

Le fichier povray.mod 0.96

Pour TeXgraph 1.96

29 juillet 2019

Résumé

Description des macros de povray.mod et povray.mod, pour l'export d'une scène 3D vers povray.

Table des matières

| | | | |
|---|----------|------------------------|----------|
| 1 Introduction | 1 | 2.9 povDroite | 5 |
| 1.1 Utilisation du modèle povray.mod | 1 | 2.10 povCurve | 5 |
| 1.2 Construction de la scène 3D | 1 | 2.11 povSurf | 5 |
| 1.3 Paramètres globaux | 2 | 2.12 povImplicit | 5 |
| 1.4 Structure des fichiers exportés *.pov | 2 | 2.13 povAxes | 6 |
| 1.4.1 Commentaires | 2 | 2.14 povAngleD | 6 |
| 1.4.2 Ajouter du code povray | 2 | 2.15 povDot | 6 |
| 2 Macros de construction | 2 | 2.16 povLabel | 6 |
| 2.1 povCylinder | 2 | 2.17 povArc | 7 |
| 2.2 povCone | 3 | 2.18 povCercle | 7 |
| 2.3 povTorus | 3 | 3 Exemples | 7 |
| 2.4 povSphere | 3 | 3.1 Section d'un tore | 7 |
| 2.5 povFacet | 3 | 3.2 Arcs et segments | 8 |
| 2.6 povLine | 4 | 3.3 Une hyperbole | 9 |
| 2.7 povPlan | 4 | 3.4 Fenêtre de Viviani | 10 |
| 2.8 povPlanEqn | 5 | 3.5 L'horoptère | 11 |
| | | 3.6 Courbes de niveaux | 12 |
| | | 3.7 Rendu personnalisé | 13 |

1 Introduction

1.1 Utilisation du modèle povray.mod

Ce fichier modèle permet de charger en mémoire les macros du fichier *povray.mod* et de créer un menu sous forme de boutons : un permet de modifier les paramètres à passer à povray, un autre permet de lancer l'export de la scène 3D dans un fichier au format *.pov, qui est le format du logiciel povray¹. Sous linux, povray est lancé directement en ligne de commande par TeXgraph dès la fin de l'export et l'on voit l'image se créer, celle-ci est enregistrée par povray dans le **répertoire courant**. Sous windows, il vous faudra lancer l'interface graphique de povray pour charger le fichier exporté et l'exécuter, sauf si vous avez installé l'exécutable *povray.exe* sans GUI pour windows. Sous windows, au premier chargement du modèle, le logiciel demande à l'utilisateur de définir le chemin d'accès au fichier *include* de povray, celui-ci sera enregistré dans un fichier nommé *povray.cfg* dans le dossier TeXgraphMac/povray, ce fichier sera ensuite lu à chaque chargement du modèle. Par défaut le fichier *povray.cfg* n'existe pas, et le chemin par défaut est réglé à `c:\Program Files\povray\include`.

Les paramètres peuvent être également modifiés en éditant la macro *povparam*, et on peut lancer l'export directement dans la ligne de commande en tapant :

```
povray(" <nom sans extension>", [options])
```

où les options peuvent être :

- `imagesize := < nombre >`. Facteur d'échelle pour la taille de l'image (1 par défaut).
- `defaultshadow := < 0/1 >`. Indique la prise en compte globale de l'ombre ou non (1 par défaut).

1. <http://www.povray.org/>.

1.2 Construction de la scène 3D

Celle-ci est tout à fait identique à la construction de la scène 3D décrite dans le fichier d'aide *scene3D.pdf* avec la commande **Build3D**. D'ailleurs les macros du fichier *povray.mac* sont des redéfinitions des macros du fichier *scene3D.mac*, et elles utilisent celles-ci pour l'affichage écran. La différence est que l'on construit la scène avec la macro **povScene** (et non plus **Build3D**) comme ceci :

```
[...
povScene( element1, element2, ..., elementN ),
Display3D(),
...]
```

Les différents éléments peuvent être des axes, des surfaces, des courbes, ...etc. Il y a une macro pour définir chaque type d'élément, elles auront le préfixe **pov** (comme *povAxes*, *povSurf*, ...) et non plus le préfixe **bd** comme c'était le cas dans *scene3D.mac*.

On peut regrouper deux éléments (ou plus) en un seul à condition de les séparer par la constante **sep3D**, comme ceci :

```
[element1, sep3D, element2, sep3D, element3]
```

Le fait que la macro **povScene** appelle la commande **Build3D**, assure que les autres exports restent valables, en particulier les exports *obj*, *geom* et *jvx*.

Important : le contenu de la macro *povScene* est exécuté deux fois, une fois en vue de l'export *povray*, et une autre fois pour le rendu écran. Cela a pour conséquence que si la macro *povScene* modifie le contenu d'une variable, elle doit lui restituer sa valeur initiale à la fin pour que la deuxième exécution soit identique à la première.

1.3 Paramètres globaux

Les paramètres globaux que l'on peut utiliser dans l'élément graphique où on définit la scène sont au nombre de trois :

- **imagesize** := < nombre >. Facteur d'échelle pour la taille de l'image (1 par défaut).
- **defaultshadow** := < 0/1 >. Indique la prise en compte de l'ombre ou non dans le rendu *povray* (1 par défaut).

1.4 Structure des fichiers exportés *.pov

Ceux-ci sont structurés en trois parties :

1. Partie 1 : un préambule où on définit le fond, la caméra, la lumière.
2. Partie 2 : partie déclarative où tous les objets qui doivent être rendus sont définis, cette partie déclarative est assez peu modifiable à la main.
3. Partie 3 : partie où chaque objet est rendu (sauf si l'objet est créé avec l'option *render:=0*), c'est ici que l'on peut modifier les paramètres du rendu : *pigment*, *texture*, *finish*...

1.4.1 Commentaires

L'utilisateur peut mettre des commentaires grâce à la macro :

```
povComment(<chaîne>)
```

Ceci qui permet de donner par exemple un nom à l'élément graphique qui va suivre, pour le repérer facilement dans le fichier *povray* car ce commentaire apparaîtra dans les parties 2 et 3.

1.4.2 Ajouter du code povray

L'utilisateur peut ajouter du code *povray* dans le fichier exporté grâce à la macro :

```
povSpecial(<chaîne>)
```

L'argument est évalué sous forme de chaîne de caractères, celle-ci est écrite dans le fichier exporté, dans la partie 3.

2 Macros de construction

2.1 povCylinder

```
povCylinder( sommet A, vecteur V, rayon R, [options])
```

Définit un cylindre de sommet A, d'axe porté par le vecteur V et de rayon égal à R. Options de *povCylinder* :

- **clip** := < -1/0/1 >. Indique si les facettes doivent être clippées par la fenêtre définie par l'option *clipwin* lorsque **clip**=1, ou bien par le plan défini par l'option *clipwin* lorsque **clip**=-1 (**clip**=0 par défaut).

- `clipwin := < [M(xinf,yinf,zinf), M(xsup,yup,zsup)] >`. Définit la fenêtre 3D pour un éventuel clipping lorsque `clip=1`, la fenêtre est alors donnée par sa grande diagonale : `[M(xinf,yinf,zinf), M(xsup,yup,zsup)]` (c'est la fenêtre courante par défaut). Mais lorsque `clip=-1` l'option `clipwin` est interprétée comme un plan : `[point3D, vecteur normal]`.
- `color := < couleur >`. Choix de la couleur (white par défaut).
- `render := < 0/1 >`. Indique si l'élément doit être rendu ou non (1 par défaut).
- `shadow := < 0/1 >`. Indique si l'élément a une ombre ou non (1 par défaut).
- `hollow := < 0/1 >`. Indique si le cylindre est creux ou non (1 par défaut).
- `smooth := < 0/1 >`. Indique si l'algorithme de Gouraud (lissage des facettes) doit être utilisé ou non lors de l'exportation pstricks ou eps (0 par défaut). Le lissage est automatique dans l'export povray.
- `opacity := < nombre entre 0 et 1 >`. Valeur de l'opacité (1 par défaut), permet d'introduire la transparence lorsque l'opacité est strictement inférieure à 1.
- `matrix := < matrice 3d >`. Permet de définir une matrice de transformation qui sera appliquée aux facettes (l'identité par défaut). La transformation s'effectue avant l'éventuel clipping.
- `nbfacet := < nombre de facettes >`. Définit le nombre de facettes pour l'affichage écran (35 par défaut).
- `border := < 0/1 >`. Indique si le contour doit être dessiné ou non (0 par défaut).
- `bordercolor := < couleur >`. Indique la couleur du contour (identique à `color` par défaut).

2.2 povCone

`povCone(sommet A, vecteur V, rayon R, [options])`

Définit un cône de sommet A, d'axe porté par le vecteur V et de rayon à la base égal à R. Les options sont identiques à `povCylinder`.

2.3 povTorus

`povTorus(centre A, R, r, Normale, [options])`

Dessine un tore de centre A, de grand rayon R et de petit rayon r, le paramètre *Normale* est un vecteur 3D qui représente un vecteur normal au plan du tore. Les options :

- `clip := < -1/0/1 >`. Indique si les facettes doivent être clippées par la fenêtre définie par l'option `clipwin` lorsque `clip=1`, ou bien par le plan défini par l'option `clipwin` lorsque `clip=-1` (`clip=0` par défaut).
- `clipwin := < [M(xinf,yinf,zinf), M(xsup,yup,zsup)] >`. Définit la fenêtre 3D pour un éventuel clipping lorsque `clip=1`, la fenêtre est alors donnée par sa grande diagonale : `[M(xinf,yinf,zinf), M(xsup,yup,zsup)]` (c'est la fenêtre courante par défaut). Mais lorsque `clip=-1` l'option `clipwin` est interprétée comme un plan : `[point3D, vecteur normal]`.
- `color := < couleur >`. Choix de la couleur (white par défaut).
- `render := < 0/1 >`. Indique si l'élément doit être rendu ou non (1 par défaut).
- `shadow := < 0/1 >`. Indique si l'élément a une ombre ou non (1 par défaut).
- `hollow := < 0/1 >`. Indique si le cylindre est creux ou non (1 par défaut).
- `smooth := < 0/1 >`. Indique si l'algorithme de Gouraud (lissage des facettes) doit être utilisé ou non lors de l'exportation pstricks ou eps (0 par défaut). Le lissage est automatique dans l'export povray.
- `opacity := < nombre entre 0 et 1 >`. Valeur de l'opacité (1 par défaut), permet d'introduire la transparence lorsque l'opacité est strictement inférieure à 1.
- `matrix := < matrice 3d >`. Permet de définir une matrice de transformation qui sera appliquée aux facettes (l'identité par défaut). La transformation s'effectue avant l'éventuel clipping.
- `grid := < [nb méridiens, nb parallèles] >`. Pour l'affichage écran, [40,25] par défaut.

2.4 povSphere

`povSphere(centre A, rayon R, [options])`

Définit une sphère de centre A, et de rayon égal à R. Les options sont celles de `povTorus` plus :

- `border := < 0/1 >`. Indique si le contour doit être dessiné ou non (0 par défaut).
- `bordercolor := < couleur >`. Indique la couleur du contour (identique à `color` par défaut).

2.5 povFacet

`povFacet(liste facettes, [options])`

Définit une liste de facettes. Options de `povFacet` :

- `backculling := < 0/1 >`. Indique si les facettes cachées doivent être éliminées ou non (0 par défaut).
- `clip := < -1/0/1 >`. Indique si les facettes doivent être clippées par la fenêtre définie par l'option `clipwin` lorsque `clip=1`, ou bien par le plan défini par l'option `clipwin` lorsque `clip=-1` (`clip=0` par défaut).

- `clipwin := < [M(xinf,yinf,zinf), M(xsup,yup,zsup)] >`. Définit la fenêtre 3D pour un éventuel clipping lorsque `clip=1`, la fenêtre est alors donnée par sa grande diagonale : `[M(xinf,yinf,zinf), M(xsup,yup,zsup)]` (c'est la fenêtre courante par défaut). Mais lorsque `clip=-1` l'option `clipwin` est interprétée comme un plan : `[point3D, vecteur normal]`.
- `color := < couleur >`. Choix de la couleur (white par défaut).
- `contrast := < nombre ≥ 0 >`. Uniquement pour le rendu écran (non pris en compte par povray), permet d'accentuer le contraste entre les différentes facettes. Avec la valeur 0 toutes les facettes ont la même couleur, le contraste « normal » est de 1 (valeur par défaut).
- `render := < 0/1 >`. Indique si l'élément doit être rendu ou non (1 par défaut).
- `shadow := < 0/1 >`. Indique si l'élément a une ombre ou non (1 par défaut).
- `contrast := < nombre positif >`. Le contraste normal a la valeur 1 (valeur par défaut), un contraste nul signifie que la couleur est unie (sans effet sur l'export povray).
- `smooth := < 0/1 >`. Indique si l'algorithme de Gouraud (lissage des facettes) doit être utilisé ou non lors de l'exportation pstricks ou eps ou povray (0 par défaut).
- `opacity := < nombre entre 0 et 1 >`. Valeur de l'opacité (1 par défaut), permet d'introduire la transparence lorsque l'opacité est strictement inférieure à 1.
- `matrix := < matrice 3d >`. Permet de définir une matrice de transformation qui sera appliquée aux facettes (l'identité par défaut). La transformation s'effectue avant l'éventuel clipping.
- `twoside := < 0/1 >`. Indique si on doit distinguer ou non le devant-derrrière des facettes. Dans l'affirmative, les deux côtés n'ont pas la même couleur (1 par défaut). Sans effet sur l'export povray.
- `triangular := < 0/1 >`. Permet de trianguler ou non les facettes (0 par défaut). Sans effet sur l'export povray.
- `addsep := < "x" ou "y" ou "z" >`. Cette option, lorsqu'elle n'a pas la valeur *Nil* (valeur par défaut), détermine la boîte englobante de chaque facette et ajoute dans la liste une des faces de cette boîte (face perpendiculaire à l'axe *Ox* avec *x* minimal quand l'option a la valeur "x"), cette facette supplémentaire sera invisible et servira de cloison séparatrice, ainsi la « vraie » facette ne sera pas découpée par une facette qui se trouve entièrement derrière la cloison. Cette option est inutile pour les objets convexes. Sans effet sur l'export povray.

2.6 povLine

`povLine(liste points 3D, [options])`

Définit une ligne polygonale dans l'espace. Options de `povLine` :

- `arrows := < 0/1/2 >`. Indique la présence ou non de flèche (aucune, une ou deux, aucune par défaut). Cette option suppose que la ligne ne contient pas la constante *jump*.
- `arrowsscale := < nombre positif >`. Facteur d'échelle pour les flèches (1 par défaut).
- `clip := < -1/0/1 >`. Indique si la ligne doit être clippée par la fenêtre définie par l'option `clipwin` lorsque `clip=1`, ou bien par le plan défini par l'option `clipwin` lorsque `clip=-1` (`clip=0` par défaut).
- `clipwin := < [M(xinf,yinf,zinf), M(xsup,yup,zsup)] >`. Définit la fenêtre 3D pour un éventuel clipping lorsque `clip=1`, la fenêtre est alors donnée par sa grande diagonale : `[M(xinf,yinf,zinf), M(xsup,yup,zsup)]` (c'est la fenêtre courante par défaut). Mais lorsque `clip=-1` l'option `clipwin` est interprétée comme un plan : `[point3D, vecteur normal]`.
- `close := < 0/1 >`. Indique s'il faut fermer la ligne ou non, (0 par défaut).
- `color := < couleur >`. Choix de la couleur (black par défaut).
- `render := < 0/1 >`. Indique si l'élément doit être rendu ou non (1 par défaut).
- `shadow := < 0/1 >`. Indique si l'élément a une ombre ou non (1 par défaut).
- `hollow := < 0/1 >`. Lorsque l'option `tube` vaut 1, la ligne est remplacée par un tube à facettes. Celui-ci peut être creux (`hollow :=1`) ou non (`hollow :=0`) (0 par défaut).
- `linestyle := < style de ligne >`. Définit le style de tracé de ligne (solid par défaut). Sans effet sur l'export povray, pour celui-ci la ligne est exportée sous forme d'une suite de cylindres lorsque `tube` vaut 0.
- `nbfacet := < nombre de facettes >`. Définit le nombre de facettes lorsque `tube` vaut 1 (4 par défaut).
- `opacity := < nombre entre 0 et 1 >`. Valeur de l'opacité (1 par défaut), permet d'introduire la transparence lorsque l'opacité est strictement inférieure à 1.
- `radius := < rayon du tube >`. Rayon du tube lorsque `tube` vaut 1 (0.01 par défaut).
- `radiusscale := < nombre > 0 >`. Facteur d'échelle pour le rayon du tube lorsque `tube` vaut 1 (1 par défaut).
- `tube := < 0/1 >`. Indique s'il faut construire un tube (à facettes) à partir de la ligne (0 par défaut).
- `width := < épaisseur du trait >` (8 par défaut).
- `matrix := < matrice 3d >`. Permet de définir une matrice de transformation qui sera appliquée aux points de la ligne (l'identité par défaut). La transformation s'effectue avant l'éventuel clipping.

Remarque : avec l'option `tube := 1`, la macro `povFacet` est appelée, on peut donc utiliser dans ce cas les options de `povFacet`.

2.7 povPlan

`povPlan(plan, [options])`

Définit un plan, celui-ci est représenté par une liste du type : [point 3D, vecteur normal]. Options de bdPlan :

- `scale := < nombre strictement positif >`. Le plan est intersecté par la fenêtre 3D courante ce qui donne une facette, celle-ci peut être agrandie ou diminuée.
- `border := < 0/1 >`. Indique si le bord doit être dessiné ou non (0 par défaut).
- `bordercolor := < couleur >`. Indique la couleur du contour (identique à `color` par défaut).

Remarque : cette macro appelle `povFacet`, on peut donc utiliser les options de `povFacet`. Par défaut l'option `twoside` vaut 0 (on ne distingue pas le devant-derrrière de la facette).

2.8 povPlanEqn

`povPlanEqn(équation, [options])`

Définit le plan d'équation $ax + by + cz = d$, celui-ci est représenté par la liste : [a,b,c,d]. Les options de `povPlanEqn` sont les mêmes que celles de l'option `povPlan`.

2.9 povDroite

`povDroite(droite, [options])`

Définit une droite, celle-ci est représentée par une liste du type : [point 3D, vecteur directeur]. Options de `povDroite` :

- `scale := < nombre strictement positif >`. La droite est intersectée par la fenêtre 3D courante ce qui donne un segment, celui-ci peut être agrandi ou diminué.

Remarque : cette macro appelle `povLine`, on peut donc utiliser les options de `povLine`.

2.10 povCurve

`povCurve(f(t), [options])`

Définit une courbe dans l'espace, celle-ci est paramétrée par $f(t)=[x(t)+i*y(t), z(t)]$, où $x(t)$, $y(t)$ et $z(t)$ sont des fonctions d'une variable t . Options de `povCurve` :

- `t := < [tmin, tmax] >`. Intervalle pour le paramètre t , [-5,5] par défaut.
- `nbdot := < entier positif >`. Définit le nombre de points, celui-ci est de 25 par défaut.

Remarque : cette macro appelle `povLine`, on peut donc utiliser les options de `povLine`.

2.11 povSurf

`povSurf(f(u,v), [options])`

Définit une surface paramétrée, par $f(u,v)=[x(u,v)+i*y(u,v), z(u,v)]$, où x , y et z sont des fonctions de deux variables u et v . Options de `povSurf` :

- `u := < [umin, umax] >`. Intervalle pour le paramètre u , [-5,5] par défaut.
- `v := < [vmin, vmax] >`. Intervalle pour le paramètre v , [-5,5] par défaut.
- `grid := < [unbdot, vnbdot] >`. Définit la grille, c'est à dire le nombre de points pour u et pour v , celle-ci est de [25,25] par défaut.

Remarque : cette macro appelle `povFacet`, on peut donc utiliser les options de `povFacet`.

2.12 povImplicit

`povImplicit(f(x,y,z), [options])`

Définit une surface implicite d'équation $f(x, y, z) = 0$. Options de `povImplicit` :

- `clipwin := < [M(xinf,yinf,zinf), M(xsup,yup,zsup)] >`. Cet objet est obligatoirement clippé, et cette option permet de définir la fenêtre 3D pour le clipping, la fenêtre est donnée par sa grande diagonale : [M(xinf,yinf,zinf), M(xsup,yup,zsup)] (c'est la fenêtre courante par défaut).
- `color := < couleur >`. Choix de la couleur (white par défaut).
- `opacity := < nombre entre 0 et 1 >`. Valeur de l'opacité (1 par défaut), permet d'introduire la transparence lorsque l'opacité est strictement inférieure à 1.
- `matrix := < matrice 3d >`. Permet de définir une matrice de transformation qui sera appliquée (l'identité par défaut).

Il n'y aura pas de rendu écran car cette fonctionnalité n'est pas encore implémentée dans TeXgraph, la surface est donc visible seulement dans le rendu povray.

ATTENTION : povray ne gère pas les éventuelles divisions par zéro et ne connaît la fonction puissance que sous la forme $pow(x,n)$.

2.13 povAxes

`povAxes(origine, [options])`

Définit les axes, *origine* est un point 3D qui désigne le point de concours des trois axes. Options de `povAxes` :
 — `labels := < 0/1 >`. Indique la présence ou non des lettres *x*, *y* et *z* au bout des trois axes (1 par défaut).
 Remarque : cette macro appelle `povLine`, on peut donc utiliser les options de `povLine`.

2.14 povAngleD

`povAngleD(B,A,C, longueur, [options])`

Crée « l'angle droit » de l'espace définie par les deux droites (*AB*) et (*AC*), où *A*, *B* et *C* sont des points 3D.
 La macro `povAngleD` appelle `povLine`, on peut donc utiliser les options de `povLine`.

2.15 povDot

`povDot(liste de points 3D, [options])`

Définit une liste de points de l'espace. Options de `povDot` :

- `color := < couleur >`. Définit la couleur des points (black par défaut).
- `dir := < vecteur1 ou [vecteur1,vecteur2] >`. Lorsque `dotstyle=line` (un trait), l'option `dir` doit contenir un vecteur directeur du trait à tracer (dans l'espace). Lorsque `dotstyle=cross` (croix), l'option `dir` doit contenir une liste de deux vecteurs directeurs pour les traits à tracer (dans l'espace). par défaut `dir` vaut Nil.
- `dotstyle := < disc/cube/line/cross >`. Définit le style de points (disc par défaut, ce qui donne une sphère dans le rendu povray).
- `dotstyle := < nombre positif >`. Définit un facteur d'échelle (1 par défaut).

Lorsque `dotstyle=cube` la macro `povFacet` est appelée, on peut dans ce cas utiliser les options de `povFacet`, lorsque `dotstyle=line` ou `cross` la macro `povLine` est appelée, on peut alors utiliser les options de `povLine`.

2.16 povLabel

`povLabel(point d'ancrage, texte, [options])`

Définit un label dans l'espace, le *point d'ancrage* est un point 3D. Options de `povLabel` :

- `TeXify := < 0/1 >`. Indique si le label est une formule mathématique qui doit être compilée par \TeX , si c'est le cas, on peut ne pas mettre **les dollars** dans la formule en fonction de l'option `dollar`, `TeXgraph` lance une compilation `pdflaTeX` en arrière-plan puis appelle l'utilitaire `pstoedit`² qui traduit le fichier pdf en flattened postscript que `TeXgraph` peut ensuite parser pour récupérer la formule sous forme de chemins. Cela suppose donc qu'une distribution \TeX est installée ainsi que le programme `pstoedit`. Le fichier compilé s'appelle `tex2FlatPs.tex`, et se trouve dans le dossier `.TeXgraph` de l'utilisateur sous linux, et dans `c:/tmp` sous windows. On en trouve également une copie dans le dossier d'installation de `TeXgraph`, par défaut ce fichier utilise la police *Fourier* en 12pt, la formule est insérée entre les deux délimiteurs : `\[. . \]` et composée avec la taille `\large`. Par défaut l'option `TeXify` vaut 0.
- `dollar := < 0/1 >`. Lorsque l'option `TeXify` vaut 1, le label est inclus automatiquement dans un environnement `displaymath` avant d'être compilé si l'option `dollar` vaut 1, sinon le label est compilé en temps que texte. Par défaut l'option `dollar` vaut 1.
- `scale := < nombre>0 >`. Lorsque l'option `TeXify` vaut 1, la taille du label peut être modifiée avec cette option. Par défaut cette option vaut 0.
- `color := < couleur >`. Définit la couleur du label (black par défaut).
- `render := < 0/1 >`. Indique si l'élément doit être rendu ou non (1 par défaut).
- `shadow := < 0/1 >`. Indique si l'élément a une ombre ou non (0 par défaut).
- `dotcolor := < couleur >`. Définit la couleur du point d'ancrage si celui-ci doit être affiché (égale à `color` par défaut).
- `labelpos := < [distance cm, affixe direction] >`. Indique la position du label par rapport au point d'ancrage **sur le plan de projection** (Nil par défaut, dans ce cas la distance est considérée comme nulle).
- `labeldir := < [vecteur1, vecteur2, épaisseur] >`. Ces deux vecteurs de l'espace indiquent le sens de l'écriture, les lettres sont écrites le long de l'axe porté par `vecteur1` et dans le sens du `vecteur2`. Par défaut ces deux vecteurs sont sur les axes *Ox* et *Oy* de l'écran, et le troisième paramètre vaut 0.05.
- `labelsize := < >`. Définit la taille du label comme `LabelSize` (égal à `LabelSize` par défaut) lorsque l'option `TeXify` vaut 0.
- `labelstyle := < type de label >`. Définit le style de label comme `LabelStyle` (égal à `LabelStyle` par défaut).
- `showdot := < 0/1 >`. Indique si le point d'ancrage doit être affiché (0 par défaut).

Lorsque `showdot=1`, on peut utiliser les options de `povDot` car celle-ci sera appelée.

2. <http://www.pstoedit.net/>.

2.17 povArc

`povArc(B,A,C,R,sens, [options])`

Définit un arc de cercle dans l'espace de rayon R , allant de (AB) vers (AC) . Le plan (BAC) est orienté par la base $(B - A, C - A)$ et le *sens* doit valoir 1 s'il est direct ou -1 sinon. Options de `povArc` :

- `labelarc(<texte>)`. Permet de placer un label sur l'arc.
- `normal := < vecteur non nul >`. Vecteur qui sera considéré comme le vecteur normal au plan si l'angle est plat (Nil par défaut).
- `radscale := < nombre >`. Nombre qui, multiplié par le rayon de l'arc, donnera la distance du label au centre de l'arc (1.25 par défaut).

C'est la macro `povCurve` qui est appelée pour dessiner l'arc, on peut donc utiliser les options de `povCurve`.

2.18 povCercle

`povCercle(centre, rayon, vecteur normal, [options])`

Définit un cercle dans l'espace, le plan du cercle est orthogonal au *vecteur normal* passé en troisième paramètre. C'est la macro `povTorus` qui est appelée pour dessiner le cercle lors de l'export `povray`, et la macro `bdCercle` pour le dessin à l'écran.

3 Exemples

Pour tester ces exemples :

1. Lancer l'interface graphique,
2. charger le modèle `povray.mod` (F3), deux boutons apparaissent à gauche de la fenêtre, c'est le deuxième qui permet l'export vers `povray`,
3. copier-coller le code proposé dans un élément graphique Utilisateur (ctrl+U).

3.1 Section d'un tore

Code

```
\begin{texgraph}[name=tore, file]
Include "povray.mod";
Graph image = [
view(-5,5,-5,5), view3D(-5,5,-5,5,-4,4),
Marges(0,0,0,0), size(12,1),
backcolor :=darkgray, TeXifyLabels :=1,
povScene(
povPlan([2,0,-vecI], [color :=darkseagreen, scale :=0.85, opacity :=0.3]),
povTorus( [0,0],3,1,vecK, [color :=steelblue, clip :=-1,
clipwin :=[2,0,-vecI]]),
povAxes([0,0], [color :=gold, arrows :=1, arrowsscale :=1.5])
),
Display3D()
];
\end{texgraph}
```

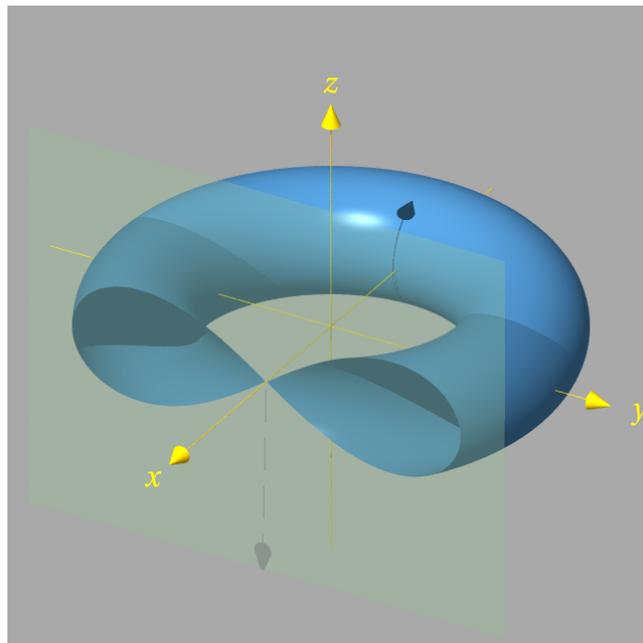


FIGURE 1: Section d'un tore

3.2 Arcs et segments



Code

```

\begin{texgraph}[name=sphere, file]
Include "povray.mod";
Graph image = [
view(-5.5,5.5,-5.5,5.5), view3D(-5.5,-5.5,-5.5), Marges(0,0,0,0), size(12,1),
A :=4*M(1,1,1)/sqrt(3), Az :=pz(A), Axy :=pxy(A), Ax :=px(A), Ay :=py(A),
backcolor :=darkgray, TeXifyLabels :=1,
povScene(
povPlan([0,0,vecK], [color :=darkseagreen, scale :=0.9, opacity :=0.8]),
povSphere([0,0],4, [color :=darkblue, opacity :=0.6]),
povLabel(A,"A\begin{pmatrix}\cos(\theta)\sin(\varphi)\\"
\sin(\theta)\sin(\varphi)\\"
\cos(\varphi)\end{pmatrix}",
[ color :=gold, showdot :=1, dotscale :=1.5,scale :=0.85,
labelpos :=[0.5,1], labelstyle :=left]),
povLabel(M(4.25,4,0),"plan P",[labelstyle :=right+bottom,
labeldir :=[vecJ,-vecI]]),
povLine([Az,A,0,0,Axy,A,jump,Axy,4*normalize(Axy),jump,Ax,Axy,Ay],
[ color :=red]),
povArc(Az,Origin,A,4,1,[color :=brown, arrows :=1,
labelarc("\varphi")]),
povArc(pxx(A),Az,A,Norm(A-Az),1, [color :=brown, arrows :=1,
labelarc("\theta")]),
povArc(Az,Origin,[4,0],4,1,[color :=brown]),
povAxes([0,0],[color :=gold, arrows :=1, arrowsscale :=1.5]),
),
Display3D()
];
\end{texgraph}

```

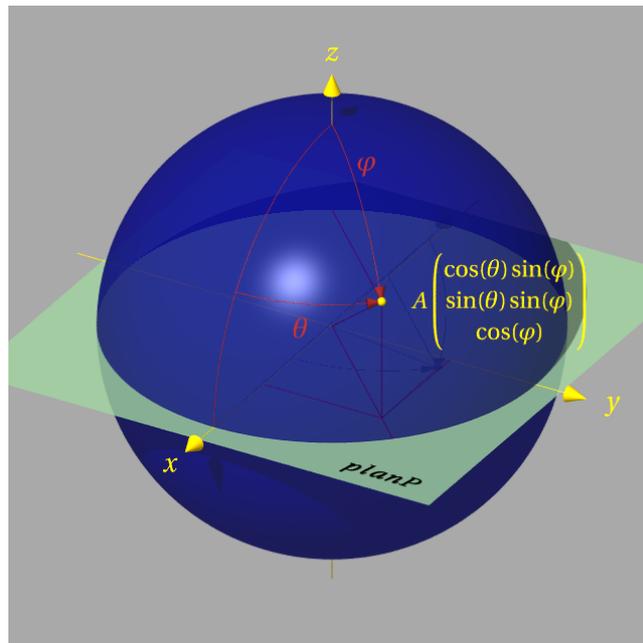


FIGURE 2: Sphère et arcs de cercle

3.3 Une hyperbole



Code

```

\begin{texgraph}[name=hyperbole, file]
Include "povray.mod";
Graph image = [
view(-7,7,-7,7), Marges(0,0,0,0), size(12,1), view3D(-6,6,-6,6,6,6),
H :=5, R :=4, a :=2, Plan :=[M(a,0,0),vecI],
C1 :=Cone(Origin, H*vecK,R,45),
C2 :=Cone(Origin, -H*vecK,R,45),
L1 :=Intersection(Plan,C1), L2 :=Intersection(Plan,C2),
Asymp1 :=getdroite( [M(a,0,0),M(0,R,H)] ),
Asymp2 :=getdroite( [M(a,0,0),M(0,-R,H)] ),
backcolor :=gray, imagescale :=0.75,
povScene(
povComment("plan1"),
povPlan(Plan, [color :=darkseagreen, opacity :=0.6]),
povComment("cone"),
povCone(Origin, H*vecK,R,[color :=slategray]),
povCone(Origin, -H*vecK,R,[color :=slategray]),
povComment("asymptotes"),
povLine([ Asymp1,Asymp2], [color :=blue, width :=4]),
povComment("hyperbole"),
povLine(Merge3d([L1,L2]), [color :=red, width :=8]),
povComment("Axes"),
povAxes([0,0], [color :=gold, labels :=1, arrows :=1, arrowsscale :=1.5,
newxlegend("x"),newylegend("y"),newzlegend("z")])
),
Display3D()
];
\end{texgraph}

```

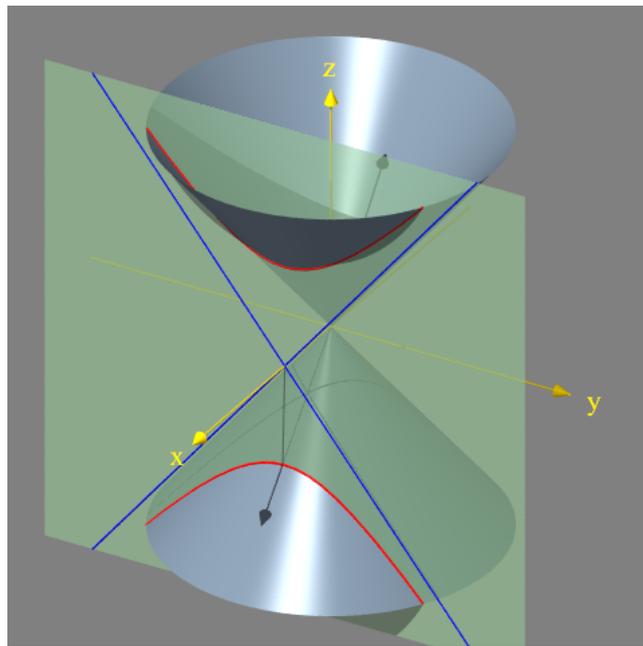


FIGURE 3: Une hyperbole

Dans cet exemple on a ajouté un commentaire avant chaque élément avec la macro `povComment("texte")`, ce commentaire est écrit dans le fichier povray juste avant la déclaration de l'objet suivant, et juste avant son rendu. Cela permet de retrouver facilement un objet pour modifier son rendu.

3.4 Fenêtre de Viviani

Code

```

\begin{texgraph}[name=viviani, file]
Include "povray.mod";
Graph image = [
view3D(-1.5,2,-1.5,1.5,-1.5,2),
view(-1.5,1.75,-1.5,2.25), Marges(0,0,0,0), size(12,1),
backcolor :=gray,
povScene(
povCylinder([0.5,-1.15], 2.5*vecK,0.5, [color :=darkseagreen, opacity :=0.8]),
povSphere( [0,0], 1, [color :=steelblue, smooth :=1, backculling :=1]),
povAxes([0,0],[color :=gold, labels :=0, arrows :=1, arrowsscale :=0.75]),
povCurve([cos(t)*exp(-i*t),sin(t)],
[t=[0,2*pi], nbdot :=100, nbfacet :=6, radiusscale :=3, tube :=1,
color :=red, smooth :=1])
),
Display3D()
];
\end{texgraph}

```

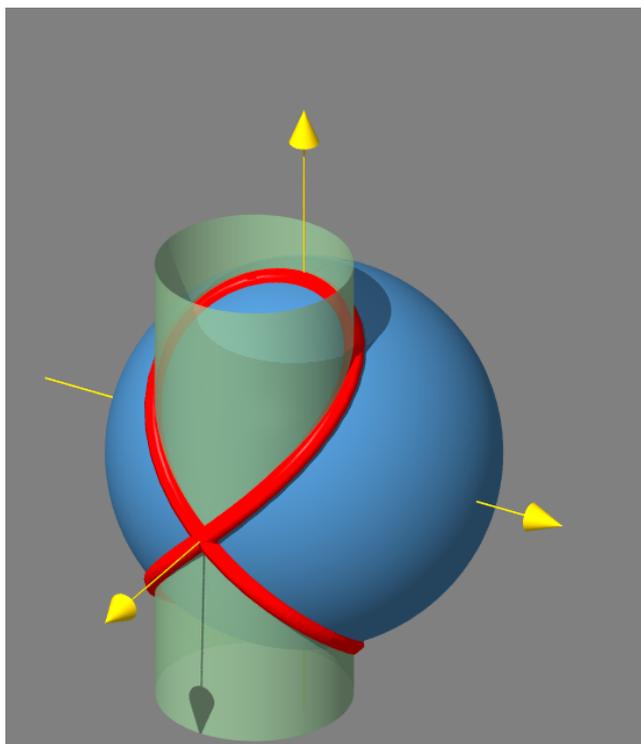


FIGURE 4: Fenêtre de Viviani

3.5 L'horoptère

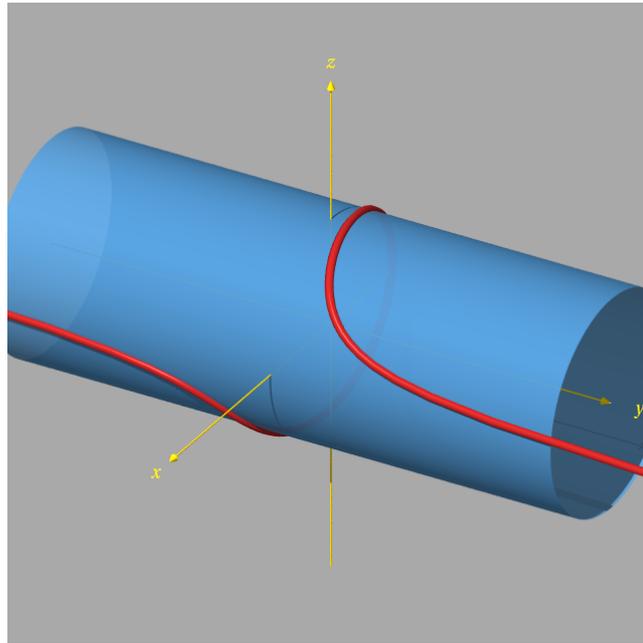


Code

```

\begin{texgraph}[name=horoptere, file]
Include "povray.mod";
Graph image = [
a :=3, b :=0.5, Marges(0,0,0,0), view3D(-5,8,-8,8,-7,7),
view(-8,8,-8,8),size(12,1),
backcolor :=darkgray,
povScene(
  povCylinder(M(0,-8,0), 16*vecJ, a,
    [color :=steelblue, hollow :=1, opacity :=0.9]),
  povAxes([0,0], [color :=gold,labels :=1,TeXify :=1,arrows :=1]),
  povCurve([-a*cos(t)+i*b*tan(t/2), a*sin(t)],
    [color :=firebrick, tube :=1, hollow :=1, t :=0.99*[-pi,pi],
    nbdot :=100, smooth :=1, radiusscale :=1.5, nbfacet :=6]
  ),
Display3D0
];
\end{texgraph}

```

FIGURE 5: *L'horoptère*

3.6 Courbes de niveaux

Code

```

\begin{texgraph}[name=niveaux, file]
Include "povray.mod";
Graph image = [
Marges(0,0,0, view(-5,5,-5,5),size(12,1),
stock :=GetSurface([u+i*v,3*(sin(u)+cos(v))], pi*(-1+i),pi*(-1+i)),
R :=rectangle3d(stock), z1 :=Zde(R[1,2]), z2 :=Zde(R[3,2]),
nb :=8, pas :=(z2-z1)/nb,
transformbox3d(R), Mat :=GetMatrix3D(),
IdMatrix3D(),view3D(-4,4,-4,4,-4,4),
palette :=[purple,darkblue, blue,darkgreen, green, yellow,orange,red],
backcolor :=gray,
povScene( S :=stock, z :=z1,
for k from 1 to nb-1 do
Inc(z,pas),
S :=ClipFacet(S,[M(0,0,z),-vecK], S'), Inc(k,1),
povFacet(S,[color :=palette[k], smooth :=1, matrix :=Mat]), sep3D,
Echange(S,S')
od,
povFacet(S,[color :=crimson, smooth :=1, matrix :=Mat]),
povAxes([0,0],[color :=gold, arrows :=1, labels :=0, arrowsscale :=1.5])),
Display3D()
];
\end{texgraph}

```

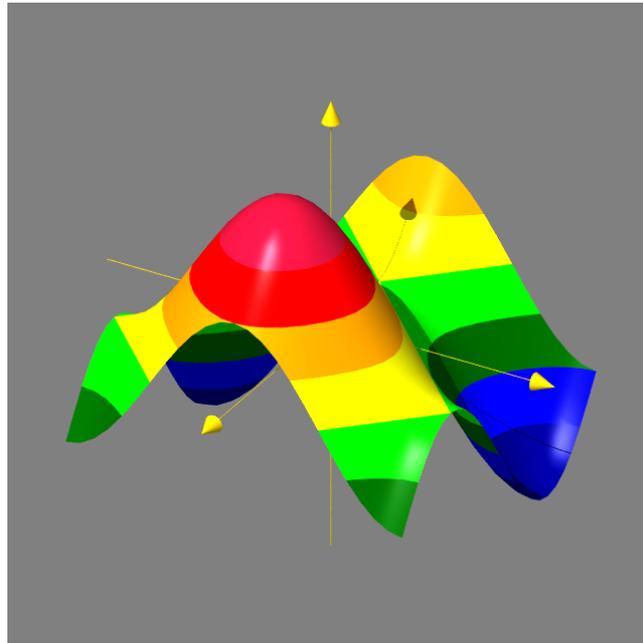


FIGURE 6: Courbes de niveaux

3.7 Rendu personnalisé

Code

```

\begin{texgraph}[name=verreEau, file]
Include "povray.mod";
Var textureEau = "texture { // texture eau
  pigment { rgbf <1,1,1, 0.7> }
  normal { ripples 0.15 frequency 5 }
  finish { reflection {0.1,1 fresnel} conserve_energy }
} interior {ior 1.33 caustics 0.1}";
textureVerre = "texture { // texture verre
  pigment { color rgbt <0.9,0.9,1, 0.9> }
  finish { ambient 0.1 diffuse 0.6 reflection 0.025 specular 0.5
    roughness 0.003 phong 1 phong_size 400 }
} interior { ior 1.5}";
Graph image = [
view(-7,7,-2,10),view3D(-6,6,-6,6,-1,10),Marges(0,0,0,0),size(12,1), r :=3, bgcolor :=white,
povScene(
  povComment("eau"),
  povCylinder([0,0,1],[2*r]*vecK,r,[color :=aqua,smooth :=1,opacity :=0.2,
    render :=0, hollow :=0]),
  povSpecial(["object{ objet1",LF, textureEau,LF,"}"]),
  povComment("grand cylindre"),
  povCylinder([0,0],[3*r]*vecK,r+0.15,[color :=Rgb(0.74,1,0.73),smooth :=1,
    opacity :=0.4, render :=0, hollow :=0]),
  povComment("cylindre intérieur"),
  povCylinder([0,0.05],[3*r]*vecK,r+0.05,[color :=Rgb(0.74,1,0.73),smooth :=1,
    opacity :=0.4, render :=0, hollow :=0]),
  povComment("bord supérieur"),
  povTorus([0,3*r], r+0.05, 0.15, vecK, [color :=beige, render :=0]),
  povSpecial(["union{ difference{ object{objet2} object{objet3} }
    object{objet4}",LF, textureVerre,LF,"}"]),
  povComment("sphere"),
  povSphere([0+0*i,r+0.1],r-0.01,[color :=gold,smooth :=1,opacity :=1]),
  povComment("segment fleché"),
  povLine([[0+0*i,3*r],[0+r*i,3*r]],[color :=slategray,width :=8,
    arrows :=2, shadow :=0]),
  povComment("label"),
  povLabel([0+(r/2)*i,3.1*r],"9", [labelstyle :=scriptsize]),
  povComment("plan de pose"),
  povPlan([0,-0.1,vecK], [scale :=5, color :=whitesmoke, contrast :=0]
),
Display3D0];
\end{texgraph}

```



FIGURE 7: *Rendu personnalisé*