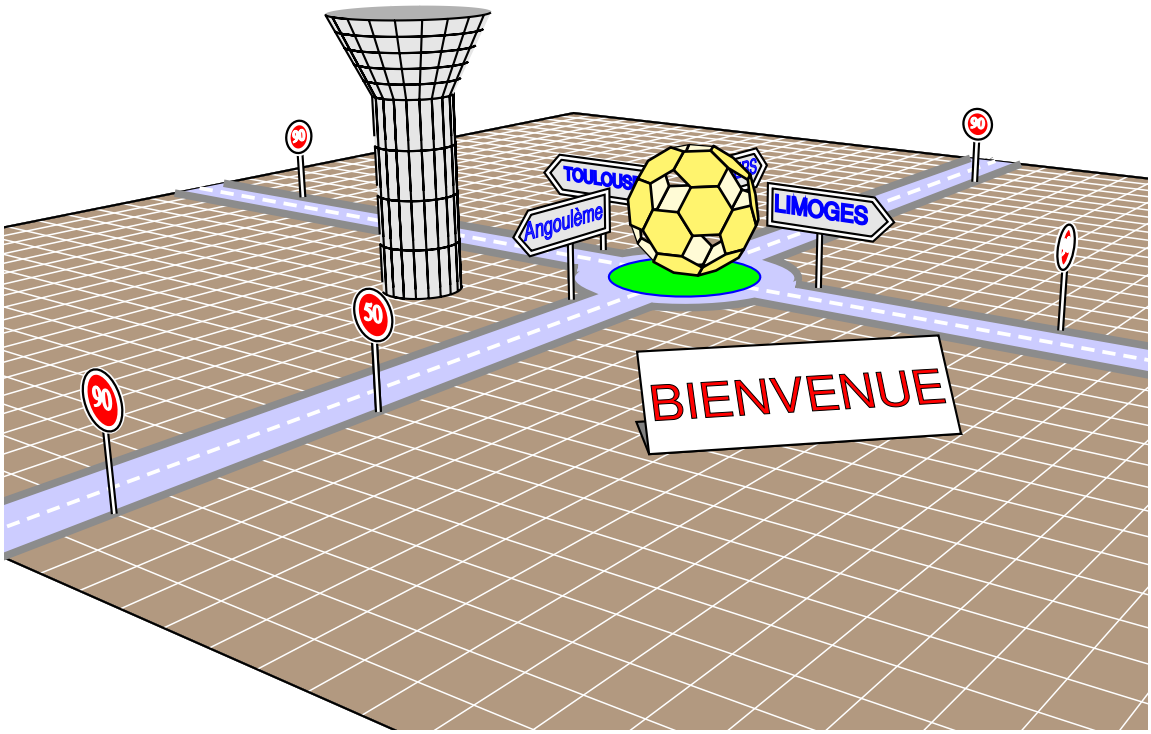
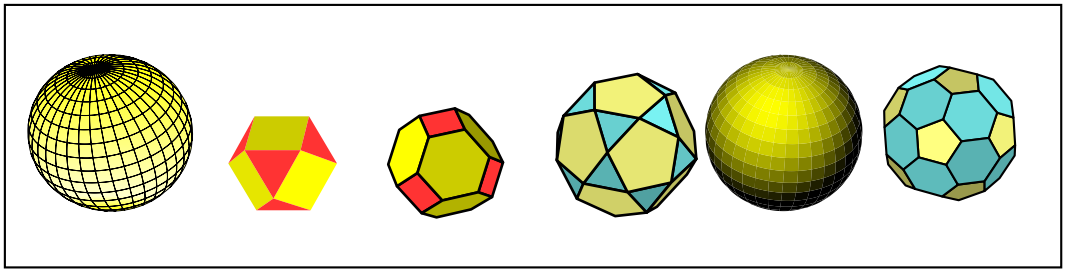
Essai de représentation en perspective conique d'un texte ou d'une figure géométrique

Documentation révisée le 13 août 2006, package version 0.18

13 août 2006





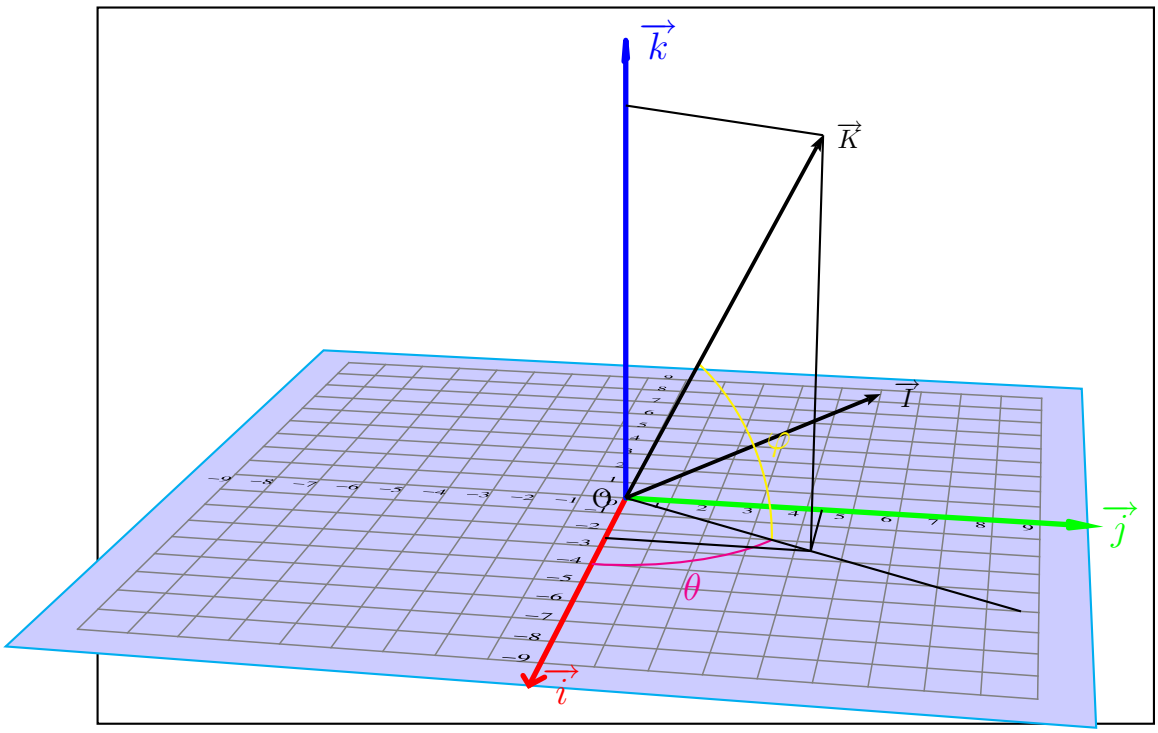
1 Présentation

`pst-3D` est un outil remarquable pour la représentation en 3D suivant la méthode de la projection parallèle. Dans ce package Timothy Van Zandt a eu l'idée très ingénieuse de construire la `matrice de transformation courante(CTM)` de `PostScript` à partir des formules de transformation permettant de représenter en projection parallèle dans une direction donnée par les coordonnées de [`viewpoint` = p_x p_y p_z] un plan défini par une normale à ce plan [`normal` = n_x n_y n_z] et un point origine appartenant à ce plan.

Cette étude reprend la même idée, mais elle se propose de représenter un texte ou une figure d'un plan dans la représentation en perspective conique.
La source de toute la partie concernant la perspective conique est le remarquable livre de Robert Dony : Graphisme scientifique sur micro-ordinateur, de la 2e à la 3e dimension, publié aux éditions Masson. La mise en perspective d'un texte doit beaucoup à l'étude du fichier fichier original de P. Kleiweg : `http://www.let.rug.nl/~kleiweg/postscript/circles.ps` et des contributions apportées par Arnaud Schmittbuhl et Jean-Paul Vignault lors d'échanges sur la liste de diffusion Syracuse :

`http://melusine.eu.org/cgi-bin/mailman/listinfo/syracuse`.

2 Formules de transformation

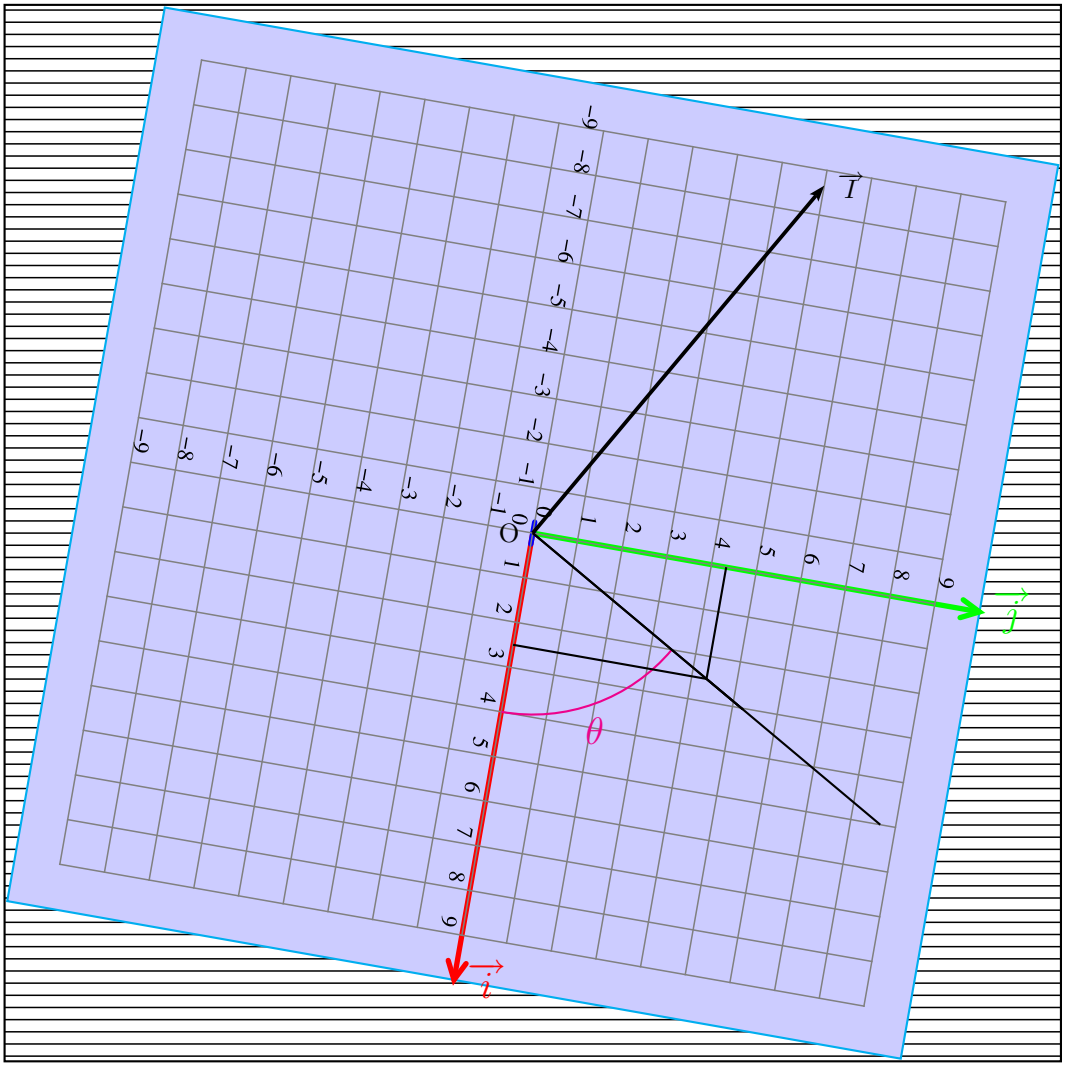


Établissons les formules permettant de passer du repère $(O, \vec{I}, \vec{J}, \vec{K})$ au repère $(O, \vec{i}, \vec{j}, \vec{k})$.

$\vec{K}$ représentera la normale au plan que l'on souhaite dessiner en perspective conique. Ce vecteur $\vec{K}$ est défini par la longitude θ et la latitude φ.

Dans $(O, \vec{i}, \vec{j}, \vec{k})$

$$\vec{K} = \begin{pmatrix} \cos\varphi\cos\theta \\ \cos\varphi\sin\theta \\ \sin\varphi \end{pmatrix}$$



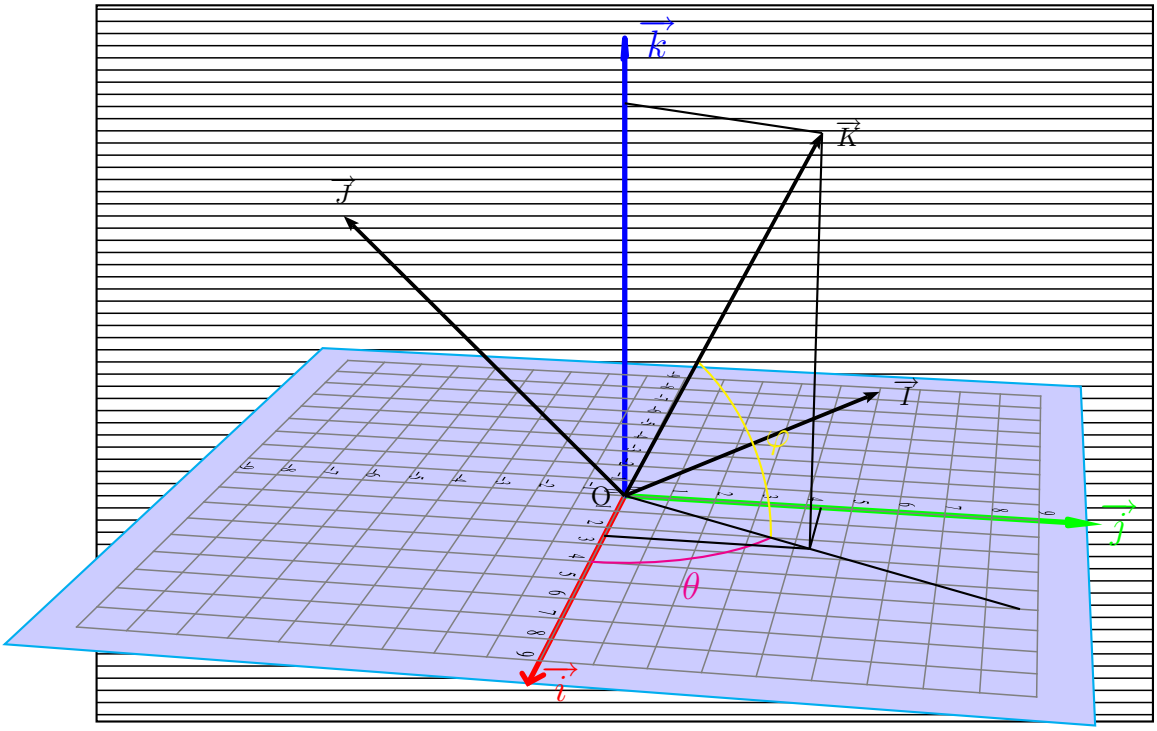
Il faut ensuite choisir les deux autres vecteurs de la base $(\vec{I}, \vec{J}, \vec{K})$. Je choisis de garder $\vec{I}$ dans le plan Oxy, orienté ainsi pour des raisons de cohérence, mais ce n'est pas sans inconvénient comme nous le verrons par la suite(voir).

Vu de dessus, dans le plan Oxy :

$$\vec{I} = \begin{pmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{pmatrix}$$

Il reste à trouver $\overrightarrow{J}$ pour que la base $(\overrightarrow{I}, \overrightarrow{J}, \overrightarrow{K})$ soit directe : $\overrightarrow{J} = \overrightarrow{K} \wedge \overrightarrow{I}$

$$\overrightarrow{J} = \begin{pmatrix} \cos\varphi\cos\theta \\ \cos\varphi\sin\theta \\ \sin\varphi \end{pmatrix} \wedge \begin{pmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{pmatrix} = \begin{pmatrix} -\sin\varphi\cos\theta \\ -\sin\varphi\sin\theta \\ \cos\varphi \end{pmatrix}$$



La matrice de transformation :

$$A = \begin{pmatrix} -\sin\theta & -\sin\varphi\cos\theta & \cos\varphi\cos\theta \\ \cos\theta & -\sin\varphi\sin\theta & \cos\varphi\sin\theta \\ 0 & \cos\varphi & \sin\varphi \end{pmatrix}$$

permet de déterminer les coordonnées (x, y, z) d'un point M si on connaît ses coordonnées (X, Y, Z) dans le repère $(O, \overrightarrow{I}, \overrightarrow{J}, \overrightarrow{K})$.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -\sin\theta & -\sin\varphi\cos\theta & \cos\varphi\cos\theta \\ \cos\theta & -\sin\varphi\sin\theta & \cos\varphi\sin\theta \\ 0 & \cos\varphi & \sin\varphi \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

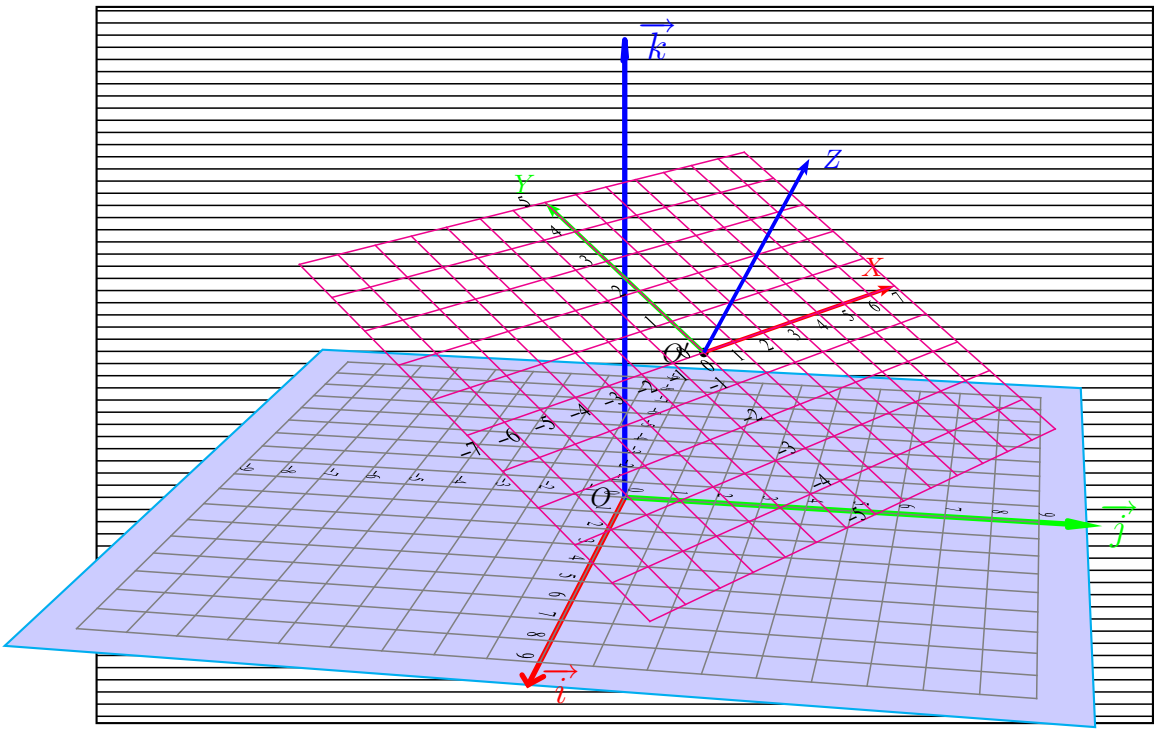
$$\begin{cases} x &= -X\sin\theta &- Y\sin\varphi\cos\theta &+ Z\cos\varphi\cos\theta \\ y &= X\cos\theta &- Y\sin\varphi\sin\theta &+ Z\cos\varphi\sin\theta \\ z &= 0 &+ Y\cos\varphi &+ Z\sin\varphi \end{cases}$$

Si l'on considère un point du plan appartenant au plan XOY :

$$\begin{cases} x &= -X\sin\theta &- Y\sin\varphi\cos\theta \\ y &= X\cos\theta &- Y\sin\varphi\sin\theta \\ z &= 0 &+ Y\cos\varphi \end{cases}$$

Et si maintenant, ce repère $OXYZ$ subit une translation en un point $O'(x_{O'}, y_{O'}, z_{O'})$

$$\left\{\begin{array}{rcrcrcl} x & = & -X\sin\theta & - & Y\sin\varphi\cos\theta & + & x_{O'} \\ y & = & X\cos\theta & - & Y\sin\varphi\sin\theta & + & y_{O'} \\ z & = & 0 & + & Y\cos\varphi & + & z_{O'} \end{array}\right.$$



3 Les commandes et les paramètres

3.1 Définir un plan

Un plan est défini par sa normale $\vec{n}$ et son origine O'. Le vecteur unitaire normal au plan est déterminé par deux angles en degrés :

- θ : longitude
- φ : latitude

Le paramètre permettant de fixer $\vec{n}$ est : `normale` $= \varphi\ \theta$

Attention de bien respecter l'ordre : `latitude longitude` et l'espace entre les deux valeurs.

La commande permettant de dessiner le plan est :

```
\planThreeDput[normale=60 60](2,2,4){données graphiques}
```

Les coordonnées de l'origine (x'_O, y'_O, z'_O) de O' du plan sont dans la parenthèse `(2,2,4)`.

Les données graphiques sont soit des codes `PostScript` soit des commandes pré-définies ou définies par l'utilisateur.

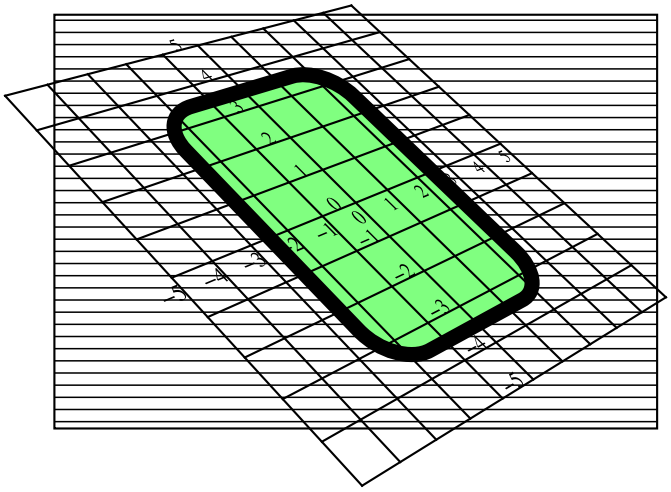
Quelques commandes pré-définies existent, comme :

- `\Grille(x1,y1)(x2,y2)` qui est une version simplifiée de la commande `psgrid` de `PSTricks` car elle ne permet pas les sous-divisions : il est évidemment possible de fabriquer une commande plus complète... Par contre elle prend les mêmes paramètres pour l'épaisseur du trait et la couleur :
  - `gridwidth`
  - `gridcolor`

  de plus on peut modifier la position des labels avec les options :
  - `xlabelsep=-0.5`
  - `ylabelsep=-0.5`

  Ce sont des valeurs en cm, celles qui sont indiquées sont les valeurs par défaut. Pour supprimer les labels il faut ajouter l'option `fontscale=0`.
- `\Rectangle(x1,y1)(x2,y2)` identique à `\psframe`, avec les mêmes paramètres sauf `framearc` et `framesep`
- `\cercle(x,y){R}` identique à `\pscircle` avec les mêmes options.
- `\ARC(x,y){R}{angle_début}{angle_fin}` identique à `\psarc` avec les mêmes options.
- `\ARCN(x,y){R}{angle_début}{angle_fin}` identique à `\psarcn` avec les mêmes options.
- `\PslineIIID(x1,y1)(x2,y2)(x3,y3)...(xn,yn)` identique à `\psline` avec quelques options en moins.

L'argument de la commande peut-être un code `PostScript` quelconque, comme par exemple celui-ci qui dessine un rectangle aux coins arrondis par un arc de cercle de rayon 1 cm :



```
\psset{THETA=10,PHI=30,Dobs=50,Decran=30,normale=60 60}
\planThreeDput[linewidth=0.2,fillstyle=solid,fillcolor=green!50]{%
-0.5 3.5 moveto
-2.5 3.5 -2.5 0 1 arcto
-2.5 -3.5 -0.5 -3.5 1 arcto
```

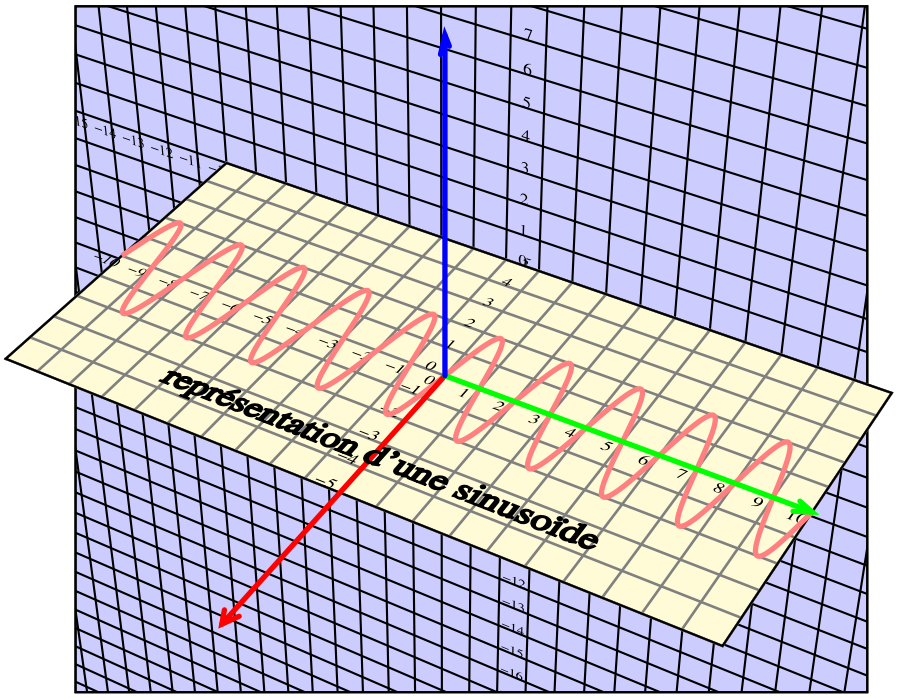
```
2.5 -3.5 2.5 0 1 arcto
2.5 3.5 -0.5 3.5 1 arcto
16 { pop } repeat
closepath}
\planThreeDput{\Grille(-5,-5)(5,5)}
```

Le code suivant permet de dessiner une sinusoïde :

```
\begin{pspicture}(-6,-7)(8,9)
\psframe(-6,-6)(8,7)
\psset{THETA=30,PHI=50,Dobs=20,Decran=15,normale=90 0}
\planThreeDput[gridwidth=0.05,gridcolor=gray]{\Grille(-10,-5)(10,5)}
\planThreeDput[linewidth=0.1,linecolor=red]{\sinus}
\axesIIID(10,10,10)%
\end{pspicture}
```

avec comme argument la commande `\sinus` :

```
\def\sinus{%
newpath
  -10 0 moveto
  /T 360 def % une période
  0 1 10 T mul { % representation de 10 périodes
  /t exch def
   /x t 10 mul 5 div 360 div def %  sur 10 divisions
   x 10 sub
   t sin 2 mul
   lineto} for
   }%
```

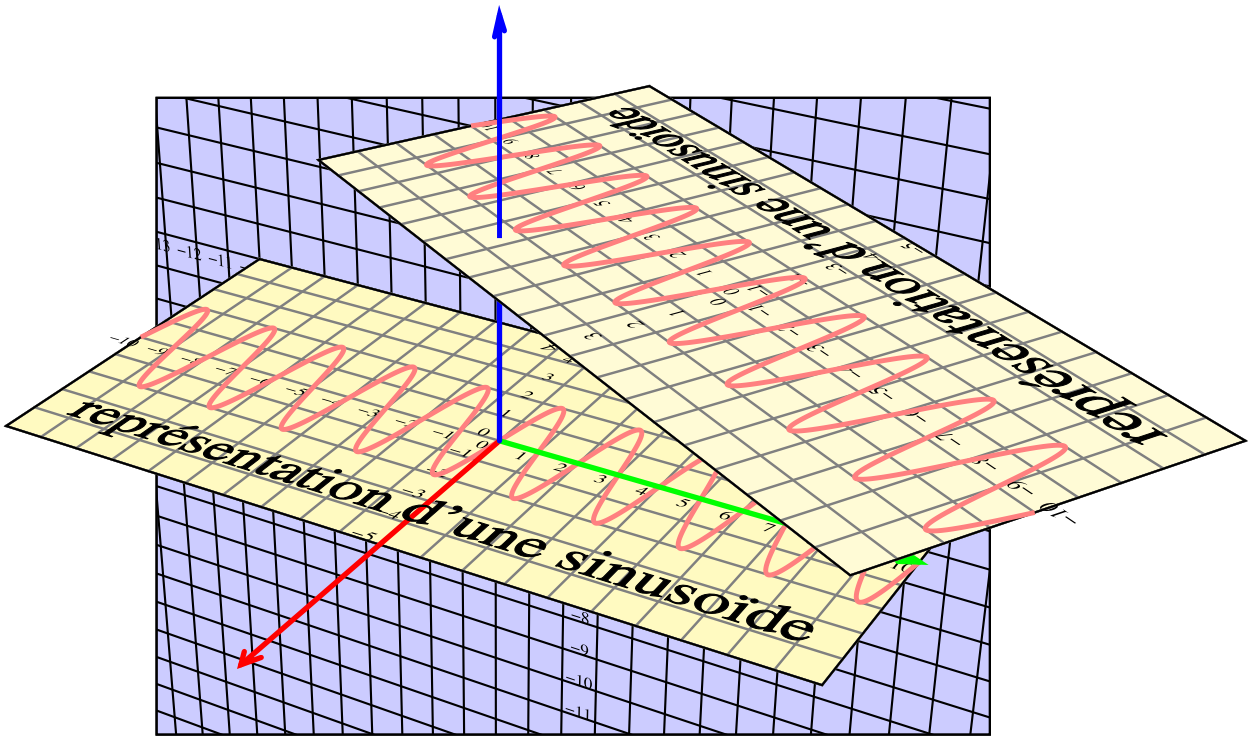


Revenons aux options disponibles avec la commande `\planThreeDput`

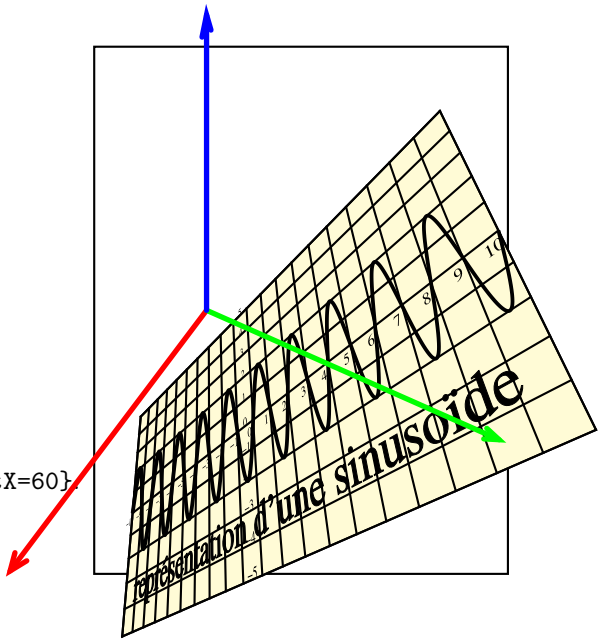
- Le plan défini par son origine et sa normale peut subir une translation et des rotations autour des axes Ox, Oy et Oz en utilisant les paramètres :
  - `translation=vx vy vz`
  - `RotX`, `RotY` et `RotZ`

  Les angles sont en degrés.
- L'unité de longueur par défaut est le centimètre, mais avec la commande de `PSTricks` : `unit=0.5` on peut par exemple choisir comme unité le demi-centimètre.

La sinusoïde précédente avec une translation de vecteur $\overrightarrow{V}(0,5,5)$ et rotation de 150° autour de l'axe Oz, avec les paramètres : `\psset{translation=0 5 5,RotZ=150}`.



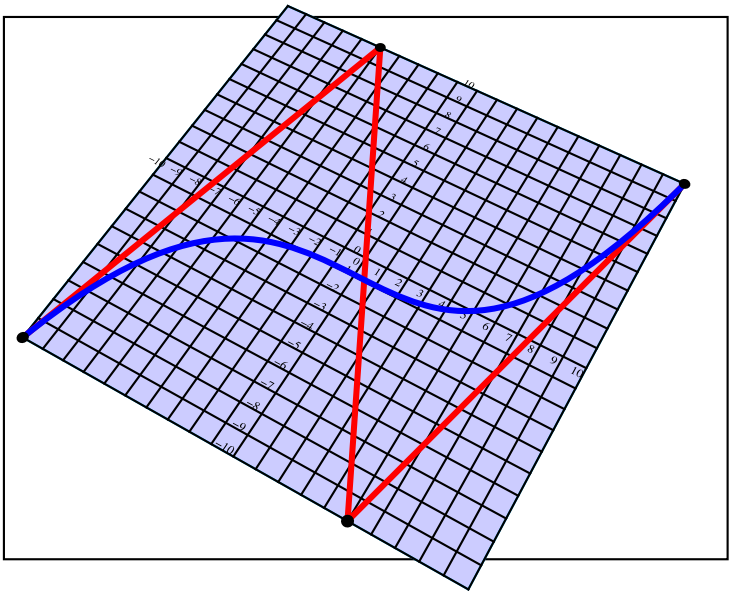
Un autre exemple, toujours avec la même sinusoïde ayant subit une translation de vecteur $\overrightarrow{V}(2, 3, -2)$ et des rotations de 30^o autour de l'axe Oz et de 60^o autour de l'axe Ox avec les paramètres : `\psset{translation=2 3 -2,RotZ=30,RotX=60}`.



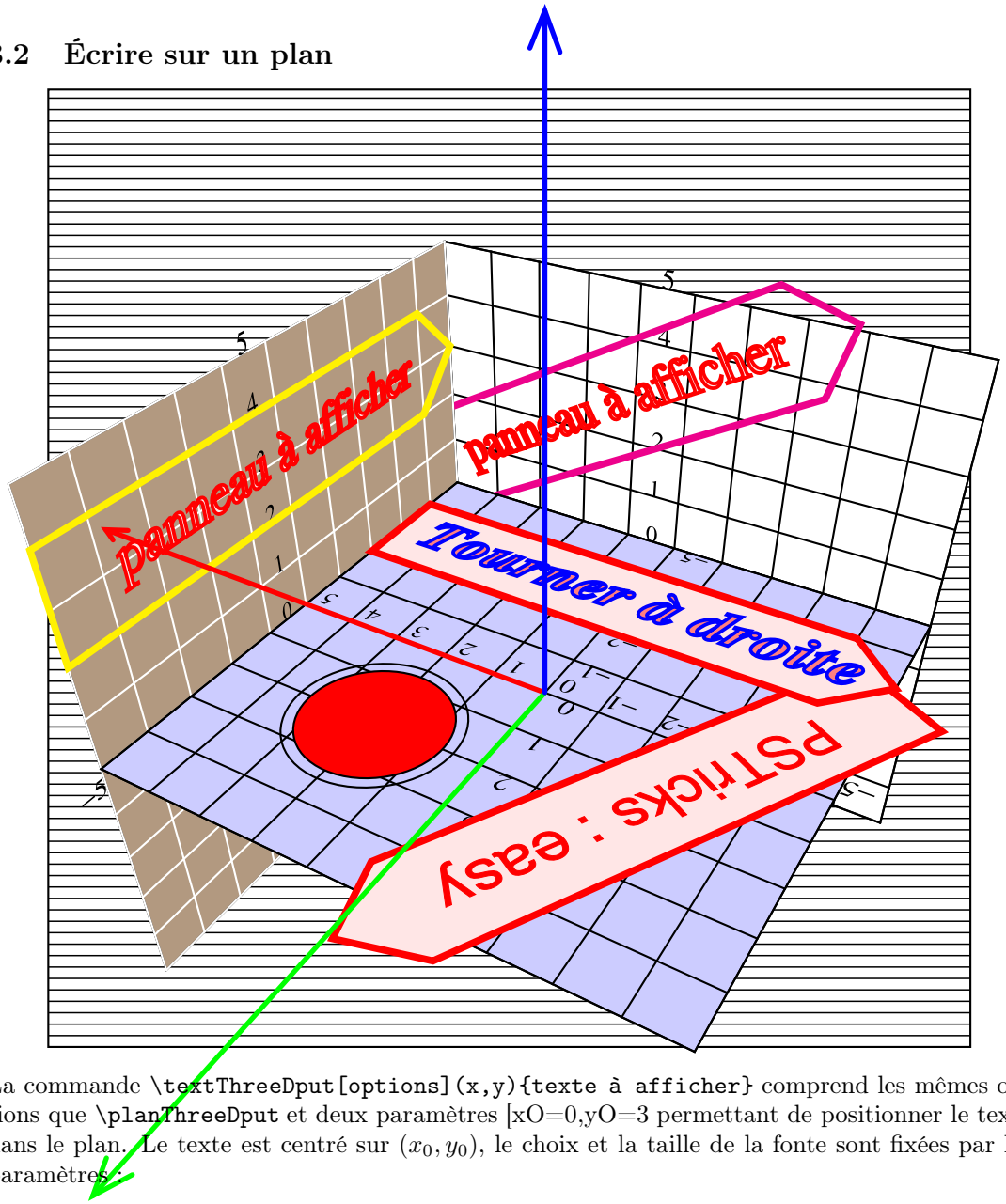
La commande `\planThreeDput` possède l'option `visibilty` qui par défaut est `true`. Avec cette option le plan n'est dessiné que s'il est visible. Si, malgré tout, on souhaite dessiner le plan et son contenu lorsqu'il est caché il suffit de l'indiquer avec `visibilty=false`.

Un exemple d'utilisation de `\PslineIIID` et de l'instruction `PostScript curveto` utilisée directement :

```
\PslineIIID[linecolor=red](-10,-10)(-5,10)(5,-10)(10,10)
\planThreeDput[linecolor=blue]{-10 -10 moveto -5 10 5 -10 10 10 curveto}
```



3.2 Écrire sur un plan



La commande `\textThreeDput[options](x,y){texte à afficher}` comprend les mêmes options que `\planThreeDput` et deux paramètres [xO=0,yO=3 permettant de positionner le texte dans le plan. Le texte est centré sur (x_0, y_0), le choix et la taille de la fonte sont fixées par les paramètres :

- `[fontscale=1]` en cm (1 cm valeur par défaut ;
- `[PSfont=Times-Roman]`, fonte par défaut.

La commande suivante :

```
\textThreeDput[xO=0,yO=3,linewidth=0.025,linecolor=blue,
               fillstyle=solid,fillcolor=red!50,%
               PSfont=NewBaskerville-BoldItalic,fontscale=1.3]{Tourner à droite}%
```

dessine le texte en NewBaskerville-BoldItalic, d'une taille de 1,3 cm, entré autour du point $(x = 0, y = 3)$, les caractères sont tracés en bleu et remplis avec une nuance de rouge. L'origine du plan et sa normale ayant été définies précédemment.

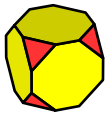
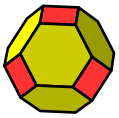
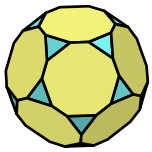
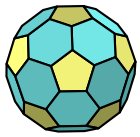
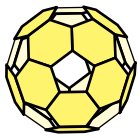
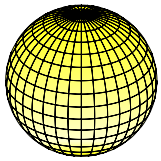
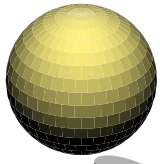
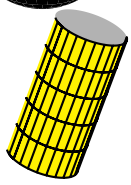
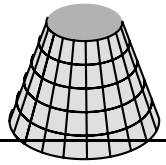
Pour améliorer la lisibilité d'un texte en petits caractères placer en option :

- `[linewidth=0,fillstyle=solid,fillcolor=couleur]`

on ne dessine pas ainsi le contour du caractère.

3.3 Placer des objets pré-définis

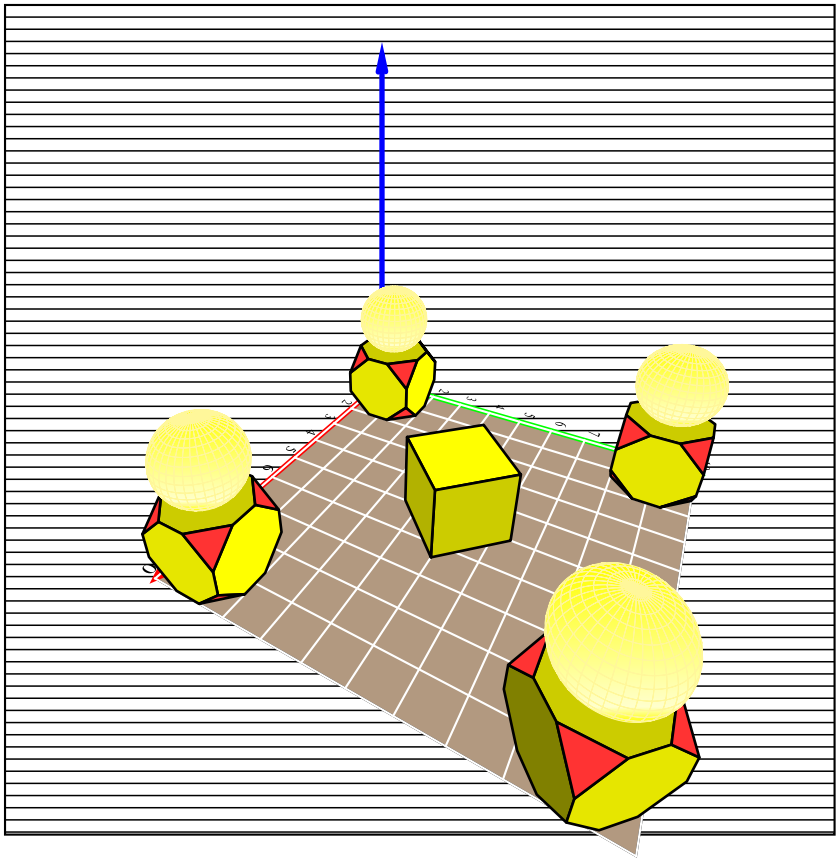
3.3.1 Solides qui sont pour l'instant prêts à l'emploi :

tétraèdre	`\psTetrahedron[options](xC,yC,zC)`	
cube	`\psCube[options](xC,yC,zC)`	
octaèdre	`\psOctahedral[options](xC,yC,zC)`	
dodécaèdre	`\psDodecahedron[options](xC,yC,zC)`	
icosaèdre	`\psIcosahedron[options](xC,yC,zC)`	
icosaèdre troué	`\psIcosahedronH[options](xC,yC,zC)`	
sphère (modèle 1)	`\psSphere[options](xC,yC,zC){Rayon}`	
sphère (modèle 2)	`\psSphereII[options](xC,yC,zC){Rayon}`	
cylindre	`\psCylindre[options](xC,yC,zC){Rayon}`	
cône	`\psCone[options](xC,yC,zC){Rayon}{Hauteur}`	

3.3.2 Les options

Tétraèdre, cube(parallélépipède), octaèdre, dodécaèdre et icosaèdre sont tronqués grâce au paramètre `d` qui représente la fraction $\frac{1}{d}$ de l'arête coupée à partir du sommet, **cette option ne fonctionna pas correctement avec un parallélépipède quelconque**. Avec une grande valeur, par exemple $d = 1e6$, on retrouve les solides classiques. Les options de rotations `RotX`, `RotY` et `RotZ` autour des axes restent valables pour tous les solides.

- arêtes du parallélépipède `A=2,B=A,C=A` (les valeurs sont égales pour le cube) ;
- rayon de la sphère circonscrite `radius=3` pour tétraèdre, octaèdre, dodécaèdre et icosaèdre ;
- ne pas dessiner les arêtes avec `arete=false` ;
- les couleurs des faces avec `ColorFace1= R G B` jusqu'à `ColorFace8` et des petites facettes du cube, du tétraèdre, du dodécaèdre et de l'octaèdre tronqué avec `ColorFacette=R G B` ;
- pour le modèle de la sphère 2, on peut décaler le tracé des arcs de méridiens avec `sepTheta=5`, le pigment de la sphère avec `hsbcolor=0.1667 1` qui sont les valeurs de la teinte et de la saturation dans le système de couleurs `HSB`, et la position de la source de lumière blanche avec : `thetaLight=70,phiLight=60,dLight=10` en coordonnées sphériques.
- Pour le cône, le paramètre `fracHeight=0.5` permet de choisir la fraction de hauteur à représenter (`fracHeight` < 1).
- Cylindre et cône ont deux paramètres de maillage `nF=20,nH=5` :
  - `nF=20` : nombre de facettes autour de l'axe ;
  - `nH=5` : nombre de facettes en hauteur.



L'arête du cube est par défaut `A=2` et le centre des cubes est placé à l'altitude $z = 1$, ils reposent donc tous sur le plan horizontal $z = 0$

```
\psCube(1,1,1)\psCube[RotZ=45](9,9,1)\psCube(1,9,1)\psCube(9,1,1)
\psCube[RotZ=135,RotX=90,d=1e6](5,5,1)
\psSphere[style=GradWhiteYellow](1,9,3){1}\psSphere[style=GradWhiteYellow](9,1,3){1}
\psSphere[style=GradWhiteYellow](9,9,3){1}\psSphere[style=GradWhiteYellow](1,1,3){1}
```

Il existe un modèle de cube avec projection de son ombre, voir les fichiers :

- `test_ombre_cube.tex` et les fichiers associés :
- `cube_object_shadow.tex` ;
- `cube_shadow.pro` ;

Dans ce cas, convertissez le fichier `ps` au format `pdf` pour activer la transparence.

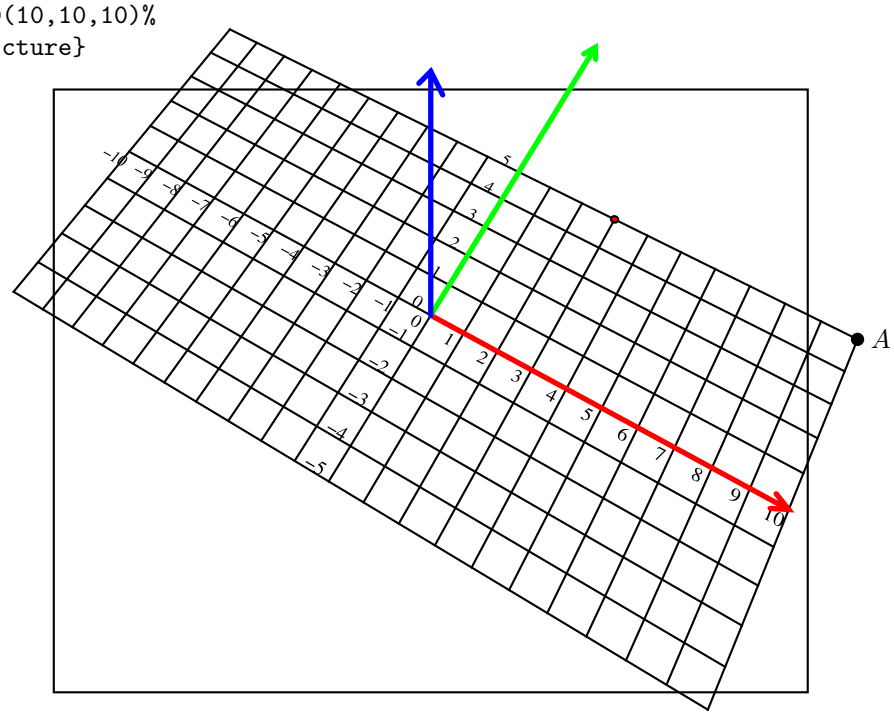
3.4 Placer un point sur un plan avec \NodeIIItoIID

```
\psset{THETA=-60,PHI=70,Dobs=20,Decran=10,normale=90 -90 }%
\planThreeDput[linewidth=0.05,linecolor=gray]{\Grille(-10,-5)(10,5)}%
```

```
\NodeIIItoIID[origine=0 0 0 ](10,5){A}%
\psdot[dotsize=5pt](A)%
\uput[r](A){$A$}
\planThreeDput[fillstyle=solid,fillcolor=red]{\cercle(3,5){0.1}}
\axesIIID(10,10,10)%
```

`\NodeIIItoIID[origine=x0 y0 z0,normale=phi theta](x,y){M}` place le nœud M dans le plan dont l'origine est $O'(x_0, y_0, z_0)$ et la normale $\overrightarrow{n}(\phi\ \theta)$, ce sont les mêmes options que pour `\planThreeDput`. On pourra donc aussi translater ce plan et le faire tourner. Par la suite les nœuds ainsi définis s'utilisent avec les commandes classiques de `PSTricks` et les propriétés du package `pst-node`.

```
\begin{pspicture}(-5,-5)(5,3)
\psframe(-5,-5)(5,3)
\psset{THETA=-60,PHI=70,Dobs=20,Decran=10,normale=90 -90 }%
\planThreeDput[linecolor=gray,fontscale=0.5]{\Grille(-10,-5)(10,5)}%
\NodeIIItoIID[origine=0 0 0 ](10,5){A}%
\psdot[dotsize=5pt](A)%
\uput[r](A){$A$}
\planThreeDput[fillstyle=solid,fillcolor=red]{\cercle(3,5){0.1}}
\axesIIID(10,10,10)%
\end{pspicture}
```



3.5 Placer un point dans l'espace avec `\pnodeXYZ(x,y,z){A}` ou `\pnodeSphericalCoor(R;θ;ϕ){X}`

Les coordonnées sont définies dans le repère absolu : donc ces points seront invariants par translation ou rotation : les options `translation`, `RotX`, `RotY` et `RotZ` sont sans effet. Par la suite les nœuds ainsi définis s'utilisent avec les commandes classiques de `PSTricks` et les propriétés du package `pst-node`.

3.6 Le repère dans le plan choisi

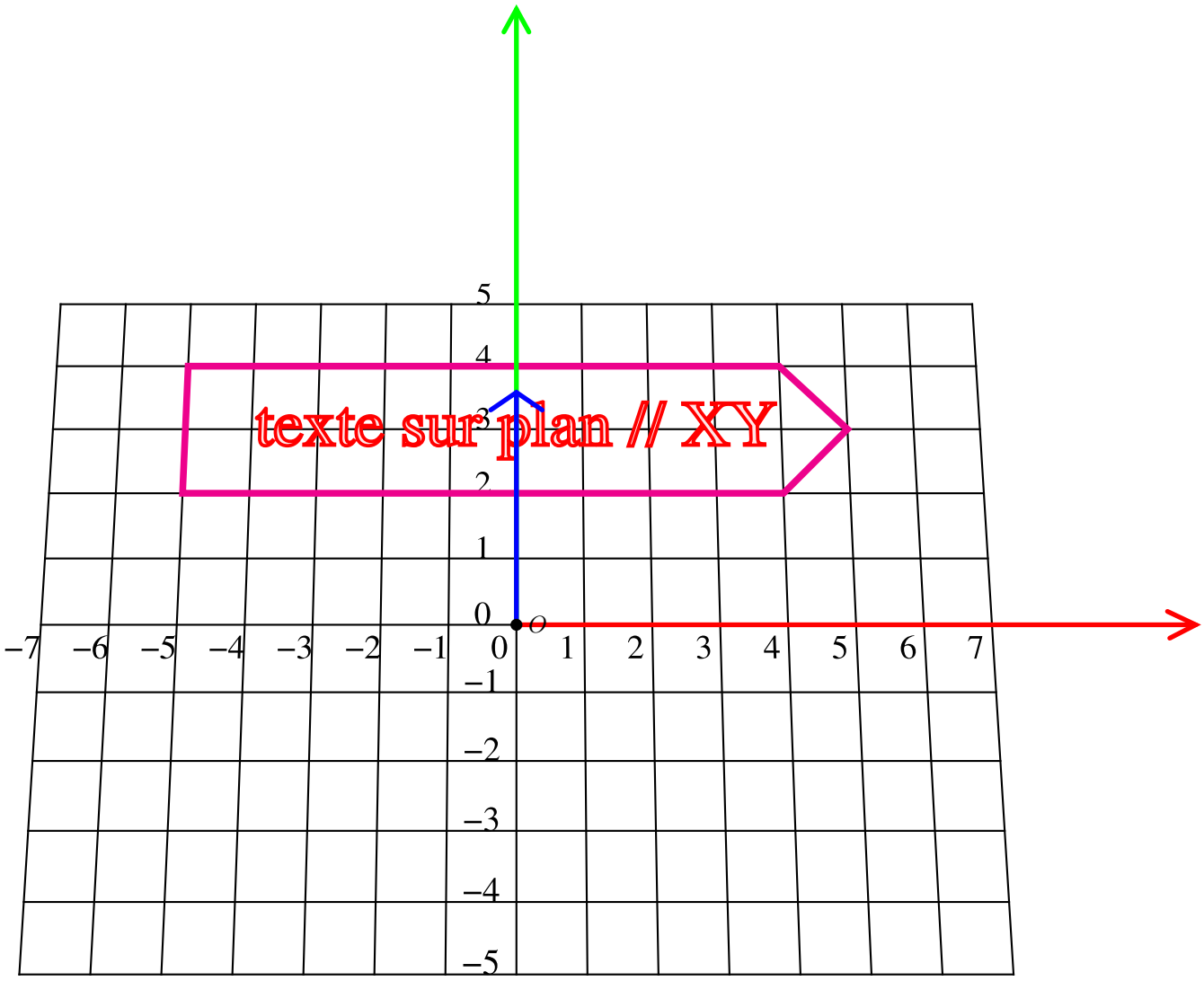
Placer $\overrightarrow{I}$ dans la plan horizontal avec l'orientation choisie permet de retrouver les plans Oxy, Oyz et Ozx à condition, bien sûr de prendre la normale correcte. En couleurs c'est le repère de référence (Ox, Oy, Oz).

Dans tous les cas, on peut s'aider de la commande `\Grille(-7,-5)(7,5)`, avec des valeurs différentes pour savoir comment se trouvent orientés les axes du plan OX et OY.

3.6.1 Le repère dans le plan Oxy

Le vecteur normal a pour coordonnées : `normale=90 -90`

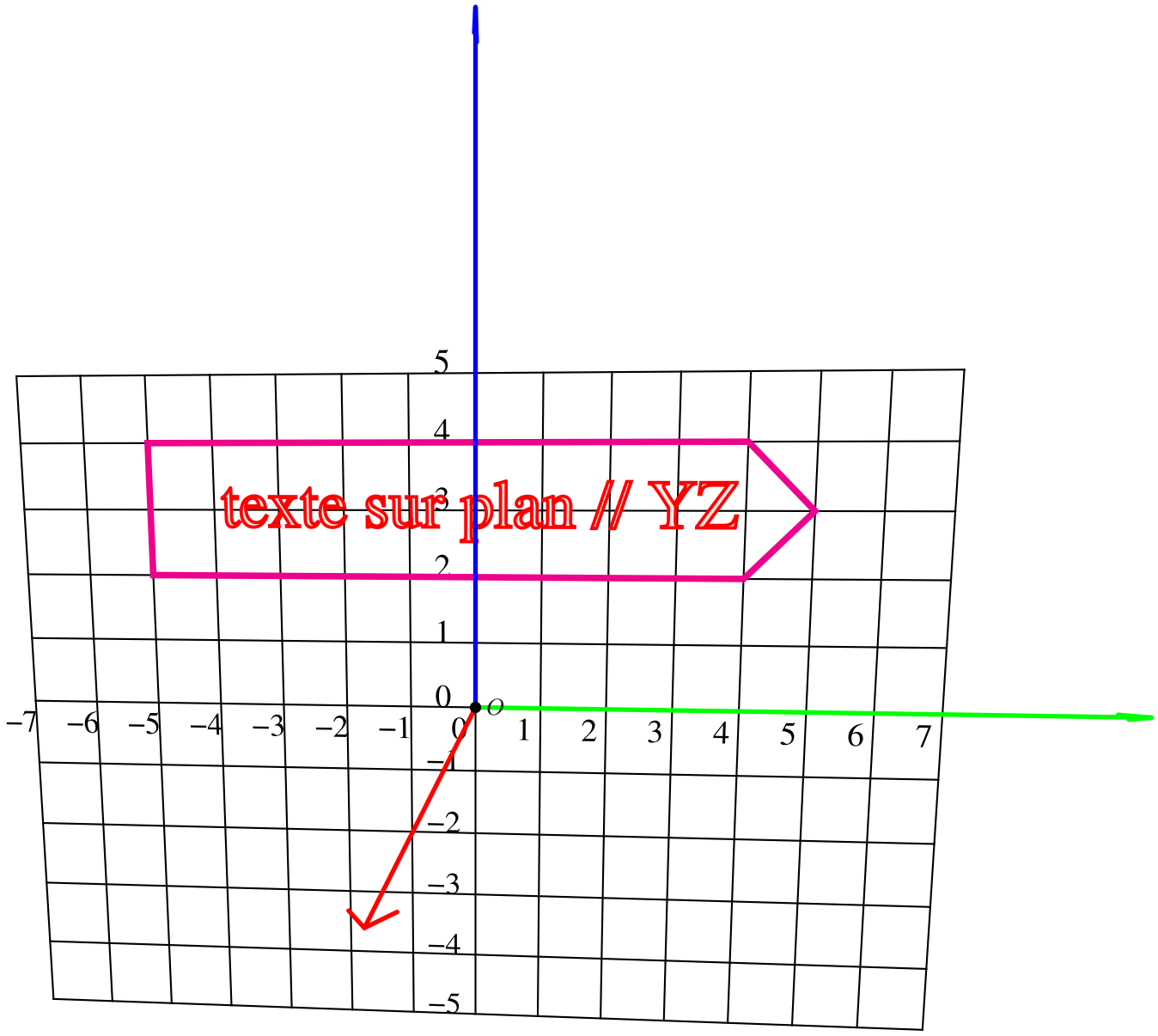
```
\psset{THETA=-90,PHI=80,Dobs=20,Decran=20,normale=90 -90}
\planThreeDput[linewidth=0.05,linecolor=green]{\Grille(-7,-5)(7,5)}
\planThreeDput[linewidth=0.1,linecolor=magenta]{\Fleche}
\textThreeDput[xO=0,yO=3,linewidth=0.05,linecolor=red,fillstyle=solid,%
    fillcolor=red!10]{texte sur plan // XY}
\axesIIID(10,10,10)%
\NodeIIItoIID[origine=0 0 0 ](0,0){O}%
\psdot[dotsize=5pt](O)%
\uput[r](O){$O$}
```



3.6.2 Le repère dans le plan Oyz

Le vecteur normal a pour coordonnées : `normale=0 0`

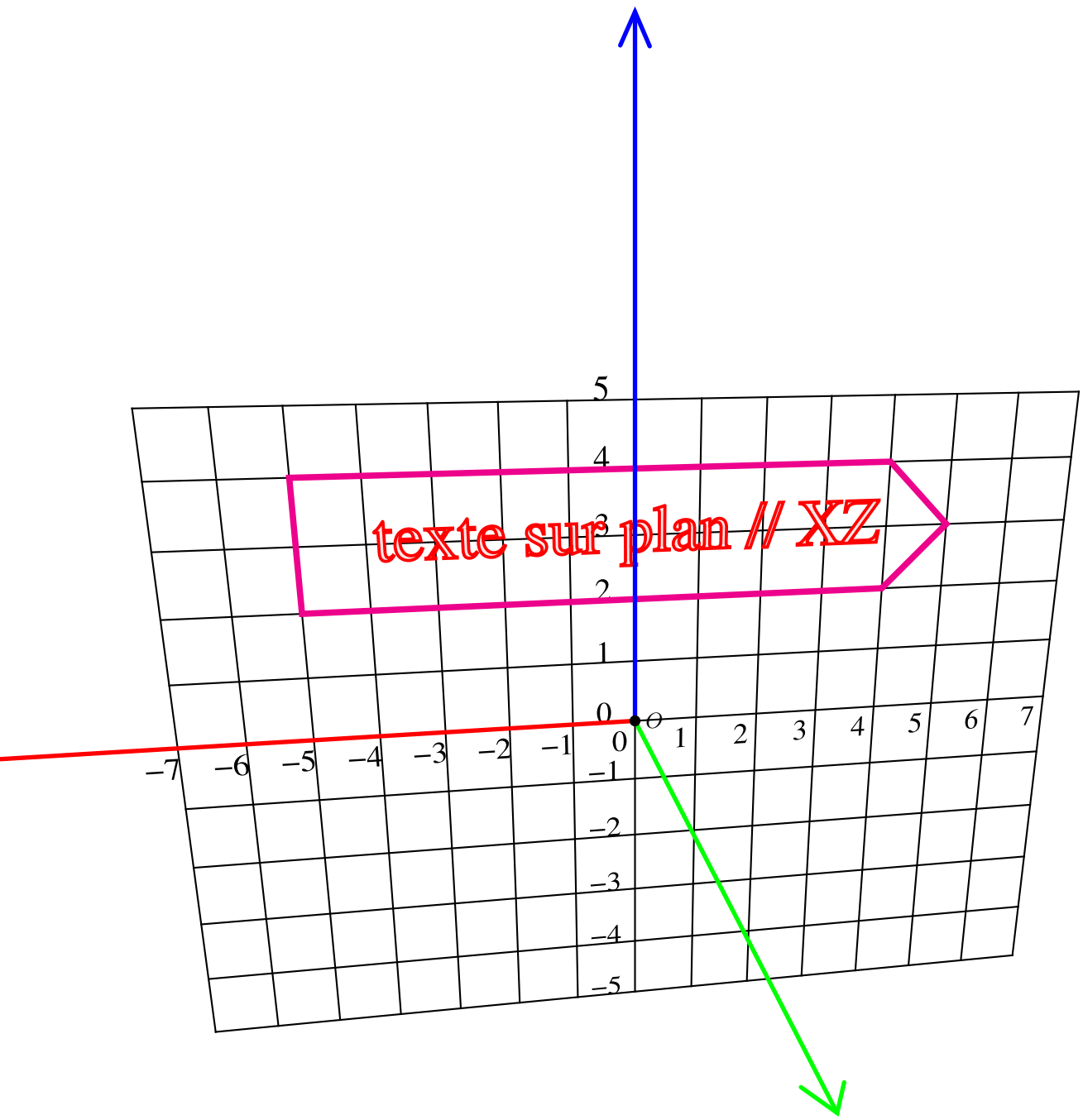
```
\psset{THETA=5,PHI=10,Dobs=20,Decran=20,normale=0 0}
\planThreeDput[linewidth=0.05,linecolor=green]{\Grille(-7,-5)(7,5)}
\planThreeDput[linewidth=0.1,linecolor=magenta]{\Fleche}
\textThreeDput[xO=0,yO=3,linewidth=0.05,linecolor=red,fillstyle=solid,
    fillcolor=red!10](0,0,0){texte sur plan // YZ}
\axesIIID(10,10,10)%
\NodeIIItoIID[origine=0 0 0 ](0,0){O}%
\psdot[dotsize=5pt](O)%
\uput[r](O){$O$}
```



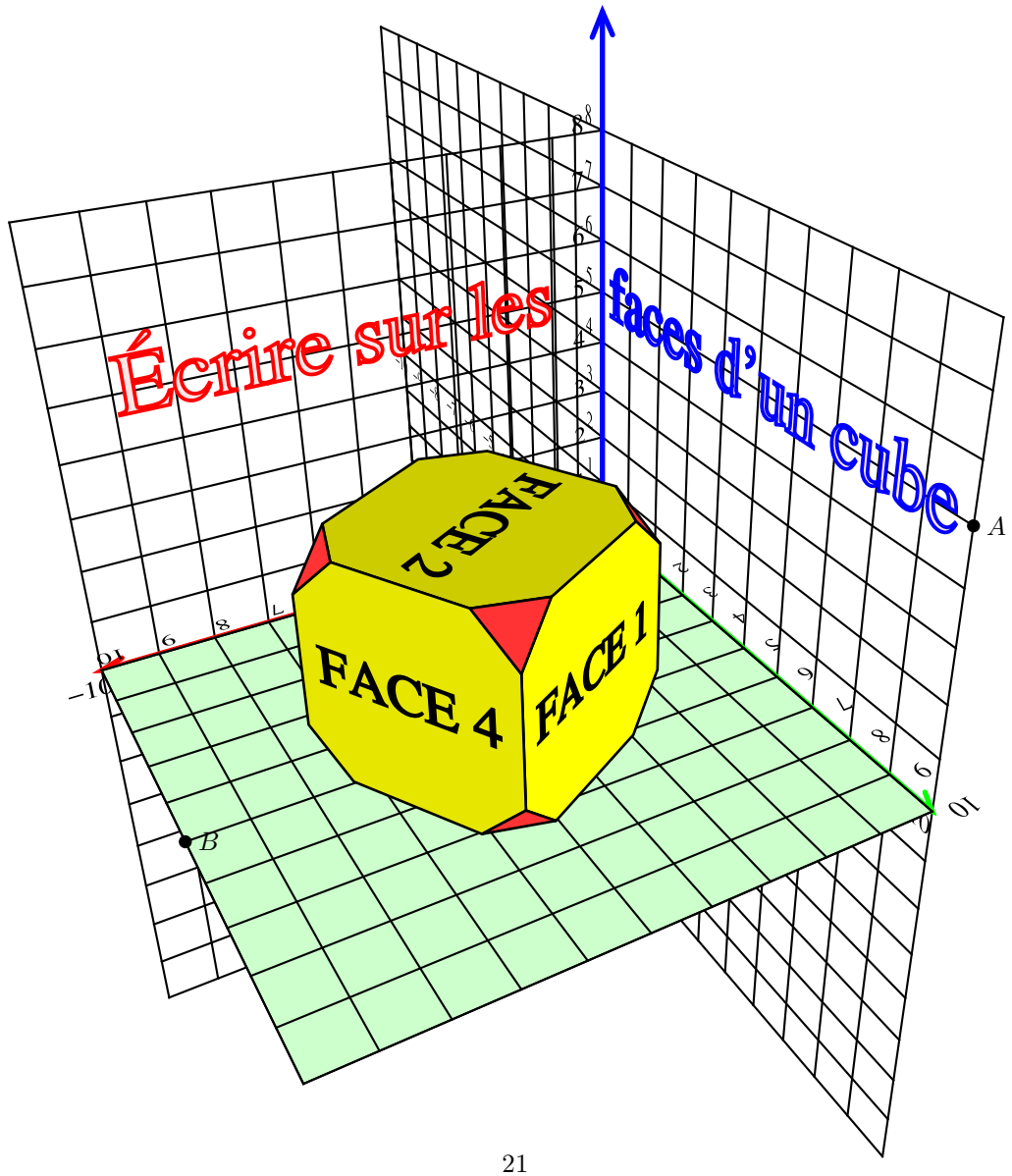
3.6.3 Le repère dans le plan Ozx

Le vecteur normal a pour coordonnées : `normale=0 90`

```
\psset{THETA=80,PHI=20,Dobs=20,Decran=20,normale=0 90}
\planThreeDput[linewidth=0.05,linecolor=green]{\Grille(-7,-5)(7,5)}
\planThreeDput[linewidth=0.1,linecolor=magenta]{\Fleche}
\textThreeDput[xO=0,yO=3,linewidth=0.05,linecolor=red,fillstyle=solid,%
    fillcolor=red!10](0,0,0){texte sur plan // XZ}
\axesIIID(10,10,10)%
\NodeIIItoIID[origine=0 0 0 ](0,0){O}%
\psdot[dotsize=5pt](O)%
\uput[r](O){$O$}
```

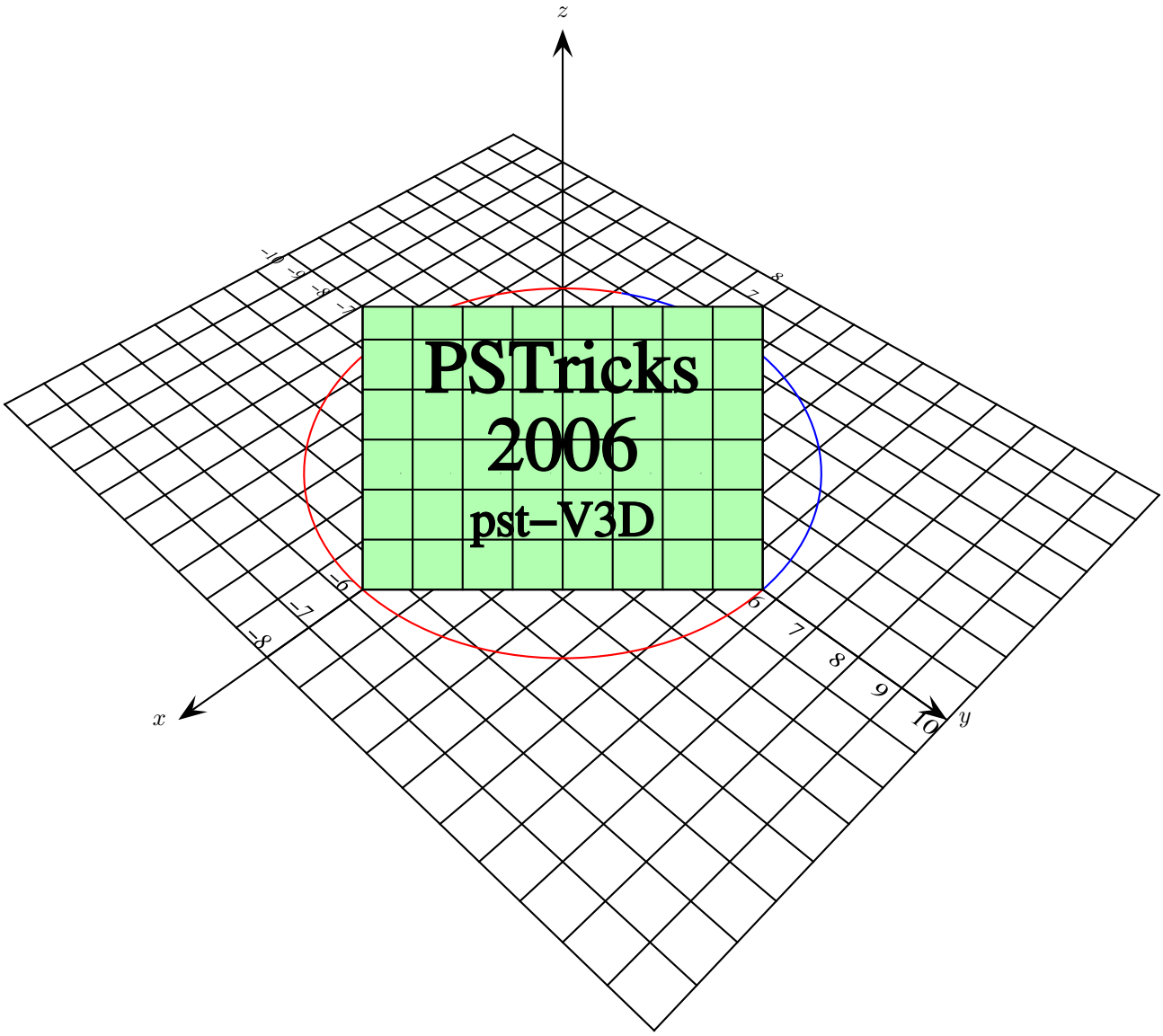

texte sur plan // XZ
5
4
3
2
1
0
−1
−2
−3
−4
−5
−7
−6
−5
−4
−3
−2
−1
0
1
2
3
4
5
6
7
O

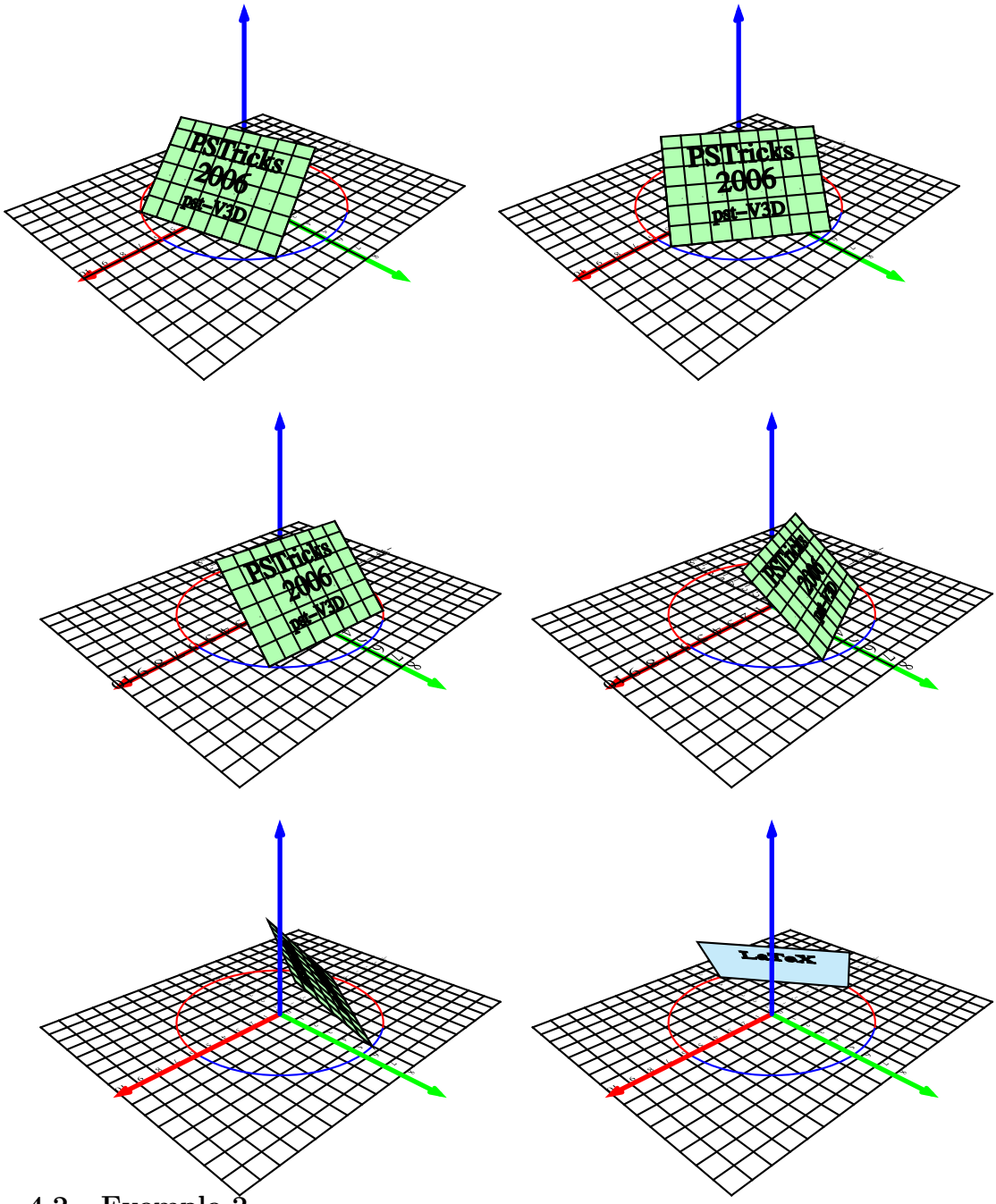
```
\psset{THETA=60,PHI=35,Dobs=30,Decran=20}
{\psset{fontscale=1.5,linewidth=0.05,linecolor=red,fillstyle=solid,fillcolor=red!10}
\planThreeDput[normale=0 0]{\Grille(-10,-8)(10,8)}
\planThreeDput[normale=0 -90]{\Grille(0,0)(10,8)}
\textThreeDput[xO=-5,yO=5,normale=0 90]{Écrire sur les}
\textThreeDput[linecolor=blue,xO=5,yO=5,normale=0 0]{faces d'un cube}}
\axesIIID(10,10,10)%
\planThreeDput[fillstyle=solid,fillcolor=green!20,normale=-90 0]{\Rectangle(0,0)(10,10)}
\planThreeDput[normale=-90 0]{\Grille(0,0)(10,10)}
\psCube[RotZ=-60,d=5,A=4](5,5,2)
\psset{translation=5 5 2,xO=0,yO=0,RotZ=-60,fillstyle=solid,fillcolor=black}
\textThreeDput[normale=0 90](0,2,0){FACE 1}%
\textThreeDput[normale=90 0](0,0,2){FACE 2}%
\textThreeDput[normale=0 -180](-2,0,0){FACE 3}%
\NodeIIItoIID[origine=0 0 0,normale=0 0 ](10,5){A}%
\psdot[dotsize=5pt](A)%
\uput[r](A){$A$}
\NodeIIItoIID[origine=0 0 0,,normale=90 -90 ](10,5){B}%
\psdot[dotsize=5pt](B)%
\uput[r](B){$B$}
```



4 Panneau double face

4.1 Exemple 1

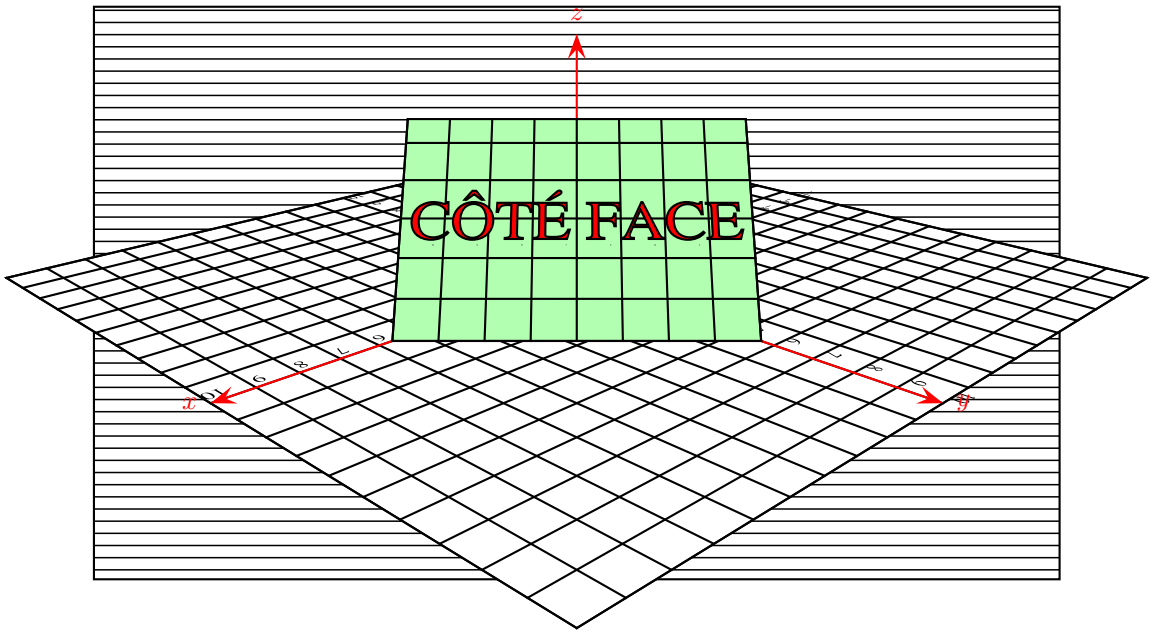


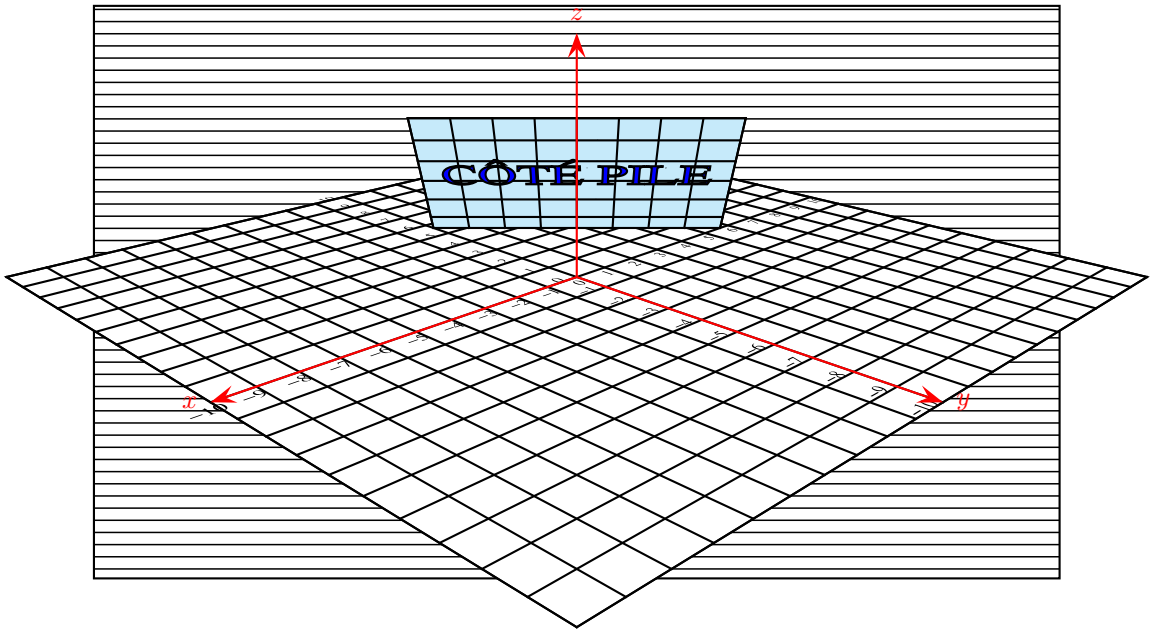


4.2 Exemple 2

Voici un panneau à deux faces, suivant l'angle de vue, c'est le côté pile ou le côté face qui sera visible. Il suffit que les normales aux deux faces soient opposées.

```
[normale=45 45]   % pile [normale=-45 225] % face
```


z
CÔTÉ FACE
x
y


z
CÔTÉ PILE
x
y

5 Les paramètres du point de vue

- `Dobs` : distance à laquelle est placé le point de vue ;
- `Decran` : distance à laquelle est placé l'écran où la scène est fixée ;
- `THETA` : longitude dans la direction du point de vue ;
- `PHI` : latitude dans la direction du point de vue.

6 Que reste-t-il à faire ? Avis aux amateurs...

1. Faire l'illustration du paragraphe précédent.
2. Afficher les faces des solides selon l'algorithme du peintre :
   - d'abord éliminer les faces non vues ;
   - puis les classer en fonction de leur distance au point de vue ;
   - commencer l'affichage par la face la plus éloignée en terminant par la plus proche.
3. Prévoir en option les différents types d'encodage en particulier `Symbol` qui permet d'afficher les caractères grecs.
4. Faire la liste de tous les solides souhaitables et les coder en postscript, comme `cube_object`, `icosahedral_object.tex`, `octahedral_object.tex` et `sphere_object.tex`.
5. Créer un fichier `pst-V3D.pro` dans lequel on placera toutes les routines `PostScript`.
6. Et bien d'autres choses encore...

13 août 2006 : Manuel Luque