Exposés sur LaTeX

Cours 7

LaTeX et les graphiques

Thierry Masson

Centre de Physique Théorique
Campus de Luminy, Marseille

Cours 7 – LaTeX et les graphiques

- Généralités sur les graphiques
- Faire des dessins à la souris
- Illustrer les mathématiques
- Programmer des graphiques
- Insertion de graphiques externes
- Insertion d'une figure avec légende

« Ce dessin m'a pris cinq minutes,
mais j'ai mis soixante ans pour y arriver. »
Pierre-Auguste Renoir

Documentation à consulter

Documentation générale sur les graphiques et LaTeX :

- **The LaTeX Graphics Companion**, M. Goossens, F. Mittelbach, S. Rahtz, D. Roegel, H. Voss, Addison-Wesley 2007.
- **Graphics in LaTeX**, un article qui recense les *packages* pour produire des graphiques au sein de LaTeX.

Documentation pour le logiciel Asymptote :

- **Le site officiel d'Asymptote** donne de nombreux exemples.
- **Asymptote: the Vector Graphics Language**, la documentation officielle.
- **Asymptote - Galeries d'exemples**, un site qui fournit des exemples simplifiés illustrant de nombreux points du langage.

Documentation pour le *package* TikZ :

- **Le site officiel des *packages* pgf et `tikz`**.
- **TikZ & PGF**, le manuel officiel.
- **TikZ and PGF examples**, un site qui collecte de nombreux exemples, une bonne façon de démarrer en faisant du copier-coller…
- **Graphics with TikZ**, un article court qui explique l'essentiel de TikZ.
- **TikZ pour l'impatient**, un excellent livre en PDF très simple et très pédagogique.

Les autres logiciels qui seront évoqués ont leur propre documentation, qui sort du stricte cadre d'une installation LaTeX.

Généralités sur les graphiques

Où l'on s'initie à regarder la nature intime d'une image et la structure informatique d'un graphique…

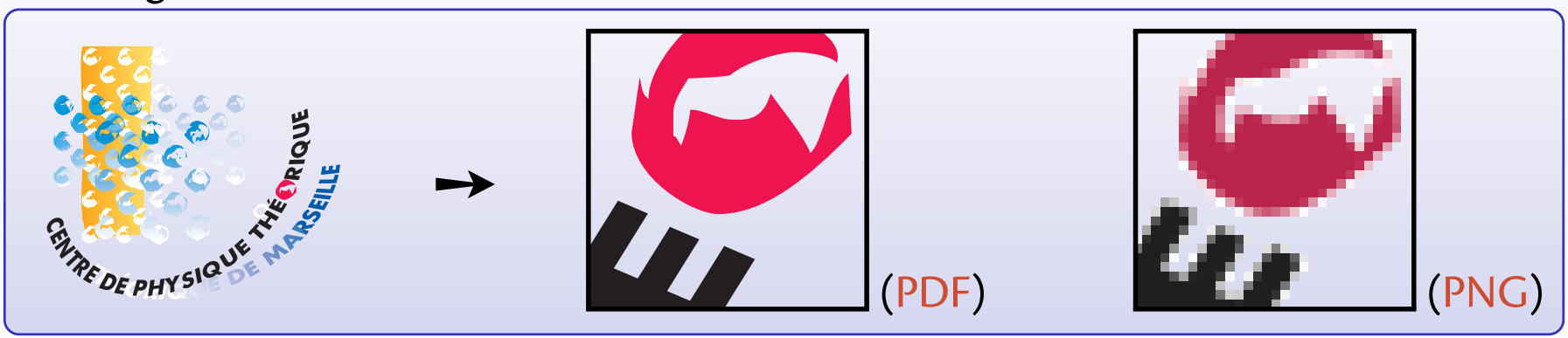
Différents formats d'images

Le monde des images se divise en deux catégories.

Les images matricielles (Bitmap) Une image matricielle est une image numérique composée d'un tableau de pixels (points de couleurs) et de données globales (taille, résolution…). Exemples de formats : JPG, PNG, TIFF, BMP…

Les images vectorielles Une image vectorielle est une image numérique composée d'objets géométriques individuels (segments de droite, polygones, arcs de cercle, etc.) définis chacun par divers attributs de forme, de position, de couleur, etc.
Exemples de formats : EPS, PDF, SVG…

L'agrandissement des images vectorielles est possible à n'importe quelle échelle, pas celui des images matricielles :

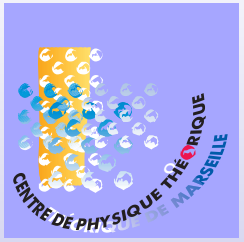


Produire des images Bitmap est relativement facile : photographie numérique, nombreux logiciels de dessin, *scan* d'une figure…
Produire des images vectorielles requiert des logiciels dédiés.
Les logiciels de création de dessin mentionnés dans ce qui suit seront de ce type.

Particularités de certains formats d'images

Les formats d'images PNG, TIFF, PDF et SVG gèrent des "couleurs transparentes". Lorsqu'on superpose l'image sur un fond coloré (ou une autre image), celle du dessous peut se voir à travers celle du dessus.

```
{\color{blue!35}\rule{1.5cm}{1.5cm}}%
\hspace{-1.5cm}%
\includegraphics[width=1.5cm]{CPT.pdf}
```



La transparence peut être totale ou partielle (on parle d'opacité).
JPG et EPS ne gèrent pas la transparence.

Tous les formats d'images que LaTeX utilise doivent impérativement comporter des informations sur la dimension de l'image : LaTeX réserve un espace de la taille de l'image (après d'éventuelles transformations). On parle de "BoundingBox".
L'utilitaire ebb permet d'afficher la "BoundingBox" des fichiers .pdf, .png et .jpg :

```
ebb -O image.png
```

L'utilitaire pdfcrop découpe un fichier .pdf pour qu'il ait la taille minimale du dessin :

```
pdfcrop image.pdf image-reduite.pdf
```

```
pdfcrop --margins 10 image.pdf image-reduite.pdf
```

➞ permet d'ajouter des marges supplémentaires.

Faire des dessins à la souris

Où l'on choisit d'abord de voir s'il nous est possible de dompter la souris pour produire de beaux dessins à dessein…

Principes généraux des logiciels de dessins vectoriels

Les logiciels de dessins vectoriels se ressemblent tous dans leurs principes de fonctionnement.

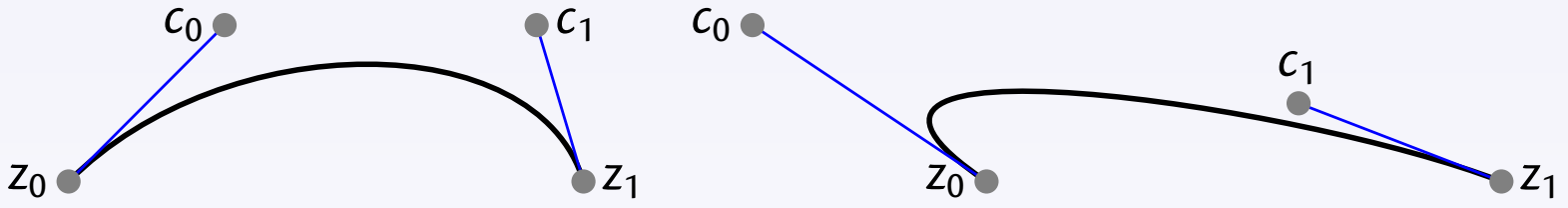
- Une zone de travail aux dimensions personnalisables.
- Possibilité d'agrandir la vue pour travailler plus finement.
- Un ensemble d'objets graphiques pré-définis : cercles, rectangles, arcs de cercle, ovales, étoiles, croix, spirales…
- Dessins de segments et de courbes passant par des points libres.
  ➟ lignes brisées, régulières, ouvertes, fermées…
  Souvent construites à partir de COURBES DE BÉZIER.
- Possibilité de remplir des zones fermées par des couleurs, des dégradés de couleurs…
- Personnalisation des lignes : couleur, épaisseur, pointillés, tiretés, flèches…
- Positionnement d'objets sur la zone de travail : positionnement horizontal, rotation, échelle, positionnement en profondeur (devant ou derrière d'autres objets)…
- Couches de dessins indépendantes qui peuvent se superposer.
- Groupement de plusieurs objets pour les manipuler ensemble.
- Outils de texte plus ou moins riches.
- Copier-coller des différents objets créés.
- Import de fichiers d'images et de dessins, export dans différents formats.

Intermède : les courbes de Bézier

Les courbes de Bézier sont des courbes polynomiales paramétriques qui permettent de dessiner une courbe lisse à partir de la donnée de certains points du plan.

La courbe de Bézier la plus utilisée est une courbe du plan complexe paramétrée par un polynôme de degré 3 :

$$t \mapsto (1-t)^3 z_0 + 3t(1-t)^2 c_0 + 3t^2(1-t)c_1 + t^3 z_1, \qquad t \in [0,1]$$



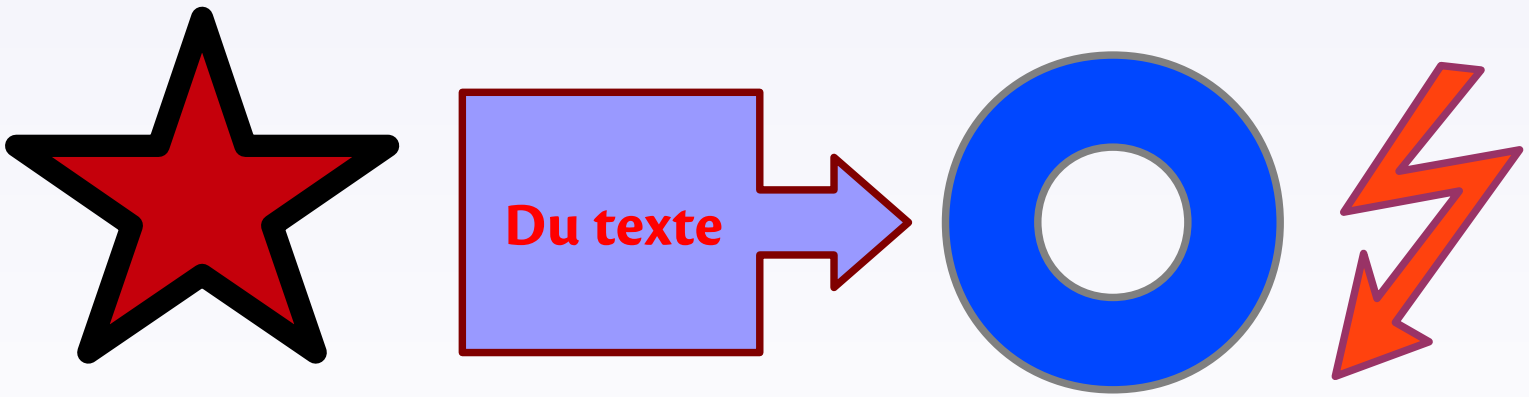
Les points z_0 et z_1 sont appelés les points initiaux et finaux (ou points extrêmes, nœuds), les points c_0 et c_1 sont appelés les points de contrôle (ou poignées).
De nombreuses variantes de ce paramétrage existent : certains donnent la "tension" entre les points initiaux et finaux, d'autres les directions tangentes en les points extrêmes, d'autres la "courbure de la courbe" aux points extrêmes.

Il existe aussi des courbes de Bézier quadratique de la forme $(1-t)^2 z_0 + 2t(1-t)c + t^2 z_1$, ainsi que des courbes de Bézier linéaires (segment de droite).

Le module de dessin de OpenOffice

OpenOffice est un logiciel libre qui se présente comme une suite bureautique, avec de nombreux modules : traitement de texte, tableur, présentation et surtout dessin.
Les principaux avantages du module Draw de OpenOffice :

- Nombreux objets géométriques prédéfinis.
- Grande simplicité de manipulation des objets graphiques, grande souplesse.
- Format de fichier de sauvegarde ouvert, assuré d'une certaine pérennité, et format d'exportation en PDF pour l'insertion dans LaTeX.
- Fonctionne sur tous les systèmes d'exploitations courants.

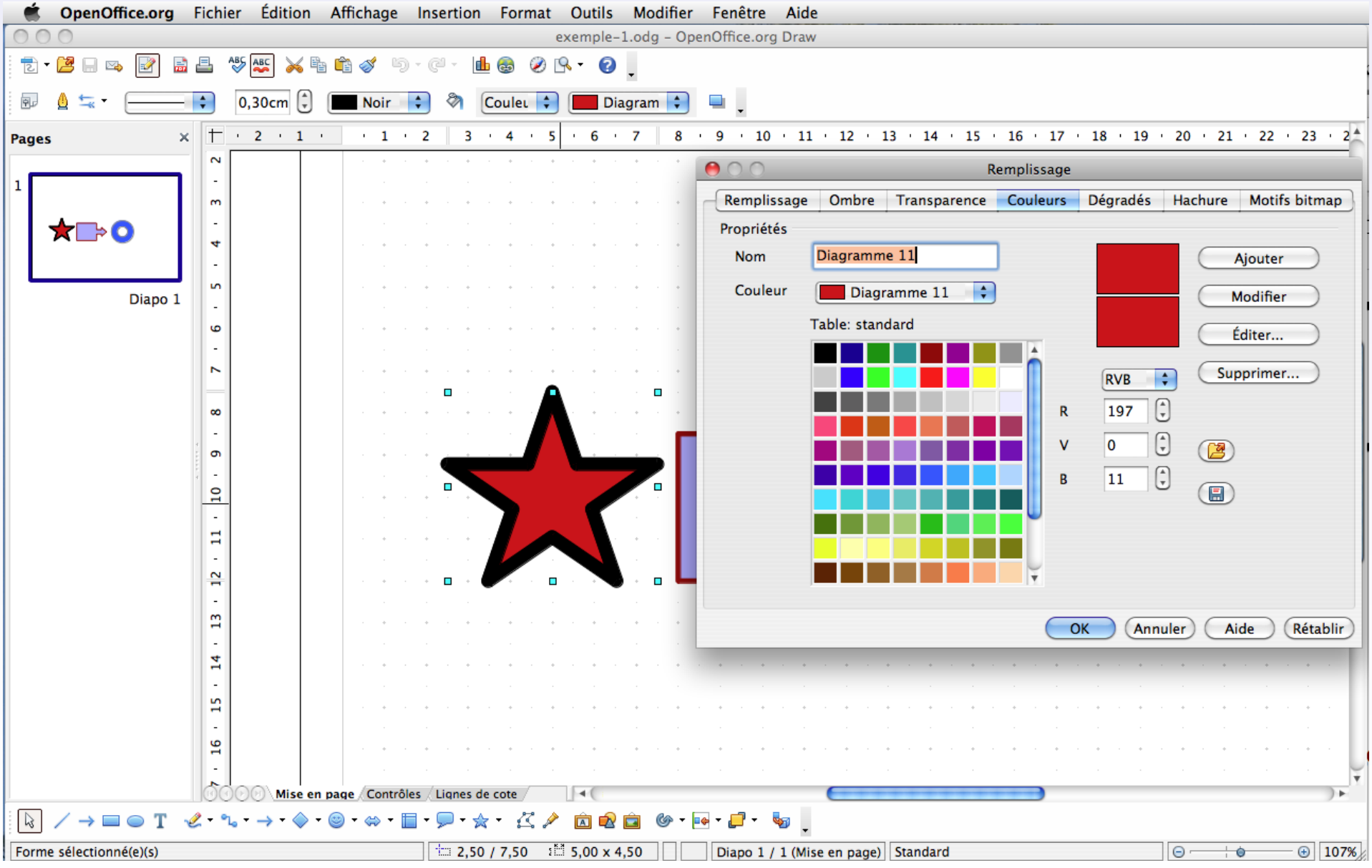


⚠ Suite à un bug qui résiste aux protestations, lorsqu'on exporte un dessin en PDF, il n'a pas la bonne taille (il conserve la taille de la page entière).
➟ on peut lui appliquer le script `pdfcrop` :

```
pdfcrop image.pdf image-reduite.pdf
```

Faire un dessin avec OpenOffice

L'environnement de travail de OpenOffice :



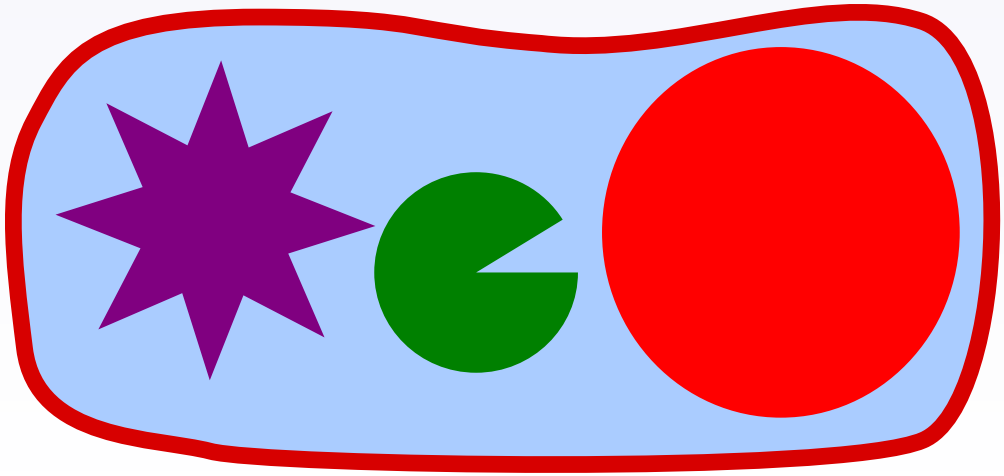
Le logiciel Inkscape

Inkscape est un logiciel libre d'édition de graphismes vectoriels, doté de capacités similaires à des logiciels professionnels du type Illustrator.

Les principaux avantages de Inkscape :

- Son format de fichier natif est le SVG (*Scalable Vector Graphics*) qui est un des standards du Web → assuré d'une pérennité.
- Nombreux objets géométriques prédéfinis.
- Outil très professionnel, grande puissance.
- Format d'exportation en PDF pour l'insertion dans LaTeX.
- Formats d'importation très divers : PostScript, EPS, PDF, JPG, PNG, TIFF…
- Fonctionne sur tous les systèmes d'exploitations courants.

→ on peut importer un fichier PDF pour le retravailler.



Inkscape et LaTeX

Inkscape est intéressant car il est conçu pour exporter en LaTeX.
La procédure à suivre est la suivante :

- Menu “Fichier” > “Enregistrer sous...”
- Choisir “Portable Document Format (PDF)” et cliquer sur “Sauver”
- Cocher les cases “PDF+LaTeX...” et “Exporter le dessin”.

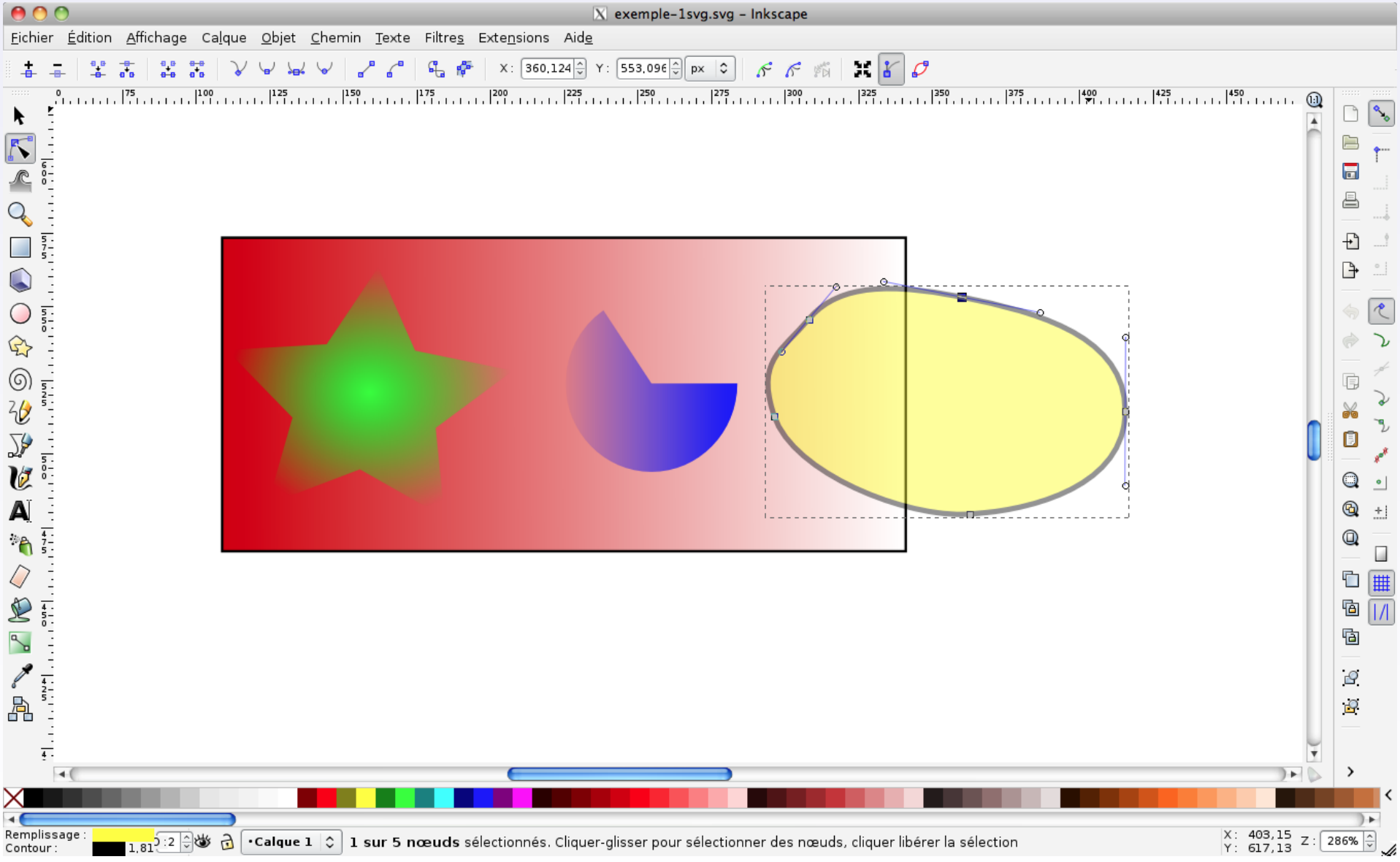
Le résultat produit deux fichiers :
un fichier image image.pdf et un fichier texte image.pdf_tex.
Dans un document LaTeX, on insère alors l’image par le code suivant :

```
\usepackage{graphicx}
\usepackage{xcolor}
...
\def\svgwidth{5cm}
\input{image.pdf_tex}
```

Le fichier image.pdf_tex s’occupe d’insérer l’image, et en plus il lui superpose les libellés de texte qu’on a placé sur le dessin dans Inkscape.
Ces libellés de texte peuvent être du code LaTeX quelconque, qui sera compilé dans le document LaTeX où l’image est insérée par cette procédure.
On peut éditer le fichier image.pdf_tex pour modifier des libellés de texte.
\svgwidth sert à donner la taille finale de l’image en largeur.

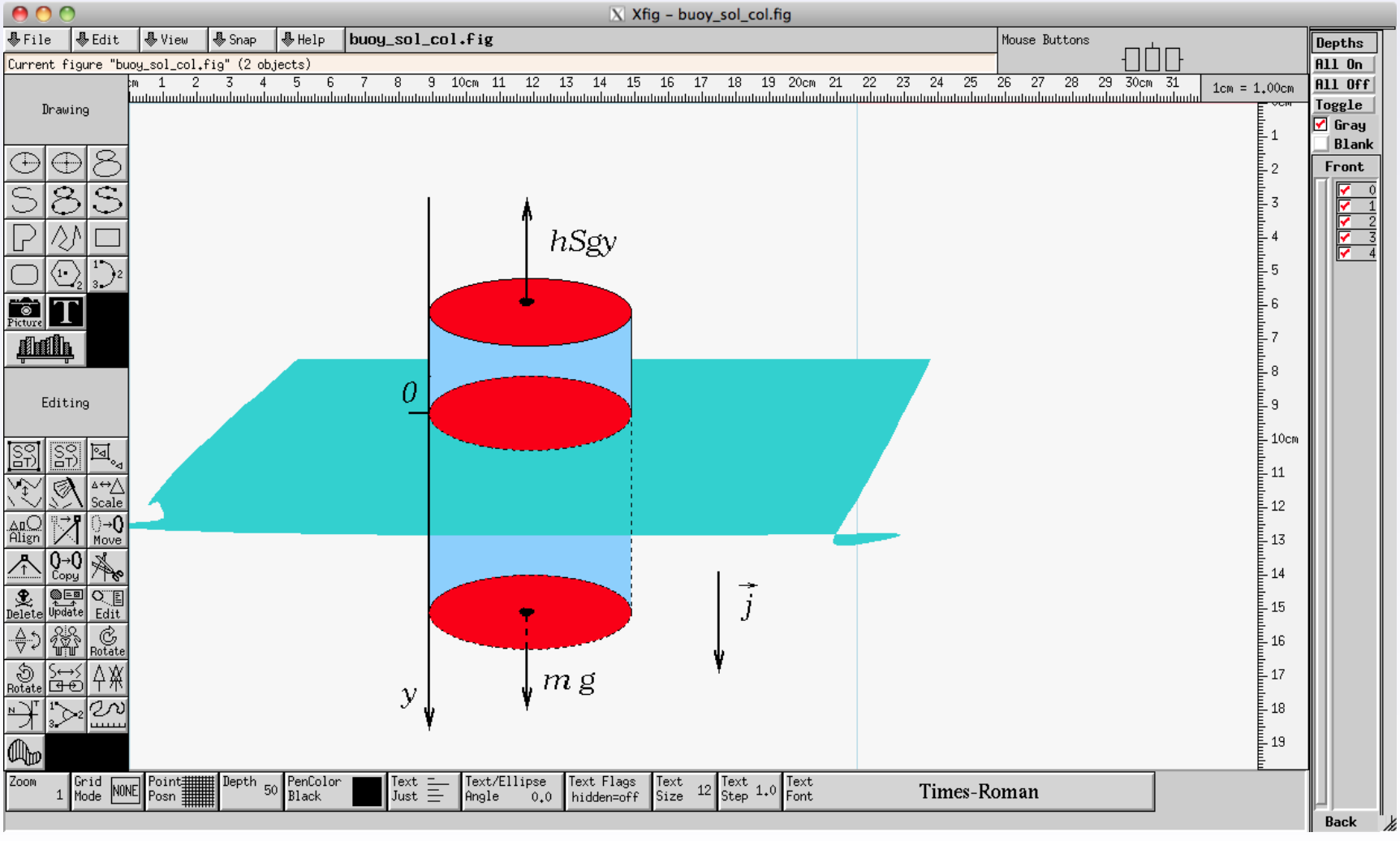
Faire un dessin avec Inkscape

L'environnement de travail de Inkscape :



Le logiciel XFig

XFig est un logiciel de dessin vectoriel compatible sur bien des points avec LaTeX.
On peut lui reprocher son interface vieillotte et peu compatible avec les canons actuels.



Le logiciel `fig2vect` permet de convertir des fichiers générés par XFig en des fichiers SVG éditables par Inkscape.

Autres logiciels de dessin à la souris

Il existe de nombreux logiciels de dessins vectoriels, soit gratuits, soit payants.

jfig Ce logiciel est une version JAVA de XFig. Il en reprend l'interface, le format de fichier, et les formats d'exportation.
Il est compatible avec LaTeX de la même façon que XFig.

Ipe C'est un logiciel libre destiné à la création de dessins vectoriels à insérer dans LaTeX.
Les objets de texte peuvent contenir des commandes LaTeX.

Adobe Illustrator (payant) C'est le logiciel de référence en matière de dessins vectoriels. Il est intégré à la suite Adobe Creative Suite, qui contient Adobe Photoshop (dessins BITMAP) et Adobe InDesign (mise en page).

CorelDRAW (payant) C'est le concurrent direct du logiciel précédent. Il est accompagné d'un logiciel de retouche photographique.

sK1 illustration program Ce logiciel libre est encore en développement.
Son but est d'obtenir un logiciel multiplateforme comparable à Adobe Illustrator et CorelDRAW.

D'autres logiciels plus confidentiels et plus spécialisés sont disponibles.
On trouvera sur la page **Comparison of vector graphics editors** une liste commentée de tels logiciels.

On peut désormais utiliser des logiciels en ligne (à l'intérieur d'un navigateur Internet) pour créer des dessins vectoriels, comme par exemple sur le site **Aviary**.

Illustrer les mathématiques

Où l'on conçoit que les données numériques sont mieux présentées avec des outils adéquats, que les courbes analytiques méritent des logiciels dédiés et que les constructions géométriques usuelles peuvent s'informatiser…

Différents types de logiciels

Différents problèmes mathématiques peuvent s'illustrer selon diverses méthodes, donc selon divers types de logiciels :

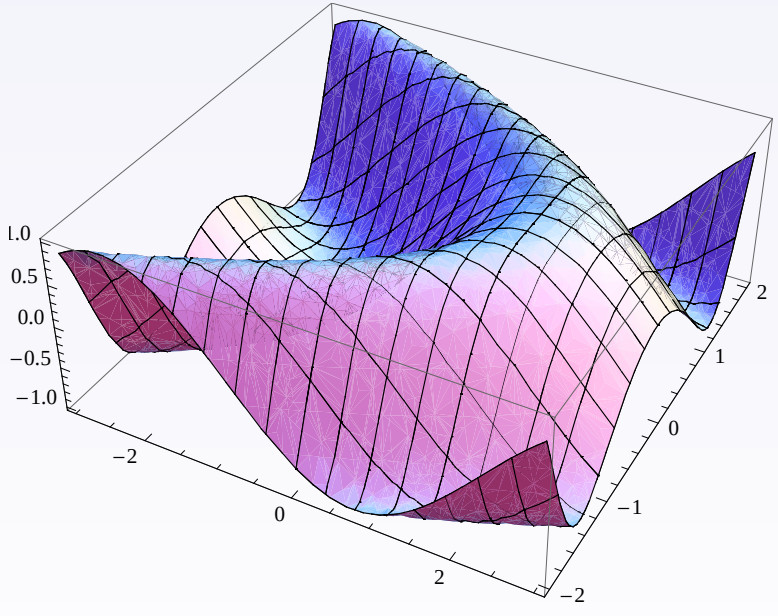
Les logiciels à vocations mathématiques : Il s'agit de logiciels utilisés pour manipuler des expressions mathématiques (analyse, algèbre, géométrie…), pour générer et traiter des données numériques (codes de simulations, traitements statistiques, visualisation dynamiques…) :
Mathematica, Maple, Matlab, Scilab, Maxima, Octave, Magma, Sage, GNUPlot…

Les logiciels de type tableurs : Logiciels de manipulation de séries de données, avec diverses possibilités de programmation :
Excel, Calc de OpenOffice…

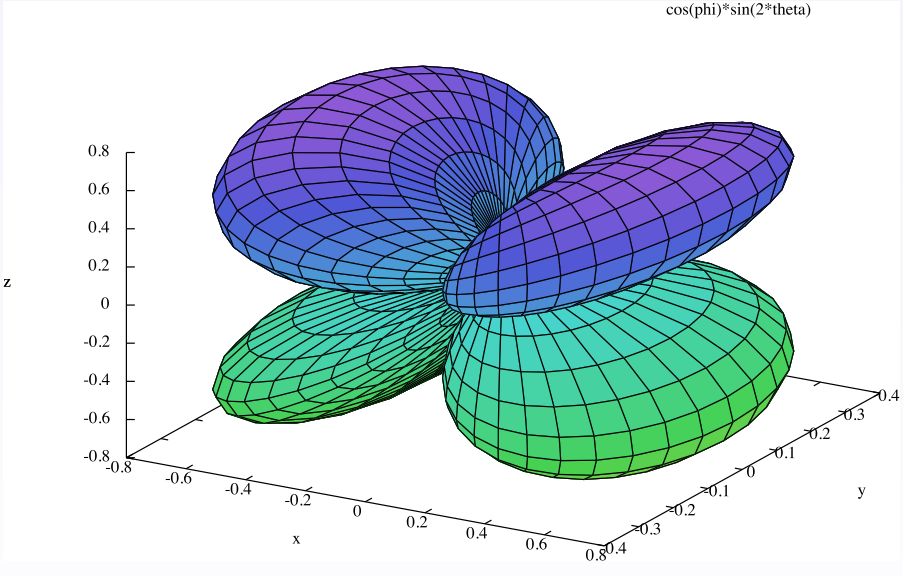
Les logiciels de "géométrie dynamique" : Logiciels dont le but est de réaliser des constructions géométriques en se basant sur des relations mathématiques précises :
GeoGebra, CaRMetal…

Les logiciels polyvalents

Mathematica, Maple, Maxima, Octave… sont des logiciels très polyvalents : calculs formels, manipulation et représentation d'expressions analytiques, traitement de données…
Ces logiciels sont prévus pour représenter graphiquement les données et les structures qu'ils manipulent.



Mathematica

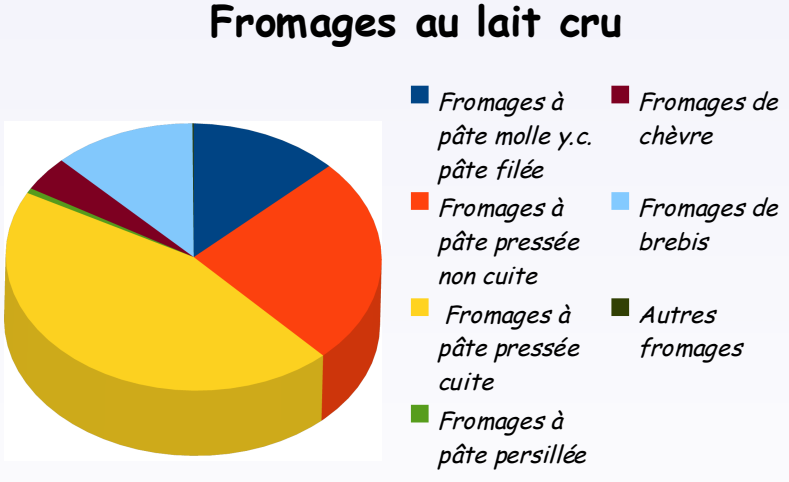


Maxima

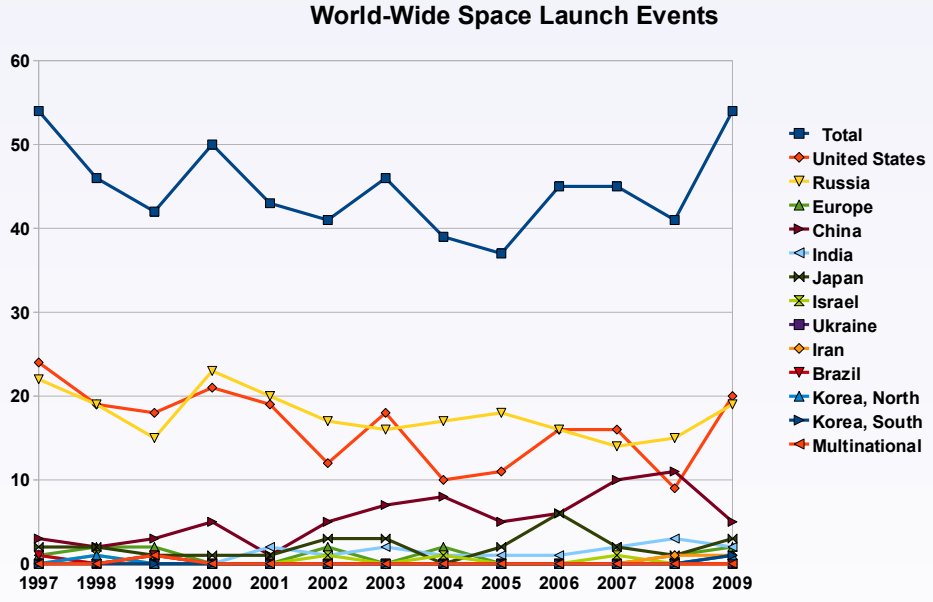
Les tableurs

Les logiciels de type tableurs permettent non seulement de traiter des grandes séries de données, mais aussi de représenter graphiquement ces données sans avoir à apprendre un langage spécifique et technique.

➟ rapidité de mise en œuvre, personnalisations possibles.



Calc de OpenOffice

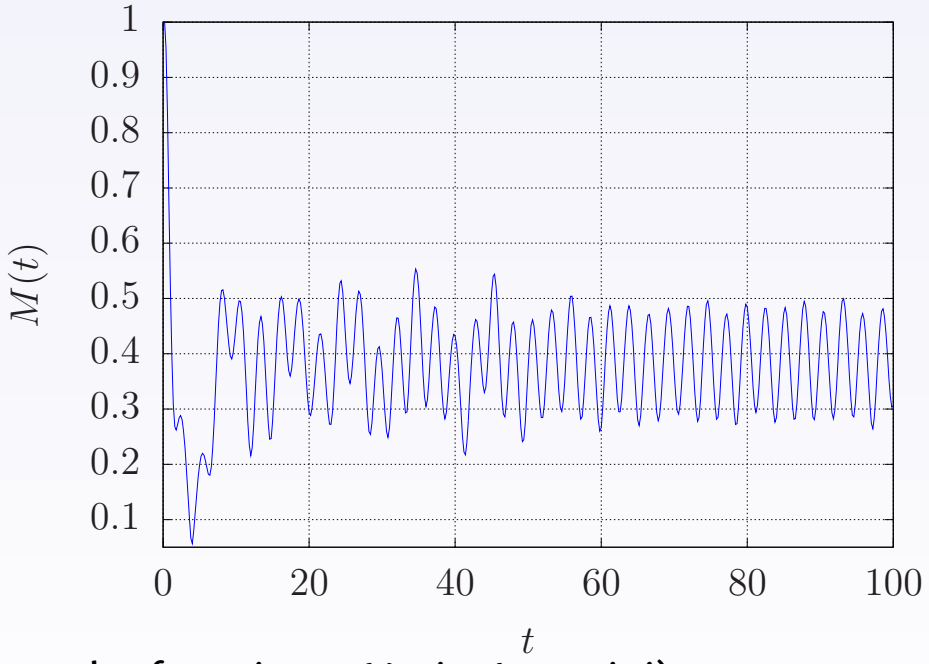


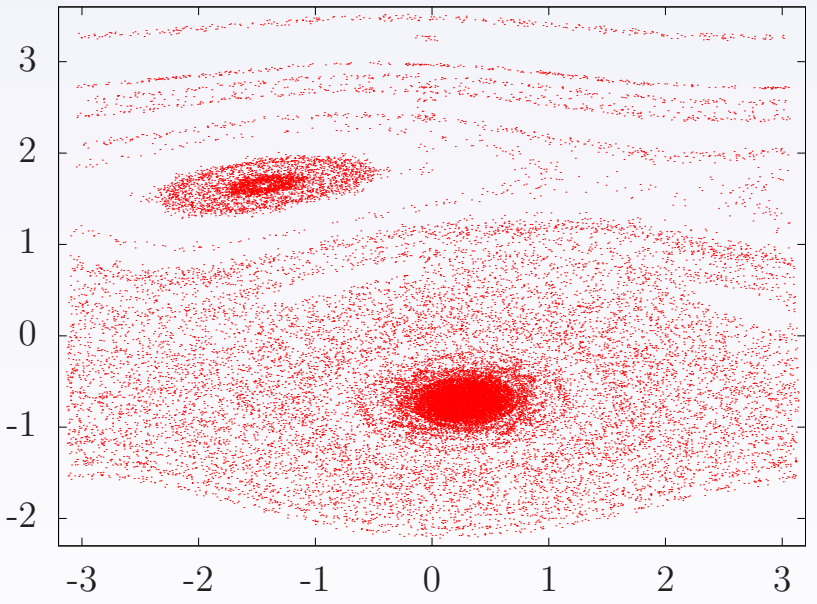
Calc de OpenOffice

Représentations de données avec GNUPlot

Le logiciel GNUPlot est très utilisé pour tracer des graphes, soit à partir de fonctions mathématiques, soit à partir de données numériques.

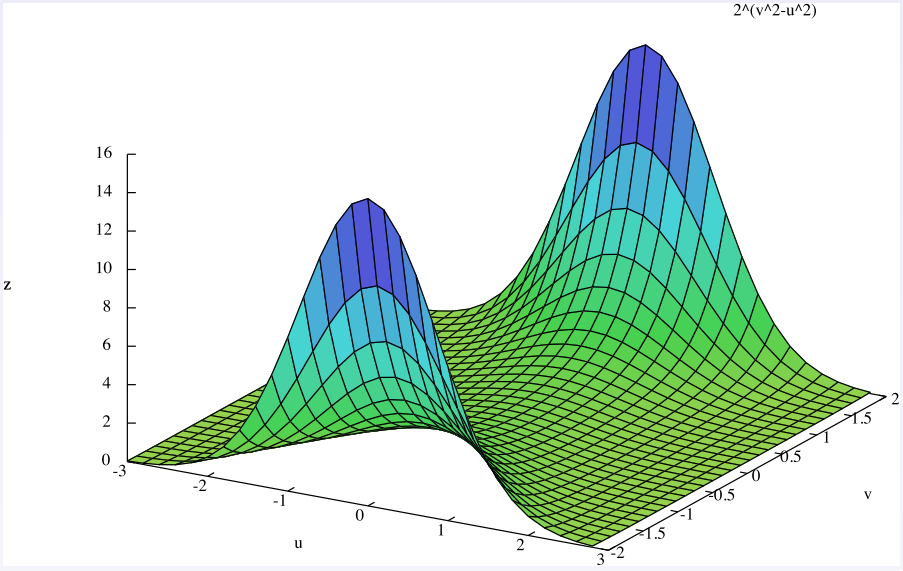
Il est utilisé dans Maxima et Octave pour la génération des graphes.
Il exporte dans de nombreux formats : EPS, PDF, PNG, SVG…
Il peut séparer l'image et les libellés pour que LaTeX s'occupe des libellés à la compilation.

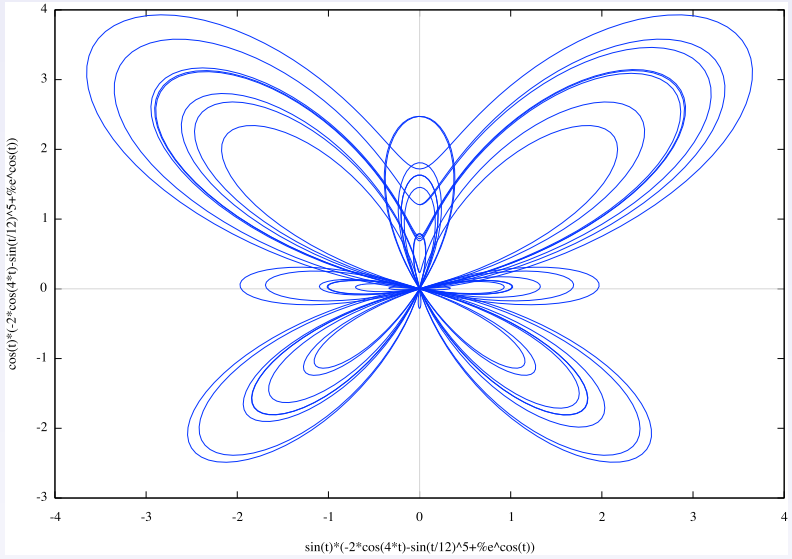


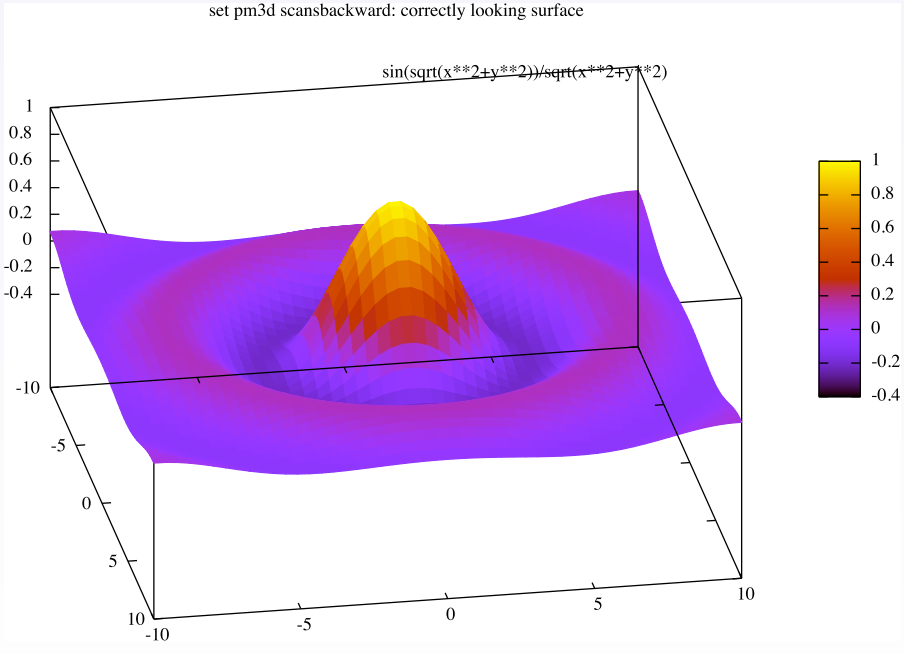


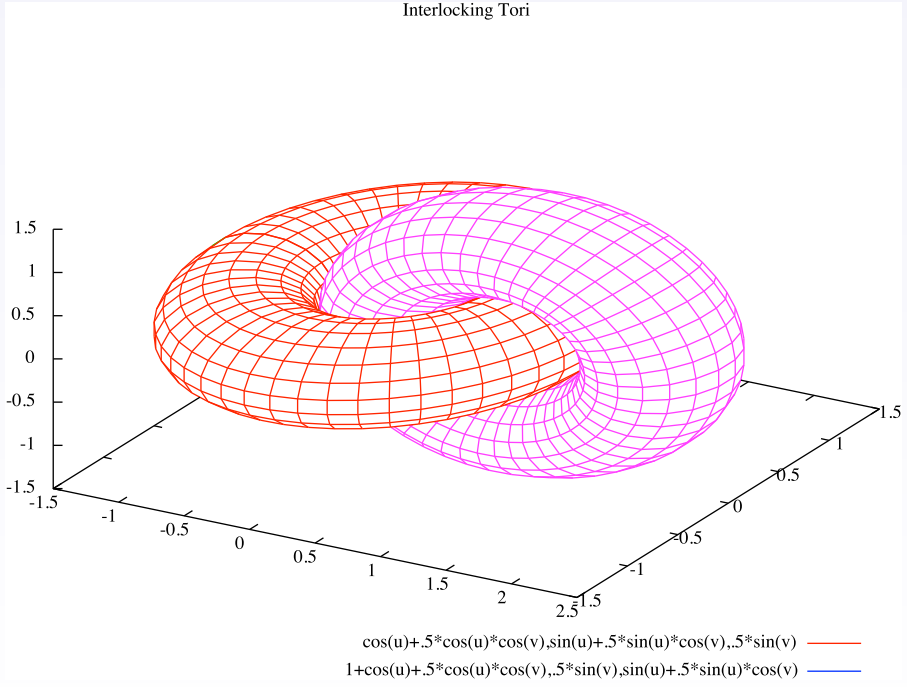
(Exemples fournis par Xavier Leoncini)

Représentations analytiques avec GNUPlot









On consultera la documentation pour plus de renseignements.

Les logiciels de géométrie dynamique

À l'origine, les logiciels de *géométrie dynamique* se donnent pour but de permettre de réaliser des constructions géométriques du type "à la règle et au compas".
En réalité ils font beaucoup plus : tracé de courbes analytiques, insertion de texte et d'images, programmation de dessins, réalisation de démonstrations dynamiques…

Entités élémentaires : points, droites, segments, demi-droites, cercles, arcs de cercles, polygones réguliers ou non, vecteurs, texte, images…

Opérations géométriques : placer des points au milieu de deux autres ou à l'intersection de deux droites, tracer des parallèles, des perpendiculaires, des médiatrices et des bissectrices, cercles passant par certains points…

Informations et tests : mesures de longueurs et d'angles, coordonnées des points, appartenance d'un point à une droite ou à un cercle, alignement de points, position relatives de deux droites (parallèles ou perpendiculaires)…

Fonctionnalités diverses : possibilité de bouger les objets initiaux (on voit la construction s'actualiser en direct), conservation de l'historique, exportations diverses (internet, images…)…

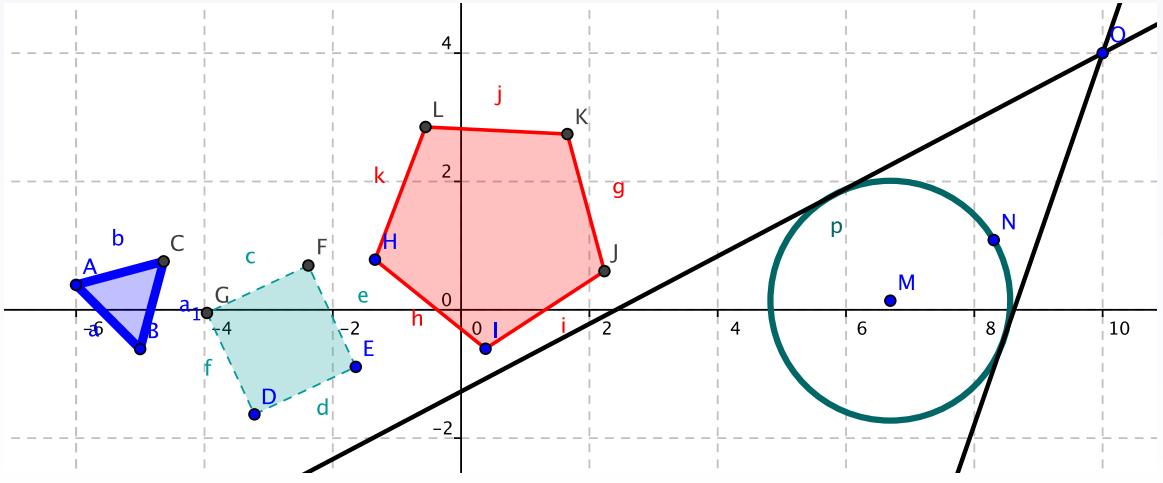
Utilisation : géométrie plane, géométrie dans l'espace, physique (mécaniques, électricité, astronomie, électromagnétisme)…

⚠ Ce ne sont pas des logiciels de dessin à la souris comme ceux évoqués auparavant : ils ne permettent pas par exemple de tracer des courbes de Bézier de façon native.

Les logiciels GeoGebra et CaRMetal

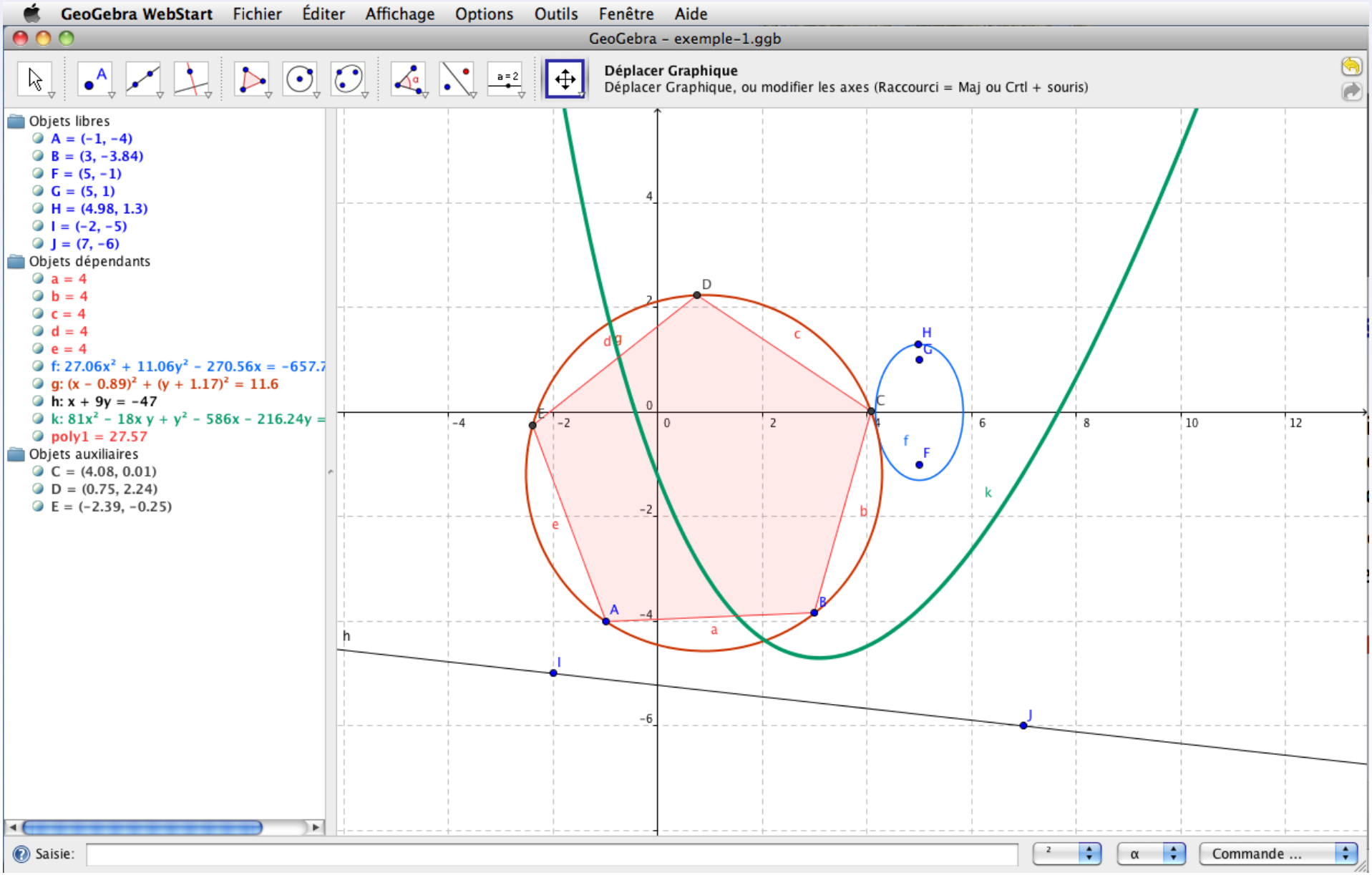
GeoGebra et CaRMetal sont des logiciels libres de géométrie dynamique.

- Ils sont écrits en JAVA ➙ disponibles sur toutes les plateformes.
- GeoGebra a des fonctions de tableur.
- Il est possible de programmer des dessins en JavaScript, de créer des outils auxiliaires, de créer des interfaces de contrôles.
- GeoGebra peut exporter en PNG, EPS et dans le format TikZ, ce qui le rend compatible avec LaTeX.
  CaRMetal peut exporter en PNG, EPS et SVG.
- Ces logiciels sont très utilisés dans le milieu enseignant du secondaire, et aussi au Baccalauréat.



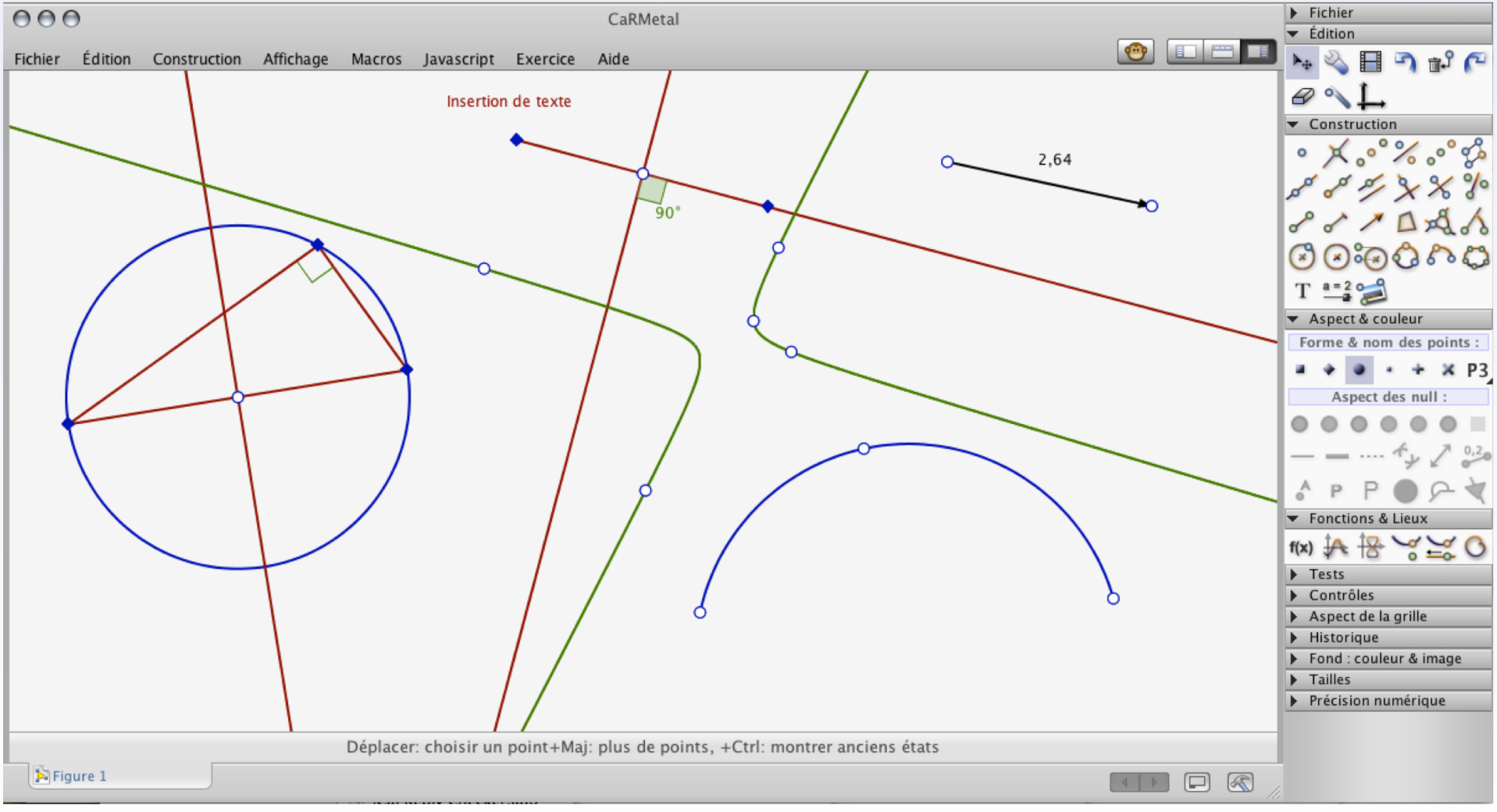
Faire un dessin avec GeoGebra

L'environnement de dessin de GeoGebra :



Faire un dessin avec CaRMetal

L'environnement de dessin de CaRMetal :



Programmer des graphiques

Où l'on découvre qu'il est possible de programmer un dessin comme on a appris auparavant à programmer du texte…

Les langages de description de dessins vectoriels

Les langages de description de dessins vectoriels peuvent être différents dans leur implémentation, mais ils utilisent les mêmes idées de base :

- Un fichier texte dans lequel le code de description du dessin est écrit.
- Un compilateur qui produit le dessin.
- Des variables qui représentent des longueurs, des épaisseurs de trait, des couleurs, des points dans le plan, voire des points dans l'espace, des chemins…
- Des commandes de dessin élémentaires : dessiner un contour, remplir un contour…
- Des commandes pour réaliser des formes standard : lignes droites, rectangles, cercles, ovales, arc de cercle, étoiles, croix…
- Définir des chemins basés sur les lignes droites et des courbes de Bézier quadratiques ou cubiques.
- Placer du texte sur le dessin, avec des choix typographiques divers, éventuellement le long de chemins compliqués…
- Des transformations géométriques élémentaires : translation, homothétie, réflexion, matrice linéaire…
- La possibilité de sauver un morceau de dessin pour l'utiliser à plusieurs reprises.
- Des commandes de programmation : des boucles, des conditions, des nombres aléatoires, des commentaires…
- Des modules externes qui ajoutent de nombreuses fonctionnalités.

Le logiciel Asymptote

Asymptote est issu d'une histoire liée à celle de TeX :

- En même temps que TeX, D. E. Knuth crée METAFONT, un moteur et un langage destinés à créer des fontes pour TeX.
- En 1994, John D. Hobby adapte le langage METAFONT et son moteur pour produire des fichiers EPS en sortie : c'est METAPOST.
- Asymptote a été créé en 2002 pour dépasser les limitations de METAPOST : calculs en haute précision, export dans différents formats d'images (EPS, PDF, SVG…).

Principaux avantages de Asymptote :

- Intégration très forte avec LaTeX pour placer des libellés dans les dessins, mais aussi pour créer les images (dans un document LaTeX).
- Langage de description de dessins très complet avec possibilité de l'enrichir avec des modules : formes usuelles, couleurs, transparences, dégradés, textures, boucles, conditions…
- Possibilité de résoudre des problèmes d'intersections de chemins, de trouver la valeur du paramètre d'une courbe en un point donné, de placer du texte sur un chemin courbe, de superposer des couches…
- Modules très divers : dessins en 3 dimensions, géométrie plane de type "collège" (perpendiculaires, triangles…), diagrammes de Feynman, statistiques, nœuds…
- Installé sur les dernières moutures de TeXLive ➡ accessible à tous par défaut.

Le logiciel Asymptote : utilisation

Il y a trois façons simples d'utiliser Asymptote :

1. Créer un fichier .asy dans lequel figure le code de l'image à composer.
   `asy -vv -f pdf image.asy` produit le fichier image image.pdf.
   asy est le moteur de compilation de Asymptote.
2. Créer un document LaTeX, nommé fichier.tex, avec les commandes

```
\usepackage{asymptote}
...
\begin{asy}
    code de l'image
\end{asy}
```

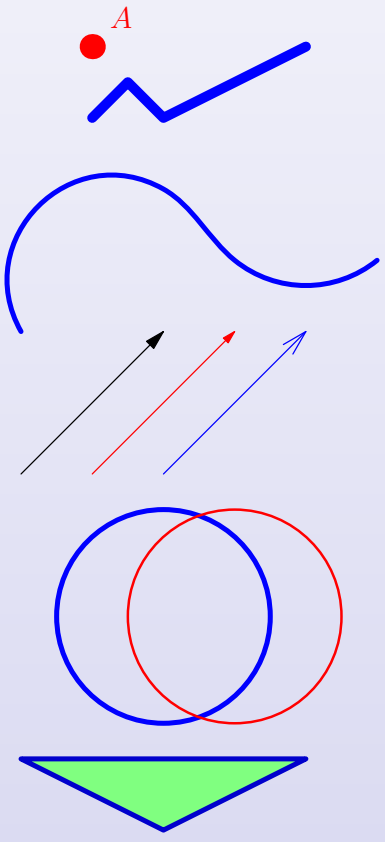
   `pdflatex fichier.tex` ➞ produit des fichiers fichier-n.asy (n numéro),
   `asy -vv fichier-*.asy` ➞ produit les images fichier-n.pdf
   `pdflatex fichier.tex` ➞ insère les images dans fichier.pdf
3. Créer un document LaTeX comme ci-dessus avec l'option

```
\usepackage[inline]{asymptote}
```

   et procéder de la même façon. Cette fois les libellés de texte placés dans le code de l'image seront composés par pdflatex lors de la seconde compilation : ils ne sont pas dans les fichiers images fichier-n.pdf.

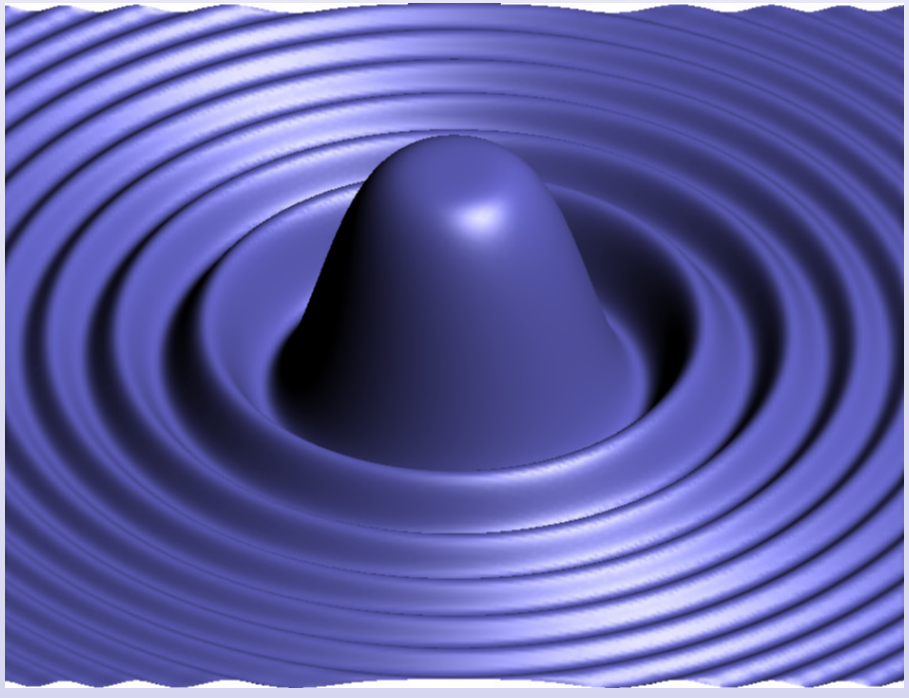
Le logiciel Asymptote : exemple

```
import settings;
outformat="pdf";
unitsize(1cm);

pair z=(1,6);
dot(Label("$A$",red),z,(2,2),red+10bp);
draw((1,5)--(1.5,5.5)--(2,5)--(4,6),blue+4bp);
draw((0,2)..(2,4)..(3,3)..(5,3),blue+2bp);

draw((0,0)--(2,2),Arrow);
draw((1,0)--(3,2),red,Arrow(5bp));
draw((2,0)--(4,2),blue,Arrow(SimpleHead,10bp));

path p=circle((2,-2),1.5);
draw(p,blue+2bp);
draw(shift(1,0)*p,red+1bp);

path q=(0,-4)--(4,-4)--(2,-5)--cycle;
filldraw(q,lightgreen,2bp+.8blue);
```



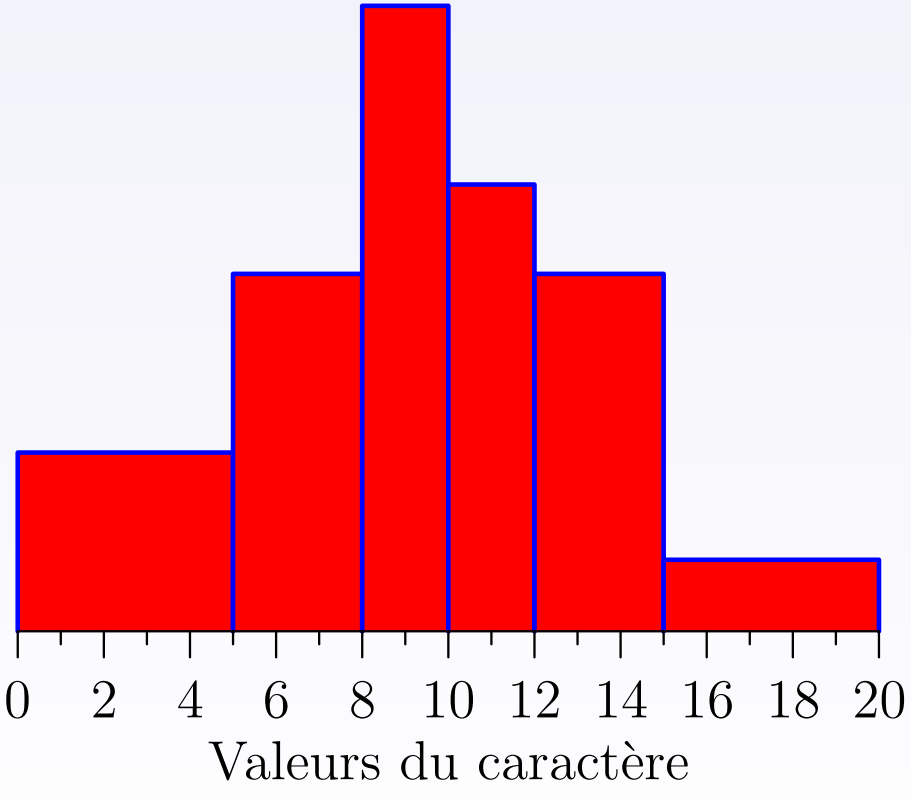
Courbes analytiques avec Asymptote

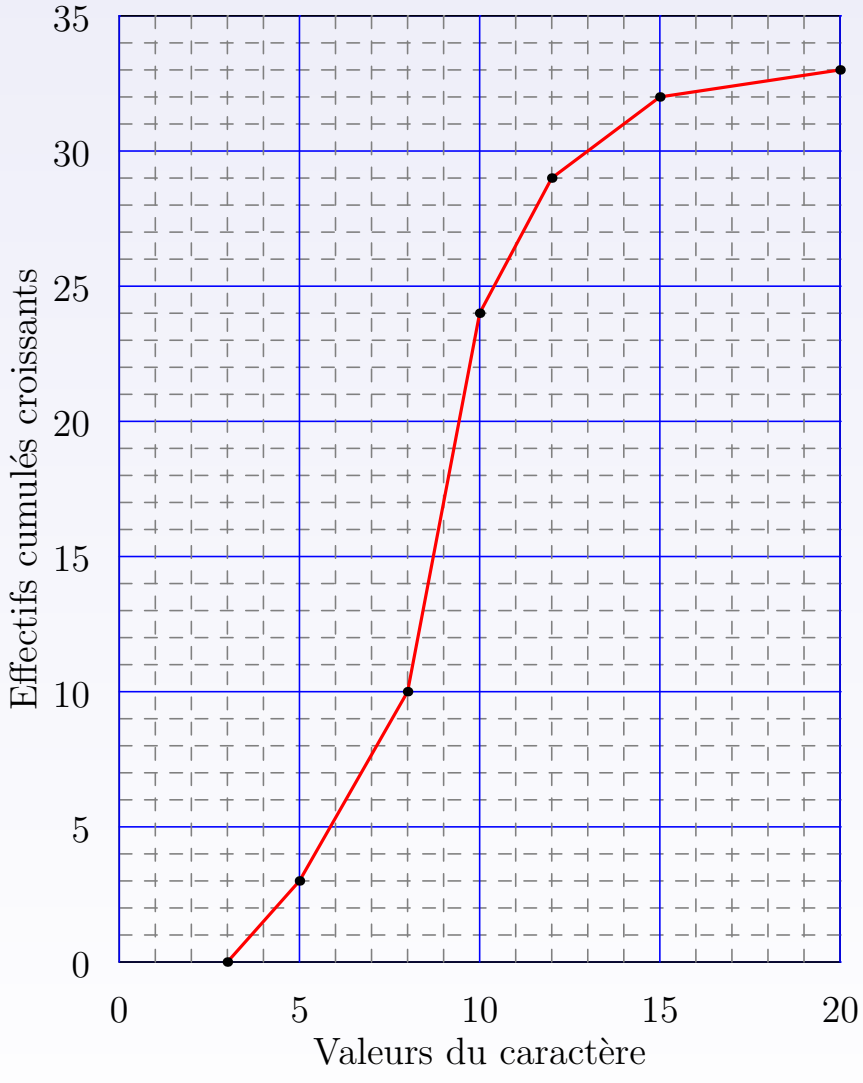
```
import graph3;
size(7.5cm,0);
currentprojection=orthographic(2,0,1.5);
real f(pair z) {
        real r=2pi*(z.x^2+z.y^2);
        if (r!=0)  return sin(r)/r;
                else return 1;}
draw(surface(f,(-2.5,-2),(2.5,2),100,Spline),lightblue);
```



Représentations de données avec Asymptote

Le module `stats` de Asymptote permet de gérer des données statistiques.





(Exemples extraits de **Asymptote - Galeries d'exemples**)

Le *package* TikZ

Le *package* **tikz** repose sur le *package* **pgf**, *Portable Graphic Format*.
Origine du nom : TikZ = "TikZ ist *kein* Zeichenprogramm"

pgf est un langage de dessin vectoriel bas niveau, qui définit des primitives essentielles pour dessiner. Il est construit comme couche intermédiaire entre l'utilisateur et les *drivers* utilisés par TeX pour produire les fichiers .dvi, .ps et .pdf.

TikZ est une surcouche de **pgf** qui définit des commandes riches et intuitives pour dessiner. On peut l'étendre avec des modules supplémentaires.

Principaux avantages de TikZ :

- 100% LaTeX ! Le document est donc unique, pas de fichiers externes.
- TikZ peut intervenir ailleurs que dans des boîtes d'images : encadrement de textes, entêtes et bas de pages, tableaux, flèches entre éléments du texte (même page)…
- Langage de programmation compatible avec celui de LaTeX : familiarité, utilisation dans des commandes LaTeX personnelles…
- Nombreuses fonctionnalités de dessin : formes usuelles, couleurs, transparences, dégradés, textures, boucles, conditions, couches…
- Nombreux modules pour des fonctionnalités diverses : matrices, arbres, flèches, décorations, calendriers, chaînes, ombres…
- Installé par défaut avec TeXLive ➡ utilisable partout où il y a LaTeX.

La gourmandise de TikZ

TikZ est très gourmand en registres (longueurs, compteurs…).

Lorsqu'on l'installe avec d'autres *packages*, comme par exemple **xypic**, il arrive qu'on sature ce nombre de registres.
On aboutit alors à des messages d'erreurs du type :

```
! No room for a new \dimen
! No room for a new \count
```

Heureusement, le moteur pdftex (basé sur ε-TeX, une extension de TeX apparue en 1996) utilisé par défaut aujourd'hui peut allouer beaucoup plus de registres :
de $256 = 2^8$ on passe à $32768 = 2^{15}$.

Mais ces registres ne sont pas disponibles par défaut ! Il faut utiliser :

```
\usepackage{etex}
```

juste après `\documentclass[-]{-}` pour activer ces registres supplémentaires.
Il doit être placé avant l'installation d'autres *packages* afin que ces derniers profitent de tous les registres disponibles.

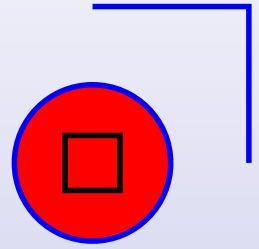
Lorsqu'on utilise TikZ, il est préférable d'utiliser le *package* **etex**.

Cette solution est préconisée dès que le problème de saturation intervient.

La syntaxe de TikZ : les bases

Les figures créées avec TikZ sont contenues soit dans la commande `\tikz{-}`, soit dans l'environnement `tikzpicture`. Pas de différence de rendu entre les deux syntaxes. Cette commande et cet environnement acceptent des options qui seront appliquées à tout le dessin en cours.

```
\usepackage{tikz}
...
\begin{tikzpicture}[line width=1pt,fill=red,draw=blue]
\path[draw] (0,1)--(1,1)--(1,0);
\filldraw (0,0) circle (0.5cm);
\draw[black] (-5pt,-5pt) rectangle (5pt,5pt);
\end{tikzpicture}
```



```
\tikz[line width=2pt,fill=orange,draw=red]{
\draw[fill] (0,1)--(1,1)--(1,0)--cycle;
\draw[dashed] (0,-0.5) to[out=45,in=135] (1,-0.5);
}
```

TikZ peut être étendu à l'aide de bibliothèques qu'on appelle dans le préambule :

```
\usetikzlibrary{biblio1,biblio2,...}
```

La syntaxe de TikZ : les fondamentaux

Un dessin TikZ obéit aux règles suivantes.

- Les éléments d'un dessin sont des chemins : `\path ... instructions ... ;`.
  ➟ `\path` admet des variantes : `\draw`, `\fill`, `\filldraw`, `\node`, `\coordinate`, `\shade`, `\shadedraw`…
  ⚠ La construction d'un chemin se termine avec `;`.
  ➟ La lecture/construction d'un chemin se fait de gauche à droite.
- Les éléments d'un chemin sont des "opérations" élémentaires qui utilisent le *point courant actuel* et qui définissent le nouveau point courant : aller à un point, insérer une ligne droite, une courbe de Bézier, une forme (rectangle, cercle, ellipse…), un arc, une grille, un graphique…
- En cours de réalisation, le chemin *accumule des options* (`[-]`) : dessiner, remplir, épaisseur du trait, couleurs, style du trait… *Sans option, un chemin est créé sans être rendu.*
  Le chemin une fois terminé (arrivé au `;`) est rendu selon les options données.
- Des nœuds (`node`) peuvent être insérés en cours de réalisation.
  Ce sont des morceaux de textes qui seront ajoutés *après* le rendu du chemin.

Les points peuvent être nommés pour les utiliser après.
Il est possible de définir et de nommer des "styles" utilisables dans tous les dessins.

```
\path[draw] (0,0) circle (5pt) to[out=20,in=240] (1cm,2cm) [fill]
        node[xshift=10pt] {$A_1$} -| (90:10pt) [thick] -- cycle;
```

Bien se coordonner avec TikZ

TikZ définit différents types de coordonnées.

Coordonnées cartésiennes absolues : ce sont des couples `(`⟨*dimx*⟩`,`⟨*dimy*⟩`)`.

Coordonnées cartésiennes relatives : ce sont des facteurs `(`*fx*`,`*fy*`)`, relativement aux unités par défaut définies par `x=1cm` et `y=1cm`.

Coordonnées polaires : ce sont des couples `(`*angle*`:`*rayon*`)`, où l'angle est en dégrés.

Coordonnées des nœuds : les nœuds peuvent être nommés et il est possible d'utiliser ces noms comme points.
`\coordinate (`*nom*`) at (`*point*`);` définit un tel point.

Coordonnées relatives : `++(`*coord.*`)` et `+(`*coord.*`)` translatent le point courant des coordonnées `(`*coord.*`)` (cartésiennes, polaires…).
La première syntaxe redéfinit le point courant, la seconde non.

Expression mathématique : avec la bibliothèque `calc`, on peut écrire des expressions mathématiques du type `($f*(`*point*`)+g*(`*point*`)...$)`.

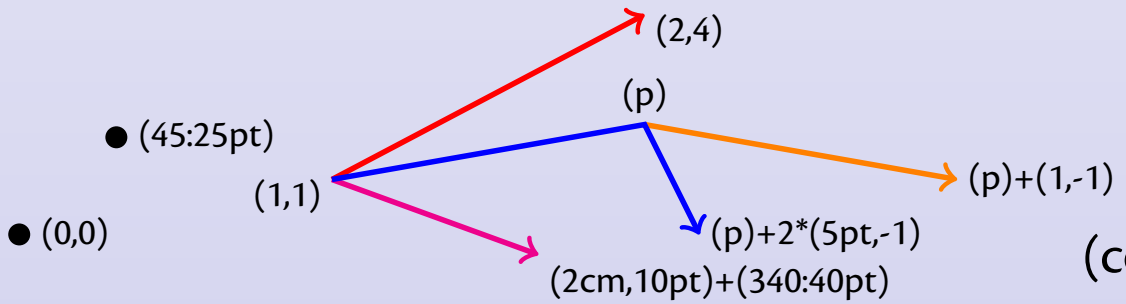
Coordonnées barycentriques : il est possible de désigner un point comme barycentre d'autres points.

Coordonnées calculées : point de tangence à un chemin, point d'intersection entre deux chemins…

D'autres systèmes de coordonnées peuvent être construits.

Bien se coordonner avec TikZ : exemples

```
\usepackage{tikz}
\usetikzlibrary{calc}
...
\begin{tikzpicture}[x=2cm,y=10pt,line width=1pt]
\fill (0,0) circle (2pt);
\fill (45:25pt) circle (2pt);

\draw[red,->] (1,1)--(2,4);

\draw[magenta,->] (2cm,10pt)--++(340:40pt);

\coordinate (p) at (2,2); \draw[orange,->] (p)--+(1,-1);

\draw[blue,->] (1,1)--(p)--($(p)+2*(5pt,-1)$);
\end{tikzpicture}
```



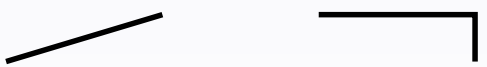
(coordonnées ajoutées)

Faire son chemin avec TikZ : opérations élémentaires

Un chemin commence toujours avec la commande `\path` (ou variante).
Il continue avec une série d'opérations élémentaires utilisant chacune le point courant.
Chaque opération définit un nouveau point courant, en général il s'agit du dernier point mentionné, mais pas toujours.

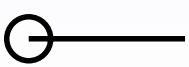
- `(`*point*`)` : le point courant devient le point indiqué.
- `--(`*point*`)` : ligne droite entre le point courant et le point indiqué.
  `-|(`*point*`)` et `|-(`*point*`)` : lignes brisées (horiz.-vert., vert.-horiz.) jusqu'au point.
  `to[-]` *(point)* définit une ligne (droite par défaut), ses options définissent l'angle de départ et d'arrivée.
- `..controls` *(point)* `and` *(point)* `..` *(point)* trace une courbe de Bézier.
- `circle[-](`*rayon*`)`, `rectangle[-]` *(point)*, `ellipse[-]`, `arc[-]`, `sin` *(point)*, `cos` *(point)*, `parabola[-]` *(point)*, `grid[-]` *(point)*… insèrent des formes.
  Le point courant n'est pas nécessairement modifié.
- `--cycle` ferme un chemin.

```
\draw (0,0)--(1,0.3) (2,0.3)-|(3,0);
```



```
\draw (0,0) rectangle (12pt,10pt)--(1,0);
```

```
\draw (0,0) circle (4pt) --(1,0);
```



```
\draw (0,0) .. controls (1,1) and (2,1) .. (4,0);
```

Faire son chemin avec TikZ : options de rendu

- `[draw]` ou `[draw=`*couleur*`]` : le chemin est dessiné dans la couleur mentionnée.
- `[fill]` ou `[fill=`*couleur*`]` : le chemin est rempli avec la couleur mentionnée.
  Il faut terminer le chemin par un `--cycle` (sinon TikZ l'ajoute).
- `[color=`*couleur*`]` : installe la couleur désignée, elle remplace celles de `draw` et `fill`.
- `[line width=`⟨*dim*⟩`]` : épaisseur du trait. Des épaisseurs prédéfinies sont fournies :
  `ultra thin`, `very thin`, `thin`, `semithick`, `thick`, `very thick`, `ultra thick`.
- `[arrows=...]` : définit le type de flèches à insérer au début et à la fin du chemin.
  `<-`, `>-`, `o-`, `>>-`, `|-`, `|>-`, `->`, `-<`, `-o`, `->>`, `>->`, `o->`,…
  `[>=...]` : détermine le type de bout de flèche, `to`, `latex`, `stealth`.
  La bibliothèque `arrows` ajoute d'autres types de flèches.
- `[double=`*couleur*`]`, `[double distance=`⟨*dim*⟩`]`, … : définit les traits doubles.
- `[dash pattern=on` ⟨*dim*⟩ `off` ⟨*dim*⟩ `on` ⟨*dim*⟩ `...]` : définit le type de trait.
  Styles prédéfinis : `solid`, `dotted`, `dashed`, `dashdotted`…
- `[rounded corners]` ou `[rounded corners=`⟨*dim*⟩`]` : les brisures de lignes droites sont arrondies selon le rayon donné (ou un rayon par défaut).
- De nombreuses autres options sont disponibles : `rotate`, `scale`…

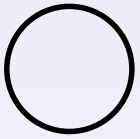
La commande `\path` est fondamentale, les autres en dérivent :
`\draw`=`\path[draw]`, `\fill`=`\path[fill]`, `\filldraw`=`\path[fill,draw]`.

➟ Les options passées aux commandes `\draw`, `\fill`, … sont passées à `\path`.

⚠ **Les options les plus récentes écrasent les anciennes de même nature.**

Faire son chemin avec TikZ : exemples d'options

```
\path [draw] (0,0) circle (4mm);
\path (0,0) [draw] circle (4mm);
\path (0,0) circle (4mm) [draw];
```



→ On peut écrire la même commande de différentes façons.

```
\path[fill=black] (0,0) circle (2mm) [fill=red];
```

```
\draw[red,line width=3pt] (0,0)--(1,0.3)-|(2,0);
```

```
\fill[orange] (0,0) rectangle (24pt,12pt);
```

```
\draw[dotted] (12,0) +(30:10pt) arc (30:95:10pt);
```

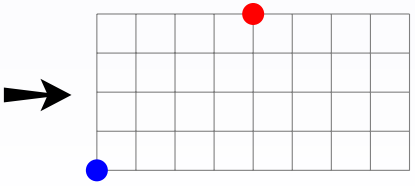
(centre ajouté)

```
\draw[blue,line width=2pt,dashed] (0,0)--(1,0.3);
```

```
\draw[double=white,thick,|->] (0,0)--(10,1);
```

```
\draw[rounded corners,dashdotted] (0,2)--(5,0)-|(8,2);
```

```
\draw[help lines] (0,0) grid[step=.25cm] (2,1);
\fill[blue] (0,0) circle (2pt);
\fill[red] (1,1) circle (2pt);
```



Un sac de nœuds dans TikZ

Un nœud est un objet graphique qui peut contenir du texte ajouté au chemin courant.
Il est défini lors de la construction du chemin, mais il est dessiné *après* le chemin.
Un nœud a une forme et donc une taille.

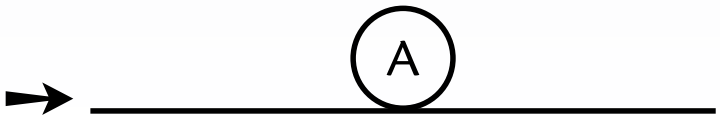
La syntaxe générale est la suivante :

```
\path ... node[options] (nom) at(point) {texte} ...;
```

- Les options `[options]` définissent la forme et donnent les instructions de rendu.
- `(nom)` est optionnel : ce nom peut servir de point nommé utilisable après.
  Selon la forme, on peut faire référence à certains points sur le bord du nœud :
  `(nom.south)`, `(nom.south east)`, `(nom.north west)`, `(nom.60)` (60°)…
- `at(point)` est optionnel : par défaut le nœud est placé au point courant.
  Avec cette option, il peut être placé ailleurs.
  Un nœud ne change pas le point courant du chemin dans lequel il s'insère.
- `{texte}` est le texte que contient le nœud si la forme (`shape`) l'accepte.
  Pour certaines formes, les accolades sont obligatoires mais peuvent être vides.

TikZ définit les raccourcis suivants : `\node`=`\path node`,
`\coordinate`=`\path coordinate`=`\node[shape=coordinate]`.

```
\draw (0,0)--(2,0) node[anchor=south,circle,draw] {A} --(4,0);
```



Un sac de nœuds dans TikZ : les options

Les options de nœuds les plus importantes sont :

Forme : `shape=` parmi `circle`, `rectangle`, `coordinate`.

`coordinate` n'accepte pas de texte ➡ crée un point "vide", donc sans taille.

Les bibliothèques `shape.geometric` et `shape.misc` ajoutent d'autres formes.

Rendu : `fill`, `draw`, `rotate`, `scale`…

➡ couleurs de rendu, angle de rotation, facteur d'échelle…

Les options usuelles des chemins sont souvent utilisables.

⚠ Certaines options du chemin ambiant ne sont pas appliquées aux nœuds.

Géométrie : `inner sep`, `outer sep`, `minimum height`, `minimum width`…

➡ taille de la forme, marges autour du texte, marges externes…

Positionnement : `anchor=` parmi `south west`, `north east`, `base`, `center`, `east`…

➡ point d'ancrage du nœud sur le point courant en fonction de sa forme.

`above=`⟨*dim*⟩ équivaut à `anchor=south` avec un décalage vertical de ⟨*dim*⟩.

➡ `below=`⟨*dim*⟩, `right=`⟨*dim*⟩, `left=`⟨*dim*⟩, `above right`, `below left`…

La bibliothèque `positioning` étend ces options.

Texte : `text=`*couleur*, `font=`… sont les attributs du texte ;

`align=` parmi `left`, `center`, `right`… pour la mise en forme ;

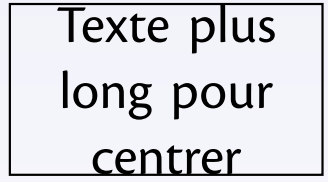
`text width`, `text height`, `text depth`… pour l'aspect général.

Dénouer les nœuds de TikZ

```
\node[draw,rectangle,thick,rotate=30] {Texte};
```
→ 


```
\node[font=\bfseries] at (0,0) {Texte};
```
→ **Texte**

```
\fill[blue] (0,0) circle (4pt) node[left=3pt,red]{$A$};
```
→ 


```
\node[text width=2cm,align=center,
      inner sep=0pt,draw]
         at (0,0) {Texte plus long pour centrer};
```
→ 

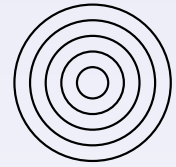

```
\node[top color=blue!10,bottom color=blue!50,
      rounded corners=5pt,
      draw=red!80!black,thick]
         at (0,0) {Texte};
```
→ 


```
\usetikzlibrary{shapes.symbols}
\node[starburst,fill=yellow,
      draw=red,line width=2pt,
      inner sep=6pt]
  at (0,0) {\bfseries Vive \LaTeX{} !};
```
→ 

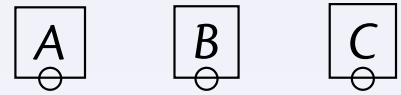
Autres fonctionnalités de TikZ

TikZ admet quelques fonctionnalités de programmation comme les boucles :

```
\foreach \r in {1,2,...,5} \draw (0,0) circle (\r mm);
```



```
\foreach \x/\t in {1/A,2/B,3/C}{
   \node[rectangle,draw,above] at (\x,0) {$\t$};
   \draw (\x,0) circle (2pt);}
```



Il est possible de tracer des courbes analytiques de fonctions usuelles :

```
\draw[red] plot [domain=-1:1] (\x,{(\x)^2});
```

```
\draw plot[domain=-pi:pi] (\x,{sin(\x r)});
```

TikZ permet de définir des “styles” qui résument sous un nom court un lot de réglages. Ces styles sont utilisables dans plusieurs graphiques.

```
\tikzset{rond bleu/.style={draw,color=blue,circle,thick}}
\tikz{\node[rond bleu] at (0,0) {1};}
```

Si on n’utilise un style que pour un seul graphique, il est possible de le définir dans la partie optionnelle de `\tikz` ou de `tikzpicture` (sans la commande `\tikzset{-}`).

Les modules externes de TikZ

TikZ est accompagné de nombreux modules externes qui ajoutent des fonctionnalités.

`calc` : Permet de spécifier des coordonnées par des opérations mathématiques.
`($({1+1}*(1,.5)$)`, `($cos(45)*sin(30)*(1,1)$)`

`intersections` : Permet de lister les points d'intersection entre deux chemins.

`positioning` : Permet de positionner des nœuds les uns relativement aux autres.

`matrix` : Permet de distribuer des nœuds sur un réseau de points.
➡ Syntaxe semblable à celle des tableaux LaTeX…

`arrows` : Ajoute des types de flèches.

`shapes` : Ajoute de nouvelles formes pour les nœuds (`shape`).

`decorations` : Ajoute des "décorations" au chemins : ondes, ressort, symboles, zigzags…

`plotmarks` : Ajoute des nouveaux types de points pour les graphes.

`shadows` : Ajoute des types d'ombres semi-transparentes.

`shadings` : Ajoute différents types de dégradés de couleurs.

`patterns` : Ajoute des types de motifs pour remplir des zones fermées.

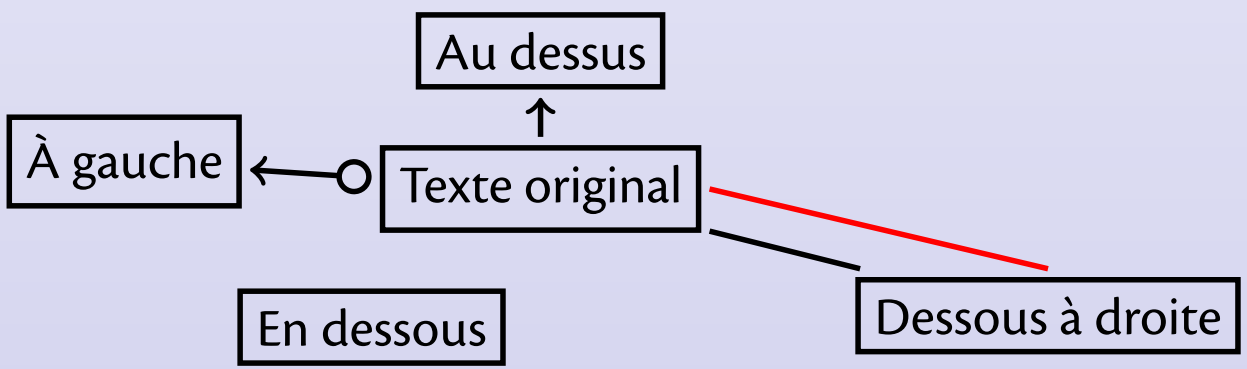
`fit` : Permet d'ajuster un nœud pour qu'il contienne un ensemble de coordonnées.

`external` : Donne la possibilité d'exporter facilement un dessin en un fichier externe.

Voir aussi :
`fading`, `backgrounds`, `circuits`, `automata`, `calendar`, `chains`, `mindmap`…

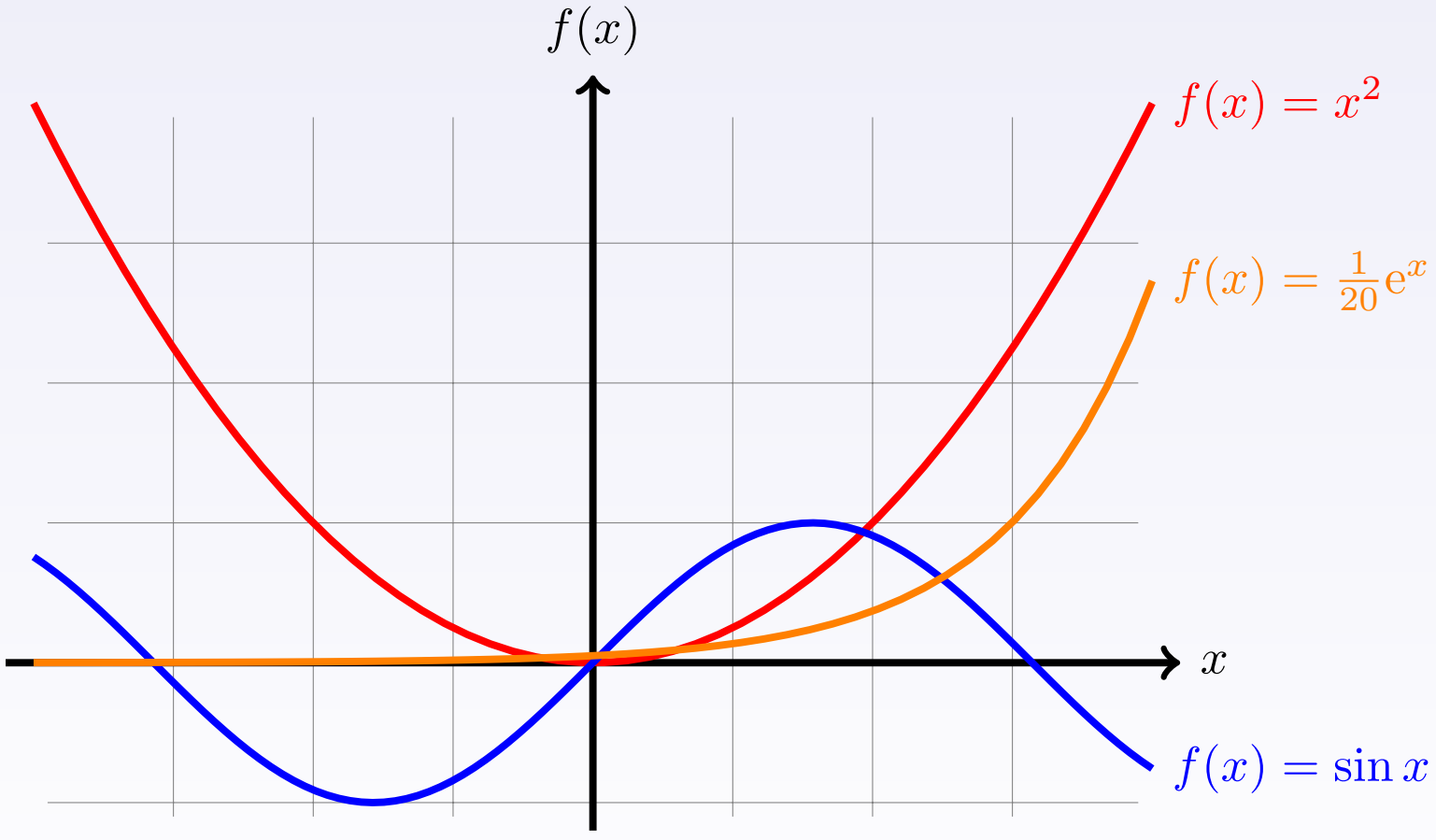
Illustration du module positioning

```
\usetikzlibrary{positioning}
...
\begin{tikzpicture}[node distance=0.25cm,line width=1pt,
  every node/.style={draw,rectangle,inner sep=3pt,outer sep=2pt}]

\node (TO) {Texte original};% par défaut: (0,0)
\node[above=of TO] (AD) {Au dessus};
\node[left=of TO,yshift=+5pt,xshift=-15pt] (AG) {À gauche};
\node[below right=of TO,xshift=+20pt] (DD) {Dessous à droite};
\node[below=of TO.south west] (ED) {En dessous};

\draw[->] (TO)-(AD);   \draw[o->] (TO)-(AG);
\draw (TO)-(DD);       \draw[red] (TO.east)-(DD.north);
\end{tikzpicture}
```



Courbes analytiques dans TikZ avec GNUPlot

Avec l'aide de GNUPlot, il est possible de tracer des courbes analytiques.
TikZ délègue à GNUPlot le soin de préparer un certain nombre de points de la courbe.



Lors d'une première compilation, TikZ génère un fichier .gnuplot qu'il faut faire interpréter par GNUPlot pour produire des fichiers de points.
Une seconde compilation permet à TikZ de lire ces fichiers de données.

Quelques conseils pour réaliser un dessin

À partir d'une idée précise, comment réaliser un dessin programmé ?

Quelques conseils pratiques pour concrétiser ses projets :

1. Dessiner un croquis à main levé sur du papier.
   ➟ En extraire la taille globale évaluée du dessin final.
2. Repérer les points essentiels, les relations entre les objets, les structures mathématiques (courbes, intersections, tangentes, perpendiculaires…)…
   ➟ Cette étape est compliquée mais cruciale, car elle détermine la suite.
3. Créer un fichier LaTeX uniquement pour l'image.
   ➟ Compilations rapides et donc fréquentes.
   Créer un environnement `tikzpicture` et placer une grille à la taille évaluée :
   `\draw[step=.5cm,help lines] (-1,-3) grid (10,3);`
   Préférer les coordonnées relatives (`1cm`).
4. Placer les points essentiels du dessin en les nommant :
   `\coordinate (p1) at (2,3);`, *etc*
   ➟ Les visualiser momentanément avec `\draw (p1) circle (3pt);`
5. Concrétiser les relations structurelles entre les objets, en partant des points nommés.
   ➟ C'est la partie la plus difficile, où la documentation peut se révéler très utile !
6. Commenter les aides au dessin : grille de fond, points de structure…
7. Fignolage : taille globale avec les options `x=...` et `y=...`, épaisseurs de certains traits, ajustement de la position de certains labels, mise en place de styles…

Asymptote ou TikZ ?

La finalité et l'utilisation de ces deux "logiciels" de dessin ne sont pas les mêmes :

- La puissance de calcul de Asymptote est supérieure à celle de TikZ.
  ➟ Opérations de base plus nombreuses, fonctions analytiques, manipulations mathématiques avancées, dessins en 3D, rapidité du rendu…
- Asymptote produit par défaut des dessins extérieurs.
  ➟ Gain de temps à chaque compilation du document, usages diversifiés…
- La syntaxe de Asymptote est indépendante de celle de LaTeX.
  ➟ Plus proche de langages de programmation courants…
- TikZ interagit très facilement avec les autres éléments d'un document.
  ➟ Utilisation des mêmes polices d'écriture, création de cadres et de fonds colorés, décorations dans l'entête ou le pied de page, insertion de dessins "dynamiques" dans `beamer`, création de "tableaux" et de diagrammes mathématiques…
- TikZ définit la notion de style.
  ➟ Séparation du fond sémantique du dessin et de sa forme, code plus léger…
- TikZ admet plus de possibilités quant au positionnement des éléments entre eux.
  ➟ Inutilité de recourir à des coordonnées exactes, grand choix de points d'ancrage sur des éléments existants, création de diagrammes structurels plus aisée…

➟ Il faut définir correctement ses besoins pour choisir l'un ou l'autre…

Autres environnements de programmation de dessins

D'autres environnements de programmation de dessins sont disponibles :

L'environnement `picture` : C'est un environnement natif de LaTeX qui permet de faire des dessins *très rudimentaires*. Les éléments du dessins sont extraits de fontes spéciales.
Des *packages* étendent ses fonctionnalités : **`epic`**, **`eepic`**, **`pict2e`**, **`curve2e`**…
⚠ Pour des raisons de compatibilité, c'est `picture` qui est utilisé par `Inkscape` et d'autres logiciels pour superposer une image et ses libellés textuels…

Le *package* `PSTricks` : Programmation de dessin très riche et puissante au sein du document LaTeX, report des ordres graphiques sur le *driver*.
Défaut : compilation vers PostScript ➡ non compatible avec `pdflatex`.
➡ On lui préférera TikZ, qui est beaucoup plus riche et universel.

Le *package* `xy` : La syntaxe est très compliquée, nombreuses limitations.
➡ à réserver pour les diagrammes mathématiques où il reste efficace.

Le moteur MetaPost : C'est l'ancêtre de Asymptote.
Limitations : précision des calculs en virgule flottante, formats de sorties…
➡ On lui préférera Asymptote qui le remplace avantageusement.
Convertir du code MetaPost en Asymptote n'est pas un gros travail.

Logiciels externes au monde LaTeX : De nombreux langages de programmation proposent des modules de dessins programmés.
Par exemple Matplotlib permet de dessiner dans le langage Python.

Insertion de graphiques externes

Où l'on apprend l'art élémentaire du collage d'images au cœur de nos textes…

La commande `\includegraphics[-]{-}` de graphicx

C'est la commande la plus importante du *package* **graphicx**.
Cette commande insère une image dans le flot courant, en tant que boite.

L'argument obligatoire est le nom d'un fichier d'image, avec éventuellement le chemin vers un sous dossier, l'argument optionnel définit des options sous la forme d'une liste d'éléments du type `clé=`*valeur* :

```
\includegraphics[width=2cm]{image.png}
\includegraphics[angle=45,origin=c]{dossier/image.pdf}
```

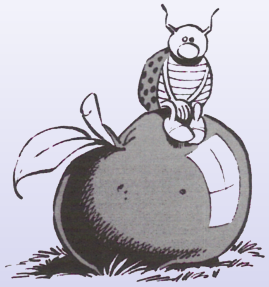
Les clés principales et les plus utiles sont le suivantes :

Clé	Description	Exemple
`width`	largeur de l'image	`width=1cm`
`height`	hauteur de l'image	`height=50pt`
`scale`	mise à l'échelle de l'image	`scale=2`
`angle`	angle de rotation	`angle=45`
`origin`	origine de la rotation	`origin=br`
`viewport`	définit la zone à afficher (`bp` par défaut)	`viewport=0 0 72 72`
`trim`	définit les marges à enlever	`trim=10 15 5 7`
`clip`	réduit la zone à afficher	`clip=true`

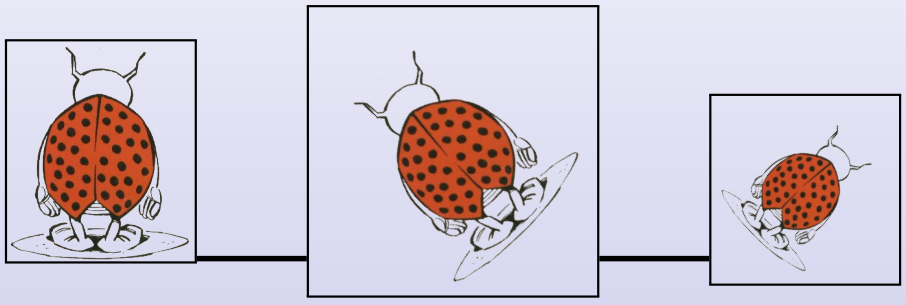
D'autres clés sont décrites dans la documentation du *package* **graphicx**.

La commande \includegraphics[-]{-} : exemples

```
\includegraphics[width=1.7cm]{pomme.png}
```



```
\includegraphics[width=1.2cm]{dos.png}
\includegraphics[width=1.2cm,origin=c,angle=45]{dos.png}
\includegraphics[origin=c,angle=-45,width=1.2cm]{dos.png}
```



(cadres et lignes ajoutés)

➟ L'ordre des options est important.

➟ Une figure sans rotation n'a pas de profondeur, une figure tournée peut en avoir une.

```
\includegraphics[viewport=4.6cm 1.9cm 5.1cm 2.4cm,
                 clip,
                 height=1cm]{CPT.pdf}
```

Autres commandes utiles du *package* graphicx

La commande `\graphicspath{-}` permet de spécifier une liste de dossiers où LaTeX peut rechercher des images :

```
\graphicspath{{../pdf/}{png/}}
```

La syntaxe est celle des chemins UNIX.
Rappelons que LaTeX cherche aussi les images dans les dossiers texmf.

La commande `\DeclareGraphicsExtensions{-}` permet de désigner une liste d'extensions de fichiers :

```
\DeclareGraphicsExtensions{pdf,png,jpg}
```

➟ On peut alors écrire `\includegraphics[width=1.2cm]{image}`
et LaTeX cherche dans l'ordre image.pdf, puis image.png et image.jpg.

`\includegraphics[angle=-40]{dahu}` ➟ 

Rotations avec le *package* graphicx

La commande `\rotatebox[-]{-}{-}` permet de tourner tout objet géré par LaTeX: graphiques, textes, tableaux…
Le premier argument obligatoire est l'angle (en degrés, sens trigonométrique), le second argument est l'objet à tourner.
L'argument optionnel définit des options sous la forme d'une liste de `clé=`*valeur* :

Clé	Description	Exemple
`origin`	origine de la rotation (`lrctbB`)	`origin=br`
`x`	abscisse du centre de rotation	`x=50pt`
`y`	ordonnée du centre de rotation	`y=5mm`
`units`	changement d'unité de rotation	`units=6.283185`

`l` = *left*, `r` = *right*, `c` = *center*, `t` = *top*, `b` = *bottom*, `B` = *Baseline*.
`units=6.283185` ($\simeq 2\pi$) permet de donner l'angle en radians (défaut : `units=360`).

"angle final en degrés, sens trigo." $= \frac{360}{\texttt{units}} \times$ "valeur donnée".

```
ABCD
\rotatebox[origin=c]{45}{ABCD}
\rotatebox[origin=tr]{90}{ABCD}
\rotatebox[x=0pt,y=15pt]{60}{ABCD}
\rotatebox[origin=c,units=-360]{45}{ABCD}
```

ABCD ABCD ABCD ABCD ABCD

Mises à l'échelle par un facteur

Les commandes suivantes du *package* **graphicx** permettent d'appliquer des mises à l'échelle sur tout objet géré par LaTeX : graphiques, textes, tableaux…

La commande `\scalebox{`*H*`}[`*V*`]{`*texte*`}` met à l'échelle *texte* : facteur *H* dans la direction horizontale, facteur (optionnel) *V* dans la direction verticale.
Si le facteur *V* n'est pas spécifié, la mise à l'échelle se fait avec *H* dans les deux directions.

```
                    ABCD   ABCD
\scalebox{2}[0.5]{ABCD}
\scalebox{2}{ABCD}
\scalebox{1}[-2]{ABCD}
```

`\reflectbox{`*texte*`}` équivaut à `\scalebox{-1}[1]{`*texte*`}`

```
                    ABCD   ABCD
\reflectbox{ABCD}          ↄꓭꓒA
```

➟ Utile pour insérer des textes de Léonard de Vinci.

Mises à l'échelle vers des dimensions données

La commande `\resizebox{`⟨*dim-h*⟩`}{`⟨*dim-v*⟩`}{`*texte*`}` du *package* **graphicx** met à l'échelle *texte* pour atteindre un boite à la dimension horizontale ⟨*dim-h*⟩ et à la dimension verticale ⟨*dim-v*⟩.

Si ⟨*dim-h*⟩ ou ⟨*dim-v*⟩ vaut `!`, l'autre dimension impose la mise à l'échelle en préservant les proportions de *texte*.

On peut utiliser `\height`, `\width`, `\totalheight`, `\depth` dans ces arguments pour faire référence aux dimensions naturelles de *texte*.

```
                                ABCD
\resizebox{1cm}{0.5cm}{ABCD}
\resizebox{2.5cm}{!}{ABCD}
\resizebox{!}{15pt}{ABCD}
\resizebox{\width}{15pt}{ABCD}
\resizebox{\width}{3\height}{ABCD}
```

ABCD

ABCD

ABCD

ABCD

ABCD

ABCD

Le *package* pdfpages

Et si on veut insérer une page entière ? Le *package* **pdfpages** répond à ce problème.

La commande `\includepdf[clé=valeur]{fichier.pdf}` insère certaines pages du fichier PDF *fichier.pdf* dans le document courant, *en tant que pages*.
`clé=valeur` sert à fournir les options :

- `pages={1,3,{},10-15}` insère les pages 1 et 3 du fichier, puis une page blanche, puis les pages 10 à 15. `pages={5-}` insère toutes les pages à partir de la 5. `pages={-}` insère tout le document.
- `nup=2x1` place 2 pages côte à côte sur une page finale.
  ⚠ Il vaut mieux alors utiliser l'option `landscape` avec le **geometry**.
- `frame` ajoute un cadre autour de chaque page insérée,
  `delta=5pt 10pt` ajoute des espaces horizontaux et verticaux entre les pages.
- D'autres options permettent de disposer les pages insérées de nombreuses façons : `openright`, `landscape`, `offset`, `column`, `signature`…

➞ Consulter la documentation du *package* **pdfpages**.

```
\usepackage[a4paper,margin=1cm,landscape]{geometry}
\usepackage{pdfpages}
...
\includepdf[pages={-}, nup=2x1, openright]{document.pdf}
```

➞ permet de préparer un document pour imprimer deux pages sur une page physique.

Insertion d'une figure avec légende

Où l'on découvre quelques commandes pour disposer au mieux nos images légendées dans le texte…

Rappels sur l'environnement `figure`

`\includegraphics[-]{-}` se contente d'insérer une image en tant que boite.
➟ l'image est placée à l'endroit où la commande est appelée, comme s'il s'agissait d'un simple (gros) caractère.
Une figure est souvent plus qu'une simple image : légende, numéro pour y faire référence, disposition centrée dans le texte… L'environnement `figure` de LaTeX sert à :

- définir une figure comme un flottant
  ➟ disposition dans le flux du texte à un endroit approprié ;
- insertion d'une légende avec numérotation automatique ;
- possibilité de faire référence à la figure grâce à un label.

```
\begin{figure}[ht]
  \centering
  \includegraphics[width=0.5\textwidth]{image.png}
  \caption{Une légende pour décrire la figure}
  \label{fig-image}
\end{figure}
```

La commande `\label{-}` doit être placée après la commande `\caption{-}` puisqu'elle fait référence au numéro de la figure créé par cette dernière.
Voir le cours sur les flottants pour gérer cet aspect de l'environnement `figure`.
⚠ N'utiliser `figure` que si on a besoin d'une légende et/ou d'un numéro…

Personnaliser l'environnement `figure`

Quelques dimensions et commandes sont accessibles pour personnaliser l'aspect d'une figure, et plus généralement d'un flottant :

- Les longueurs `\abovecaptionskip` et `\belowcaptionskip` gèrent les espaces verticaux avant et après la légende.
  ➟ Utile lorsqu'on place la légende au-dessus de l'image.
- La commande `\topfigrule` est exécutée après la dernière figure *top* d'une page pour séparer les figures du texte.
  La commande `\botfigrule` fait de même avant la première figure *bottom*.
  ⚠ Par défaut, ces commandes sont "vides" : elles se modifient (la première fois) avec `\newcommand{-}{-}` et la hauteur finale doit être nulle.

```
\newcommand{\topfigrule}{%
   \vspace*{10pt}\hrule\vspace{-10.4pt}}
```

- `\floatsep` est l'espacement vertical entre les flottants *top* ou *bottom*.
  `\intextsep` est l'espacement vertical avant et après un flottant de type `h`.
  `\textfloatsep` est l'espacement vertical entre les flottants et le texte.

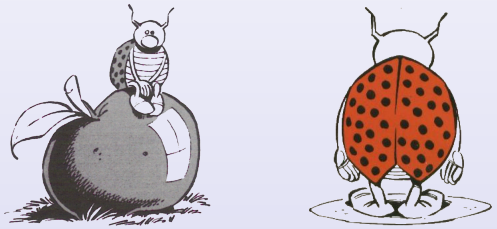
Le document **Using imported Graphics in LaTeX and pdfLaTeX** donne de nombreux trucs et astuces concernant le positionnement de graphiques dans le texte.
⚠ Ce texte comporte quelques erreurs, il est donc utile de tester ce qu'il énonce.

Le *package* subfig

Le *package* **subfig** place dans un même flottant plusieurs "sous flottants" avec leur sous numérotations et des sous légendes.

➙ fonctionne aussi avec l'environnement `table`.

```
\begin{figure}[t]
\centering
 \subfloat[Assis]{
  \includegraphics[height=3cm]{pomme.png}
  \label{fig-pomme}}
 \subfloat[De dos]{
  \includegraphics[height=3cm]{dos.png}
  \label{fig-dos}}
\caption{Deux images}\label{fig-double}
\end{figure}
```



(a) Assis (b) De dos

FIGURE 1 – Deux images

Ce *package* permet de lister les sous flottants, de modifier l'aspect et le positionnement des légendes…

⚠ **Conflit de noms :** La commande `\subfloat` n'a rien à voir avec le *package* **subfloat** qui définit les environnement `subfigures` (!!) et `subtables` (dans le même esprit que `subequations`) pour modifier les numérotations des figures et tableaux.

Le *package* wrapfig

Le *package* **wrapfig** permet d'insérer une figure dans un paragraphe.

```
\begin{wrapfigure}{r}{4cm}
\includegraphics[width=4cm]{pomme.png}
\caption{Sur une pomme}
\end{wrapfigure}
Lorem ipsum dolor sit amet, ...
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque ut consequat magna. Cras commodo enim nec metus gravida quis porttitor erat volutpat. Maecenas ac augue in neque vehicula tristique nec vitae orci. Quisque accumsan laoreet rhoncus. Morbi elit ipsum, feugiat id fringilla a, auctor a urna. Etiam quis mauris sed ipsum dignissim volutpat. Integer in ligula ligula, vel auctor lacus. Maecenas in lorem ligula. Vivamus a arcu neque, in porta velit. In euismod arcu nec lectus cursus sit amet dignissim odio gravida. Nullam commodo eleifend tellus, id sodales mauris adipiscing eget. Duis at blandit lacus. Pellentesque magna dolor, eleifend et porttitor eu, facilisis non odio. Nam et urna urna, vel elementum ipsum. Fusce eu dolor massa, at blandit leo. Suspendisse dignissim luctus ipsum, ut volutpat velit tristique at. Pellentesque habitant morbi tristique senectus et

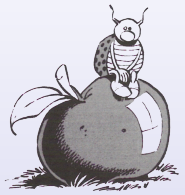


Figure 1 – Sur une pomme

L'insertion peut se faire à droite, à gauche, sur le côté intérieur ou le côté extérieur dans le mode recto-verso (`twoside`).
L'image peut déborder dans la marge, l'espacement entre le texte et l'image est modifiable.

Ce *package* est déjà ancien, mais il fonctionne toujours aussi bien.
Par contre, sa documentation n'est pas fournie en PDF.
`texdoc wrapfig` affiche le fichier wrapfig.sty…

Les *packages* `caption` et `float`

Il est possible de personnaliser les légendes des flottants avec le *package* **`caption`**.

- Une commande `\captionsetup[-]{-}` permet de modifier de façon globale l'aspect des légendes d'un type particulier de flottants.
- Il est possible de modifier le format général de la légende, la typographique des éléments d'une légende, les espacements verticaux et horizontaux…
- Des styles peuvent être définies et chargés.

Il est possible de définir de nouveaux types de flottants avec le *package* **`float`**.

- Une commande `\newfloat` permet de définir de nouveaux types de flottants avec des styles prédéfinis.
  ➟ commande dans le même esprit que `\newtheorem`.
- Une liste des flottants ainsi définis est accessible.
- Ce *package* a déjà été mentionné car il définit l'option de placement `[H]`.

Consulter la documentation de ces *packages* pour plus de détails et d'exemples.

Ces *packages* sont compatibles avec les *packages* présentés auparavant.