# Une introduction à METAPOST

## ou comment créer *simplement*[1] de belles figures.

Christophe Poulain
christophe.poulain@melusine.eu.org
Collège Paul Eluard

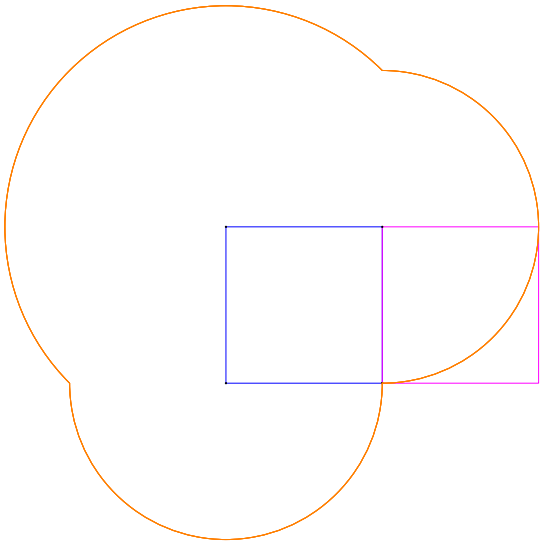
1. Bon, d'accord, c'est un peu subjectif !

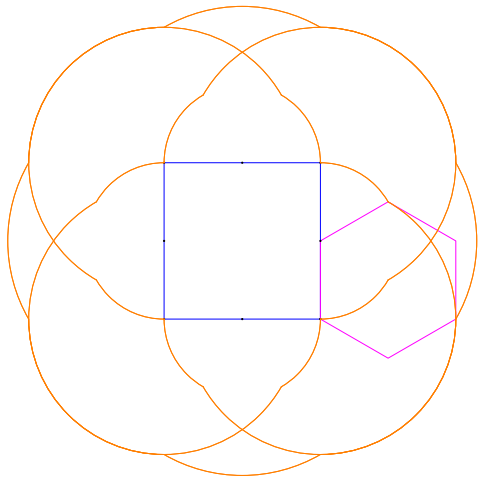
# Plan

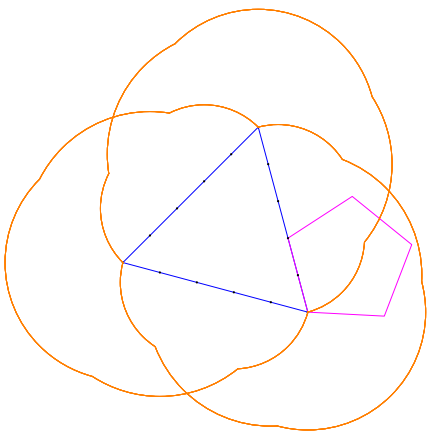
BEUVRAGES

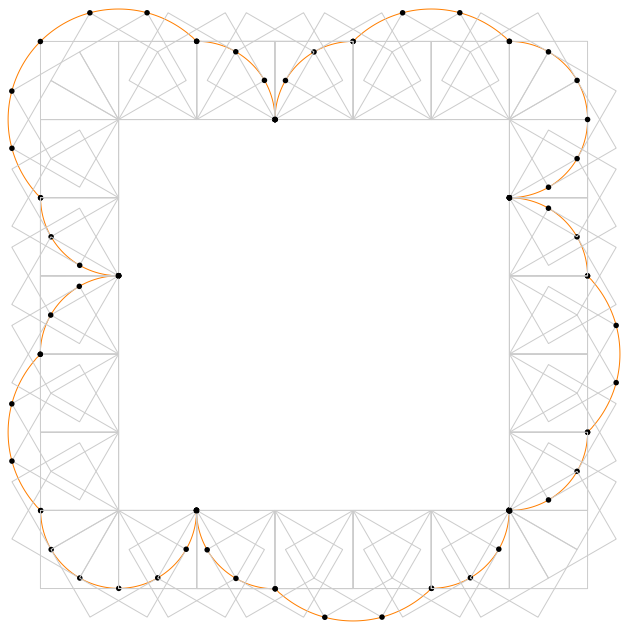
METAPOST

BEUVRAGES

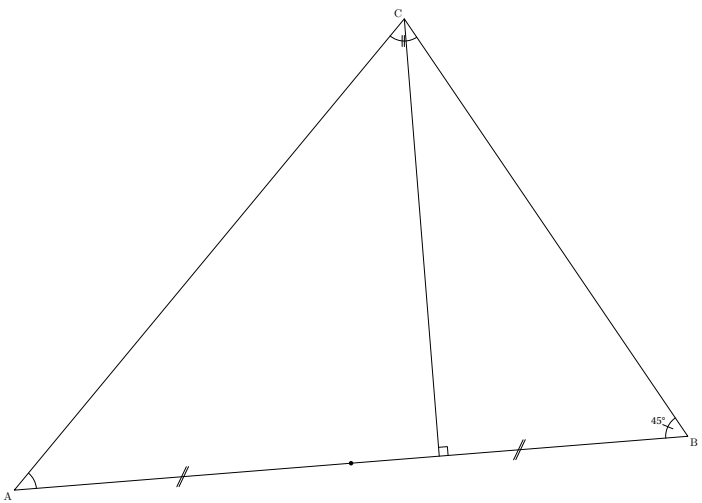


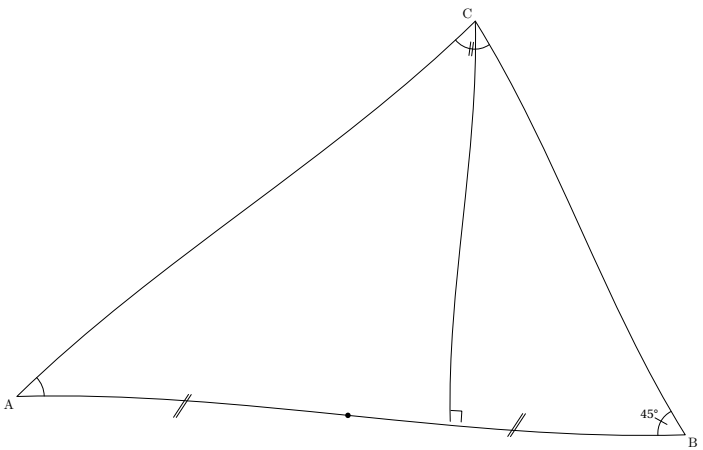




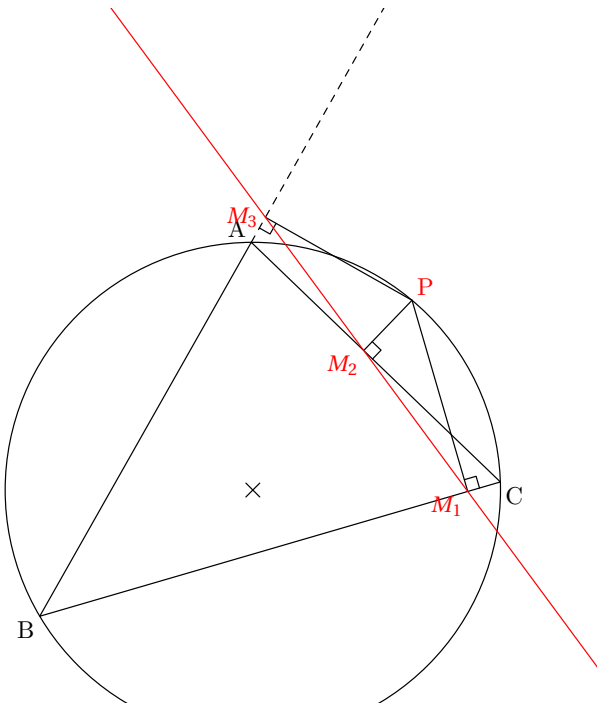


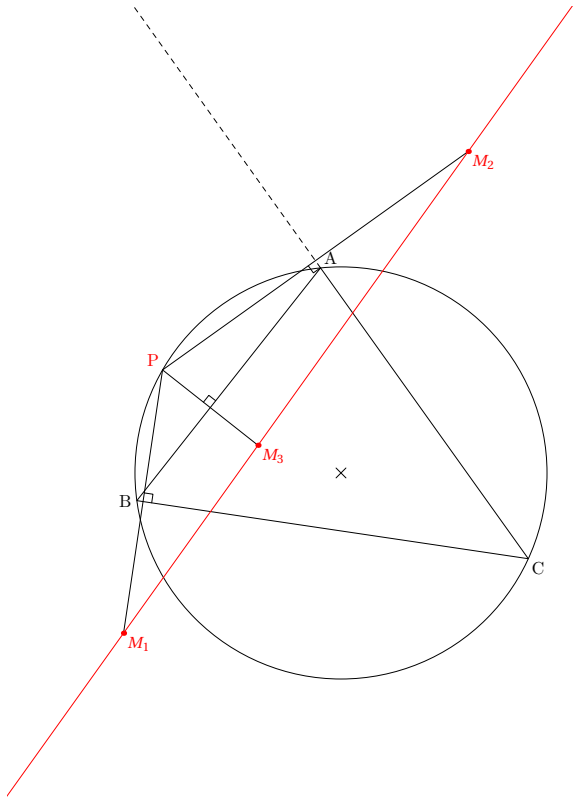
BEUVRAGES



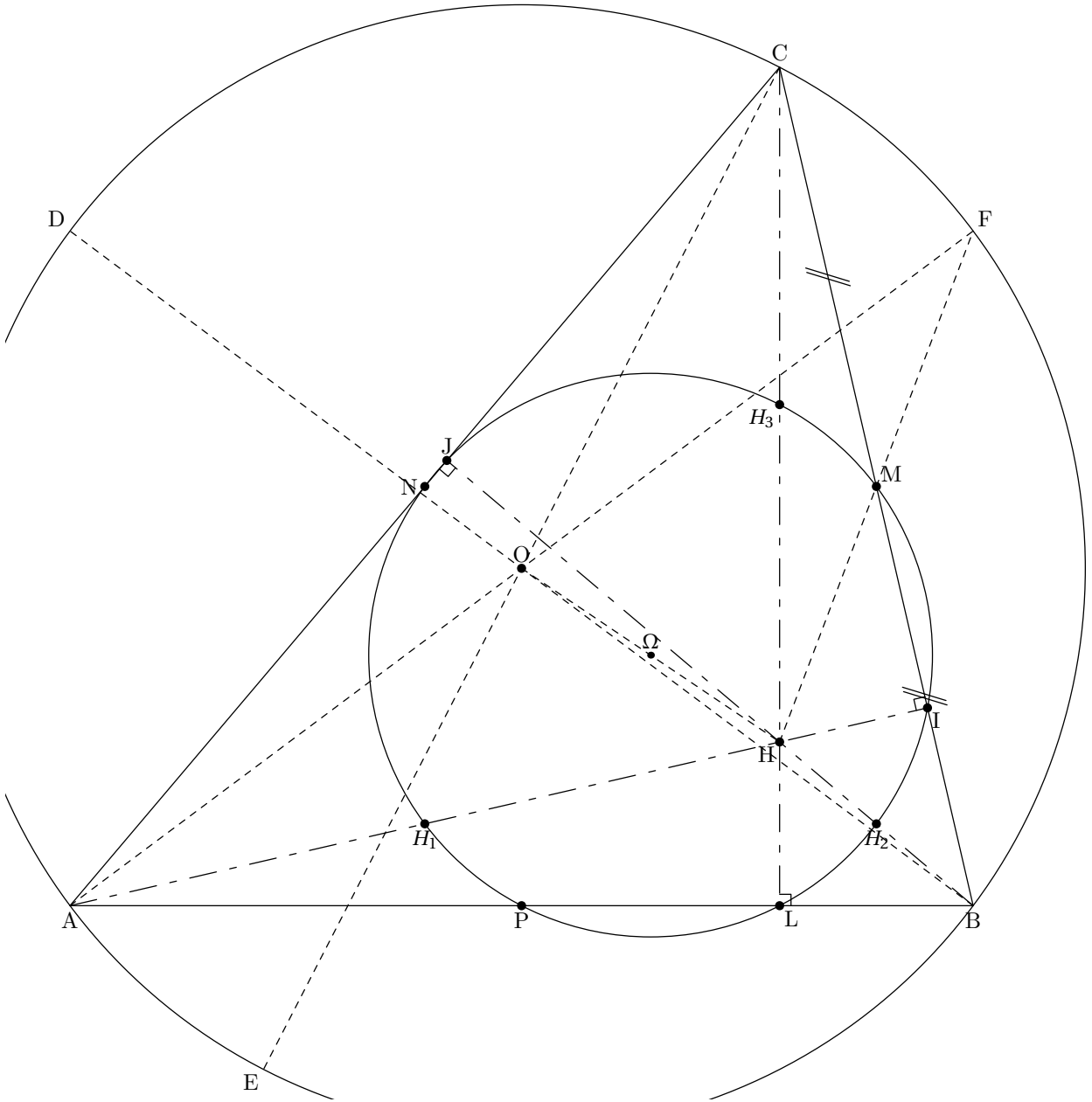



BEUVRAGES

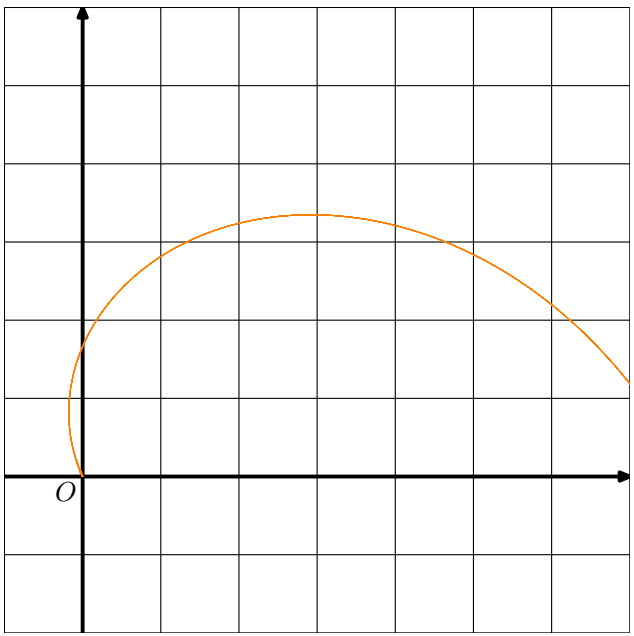


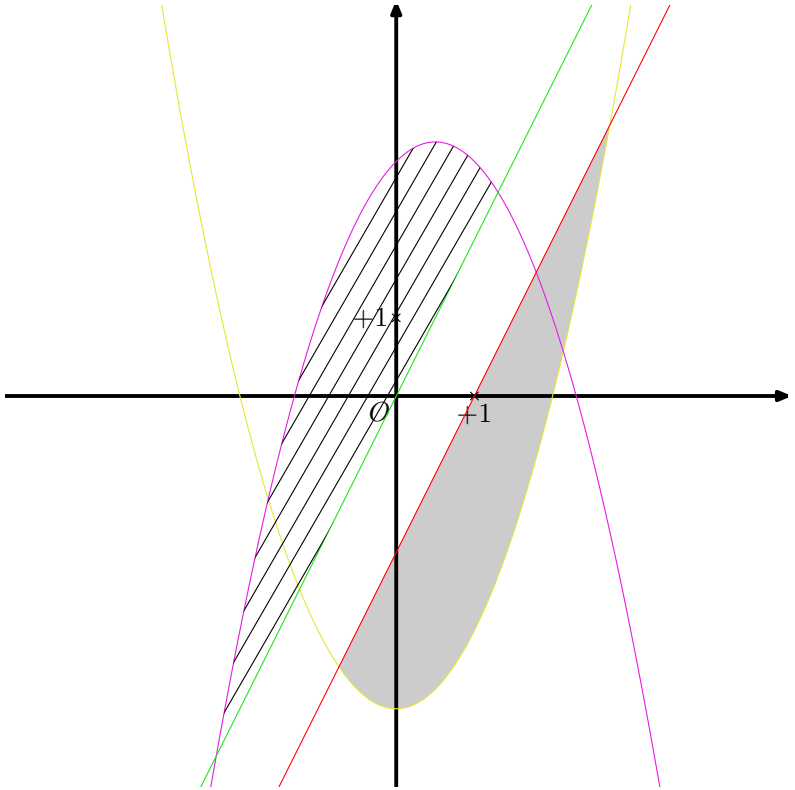



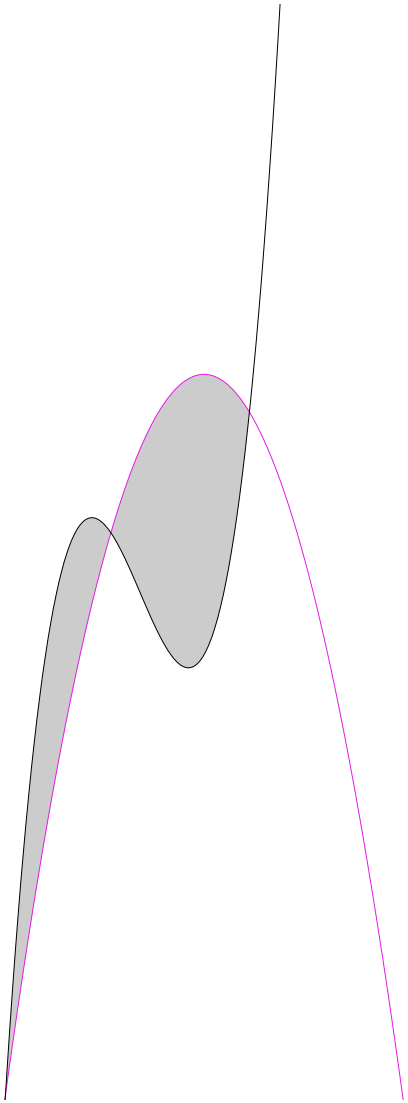

METAPOST

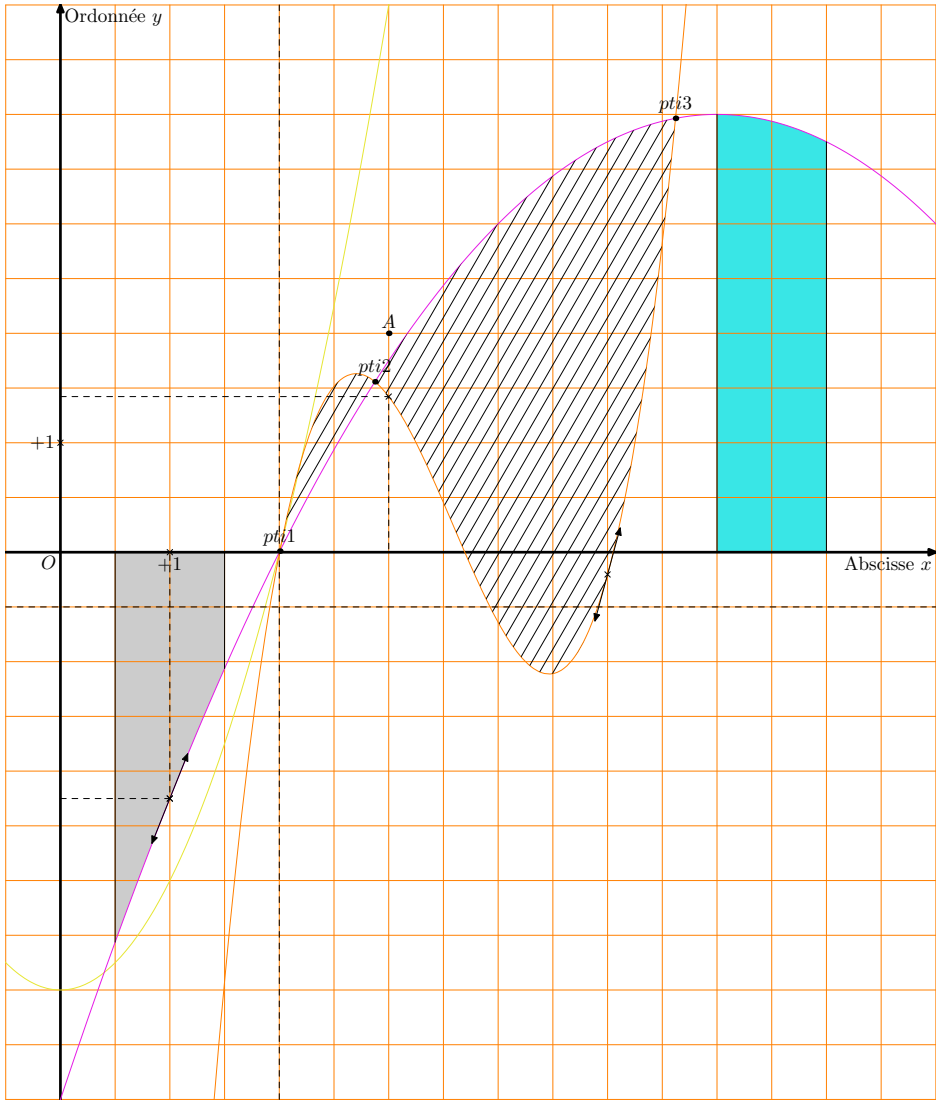


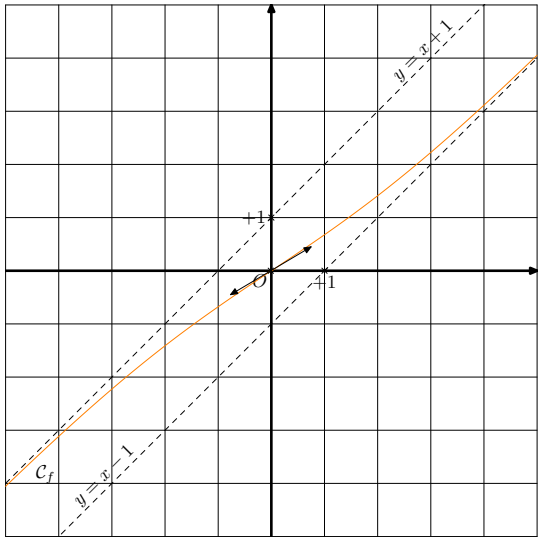

METAPOST

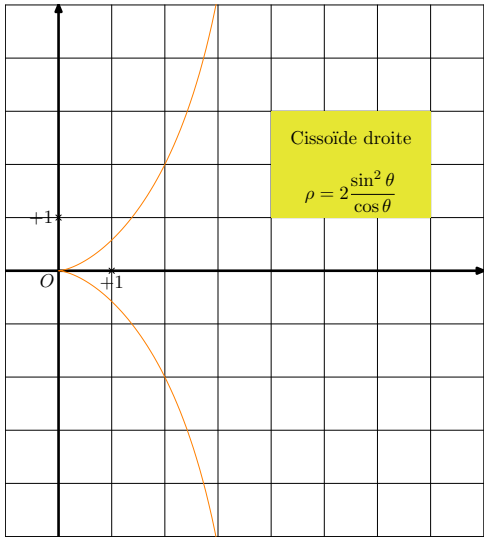

O

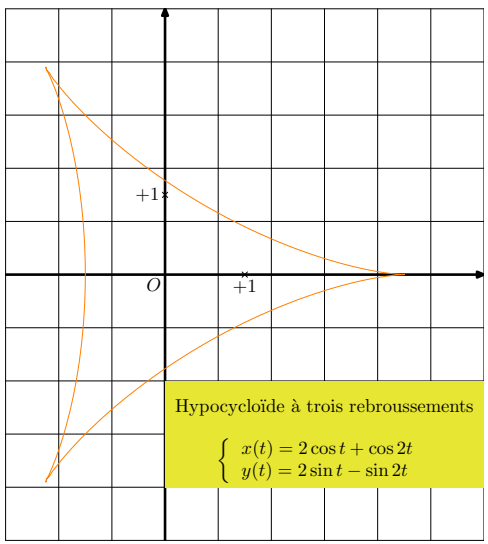

+1
O
+1

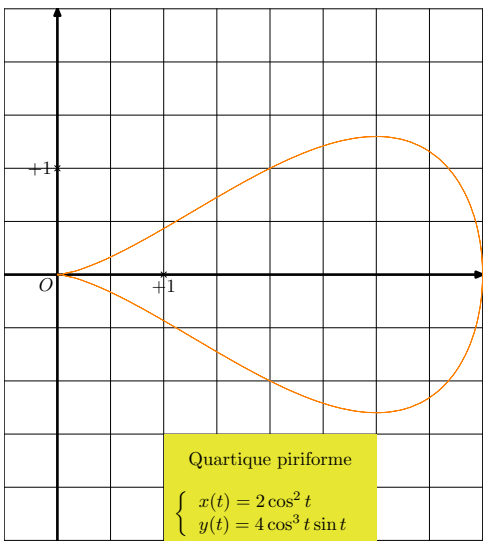


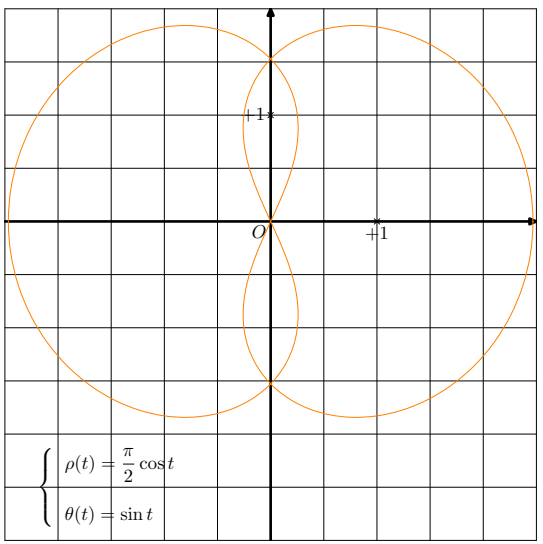

Ordonnée $y$
$pti3$
$A$
$pti2$
$+1$
$pti1$
$O$
$+1$
Abscisse $x$

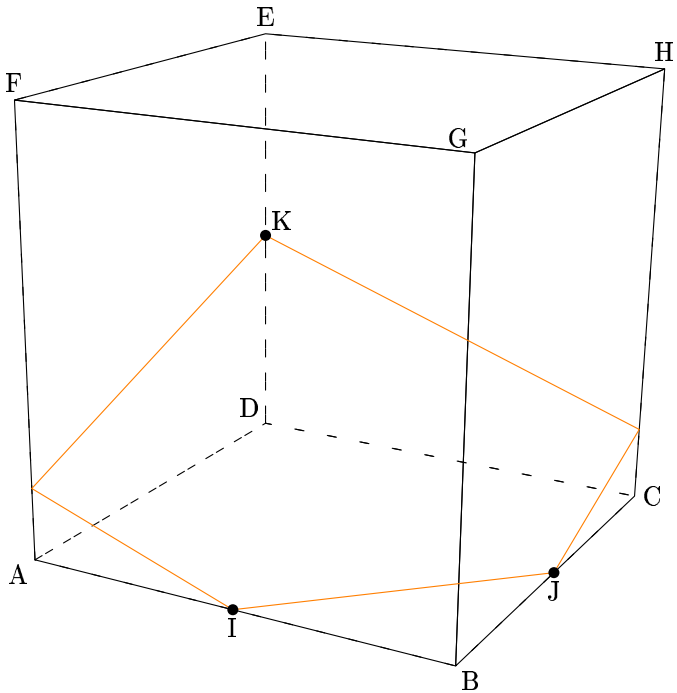

$y = x + 1$
$y = x - 1$
$\mathcal{C}_f$
$O$
$+1$
$+1$

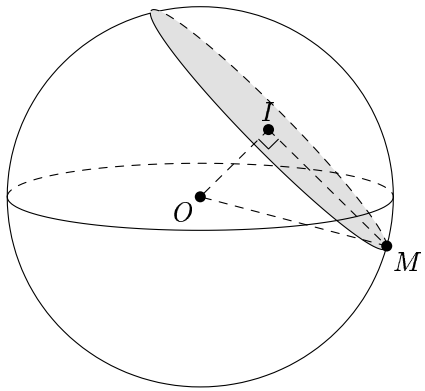

Cissoïde droite
$\rho = 2\dfrac{\sin^2\theta}{\cos\theta}$
$O$
$+1$
$+1$

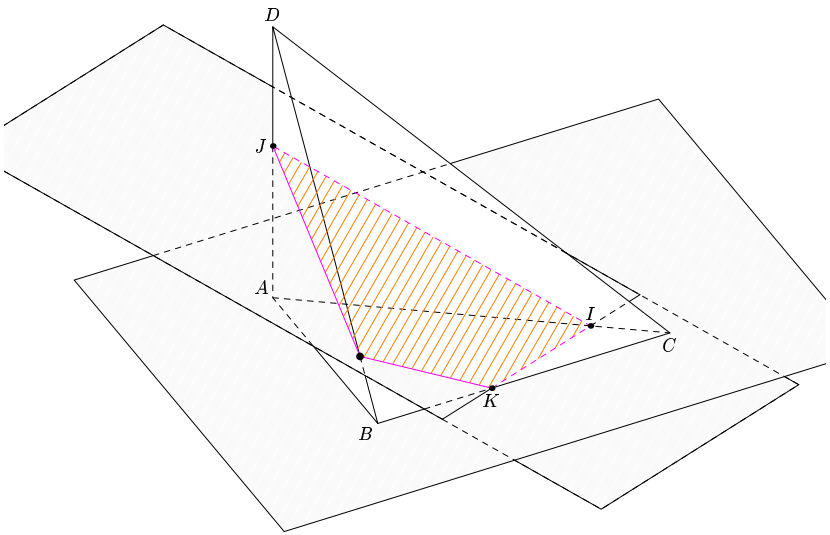

Hypocycloïde à trois rebroussements
$\begin{cases} x(t) = 2\cos t + \cos 2t \\ y(t) = 2\sin t - \sin 2t \end{cases}$
$O$
$+1$
$+1$

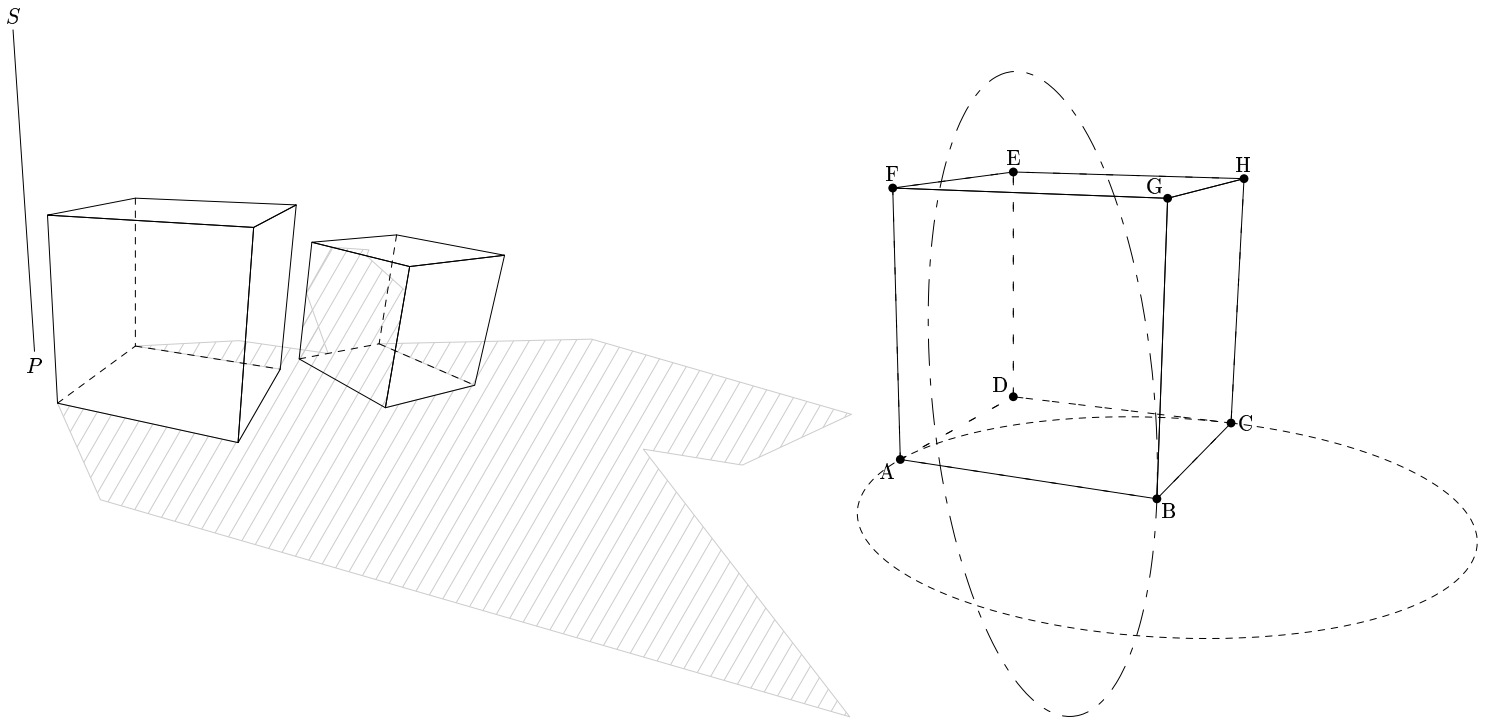

Quartique piriforme
$\begin{cases} x(t) = 2\cos^2 t \\ y(t) = 4\cos^3 t \sin t \end{cases}$
$O$
$+1$
$+1$

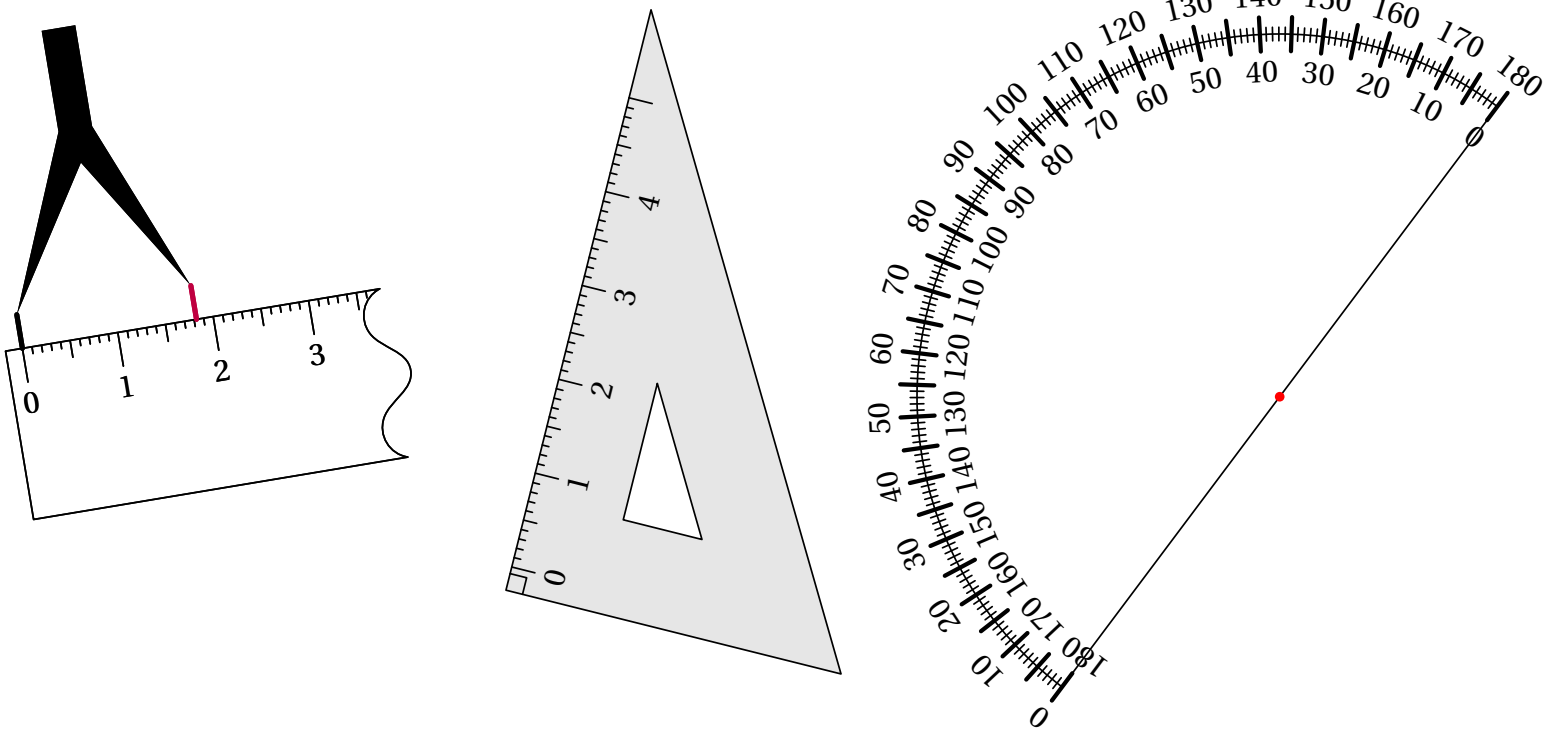

$\begin{cases} \rho(t) = \dfrac{\pi}{2}\cos t \\ \theta(t) = \sin t \end{cases}$
$O$
$+1$
$+1$

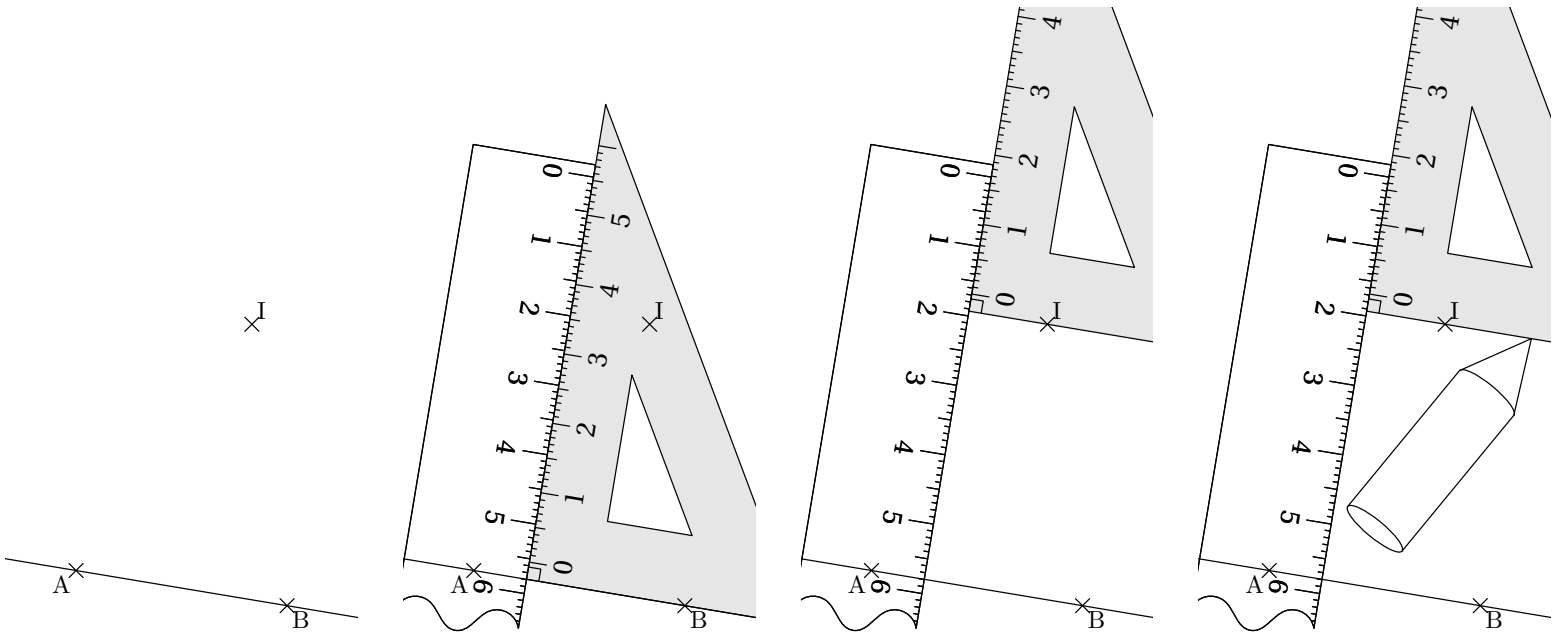

E
H
F
G
K
D
C
A
J
I
B


I
O
M

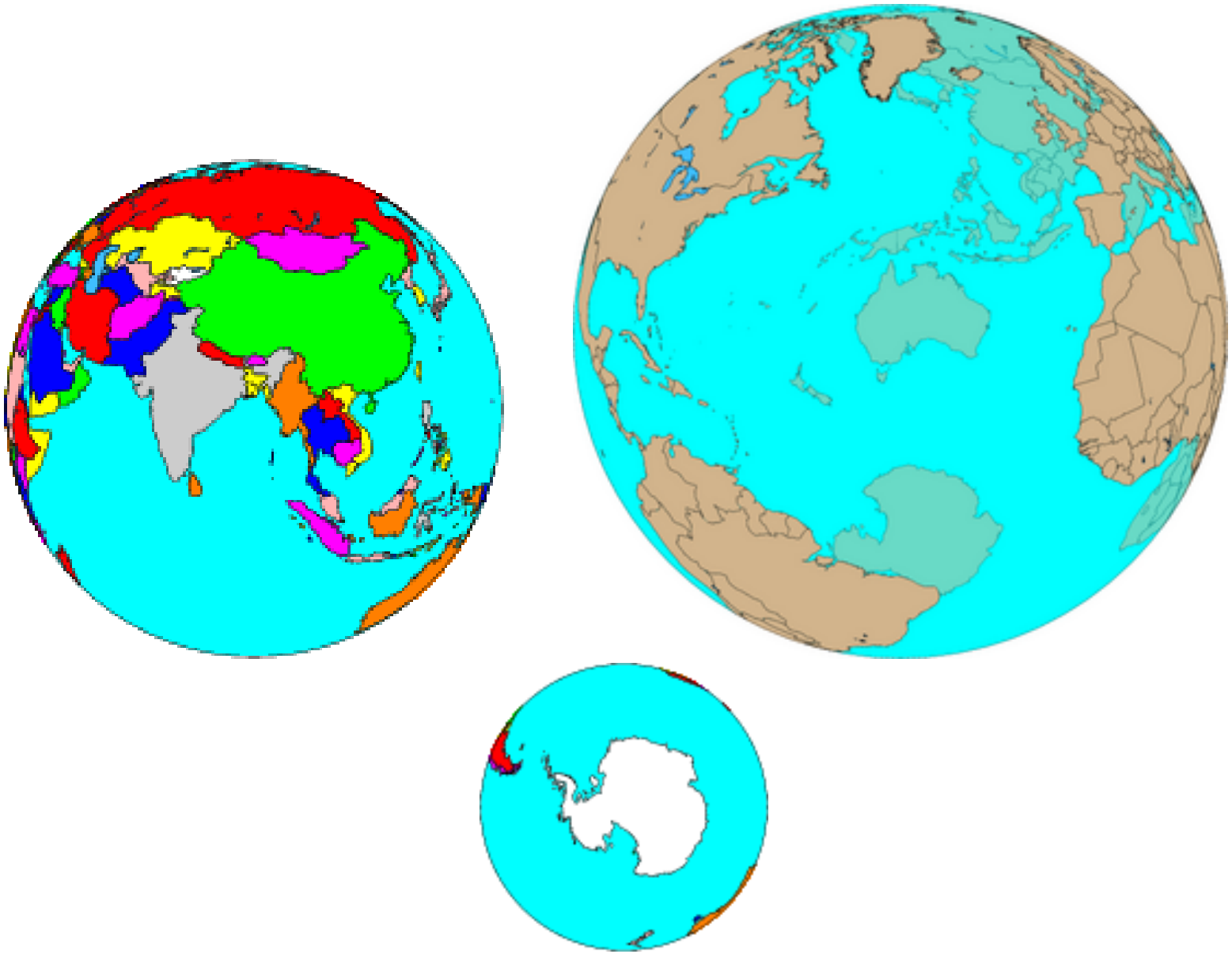

D
J
A
I
C
K
B


S
P
F
E
H
G
D
C
A
B


0
1
2
3
0
1
2
3
4
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180
180 170 160 150 140 130 120 110 100 90 80 70 60 50 40 30 20 10 0

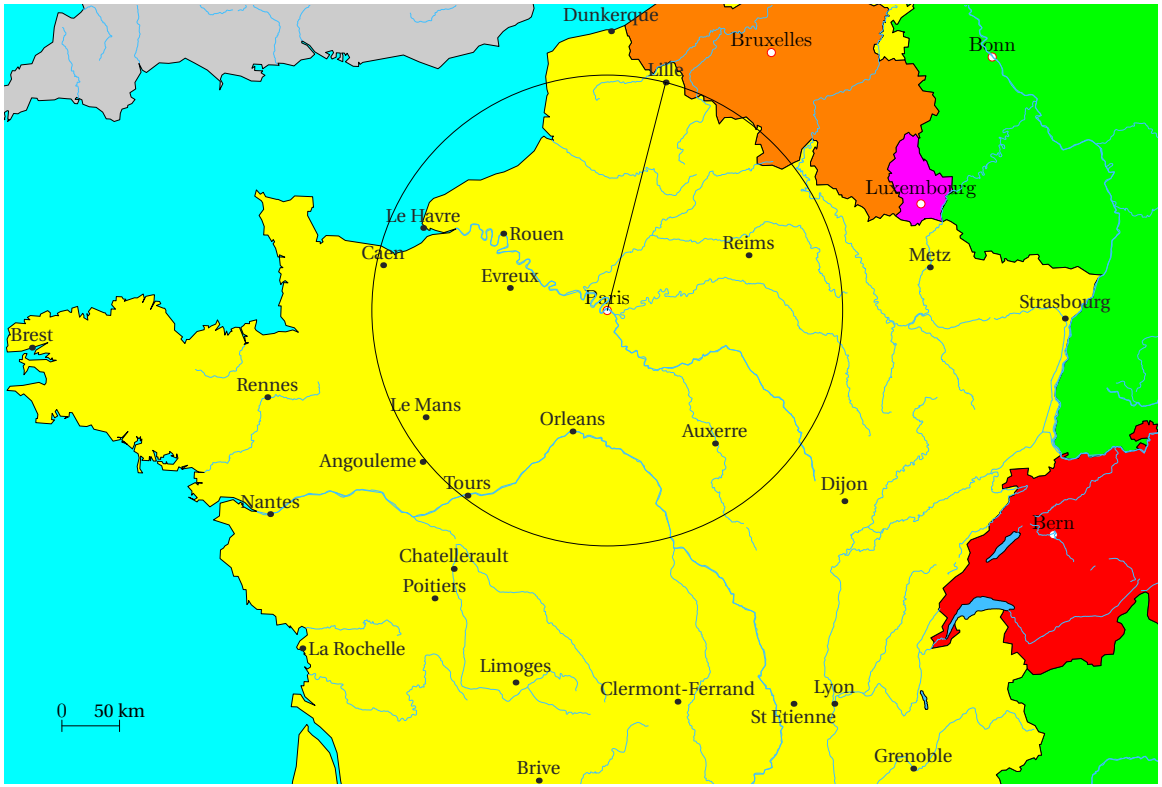

I
A
B
I
A
B
I
A
B
I
A
B

BEUVRAGES

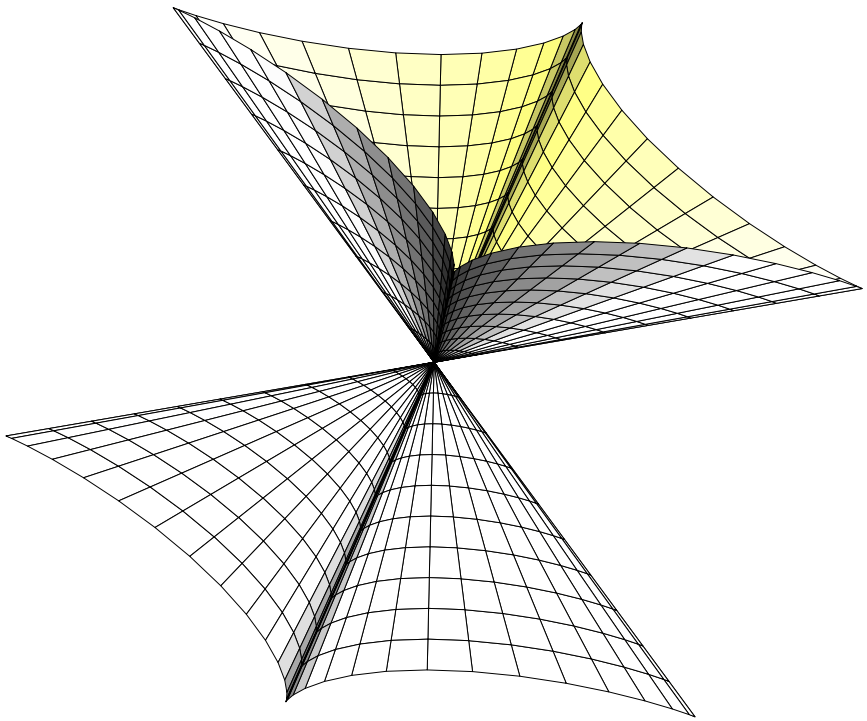


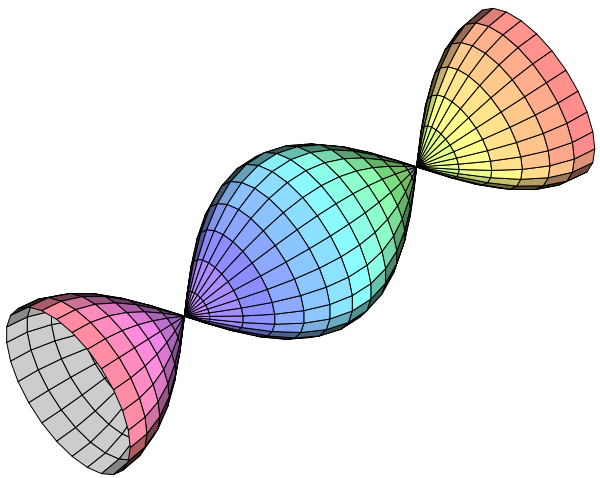
METAPOST

BEUVRAGES


Dunkerque
Bruxelles
Bonn
Lille
Luxembourg
Le Havre
Rouen
Caen
Reims
Metz
Evreux
Paris
Strasbourg
Brest
Rennes
Le Mans
Orleans
Auxerre
Angouleme
Tours
Dijon
Nantes
Bern
Chatellerault
Poitiers
La Rochelle
Limoges
Clermont-Ferrand
Lyon
St Etienne
Grenoble
Brive
0 50 km

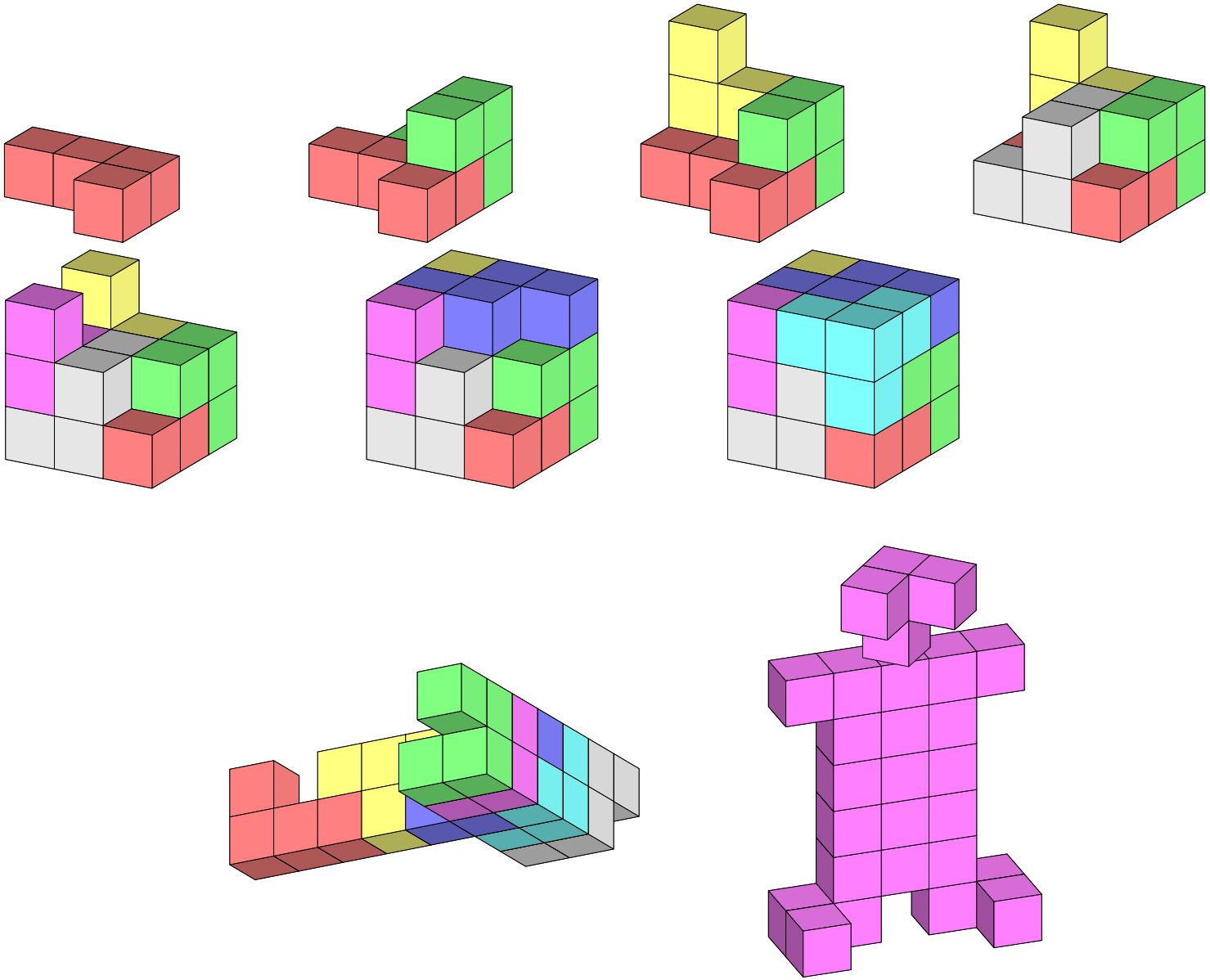
BEUVRAGES



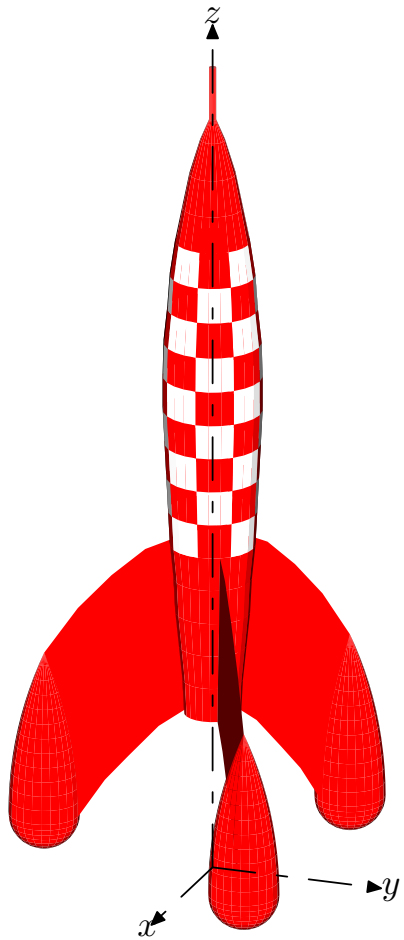


METAPOST

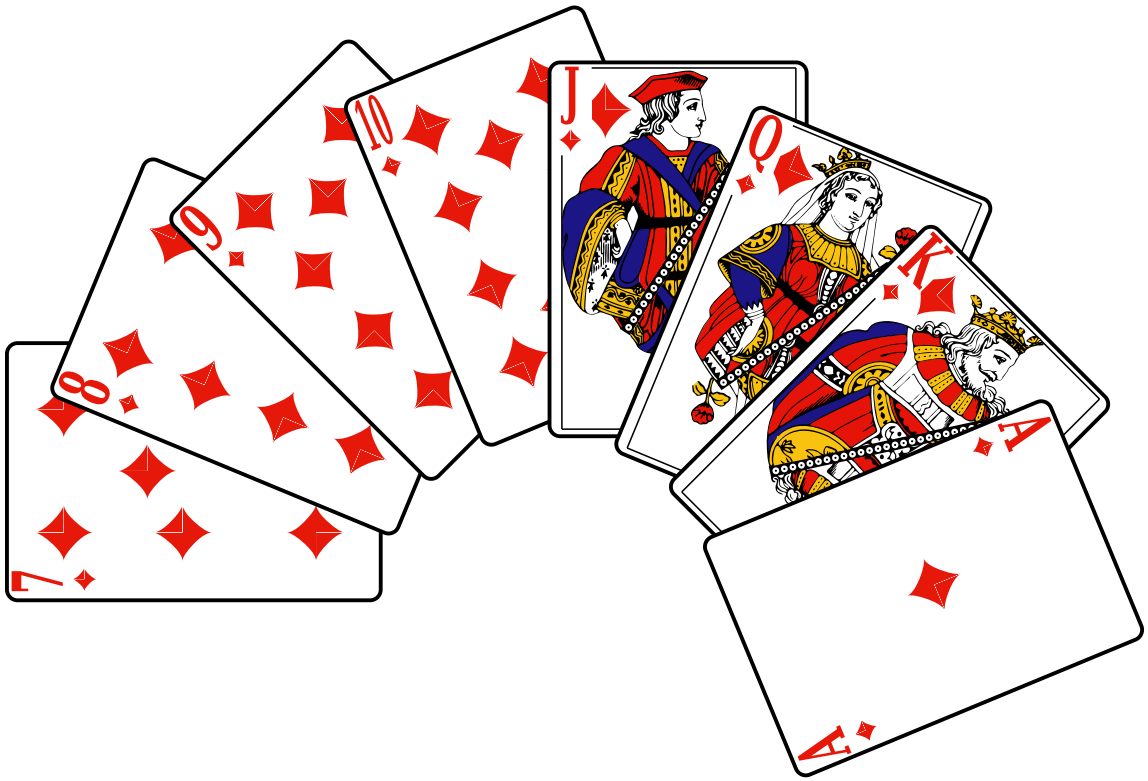
BEUVRAGES



BEUVRAGES



BEUVRAGES



Magnifique journée ! L'organisation est super, les intervenants sont super, le public est super ! Vive Denis !

Magnifique journée ! L'organisation est super, les intervenants sont super, le public est super ! Vive Denis !

METAPOST

# Plan

**fin des années 1990** Première utilisation de TeX avec une distribution WinGut.

**fin des années 1990** Première utilisation de TeX avec une distribution WinGut.

**2001** Participation à l'atelier LaTeX de Daniel Flipo aux journées APMEP de Lille.

**fin des années 1990** Première utilisation de TeX avec une distribution WinGut.

**2001** Participation à l'atelier LaTeX de Daniel Flipo aux journées APMEP de Lille.

**fin 2001** Révélation ! Lecture de « the not so short. . . » traduite par Daniel Flipo : METAPOST ? Quesako ?

- Crée par John D. Hobby

2.

- Crée par John D. Hobby
- Basé sur METAFONT[2], outil de création de fontes.

2. de Knuth, l'auteur de TEX

- Crée par John D. Hobby
- Basé sur METAFONT[2], outil de création de fontes.
- Produit des fichiers postscript.

2. de Knuth, l'auteur de TEX

- Crée par John D. Hobby
- Basé sur METAFONT[2], outil de création de fontes.
- Produit des fichiers postscript.
- Utilise le principe de description d'image.

2. de Knuth, l'auteur de TEX

# **Avantages** de METAPOST.

## **Avantages** de METAPOST.

- Intégration parfaite à LaTeX ;

## **Avantages** de METAPOST.

- Intégration parfaite à LaTeX ;
- facile à programmer donc paramétrable et personnalisable ;

## **Avantages** de METAPOST.

- Intégration parfaite à LaTeX ;
- facile à programmer donc paramétrable et personnalisable ;
- fonctionnement proche de LaTeX ;

## **Avantages** de METAPOST.

- Intégration parfaite à LaTeX ;
- facile à programmer donc paramétrable et personnalisable ;
- fonctionnement proche de LaTeX ;
- plusieurs figures en un seul fichier ;

## **Avantages** de METAPOST.

- Intégration parfaite à LaTeX ;
- facile à programmer donc paramétrable et personnalisable ;
- fonctionnement proche de LaTeX ;
- plusieurs figures en un seul fichier ;
- résous des équations linéaires.

**Avantages** de METAPOST.

- Intégration parfaite à LaTeX ;
- facile à programmer donc paramétrable et personnalisable ;
- fonctionnement proche de LaTeX ;
- plusieurs figures en un seul fichier ;
- résous des équations linéaires.

**Inconvénients** de METAPOST.

## **Avantages** de METAPOST.

- Intégration parfaite à LaTeX ;
- facile à programmer donc paramétrable et personnalisable ;
- fonctionnement proche de LaTeX ;
- plusieurs figures en un seul fichier ;
- résous des équations linéaires.

## **Inconvénients** de METAPOST.

- utilisation d'un programme « externe » à LaTeX ;

## **Avantages** de METAPOST.

- Intégration parfaite à LaTeX ;
- facile à programmer donc paramétrable et personnalisable ;
- fonctionnement proche de LaTeX ;
- plusieurs figures en un seul fichier ;
- résous des équations linéaires.

## **Inconvénients** de METAPOST.

- utilisation d'un programme « externe » à LaTeX ;
- une compilation supplémentaire ;

## **Avantages** de METAPOST.

- Intégration parfaite à LaTeX ;
- facile à programmer donc paramétrable et personnalisable ;
- fonctionnement proche de LaTeX ;
- plusieurs figures en un seul fichier ;
- résous des équations linéaires.

## **Inconvénients** de METAPOST.

- utilisation d'un programme « externe » à LaTeX ;
- une compilation supplémentaire ;
- encore de la programmation ;

## **Avantages** de METAPOST.

- Intégration parfaite à LaTeX ;
- facile à programmer donc paramétrable et personnalisable ;
- fonctionnement proche de LaTeX ;
- plusieurs figures en un seul fichier ;
- résous des équations linéaires.

## **Inconvénients** de METAPOST.

- utilisation d'un programme « externe » à LaTeX ;
- une compilation supplémentaire ;
- encore de la programmation ;
- . . . euh. . .

# Principe général

Comme LaTeX, c'est-à-dire. . .

# Principe général

Comme LaTeX, c'est-à-dire...

- en créant un fichier source sauvegardé sous le nom *nomfichier.mp* et qui aura la forme

# Principe général

Comme LaTeX, c'est-à-dire...

- en créant un fichier source sauvegardé sous le nom *nomfichier.mp* et qui aura la forme

```
beginfig(1);
...
endfig;

beginfig(2);
...
endfig;
end;
```

# Principe général

Comme LaTeX, c'est-à-dire...

- en créant un fichier source sauvegardé sous le nom *nomfichier.mp* et qui aura la forme

```
beginfig(1);
...
endfig;

beginfig(2);
...
endfig;
end;
```

- en compilant ce fichier. (Attention aux noms de commandes).

- Obtention de fichiers postscript appelés *nomfichier.1*, *nomfichier.2*,. . . ;

- Obtention de fichiers postscript appelés *nomfichier.1*, *nomfichier.2*,. . . ;
- intégrés à des documents LaTeX grâce au package `graphicx`

- Obtention de fichiers postscript appelés *nomfichier.1*, *nomfichier.2*,. . . ;
- intégrés à des documents LaTeX grâce au package `graphicx` et à la commande

```
\includegraphics{nomfichier.1}
```

- Obtention de fichiers postscript appelés *nomfichier.1*, *nomfichier.2*,. . . ;
- intégrés à des documents LaTeX grâce au package `graphicx` et à la commande

```
\includegraphics{nomfichier.1}
```

Attention, dans

- Obtention de fichiers postscript appelés *nomfichier.1*, *nomfichier.2*,. . . ;
- intégrés à des documents LaTeX grâce au package `graphicx` et à la commande

```
\includegraphics{nomfichier.1}
```

Attention, dans

METAPOST+LaTeX la commande précédente est utilisée telle que ;

- Obtention de fichiers postscript appelés *nomfichier.1*, *nomfichier.2*,. . . ;
- intégrés à des documents LaTeX grâce au package `graphicx` et à la commande

```
\includegraphics{nomfichier.1}
```

Attention, dans

METAPOST+LaTeX la commande précédente est utilisée telle que ;

METAPOST+PDFLaTeX il y a un bémol : il faut ajouter, *dans le préambule*, la commande

```
\DeclareGraphicsRule{*}{mps}{*}{}
```

# Plan

# Segments, chemins, polygones

# Segments, chemins, polygones

## Le code

```
1 draw (20,20)--(50,30);
```

produira...

# Segments, chemins, polygones

## Le code

```
1 draw (20,20)--(50,30)--(90,100);
```

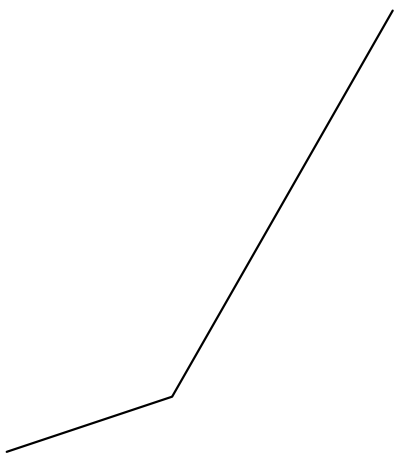
produira...

# Segments, chemins, polygones

## Le code

```
1 draw (20,20)--(50,30)--(90,100);
```

produira...



METAPOST

# Segments, chemins, polygones

## Le code

```
1 draw (20,20)--(50,30)--(90,100)--cycle;
```

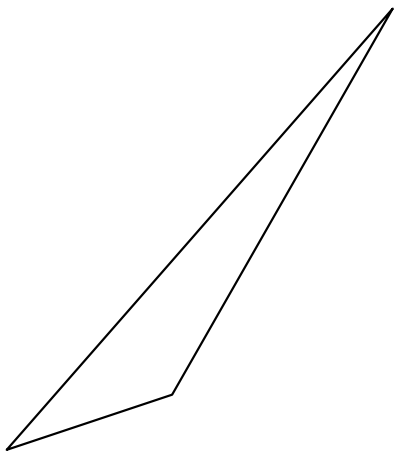
produira...

# Segments, chemins, polygones

## Le code

```
1 draw (20,20)--(50,30)--(90,100)--cycle;
```

produira...



# Segments, chemins, polygones

Attention, le code

```
1 draw (20,20)..(50,30)..(90,100);
```

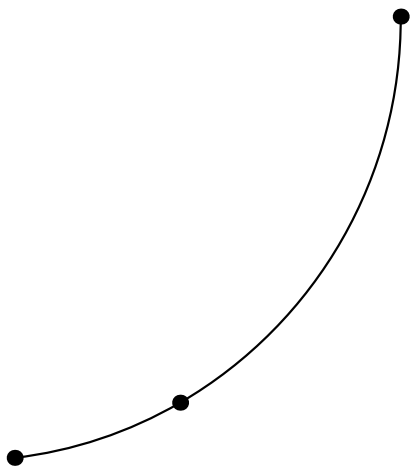
produira...

# Segments, chemins, polygones

## Attention, le code

```
1 draw (20,20)..(50,30)..(90,100);
```

produira...



# Segments, chemins, polygones

et le code

```
1 draw (20,20)..(50,30)..(90,100).. cycle;
```

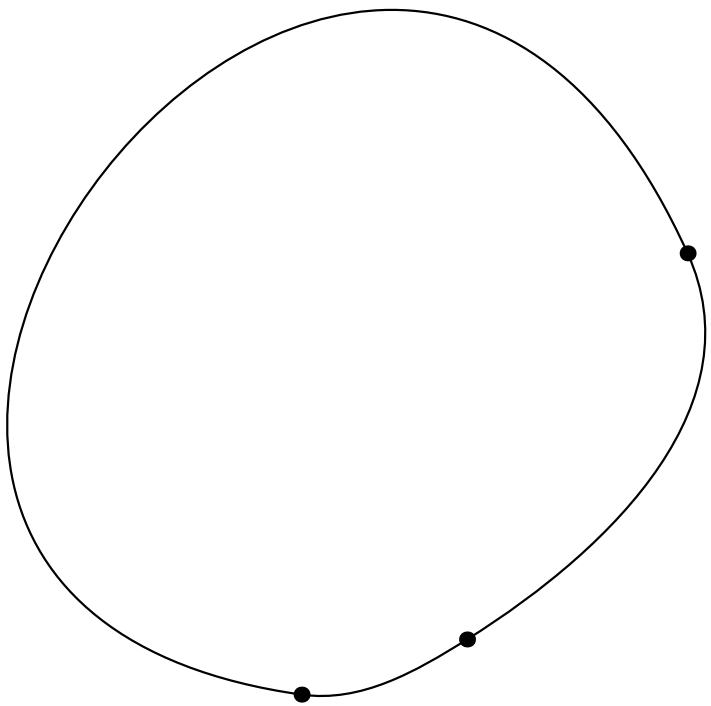
produira...

# Segments, chemins, polygones

et le code

```
1 draw (20,20)..(50,30)..(90,100).. cycle;
```

produira...



# Segments, chemins, polygones

Pour les flèches,

```
1 drawarrow (20,20)--(50,30)--(90,100);
```

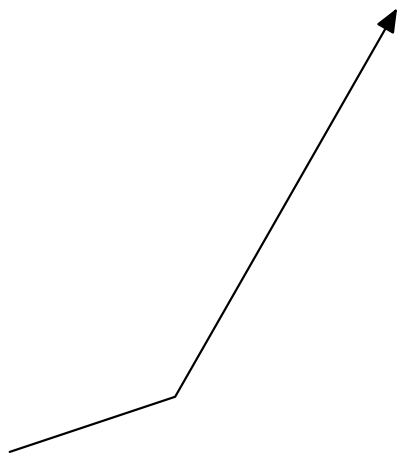
produira...

# Segments, chemins, polygones

Pour les flèches,

```
1 drawarrow (20,20)--(50,30)--(90,100);
```

produira...



METAPOST

# Segments, chemins, polygones

ou encore

```
1 drawdblarrow (20,20)..(50,30)..(90,100);
```

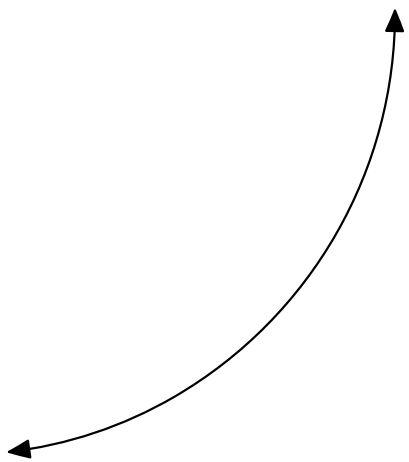
produira...

# Segments, chemins, polygones

ou encore

```
1 drawdblarrow (20,20)..(50,30)..(90,100);
```

produira...



Pour faciliter la lecture et/ou la suite de la construction, on peut remplacer

```
1 draw (20,20)--(50,30)--(90,100)--cycle;
```

par

Pour faciliter la lecture et/ou la suite de la construction, on peut remplacer

```
1 draw (20,20)--(50,30)--(90,100)--cycle;
```

par

```
1 pair A,B,C;
```

Pour faciliter la lecture et/ou la suite de la construction, on peut remplacer

```
1 draw (20,20)--(50,30)--(90,100)--cycle;
```

par

```
1 pair A,B,C;
  A=(20,20);
  B=(50,30);
  C=(90,100);
```

Pour faciliter la lecture et/ou la suite de la construction, on peut remplacer

```
draw (20,20)--(50,30)--(90,100)--cycle;
```

par

```
pair A,B,C;
A=(20,20);
B=(50,30);
C=(90,100);
draw A--B--C--cycle;
```

Pour faciliter la lecture et/ou la suite de la construction, on peut remplacer

```
1 draw (20,20)--(50,30)--(90,100)--cycle;
```

par

```
1 pair A,B,C;
  A=(20,20);
  B=(50,30);
  C=(90,100);
5 draw A--B--C--cycle;
```

Et pour les droites et demi-droites ?

Par défaut :

- METAPOST ne connaît que les segments ;

Par défaut :

- METAPOST ne connaît que les segments ;
- mais connaît la définition de *chemins*.

Par défaut :

- METAPOST ne connaît que les segments ;
- mais connaît la définition de *chemins*.
- que sait-il faire pour le type `pair` ?

Par défaut :

- METAPOST ne connaît que les segments ;
- mais connaît la définition de *chemins*.
- que sait-il faire pour le type `pair` ?
- Il reste donc à écrire

```
pair A,B,C;
A=(20,20);
B=(50,30);
path droiteab;
droiteab=2[A,B]--2[B,A];
draw droiteab;%trace 'la droite' (AB)
```

# Cercles

# Cercles

Par défaut, METAPOST connaît la commande `fullcircle` qui représente un cercle de diamètre unité (unité METAPOST : point postscript) centré sur l'origine.

# Cercles

Par défaut, METAPOST connaît la commande `fullcircle` qui représente un cercle de diamètre unité (unité METAPOST : point postscript) centré sur l'origine.
Comment tracer un cercle quelconque ?

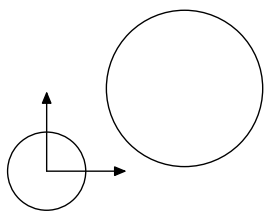
# Cercles

Par défaut, METAPOST connaît la commande `fullcircle` qui représente un cercle de diamètre unité (unité METAPOST : point postscript) centré sur l'origine.
Comment tracer un cercle quelconque ? À l'aide de *transformations*.

# Cercles

Par défaut, METAPOST connaît la commande `fullcircle` qui représente un cercle de diamètre unité (unité METAPOST : point postscript) centré sur l'origine.
Comment tracer un cercle quelconque ? À l'aide de *transformations*.

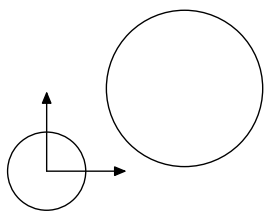
```
1 pair O,B; O=(0,0); B=(50,30);
  draw fullcircle scaled 1cm;%C(O,1/2)
  draw fullcircle scaled 2cm shifted B;%C(B,1)
```



# Cercles

Par défaut, METAPOST connaît la commande `fullcircle` qui représente un cercle de diamètre unité (unité METAPOST : point postscript) centré sur l'origine.
Comment tracer un cercle quelconque ? À l'aide de *transformations*.

```
1 pair O,B; O=(0,0); B=(50,30);
  draw fullcircle scaled 1cm;%C(O,1/2)
  draw fullcircle scaled 2cm shifted B;%C(B,1)
```



À noter la commande `abs` pour calculer la distance entre deux `pairs`.

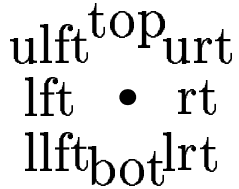
# Labreliser

# Labreliser

Pour écrire le nom d'un point, un texte...

```
1 label.pos(btex texte etex,A);
```

avec les conventions suivantes pour *pos* :

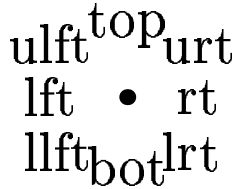


# Labreliser

Pour écrire le nom d'un point, un texte...

```
1 label.pos(btex texte etex,A);
```

avec les conventions suivantes pour *pos* :




Toutes les commandes TEX sont disponibles.

# Labreliser

Pour écrire le nom d'un point, un texte...

```
1 label.pos(btex texte etex,A);
```

avec les conventions suivantes pour *pos* : $\begin{matrix} \mathrm{ulft}^{\mathrm{top}}\mathrm{urt} \\ \mathrm{lft} \bullet \mathrm{rt} \\ \mathrm{llft}_{\mathrm{bot}}\mathrm{lrt} \end{matrix}$

Toutes les commandes TEX sont disponibles. Pour utiliser LATEX, on ajoutera

# Labreliser

Pour écrire le nom d'un point, un texte...

```
1 label.pos(btex texte etex,A);
```

avec les conventions suivantes pour *pos* : $\begin{matrix}\mathrm{ulft}^{\mathrm{top}}\mathrm{urt}\\ \mathrm{lft}\ \bullet\ \mathrm{rt}\\ \mathrm{llft}_{\mathrm{bot}}\mathrm{lrt}\end{matrix}$

Toutes les commandes TEX sont disponibles. Pour utiliser LATEX, on ajoutera avant la 1re figure :

# Labreliser

Pour écrire le nom d'un point, un texte...

```
1 label.pos(btex texte etex,A);
```

avec les conventions suivantes pour *pos* : $\begin{matrix} \mathrm{ulft} & & ^{\mathrm{top}} & & \mathrm{urt} \\ & \mathrm{lft} & \bullet & \mathrm{rt} & \\ \mathrm{llft} & & _{\mathrm{bot}} & & \mathrm{lrt} \end{matrix}$

Toutes les commandes TEX sont disponibles. Pour utiliser LATEX, on ajoutera avant la 1re figure :

```
1 verbatimtex
  %&latex
  \documentclass{article}
  ...%préambule qui va bien :)
5 \begin{document}
  etex
```

METAPOST

**couleur** On utilisera

**couleur** On utilisera

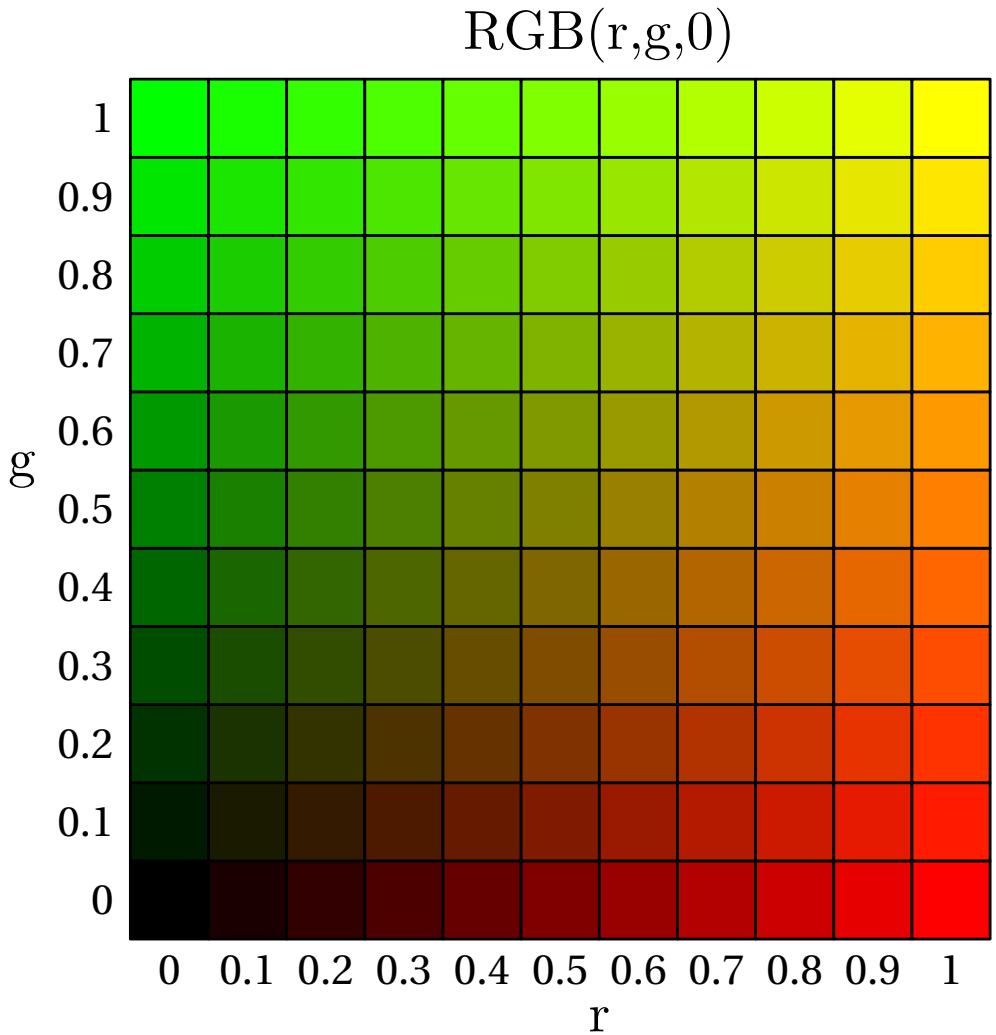
```
1 draw droiteab withcolor red;
```

▷ *couleurs*

**couleur** On utilisera

```
1 draw droiteab withcolor red;
```

▷ *couleurs*




METAPOST

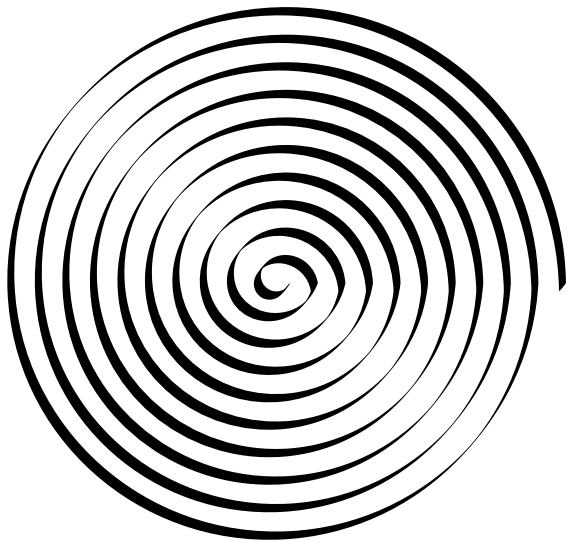
**épaisseur** METAPOST utilise une *plume* pour dessiner.

**épaisseur** METAPOST utilise une *plume* pour dessiner.
On utilisera donc, par exemple,

```
1 draw droiteab withpen pencircle scaled 2bp;
```

**épaisseur** METAPOST utilise une *plume* pour dessiner.
On utilisera donc, par exemple,

```
1 draw droiteab withpen pencircle scaled 2bp;
```



La taille des flèches changent en fonction de la plume :

La taille des flèches changent en fonction de la plume :

```
1 drawarrow A--B
    withpen pencircle
     scaled 1;
```

La taille des flèches changent en fonction de la plume :

```
1 drawarrow A--B
    withpen pencircle
     scaled 2;
```

La taille des flèches changent en fonction de la plume :

```
1 drawarrow A--B
    withpen pencircle
     scaled 3;
```

La taille des flèches changent en fonction de la plume :

```
1 drawarrow A--B
    withpen pencircle
     scaled 3;
```

On peut changer la forme des flèches avec les paramètres `ahlength` et `ahangle`.

```
1 ahangle:=10;
  ahlength:=8bp;
  drawarrow A--B;
```

**style du tracé** On utilisera

- `dashed withdots` pour tracer avec points ;

**style du tracé** On utilisera

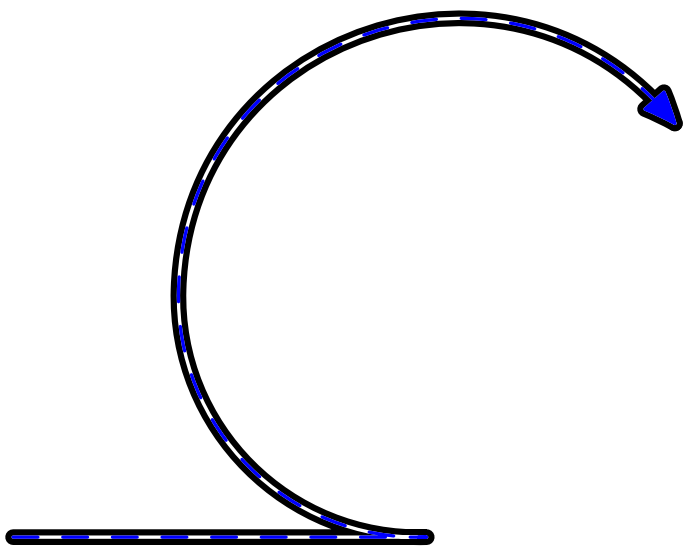
- `dashed withdots` pour tracer avec points ;
- `dashed evenly` pour tracer en pointillés ;

**style du tracé** On utilisera

- `dashed withdots` pour tracer avec points ;
- `dashed evenly` pour tracer en pointillés ;
- `dashed dashpattern(motif)` avec un motif tel que `on12bp off6bp on3bp off6bp` pour obtenir des traits d'axes (dans le cas présenté).

**style du tracé** On utilisera

- `dashed withdots` pour tracer avec points ;
- `dashed evenly` pour tracer en pointillés ;
- `dashed dashpattern(motif)` avec un motif tel que `on12bp off6bp on3bp off6bp` pour obtenir des traits d'axes (dans le cas présenté).



METAPOST

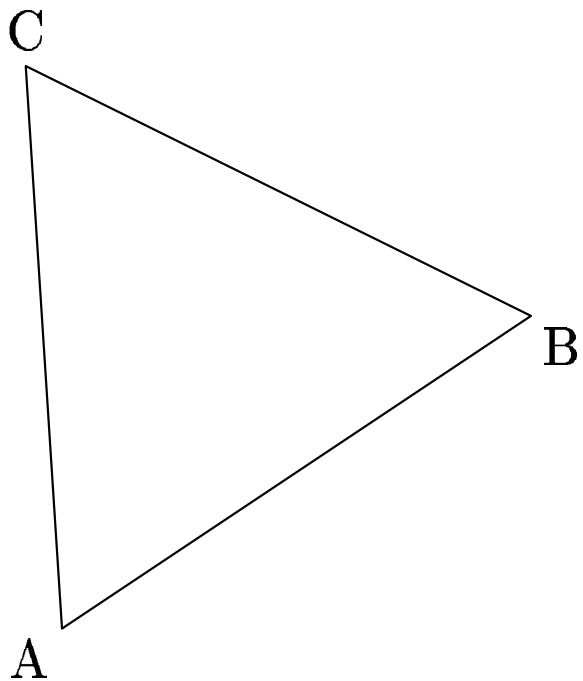
# Plan

# Polygones

- Triangle équilatéral.

# Polygones

- Triangle équilatéral.

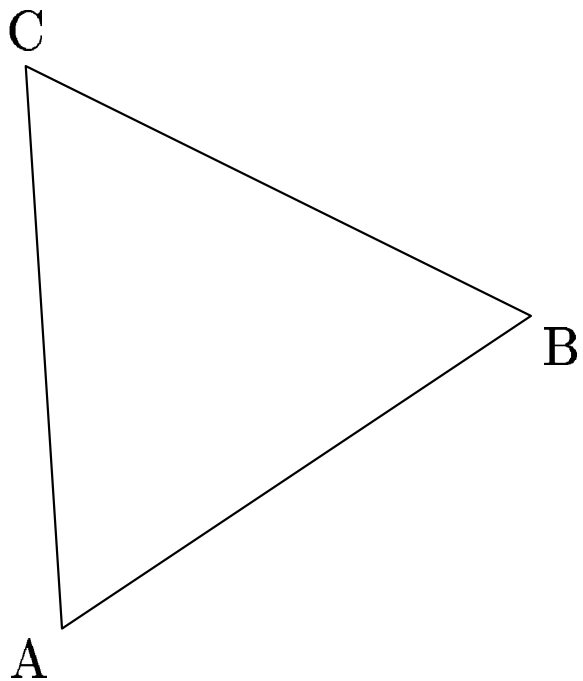



METAPOST

# Polygones

- Triangle équilatéral.

```
pair A,B,C;
A=(1cm,1cm);
B=(3cm,2cm);
C-A=(B-A) rotated 60;
draw A--B--C--cycle;
```

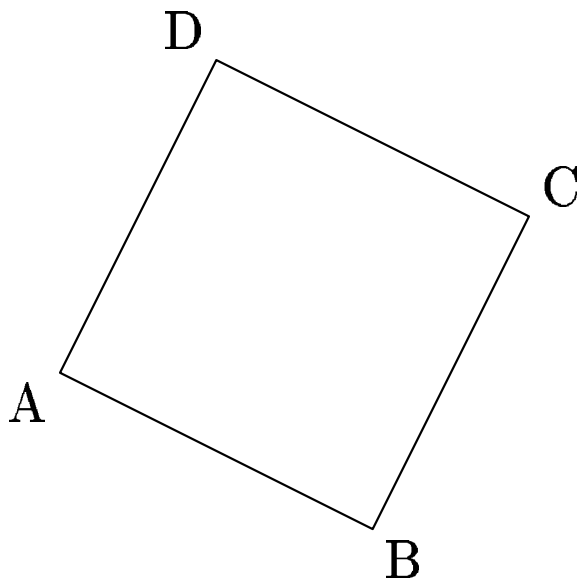


# Polygones

- Carré.

# Polygones

- Carré.

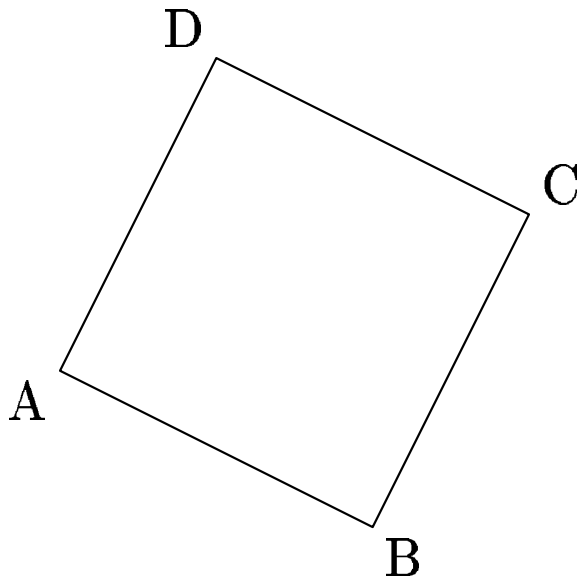



METAPOST

# Polygones

- Carré.

```
pair A,B,C,D;
A=(1cm,2cm);
B=(3cm,1cm);
D-A=(B-A) rotated 90;
C-D=B-A;
draw A--B--C--D--cycle;
```



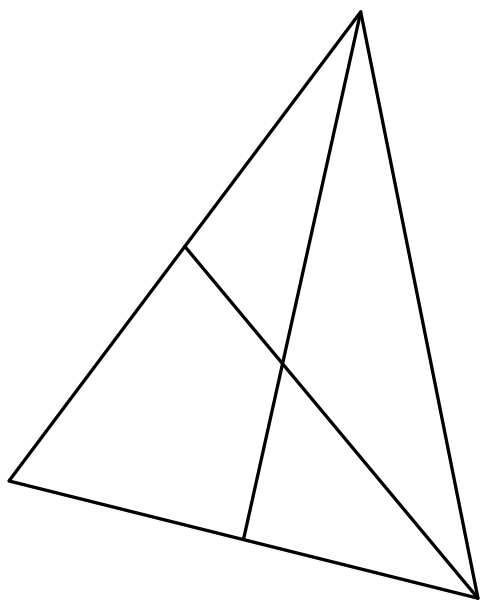

METAPOST

# Points

- Centre de gravité d'un triangle.

# Points

- Centre de gravité d'un triangle.

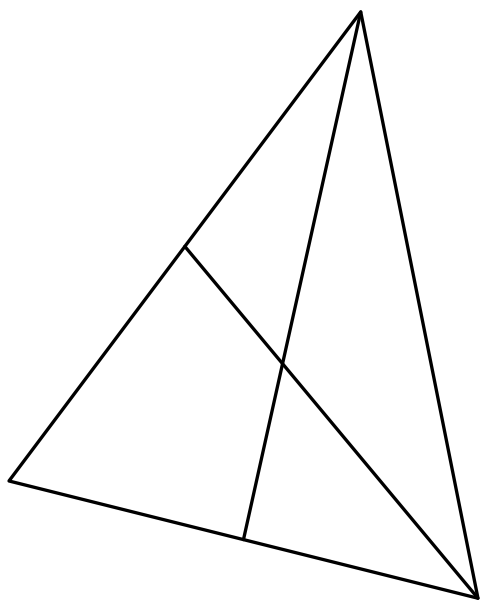


METAPOST

# Points

- Centre de gravité d'un triangle.

```
pair A,B,C,G;
A=(1cm,2cm); B=(3cm,1.5cm); C=(2.5cm,4cm);
draw A--B--C--cycle;
draw C--1/2[A,B];
draw B--1/2[A,C];
G=(B--1/2[A,C]) intersectionpoint (C--1/2[A,B]);
```

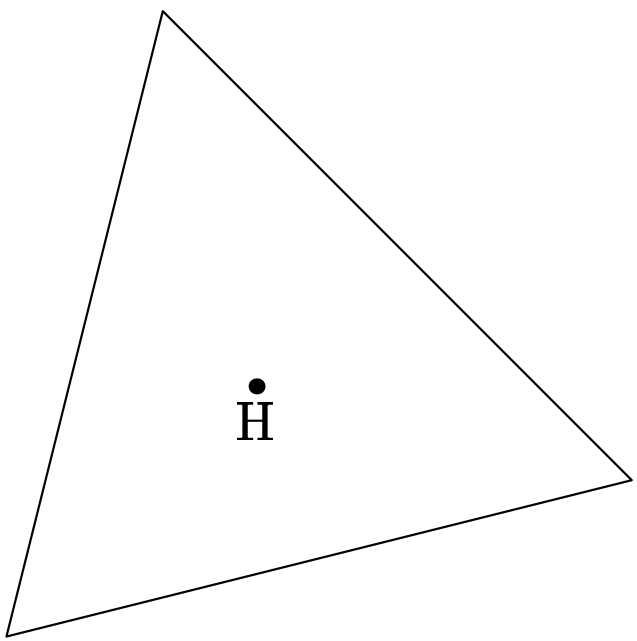


# Points

- Orthocentre d'un triangle.

# Points

- Orthocentre d'un triangle.

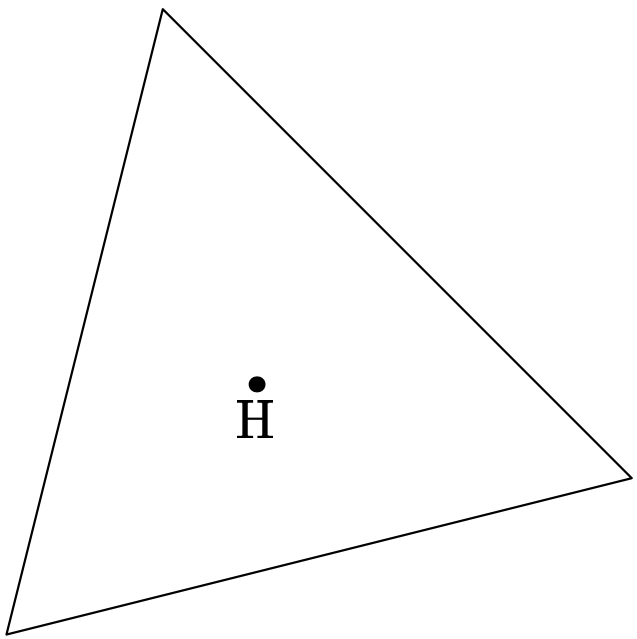



METAPOST

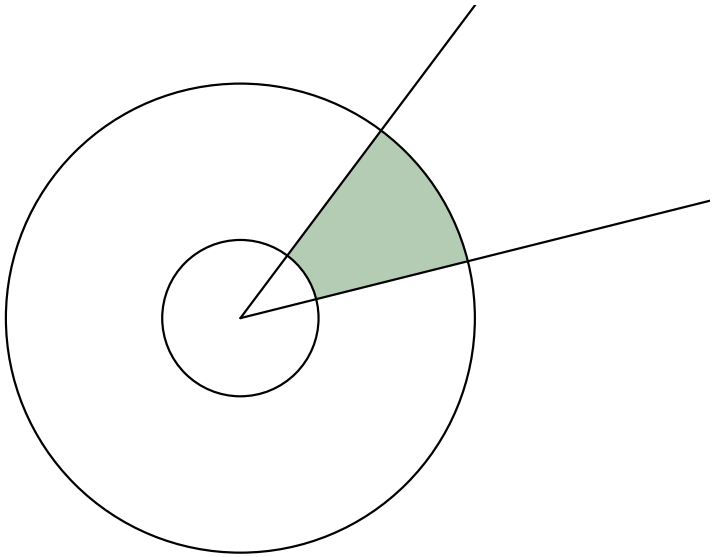
# Points

- Orthocentre d'un triangle.

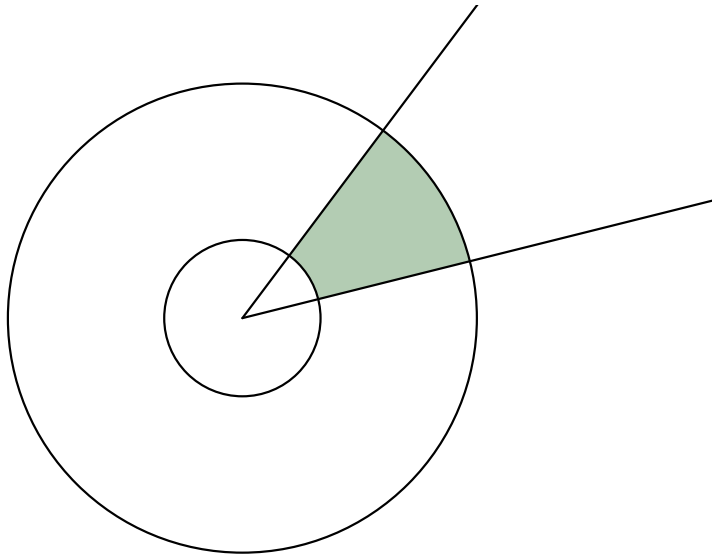
```
pair A,B,C,H;
A=(1cm,1cm);  B=(5cm,2cm);  C=(2cm,6cm);
H-A=whatever*((B-C) rotated 90);
H-C=whatever*((B-A) rotated 90);
draw A--B--C--cycle;
dotlabel.bot(btex H etex,H);
```



# Intersection de chemins



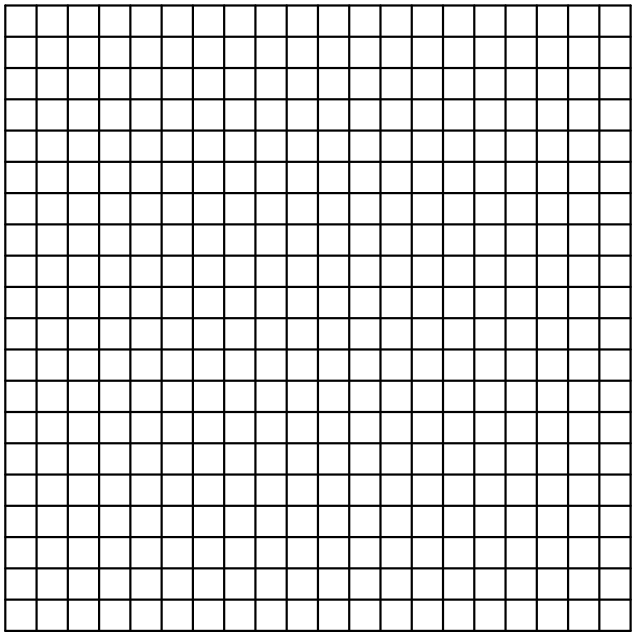
# Intersection de chemins



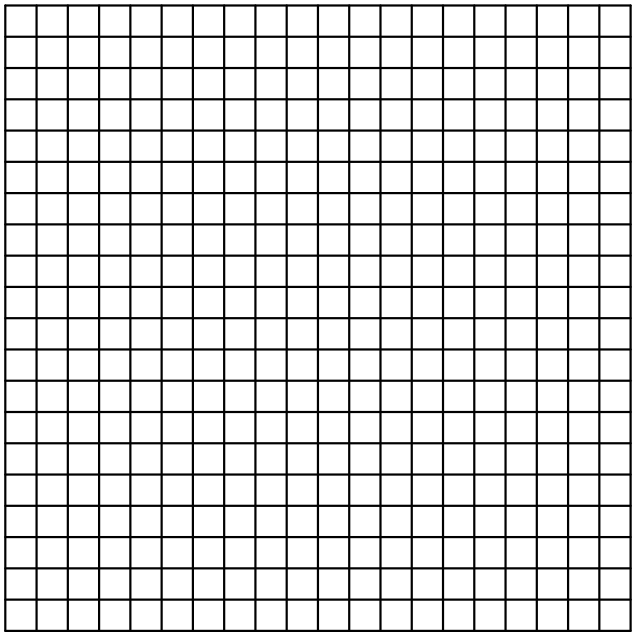
```
pair A;   A=(3cm,2cm);
path cd,ce,cf,cg,ch;
cd=fullcircle scaled 3cm shifted A;
ce=A--(A shifted(4cm,1cm));
cf=A--(A shifted(3cm,4cm));
cg=fullcircle scaled 1cm shifted A;
ch=buildcycle(ce,cd,cf,cg);
fill ch withcolor (0.7,0.8,0.7);
```

METAPOST

- . . . de constructions



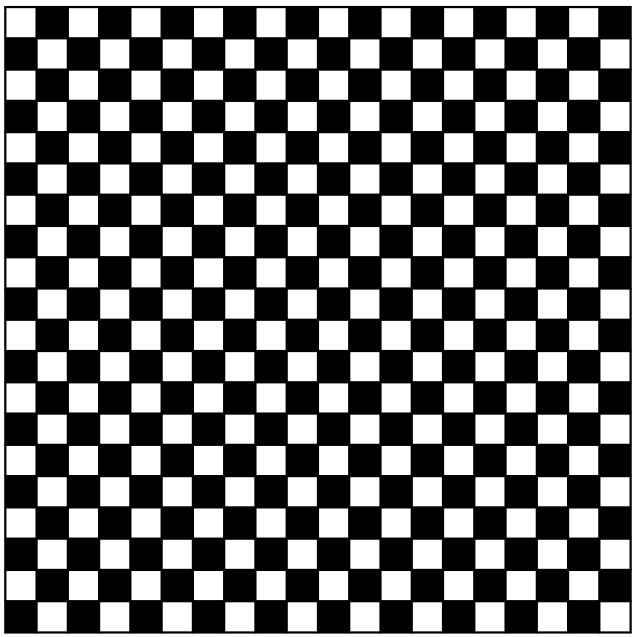
- . . . de constructions



```
path hor,ver;
hor=(0,0)--(4cm,0);
ver=(0,0)--(0,4cm);
for i=0 step 0.2 until 4:
  draw hor shifted (0,i*cm);
  draw ver shifted (i*cm,0);
endfor;
```

METAPOST

- . . . de points et de chemins



```
path hori[],verti[];
for i=0 upto 20:
  hori[i]=hor shifted(0,i*2mm);
  verti[i]=ver shifted(i*2mm,0);
endfor;
for k=0 upto 19:
  if (k mod 2)=0:
    for l=0 step 2 until 18:
      fill buildcycle(hori[k],verti[l],hori[k
          +1],verti[l+1]);
    endfor;
  else:
    for l=1 step 2 until 19:
      fill buildcycle(hori[k],verti[l],hori[k
          +1],verti[l+1]);
    endfor;
  fi;
endfor;
```

# Paramétrisation d'un chemin

On définit un chemin avec quatre points sous la forme

```
1 path ch;
  ch=(0,0)..(40,30)..(50,100)..(20,50);
```

# Paramétrisation d'un chemin

On définit un chemin avec quatre points sous la forme

```
1 path ch;
  ch=(0,0)..(40,30)..(50,100)..(20,50);
```

On considère un point particulier de ce chemin avec

```
1 A=point 1 of ch; B=point 3 of ch;
  C=point(length ch/2) of ch;
```

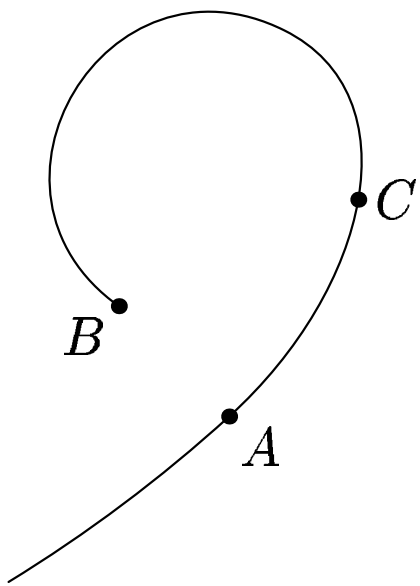
# Paramétrisation d'un chemin

On définit un chemin avec quatre points sous la forme

```
path ch;
ch=(0,0)..(40,30)..(50,100)..(20,50);
```

On considère un point particulier de ce chemin avec

```
A=point 1 of ch; B=point 3 of ch;
C=point(length ch/2) of ch;
```




METAPOST définit aussi l'abscisse curviligne en fonction de la longueur du chemin.

METAPOST

On peut aussi définir une partie d'un chemin.

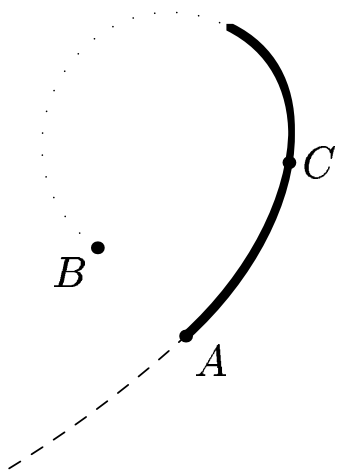
## On peut aussi définir une partie d'un chemin.

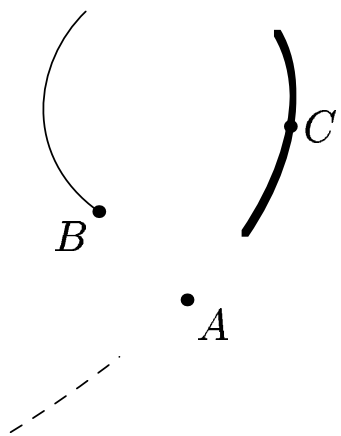
```
draw subpath(0,1) of ch dashed evenly;
draw subpath(1,2) of ch withpen pensquare
    scaled 1.5bp;
draw subpath(2,3) of ch dashed withdots;
%ou
draw subpath(0,length ch/5) of ch dashed
    evenly;
draw subpath(2*length ch/5,3*length ch/5) of
    ch withpen pensquare scaled 1.5bp;
draw subpath(4*length ch/5,length ch) of ch;
```

METAPOST

## On peut aussi définir une partie d'un chemin.

```
draw subpath(0,1) of ch dashed evenly;
draw subpath(1,2) of ch withpen pensquare
    scaled 1.5bp;
draw subpath(2,3) of ch dashed withdots;
%ou
draw subpath(0,length ch/5) of ch dashed
    evenly;
draw subpath(2*length ch/5,3*length ch/5) of
    ch withpen pensquare scaled 1.5bp;
draw subpath(4*length ch/5,length ch) of ch;
```

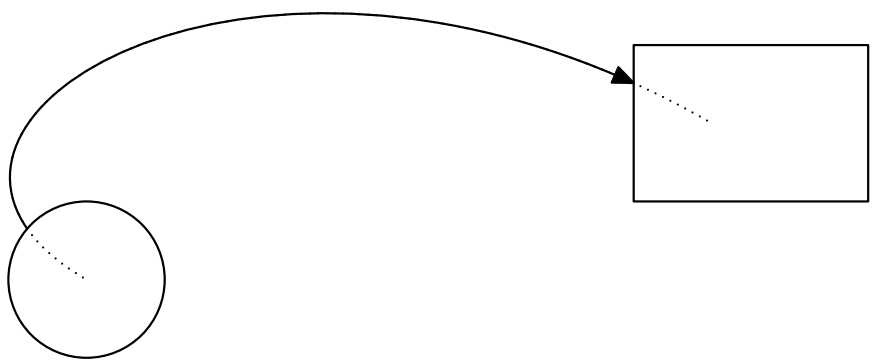



METAPOST

Un chemin peut également être coupé en fonction de ses intersections avec d'autres chemins avec les commandes `cutbefore` et `cutafter`.

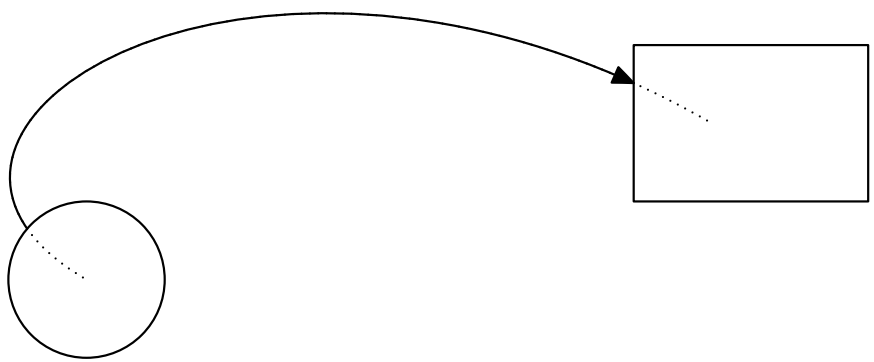
Un chemin peut également être coupé en fonction de ses intersections avec d'autres chemins avec les commandes `cutbefore` et `cutafter`.



METAPOST

Un chemin peut également être coupé en fonction de ses intersections avec d'autres chemins avec les commandes `cutbefore` et `cutafter`.

```
z0=(0,0);z1=(4cm,1cm);
cercle=fullcircle scaled 1cm shifted z0;
rectangle=((-5mm,-5mm)--(10mm,-5mm)--(10mm,5
    mm)--(-5mm,5mm)--cycle) shifted z1;
p=z0{dir150}..z1{dir-30};
draw p dashed withdots scaled 0.3;
drawarrow (p cutbefore cercle cutafter
    rectangle);
```



# Plan

- De nombreux fichiers disponibles ;

- De nombreux fichiers disponibles ;
- être attentif aux fichiers nécessaires ;

```
1    input nomfichiermacro;
```

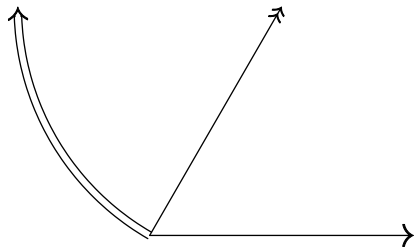
- De nombreux fichiers disponibles ;
- être attentif aux fichiers nécessaires ;

```
1    input nomfichiermacro ;
```

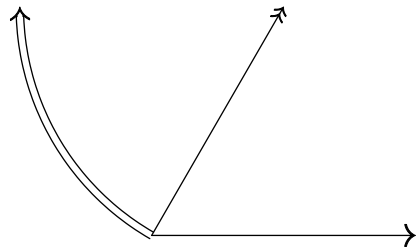
- apprentissage favorisé.

BEUVRAGES

- http://www.student.nada.kth.se/~f91-tek/cm_arrows.html

- `http://www.student.nada.kth.se/~f91-tek/cm_arrows.html`
- 

- http://www.student.nada.kth.se/~f91-tek/cm_arrows.html
-

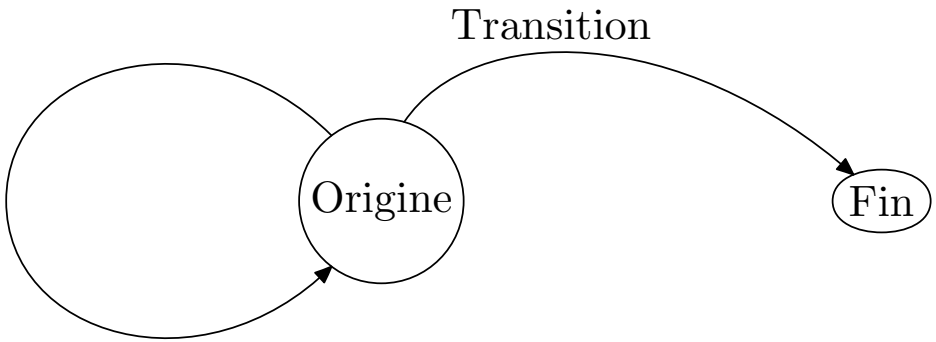


```
input cmarrows;

setup_cmarrows(
  arrow_name="texarrow";  parameter_file="cmr10.mf";  macro_name="arrowa"
      );

setup_cmarrows(
  arrow_name="twoheadarrow";  parameter_file="cmr9.mf";  macro_name="
      arrowb");

setup_cmarrows(
  arrow_name="doublearrow";  parameter_file="cmr8.mf";  macro_name="
      arrowc");
beginfig(1);
  arrowa (0,0)--60pt*dir 0;
  arrowb (0,0)--60pt*dir 60;
  arrowc (0,0)..{up}60pt*dir 120;
endfig;
```

METAPOST

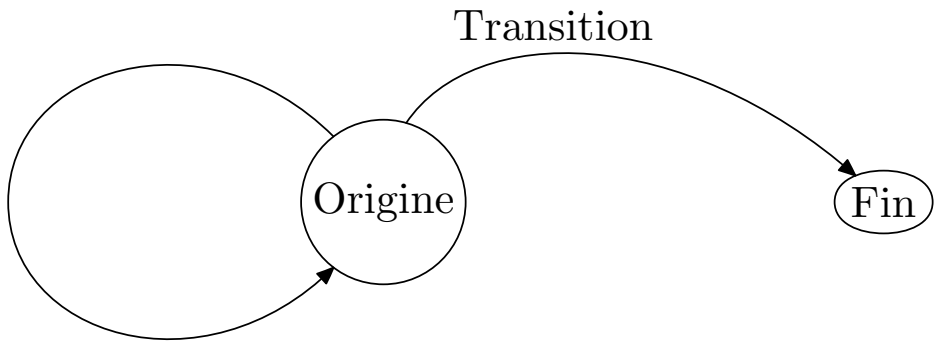
- fourni dans les distributions LaTeX.

- fourni dans les distributions LaTeX.
-




METAPOST

- fourni dans les distributions LaTeX.
-

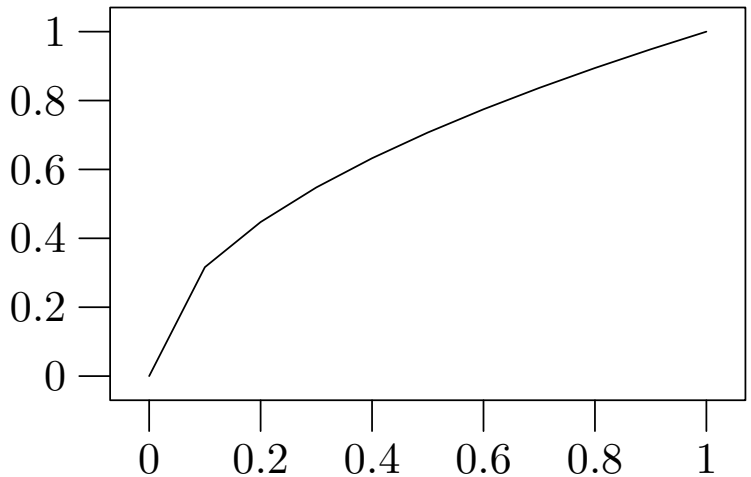



```
input boxes;

beginfig(1);
  circleit.a("Origine");

  circleit.b("Fin");
  b.dx=b.dy;

  a.c=(0,0);b.c=(4cm,0);
  drawboxed(a,b);

  path p;  p=a.c{up}..b.c{dir-45};
  drawarrow p cutbefore bpath.a cutafter bpath.b;
  label.top("Transition",point 0.5 of p);

  path q;  q=a.c{dir120}..a.c shifted(-3cm,0)..a.c{dir60};
  drawarrow q cutbefore bpath.a cutafter bpath.a;
endfig;
```

METAPOST

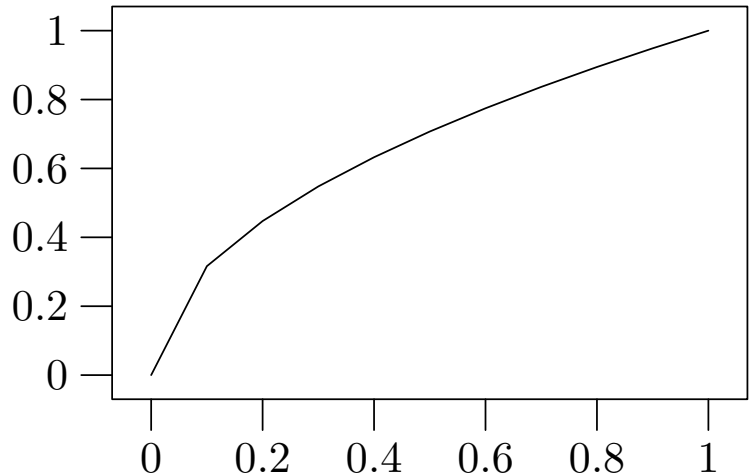
- fourni dans les distributions LaTeX.

- fourni dans les distributions LaTeX.
-




METAPOST

- fourni dans les distributions LaTeX.
-

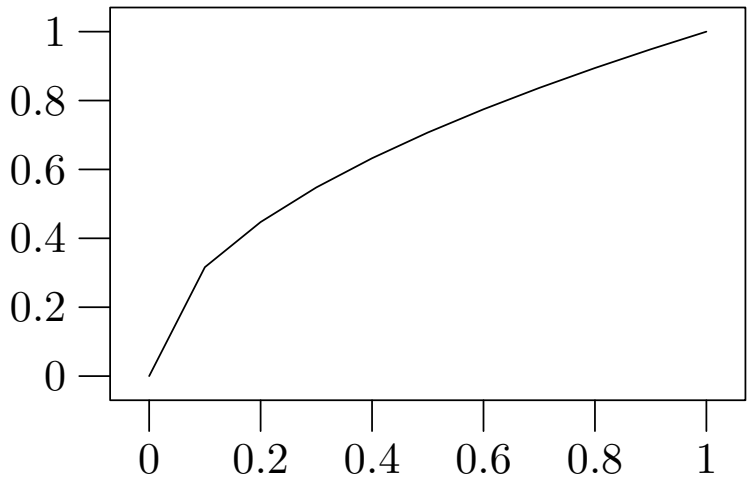



Fichier `data.d`

```
0       0
0.1     0.316227766
0.2     0.447213595
0.3     0.547722558
0.4     0.632455532
0.5     0.707106781
0.6     0.774596669
0.7     0.836660027
0.8     0.894427191
0.9     0.948683298
1       1
```

METAPOST

- fourni dans les distributions LaTeX.
-




Fichier `data.d`

```
0       0
0.1     0.316227766
0.2     0.447213595
0.3     0.547722558
0.4     0.632455532
0.5     0.707106781
0.6     0.774596669
0.7     0.836660027
0.8     0.894427191
0.9     0.948683298
1       1
```

```
1 input graph;

  beginfig(1);
    draw begingraph(144bp,89bp);
5     gdraw "data.d";
    endgraph;
  endfig;

  end
```

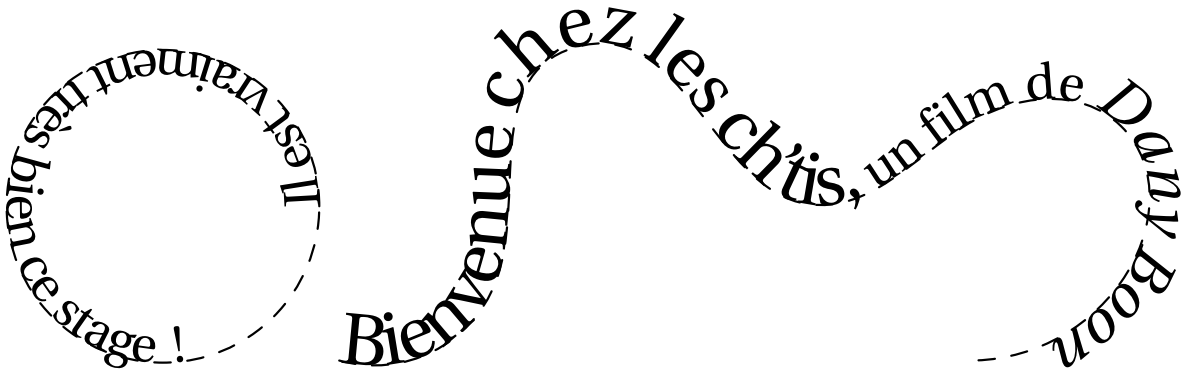
METAPOST

BEUVRAGES

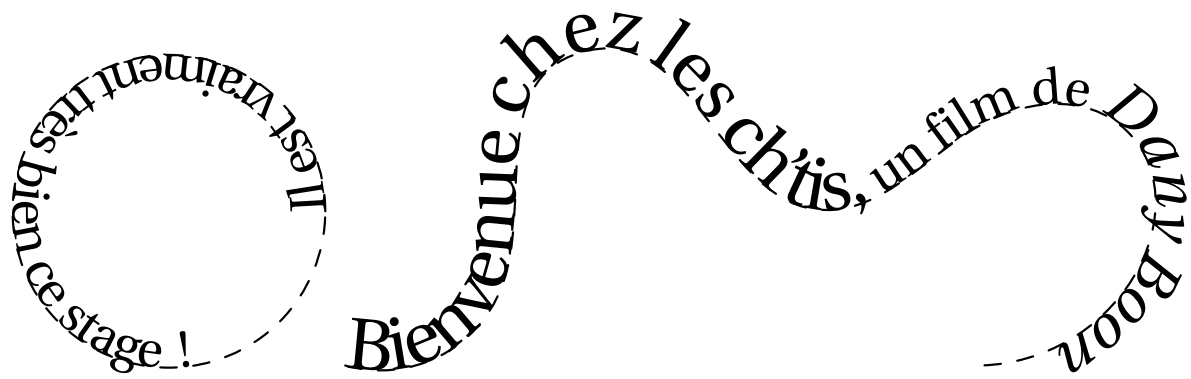
- ctan.org/tex-archive/graphics/metapost/contrib/macros/textpath/

BEUVRAGES

- ctan.org/tex-archive/graphics/metapost/contrib/macros/textpath/
-



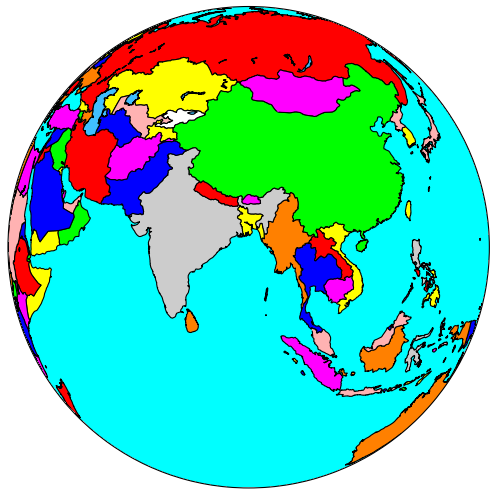
- ctan.org/tex-archive/graphics/metapost/contrib/macros/textpath/
-



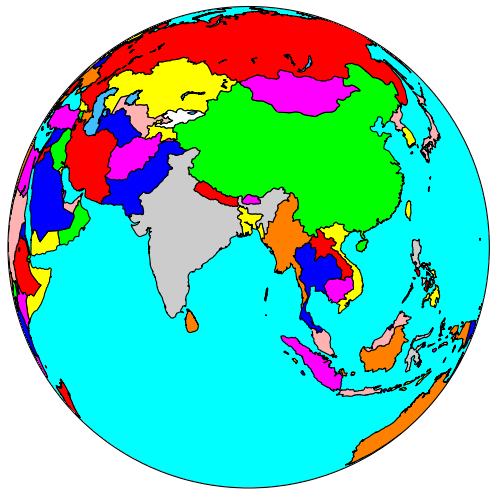

```
input latexmp
setupLaTeXMP(class="article",options="10pt",fontencoding="T1",
      inputencoding="latin1",language="frenchb",packages="fourier,
      textpathmp");
input textpath;

prologues:=0;

beginfig(1);
  path cc;
  cc=fullcircle scaled 2cm shifted (3cm,3cm);
  draw cc dashed evenly;
  draw textpath("Il est vraiment très bien ce stage !", cc, 0);
endfig;
```

BEUVRAGES

- `syracuse-dev.org/mpst-globe/browser/trunk/mp-geo`

- `syracuse-dev.org/mpst-globe/browser/trunk/mp-geo`
- 

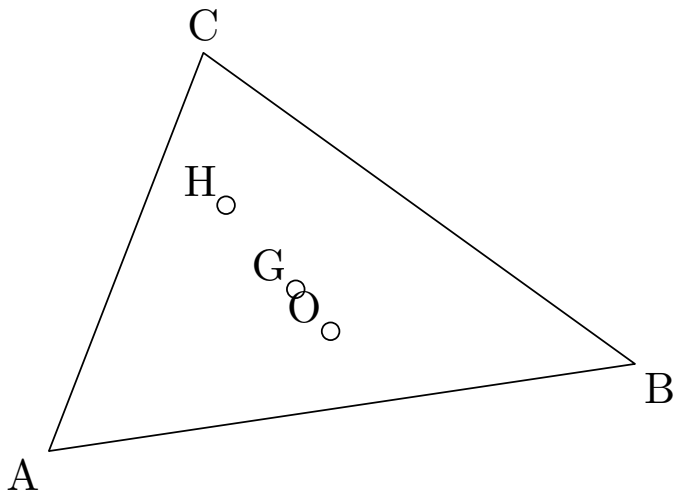
BEUVRAGES

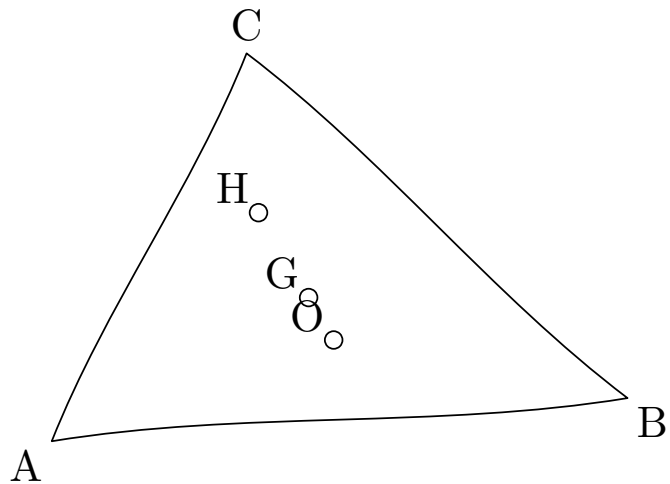
- syracuse-dev.org/mpst-globe/browser/trunk/mp-geo
- 

```
input mp-geo;

Mappemonde(90,20);

end
```

BEUVRAGES

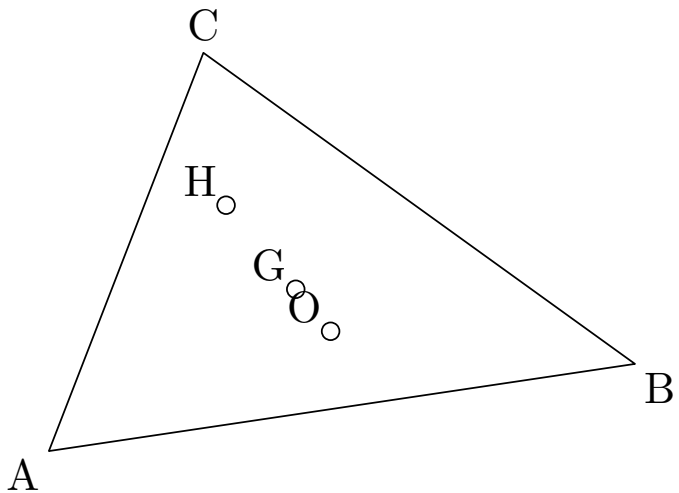
- melusine.eu.org/syracuse/poulecl/geometriesyr16/
-

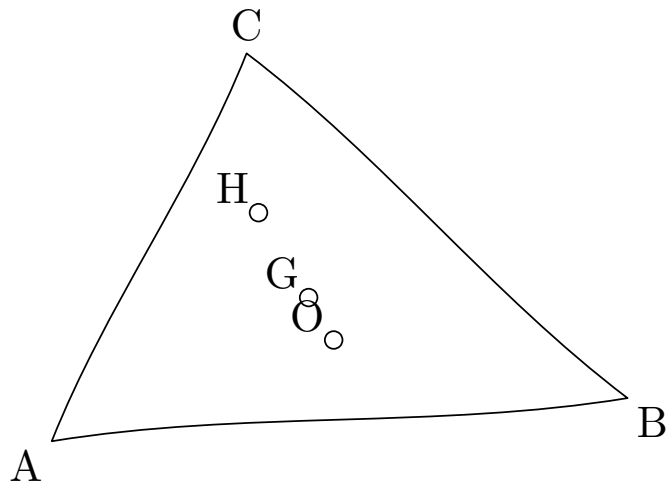


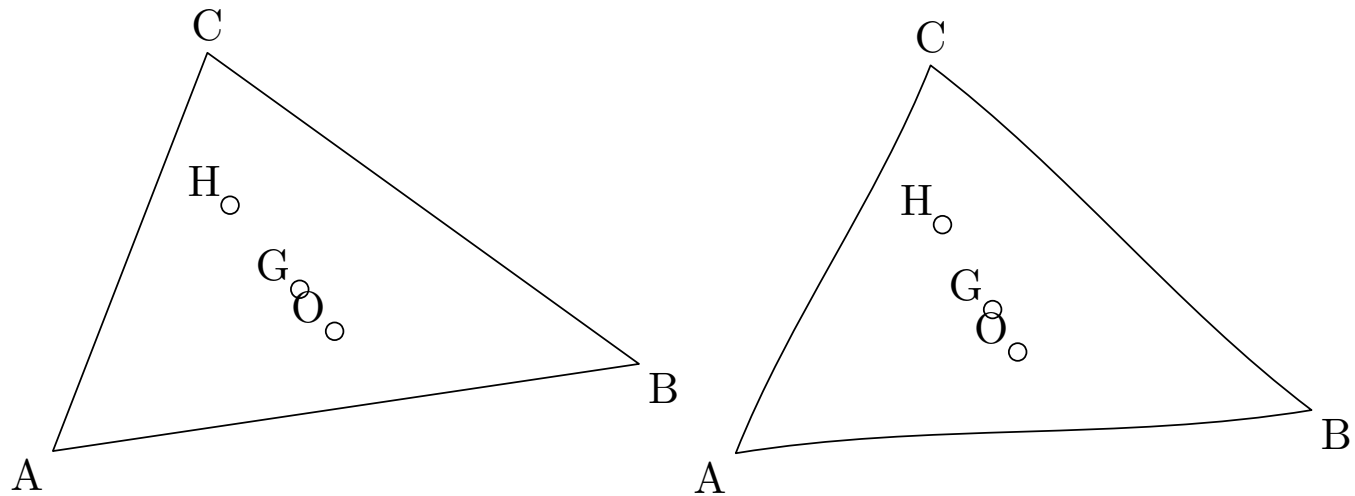
BEUVRAGES

- melusine.eu.org/syracuse/poulecl/geometriesyr16/
-



- melusine.eu.org/syracuse/poulecl/geometriesyr16/
-




```
input geometriesyr16;

figure(0,0,10u,10u);
pair A,B,C,H,G,O;
trace triangleqcq(A,B,C);
H=Orthocentre(A,B,C);
O=CentreCercleC(A,B,C);
G=iso(A,B,C);
nomme.llft(A);nomme.lrt(B);nomme.top(C);
marque_p:=''creux'';
nomme.top(H);nomme.top(G);nomme.top(O);
fin;
```

METAPOST

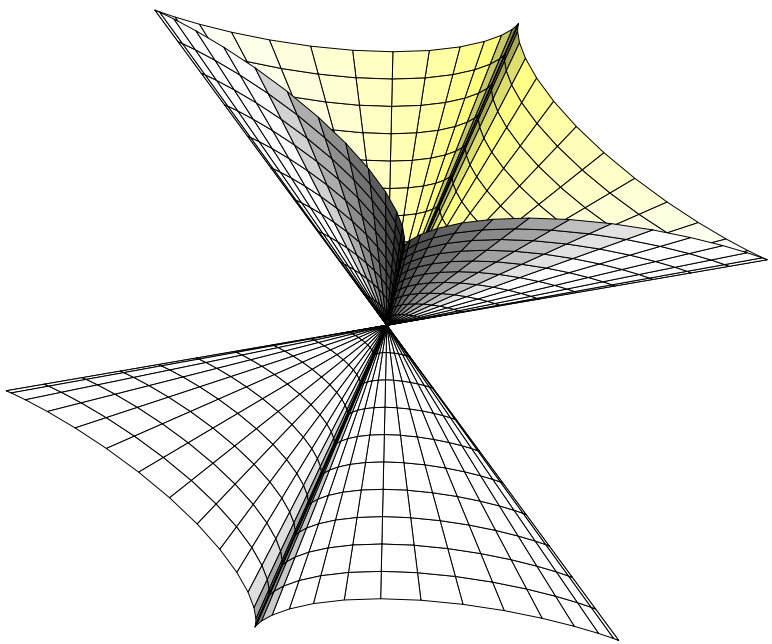
# mp-solid

# mp-solid

```
input mp-solid

outcolor:=blanc;
incolor:=0.5[jaune,blanc];

figureespace(-10u,-10u,10u,10u);
Initialisation(500,10,20,50);
nb:=60;subh:=10;
angx:=-20;
Ferme2:=false;
ObjetCone2("(3*(cos(u)**3),3*(sin(u
      )**3),-1)",-pi,pi,-1,"orig
      =(0,0,1)");
AffichageObjet2;
finespace;
end
```

# mp-solid

```
1  input mp-solid

   outcolor:=blanc;
   incolor:=0.5[jaune,blanc];
5
   figureespace(-10u,-10u,10u,10u);
   Initialisation(500,10,20,50);
   nb:=60;subh:=10;
   angx:=-20;
10 Ferme2:=false;
   ObjetCone2("(3*(cos(u)**3),3*(sin(u
         )**3),-1)",-pi,pi,-1,"orig
         =(0,0,1)");
   AffichageObjet2;
   finespace;
   end
```

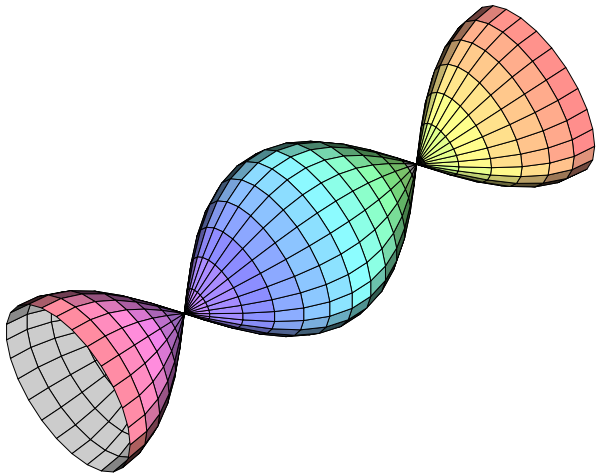


# mp-solid

```
input mp-solid

% 1''
figureespace(-10u,-10u,10u,10u);
Initialisation(1000,50,50,50);
incolor:=gris;
arcenciel:=true;
draw Sparam("(u,cos(u)*cos(v),cos(u)*
      sin(v))",0,2*pi,0.25132,0,2*pi
      ,0.25132);
finespace;
end
```

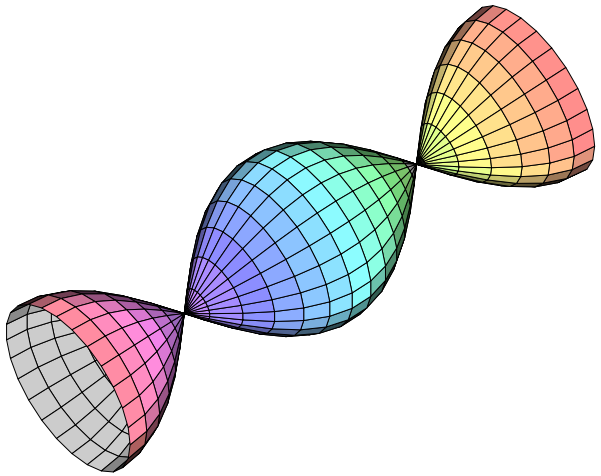
# mp-solid

```
1  input mp-solid

   % 1''
   figureespace(-10u,-10u,10u,10u);
5  Initialisation(1000,50,50,50);
   incolor:=gris;
   arcenciel:=true;
   draw Sparam("(u,cos(u)*cos(v),cos(u)*
        sin(v))",0,2*pi,0.25132,0,2*pi
        ,0.25132);
   finespace;
10 end
```



# mp-solid

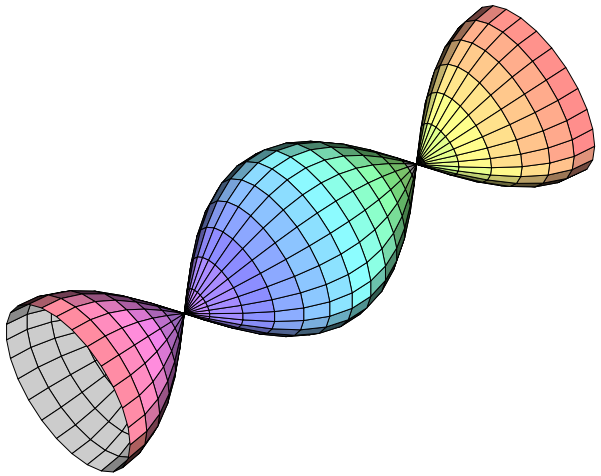
```
1  input mp-solid

   % 1''
   figureespace(-10u,-10u,10u,10u);
5  Initialisation(1000,50,50,50);
   incolor:=gris;
   arcenciel:=true;
   draw Sparam("(u,cos(u)*cos(v),cos(u)*
        sin(v))",0,2*pi,0.25132,0,2*pi
        ,0.25132);
   finespace;
10 end
```



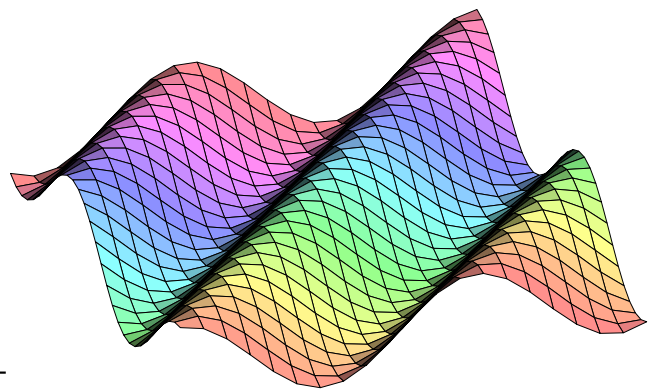
```
1  input mp-solid

   % 1'
   figureespace(-10u,-10u,10u,10u);
5  Initialisation(500,70,30,20);
   arcenciel:=true;
   incolor:=1.1*gris;
   draw SurfZ("cos(abs(X-Y))"
        ,-5,5,-5,5,30,60);
   finespace;
10 end
```

# mp-solid

```
1  input mp-solid

   % 1''
   figureespace(-10u,-10u,10u,10u);
5  Initialisation(1000,50,50,50);
   incolor:=gris;
   arcenciel:=true;
   draw Sparam("(u,cos(u)*cos(v),cos(u)*
        sin(v))",0,2*pi,0.25132,0,2*pi
        ,0.25132);
   finespace;
10 end
```



```
1  input mp-solid

   % 1'
   figureespace(-10u,-10u,10u,10u);
5  Initialisation(500,70,30,20);
   arcenciel:=true;
   incolor:=1.1*gris;
   draw SurfZ("cos(abs(X-Y))"
        ,-5,5,-5,5,30,60);
   finespace;
10 end
```



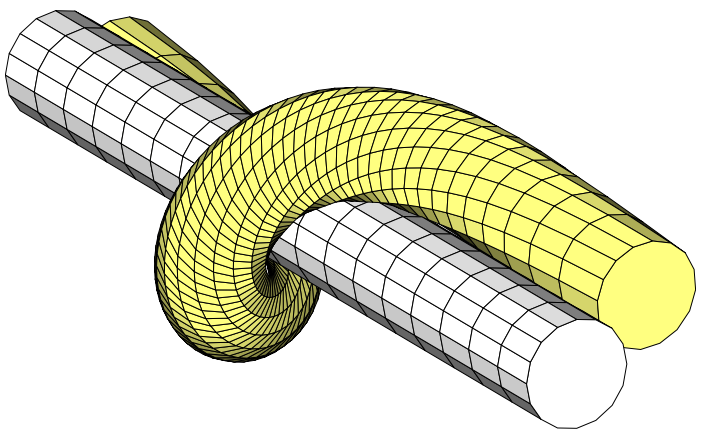
METAPOST

# mp-solid

```
%by pst-solides3d
input mp-solid %27''

figureespace(-20u,-20u,20u,20u);
Initialisation(500,60,20,30);
nb:=16; subh:=18;

outcolor:=blanc;  incolor:=0.5[vert,white]; angx:=90; TR:=(2,9,0);
Objetcylindre1("r=1","h=18");

outcolor:=0.5[jaune,blanc]; incolor:=0.5[violet,blanc]; angx:=0; TR
    :=(0,0,0);
ObjetTube2("(2*(1+cos(t)),2*tan(t/2),2*sin(t))","(-2*sin(t),2/((cos(t/2)
    )**2),2*cos(t))",1,-2.7468,71,0.0763);

nbobj:=2; DessineFusion;

finespace;
```

# mp-solid

```
%by pst-solides3d
input mp-solid %27''

figureespace(-20u,-20u,20u,20u);
Initialisation(500,60,20,30);
nb:=16; subh:=18;

outcolor:=blanc;  incolor:=0.5[vert,white]; angx:=90; TR:=(2,9,0);
Objetcylindre1("r=1","h=18");

outcolor:=0.5[jaune,blanc]; incolor:=0.5[violet,blanc]; angx:=0; TR
    :=(0,0,0);
ObjetTube2("(2*(1+cos(t)),2*tan(t/2),2*sin(t))","(-2*sin(t),2/((cos(t/2)
    )**2),2*cos(t))",1,-2.7468,71,0.0763);

nbobj:=2; DessineFusion;

finespace;
```



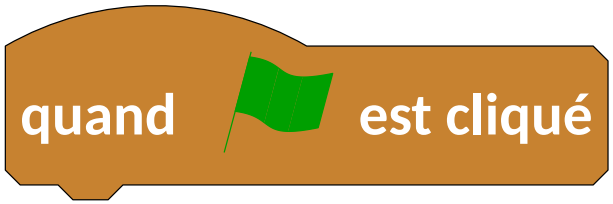
# mp-scratch

# mp-scratch

- `http://melusine.eu.org/syracuse/G/mp-scratch/`

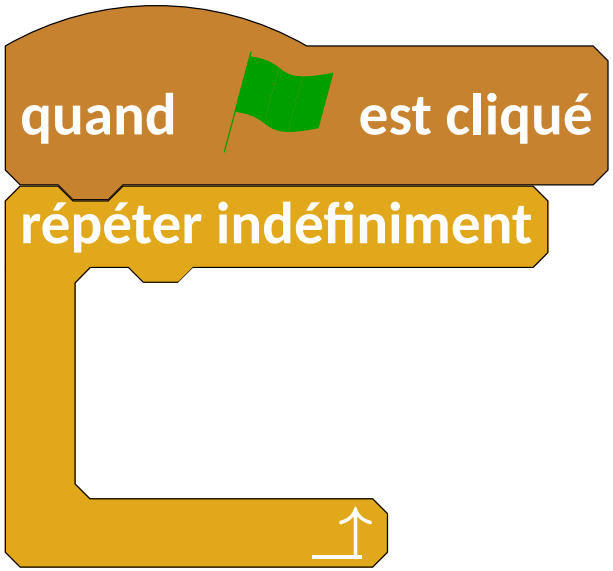
# mp-scratch

- `http://melusine.eu.org/syracuse/G/mp-scratch/`



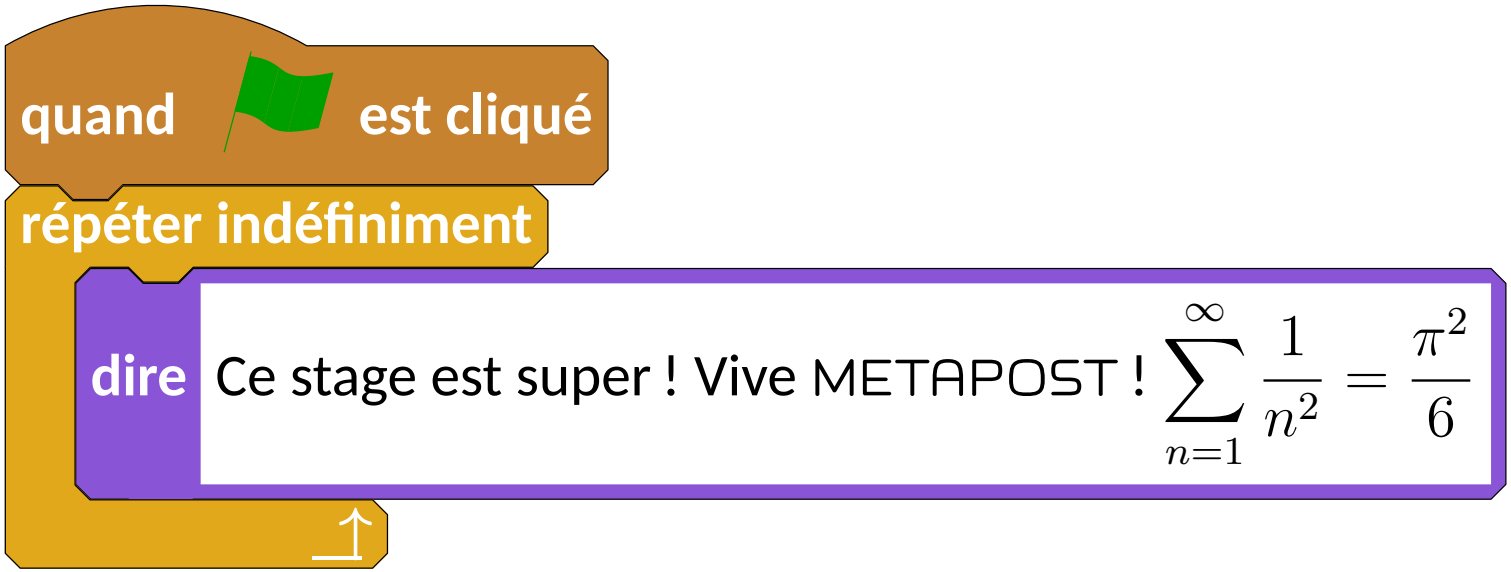
# mp-scratch

- `http://melusine.eu.org/syracuse/G/mp-scratch/`



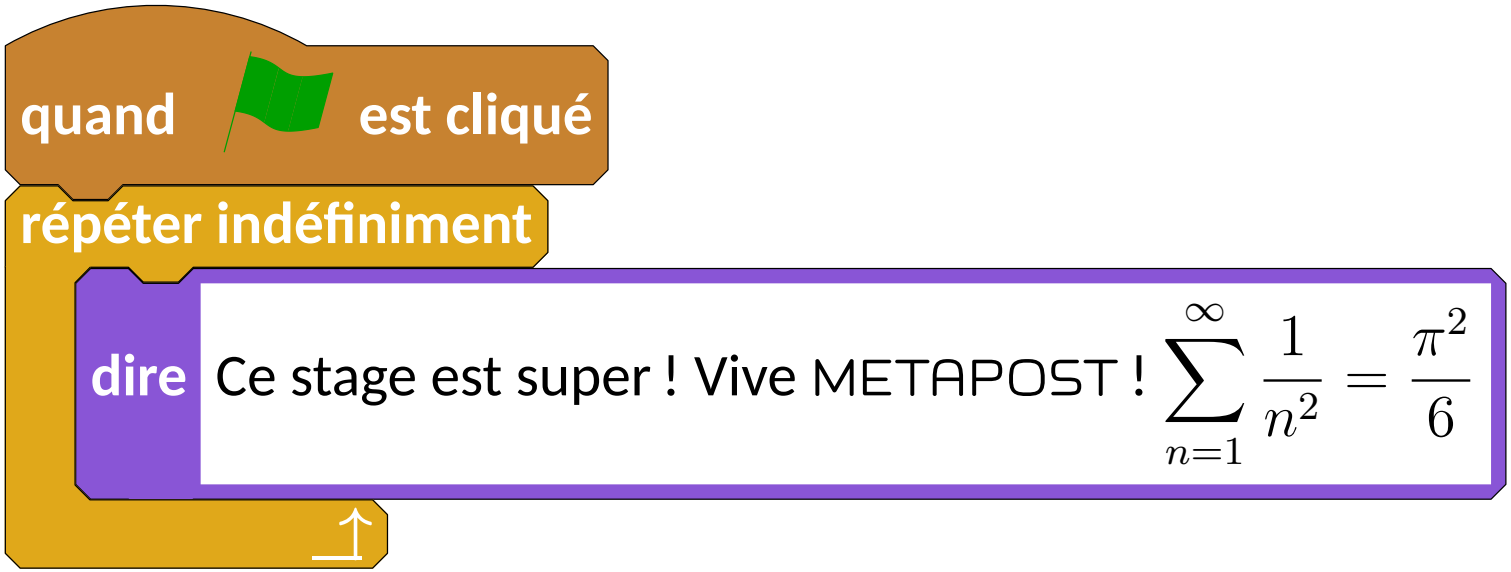
# mp-scratch

- `http://melusine.eu.org/syracuse/G/mp-scratch/`



# mp-scratch

- http://melusine.eu.org/syracuse/G/mp-scratch/



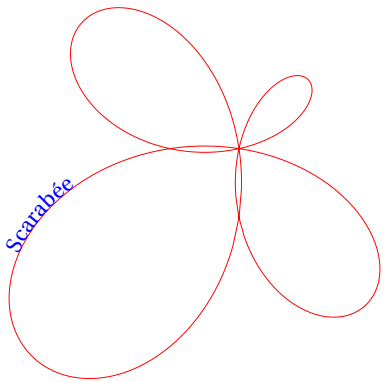

```
1 input mp-scratch

  beginfig(1);
  draw Drapeau;
5 draw RepeterI;
  draw Dire("\opSimple{Ce␣stage␣est␣super␣!␣Vive␣\MP␣!␣$\displaystyle\sum_{
       n=1}^\infty
␣␣\frac1{n^2}=\frac{\pi^2}6$}");
  draw FinBlocRepeterI(10);
  endfig;
```

METAPOST

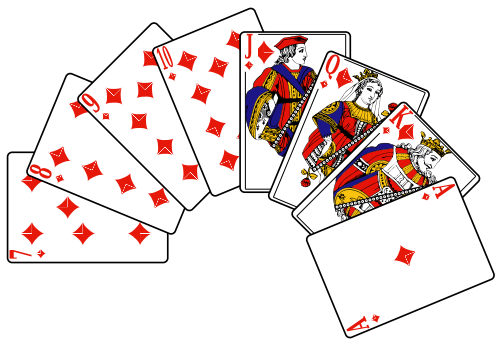
# packages utiles

# packages utiles

textpath




exteps



gmp (associé à LaTeX)

Magnifique journée ! L'organisation est super, les intervenants sont super, le public est super ! Vive Denis !

Magnifique journée ! L'organisation est super, les intervenants sont super, le public est super ! Vive Denis !

METAPOST

- `epsincl` pour include des fichiers eps dans une figure METAPOST.
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/epsincl/`

- `epsincl` pour include des fichiers eps dans une figure METAPOST.
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/epsincl/`
- `hatching` pour hachurer.
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/hatching/`

- `epsincl` pour include des fichiers eps dans une figure METAPOST.
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/epsincl/`
- `hatching` pour hachurer.
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/hatching/`
- `tableauvariations`
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/tableauvariations/`

- `epsincl` pour include des fichiers eps dans une figure METAPOST.
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/epsincl/`
- `hatching` pour hachurer.
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/hatching/`
- `tableauvariations`
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/tableauvariations/`
- `mpcirc` pour des circuits électriques
  `ci.uofl.edu/tom/software/LaTeX/mpcirc/`

- `epsincl` pour include des fichiers eps dans une figure METAPOST.
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/epsincl/`
- `hatching` pour hachurer.
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/hatching/`
- `tableauvariations`
  `ctan.org/tex-archive/graphics/metapost/contrib/macros/tableauvariations/`
- `mpcirc` pour des circuits électriques
  `ci.uofl.edu/tom/software/LaTeX/mpcirc/`
- et plein d'autres...

# Plan


John Hobby. *METAPOST : A User's Manual*. 2017.
`https://www.tug.org/docs/metapost/mpman.pdf`


André Heck. *Learning METAPOST by doing*. 2005.
`https://staff.science.uva.nl/a.j.p.heck/Courses/mptut.pdf`


Urs Oswald *A very brief tutorial*. 2002.
`http://www.ursoswald.ch/metapost/tutorial.html`


Laurent Chéno. *Une introduction à METAPOST*. 1999.


`www.melusine.eu.org/syracuse/metapost/`


`www.melusine.eu.org/syracuse/poulecl/albums/`


`www.melusine.eu.org/lab/bmp/`

# Type `pair`

METAPOST sait :

# Type `pair`

METAPOST sait :

- ajouter ou soustraire des couples de coordonnées ;

# Type `pair`

METAPOST sait :

- ajouter ou soustraire des couples de coordonnées ;
- les multiplier par un scalaire ;

# Type `pair`

METAPOST sait :

- ajouter ou soustraire des couples de coordonnées ;
- les multiplier par un scalaire ;
- définir un barycentre de deux points

$$t[a,b] = ta + (1-t)b = a + (1-t)(b-a)$$

*Retour*

# Type `path`

- On définit un chemin avec

```
1 path p;
```

# Type `path`

- On définit un chemin avec

```
1 path p;
```

- Si `p` représente un chemin, METAPOST sait :

# Type path

- On définit un chemin avec

```
path p;
```

- Si p représente un chemin, METAPOST sait :
  - le tracer :

```
draw p;
```

# Type path

- On définit un chemin avec

```
path p;
```

- Si p représente un chemin, METAPOST sait :
  - le tracer :

```
draw p;
```

  - remplir un chemin fermé :

```
fill p--cycle;
```

*Retour*

# Type `transform`

- Représente toutes les transformations affines. Applicable à des objets de type `pair`, `path`.

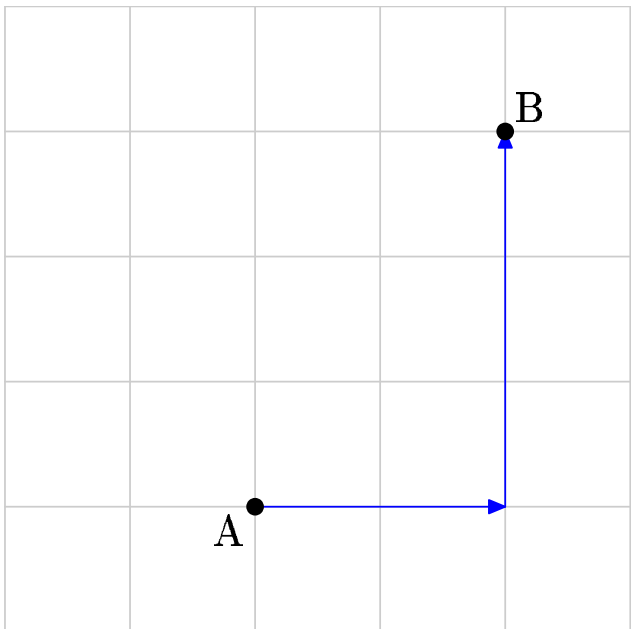
# Type `transform`

- Représente toutes les transformations affines. Applicable à des objets de type `pair`, `path`.
- METAPOST connaît

# Type `transform`

- Représente toutes les transformations affines. Applicable à des objets de type `pair`, `path`.
- METAPOST connaît les translations,

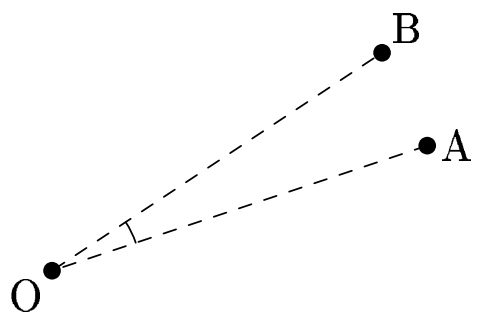
```
1 B=A shifted (2,3);
```



# Type `transform`

- Représente toutes les transformations affines. Applicable à des objets de type `pair`, `path`.
- METAPOST connaît les translations, les rotations,

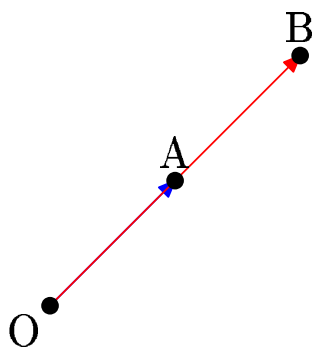
```
1 %B=A shifted (2,3);
  B=A rotated 15;
```



# Type `transform`

- Représente toutes les transformations affines. Applicable à des objets de type `pair`, `path`.
- METAPOST connaît les translations, les rotations, les homothéties.

```
%B=A shifted (2,3);
%B=A rotated 15;
B=A scaled 2;%OB=2OA
```



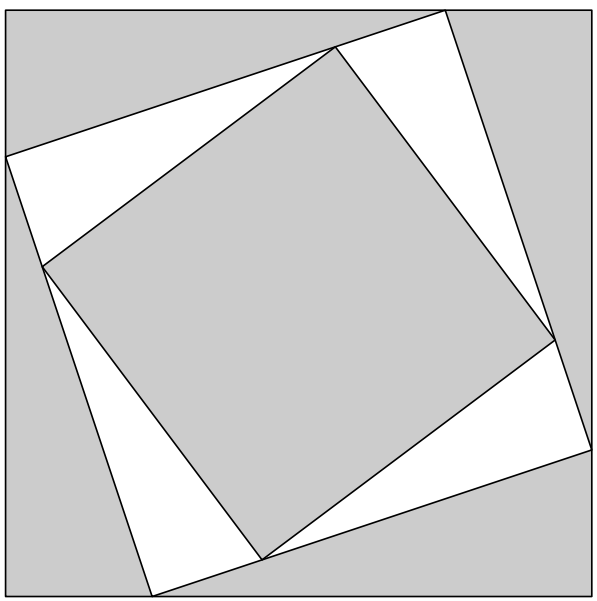
METAPOST sait déterminer *seul* une fonction affine.

# METAPOST sait déterminer *seul* une fonction affine.

```
z0=(0,0);z1=(5cm,0);z2=(5cm,5cm);z3=(0,5cm);
transform T;
z0 transformed T=1/4[z0,z1];
z1 transformed T=1/4[z1,z2];
z2 transformed T=1/4[z2,z3];
path carre;
carre=z0--z1--z2--z3--cycle;
fill carre withcolor 0.8white;
fill carre transformed T withcolor white;
fill carre transformed T transformed T withcolor 0.8white;
draw carre;
draw carre transformed T;
draw carre transformed T transformed T;
```

# METAPOST sait déterminer *seul* une fonction affine.

```
z0=(0,0);z1=(5cm,0);z2=(5cm,5cm);z3=(0,5cm);
transform T;
z0 transformed T=1/4[z0,z1];
z1 transformed T=1/4[z1,z2];
z2 transformed T=1/4[z2,z3];
path carre;
carre=z0--z1--z2--z3--cycle;
fill carre withcolor 0.8white;
fill carre transformed T withcolor white;
fill carre transformed T transformed T withcolor 0.8white;
draw carre;
draw carre transformed T;
draw carre transformed T transformed T;
```



# Type `color`

- Représentées par un triplet de composantes RVB.

```
1  color cyan;
2  cyan=(0,1,1);
```

# Type `color`

- Représentées par un triplet de composantes RVB.

```
1  color cyan;
2  cyan=(0,1,1);
```

- Certaines sont définies (`black`,`white`, `red`, `green`, `blue`).

# Type `color`

- Représentées par un triplet de composantes RVB.

```
1  color cyan;
2  cyan=(0,1,1);
```

- Certaines sont définies (`black`,`white`, `red`, `green`, `blue`).
- Opérations possibles : addition, soustraction, multiplication par un scalaire.

```
yellow=red+green;
gris=0.95white;
```

*Retour*

```
input LATEX; u:=1/2cm;

beginfig(1);
  path sqr;    sqr = unitsquare scaled u;
  for i=0 upto 10:
    label.bot(LATEX(""&decimal(i/10)&"") scaled
        0.7,((i+1/2)*u,0));
    label.lft(LATEX(""&decimal(i/10)&"") scaled
        0.7,(0,(i+1/2)*u));
    for j=0 upto 10:
      fill sqr shifted (i*u,j*u) withcolor (i
          *0.1,j*0.1,0);
      draw sqr shifted (i*u,j*u);
    endfor;
  endfor;
  label.bot("r",(6u,-2/3u));    label.lft("g",(-u
      ,6u));
  label.top("RGB(r,g,0)",(6u,11u));
endfig;
```

*Retour*

# Type `pen`

- Sont définies : `pencircle` (celle par défaut), `pensquare` ; `penrazor`.

# Type `pen`

- Sont définies : `pencircle` (celle par défaut), `pensquare` ; `penrazor`.
- `withpen...` applique la plume choisie au tracé concerné.

# Type `pen`

- Sont définies : `pencircle` (celle par défaut), `pensquare` ; `penrazor`.
- `withpen...` applique la plume choisie au tracé concerné.
- `pickup...` sélectionne la plume pour la suite des tracés.

# Type `pen`

- Sont définies : `pencircle` (celle par défaut), `pensquare` ; `penrazor`.
- `withpen...` applique la plume choisie au tracé concerné.
- `pickup...` sélectionne la plume pour la suite des tracés.
- Définir un plume :

```
1 pen calli;
  calli=pencircle xscaled 2bp yscaled 0.25bp
     rotated 60 withcolor red;
```

*Retour*

```
beginfig(1);
  drawarrow (0,0)--(50,0)..(20,30)..(80,50)
    withpen pencircle scaled 2;
  drawarrow (0,0)--(50,0)..(20,30)..(80,50)
    withcolor white;
  drawarrow (0,0)--(50,0)..(20,30)..(80,50)
    withcolor blue dashed evenly;
endfig;
```

*Retour*