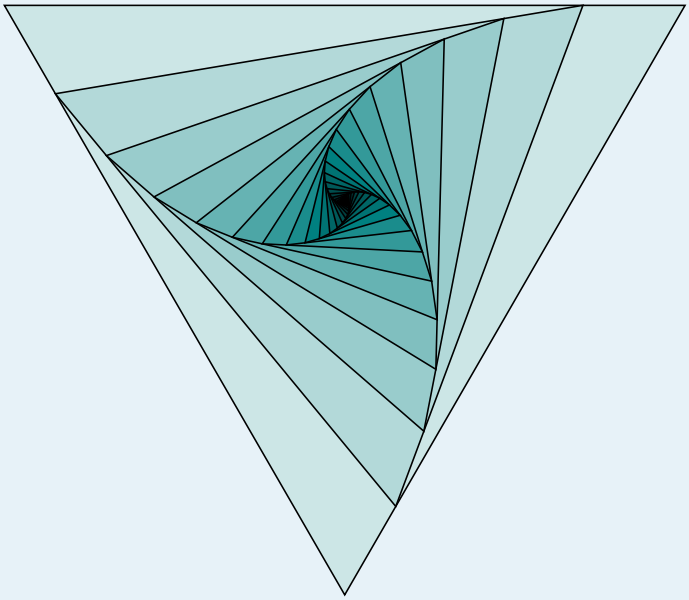
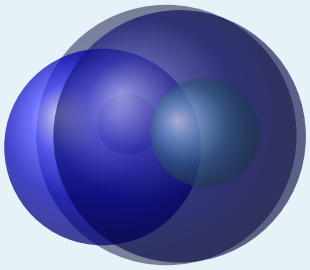
tkz-base 3.02c

AlterMundus





Alain Matthes

6 février 2020 Documentation V.3.02c

http://altermundus.fr

tkz-base

AlterMundus

Alain Matthes

tkz-base est un package basé sur TikZ permettant d'obtenir des graphiques le plus simplement possible. Il est la base sur laquelle sera construite une série de packages ayant comme point commun, la création de dessins utiles dans l'enseignement des mathématiques. Le rôle de **tkz-base** 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 nécessite la version égale ou supérieure à 3 de TikZ.

Je souhaite remercier **Till Tantau** pour avoir créé le merveilleux outil TikZ.

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 mon site : altermundus.fr

Vous pouvez envoyer vos remarques, et les rapports sur des erreurs que vous aurez constatées à l'adresse suivante : Alain Matthes.

Table des matières

1 Nouveautés et présentation

Ce package est le socle en particulier de **tkz-euclide** et de **tkz-fct**. Il fournit un repère cartésien qui sera défini avec la macro **\tkzInit**. Le package a été modifié et des transferts d' objets entre **tkz-base** et **tkz-euclide** ont été effectués. Dans le futur, les macros de définition seront isolées.

La nouveauté principale est le remplacement récent du package **fp** par **xfp**. L'apparition de celui-ci est un pas vers la version 3 de LaTeX. Le prochain pas sera la création d'un nouveau paquet

Voici quelques unes des modifications. Le package `tkz-euclide` apporte davantage de nouveautés.

— Amélioration du code

— Avec **tkz-euclide** charge tous les objets, donc plus besoin de placer **\usetkzobjall**.

— Correction de bugs

— La "bounding box" est désormais contrôlée dans chaque macro (enfin je l'espère) cela permet d'éviter l'utilisation de **\tkzInit** suivi de **\tkzClip**

— Ajout de macros concernant la "bounding box" : **\tkzSaveBB** **\tkzClipBB** etc.

— Logiquement la plupart des macros acceptent les options de TikZ. J'ai donc retiré les options "doublons".

— Suppression de l'option "label options"

— Les points aléatoires sont désormais dans **tkz-euclide** et la macro **\tkzGetRandPointOn** est remplacée par **\tkzDefRandPointOn**. Pour des raisons d'homogénéité, il faut récupérer les points avec **\tkzGetPoint**.

— Les options **end** et **start** qui permettaient de donner un label à une droite sont supprimées. Il faut désormais utiliser la macro **\tkzLabelLine**

— Introduction des librairies `quotes` et `angles` cela permet de donner un label à un point.même si je ne suis pas favorable à cette pratique.

— La notion de vecteur disparait pour tracer un vecteur il suffit de passer "->" en option de **\tkzDrawSegment**.

— Apparition de la macro **\usetkztool** qui permet de charger de nouveaux "outils".

2 Installation

est désormais sur le serveur du **CTAN**[1]. Si vous voulez tester une version beta, il vous suffit de placer les fichiers suivants dans un dossier texmf que votre système pourra trouver.

2.1 Fichiers présents

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

- **tkz-base.cfg**
- **tkz-base.sty**
- **tkz-lib-marks.tex**
- **tkz-obj-axes.tex**
- **tkz-obj-grids.tex**
- **tkz-obj-marks.tex**
- **tkz-obj-points.tex**
- **tkz-obj-rep.tex**
- **tkz-tools-arith.tex**
- **tkz-tools-base.tex**
- **tkz-tools-BB.tex**
- **tkz-tools-math.tex**
- **tkz-tools-misc.tex**
- **tkz-tools-modules.tex**
- **tkz-tools-print.tex**
- **tkz-tools-text.tex**
- **tkz-tools-utilities.tex**

Celui qui contient les principales macros est **tkz-tools-base.tex** , il est appelé par **tkz-base** 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.

De plus, TikZ est chargé avec les librairies suivantes :

```
\usetikzlibrary{angles, arrows, arrows.meta ,babel, calc, decorations, decorations.markings,
decorations.pathreplacing, intersections, patterns, plotmarks, positioning, quotes,
shapes.misc,shapes.misc, through}
```

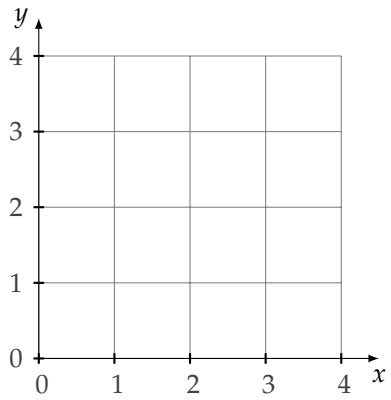
1. fait partie de TeXLive et tlmgr permet de l'installer, il fait aussi partie de MikTeX sous Windows

3 Compilation des exemples

3.1 Test de l'installation

Le code ci-dessous permet de tester votre installation de **tkz-base**. Je vous signale que , tout comme doit être présent ainsi que la version 3.01 (ou plus) de **pgf**. Tous les exemples ainsi que cette documentation ont été compilés à l'aide de LuaLaTeX

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



Remarques sur ce test

1. La compilation de ce document et des exemples est obtnue avec **luaLatex-dev**.
2. En principe, **tkz-base** n'est pas chargé par l'utilisateur, il sera chargé par un autre package comme **tkz-euclide** ou **tkz-fct**. **tkz-base** charge **numprint** avec l'option **autolanguage**, **xfp** et bien sûr TikZ.
3. TikZ était parfois allergique aux caractères actifs, il semblerait que la version 3. de pgf est réglé ces problèmes. En cas de difficulté, il est recommandé de charger la librairie `babel` avec **\usetikzlabry{babel}**. Une autre possibilité est de compiler avec **lualatex**.

3.2 Pourquoi xfp et numprint

xfp remplace désormais **fp** dans ce package. Un des avantages pour l'utilisateur est une syntace simplifiée. Il permet de gérer des calculs sur des grands nombres ou des très petits avec précision. Cela ralentit un peu la compilation, aussi il est préférable de ne pas en abuser. **xfp** est avant tout utilisé, pour obtenir des graduations correctes.

numprint était présent quand j'ai commencé à écrire cette série de packages, depuis **siunitx** 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.

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 **xfp** , 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 requièrent 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 non 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.

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 graphi-ques 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 [...].

5 Initialisation \tkzInit

5.1 La macro principale \tkzInit

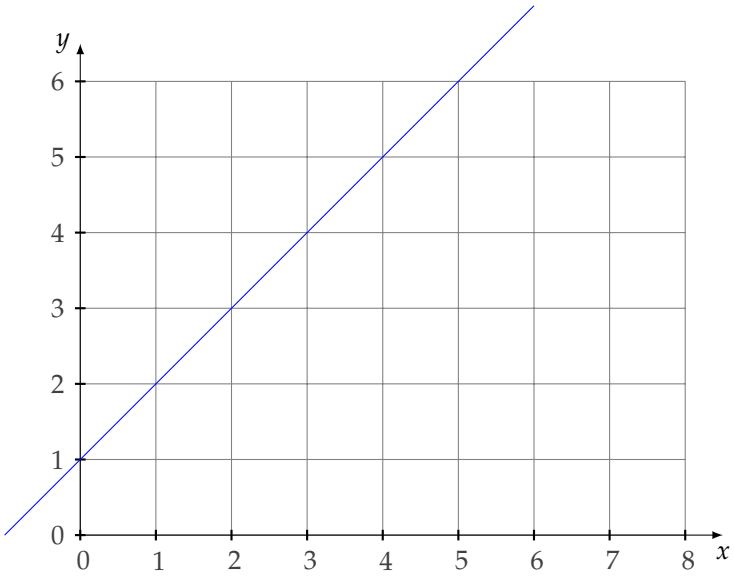
\tkzInit[⟨local options⟩]

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 allez 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.

5.1.1 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** mais attention aux unités ! il faut réserver cette possibilité que pour des cas exceptionnels.

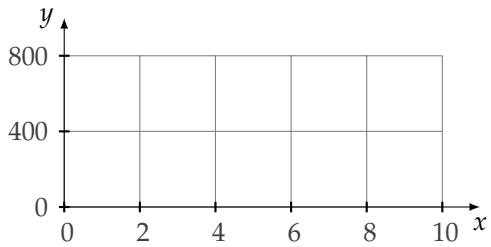



```
\begin{tikzpicture}
   \tkzInit[xmax=8,ymax=6]
   \tkzGrid
   \tkzAxeXY
   \draw[blue](-1,0)--(6,7);
\end{tikzpicture}
```

5.1.2 Rôle de xstep , ystep

☞ Attention, une graduation est représentée par 1 cm, sauf si vous redimensionnez 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

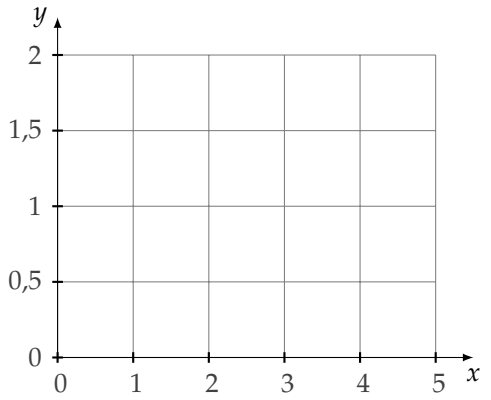
même ystep=400, il y a donc 2 cm entre 0 et 800. Il n'est pas possible d'utiliser les options de TikZ, x=... et y=....




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

\end{tikzpicture}
```

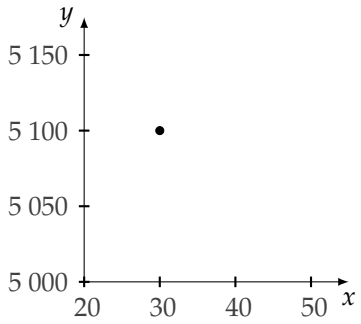
5.2 Autre exemple avec xstep et ystep




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

5.2.1 Origine personnalisée.

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



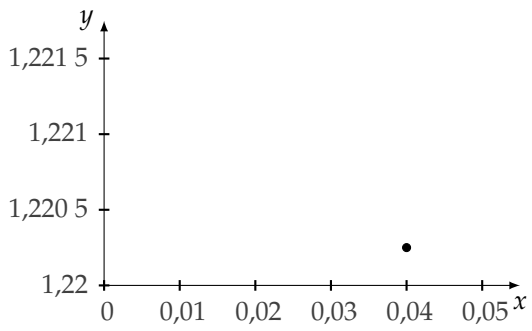

```
\begin{tikzpicture}
  \tkzInit[xmin=20,
           xmax=50,
           xstep=10,
           ymin=5000,
           ymax=5150,
           ystep=50]
  \tkzAxeXY
  \tkzDefPoint(30,5100){A}
  \tkzDrawPoint(A)
\end{tikzpicture}
```

5.2.2 Utilisation des décimaux

Il est préférable d'écrire les différents arguments relatifs à un axe avec le même nombre de décimales. numprint est utilisé pour afficher les graduations correctement.

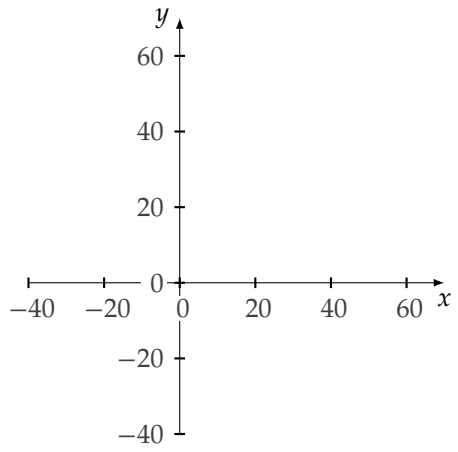
Dans l'exemple suivant, numprint utilise les conventions françaises pour l'écriture des nombres car j'ai utilisé :

```
\usepackage[french]{babel}
```




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

5.2.3 Valeurs négatives




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

6 Macros pour les axes

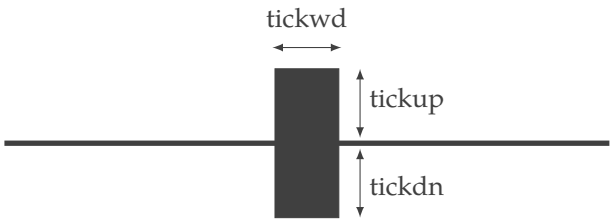
Attention ces macros ont été modifiées. Il est désormais plus faciles d'utiliser les stiles de TikZ . **\tkzDrawX** permet de tracer un axe, **\tkzLabelX** place des graduations et enfin dans les cas simples **\tkzAxeX** trace et gradue. Les options de TikZsont accessibles. Pour les graduations, il est 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 <>0 pi/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 dessus de l'axe

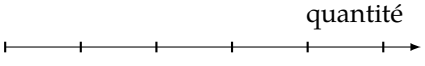


Cette macro permet de tracer l'axe des abscisses. Le plus important est de tester l'ensemble des options. Ci-dessus, vous avez les valeurs qui définissent un tick. Sinon les options de TikZ s'appliquent et en particulier **text**, **color**, **fill** et **font**.

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.

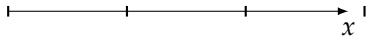

```
\begin{tikzpicture}
\tkzInit[xmax=0.4,xstep=0.1]
\tkzDrawX[text=blue,color=red,
          right=12pt,right space=1]
\end{tikzpicture}
```

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=4,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=4,ymin=-1,ymax=1]
  \tkzDrawX[trig=2]
\end{tikzpicture}
```

6.2 \tkzLabelX

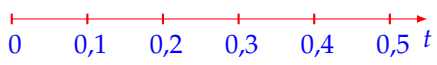
\tkzLabelX[⟨local options⟩]

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 <>0 graduations = num/frac "frac est un entier"
trig	0	si <>0 pi/trig "trig est un entier"
font	\textstyle	taille de la graduation.
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

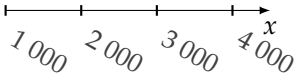
frac et **trig** sont des nombres entiers permettant de passer à une écriture fractionnaire ou trigonométrique.

6.2.1 Position des graduations




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

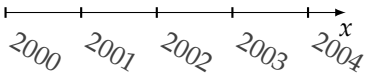
6.2.2 Position des graduations avec xlabel style




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

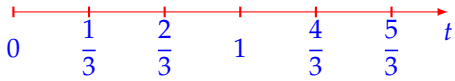
6.2.3 Dates avec np off

Pour les dates, il faut désactiver numprint.



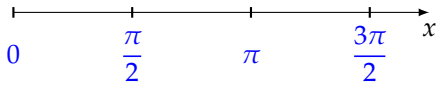

```
\begin{tikzpicture}
 \tkzInit[xmin=2000,xmax=2004]
 \tkzDrawX
 \tikzset{xlabel style/.append style={rotate=-30}}
 \tkzLabelX[np off,below right=3 pt,inner sep =1pt]
\end{tikzpicture}
```

6.2.4 frac




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

6.2.5 trig

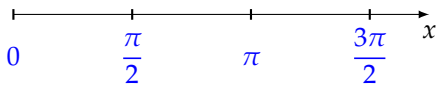



```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=5,ymin=-1,ymax=1]
  \tkzDrawX[trig=2]
  \tkzLabelX[trig=2,text=blue,below = 8pt]
\end{tikzpicture}
```

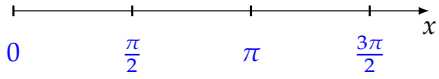
6.2.6 Taille des graduations

Deux possibilités. Il est possible de définir le style employé par défaut pour le mode math :

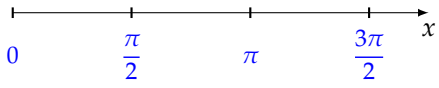
```
\let\tkzmathstyle\textstyle
```



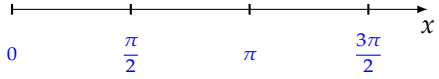

```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=5,ymin=-1,ymax=1]
  \tkzDrawX[trig=2]
  \tkzLabelX[trig=2,text=blue,below = 8pt]
\end{tikzpicture}
```




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



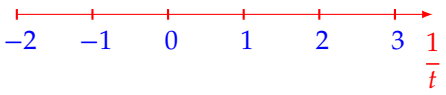

```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=5,ymin=-1,ymax=1]
  \tkzDrawX[trig=2]
  \tkzLabelX[trig=2,text=blue,
           below = 8pt,node font=\small]
\end{tikzpicture}
```




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

6.2.7 Couleur des graduations

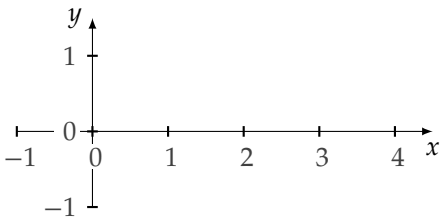
Il s'agit ici de bien utiliser les options **color**, **text** et **fill**




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

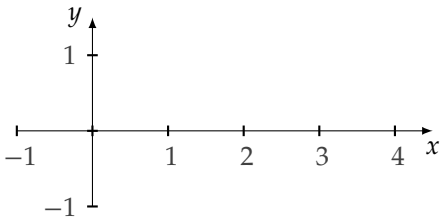
6.2.8 Tracés des axes avant la graduation

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



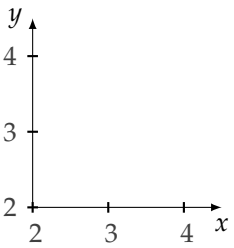

```
\begin{tikzpicture}
\tkzInit[xmin = -1,xmax = 4,
         ymin = -1,ymax = 1]
\tkzDrawXY \tkzLabelX  \tkzLabelY
\end{tikzpicture}
```

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



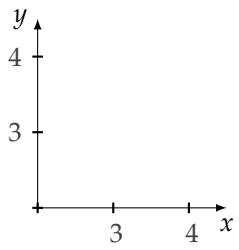

```
\begin{tikzpicture}
   \tkzInit[xmin = -1,xmax = 4,
            ymin = -1,ymax = 1]
   \tkzLabelX[orig=false]
   \tkzLabelY[orig=false]
   \tkzDrawXY
\end{tikzpicture}
```

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}
```

6.3 \tkzAxeX

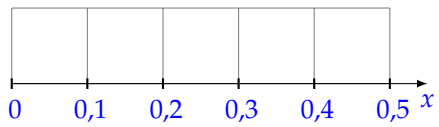
\tkzAxeX[⟨local options⟩]

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 que dans les cas simples.

options	défaut	définition
label	x	nom attribué au label
trig	0	graduation fraction de π
frac	0	graduation fractionnaire, de dénominateur « frac »
swap	false	permet de lancer **\tkzLabelX** avant **\tkzDrawX**

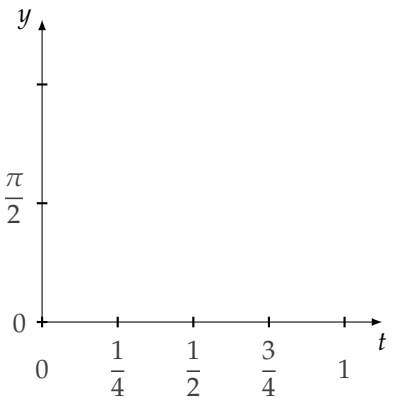
text définit la couleur des graduations.

6.3.1 exemple avec \tkzAxeX




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

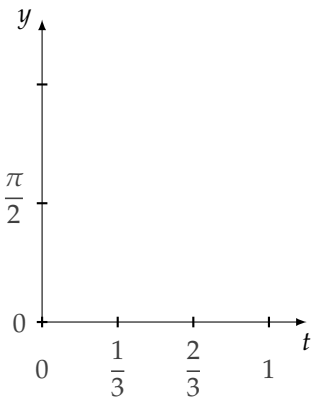
6.3.2 Usage de pi et \tkzAxeX




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

6.3.3 Option frac et trig

Dans cet exemple, on positionne le label *t* ainsi que les graduations. **\below=10pt** sert à placer les graduations en-dessous.




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

6.4 \tkzDrawY

\tkzDrawY[⟨local options⟩]

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	*x*	nom attribué au label
trig	0	si <>0 pi/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

6.5 \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 <>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.

6.6 \tkzAxeY

\tkzAxeY[⟨local options⟩]

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

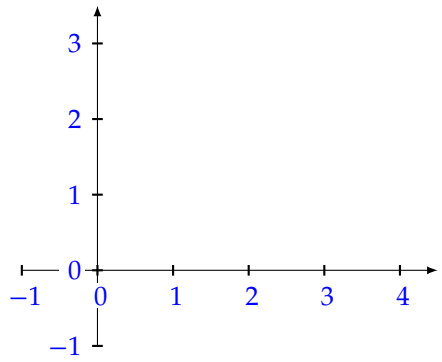
6.7 \tkzAxeXY

\tkzAxeXY[⟨local options⟩]

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

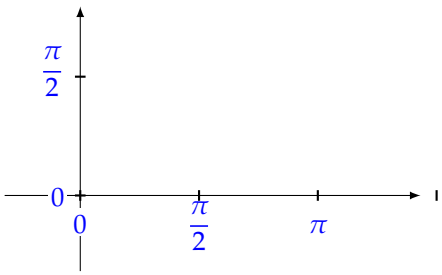
Il est nécessaire d'utiliser des options communes comme dans l'exemple ci-dessous, mais cela signifie que les mêmes options sont appliquées aux deux macros. Ainsi il n'est pas possible de modifier **label**

6.7.1 Couleur des axes, des graduations



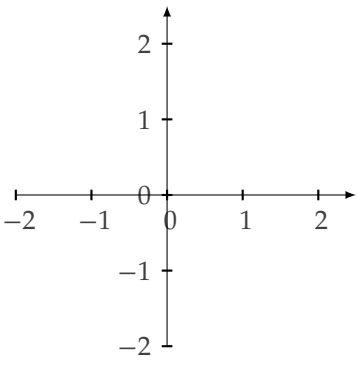

```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=4,ymin=-1,ymax=3]
  \tkzAxeXY[label={},text=blue]
\end{tikzpicture}
```

6.7.2 Option {label={}}




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

6.7.3 Option swap




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

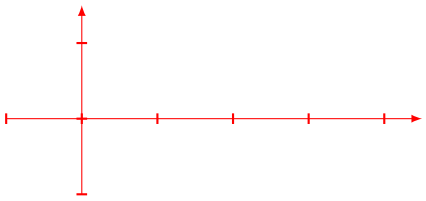
6.8 \tkzDrawXY

\tkzDrawXY[⟨local options⟩]

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

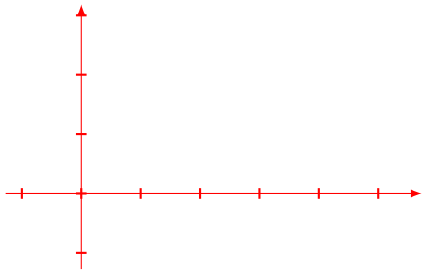
Il est nécessaire d'utiliser des options communes comme dans l'exemple ci-dessous

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

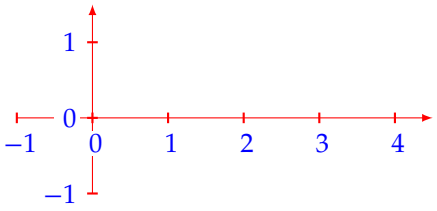
\tkzLabelXY[⟨local options⟩]

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



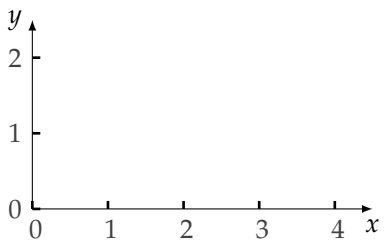

```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=4,ymin=-1,ymax=1]
  \tkzDrawXY[label={},color=red]
  \tkzLabelXY[text=blue]
\end{tikzpicture}
```

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	\textstyle	taille de la graduation.

6.10.1 Modification des axes par défaut




```
\begin{tikzpicture}[scale=1]
 \tkzInit[ymax=2,xmax=4]
 \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]
```

7 Utilisation de \tkzGrid

\tkzGrid[⟨local options⟩](⟨x_A ; y_A⟩) (⟨x_B ; y_B⟩)

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.

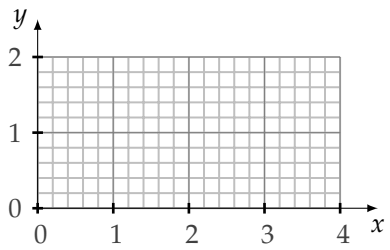
options	défaut	définition
(⟨x_A ; y_A⟩) (⟨x_B ; y_B⟩)	(xmin,ymin)(xmax,ymax)	trace une grille

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.

7.0.1 \tkzGrid et l'option sub

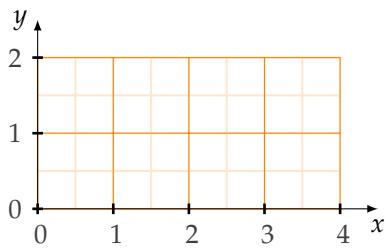
L'option **sub** permet d'afficher une grille secondaire plus fine. Il est préférable de lancer **\tkzGrid** en premier, pour éviter que la grille se superpose à d'autres éléments .




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

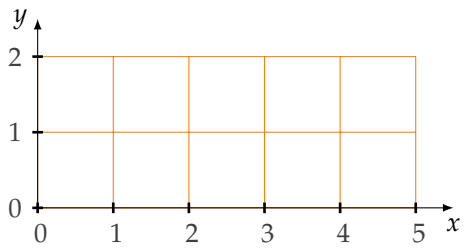
7.0.2 Option sub

L'option **sub** permet d'afficher un grille secondaire plus fine. Certains paramètres sont mofiables.



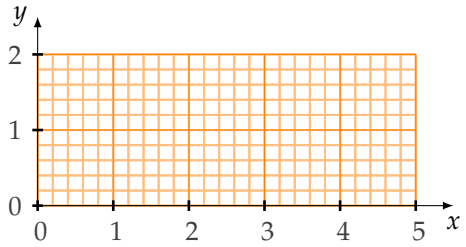

```
\def\tkzCoeffSubColor{20}
\def\tkzCoeffSubLw{0.2}
\begin{tikzpicture}
 \tkzInit[xmax=4, ymax=2]
 % on peut modifier le pas pour la seconde grille
 \tkzGrid[sub,color=orange,
          subxstep=.5,subystep=.5]
  \tkzAxeXY
\end{tikzpicture}
```

7.0.3 Presque par défaut



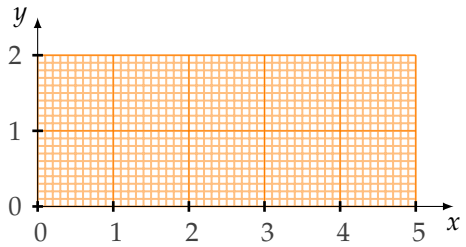

```
\begin{tikzpicture}
  \tkzInit[xmax=5,ymax=2]
  \tkzGrid[color=orange]
  \tkzAxeXY
\end{tikzpicture}
```

7.0.4 Sous grille en plus, option sub



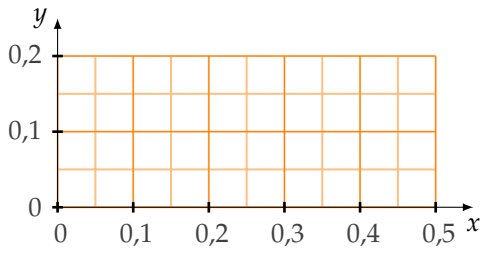

```
\begin{tikzpicture}
  \tkzInit[xmax=5,ymax=2]
  \tkzGrid[sub,color=orange]
  \tkzGrid[color=orange]
  \tkzAxeXY
\end{tikzpicture}
```

7.0.5 Changement de maille



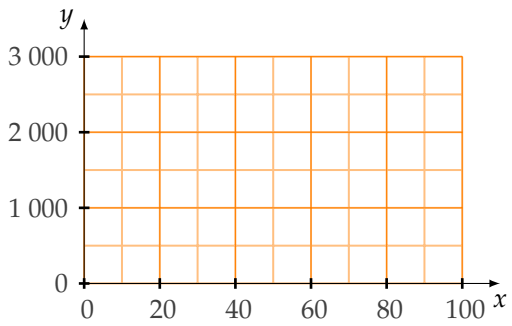

```
\begin{tikzpicture}
  \tkzInit[xmax=5,ymax=2]
  \tkzGrid[color    = orange,
           sub,
           subxstep = 0.1,
           subystep = 0.1]
   \tkzAxeXY
\end{tikzpicture}
```

7.0.6 Option xstep, xstep, subxstep et subystep




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

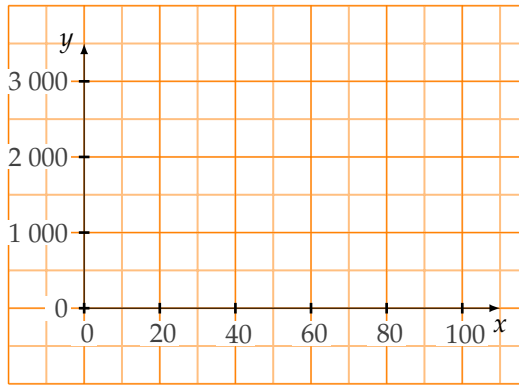
7.0.7 Avec des intervalles importants




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

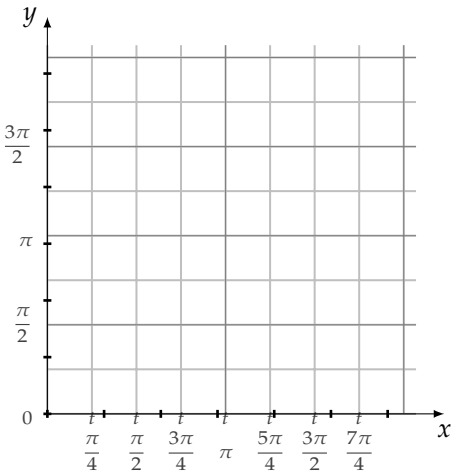
7.0.8 \tkzGrid et les arguments

La grille peut avoir une taille quelconque.



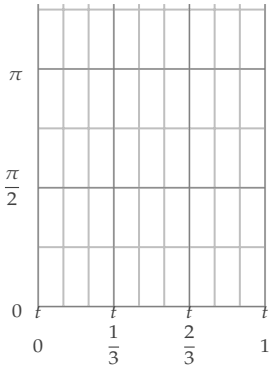

```
\begin{tikzpicture}
   \tkzInit[xmax=100,xstep=20,
            ymax=3000,ystep=1000]
   \tkzGrid[sub,subxstep=10,
            subystep=500,
            color=orange]
            (-20,-1000)(115,4000)%
  \tkzAxeXY
\end{tikzpicture}
```

7.0.9 Usage de pi avec \tkzGrid



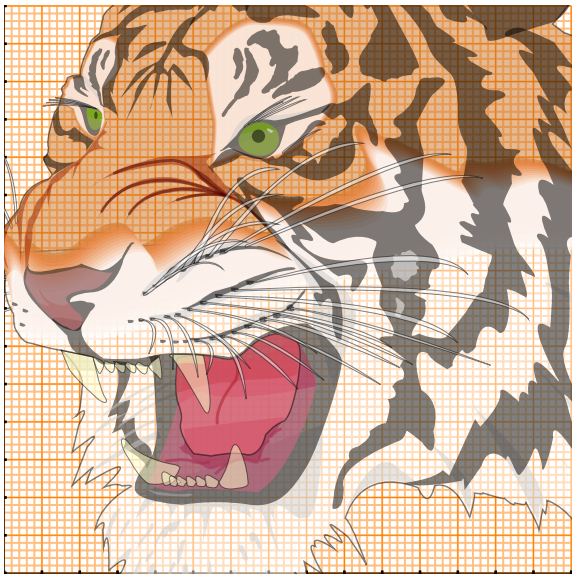

```
\begin{tikzpicture}[scale=.75]
   \tkzInit[xmax=6.5,ymax=6.5]
   \tkzGrid[xstep=pi,ystep=pi/2,sub,
            subxstep=pi/4,subystep=pi/4]
   \tkzLabelX[label=$t$,orig=false,trig=4,
            below=6pt,font=\scriptsize]
   \tkzLabelY[trig=2,font=\scriptsize]
   \tkzDrawXY
\end{tikzpicture}
```

7.0.10 Options frac 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]
   \tkzLabelX[label=$t$,orig=false,frac=3,
            below=6pt,font=\scriptsize]
   \tkzLabelY[trig=2,font=\scriptsize]
\end{tikzpicture}
```

7.0.11 Utilisation d'une grille de répérage



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

8 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 **xfp** qui s'en charge. On peut utiliser les coordonnées cartésiennes ou polaires.

8.1 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

Toutes les options de TikZ que l'on peut appliquer à **coordinate**, sont applicables (enfin je l'espère !) comme par exemple l'option **label** définit avec la librairie **quotes**.

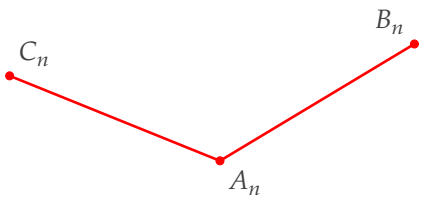
8.1.1 Utilisation de shift

shift permet de placer les points par rapport à un autre.

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

8.1.2 Placer un label avec la librairie quotes

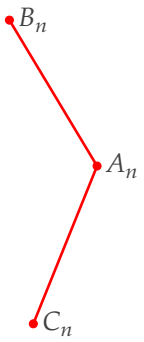
Je préfère ne pas mélanger les opérations et utiliser **\tkzLabelPoint** pour placer les labels. voir la section 17.10.4 The Quotes Syntax.




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

8.1.3 Rotation avec shift et scope

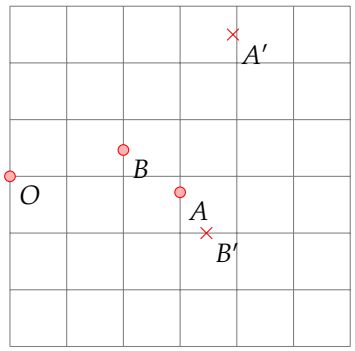
Préférable pour effectuer une rotation, est d'utiliser un environnement **scope**.




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

8.1.4 Formules et coordonnées

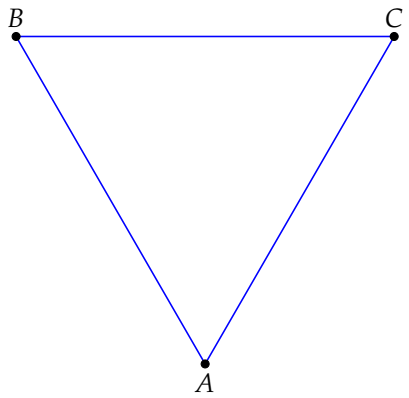
Il faut ici respecter la syntaxe de **xfp**. Il est toujours possible de passer par **pgfmath** mais dans ce cas, il faut calculer les coordonnées avant d'utiliser la macro **\tkzDefPoint**.




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

8.1.5 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.



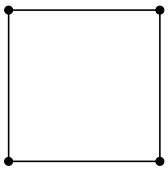

```
\begin{tikzpicture}[scale=1]
 \begin{scope}[rotate=30]
  \tkzDefPoint(2,3){A}
  \begin{scope}[shift=(A)]
     \tkzDefPoint(90:5){B}
     \tkzDefPoint(30:5){C}
  \end{scope}
 \end{scope}
\tkzDrawSegments[color=blue](A,B B,C C,A)
\tkzDrawPoints(A,B,C)
\tkzLabelPoints[above](B,C)
\tkzLabelPoints[below](A)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

8.2 Définition de points en coordonnées cartésiennes : \tkzDefPoints

\tkzDefPoints[⟨local options⟩]{⟨$x_1/y_1/n_1$,$x_2/y_2/n_2$, ...⟩}

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

arguments	exemple
$x_i/y_i/n_i$	**\tkzDefPoints{0/0/O,2/2/A}**



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

8.3 Point relativement à un autre : \tkzDefShiftPoint

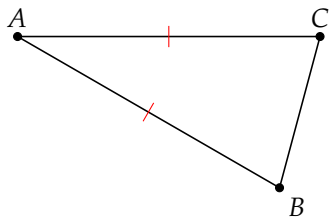
\tkzDefShiftPoint[⟨Point⟩](⟨x,y⟩){⟨name⟩} ou (⟨a:r⟩){⟨name⟩}

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
point	no default	\tkzDefShiftPoint[A](0:4){B}

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

8.3.1 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}
```

8.4 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'oigine.

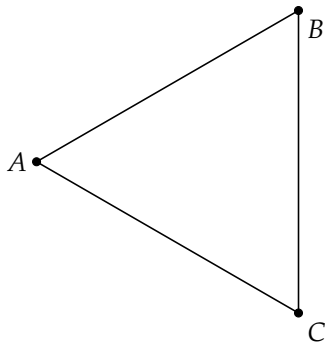
arguments	défaut	définition
(x,y)	no default	x et y sont deux dimensions, par défaut en cm.
(a:r)	no default	a est un angle en degré, r une dimension

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

L'option est obligatoire

8.4.1 Triangle équilatéral avec \tkzDefShiftPointCoord

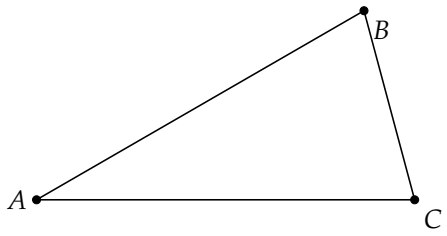
Voyons comment obtenir un triangle équilatéral (il y a beaucoup plus simple)




```
\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)
 % or \tkzDrawPolygon
  \tkzDrawPoints(A,B,C)
  \tkzLabelPoints(B,C)
  \tkzLabelPoint[left](A){$A$}
\end{tikzpicture}
```

8.4.2 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}
```

8.5 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	2 pt	taille du disque
color	black	la couleur par défaut peut être changée

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

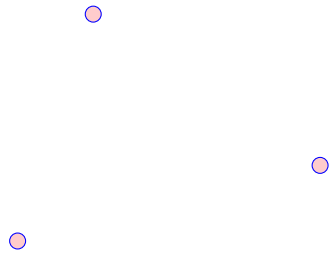
8.5.1 Style des points par défaut


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

8.5.2 Modification du style

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

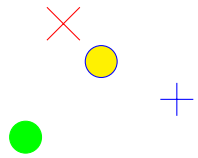
```
\tikzset{point style/.style={draw         = \tkz@euc@pointcolor,
                             inner sep    = 0pt,
                             shape        = \tkz@euc@pointshape,
                             minimum size = \tkz@euc@pointsize,
                             fill         = \tkz@euc@pointcolor!50}}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



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

8.5.3 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.

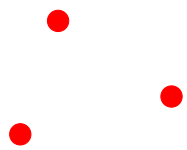
8.6 Tracer des points \tkzDrawPoints

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

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

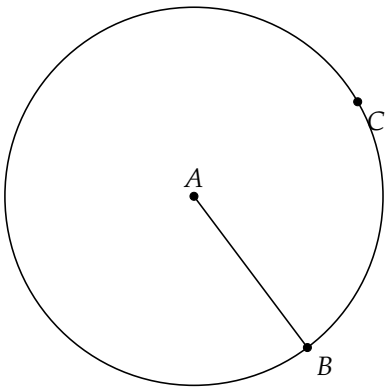
Attention au « s » final, un oubli entraîne des erreurs en cascade si vous tentez de tracer des points multiples. Les options sont les mêmes que pour la macro précédente.

8.6.1 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}
```

8.6.2 Exemple plus complexe




```
\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}
```

8.7 Ajouter un label à un point \tkzLabelPoint

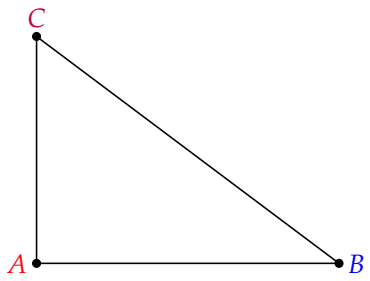
Il est possible d'ajouter plusieurs labels à un même point en utilisant plusieurs fois cette macro.

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

arguments	exemple	
point	**\tkzLabelPoint(A){A_1}**	
options	défaut	définition
TikZ options		couleur,position etc.

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

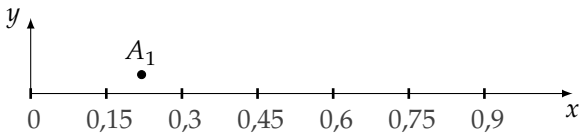
8.7.1 Exemple avec \tkzLabelPoint




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

8.7.2 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}
```

8.8 Ajouter des labels aux points \tkzLabelPoints

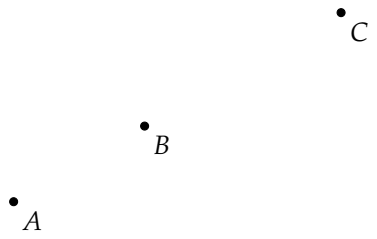
Il est possible de placer plusieurs labels rapidement quand les références des points sont identiques aux labels et quand les labels sont placés de la même manière par rapport aux points. Par défaut, c'est **below right** qui a été choisi.

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

arguments	exemple	résultat
list of points	**\tkzLabelPoint(A,B,C)**	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.

8.8.1 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}
```

8.9 Position automatique des labels \tkzAutoLabelPoints

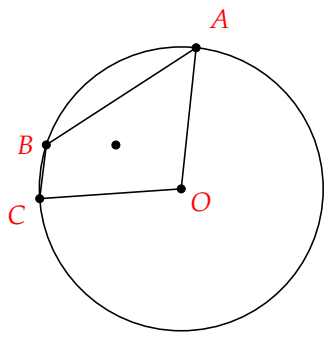
Le label d'un point est placé suivant une direction définie par un centre et un point **center**. La distance au point est déterminé par un pourcentage de la distance entre le centre et le point. Ce pourcentage est donné par **dist**.

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

arguments	exemple	résultat
list of points	**\tkzLabelPoint(A,B,C)**	Affichage de A, B et C

8.9.1 Exemple avec \tkzAutoLabelPoints

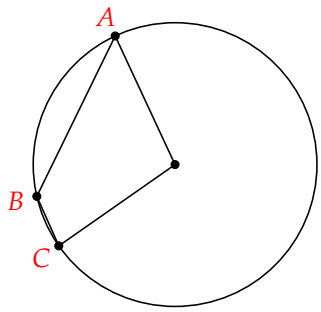
Ici les points sont positionnés par rapport au centre de gravité de A, B, Cet$O$




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

8.9.2 Exemple avec \tkzAutoLabelPoints

Cette fois la référence est *O* et la distance est par défaut de 0.15




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

8.10 Style des points avec \tkzSetUpPoint

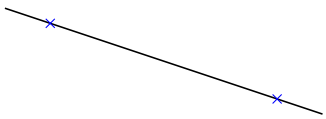
Il est important de comprendre que la taille d'un point dépend de la taille d'une ligne.

\tkzSetUpPoint[⟨local options⟩]

options	défaut	définition
shape	circle	possible : circle, cross, cross out
size	current	la taille du point est size * line width
color	current	exemple \tkzLabelPoint(A,B,C)
fill	current!50	exemple \tkzLabelPoint(A,B,C)

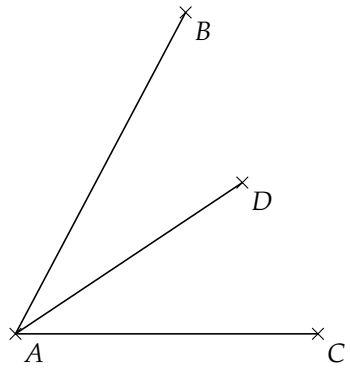
Il s'agit d'une macro permettant de choisir un style pour les points. La macro **\tkzDrawSegments** est décrite ici.

8.10.1 Exemple simple avec \tkzSetUpPoint



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

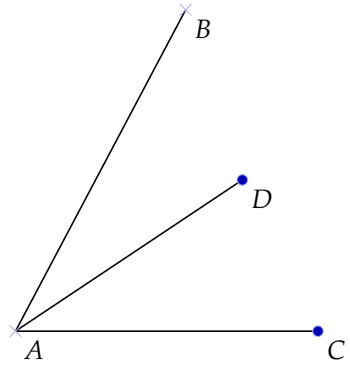
8.10.2 Exemple avec \tkzSetUpPoint




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

8.10.3 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(02.25,04.25){B}
  \tkzDefPoint(4,0){C}
  \tkzDefPoint(3,2){D}
  \tkzDrawSegments(A,B A,C A,D)
{\tkzSetUpPoint[shape=cross out,
            fill= blue!70!black!!50,
            size=4,color=blue!70!black!30]
  \tkzDrawPoints(A,B)}
  \tkzSetUpPoint[fill= blue!70!black!!50,size=4,
              color=blue!70!black!30]
   \tkzDrawPoints(C,D)
  \tkzLabelPoints(A,B,C,D)
\end{tikzpicture}
```

8.11 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.

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

argument	exemple	explication
(⟨ref⟩)	\tkzPointShowCoord(A)	Montre les coordonnées du point A

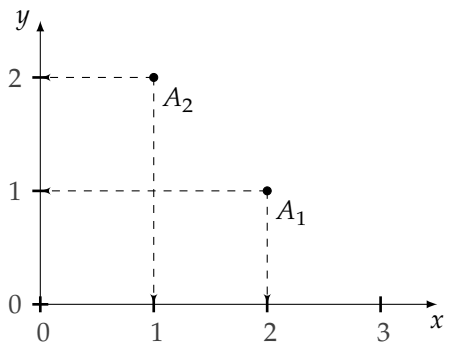
option	défaut	explication
xlabel	empty	label pour l'abscisse
xstyle	empty	style pour le node du label de l'abscisse
noxdraw	false	booléen pour ne pas tracer de flèche vers $(x'x)$

8.11.1 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}}
```

8.11.2 Exemple avec \tkzPointShowCoord

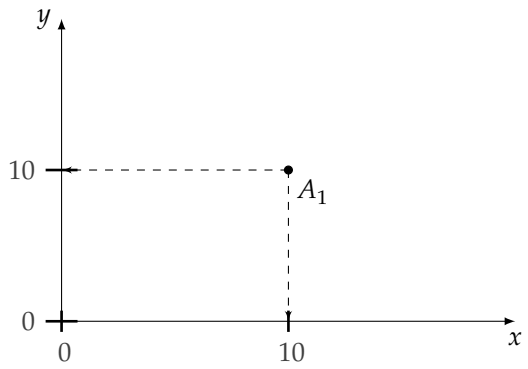
Sans les options, on n'obtient que les flèches.




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

8.11.3 Exemple avec \tkzPointShowCoord et xstep

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




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

8.12 \tkzDefSetOfPoints

: 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.

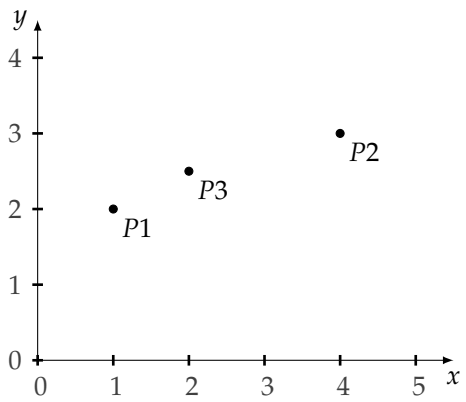
C'est ce qu'on appelle fréquemment « nuage de points». La différence par rapport à la macro **\tkzDefPoints**, c'est que la référence aux points est donnée par un préfixe (par défaut tkzPt) et le numéro du point. Les points ne sont pas tracés.

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

arguments	défaut	définition
x_n/y_n	no default	Liste de couples x_n/y_n séparés par des virgules

options	défaut	définition
prefix	tkzPt	préfixe pour les noms des points

8.12.1 Création d'un nuage avec \tkzDefSetOfPoints




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

9 Utilisation des styles

9.1 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**.

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

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

Vous pouvez les modifier, par exemple :

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

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

9.2 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. Ceci peut se faire dans **tkz-base.cfg** ou bien dans votre code.

```
\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,xmax=5]
  \tkzDrawX
  \tkzLabelX
\end{tikzpicture}
```

9.3 Macros de configuration

— \tkzSetUpPoint
— \tkzSetUpAxis

10 Gestion de la bounding box

La bounding box initiale après usage de la macro **\tkzInit** est définie par le rectangle basé sur les points (0,0) et (10,10). La macro **\tkzInit** permet de modifier cette bounding box initiale en utilisant les arguments (**xmin**, **xmax**, **ymin**, et **ymax**). Bien sûr tout tracé extérieur modifie la bounding box. TikZtient à jour cette bounding box. Il est possible d'influer sur ce comportement soit directement avec des commandes ou des options de TikZcomme une commande comme **\useasboundingbox** ou l'option **use as bounding box**. Une conséquence possible est de réserver une boite pour une figure mais la figure peut déborder de la boite et se repandre au-dessus du texte principal. La commande suivante **\pgfresetboundingbox** permet d'effacer une bounding box et d'en établir une nouvelle.

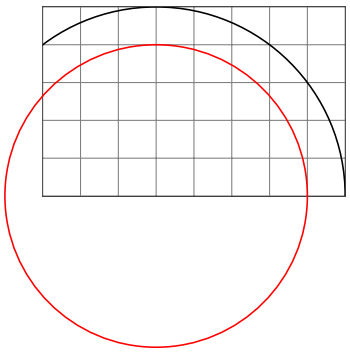
10.1 tkzShowBB

La macro la plus simple.

\tkzShowBB[⟨local options⟩]

Cette macro permet de visualiser la bounding box. Un cadre rectangulaire entoure celle-ci. Cette macro accepte les options de TikZ.

10.1.1 Exemple 2 avec \tkzShowBB



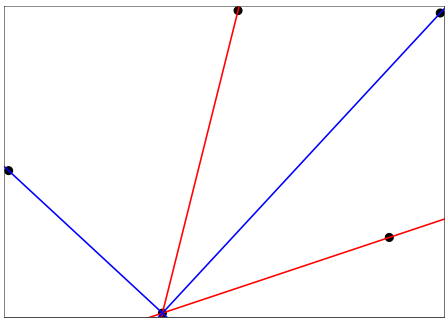
```
\begin{tikzpicture}[scale=.5]
  \tkzInit[ymax=5,xmax=8]
  \tkzGrid
  \tkzDefPoint(3,0){A}
   \begin{scope}
    \tkzClipBB
    \tkzDrawCircle[R](A,5 cm)
     \tkzShowBB
   \end{scope}
\tkzDrawCircle[R,red](A,4 cm)
\end{tikzpicture}
```

10.2 tkzClipBB

\tkzClipBB

Il s'agit de limiter les futures constructions à la bounding box actuelle.

10.2.1 Exemple avec \tkzClipBB et les bissectrices



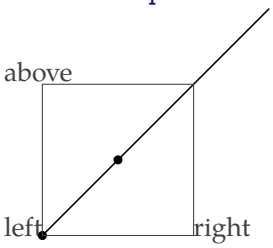
```
\begin{tikzpicture}
\tkzInit[xmin=-3,xmax=6, ymin=-1,ymax=6]
\tkzDefPoint(0,0){O}\tkzDefPoint(3,1){I}
\tkzDefPoint(1,4){J}
\tkzDefLine[bisector](I,O,J) \tkzGetPoint{i}
\tkzDefLine[bisector out](I,O,J) \tkzGetPoint{j}
\tkzDrawPoints(O,I,J,i,j)
\tkzClipBB
\tkzDrawLines[add = 1 and 2,color=red](O,I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

10.3 tkzSetBB

\tkzSetBB(⟨x_A ; y_A⟩) (⟨x_B ; y_B⟩) ou bien (⟨A⟩) (⟨B⟩)

Cette macro permet de définir le rectangle ayant pour coordonnées $(x_A\,;y_A)$ et $(x_B\,;y_B)$ comme la nouvelle bounding box.

10.3.1 Exemple avec \tkzShowBB

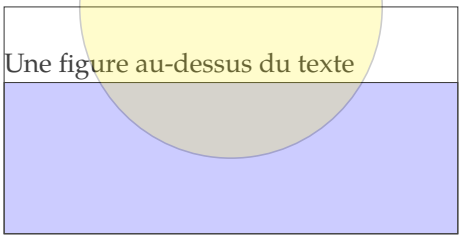



```
above\\
left
\begin{tikzpicture}
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(3,3){B}
  \tkzDefPoint(1,1){C}
  \tkzSetBB(A)(2,2)
  \tkzDrawSegment(A,B)
  \tkzDrawPoints(A,C)
  \tkzShowBB
\end{tikzpicture}right
```

10.4 tkzSaveBB

\tkzSaveBB

Cette macro permet de sauvegarder la bounding box, autrement dit elle enregistre les coordonnées de deux points qui définissent un rectangle.



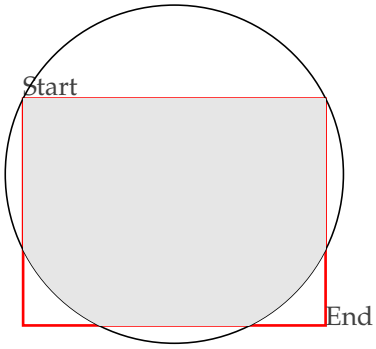

```
Une figure au-dessus du texte\\
\begin{tikzpicture}
 \begin{scope}
   \tkzSetBB(0,0)(6,2) \tkzShowBB[fill=blue!20]
   \tkzSaveBB
 \end{scope}
  \tkzDefPoint(3,3){A}\tkzShowBB
  \tkzDrawCircle[R,fill=yellow,opacity=.2](A,2cm)
  \tkzRestoreBB
\end{tikzpicture}
```

10.5 tkzRestoreBB

\tkzRestoreBB

Cette macro récupère la sauvegarde de la bounding box. Comme vous le constater, la figure déborde de la boîte. La bounding box a été réduite.

10.5.1 Exemple d'utilisation de \tkzRestoreBB




```
   \vspace{ 2cm}
Start\\
\begin{tikzpicture}
 \tkzDefPoint(-2,-2){A}
 \tkzDefPoint(2,1){B}
 \tkzDefPoint(0,0){O}
 \tkzSaveBB
 \tkzShowBB[red,line width=1pt]
 \tkzRestoreBB
 \tkzDrawCircle(O,B)
 \tkzClipBB
 \tkzFillCircle[gray!20](O,B)
\end{tikzpicture}
End
```

10.6 tkzClip

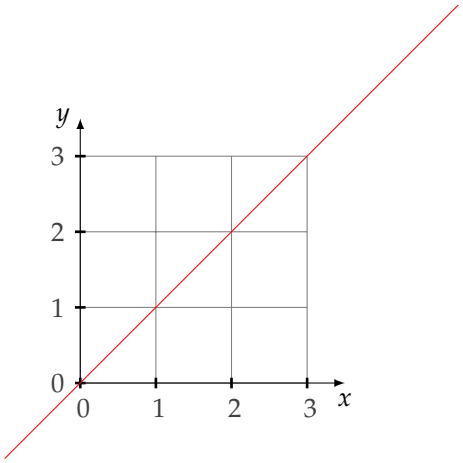
\tkzClip[⟨local options⟩]

Le rôle de cette macro est de rendre invisible ce qui est hors du rectangle défini par $(xmin\ ;\ ymin)$ et $(xmax\ ;\ ymax)$.

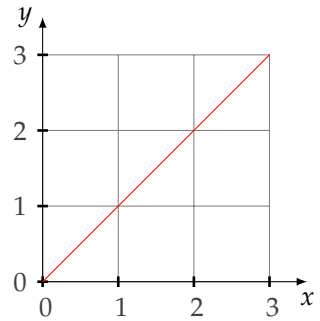
options	défaut	définition
space	1	valeur ajoutée à droite, à gauche, en bas et en haut du background

Le rôle de l'option **space** est d'agrandir la partie visible du dessin. Cette partie devient le rectangle définit par $(xmin - space\ ;\ ymin - space)$ et $(xmax + space\ ;\ ymax + space)$. **space** peut être négatif ! L'unité est le cm et ne doit pas être indiquée.

10.6.1 Premier exemple avec \tkzClip




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

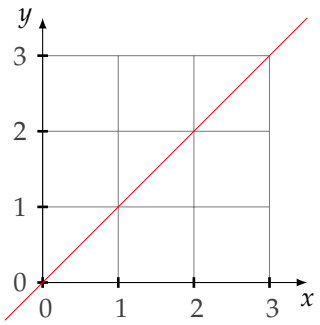



```
\begin{tikzpicture}
 \tkzInit[xmax=3, ymax=3]
 \tkzGrid
 \tkzAxeXY
 \tkzClip
 \draw[red] (-1,-1)--(5,5);
\end{tikzpicture}
```

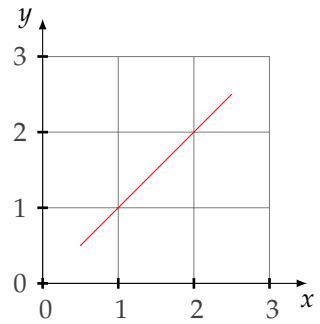
Il est possible d'ajouter un peu d'espace **\tkzClip[space]**

10.6.2 \tkzClip et l'option space

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

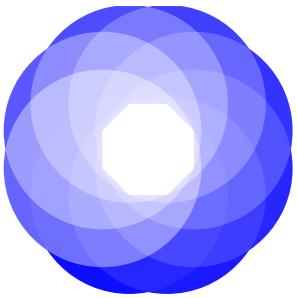



```
\begin{tikzpicture}
 \tkzInit[xmax=3, ymax=3]
 \tkzGrid  \tkzAxeXY
 \draw[red] (-0.5,-0.5)--(3.5,3.5);
\end{tikzpicture}
```




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

10.7 style tkzreverseclip



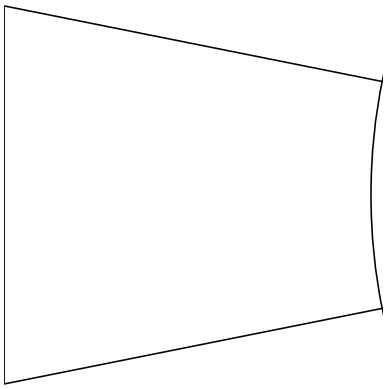
```
\begin{tikzpicture}[scale=.5]
  \tkzInit[xmin=-5,xmax=5,ymin=-5,ymax=5]
  \pgfinterruptboundingbox
  \tkzDefPoints{-.5/0/P1,.5/0/P2}
  \foreach \i [count=\j from 3] in {2,...,7}{%
      \tkzDefShiftPoint[P\i]({45*(\i-1)}:1 cm){P\j}
  }
  \endpgfinterruptboundingbox
   \tkzClipOutPolygon(P1,P2,P3,P4,P5,P6,P7,P8)
   \tkzCalcLength[cm](P1,P5)\tkzGetLength{r}
  \begin{scope}[blend group=screen]
     \foreach \i in {1,...,8}{%
       \pgfmathparse{100-5*\i}
       \tkzFillCircle[R,color=blue!%
      \pgfmathresult](P\i,\r)
      }
    \end{scope}
\end{tikzpicture}
```

10.8 option tikz : trim left or right

voir le **pgfmanual**

10.9 Commandes de TikZ \pgfinterruptboundingbox et \endpgfinterruptboundingbox

Cette commande interrompt temporairement le calcul de la boîte et configure une nouvelle boîte



```
\begin{tikzpicture}
\tkzDefPoint(0,5){A}\tkzDefPoint(5,4){B}
\tkzDefPoint(0,0){C}\tkzDefPoint(5,1){D}
\pgfinterruptboundingbox
   \tkzInterLL(A,B)(C,D)\tkzGetPoint{I}
\endpgfinterruptboundingbox
\tkzClipBB
   \tkzDrawCircle(I,B)
\tkzDrawSegments(A,B C,D A,C)
\end{tikzpicture}
```

11 Outils divers

11.1 Dupliquer un segment

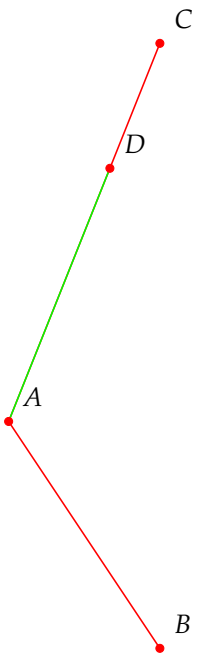
Il s'agit de construire un segment sur une demi-droite donnée de même longueur qu'un segment donné.

\tkzDuplicateSegment(⟨pt1,pt2⟩)(⟨pt3,pt4⟩){⟨pt5⟩}

Il s'agit de créer un segment sur une demi-droite donnée de même longueur qu'un segment donné . Il s'agit en fait de la définition d'un point.

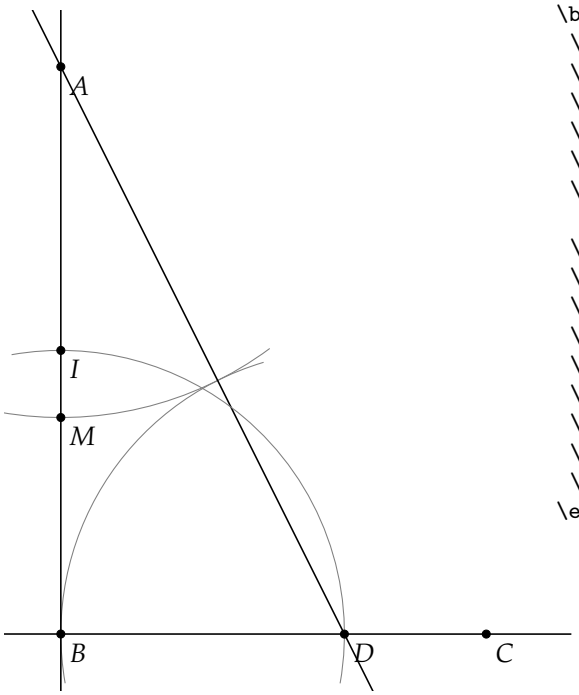
arguments	exemple	explication
(pt1,pt2)(pt3,pt4){pt5}	**\tkzDuplicateLen**(A,B)(E,F){C}	AC=EF et $C \in [AB)$

La macro **\tkzDuplicateLength** est identique à celle-ci.




```
\begin{tikzpicture}
\tkzDefPoint(0,0){A}
\tkzDefPoint(2,-3){B}
\tkzDefPoint(2,5){C}
\tkzDrawSegments[red](A,B A,C)
\tkzDuplicateSegment(A,B)(A,C)  \tkzGetPoint{D}
\tkzDrawSegment[green](A,D)
\tkzDrawPoints[color=red](A,B,C,D)
\tkzLabelPoints[above right=3pt](A,B,C,D)
\end{tikzpicture}
```

11.1.1 Proportion d'or avec \tkzDuplicateSegment




```
\begin{tikzpicture}[rotate=-90,scale=.75]
 \tkzInit[xmax=10,ymax=10]
 \tkzClip[space=1]
 \tkzDefPoint(0,0){A}
 \tkzDefPoint(10,0){B}
 \tkzDefMidPoint(A,B)    \tkzGetPoint{I}
 \tkzDefPointWith[orthogonal,K=-.75](B,A)
    \tkzGetPoint{C}
 \tkzInterLC(B,C)(B,I)   \tkzGetSecondPoint{D}
 \tkzDuplicateSegment(B,D)(D,A) \tkzGetPoint{E}
 \tkzInterLC(A,B)(A,E)    \tkzGetPoints{N}{M}
 \tkzDrawArc[delta=10](D,E)(B)
 \tkzDrawArc[delta=10](A,M)(E)
 \tkzDrawLines(A,B B,C A,D)
 \tkzDrawArc[delta=10](B,D)(I)
 \tkzDrawPoints(A,B,D,C,M,I,N)
 \tkzLabelPoints(A,B,D,C,M,I,N)
\end{tikzpicture}
```

11.2 Déterminer une pente

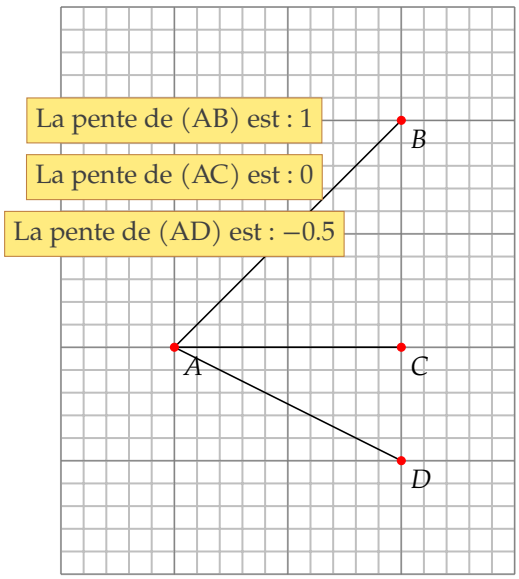
Il s'agit de déterminer si elle existe, la pente d'une droite définie par deux points. Aucune vérification de l'existence n'est faite.

\tkzFindSlope(⟨pt1,pt2⟩){⟨name of macro⟩}

Le résultat est stocké dans une macro.

arguments	exemple	explication
(pt1,pt2)pt3	**\tkzFindSlope**(A,B){slope}	**\slope** donnera le résultat de $\frac{y_B-y_A}{x_B-x_A}$

Attention à ne pas avoir $x_B = x_A$




```
\begin{tikzpicture}[scale=1.5]
  \tkzInit[xmax=4,ymax=5]\tkzGrid[sub]
  \tkzDefPoint(1,2){A}    \tkzDefPoint(3,4){B}
  \tkzDefPoint(3,2){C}    \tkzDefPoint(3,1){D}
  \tkzDrawSegments(A,B A,C A,D)
  \tkzDrawPoints[color=red](A,B,C,D)
  \tkzLabelPoints(A,B,C,D)
  \tkzFindSlope(A,B){SAB} \tkzFindSlope(A,C){SAC}
  \tkzFindSlope(A,D){SAD}
  \pgfkeys{/pgf/number format/.cd,fixed,precision=2}
  \tkzText[fill=Gold!50,draw=brown](1,4)%
  {La pente de (AB) est : $\pgfmathprintnumber{\SAB}$}
  \tkzText[fill=Gold!50,draw=brown](1,3.5)%
  {La pente de (AC) est : $\pgfmathprintnumber{\SAC}$}
  \tkzText[fill=Gold!50,draw=brown](1,3)%
  {La pente de (AD) est : $\pgfmathprintnumber{\SAD}$}
\end{tikzpicture}
```

11.3 Angle formé par une droite avec l'axe horizontal

Beaucoup plus intéressante que la précédente. Le résultat est compris entre -180 degrés et +180 degrés.

\tkzFindSlopeAngle(⟨pt1,pt2⟩)

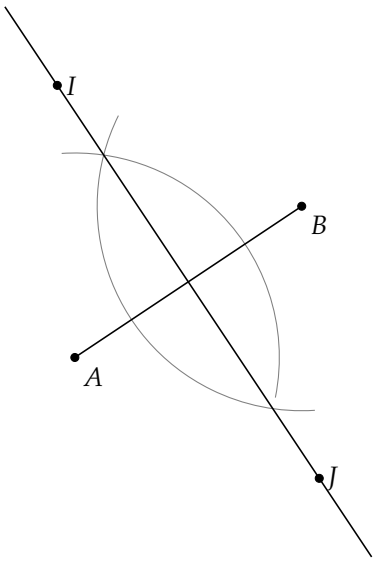
Le résultat est stocké dans une macro **\tkzAngleResult**.

arguments	exemple	explication
(pt1,pt2)	**\tkzFindSlopeAngle**(A,B)	**\tkzGetAngle** peut récupèrer le résultat

Si la récupération n'est pas nécessaire, il est possible d'utiliser **\tkzAngleResult**

11.3.1 Exemple d'utilisation de \tkzFindSlopeAngle

Voici une autre version de la construction d'une médiatrice




```
\begin{tikzpicture}
 \tkzInit
 \tkzDefPoint(0,0){A}          \tkzDefPoint(3,2){B}
 \tkzDefLine[mediator](A,B)    \tkzGetPoints{I}{J}
 \tkzCalcLength[cm](A,B)       \tkzGetLength{dAB}
 \tkzFindSlopeAngle(A,B)       \tkzGetAngle{tkzangle}
 \begin{scope}[rotate=\tkzangle]
   \tikzset{arc/.style={color=gray,delta=10}}
   \tkzDrawArc[R,arc](B,3/4*\dAB)(120,240)
   \tkzDrawArc[R,arc](A,3/4*\dAB)(-45,60)
   \tkzDrawLine(I,J)           \tkzDrawSegment(A,B)
  \end{scope}
  \tkzDrawPoints(A,B,I,J)      \tkzLabelPoints(A,B)
   \tkzLabelPoints[right](I,J)
\end{tikzpicture}
```

11.4 Récupérer un angle

Dans l'exemple précédent, j'ai utilisé la macro **\tkzGetAngle** qui permet de récupérer un angle.

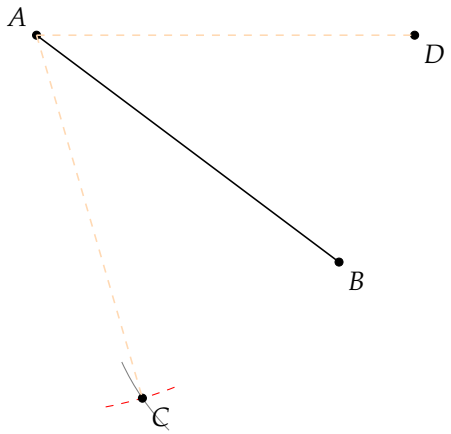
\tkzGetAngle{⟨name of macro⟩}

Cette macro récupère **\tkzAngleResult** et stocke le résultat dans une nouvelle macro.

arguments	exemple	explication
name of macro	**\tkzGetAngle**{ang}	**\ang** contient la valeur de l'angle.

11.5 Exemple d'utilisation de \tkzGetAngle

Il s'agit ici que (AB) soit la bissectrice de $\widehat{CAD}$, tel que la pente AD soit nulle. On récupère la pente de (AB) puis on effectue deux rotations.



```
\begin{tikzpicture}
  \tkzInit
  \tkzDefPoint(1,5){A} \tkzDefPoint(5,2){B}
  \tkzDrawSegment(A,B)
  \tkzFindSlopeAngle(A,B)\tkzGetAngle{tkzang}
  \tkzDefPointBy[rotation= center A angle \tkzang ](B)
   \tkzGetPoint{C}
  \tkzDefPointBy[rotation= center A angle -\tkzang ](B)
  \tkzGetPoint{D}
  \tkzCompass[length=1,dashed,color=red](A,C)
  \tkzCompass[delta=10,brown](B,C)
   \tkzDrawPoints(A,B,C,D)
  \tkzLabelPoints(B,C,D)
  \tkzLabelPoints[above left](A)
  \tkzDrawSegments[style=dashed,color=orange!30](A,C A,D)
\end{tikzpicture}
```

11.6 Angle formé par trois points

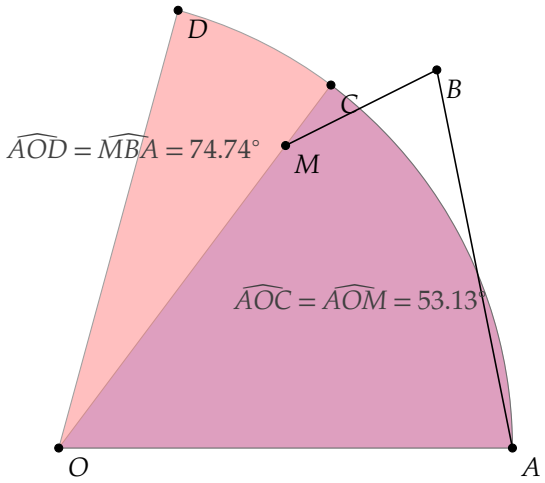
\tkzFindAngle(⟨pt1,pt2,pt3⟩)

Le résultat est stocké dans une macro **\tkzAngleResult**.

arguments	exemple	explication
(pt1,pt2,pt3)	**\tkzFindAngle**(A,B,C)	**\tkzAngleResult** donne l'angle $(\overrightarrow{BA},\overrightarrow{BC})$

Le résultat est compris entre -180 degrés et +180 degrés. pt2 est le sommet et **\tkzGetAngle** peut récupérer l'angle.

11.7 Exemple d'utilisation de \tkzFindAngle



```
\begin{tikzpicture}
   \tkzInit[xmin=-1,ymin=-1,xmax=7,ymax=7]
   \tkzClip
   \tkzDefPoint (0,0){O}  \tkzDefPoint (6,0){A}
   \tkzDefPoint (5,5){B}  \tkzDefPoint (3,4){M}
   \tkzFindAngle (A,O,M)  \tkzGetAngle{an}
   \tkzDefPointBy[rotation=center O angle \an](A)
   \tkzGetPoint{C}
   \tkzDrawSector[fill = blue!50,opacity=.5](O,A)(C)
   \tkzFindAngle(M,B,A)   \tkzGetAngle{am}
   \tkzDefPointBy[rotation = center O angle \am](A)
   \tkzGetPoint{D}
   \tkzDrawSector[fill = red!50,opacity = .5](O,A)(D)
   \tkzDrawPoints(O,A,B,M,C,D)
   \tkzLabelPoints(O,A,B,M,C,D)
\edef\an{\fpeval{round(\an,2)}}\edef\am{\fpeval{round(\am,2)}}
   \tkzDrawSegments(M,B B,A)
   \tkzText(4,2){$\widehat{AOC}=\widehat{AOM}=\an^{\circ}$}
   \tkzText(1,4){$\widehat{AOD}=\widehat{MBA}=\am^{\circ}$}
\end{tikzpicture}
```

11.8 Longueur d'un segment \tkzCalcLength

Il existe dans TikZ une option **veclen**. Cette option permet de calculer AB si A et B sont deux points.

Le seul problème pour moi est que la version de TikZ n'est pas assez précise dans certains cas particuliers. Ma version utilise le package **xfp** et est plus lente, mais plus précise

\tkzCalcLength[⟨local options⟩](⟨pt1,pt2⟩){⟨name of macro⟩}

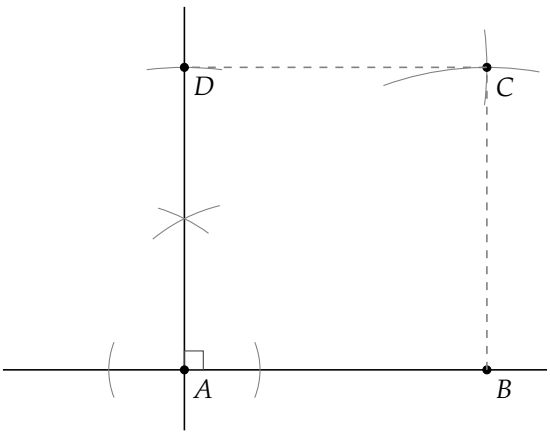
Le résultat est stocké dans une macro.

arguments	exemple	explication
(pt1,pt2){name of macro}	\tkzCalcLength(A,B){dAB}	\dAB donne AB en pt

Une seule option

options	défaut	exemple
cm	false	\tkzCalcLength[cm](A,B){dAB} \dAB donne AB en cm

11.8.1 Construction d'un carré au compas




```
\begin{tikzpicture}[scale=1]
  \tkzDefPoint(0,0){A} \tkzDefPoint(4,0){B}
  \tkzDrawLine[add= .6 and .2](A,B)
  \tkzCalcLength[cm](A,B)\tkzGetLength{dAB}
  \tkzDefLine[perpendicular=through A](A,B)
  \tkzDrawLine(A,tkzPointResult) \tkzGetPoint{D}
  \tkzShowLine[orthogonal=through A,gap=2](A,B)
  \tkzMarkRightAngle(B,A,D)
  \tkzVecKOrth[-1](B,A)\tkzGetPoint{C}
  \tkzCompasss(A,D D,C)
  \tkzDrawArc[R](B,\dAB)(80,110)
  \tkzDrawPoints(A,B,C,D)
  \tkzDrawSegments[color=gray,style=dashed](B,C C,D)
  \tkzLabelPoints(A,B,C,D)
\end{tikzpicture}
```

11.9 Transformation de pt en cm ou de cm en pt

Pas sûr que cela soit nécessaire et il ne s'agit que d'une division par 28,45274 et d'un multiplication par ce même nombre. Les macros sont :

\tkzpttocm(⟨nombre⟩){⟨name of macro⟩}

Le résultat est stocké dans une macro.

arguments	exemple	explication
(nombre)name of macro	**\tkzpttocm**(120){len}	**\len** donne un nombre de tkznamecm

Il faudra utiliser **\len** accompagné de **cm**

11.10 changement d'unité

\tkzcmtopt(⟨nombre⟩){⟨name of macro⟩}

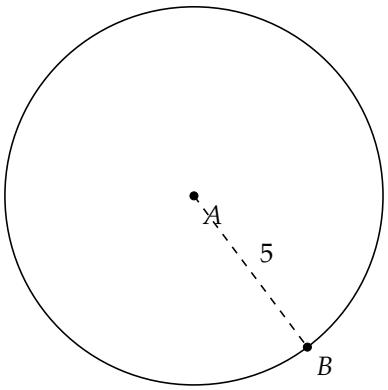
Le résultat est stocké dans une macro.

arguments	exemple	explication
(nombre){name of macro}	**\tkzcmtopt**(5){len}	**\len** longueur en **pts**

Le résultat s'utilise avec **\lenpt**

11.10.1 Exemple

La macro **\tkzDefCircle[radius](A,B)** définit le rayon que l'on récupère avec **\tkzGetLength**, mais ce résultat est en **pt**.




```
\begin{tikzpicture}[scale=.5]
   \tkzDefPoint(0,0){A}
   \tkzDefPoint(3,-4){B}
   \tkzDefCircle[through](A,B)
   \tkzGetLength{rABpt}
   \tkzpttocm(\rABpt){rABcm}
   \tkzDrawCircle(A,B)
   \tkzDrawPoints(A,B)
   \tkzLabelPoints(A,B)
   \tkzDrawSegment[dashed](A,B)
   \tkzLabelSegment(A,B){%
       $\pgfmathprintnumber{\rABcm}$}
\end{tikzpicture}
```

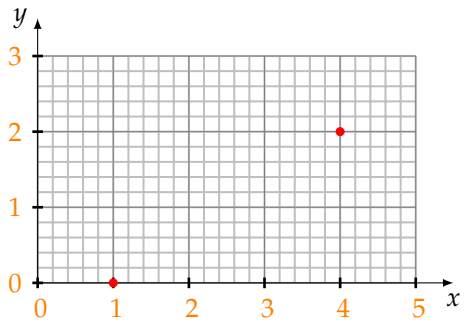
\tkzGetPointCoord(⟨A⟩){⟨name of macro⟩}

Stocke dans deux macros les coordonnées d'un point

arguments	exemple	explication
(point){name of macro}	\tkzGetPointCoord(A){A}	\Ax et \Ay donnent les coordonnées de A

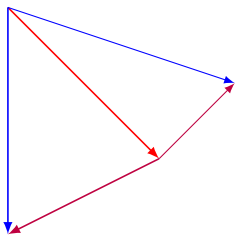
Si le nom de la macro est p, alors \px et \py donnent les coordonnées du point choisi avec le cm comme.

11.10.2 Transfert de coordonnées avec \tkzGetPointCoord




```
\begin{tikzpicture}
 \tkzInit[xmax=5,ymax=3]
 \tkzGrid[sub,orange]
 \tkzAxeXY
 \tkzDefPoint(1,0){A}
 \tkzDefPoint(4,2){B}
 \tkzGetPointCoord(A){a}
 \tkzGetPointCoord(B){b}
 \tkzDefPoint(\ax,\ay){C}
 \tkzDefPoint(\bx,\by){D}
 \tkzDrawPoints[color=red](C,D)
\end{tikzpicture}
```

11.10.3 Somme de vecteurs avec \tkzGetPointCoord



```
\begin{tikzpicture}[>=latex]
   \tkzDefPoint(1,4){a}
   \tkzDefPoint(3,2){b}
   \tkzDefPoint(1,1){c}
   \tkzDrawSegment[->,red](a,b)
   \tkzGetPointCoord(c){c}
   \draw[color=blue,->](a) -- ([shift=(b)]\cx,\cy) ;
   \draw[color=purple,->](b) -- ([shift=(b)]\cx,\cy) ;
   \tkzDrawSegment[->,blue](a,c)
   \tkzDrawSegment[->,purple](b,c)
\end{tikzpicture}
```

12 Utilisation des objets complémentaires ou des outils

12.1 Objets complémentaires

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

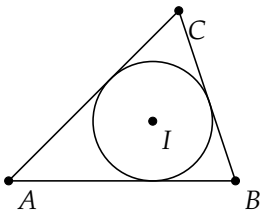
Il est possible d'utiliser certains de ces objets, sans charger complètement **tkz-euclide**, mais en utilisant la macro **\usetkzobj**.

tkz-base charge les objets les plus utilisés, marqués « présent » dans la liste ci-dessous. Cette liste a évolué et le peut encore..

\usetkzobj{⟨liste d'objets⟩}

options		définition
axes	présent	tracer des axes
circles	présent	tracer, nommer des circles,
grids	présent	tracer des grilles
lines	présent	tracer, nommer des droites
marks	présent	tracer des marques
points	présent	définir, nommer, tracer des points
rep	présent	définir, tracer un repère
segments	présent	étiqueter, tracer des segments
defcircles	absent	définir des cercles
deflines	absent	définir des droites
defpointsby	absent	définir des points obtenus par une transformation
defpointsrnd	absent	définir des points aléatoires
defpointswith	absent	définir des points obtenus vectoriellement
angles	absent	définir, nommer, tracer des angles
arcs	absent	définir, tracer des arcs
compass	absent	afficher des traces de compas
polygons	absent	définir, nommer, tracer des polygones
protractor	absent	tracer un rapporteur
sectors	absent	définir, nommer, tracer des secteurs
show	absent	afficher les traces de compas d'une construction
triangles	absent	définir, nommer, tracer des triangles

12.2 \usetkzobj{defcircles}




```
\begin{tikzpicture}[scale=0.75]
  \tkzDefPoint(0,0){A}
  \tkzDefPoint(4,0){B}
  \tkzDefPoint(3,3){C}
  \tkzDefCircle[in](A,B,C)
  \tkzGetPoint{I}\tkzGetLength{rI}
  \tkzDrawCircle[R](I,\rI)
  \tkzDrawPolygon(A,B,C)
  \tkzDrawPoints(A,B,C,I)
  \tkzLabelPoints(A,B,C,I)
\end{tikzpicture}
```

12.3 Outils complémentaires

Attention, il faut utiliser **tkz-euclide** pour avoir la possibilité d'utiliser des outils comme les transformations ou encore les intersections.

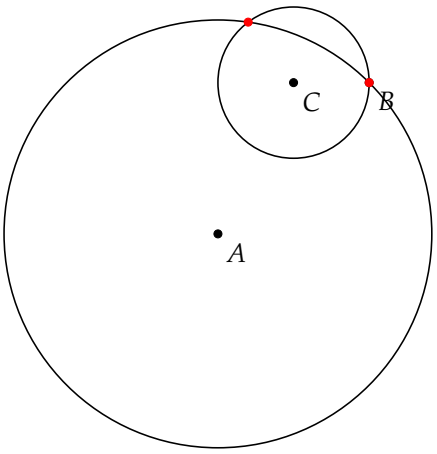
\usetkztool{⟨liste d'objets⟩}

options		définition
BB	présent	outils permettant de trvailler sur la Bounding Box (BB)
arith	présent	outils sur les nombres entiers "macros dans TikZ >=3"
base	présent	macro **\tkzInit** base essentielle
math	présent	outils mathématiques de base
print	présent	définir, nommer, tracer des points particuliers
text	présent	outils permettant de travailler sur des textes
utilities	présent	utilitaires
intersections	absent	intersection de droites, de cercles, de droite et de cercle

12.4 Exemple

```
\documentclass[border=.25cm]{standalone}
\usepackage{tkz-base}
\usetkztool{intersections}

\begin{document}
\begin{tikzpicture}
 \tkzDefPoints{0/0/A,2/2/B,1/2/C}
 \tkzDrawCircles(A,B C,B)
 \tkzDrawPoints(A,B,C)
 \tkzLabelPoints(A,B,C)
 \tkzInterCC(A,B)(C,B)
 \tkzGetPoints{D}{E}
 \tkzDrawPoints[red](D,E)
\end{tikzpicture}
\end{document}
```



13 Utilisation d'un repère

13.1 Repère avec \tkzRep

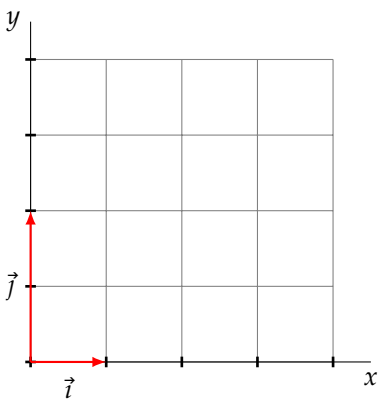
\tkzRep[⟨local options⟩]

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

13.1.1 Quelques styles modifiables

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

13.1.2 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}
```

sPour ceux qui utilisent **frenchb** avec **babel**, en cas de problème avec la version 3 de pgf, il suffit de charger la librairie **babel**. TikZ a été en effet parfois allergique aux caractères actifs.

14 Droites parallèles aux axes

14.1 Tracer une ligne horizontale avec \tkzHLine

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

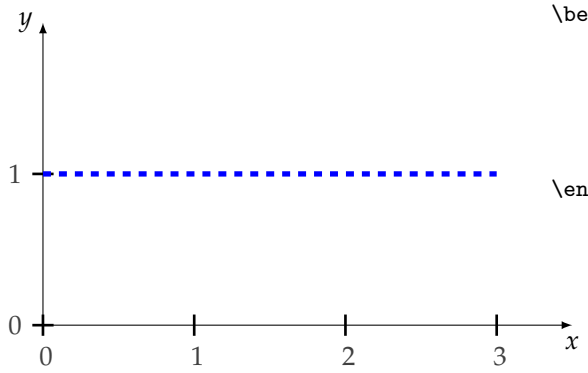
La syntaxe est celle de **xfp**!

arguments	exemple	définition
decimal number	\tkzHLine{1}	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

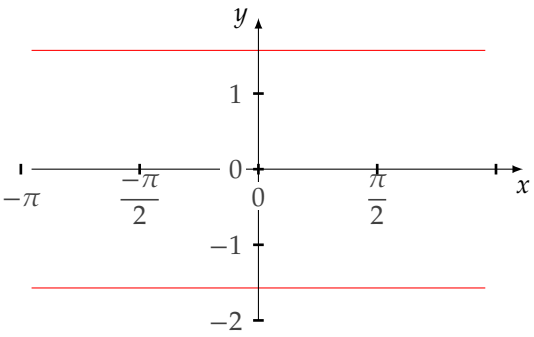
voir les options les lignes dans TikZ

14.1.1 Ligne horizontale




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

14.1.2 Ligne horizontale et valeur calculée par xfp




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

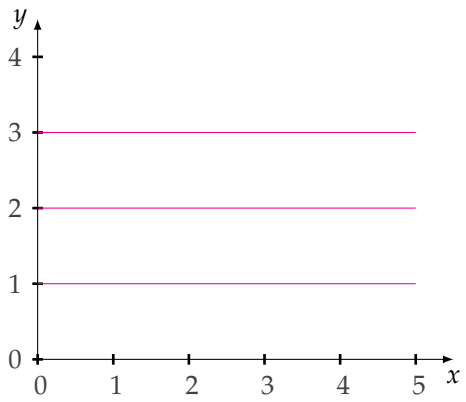
14.2 Lignes horizontales avec \tkzHLines

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

La syntaxe est celle de **xfp**!

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

14.2.1 Lignes horizontales




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

14.3 Tracer une ligne verticale avec \tkzVLine

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

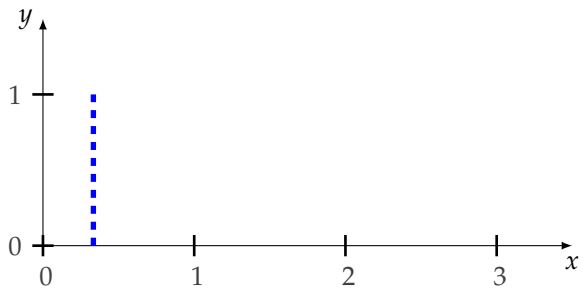
La syntaxe est celle de **xfp**!

arguments	exemple	définition
decimal number	`\tkzVLine{1}`	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

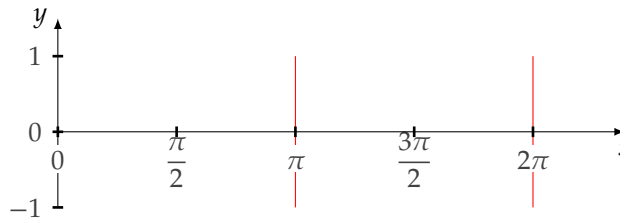
voir les options les lignes dans TikZ

14.3.1 Ligne verticale




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

14.3.2 Ligne verticale et valeur calculée par xfp




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

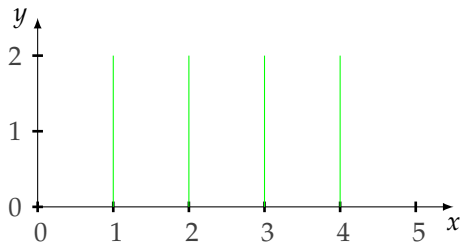
14.4 Lignes verticales avec \tkzVLines

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

La syntaxe est celle de **xfp**!

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

14.4.1 Lignes verticales




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

15 Ticks sur les axes

15.1 Tracer des ticks sur l'axe des abscisses \tkzHTick

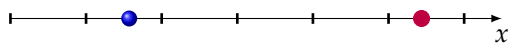
\tkzHTick[⟨local options⟩]{⟨decimal number⟩}

arguments	exemple	définition
decimal number	\tkzHTick{1}	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

15.1.1 exemple




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

15.2 Tracer des ticks sur l'axe des ordonnées \tkzHTicks

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

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

voir les options de TikZ.

15.3 Tracer des ticks sur l'axe des abscisses \tkzVTick

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

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

voir les options de TikZ.

15.4 Tracer des ticks sur l'axe des abscisses \tkzVTicks

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

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

voir les options de TikZ.

16 Marks, marques ou symboles

J'ai distingué les points utilisés en géométrie euclidienne et les « marks » ou symboles que l'on peut rencontrés 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 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. La macro **\tkzDefSetOfPoints** a été défini

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. Voir tkzDefSetOfPoints

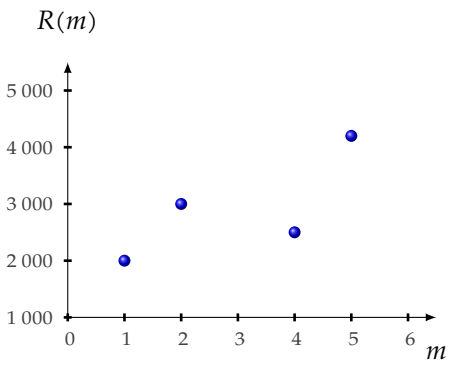
16.1 \tkzDrawSetOfPoints

\tkzDrawSetOfPoints[⟨local options⟩]

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

options	défaut	définition
prefix	tkzPt	préfixe des noms des points

16.1.1 Tracé d'un nuage avec \tkzDrawSetOfPoints




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

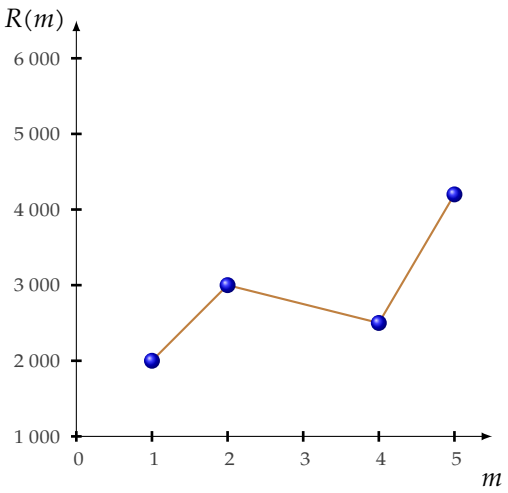
16.2 \tkzJoinSetOfPoints

\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.

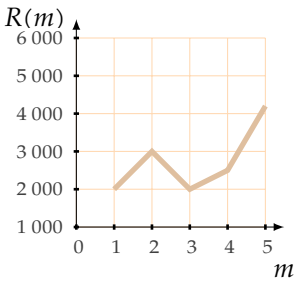
options	défaut	définition
prefix	tkzPt	préfixe des noms des points

16.2.1 Lier les points d'un nuage avec \tkzJoinSetOfPoints




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

16.2.2 Utilisation des points d'un nuage



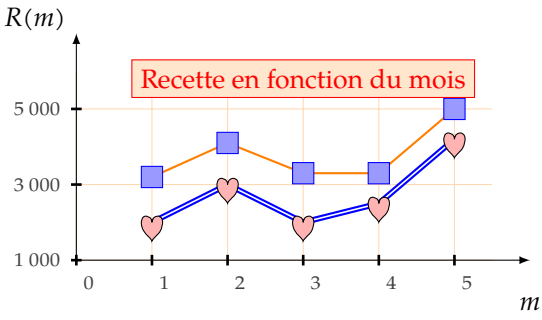

```
\begin{tikzpicture}[scale=.5]
\tkzInit[xmax=5,ymin=1000,
         ymax=6000,ystep=1000]
\tkzGrid[color=orange!30]
\tkzDrawX[label=$m$,below=13pt]
\tkzDrawY[label=$R(m)$]
\tkzLabelX[font=\scriptsize]
\tkzLabelY[font=\scriptsize]
\tkzDefSetOfPoints[prefix=P]{%
   1/2000,2/3000,3/2000,4/2500,5/4200}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

16.3 \tkzSetUpMark

\tkzSetUpMark[⟨local options⟩]

options	défaut	définition
liste	no default	exemple \tkzSetUpMark[mark=heart]

16.3.1 Deux nuages



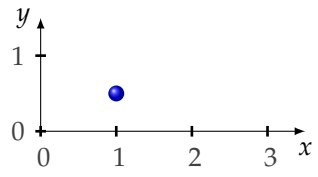

```
\begin{tikzpicture}
\tkzInit[xmax=5.5,ymin=1000,%
         ymax=6000,ystep=2000]
\tkzGrid[color=orange!30]
\tkzDrawX[label=$m$,below=13pt]
\tkzDrawY[above left,label=$R(m)$]
\tkzLabelX[below right,font=\scriptsize]
\tkzLabelY[font=\scriptsize]
\tkzDefSetOfPoints{1/2000,2/3000,3/2000,
    4/2500,5/4200}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

16.4 \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




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

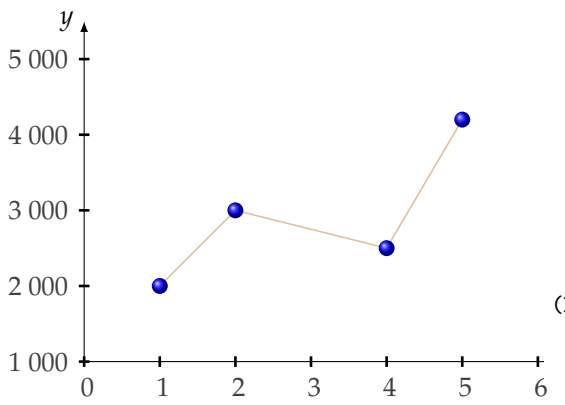
16.5 \tkzDrawMarks

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

Permet de placer une série de marques.

options	défaut	définition
prefix	tkzPt	préfixe des noms des points

16.5.1 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=brown!50]%
(P1,P2 P2,P3 P3,P4)
\tkzDrawMarks[mark=ball](P1,P2,P3,P4)
\end{tikzpicture}
```

17 Textes et Légendes

17.1 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.

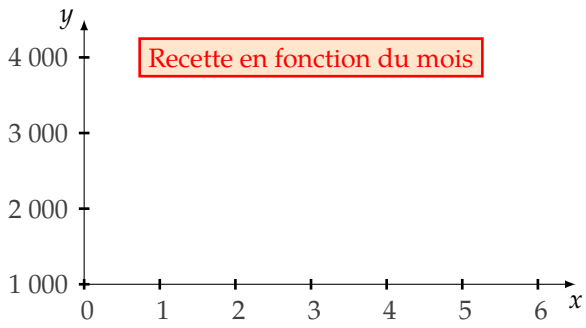
les options sont toujours celles de TikZ, en particulier les suivantes :

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

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

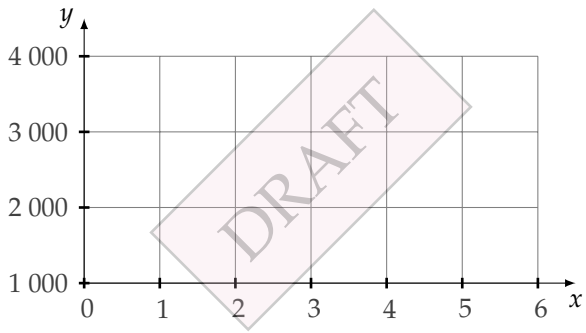
options	défaut	définition
color	black	couleur courante
text	black	couleur du texte
fill	white	couleur du fond
opacity	1	opacité

17.1.1 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}
```

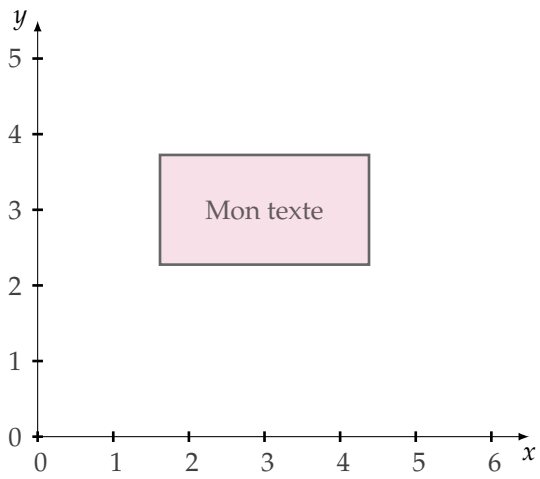
17.1.2 Draft




```
\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}
```

17.1.3 Texte avec un point

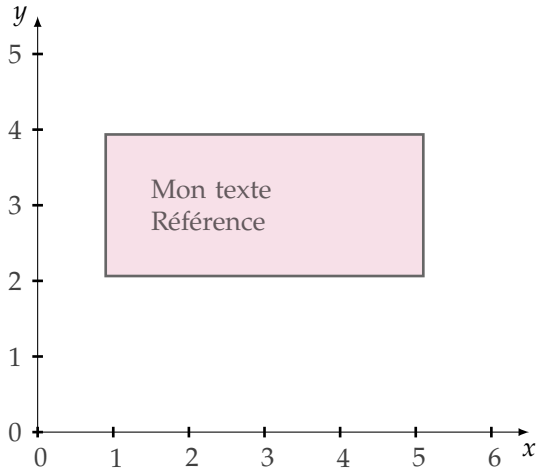
Il est possible de donner la référence d'un point à la place de ses coordonnées.




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

17.1.4 Format du texte

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




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

17.2 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 **line**.

options	défaut	définition
line	false	booléen :ligne ou symbole

Avec line=true

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

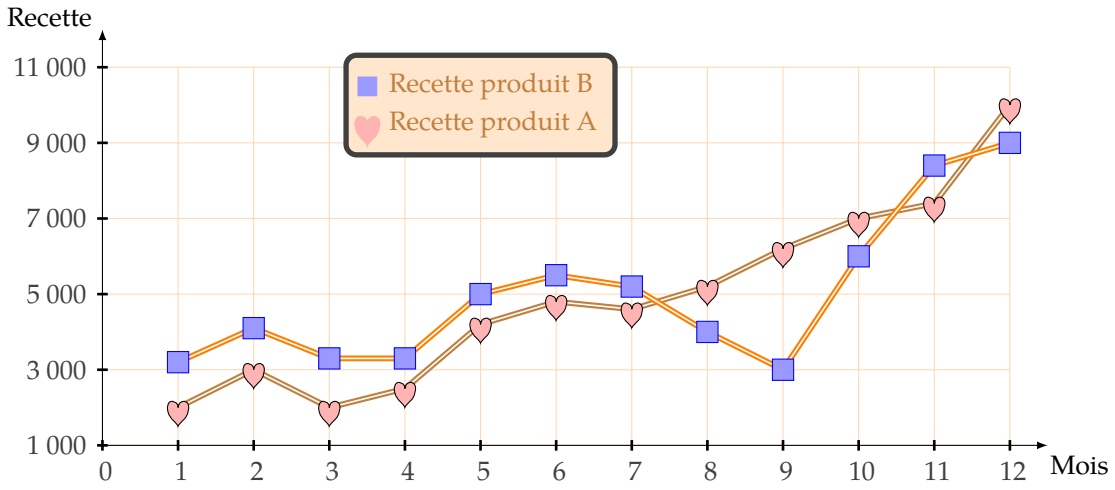
Avec line=false

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**

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

17.2.1 Légendes avec des symboles




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

18 FAQ

18.1 Questions générales

— **Pourquoi tkz-base ?** J'avais besoin en tant que professeur de Mathématiques d'outils permettant d'écrire mes cours et mes exercices rapidement. TikZ était parfait pour cela, mais je perdai trop de temps sur des détails. J'ai voulu créer une syntaxe à la fois proche de celle de LaTeX et des mathématiques afin de mieux mémoriser. J'ai créé ainsi un module pour chaque branche des mathématiques que j'enseignais. **tkz-base** est la partie commune de tous ces modules. **tkz-euclide** et **tkz-berge** sont ceux pour lesquels je me suis le plus investi.

— **Relation avec TikZ ?** TikZ est un superbe package permettant de décrire des dessins. Mes packages sont basés sur lui. Cela dit, cela n'est aucunement comparable. Mes packages ne sont utiles que pour des personnes voulant créer des figures de mathématiques.

18.2 Erreurs les plus fréquentes

— **Error unknown option : label options** Cette option n'existe plus . Vous pouvez maintenant directement utiliser les options de TikZ.

— **Error avec \tkzDrawPoint ou \tkzDefPoint \tkzDrawPoint(A,B)** alors qu'il faut **\tkzDrawPoints**. C'est vrai avec toutes les macros qui permettent de définir plusieurs objets. La forme singulière permmet d'utiliser des options personnalisées. En revanche il est possible d'utiliser la forme pluriel pour un unique objet.

— **propagation d'un style** Il est possible de restreindre la propagation d'un style en plaçant un morceau de code dans un groupe ou dans un environnement **scope** ou bien entre des parenthèses.

— **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{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 **xfp**.

Index