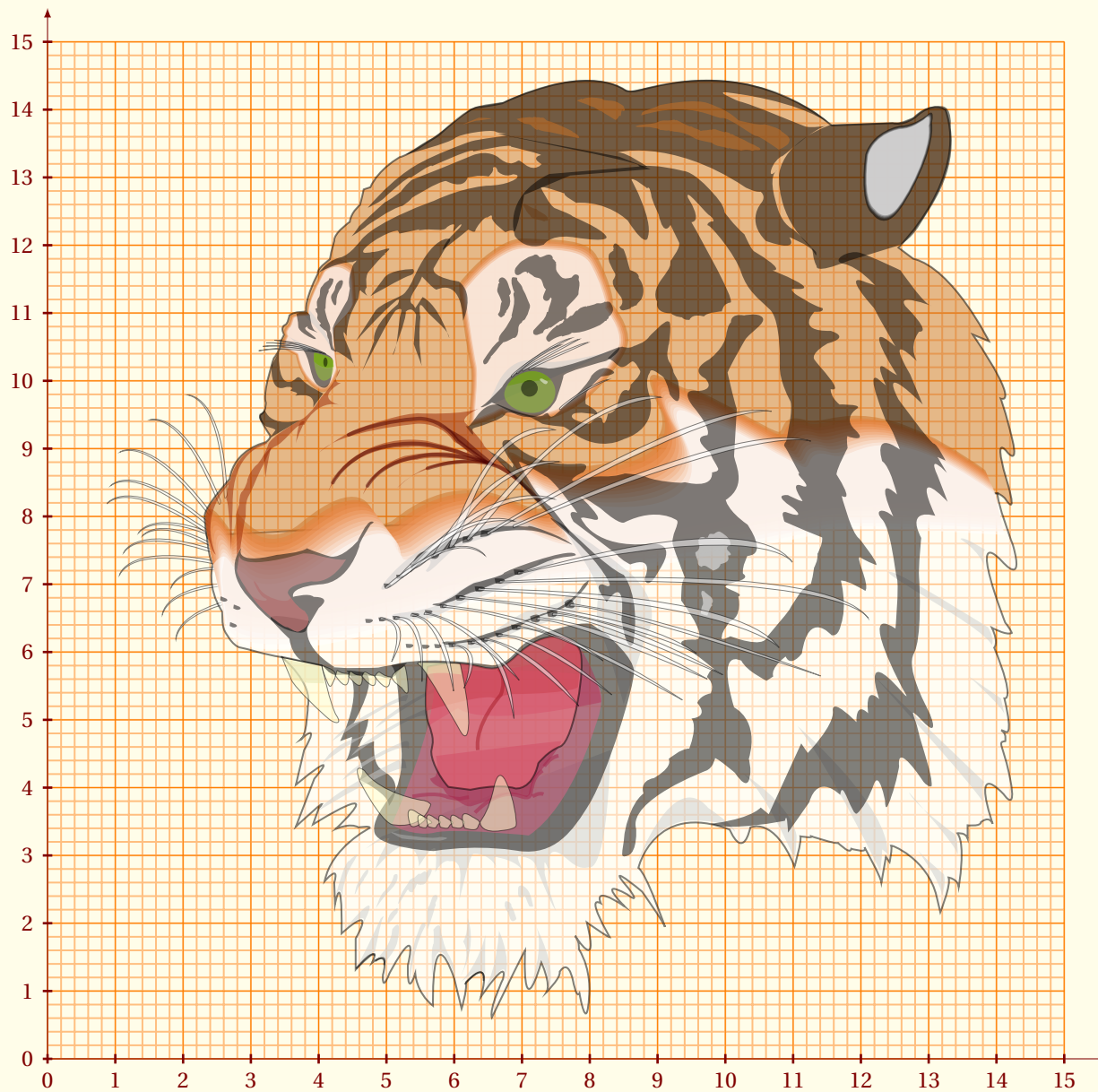
```
\begin{tikzpicture}
  \tkzInit[xmax=6.5,ymax=6.5]
  \tkzGrid[xstep=pi,ystep=pi/2,sub,
           subxstep=pi/4,subystep=pi/4]
  \tkzAxeX[label=$t$,orig=false,trig=4,
           label options={below=6pt}]
  \tkzAxeY[trig=2]
\end{tikzpicture}
```

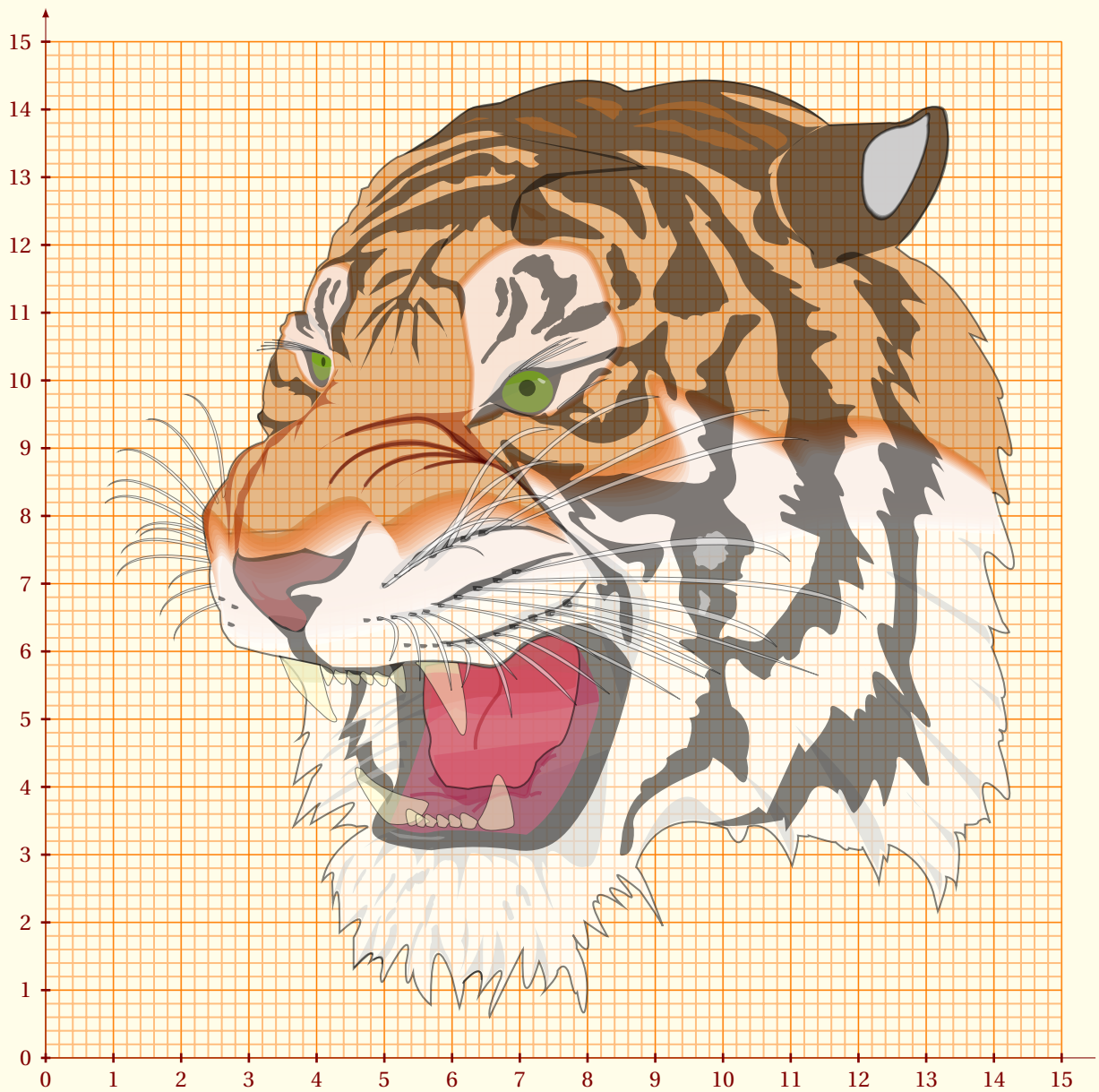
7.8.2 Options `frac` et `trig` avec `\tkzGrid`



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

7.8.3 Utilisation d'une grille de repérage





```
\begin{tikzpicture}
\tikzset{xaxe style/.style ={-}}
\tkzInit[xmax=15,ymax=15] \tkzGrid[sub,color=orange] \tkzAxeXY[label=]
\node[opacity=.5] at (8,6){\includegraphics[scale=.7]{tiger}};
\end{tikzpicture}
```

SECTION 8

Clipper une partie du plan

`\tkzClip[local options]`

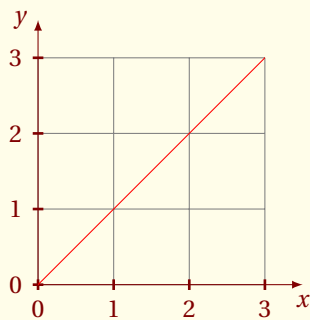
Le rôle de cette macro est de rendre invisible ce qui est hors du rectangle défini par $(x_{min}; y_{min})$ et $(x_{max}; y_{max})$.

options	défaut	définition
space	1	valeur ajoutée à droite, à gauche, en bas et en haut du background

Le rôle de l'option **space** est d'agrandir la partie visible du dessin. Cette partie devient le rectangle défini par $(x_{min} - space; y_{min} - space)$ et $(x_{max} + space; y_{max} + space)$. **space** peut être négatif! L'unité est le cm et ne doit pas être indiquée.

8.1 `\tkzClip`

Le rôle de cette macro est de « clipper » le rectangle initial afin que ne soient affichés que les tracés contenus dans ce rectangle.

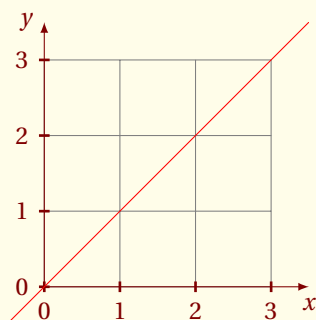


```
\begin{tikzpicture}
\tkzInit[xmax=3, ymax=3]
\tkzGrid
\tkzAxeXY
\tkzClip
\draw[red] (-1,-1)--(5,5);
\end{tikzpicture}
```

Il est possible d'ajouter un peu d'espace `\tkzClip[space]`

8.2 `\tkzClip` et l'option **space**

Les dimensions pour définir le rectangle clippé sont **xmin-1**, **ymin-1**, **xmax+1** et **ymax+1**.



```
\begin{tikzpicture}
\tkzInit[xmax=3, ymax=3]
\tkzGrid \tkzAxeXY
\tkzClip[space=.5]
\draw[red] (-0.5,-0.5)--(3.5,3.5);
\end{tikzpicture}
```

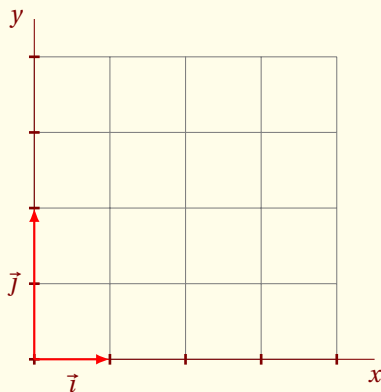
SECTION 9

Utilisation d'un repère

9.1 Repère avec `\tkzRep``\tkzRep[⟨local options⟩]`

options	défaut	définition
<code>line width</code>	0.8pt	line width définit la largeur du trait
<code>xlabel</code>	\vec{i}	étiquette pour l'axe des abscisses
<code>ylabel</code>	\vec{j}	étiquette pour l'axe des ordonnées
<code>posxlabel</code>	below=2pt	Position de l'étiquette
<code>posylabel</code>	left=2pt	Position de l'étiquette
<code>xnorm</code>	1	norme du vecteur en x
<code>ynorm</code>	1	norme du vecteur en y
<code>color</code>	black	couleur des traits
<code>colorlabel</code>	black	couleur des étiquettes

9.2 Exemple d'utilisation



```

\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}

```

☞ Pour ceux qui utilisent **frenchb** avec **babel**, en cas de problème vous pouvez utiliser les commandes suivantes `\tkzActivoff` et `\tkzActivon`. **TikZ** a été en effet parfois allergique aux caractères actifs, si le besoin se fait sentir, vous pouvez encadrer l'environnement **tikzpicture** ainsi :

```

\tkzActivoff
\begin{tikzpicture}
  \dots
\end{tikzpicture}
\tkzActivon

```

Depuis la version 2.1, il semblerait que ces problèmes disparaissent.

SECTION 10

Les points

J'ai fait une distinction entre le point utilisé en géométrie euclidienne et le point pour représenter un élément d'un nuage statistique. Dans le premier cas, j'utilise comme objet un **node**, ce qui se traduit par le fait que la représentation du point ne peut être modifiée par un **scale**; dans le second cas, j'utilise comme objet un **plot mark**. Ce dernier peut être mis à l'échelle et posséder des formes plus variées que le node.

La nouvelle macro est `\tkzDefPoint`, celle-ci permet d'utiliser des options propres à **TikZ** comme `shift` et les valeurs sont traitées avec `tkz-base`. De plus, si des calculs sont nécessaires alors c'est le package `fp.sty` qui s'en charge. On peut utiliser les coordonnées cartésiennes ou polaires.

10.1 Définition d'un point en coordonnées cartésiennes : `\tkzDefPoint`

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

arguments	défaut	définition
x,y	no default	x et y sont deux dimensions, par défaut en cm.
a:r	no default	a est un angle en degré, r une dimension

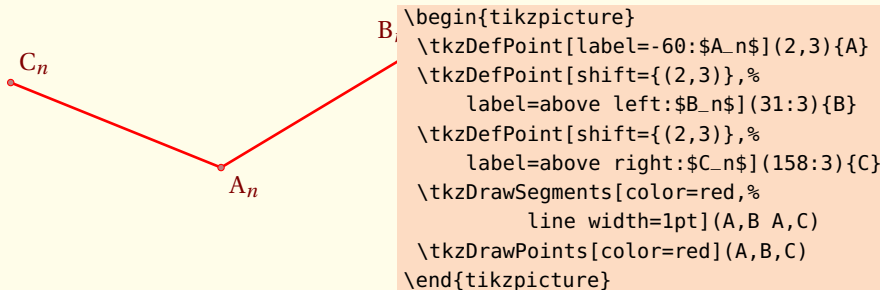
Les arguments obligatoires de cette macro sont deux dimensions exprimées avec des décimaux, dans le premier cas ce sont deux mesures de longueur, dans le second ce sont une mesure de longueur et la mesure d'un angle en degré

options	défaut	définition
shift	(0,0)	espacement entre deux valeurs
label	no default	permet de placer un label à une distance prédéfinie

Toutes les options de **TikZ** que l'on peut appliquer à **coordinate**, sont applicables (enfin je l'espère!)

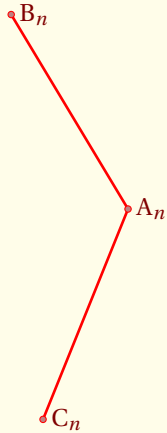
10.1.1 Utilisation de `shift` et `label`

`shift` permet de placer les points par rapport à un autre. Je n'aime guère utiliser l'option `label` mais en tout cas, c'est possible. Attention à l'utilisation de `shift`, dans certains comme celui ci-dessous, une transformation générale de la figure n'est pas possible.



10.1.2 Rotation avec `shift` et `scope`

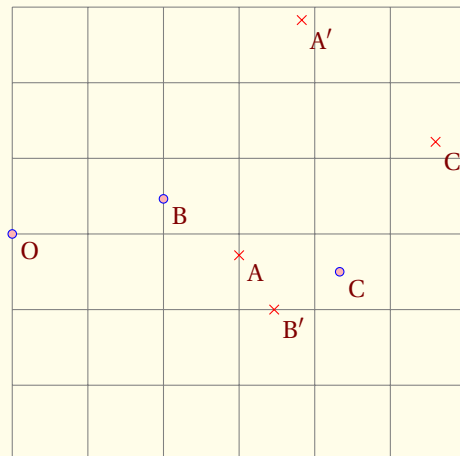
Préférable pour effectuer une rotation, est d'utiliser un environnement `scope`.



```
\begin{tikzpicture}[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}
```

10.1.3 Formules et coordonnées

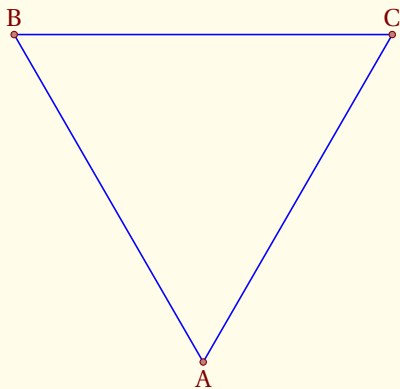
Il faut ici respecter la syntaxe de `fp.sty`. Il est toujours possible de passer par `pgfmath.sty` mais dans ce cas, il faut calculer les coordonnées avant d'utiliser la macro `\tkzDefPoint`.



```
\begin{tikzpicture}[scale=1]
\tkzInit[xmax=6,ymax=6]
\tkzGrid
\tkzSetUpPoint[shape = circle,color = red,%
size = 8,fill = red!30]
\tkzDefPoint(-1+1,-1+4){O}
\tkzDefPoint({3*\ln(exp(1))},{exp(1)}){A}
\tkzDefPoint({4*\sin(FPpi/6)},{4*\cos(FPpi/6)}){B}
\tkzDefPoint({4*\sin(FPpi/3)},{4*\cos(FPpi/3)}){B' }
\tkzDefPoint(30:5){C}
\tkzDefPoint[shift={(1,3)}](45:4){A' }
\begin{scope}[shift=(A)]
\tkzDefPoint(30:3){C' }
\end{scope}
\tkzDrawPoints[color=blue](O,B,C)
\tkzDrawPoints[color=red,%
shape=cross out](B',A,A',C')
\tkzLabelPoints(A,O,B,B',A',C,C')
\end{tikzpicture}
```

10.1.4 Scope et `\tkzDefPoint`

On peut tout d'abord utiliser l'environnement **scope** de **TikZ**. Dans l'exemple suivant, nous avons un moyen de définir un triangle isocèle.



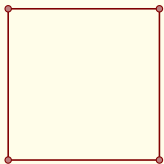
```
\begin{tikzpicture}[scale=1]
  \begin{scope}[rotate=30]
    \tkzDefPoint(2,3){A}
    \begin{scope}[shift=(A)]
      \tkzDefPoint(90:5){B}
      \tkzDefPoint(30:5){C}
    \end{scope}
  \end{scope}
  \tkzDrawSegments[color=blue](A,B B,C C,A)
  \tkzDrawPoints(A,B,C)
  % with \usetkzobj{polygons} in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

10.2 Définition de points en coordonnées cartésiennes : `\tkzDefPoints`

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

x_1 et y_1 sont les coordonnées d'un point référencé n_1

arguments	exemple
$x_i/y_i/n_i$	<code>\tkzDefPoints{0/0/0,2/2/A}</code>



```
\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 \usetkzobj{polygons} in the preamble
  % \tkzDrawPolygon
  \tkzDrawPoints(A,B,C,D)
\end{tikzpicture}
```

10.3 Point relativement à un autre : `\tkzDefShiftPoint`

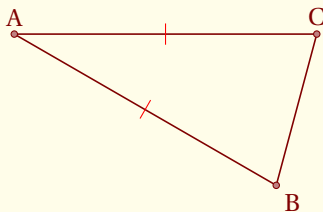
`\tkzDefShiftPoint[⟨Point⟩](⟨x,y⟩){⟨name⟩}` ou `(⟨a:r⟩){⟨name⟩}`

arguments	défaut	définition
<code>(x,y)</code>	no default	x et y sont deux dimensions, par défaut en cm.
<code>(a:r)</code>	no default	a est un angle en degré, r une dimension
<code>point</code>	no default	<code>\tkzDefShiftPoint[A](0:4){B}</code>

Pas d'option. Le nom du point est obligatoire.

10.3.1 Exemple avec `\tkzDefShiftPoint`

Cette macro permet de placer un point relativement à un autre. Cela revient à une translation. Voici comment construire un triangle isocèle de sommet principal A et d'angle au sommet de 30 degrés.



```
\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}
```

10.4 Point relativement à un autre : `\tkzDefShiftPointCoord`

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

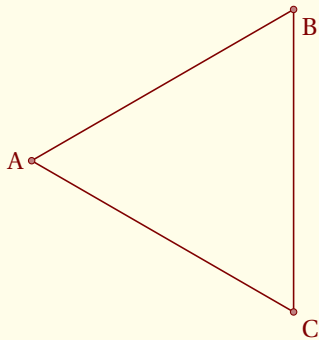
Il s'agit d'effectuer une translation de vecteur (a,b) au point défini par rapport à l'origine.

arguments	défaut	définition
<code>(x,y)</code>	no default	x et y sont deux dimensions, par défaut en cm.
<code>(a:r)</code>	no default	a est un angle en degré, r une dimension

options	défaut	exemple
<code>a,b</code>	no default	<code>\tkzDefShiftPointCoord[2,3](0:4){B}</code> <i>L'option est obligatoire</i>

10.4.1 Triangle équilatéral avec `\tkzDefShiftPointCoord`

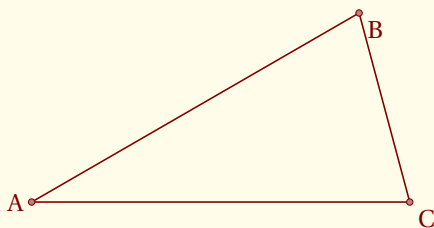
Voyons comment obtenir un triangle équilatéral (il y a beaucoup plus simple)



```
\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)
% with \usetkzobj{polygons} in the preamble
% \tkzDrawPolygon
\tkzDrawPoints(A,B,C)
\tkzLabelPoints(B,C)
\tkzLabelPoint[left](A){$A$}
\end{tikzpicture}
```

10.4.2 Triangle isocèle avec `\tkzDefShiftPointCoord`

Voyons comment obtenir un triangle isocèle dont l'angle principal est de 30 degrés. La rotation est possible.
 $AB = AC = 5$ et \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}
```

10.5 Tracer des points `\tkzDrawPoint`

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

arguments	défaut	définition
point	no default	un nom ou une référence est demandé

L'argument est obligatoire, mais il n'est pas nécessaire (bien que recommandé) d'utiliser une référence ; un couple de coordonnées place entre accolades est acceptée. Le disque prend la couleur du cercle, mais 50% plus claire. Il est possible de tout modifier. Le point est un node et donc il est invariant si le dessin est modifié par une mise à l'échelle.

options	défaut	définition
shape	circle	Possible cross ou cross out
size	6	$6 \times \text{\pgflinewidth}$
color	black	la couleur par défaut peut être changée

*On peut créer d'autres formes comme **cross***

10.5.1 Style des points par défaut

```

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

```

10.5.2 Modification du style

La définition par défaut dans le fichier **tkz-base.cfg**

```

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

```

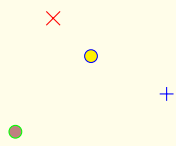
```

○ \begin{tikzpicture}
  \tkzset{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}

```

10.5.3 Exemple de tracés de points

Il faut remarquer que `scale` ne touche pas à la forme des points. Ce qui est normal. La plupart du temps, on se contente d'une seule forme de points que l'on pourra définir dès le début, soit avec une macro, soit en modifiant un fichier de configuration.



```
\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}
```

Il est possible de tracer plusieurs points en une seule fois, mais cette macro est un peu plus lente que la précédente. De plus on doit se contenter des mêmes options pour tous les points.

10.6 Tracer des points `\tkzDrawPoints`

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

arguments	défaut	définition
liste de points	no default	exemple <code>\tkzDrawPoints(A,B,C)</code>

Attention au « s » final, un oubli entraîne des erreurs en cascade si vous tentez de tracer des points multiples. Les options sont les mêmes que pour la macro précédente.

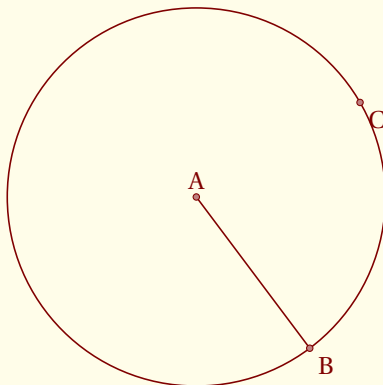
10.6.1 Exemple avec `\tkzDefPoint` et `\tkzDrawPoints`



```
\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}
```

10.6.2 Exemple plus complexe

Cet exemple nécessite `\usetkzobj{circles}`



```
\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 cm)
\tkzDrawSegment(A,B)
\tkzDrawPoints(A,B,C)
\tkzLabelPoints(B,C)
\tkzLabelPoints[above](A)
\end{tikzpicture}
```

10.7 Ajouter un label à un point `\tkzLabelPoint`

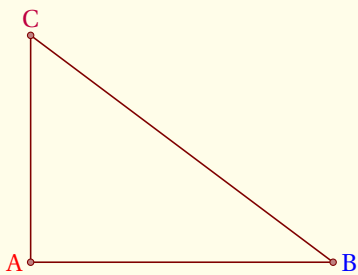
Il est possible d'ajouter plusieurs labels à un même point en utilisant plusieurs fois cette macro.

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

arguments	exemple
point	<code>\tkzLabelPoint(A){A₁}</code>

En option, on peut utiliser tous les styles de **TikZ**, en particulier le placement avec **above**, **right**, ...

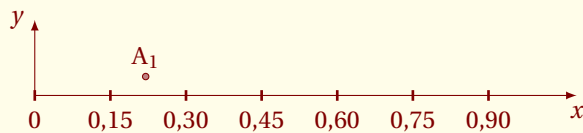
10.7.1 Exemple avec `\tkzLabelPoint`



```
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(4,0){B}
  \tkzDefPoint(0,3){C}
  \tkzDrawSegments(A,B B,C C,A)
  % with \usetkzobj{polygons} in the preamble
  % \tkzDrawPolygon
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoint[left,red](A){$A$}
  \tkzLabelPoint[right,blue](B){$B$}
  \tkzLabelPoint[above,purple](C){$C$}
\end{tikzpicture}
```

10.7.2 label et référence

La référence d'un point est l'objet qui permet d'utiliser le point, le label est le nom du point qui sera affiché.



```
\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}
```

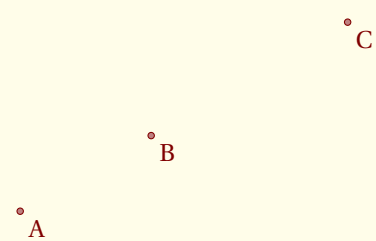
10.8 Ajouter des labels aux points `\tkzLabelPoints`

Il est possible de placer plusieurs labels rapidement quand les références des points sont identiques aux labels et quand les labels sont placés de la même manière par rapport aux points. Par défaut, c'est **below right** qui a été choisi.

<code>\tkzLabelPoints[⟨local options⟩](⟨A₁,A₂,...⟩)</code>		
arguments	exemple	résultat
list of points	<code>\tkzLabelPoint(A,B,C)</code>	Affichage de A, B et C

Cette macro diminue le nombre de lignes de codes, mais il n'est pas évident que tous les points aient besoin du même positionnement des labels.

10.8.1 Exemple avec `\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}
```

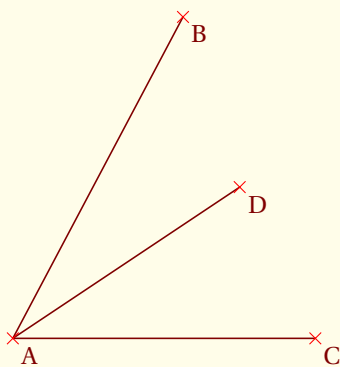
10.9 Style des points avec `\tkzSetUpPoint`

Il est important de comprendre que la taille d'un point dépend de la taille d'une ligne.

<code>\tkzSetUpPoint[⟨Local options⟩]</code>		
options	défaut	définition
shape	circle	possible : circle, cross, cross out
size	current	la taille du point est <code>size * line width</code>
color	current	exemple <code>\tkzLabelPoint(A,B,C)</code>
fill	current!50	exemple <code>\tkzLabelPoint(A,B,C)</code>

Il s'agit d'une macro permettant de choisir un style pour les points. La macro `\tkzDrawSegments` est décrite [ici](#).

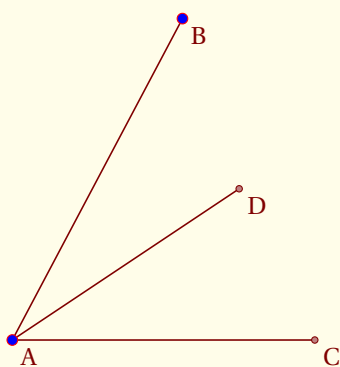
10.9.1 Exemple avec `\tkzSetUpPoint`



```
\begin{tikzpicture}
\tkzInit[ymin=-0.5,ymax=3,xmin=-0.5,xmax=7]
\tkzDefPoint(0,0){A}
\tkzDefPoint(0.25,0.25){B}
\tkzDefPoint(4,0){C}
\tkzDefPoint(3,2){D}
\tkzDrawSegments(A,B A,C A,D)
\tkzSetUpPoint[shape=cross out,size=10,color=red]
\tkzDrawPoints(A,B,C,D)
\tkzLabelPoints(A,B,C,D)
\end{tikzpicture}
```

10.9.2 Utilisation de `\tkzSetUpPoint` dans un groupe

Seuls les points du groupe sont affectés par les modifications.



```
\begin{tikzpicture}
\tkzInit[ymin=-0.5,ymax=3,xmin=-0.5,xmax=7]
\tkzDefPoint(0,0){A}
\tkzDefPoint(0.25,0.25){B}
\tkzDefPoint(4,0){C}
\tkzDefPoint(3,2){D}
\tkzDrawSegments(A,B A,C A,D)
{\tkzSetUpPoint[fill= blue,size=10,color=red]
\tkzDrawPoints(A,B)}
\tkzDrawPoints(C,D)
\tkzLabelPoints(A,B,C,D)
\end{tikzpicture}
```

10.10 Montrer les coordonnées des points `\tkzPointShowCoord`

Cette macro permet d'afficher les coordonnées d'un point et de tracer des flèches pour préciser l'abscisse et l'ordonnée. Le point est donné par sa référence (son nom). Il est possible de donner un couple de coordonnées.

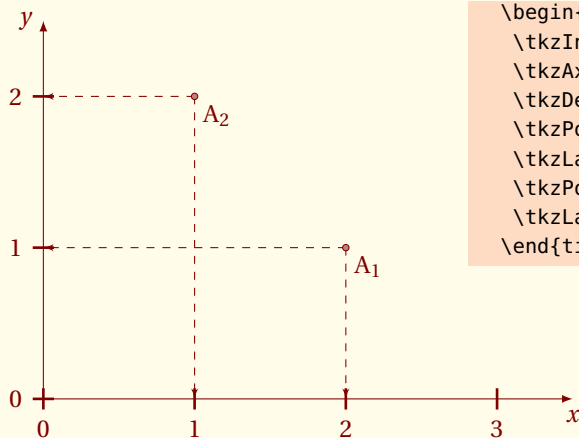
<code>\tkzPointShowCoord[<i>local options</i>](<i>point</i>)</code>		
argument	exemple	explication
<code>(<i>ref</i>)</code>	<code>\tkzPointShowCoord(A)</code>	Montre les coordonnées du point A
argument	défaut	explication
xlabel	empty	label pour l'abscisse
xstyle	empty	style pour le node du label de l'abscisse
ylabel	empty	label pour l'ordonnée
ystyle	empty	style pour le node du label de l'ordonnée
noxdraw	false	booléen pour ne pas tracer de flèche vers (x')
noydraw	false	booléen pour ne pas tracer de flèche vers (y')

10.10.1 styles par défaut

```
\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}}
```


10.10.2 Exemple avec `\tkzPointShowCoord`

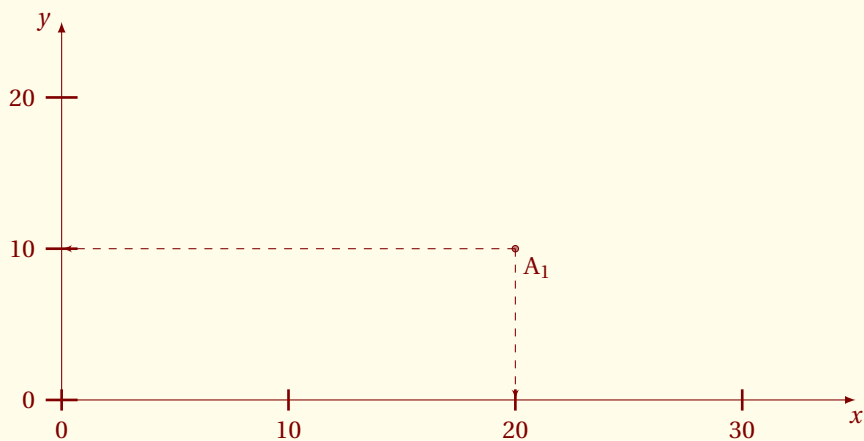
Sans les options, on n'obtient que les flèches.



```
\begin{tikzpicture}[scale=2]
\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}
```

10.10.3 Exemple avec `\tkzPointShowCoord` et `xstep`

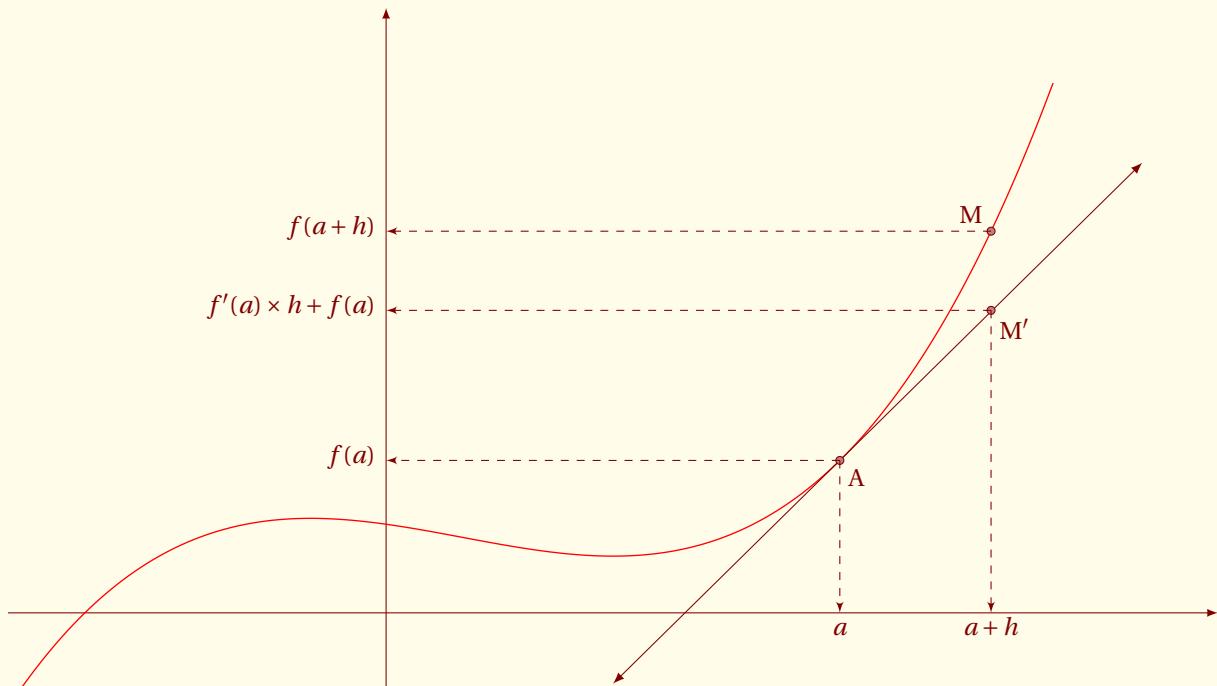
Sans les options, on n'obtient que les flèches.



```
\begin{tikzpicture}[xscale=3,yscale=2]
\tkzInit[xmax=30,ymax=20,xstep=10,ystep=10]
\tkzAxeXY
\tkzDefPoint(20,10){a} \tkzDrawPoint(a)
\tkzPointShowCoord(a)
\tkzLabelPoint(a){$A_1$}
\end{tikzpicture}
```

10.10.4 Exemple : naissance d'une tangente

Cet exemple de Gaétan Marris nécessite l'utilisation de **tkz-fct**.



```

\begin{tikzpicture}[scale=2]
  \tikzset{Style Tan/.style={solid,-,blue}}
  \tikzset{xcoord style/.append style={below=4pt}}
  \tikzset{ycoord style/.append style={left=4pt}}
  \tkzInit[xmin=-2.5,xmax=5,ymin=-.5,ymax=3.5]
  \tkzDrawX[noticks,label={}] \tkzDrawY[noticks,label={}]
  \tkzFct[domain=-3:5,samples=200,id=f,line width=0.5pt,color=red]%
    {(x-.5)*((x-.5)*(x-.5)-3)/16+.5}
  \tkzDrawTangentLine[kl=1.5,kr=2](3)
  \tkzDefPointByFct[draw](3)
  \tkzLabelPoint(tkzPointResult){$A$}
  \tkzPointShowCoord[xlabel=$a$,ylabel=$f(a)$](tkzPointResult)
  \tkzDefPointByFct[draw](4)
  \tkzLabelPoint[above left](tkzPointResult){$M$}
  \tkzPointShowCoord[noxdraw,xlabel=$a+h$,ylabel=$f(a+h)$](tkzPointResult)
  \tkzDefPoint(4,2){M'} \tkzDrawPoint(M') \tkzLabelPoint(M'){$M'$}
  \tkzPointShowCoord[ylabel=$f'(a)\times h+f(a)$](M')
\end{tikzpicture}

```

SECTION 11

Les segments

Il existe bien sûr, une macro pour tracer simplement un segment (il serait possible comme pour une demi-droite, de créer un style avec `\add`).

11.1 Tracer un segment `\tkzDrawSegment`

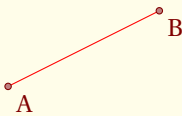
```
\tkzDrawSegment[⟨local options⟩](⟨pt1,pt2⟩)
```

Les arguments sont une liste de deux points. Les styles de **TikZ** sont accessibles pour les tracés

argument	exemple	définition
<code>(pt1,pt2)</code>	<code>(A,B)</code>	trace le segment [A,B]

C'est bien sûr équivalent à `\draw (A) -- (B);`

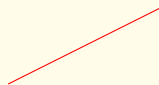
11.1.1 Exemple avec des références de points



```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=3,ymin=-1,ymax=2]
  \tkzClip
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(2,1){B}
  \tkzDrawSegment[color=red,thin](A,B)
  \tkzDrawPoints(A,B)
  \tkzLabelPoints(A,B)
\end{tikzpicture}
```

11.1.2 Exemple avec des références de points

Il est préférable de référencer les points, car les points sont placés en tenant compte de `\tkzInit`, mais il est possible d'utiliser des coordonnées.



```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=3,ymin=-1,ymax=2]
  \tkzClip
  \tkzDrawSegment[color=red,thin]({0,0},{2,1})
\end{tikzpicture}
```

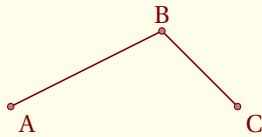
Si les options sont les mêmes, on peut tracer plusieurs segments avec la même macro.

11.2 Tracer des segments `\tkzDrawSegments`

```
\tkzDrawSegments[(local options)]((pt1,pt2 pt3,pt4 ...))
```

Les arguments sont une liste de couple de deux points. Les styles de **TikZ** sont accessibles pour les tracés

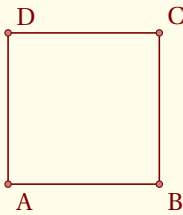
11.2.1 Exemple d'utilisation de `\tkzMarkSegments`



```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=3,ymin=-1,ymax=2]
  \tkzClip[space=1]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(2,1){B}
  \tkzDefPoint(3,0){C}
  \tkzDrawSegments(A,B B,C)
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(A,C)
  \tkzLabelPoints[above](B)
\end{tikzpicture}
```

11.2.2 Tracé d'un carré

Il y a bien sûr des méthodes plus concises pour obtenir un carré.(voir ci-dessous)



```
\begin{tikzpicture}
  \tkzInit[xmax=3,ymax=3]
  \tkzClip[space=1]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(2,0){B}
  \tkzDefPoint(2,2){C}
  \tkzDefPoint(0,2){D}
  \tkzDrawSegments(A,B B,C C,D D,A)
  \tkzDrawPoints(A,B,C,D)
  \tkzLabelPoints(A,B)
  \tkzLabelPoints[above right](C,D)
\end{tikzpicture}
```

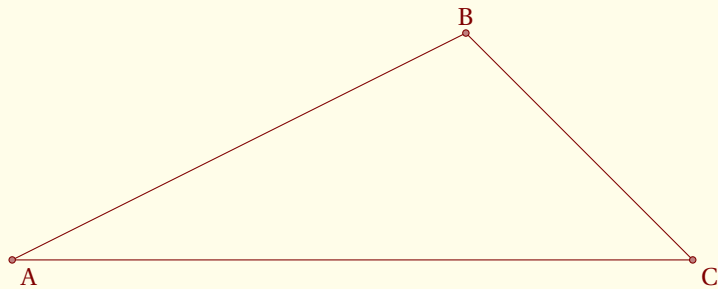
11.3 Tracer d'une ligne polygonale `\tkzDrawPolySeg`

```
\tkzDrawPolySeg[<local options>](<p_t1, p_t2, ..., p_tn>)
```

*L'argument est une liste de points. Les styles de **TikZ** sont accessibles pour les tracés*

argument	exemple	définition
<code>(p_t1, p_t2, p_t3)</code>	<code>(A,B,C)</code>	trace la ligne A,B,C

C'est bien sûr équivalent à `\draw (A)--(B)--(C);`

11.3.1 Utilisation de `\tkzDrawPolySeg`

```
\begin{tikzpicture}[scale=3]
  \tkzInit[xmin=-1,xmax=3,ymin=-1,ymax=2]
  \tkzClip[space=1]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(2,1){B}
  \tkzDefPoint(3,0){C}
  \tkzDrawPolySeg(A,B,C,A)
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(A,C)
  \tkzLabelPoints[above](B)
\end{tikzpicture}
```

11.4 Marquer un segment \tkzMarkSegment

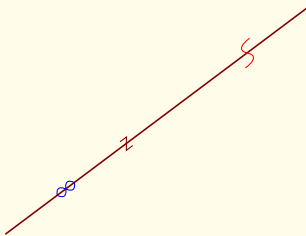
`\tkzMarkSegment[⟨local options⟩](⟨pt1,pt2⟩)`

La macro permet de placer une marque sur un segment.

options	défaut	définition
pos	.5	position de la marque
color	black	couleur de la marque
mark	none	choix de la marque
size	4pt	taille de la marque

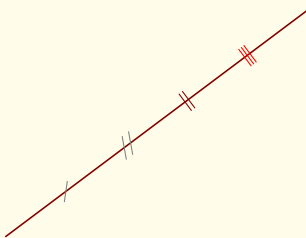
Les marques possibles sont celles fournies par **TikZ**, mais d'autres marques ont été créées d'après une idée de Yves Combe.

11.4.1 Marques multiples



```
\begin{tikzpicture}
  \tkzDefPoint(2,1){A}
  \tkzDefPoint(6,4){B}
  \tkzDrawSegment(A,B)
  \tkzMarkSegment[color=Maroon,size=2pt,pos=0.4, mark=z](A,B)
  \tkzMarkSegment[color=blue, pos=0.2, mark=oo](A,B)
  \tkzMarkSegment[pos=0.8,mark=s,color=red](A,B)
\end{tikzpicture}
```

11.4.2 Utilisation de mark



```
\begin{tikzpicture}
  \tkzDefPoint(2,1){A}
  \tkzDefPoint(6,4){B}
  \tkzDrawSegment(A,B)
  \tkzMarkSegment[color=gray,
    pos=0.2,mark=s|](A,B)
  \tkzMarkSegment[color=gray,
    pos=0.4,mark=s||](A,B)
  \tkzMarkSegment[color=Maroon,
    pos=0.6,mark=||](A,B)
  \tkzMarkSegment[color=red,
    pos=0.8,mark=|||](A,B)
\end{tikzpicture}
```

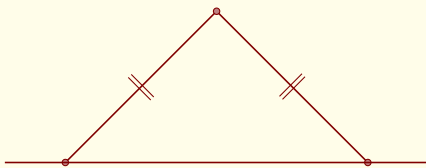
11.5 Marquer des segments `\tkzMarkSegments`

```
\tkzMarkSegments[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩)
```

Les arguments sont une liste de couple de deux points séparés par des espaces. Les styles de **TikZ** sont accessibles pour les tracés.

11.5.1 Marques pour un triangle isocèle

Cet exemple nécessite `\usetkzobj{Lines}`, mais on peut l'éviter.



```
\begin{tikzpicture}[scale=1]
\tkzDefPoints{0/0/0,2/2/A,4/0/B,6/2/C}
\tkzDrawSegments(0,A A,B)
\tkzDrawPoints(0,A,B)
\tkzDrawLine(0,B)
% \tkzDrawSegment[add=.2 and .2](0,B)
\tkzMarkSegments[mark=||,size=6pt](0,A A,B)
\end{tikzpicture}
```


11.6 Label pour un segment

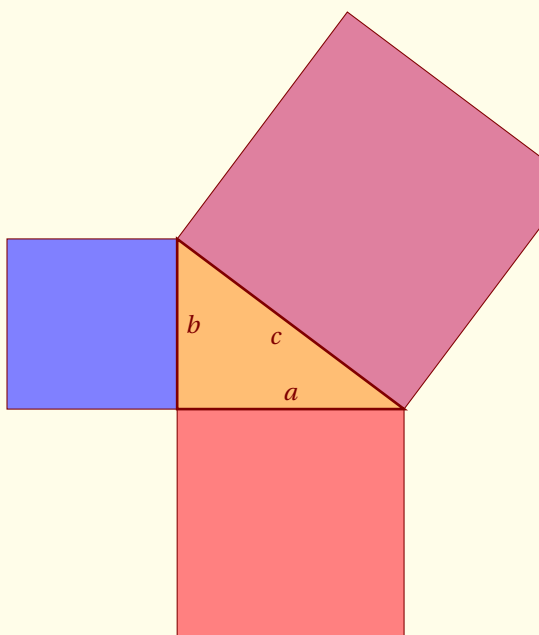
`\tkzLabelSegment[⟨local options⟩](⟨pt1,pt2⟩){⟨label⟩}`

Cette macro permet de placer une étiquette le long d'un segment ou encore d'une ligne. Les options sont celles de **TikZ** comme par exemple **pos**

argument	exemple	définition
label (pt1,pt2)	<code>\tkzLabelSegment(A,B){5}</code> (A,B)	texte de l'étiquette étiquette le long de [A,B]
options	défaut	définition
pos	.5	position du label

11.6.1 Labels et Pythagore

Cet exemple nécessite `\tkzname{usetkzobj}{\polygons}` dans le préambule



```
\begin{tikzpicture}[scale=.75]
\tkzInit[xmax=5,ymax=5]
\tkzDefPoint(0,0){C}
\tkzDefPoint(4,0){A}
\tkzDefPoint(0,3){B}
\tkzDefSquare(B,A)\tkzGetPoints{E}{F}
\tkzDefSquare(A,C)\tkzGetPoints{G}{H}
\tkzDefSquare(C,B)\tkzGetPoints{I}{J}
\tkzFillPolygon[draw,
  fill = red!50 ](A,C,G,H)
\tkzFillPolygon[draw,
  fill = blue!50 ](C,B,I,J)
\tkzFillPolygon[draw,
  fill = purple!50](B,A,E,F)
\tkzFillPolygon[draw,opacity=.5,
  fill = orange](A,B,C)
\tkzDrawPolygon[line width = 1pt](A,B,C)
\tkzLabelSegment[above](C,A){$a$}
\tkzLabelSegment[right](B,C){$b$}
\tkzLabelSegment[below left](B,A){$c$}
\end{tikzpicture}
```

11.6.2 Labels multiples



```
\begin{tikzpicture}
\tkzDefPoint(0,0){A} \tkzDefPoint(6,0){B}
\tkzDrawSegment(A,B)
\tkzLabelSegment[above,pos=.8](A,B){$a$}
\tkzLabelSegment[below,pos=.2](A,B){$4$}
\end{tikzpicture}
```

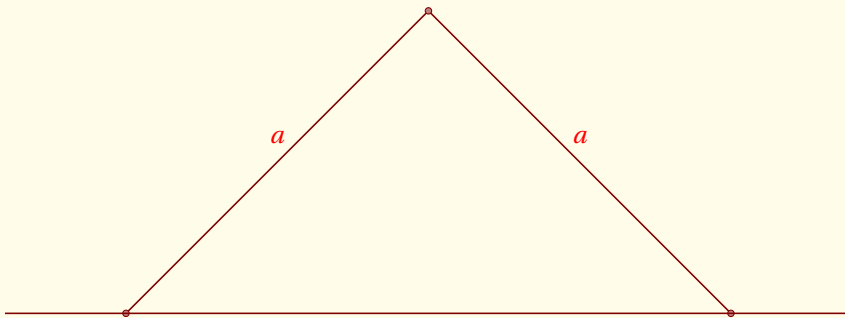
11.7 Label pour des segments

```
\tkzLabelSegments[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩)
```

Les arguments sont une liste de couple de deux points. Les styles de **TikZ** sont accessibles pour les tracés.

11.7.1 Labels pour un triangle isocèle

Cet exemple nécessite `\tkzname{usetkzobj}\{lines\}`

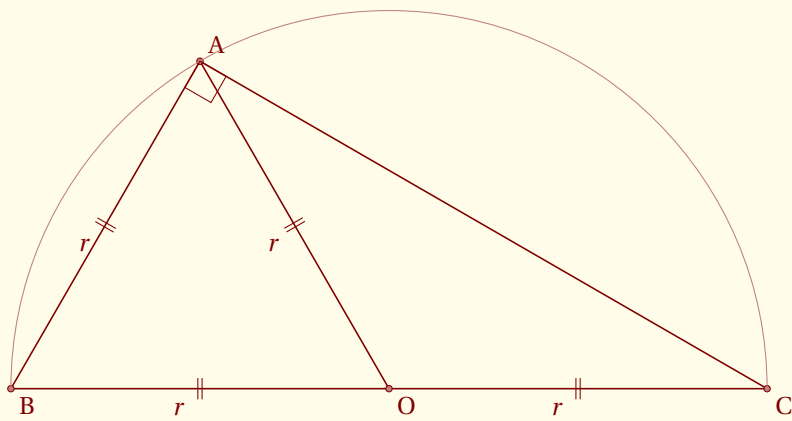


```
\begin{tikzpicture}[scale=2]
\tkzDefPoints{0/0/O,2/2/A,4/0/B,6/2/C}
\tkzDrawSegments(O,A A,B)
\tkzDrawPoints(O,A,B)
\tkzDrawLine(O,B)
% ou \tkzDrawSegment[add=.2 and .2](O,B)
\tkzLabelSegments[color=red,above=4pt](O,A A,B){$a$}
\end{tikzpicture}
```

11.7.2 Labels pour un triangle rectangle et isocèle

Cet exemple nécessite

```
\usepackage{amsmath,tkz-euclide}
\usetkzobj{all}
```



```

\begin{tikzpicture}
  \tkzInit[ymin=-1,ymax=5,xmin=-1,xmax=10]
  \tkzClip[space=.5]
  \tkzDefPoint(0,0){B}
  \tkzDefPoint(10,0){C}
  \tkzDefPoint(5,0){O}
  \tkzDefPoint(2.5,0){I}
  \tkzDefPointWith[orthogonal](I,C) \tkzGetPoint{H}
  \tkzInterLC(I,H)(0,C) \tkzGetSecondPoint{A}
  \tkzDrawSegments(B,C C,A A,B O,A)
  \tkzDrawPoints(O,A,B,C)
  \tkzDrawArc(O,C)(B)
  \tkzMarkRightAngle[size=.4](C,A,B)
  \tkzLabelSegments[below left=3pt](O,A O,B O,C A,B){$r$}
  \tkzMarkSegments[mark=||](O,A O,B O,C A,B)
  \tkzLabelPoints(B,O,C)
  \tkzLabelPoint[above right](A){$A$}
\end{tikzpicture}

```

SECTION 12

Marks, marques ou symboles

J'ai distingué les points utilisés en géométrie euclidienne et les « marks » ou symboles que l'on peut rencontrer en statistiques.

Pour positionner le symbole, on utilise la macro `\tkzDefPoint` pour définir correctement un point, puis la macro `\tkzDrawMark` pour tracer le symbole.

Il est fréquent d'avoir à tracer un nuage de points, j'ai donc créé une macro qui permet de définir plusieurs points rapidement.

Un symbole "mark" peut être mise à l'échelle, ce qui est parfois utile, mais en revanche si on met modifie différemment les abscisses et les ordonnées alors les "marks" sont déformées.

Rappel : il était déjà possible de créer un nuage de points avec la macro `\tkzDefPoints`, mais cela impose de donner une référence (un nom) à chaque point, ce qui est parfois fastidieux. La macro `\tkzSetOfPoints` permet de définir des points `tkzPt1`, `tkzPt2`, etc.

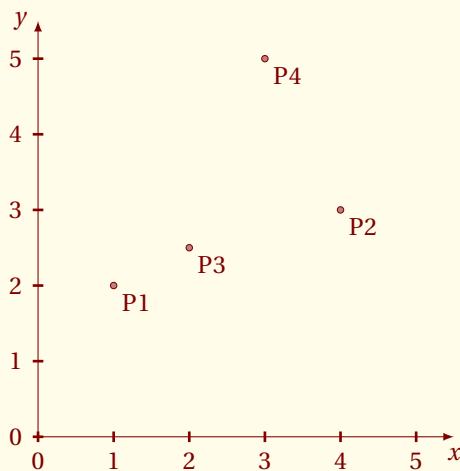
12.1 `\tkzDefSetOfPoints`

C'est ce qu'on appelle fréquemment « nuage de points ». La différence par rapport à la macro `\tkzDefPoints`, c'est que la référence aux points est donnée par un préfixe (par défaut `tkzPt`) et le numéro du point. Les points ne sont pas tracés.

```
\tkzDefSetOfPoints[⟨local options⟩]{⟨x1/y1, x2/y2, ..., xn/yn⟩}
```

arguments	défaut	définition
x_n/y_n	no default	Liste de couples x_n/y_n séparés par des virgules
options	défaut	définition
prefix	<code>tkzPt</code>	préfixe pour les noms des points

12.1.1 Création d'un nuage avec `\tkzDefPoints`



```
\begin{tikzpicture}
  \tkzInit[ymax=5,xmax=5]
  \tkzAxeXY
  \tkzDefSetOfPoints[prefix=P]%
    {1/2,4/3,2/2.5,3/5}
  \tkzDrawPoints(P1,P2,P3,P4)
  \tkzLabelPoints(P1,P2,P3,P4)
\end{tikzpicture}
```

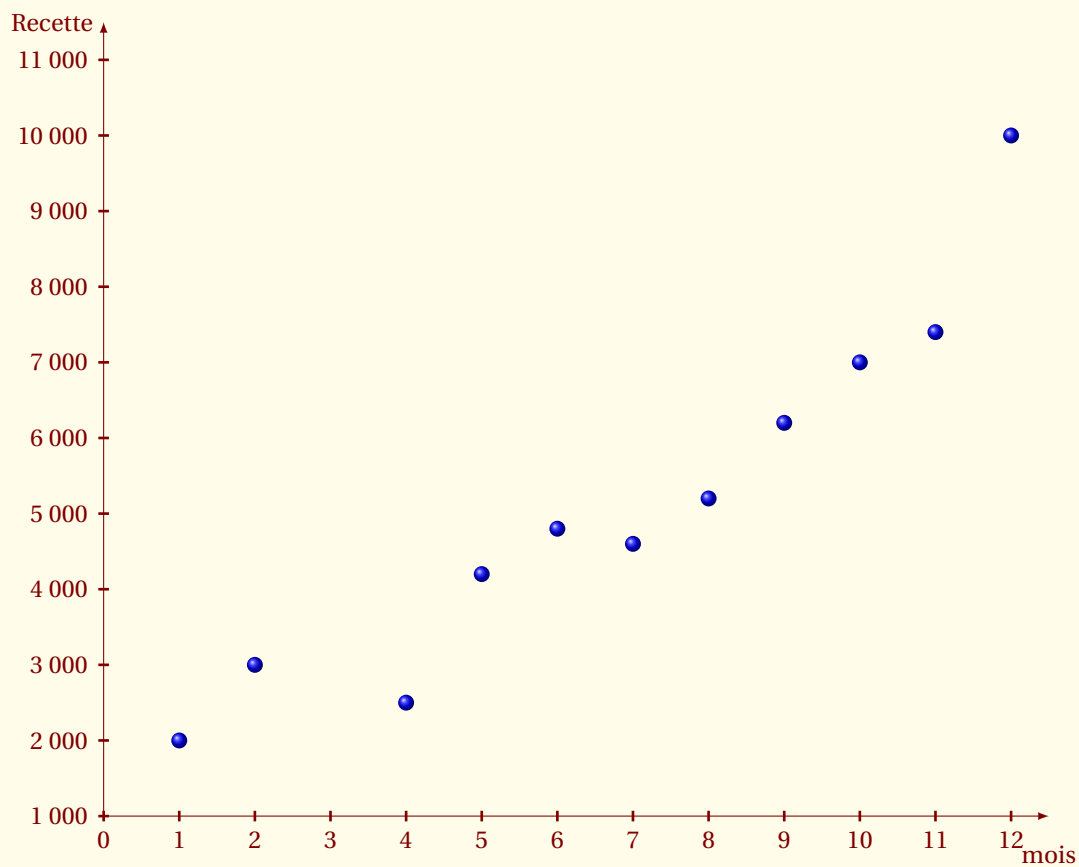
12.2 \tkzDrawSetOfPoints

`\tkzDrawSetOfPoints[<local options>]`

Permet de placer des symboles sur les points définis par `\tkzDefSetOfPoints`.

options	défaut	définition
prefix	tkzPt	préfixe des noms des points

12.2.1 Tracé d'un nuage avec \tkzDrawSetOfPoints



```
\begin{tikzpicture}
\tkzInit[xmax=12,ymin=1000,ymax=11000,ystep=1000]
\tkzAxeX[label=mois,below=10pt]
\tkzAxeY[label=Recette]
\tkzDefSetOfPoints[show]{%
  1/2000,2/3000,4/2500,5/4200,6/4800,7/4600,8/5200,9/6200,
  10/7000,11/7400,12/10000}
\tkzDrawSetOfPoints[mark=ball,mark size=3pt]
\end{tikzpicture}
```

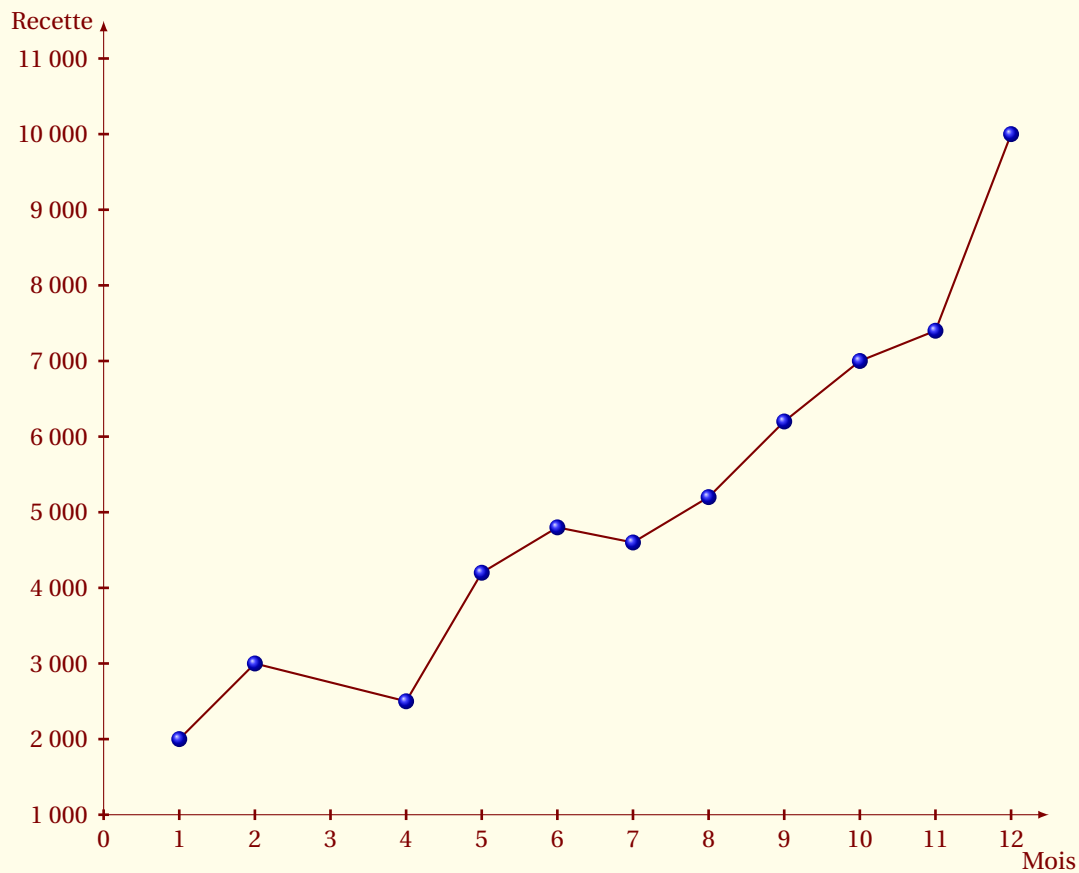
12.3 \tkzJoinSetOfPoints

`\tkzJoinSetOfPoints[⟨local options⟩]`

Permet de joindre les symboles par des segments de droite. Il est possible d'utiliser bien sûr toutes les options de **TikZ**.

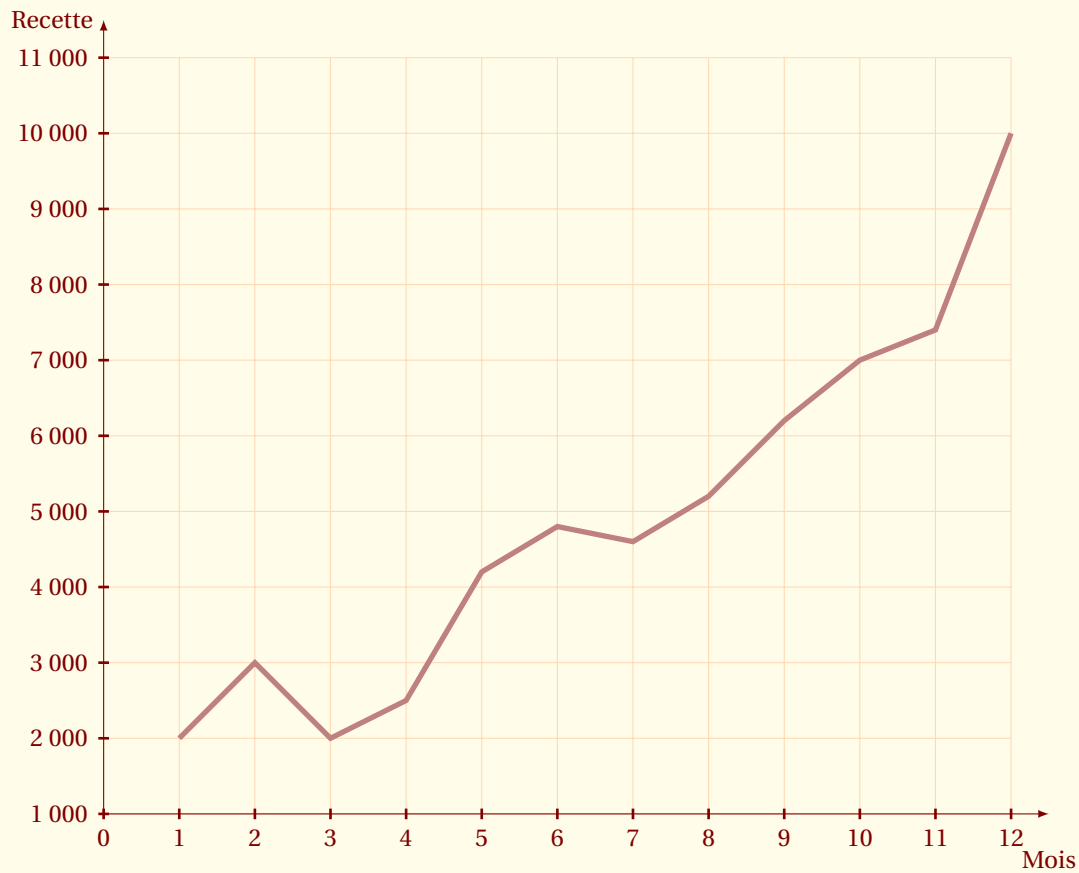
options	défaut	définition
prefix	tkzPt	préfixe des noms des points

12.3.1 Lier les points d'un nuage avec \tkzJoinSetOfPoints



```
\begin{tikzpicture}[scale=1]
\tkzInit[xmax=12,ymin=1000,ymax=11000,ystep=1000]
\tkzAxeX[label=Mois,below=13pt]
\tkzAxeY[label=Recette]
\tkzDefSetOfPoints{%
  1/2000,2/3000,4/2500,5/4200,6/4800,7/4600,8/5200,9/6200,
  10/7000,11/7400,12/10000}
\tkzJoinSetOfPoints[thick,color=Maroon]
\tkzDrawSetOfPoints[mark=ball,mark size=3pt]
\end{tikzpicture}
```

12.3.2 Utilisation des points d'un nuage



```

\begin{tikzpicture}[scale=1]
\tkzInit[xmax=12,ymin=1000,ymax=11000,ystep=1000]
\tkzGrid[color=orange!30]
\tkzAxeX[label=Mois,below=13pt]
\tkzAxeY[label=Recette]
\tkzDefSetOfPoints[prefix=P]{%
  1/2000,2/3000,3/2000,4/2500,5/4200,6/4800,7/4600,8/5200,9/6200,
  10/7000,11/7400,12/10000}
\tkzDrawPolySeg[color=Maroon!50,
  line width=2pt](P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12)
\end{tikzpicture}

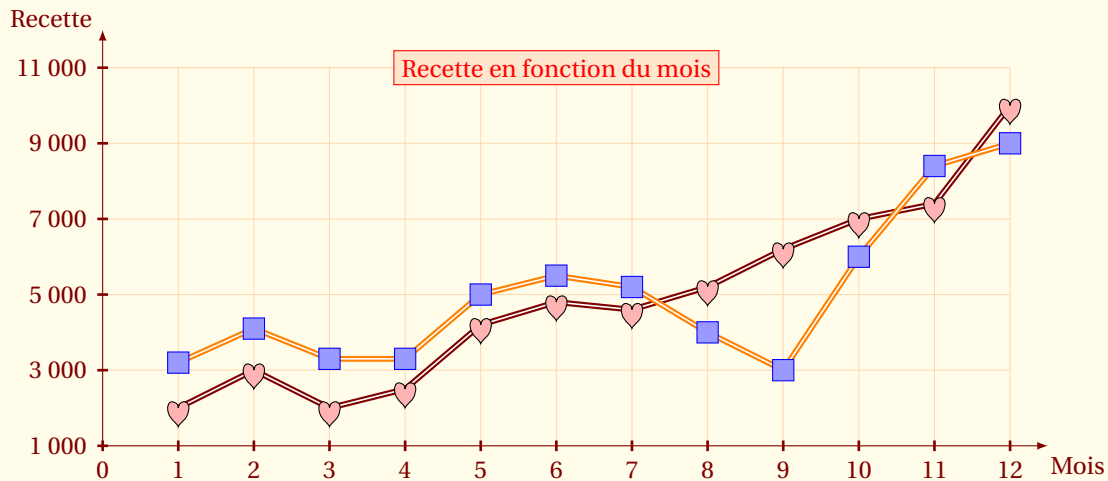
```

12.4 \tkzSetUpMark

`\tkzSetUpMark[⟨local options⟩]`

options	défaut	définition
liste	no default	exemple <code>\tkzLabelPoint(A,B,C)</code>

12.4.1 Deux nuages



```

\begin{tikzpicture}
\tkzInit[xmax=12,ymin=1000,ymax=11000,ystep=2000]
\tkzGrid[color=orange!30]
\tkzAxeX[below right,label=Mois]
\tkzAxeY[above left,label=Recette]
\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=Maroon,double]
\tkzDrawSetOfPoints
\tkzJoinSetOfPoints[prefix=P,thick,color=orange,double]
\tkzDrawSetOfPoints[prefix=P,mark=square*,mark size=4pt,
mark options={color=blue,fill=blue!40}]
\tkzText[draw,color = red,fill = orange!20](6,11000){Recette en fonction du mois}
\end{tikzpicture}

```

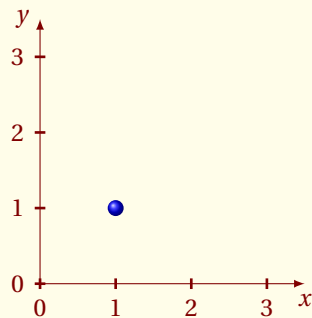

12.5 \tkzDrawMark

\tkzDrawMark[*local options*](*(*()*)point*)

Place un symbole. Plus efficace que la suivante pour placer un seul symbole.

options	défaut	définition
prefix	tkzPt	préfixe des noms des points

tkzJoinSetOfPoints



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

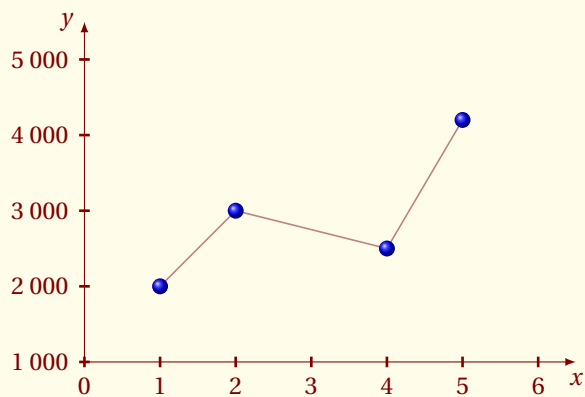
12.6 \tkzDrawMarks

\tkzDrawMarks[*local options*](*(*()*)list of points*)

Permet de placer une série de marques.

options	défaut	définition
prefix	tkzPt	préfixe des noms des points

12.6.1 Mark et nuage ; utilisation de \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=Maroon!50]%
(P1,P2 P2,P3 P3,P4)
\tkzDrawMarks[mark=ball](P1,P2,P3,P4)
\end{tikzpicture}
```

SECTION 13

Textes et Légendes

13.1 Placer un titre

On peut bien sûr utiliser **TikZ**, mais la macro que je propose permet de placer le texte en utilisant les unités choisies pour le dessin.

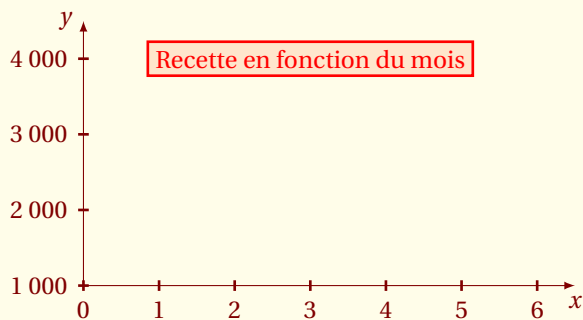
les options sont toujours celles de **TikZ**, en particulier les suivantes :

```
\tkzText[local options](point){text}
```

Le point peut soit être donné par ses coordonnées, soit par son nom.

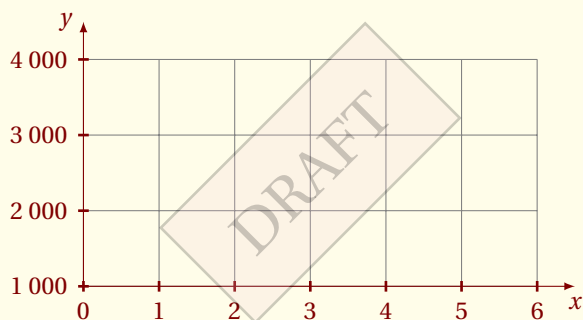
options	défaut	définition
color	black	couleur courante
text	black	couleur du texte
fill	white	couleur du fond
opacity	1	opacité

13.1.1 Un titre



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

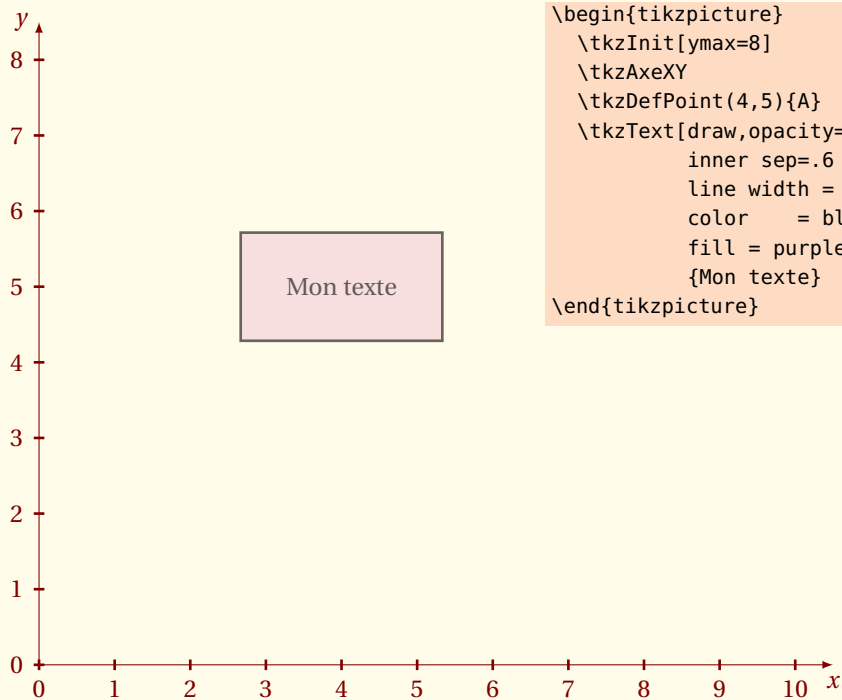
13.1.2 Draft



```
\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}
```

13.1.3 Texte avec un point

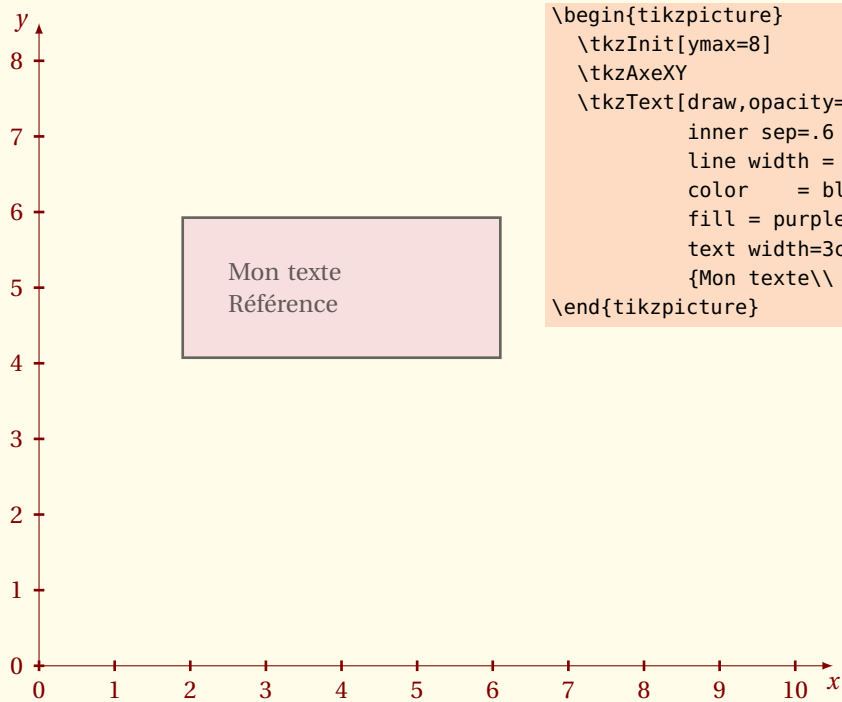
Il est possible de donner la référence d'un point à la place de ses coordonnées.



```
\begin{tikzpicture}
  \tkzInit[ymax=8]
  \tkzAxeXY
  \tkzDefPoint(4,5){A}
  \tkzText[draw,opacity=.6,
    inner sep=.6 cm,
    line width = 1pt,
    color      = black,
    fill       = purple!20](A)
    {Mon texte}
\end{tikzpicture}
```

13.1.4 Format du texte

L'option **text width** est intéressante, voir le `pgfmanual` pour plus d'informations.



```
\begin{tikzpicture}
  \tkzInit[ymax=8]
  \tkzAxeXY
  \tkzText[draw,opacity=.6,
    inner sep=.6 cm,
    line width = 1pt,
    color      = black,
    fill       = purple!20,
    text width=3cm](4,5)
    {Mon texte\\ Référence}
\end{tikzpicture}
```

13.2 Placer des légendes

Il y a deux façons d'utiliser cette macro. Soit on place des légendes pour des courbes. Alors, il faut représenter des lignes avec leur style propre, soit il s'agit de différencier des symboles (mark).

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

Les arguments diffèrent en fonction du booléen **ligne**.

options	défaut	définition
line	false	booléen :ligne ou symbole

Avec ligne=true

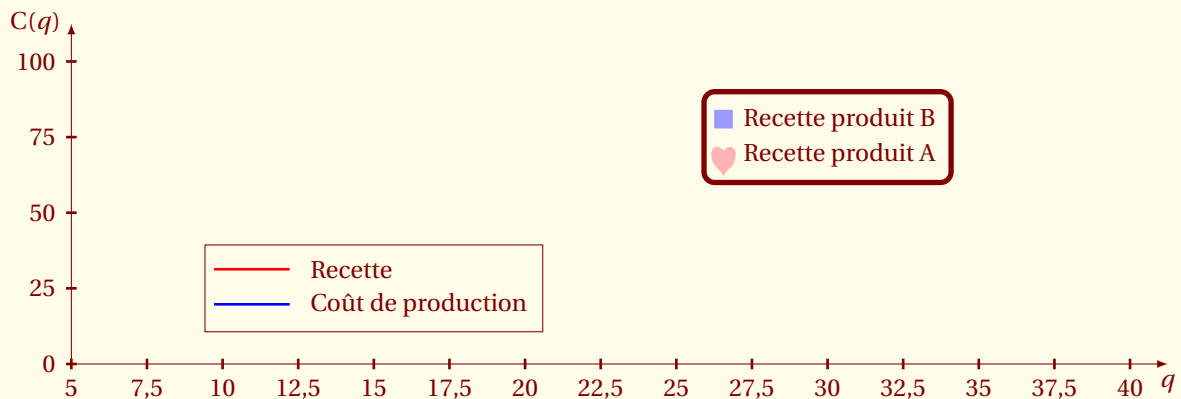
arguments	défaut	exemple
style/line width/couleur/texte	pas de défaut	dashed/1pt/red/Recette

Avec ligne=false

arguments	défaut	exemple
mark/mark size//couleur/texte	pas de défaut	heart/1ex/red!30/Recette produit A

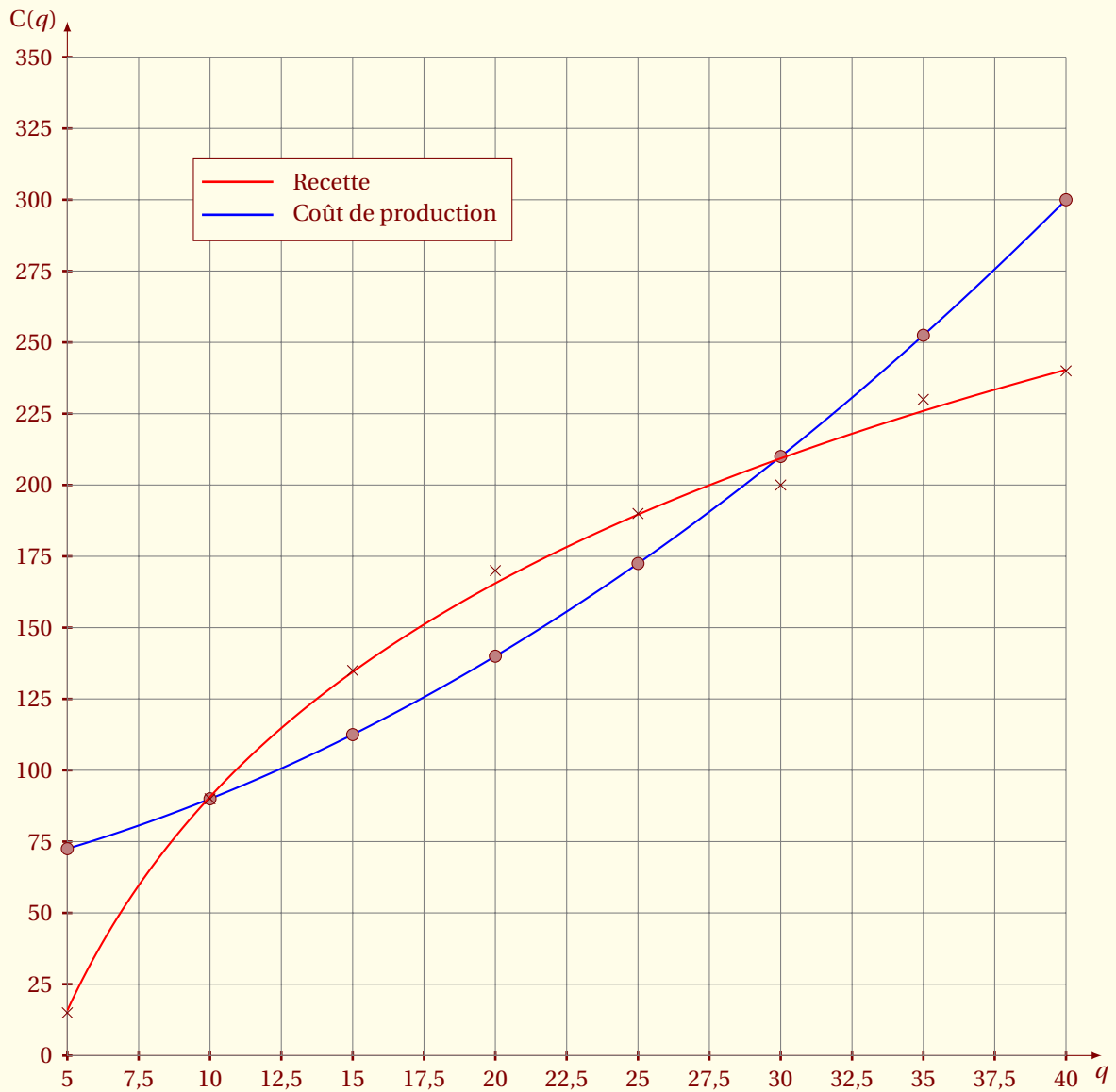
On peut modifier la longueur de la ligne dans **tkz-base.cfg**

```
\global\def\tkz@legend@line@len{.5cm}
```



```
\begin{tikzpicture}
\tkzInit[xmin=5,xmax=40,ymin=0,ymax=100,xstep=2.5,ystep=25]
\tkzAxeX[label=$q$] \tkzAxeY[label=$C(q)$]
\tkzLegend[fill=fondpaille,draw,line=true](15,25)%
{solid/1pt/blue/Coût de production,
solid/1pt/red/Recette}
\tkzLegend[draw,rounded corners,fill=fondpaille,text=Maroon,
line width=2pt](30,75)%
{heart/1ex/red!30/Recette produit A,%
square*/0.75ex/blue!40/Recette produit B}
\end{tikzpicture}
```

13.2.1 Légendes avec des lignes

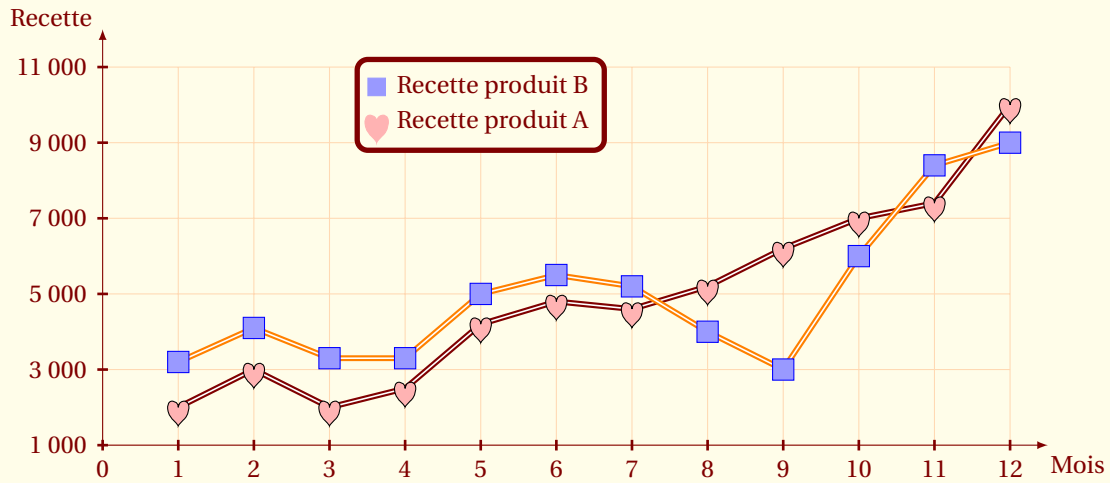


```

\begin{tikzpicture}
\tkzInit[xmin=5,xmax=40,ymin=0,ymax=350,xstep=2.5,ystep=25]
\tkzAxeX[label=$q$] \tkzAxeY[label=$C(q)$] \tkzGrid
\tkzFct[color=blue,thick,domain=5:40]{0.1*\x**2+2*\x+60}
\foreach \vv in {5,10,...,40}{%
\tkzDefPointByFct(\vv) \tkzDrawPoint(tkzPointResult)}
\tkzFct[color=red,thick,domain=5:40]{(108*log(\x)-158)}
\tkzDefSetOfPoints{5/15,10/90,15/135,20/170,25/190,30/200,35/230,40/240}
\tkzDrawSetOfPoints[mark = x,mark size=3pt]
\tkzLegend[fill=fondpaille,draw,line=true](15,300)%
{solid/1pt/blue/Coût de production, solid/1pt/red/Recette}
\end{tikzpicture}

```

13.2.2 Légendes avec des symboles



```

\begin{tikzpicture}
\tkzInit[xmax=12,ymin=1000,ymax=11000,ystep=2000]
\tkzGrid[color=orange!30]
\tkzAxeX[below right,label=Mois]
\tkzAxeY[above left,label=Recette]
\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=Maroon,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=fondpaille,text=Maroon,
line width=2pt](5,10000){heart/lex/red!30/Recette produit A,%
square*/0.75ex/blue!40/Recette produit B}
\end{tikzpicture}

```

SECTION 14

Utilisation des objets complémentaires

Ces objets complémentaires peuvent être des points particuliers, des droites, des cercles, des arcs, etc.

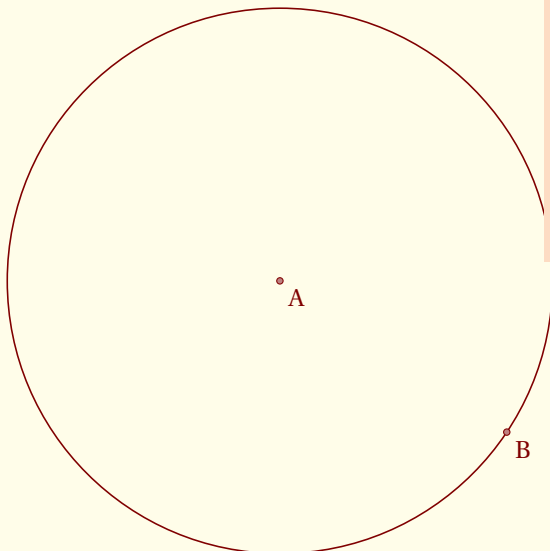
Il est possible d'utiliser certains de ces objets, sans charger complètement **tkz-euclide**, mais en utilisant la macro `\usetkzobj`. Attention, il faut utiliser **tkz-euclide** pour avoir la possibilité d'utiliser des outils comme les transformations ou encore les intersections.

tkz-base charge les objets les plus utilisés, marqués « présent » dans la liste ci-dessous. Cette liste peut évoluer.

`\usetkzobj{<liste d'objets>}`

options		définition
all	absent	tous les objets sont chargés
points	présent	définir, nommer, tracer des points
lines	absent	définir, nommer, tracer des droites
segments	présent	définir, nommer, tracer des segments
vectors	absent	définir, nommer, tracer des des vecteurs
circles	absent	définir, nommer, tracer des cercles
polygons	absent	définir, nommer, tracer des quadrilatères
arcs	absent	définir, nommer, tracer des arcs
sectors	absent	définir, nommer, tracer des secteurs
protractor	absent	tracer un rapporteur
marks	présent	définir, nommer, tracer des marques

14.1 `\usetkzobj{circles}`



```
\begin{tikzpicture}
  \tkzDefPoint(0,4){A}
  \tkzDefPoint(3,2){B}
  \tkzDefCircle[radius](A,B)
  \tkzDrawCircle(A,B)
  \tkzDrawPoints(A,B)
  \tkzLabelPoints(A,B)
\end{tikzpicture}
```


SECTION 15

Droites parallèles aux axes

15.1 Tracer une ligne horizontale avec `\tkzHLine`

```
\tkzHLine[<local options>]{<decimal number>}
```

Attention, la syntaxe est celle de **fp** car on n'utilise pas **gnuplot** pour tracer une droite.

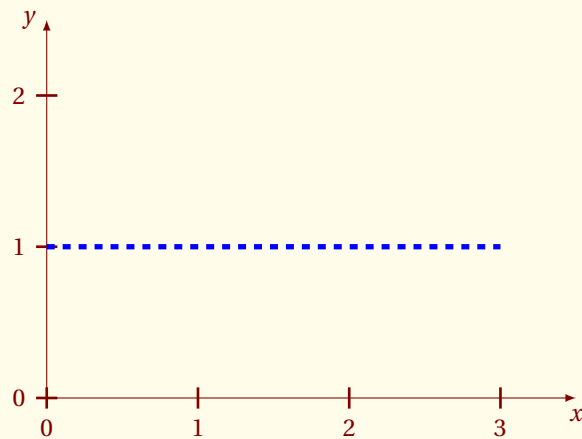
arguments	exemple	définition
decimal number	<code>\tkzHLine{1}</code>	Trace la droite $y=1$

options	défaut	définition
color	black	couleur du trait
line width	0.6pt	épaisseur du point
style	solid	style du trait

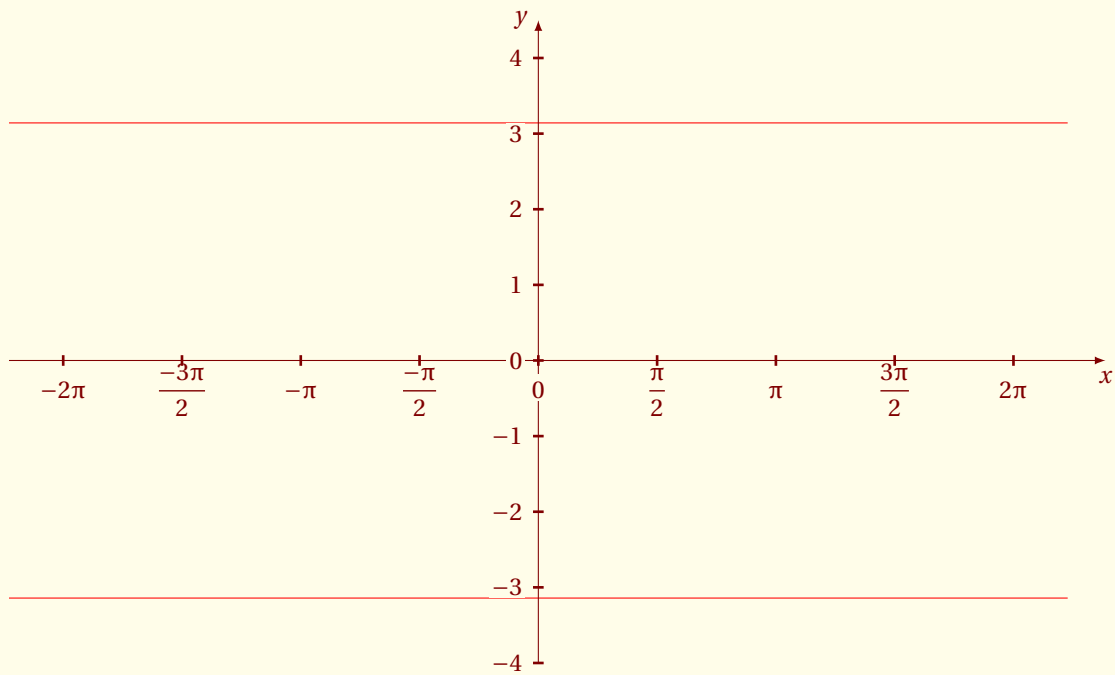
voir les options les lignes dans **TikZ**

15.1.1 Ligne horizontale

problème avec cette macro, en principe 1./3 devrait être acceptée.



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

15.1.2 Ligne horizontale et valeur calculée par fp

```
\begin{tikzpicture}
  \tkzInit[xmin=-7,xmax=7,ymin=-4,ymax=4]
  \foreach \v in {-1,1}
  {\tkzHLine[color=red]{\v*\FPpi}}
  \tkzDrawY
  \tkzAxeX[trig=2]
  \tkzLabelY
\end{tikzpicture}
```

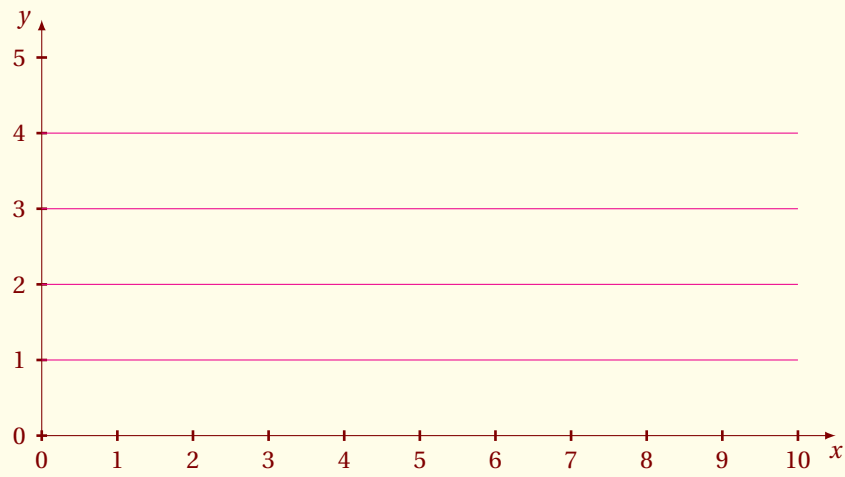
15.2 Lignes horizontales avec `\tkzHLines`

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

Attention, la syntaxe est celle de **fp** car on n'utilise pas **gnuplot** pour tracer une droite.

arguments	exemple	définition
list of values	<code>\tkzHLines{1,4}</code>	Trace les droites $x=1$ et $x=4$

15.2.1 Lignes horizontales



```
\begin{tikzpicture}
\tkzInit[xmax=10,ymax=5]
\tkzAxeXY
\tkzHLines[color = magenta]{1,...,4}
\end{tikzpicture}
```

15.3 Tracer une ligne verticale avec `\tkzVLine`

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

Attention, la syntaxe est celle de **fp** car on n'utilise pas **gnuplot** pour tracer une droite.

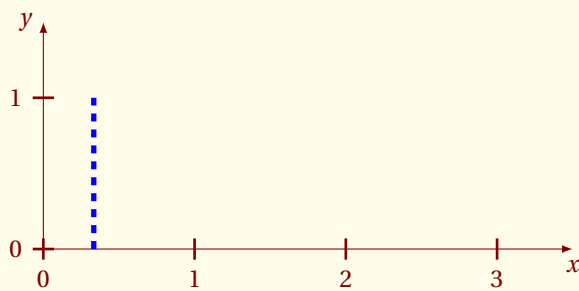
arguments	exemple	définition
decimal number	<code>\tkzVLine{1}</code>	Trace la droite $x=1$

options	défaut	définition
color	black	couleur du trait
line width	0.6pt	épaisseur du point
style	solid	style du trait

voir les options les lignes dans **TikZ**

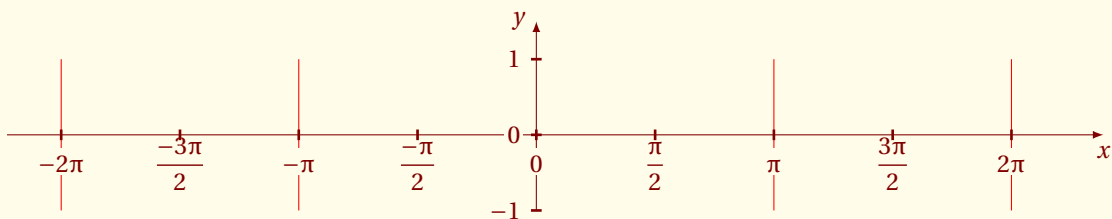
15.3.1 Ligne verticale

problème avec cette macro, en principe $1./3$ devrait être acceptée.



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

15.3.2 Ligne verticale et valeur calculée par fp

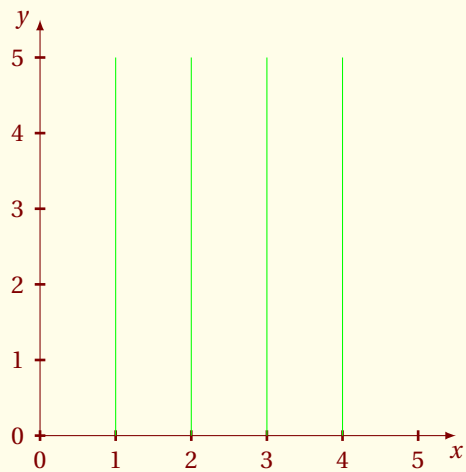


```
\begin{tikzpicture}
  \tkzInit[xmin=-7,xmax=7,ymin=-1,ymax=1]
  \foreach \v in {-2,-1,1,2}
  {\tkzVLine[color=red]{\v*\FPpi}}
  \tkzDrawY
  \tkzAxeX[trig=2]
  \tkzLabelY
\end{tikzpicture}
```

15.4 Lignes verticales avec `\tkzVLines``\tkzVLines[local options]{list of values}`

Attention, la syntaxe est celle de **fp** car on n'utilise pas **gnuplot** pour tracer une droite.

arguments	exemple	définition
list of values	<code>\tkzVLines{1,4}</code>	Trace les droites $x=1$ et $x=4$

15.4.1 Lignes verticales

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

SECTION 16

Ticks sur les axes

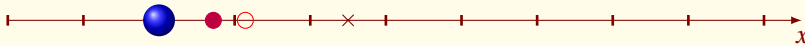
16.1 Tracer des ticks sur l'axe des abscisses `\tkzHTick`

$$\text{\tkzHTick}[\langle local options \rangle]{\langle decimal number \rangle}$$

arguments	exemple	définition
decimal number	<code>\tkzHTick{1}</code>	l'abscisse du tick est 1
options	défaut	définition
mark	*	disque plein
mark size	3 pt	taille du symbole
mark options	vide	permet d'utiliser color par exemple

voir les options de **TikZ**

16.1.1 exemple



```
\begin{tikzpicture}
  \tkzInit
  \tkzDrawX[]
  \tkzHTick[mark=ball,mark size=6pt]{2}
  \tkzHTick[mark=*,mark options={color=purple}]{exp(1)}
  \tkzHTick[mark=o,mark options={color=red}]{pi}
  \tkzHTick[mark=x,mark options={color=Maroon}]{4.5}
\end{tikzpicture}
```

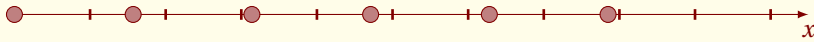
16.2 Tracer des ticks sur l'axe des ordonnées `\tkzHTicks`

$$\text{\tkzHTicks}[\langle local options \rangle]{\langle list of numbers \rangle}$$

arguments	exemple	définition
decimal number	<code>\tkzHTicks{1}</code>	l'abscisse du tick est 1

voir les options de **TikZ**.

16.2.1 exemple



```
\begin{tikzpicture}
  \tkzInit
  \tkzDrawX
  \tkzHTicks[mark=*]{0,1.57,...,9}
\end{tikzpicture}
```

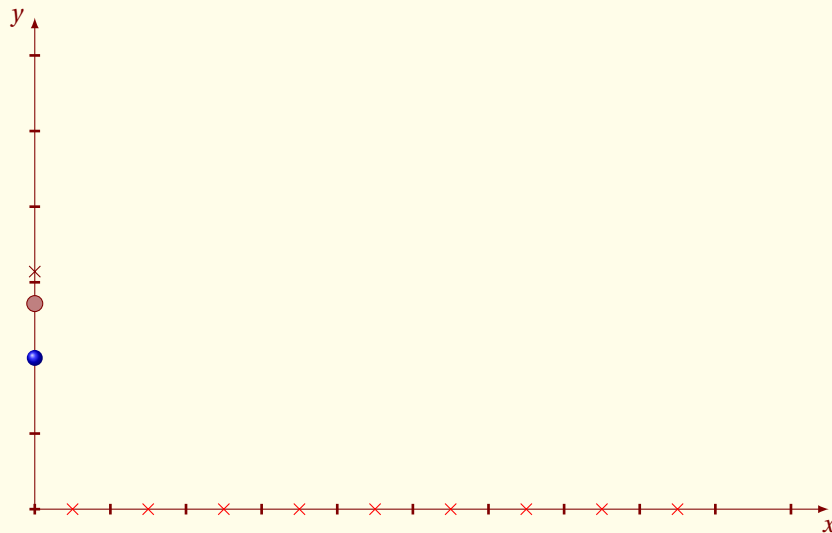
16.3 Tracer des ticks sur l'axe des abscisses `\tkzVTick`

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

arguments	exemple	définition
decimal number	<code>\tkzVTick{1}</code>	l'abscisse du tick est 1

voir les options de **TikZ**.

16.3.1 exemple



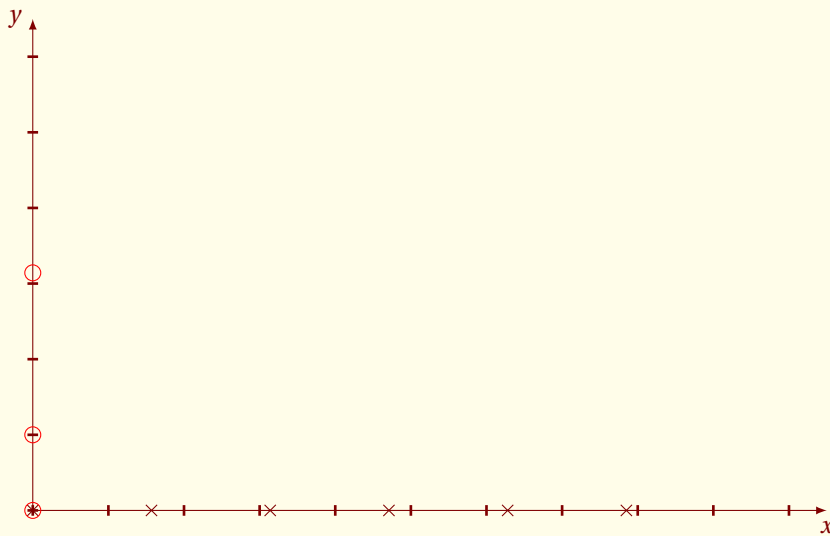
```
\begin{tikzpicture}
  \tkzInit[ymax=6]
  \tkzDrawXY
  \tkzVTick[mark=ball]{2}
  \tkzVTick[mark=*]{exp(1)}
  \tkzVTick[mark=x]{pi}
  \tkzHTicks[mark=x,mark options={color=red}]{0.5,1.5,...,9}
\end{tikzpicture}
```

16.4 Tracer des ticks sur l'axe des abscisses `\tkzVTicks`

<code>\tkzVTicks[<i>local options</i>]{<i>decimal number</i>}</code>
--

arguments	exemple	définition
<code>decimal number</code>	<code>\tkzVTicks{1,3}</code>	les ordonnées des ticks sont 1 et 3

voir les options de `TikZ`.

16.4.1 exemple

```
\begin{tikzpicture}
  \tkzInit[ymax=6]
  \tkzDrawXY
  \tkzHTicks[mark=x]{0,1.57,...,9}
  \tkzVTicks[mark=o,mark options={color=red,fill=red!50}]{0,1,pi}
\end{tikzpicture}
```


Utilisation des styles

17.1 Modification de `tkz-base.cfg`

`tkz-base.sty` possède un fichier de configuration par défaut. Son existence n'est pas obligatoire, mais s'il existe, vous pouvez le modifier pour obtenir des styles par défaut différents. Je ne donne qu'une description rapide de ce fichier, car il risque d'évoluer prochainement.

Dans `tkz-base.cfg`, on peut régler les axes, le repère (si on l'utilise), la grille, etc. ainsi que les styles qui sont liés à ces objets. Il est possible de modifier les styles des points et des segments.

Il est aussi possible de définir les dimensions d'un dessin par défaut en modifiant `xmin`, `xmax`, `ymin` et `ymax`.

```
\xdef\cmdTKZ@tkzInit@xmin{0}
\xdef\cmdTKZ@tkzInit@ymin{0}
\xdef\cmdTKZ@tkzInit@xmax{10}
\xdef\cmdTKZ@tkzInit@ymax{10}
```

Ces lignes permettent de définir les valeurs de `xmin`, `xmax`, etc.

Vous pouvez les modifier, par exemple :

```
\xdef\cmdTKZ@tkzInit@xmin{-5}
\xdef\cmdTKZ@tkzInit@ymin{-5}
\xdef\cmdTKZ@tkzInit@xmax{5}
\xdef\cmdTKZ@tkzInit@ymax{5}
```

Ce paragraphe n'est pas terminé et il sera complété prochainement.

Voici une liste des styles utilisés que vous trouverez dans `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

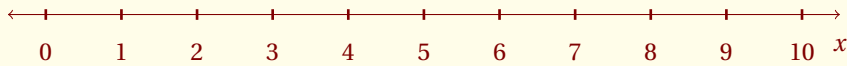
17.2 Utilisation `\tikzset`

Il est préférable d'utiliser désormais `\tikzset` plutôt que `\tikzstyle` et il est possible de s'inspirer de `tkz-base.cfg`.

Si vous voulez modifier l'aspect des axes du repère, par exemple placer des flèches à chaque extrémité ou bien les supprimer

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

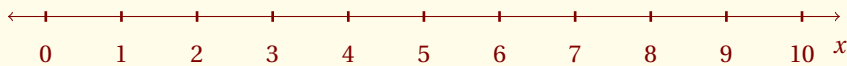
La transformation sera valable pour tout le document. Il faut noter que `xmin` a été modifié, en effet la flèche et le trait correspondant à la graduation se confondent.



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

17.3 Utilisation `\tikzset` dans un groupe

Si vous voulez limiter l'action à une figure, alors il faut utiliser un groupe au sens de $\text{T}_{\text{E}}\text{X}$, cela signifie de placer la commande entre accolades. Voici deux exemples avec l'ancienne macro puis les nouvelles pour obtenir l'axe des abscisses.

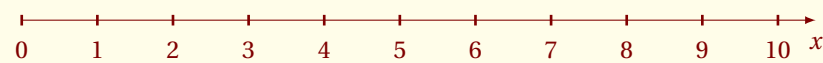


```
\begin{tikzpicture}
  \tkzInit[xmin=-0.5]
  { \tikzset{xaxe style/.style = {<->}}
    \tikzset{xlabel style/.style={below=6pt}}
    \tkzAxeX }
\end{tikzpicture}
```

17.4 Utilisation de `\tikzset` dans `tkz-base.cfg`

```
\tikzset{xlabel style/.style = {below=3pt}}
```

ceci peut se faire dans `tkz-base.cfg` ou bien dans votre code.



```
\tikzset{xlabel style/.style = {below=3pt}}
\begin{tikzpicture}
  \tkzInit
  \tkzAxeX
\end{tikzpicture}
```

17.5 Macro de configuration

– [\tkzSetUpPoint](#)

– `\tkzSetUpAxis`

Il y a aussi `\tkzSetUpColors`, placée dans le préambule elle permet de choisir la couleur du fond et la couleur du texte, pour cette documentation, j'ai choisi

```
\tkzSetUpColors[background=fondpaille,text=Maroon]
```

FAQ

- `\tkzDrawPoint(A,B)` alors qu'il faut `\tkzDrawPoints`
- L'emploi de la virgule même dans un mode Mathématique $\$2,5\$$ nécessite d'être protégé dans un groupe TeX par exemple $\{\$2,5\$$.
- `\tkzDrawSegments[color = gray,style=dashed]{B,B' C,C'}` est une erreur. Seules, les macros qui définissent un objet utilisent des accolades.
- Si une erreur survient dans un calcul lors d'un passage de paramètres, alors il est préférable de faire ces calculs avant d'appeler la macro.
- Ne pas mélanger la syntaxe de `pgfmath` et celle de `fp.sty`. J'ai choisi souvent `fp.sty` mais si vous préférez `pgfmath` alors effectuez vos calculs avant le passage de paramètres.

Index

A	
<code>\add</code>	51
D	
<code>\draw (A)--(B)--(C);</code>	53
<code>\draw (A)--(B);</code>	51
E	
Environment	
scope.....	38, 39
L	
<code>\label options={below=6pt}</code>	21
O	
Operating System	
Windows XP.....	4
P	
Package	
fp.sty.....	37, 38, 84
pgfmath.....	84
pgfmath.sty.....	38
tkz-base.....	10
tkz-fct.....	10, 50
<code>\pgflinewidth</code>	42
T	
TeX Distributions	
MikTeX.....	4
<code>\textstyle</code>	17, 26
<code>\tikzset</code>	81, 82
<code>\tikzstyle</code>	81
<code>\tkzActivOff</code>	5
<code>\tkzActivoff</code>	36
<code>\tkzActivOn</code>	5
<code>\tkzActivon</code>	36
<code>\tkzAxeX</code>	14, 21, 22
<code>\tkzAxeX: options</code>	
frac.....	21
label options.....	21
label.....	21
orig.....	21
swap.....	21
trig.....	21
<code>\tkzAxeXY</code>	22
<code>\tkzAxeXY[<i><local options></i>]</code>	22
<code>\tkzAxeX[<i><local options></i>]</code>	21
<code>\tkzAxeY</code>	22
<code>\tkzAxeY[<i><local options></i>]</code>	22

<code>\tkzClip[space]</code>	35
<code>\tkzClip</code>	35
<code>\tkzClip: options</code>	
<code>space</code>	35
<code>\tkzClip[⟨local options⟩]</code>	35
<code>\tkzDefPoint</code>	37–39, 44, 60
<code>\tkzDefPoint: arguments</code>	
<code>a:r</code>	37
<code>x,y</code>	37
<code>\tkzDefPoint: options</code>	
<code>label</code>	37
<code>shift</code>	37
<code>\tkzDefPoints{0/0/0,2/2/A}</code>	39
<code>\tkzDefPoints</code>	39, 60
<code>\tkzDefPoints: arguments</code>	
$x_i/y_i/n_i$	39
<code>\tkzDefPoints[⟨local options⟩]{⟨x₁/y₁/n₁,x₂/y₂/n₂, ...}</code>	39
<code>\tkzDefPoint[⟨local options⟩](⟨x,y⟩){⟨name⟩}</code> ou <code>(⟨a:r⟩){⟨name⟩}</code>	37
<code>\tkzDefSetOfPoints</code>	60, 61
<code>\tkzDefSetOfPoints: arguments</code>	
x_n/y_n	60
<code>\tkzDefSetOfPoints: options</code>	
<code>prefix</code>	60
<code>\tkzDefSetOfPoints[⟨local options⟩]{⟨x₁/y₁,x₂/y₂,...,x_n/y_n}</code>	60
<code>\tkzDefShiftPoint</code>	40
<code>\tkzDefShiftPoint: arguments</code>	
<code>(a:r)</code>	40
<code>(x,y)</code>	40
<code>point</code>	40
<code>\tkzDefShiftPointCoord</code>	40, 41
<code>\tkzDefShiftPointCoord: arguments</code>	
<code>(a:r)</code>	40
<code>(x,y)</code>	40
<code>\tkzDefShiftPointCoord: options</code>	
<code>a,b</code>	40
<code>\tkzDefShiftPointCoord[⟨a,b⟩](⟨x,y⟩){⟨name⟩}</code> ou <code>(⟨a:r⟩){⟨name⟩}</code>	40
<code>\tkzDefShiftPoint[⟨Point⟩](⟨x,y⟩){⟨name⟩}</code> ou <code>(⟨a:r⟩){⟨name⟩}</code>	40
<code>\tkzDrawMark</code>	60, 65
<code>\tkzDrawMark: options</code>	
<code>prefix</code>	65
<code>\tkzDrawMarks</code>	65
<code>\tkzDrawMarks: options</code>	
<code>prefix</code>	65
<code>\tkzDrawMarks[⟨local options⟩](⟨⟨⟩list of points)</code>	65
<code>\tkzDrawMark[⟨local options⟩](⟨⟨⟩point)</code>	65
<code>\tkzDrawPoint(A,B)</code>	84
<code>\tkzDrawPoint</code>	42
<code>\tkzDrawPoint: arguments</code>	
<code>point</code>	42
<code>\tkzDrawPoint: options</code>	
<code>color</code>	42
<code>shape</code>	42
<code>size</code>	42

<code>\tkzDrawPoints(A,B,C)</code>	44
<code>\tkzDrawPoints</code>	44, 84
<code>\tkzDrawPoints: arguments</code>	
liste de points	44
<code>\tkzDrawPoints[<i><local options></i>](<i><liste></i>)</code>	44
<code>\tkzDrawPoint[<i><local options></i>](<i><point></i>)</code>	42
<code>\tkzDrawPolySeg</code>	53
<code>\tkzDrawPolySeg: arguments</code>	
(<i>pt₁, pt₂, pt₃</i>)	53
<code>\tkzDrawPolySeg[<i><local options></i>](<i><pt₁, pt₂, ..., pt_n></i>)</code>	53
<code>\tkzDrawSegment</code>	51
<code>\tkzDrawSegment: arguments</code>	
(<i>pt1, pt2</i>)	51
<code>\tkzDrawSegments[<i>color = gray, style=dashed</i>]{B, B' C, C'}</code>	84
<code>\tkzDrawSegments</code>	47, 52
<code>\tkzDrawSegments[<i><local options></i>](<i><pt1, pt2 pt3, pt4 ...></i>)</code>	52
<code>\tkzDrawSegment[<i><local options></i>](<i><pt1, pt2></i>)</code>	51
<code>\tkzDrawSetOfPoints</code>	61
<code>\tkzDrawSetOfPoints: options</code>	
prefix	61
<code>\tkzDrawSetOfPoints[<i><local options></i>]</code>	61
<code>\tkzDrawX</code>	14, 21, 22, 24
<code>\tkzDrawX: options</code>	
color	14
label	14
left space	14
noticks	14
right space	14
tickdn	14
tickup	14
tickwd	14
trig	14
<code>\tkzDrawXY</code>	18, 24
<code>\tkzDrawXY[<i><local options></i>]</code>	24
<code>\tkzDrawX[<i><local options></i>]</code>	14
<code>\tkzDrawY</code>	20, 22, 24
<code>\tkzDrawY: options</code>	
color	20
down space	20
label	20
noticks	20
ticklt	20
tickrt	20
tickwd	20
trig	20
up space	20
<code>\tkzDrawY[<i><local options></i>]</code>	20
<code>\tkzGrid</code>	27, 29–31
<code>\tkzGrid: arguments</code>	
(<i><x_A ; y_A></i>) (<i><x_B ; y_B></i>)	27
<code>\tkzGrid: options</code>	
color	27
line width	27

subxstep.....	27
subystep.....	27
sub.....	27
\tkzGrid[<i><local options></i>](<i><x_A ; y_A></i>) (<i><x_B ; y_B></i>).....	27
\tkzHLine{1}.....	73
\tkzHLine.....	73
\tkzHLine: arguments	
decimal number.....	73
\tkzHLine: options	
color	73
line width.....	73
style	73
\tkzHLines{1,4}.....	75
\tkzHLines.....	75
\tkzHLines: arguments	
list of values.....	75
\tkzHLines[<i><local options></i>]{ <i><list of values></i> }.....	75
\tkzHLine[<i><local options></i>]{ <i><decimal number></i> }.....	73
\tkzHTick{1}.....	78
\tkzHTick.....	78
\tkzHTick: arguments	
decimal number.....	78
\tkzHTick: options	
mark options.....	78
mark size.....	78
mark	78
\tkzHTicks{1}.....	78
\tkzHTicks.....	78
\tkzHTicks: arguments	
decimal number.....	78
\tkzHTicks[<i><local options></i>]{ <i><list of numbers></i> }.....	78
\tkzHTick[<i><local options></i>]{ <i><decimal number></i> }.....	78
\tkzInit.....	10, 11, 51
\tkzInit: options	
xmax.....	11
xmin.....	11
xstep.....	11
ymax.....	11
ymin.....	11
ystep.....	11
\tkzInit[<i><local options></i>].....	11
\tkzJoinSetOfPoints.....	62
\tkzJoinSetOfPoints: options	
prefix.....	62
\tkzJoinSetOfPoints[<i><local options></i>].....	62
\tkzLabelPoint(A){A ₁ }.....	45
\tkzLabelPoint(A,B,C).....	46, 47, 64
\tkzLabelPoint.....	45
\tkzLabelPoint: arguments	
point.....	45
\tkzLabelPoints.....	46
\tkzLabelPoints: arguments	
list of points.....	46

<code>\tkzLabelPoints[⟨local options⟩](⟨A₁,A₂,...⟩)</code>	46
<code>\tkzLabelPoint[⟨local options⟩](⟨point⟩){⟨label⟩}</code>	45
<code>\tkzLabelSegment(A,B){5}</code>	57
<code>\tkzLabelSegment</code>	57
<code>\tkzLabelSegment: arguments</code>	
<code>(pt1,pt2)</code>	57
<code>label</code>	57
<code>\tkzLabelSegment: options</code>	
<code>pos</code>	57
<code>\tkzLabelSegments</code>	58
<code>\tkzLabelSegments[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩)</code>	58
<code>\tkzLabelSegment[⟨local options⟩](⟨pt1,pt2⟩){⟨label⟩}</code>	57
<code>\tkzLabelX</code>	14, 16, 18, 21, 22, 24
<code>\tkzLabelX: options</code>	
<code>color</code>	16
<code>font</code>	16
<code>frac</code>	16
<code>label options</code>	16
<code>np off</code>	16
<code>orig</code>	16
<code>step</code>	16
<code>trig</code>	16
<code>\tkzLabelXY</code>	24
<code>\tkzLabelXY[⟨local options⟩]</code>	24
<code>\tkzLabelX[⟨local options⟩]</code>	16
<code>\tkzLabelY</code>	18, 20, 22, 24
<code>\tkzLabelY: options</code>	
<code>color</code>	20
<code>font</code>	20
<code>frac</code>	20
<code>step</code>	20
<code>\tkzLabelY[⟨local options⟩]</code>	20
<code>\tkzLegend</code>	69
<code>\tkzLegend: arguments</code>	
<code>mark/mark size//couleur/texte</code>	69
<code>style/line width/couleur/texte</code>	69
<code>\tkzLegend: options</code>	
<code>line</code>	69
<code>\tkzLegend[⟨local options⟩]{⟨mark/couleur/size/text⟩}</code>	69
<code>\tkzMarkSegment</code>	55
<code>\tkzMarkSegment: options</code>	
<code>color</code>	55
<code>mark</code>	55
<code>pos</code>	55
<code>size</code>	55
<code>\tkzMarkSegments</code>	52, 56
<code>\tkzMarkSegments[⟨local options⟩](⟨pt1,pt2 pt3,pt4 ...⟩)</code>	56
<code>\tkzMarkSegment[⟨local options⟩](⟨pt1,pt2⟩)</code>	55
<code>\tkzmathstyle</code>	17
<code>\tkzPointShowCoord</code>	48, 49
<code>\tkzPointShowCoord: arguments</code>	
<code>(⟨ref⟩)</code>	48
<code>\tkzPointShowCoord: options</code>	

noxdraw.....	48
noydraw.....	48
xlabel.....	48
xstyle.....	48
ylabel.....	48
ystyle.....	48
\tkzPointShowCoord[<i>local options</i>](<i>point</i>).....	48
\tkzRep.....	36
\tkzRep: options	
colorlabel.....	36
color.....	36
line width.....	36
posxlabel.....	36
posylabel.....	36
xlabel.....	36
xnorm.....	36
ylabel.....	36
ynorm.....	36
\tkzRep[<i>local options</i>].....	36
\tkzSetOfPoints.....	60
\tkzSetUpAxis.....	26
\tkzSetUpAxis.....	26
\tkzSetUpAxis: options	
font.....	26
line width.....	26
ticka.....	26
tickb.....	26
tickwd.....	26
\tkzSetUpAxis[<i>local options</i>].....	26
\tkzSetUpColors.....	83
\tkzSetUpMark.....	64
\tkzSetUpMark: options	
liste.....	64
\tkzSetUpMark[<i>local options</i>].....	64
\tkzSetUpPoint.....	47
\tkzSetUpPoint: options	
color.....	47
fill.....	47
shape.....	47
size.....	47
\tkzSetUpPoint[<i>local options</i>].....	47
\tkzText.....	67
\tkzText: options	
color.....	67
fill.....	67
opacity.....	67
text.....	67
\tkzText[<i>local options</i>](<i>point</i>){ <i>text</i> }.....	67
\tkzVLine{1}.....	76
\tkzVLine.....	76
\tkzVLine: arguments	
decimal number.....	76
\tkzVLine: options	

color	76
line width	76
style	76
\tkzVLines{1,4}	77
\tkzVLines	77
\tkzVLines: arguments	
list of values	77
\tkzVLines[<i>(local options)</i>]{ <i>(list of values)</i> }	77
\tkzVLine[<i>(local options)</i>]{ <i>(decimal number)</i> }	76
\tkzVTick{1}	79
\tkzVTick	79
\tkzVTick: arguments	
decimal number	79
\tkzVTicks{1,3}	80
\tkzVTicks	80
\tkzVTicks: arguments	
decimal number	80
\tkzVTicks[<i>(local options)</i>]{ <i>(decimal number)</i> }	80
\tkzVTick[<i>(local options)</i>]{ <i>(decimal number)</i> }	79
\tkzX	14
\tkzY	14

U

\usepackage[frenchb]{babel}	13
\usetkzobj{circles}	44, 72
\usetkzobj{lines}	56
\usetkzobj{polygons, lines, circles}	6
\usetkzobj{ <i>(liste d'objets)</i> }	72
\usetkzobj	72
\usetkzobj: arguments	
all	72
arcs	72
circles	72
lines	72
marks	72
points	72
polygons	72
protractor	72
sectors	72
segments	72
vectors	72