Le fichier solides.pro

version 3.02, 15 janvier 2008

Ce document présente le fichier **`solides.pro`** utilisé pour le package PSTricks *pst-solides3d*(*). Une grande partie de ce fichier est synchrnoisée avec la « librairie jps » utilisée pour le logiciel *jps2ps*(**). Cette librairie est consultable à l'url : **`melusine.eu.org/syracuse/bbgraf/jps2ps/pps/src.xml`**.

Lorsque le fichier **`solides.pro`** contient une ligne avec l'expression `###` *file* `###`, cela signifie que les lignes qui suivent sont synchronisées avec le fichier **`melusine.eu.org/syracuse/bbgraf/jps2ps/pps/`***file***`.pps`**.

1. En-têtes, initialisations de variables globales

Le fichier solides.pro

```
%!
% PostScript prologue for pst-solides3d.tex.
% Version 3.02, 2008/01/15
%
%
/SolidesDict 100 dict def
/SolidesbisDict 100 dict def
SolidesDict begin

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %% les variables globales gerees par PSTricks %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %% les lignes dessous sont a decommenter si l on veut utiliser le
%% %% fichier solides.pro independamment du package PSTricks
%% /Dobs 20 def
%% /THETA 20 def
%% /PHI 50 def
%% /Decran 30 def
%% /XpointVue {Dobs Cos1Cos2 mul} def
%% /YpointVue {Dobs Sin1Cos2 mul} def
%% /ZpointVue {Dobs Sin2 mul} def
%% /xunit 28.14 def
%% /solidhollow false def
%% /solidbiface false def
%% /xunit 28.45 def
%% /tracelignedeniveau? true def
%% /hauteurlignedeniveau 1 def
%% /couleurlignedeniveau {rouge} def
%% /linewidthlignedeniveau 4 def
%% /solidgrid true def
/aretescachees true def
/defaultsolidmode 2 def
/activationgestioncouleurs true def


/fillstyle {} def
/startest false def
/cm {} def
/cm_1 {} def
/yunit {xunit} def
/angle_repere 90 def

/hadjust 2.5 def
/vadjust 2.5 def

/pointilles {
   [6.25 3.75] 1.25 setdash
} def
/stockcurrentcpath {} def
/newarrowpath {} def

```

(*) **`melusine.eu.org/syracuse/pstricks/pst-solides3d/`**
(**) **`melusine.eu.org/syracuse/bbgraf/`**

2. Déclaration d'une fonte accentuée

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% choix d une fonte accentuee pour le .ps %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/ReEncode { exch findfont
dup length dict begin { 1 index /FID eq {pop pop} {def} ifelse
}forall /Encoding ISOLatin1Encoding def currentdict end definefont
pop }bind def
/Font /Times-Roman /ISOfont ReEncode /ISOfont def
%Font findfont 10 scalefont setfont

```

3. Définitions des couleurs

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% extrait de color.pro pour pouvoir recuperer ses couleurs %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
/GreenYellow{0.15 0 0.69 0 setcmykcolor}def
/Yellow{0 0 1 0 setcmykcolor}def
/Goldenrod{0 0.10 0.84 0 setcmykcolor}def
/Dandelion{0 0.29 0.84 0 setcmykcolor}def
/Apricot{0 0.32 0.52 0 setcmykcolor}def
/Peach{0 0.50 0.70 0 setcmykcolor}def
/Melon{0 0.46 0.50 0 setcmykcolor}def
/YellowOrange{0 0.42 1 0 setcmykcolor}def
/Orange{0 0.61 0.87 0 setcmykcolor}def
/BurntOrange{0 0.51 1 0 setcmykcolor}def
/Bittersweet{0 0.75 1 0.24 setcmykcolor}def
/RedOrange{0 0.77 0.87 0 setcmykcolor}def
/Mahogany{0 0.85 0.87 0.35 setcmykcolor}def
/Maroon{0 0.87 0.68 0.32 setcmykcolor}def
/BrickRed{0 0.89 0.94 0.28 setcmykcolor}def
/Red{0 1 1 0 setcmykcolor}def
/OrangeRed{0 1 0.50 0 setcmykcolor}def
/RubineRed{0 1 0.13 0 setcmykcolor}def
/WildStrawberry{0 0.96 0.39 0 setcmykcolor}def
/Salmon{0 0.53 0.38 0 setcmykcolor}def
/CarnationPink{0 0.63 0 0 setcmykcolor}def
/Magenta{0 1 0 0 setcmykcolor}def
/VioletRed{0 0.81 0 0 setcmykcolor}def
/Rhodamine{0 0.82 0 0 setcmykcolor}def
/Mulberry{0.34 0.90 0 0.02 setcmykcolor}def
/RedViolet{0.07 0.90 0 0.34 setcmykcolor}def
/Fuchsia{0.47 0.91 0 0.08 setcmykcolor}def
/Lavender{0 0.48 0 0 setcmykcolor}def
/Thistle{0.12 0.59 0 0 setcmykcolor}def
/Orchid{0.32 0.64 0 0 setcmykcolor}def
/DarkOrchid{0.40 0.80 0.20 0 setcmykcolor}def
/Purple{0.45 0.86 0 0 setcmykcolor}def
/Plum{0.50 1 0 0 setcmykcolor}def
/Violet{0.79 0.88 0 0 setcmykcolor}def
/RoyalPurple{0.75 0.90 0 0 setcmykcolor}def
/BlueViolet{0.86 0.91 0 0.04 setcmykcolor}def
/Periwinkle{0.57 0.55 0 0 setcmykcolor}def
/CadetBlue{0.62 0.57 0.23 0 setcmykcolor}def
/CornflowerBlue{0.65 0.13 0 0 setcmykcolor}def
/MidnightBlue{0.98 0.13 0 0.43 setcmykcolor}def
/NavyBlue{0.94 0.54 0 0 setcmykcolor}def
/RoyalBlue{1 0.50 0 0 setcmykcolor}def
/Blue{1 1 0 0 setcmykcolor}def
/Cerulean{0.94 0.11 0 0 setcmykcolor}def
/Cyan{1 0 0 0 setcmykcolor}def
/ProcessBlue{0.96 0 0 0 setcmykcolor}def
/SkyBlue{0.62 0 0.12 0 setcmykcolor}def
/Turquoise{0.85 0 0.20 0 setcmykcolor}def
/TealBlue{0.86 0 0.34 0.02 setcmykcolor}def
/Aquamarine{0.82 0 0.30 0 setcmykcolor}def
/BlueGreen{0.85 0 0.33 0 setcmykcolor}def
/Emerald{1 0 0.50 0 setcmykcolor}def
/JungleGreen{0.99 0 0.52 0 setcmykcolor}def
```

Le fichier solides.pro (suite)

```
/SeaGreen{0.69 0 0.50 0 setcmykcolor}def
/Green{1 0 1 0 setcmykcolor}def
/ForestGreen{0.91 0 0.88 0.12 setcmykcolor}def
/PineGreen{0.92 0 0.59 0.25 setcmykcolor}def
/LimeGreen{0.50 0 1 0 setcmykcolor}def
/YellowGreen{0.44 0 0.74 0 setcmykcolor}def
/SpringGreen{0.26 0 0.76 0 setcmykcolor}def
/OliveGreen{0.64 0 0.95 0.40 setcmykcolor}def
/RawSienna{0 0.72 1 0.45 setcmykcolor}def
/Sepia{0 0.83 1 0.70 setcmykcolor}def
/Brown{0 0.81 1 0.60 setcmykcolor}def
/Tan{0.14 0.42 0.56 0 setcmykcolor}def
/Gray{0 0 0 0.50 setcmykcolor}def
/Black{0 0 0 1 setcmykcolor}def
/White{0 0 0 0 setcmykcolor}def
%% fin de l extrait color.pro

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%             autres couleurs                        %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/bleu {0 0 1 setrgbcolor} def
/rouge {1 0 0 setrgbcolor} def
/vert {0 .5 0 setrgbcolor} def
/gris {.4 .4 .4 setrgbcolor} def
/jaune {1 1 0 setrgbcolor} def
/noir {0 0 0 setrgbcolor} def
/blanc {1 1 1 setrgbcolor} def
/orange {1 .65 0 setrgbcolor} def
/rose {1 .01 .58  setrgbcolor} def
/cyan {1 0 0 0 setcmykcolor} def
/magenta {0 1 0 0 setcmykcolor} def
```

4. Passage coordonnées 3d en coordonnées 2d

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%             definition du point de vue             %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% pour la 3D conventionnelle
%% Dony : graphisme scientifique : page 187
%% Editeur : Masson

%% calcul des coefficients de la matrice
%% de transformation
/Sin1 {THETA sin} def
/Sin2 {PHI sin} def
/Cos1 {THETA cos} def
/Cos2 {PHI cos} def
/Cos1Sin2 {Cos1 Sin2 mul} def
/Sin1Sin2 {Sin1 Sin2 mul} def
/Cos1Cos2 {Cos1 Cos2 mul} def
/Sin1Cos2 {Sin1 Cos2 mul} def

/3dto2d {
6 dict begin
   /Zcote exch def
   /Yordonnee exch def
   /Xabscisse exch def
   /xObservateur
      Xabscisse Sin1 mul neg Yordonnee Cos1 mul add
   def
   /yObservateur
      Xabscisse Cos1Sin2 mul neg Yordonnee Sin1Sin2 mul sub Zcote Cos2
      mul add
   def
   /zObservateur
      Xabscisse neg Cos1Cos2 mul Yordonnee Sin1Cos2 mul sub Zcote Sin2
      mul sub Dobs add
   def
   %% maintenant on depose les resultats sur la pile
   Decran xObservateur mul zObservateur div cm
```

Le fichier solides.pro (suite)

```
   Decran yObservateur mul zObservateur div cm
end
} def

/getpointVue {
   XpointVue
   YpointVue
   ZpointVue
} def

/GetCamPos {
   getpointVue
} def

```

5. Transcription PSTricks –> jps

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%         jps modifie pour PSTricks                  %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/solid {continu} def
/dashed {pointilles} def

```

6. Pour l'option algebraic

Le fichier solides.pro

```
%%% les 3 procedures utilisees pour transformer les depots de AlgToPs en nombres
/pstrickactionR3 {
3 dict begin
  /len@3 exch def
  /len@2 exch def
  /len@1 exch def
  len@1 exec
  len@2 exec
  len@3 exec
end
} def

/pstrickactionR2 {
   exec exch exec exch
} def

/pstrickactionR {
   exec
} def

```

7. Géométrie 2d basique

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%             geometrie basique                      %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% syntaxe~: [x1 y1 ... xn yn] ligne
/ligne {
gsave
   newpath
      dup 0 getp smoveto
      ligne_
```

Le fichier solides.pro (suite)

```
      starfill
   stroke
grestore
} def

%% syntaxe~: [x1 y1 ... xn yn] ligne_
/ligne_ {
   reversep
   aload length 2 idiv
   {
      slineto
   } repeat
} def

%% syntaxe~: [x1 y1 ... xn yn] polygone
/polygone* {
1 dict begin
   /startest {true} def
   polygone
end
} def

/polygone {
   gsave
      newpath
         aload length 2 idiv
         3 copy pop
         smoveto
         {
            slineto
         } repeat
      closepath
      starfill
      currentlinewidth 0 eq {} {stroke} ifelse
   gsave
} def

%% syntaxe : x y point
/point {
gsave
   1 setlinecap
   newpath
      smoveto
      0 0 rlineto
      5 setlinewidth
   stroke
grestore
} def

```

8. Insertion librairie jps

8.1 - Le repère utilisateur (repère jps)

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                                                        %%%%
%%%%          insertion librairie jps                       %%%%
%%%%                                                        %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%              le repere jps                             %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### AAAscale ###
%%%%%%%%%%%%%%%% les deplacements a l echelle %%%%%%%%%%%%%%%%%

 /v@ct_I {xunit 0} def
 /v@ct_J {angle_repere cos yunit mul angle_repere sin yunit mul} def

/xscale {} def
/yscale {} def

/xscale-1 {} def
/yscale-1 {} def

/gtransform {} def
/gtransform-1 {} def

/jtoppoint {
2 dict begin
   gtransform
   /y exch yscale def
   /x exch xscale def
   v@ct_I x mulv
   v@ct_J y mulv
   addv
end
} def

/rptojpoint {
   xtranslate ytranslate
   3 1 roll         %% xA yB yA xB
   4 1 roll         %% xB xA yB yA
   sub neg 3 1 roll %% yB-yA xB xA
   sub neg exch
   ptojpoint
} def

/rptoppoint {
   xtranslate ytranslate
   3 1 roll         %% xA yB yA xB
   4 1 roll         %% xB xA yB yA
   sub neg 3 1 roll %% yB-yA xB xA
   sub neg exch
} def

/ptojpoint {
4 dict begin
   /Y exch yscale-1 def
   /X exch xscale-1 def
   /y Y yunit angle_repere sin mul div def
   /x X y yunit mul angle_repere cos mul sub xunit div def
   x y
   gtransform-1
end
} def

/smoveto {
   jtoppoint
   moveto
} def

/srmoveto {
   jtoppoint
   rmoveto
} def

/slineto {
   jtoppoint
   lineto
} def

/srlineto {
   jtoppoint
   rlineto
} def

/stranslate {
   jtoppoint
   translate
} def

%%%% ### fin insertion ###
```

```

```

8.2 - Routines de tests

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                  les tests                        %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### isbool ###
%% syntaxe : any isbool --> booleen
/isbool {
   type (booleantype) cvn eq
} def

%%%%% ### isarray ###
%% syntaxe : any isarray --> booleen
/isarray {
   type (arraytype) cvn eq
} def

%%%%% ### isstring ###
%% syntaxe : any isstring --> booleen
/isstring {
   type (stringtype) cvn eq
} def

%%%%% ### isinteger ###
%% syntaxe : any isinteger --> booleen
/isinteger {
   type (integertype) cvn eq
} def

%%%%% ### isnum ###
%% syntaxe : any isnum --> booleen
/isnum {
   dup isreal
   exch isinteger or
} def

%%%%% ### isreal ###
%% syntaxe : any isreal --> booleen
/isreal {
   type (realtype) cvn eq
} def

%%%%% ### eq ###
%% syntaxe : A B eqp3d --> booleen = true si les points A et B sont identiques
/eqp3d {
               %% x1 y1 z1 x2 y2 z2
   4 -1 roll   %% x1 y1 x2 y2 z2 z1
   eq {        %% x1 y1 x2 y2
      eqp
   } {
      pop pop pop pop false
   } ifelse
} def

%% syntaxe : A B eqp --> booleen = true si les points A et B sont identiques
/eqp {
   3 -1 roll
   eq
      {
         eq
            {true}
            {false}
         ifelse
      }
      {pop pop false}
   ifelse
} def

```

```
%% syntaxe : z z' eqc --> true si z = z', false sinon
/eqc {
   eqp
} def

%%%%% ### fin insertion ###

```

8.3 - Conversions de types

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                conversions de types                %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### astr2str ###
%% syntaxe : array str astr2str --> str
%% convertit le contenu de array en chaines de caracteres puis les
%% concatene avec str, en inserant un caractere "space" apres chaque
%% element du tableau array
/astr2str {
5 dict begin
   /str exch def
   /table exch def
   /n table length def
   n 0 eq {
      str
   } {
      table 0 n 1 sub getinterval
      table n 1 sub get (                               ) cvs
      ( ) append
      str append
      astr2str
   } ifelse
end
} def

%%%%% ### numstr2array ###
%% syntaxe : str numstr2array -> array
%% ou str est une chaine de nombres entiers separes par des espaces
%% et array est constitue des elements numeriques entiers de string.
%% exemple :
%% (0 12 4 54) --> [0 12 4 54]
/numstr2array {
3 dict begin
   /str exch def
   /n str length def
   /j -1 def
   [
      0 1 n 1 sub {
         /i exch def
         /j j 1 add store
         str i get
         dup 32 eq {
            %% c est un espace
            /j -1 store
            pop
         } {
            j 1 ge {
               exch 10 mul 48 sub add
            } {
               48 sub
            } ifelse
         } ifelse
      } for
   ]
end
} def

%% syntaxe : array numstr2array -> array
/arraynumstr2arrayarray {
   {numstr2array} apply
} def

%%%%% ### fin insertion ###

```

8.4 - Projection de chaînes de caractères

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                  macros de projection                %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### projtext ###
 /initpr@jtext {
5 dict begin
   dup isbool {
      /mybool exch def
   } {
      /mybool true def
   } ifelse
   dup isarray {
      %% c est un planprojpath
      /type_plan_proj true def
      /table exch def
      /z0 exch def
      /y0 exch def
      /x0 exch def
      0 0
   } {
      %% c est un solidprojpath
      /type_plan_proj false def
      %% y a-t-il un str2
      dup isstring {
         /str2 exch def
      } {
         /str2 {} def
      } ifelse
      %% y a-t-il un alpha
      2 copy pop issolid {
         /alpha 0 def
      } {
         /alpha exch def
      } ifelse
      /i exch def
      /solid exch def
      0 0
   } ifelse
} def
 /closepr@jtext {
   type_plan_proj {
      x0 y0 z0 table mybool projpath
   } {
      solid i alpha str2 mybool projpath
   } ifelse
   fill
   stroke
end
} def

%% syntaxe : str x0 y0 z0 [normal_vect] ultextp3d --> -
%% syntaxe : str x0 y0 z0 [normal_vect] bool ultextp3d --> -
%% syntaxe : str1 solid i str2 ultextp3d --> -
%% syntaxe : str1 solid i str2 bool ultextp3d --> -
%% syntaxe : str1 solid i alpha str2 bool ultextp3d --> -
/ultextp3d {initpr@jtext ultext_ closepr@jtext} def
/cltextp3d {initpr@jtext cltext_ closepr@jtext} def
/bltextp3d {initpr@jtext bltext_ closepr@jtext} def
/dltextp3d {initpr@jtext bltext_ closepr@jtext} def
/ubtextp3d {initpr@jtext ubtext_ closepr@jtext} def
/cbtextp3d {initpr@jtext cbtext_ closepr@jtext} def
/bbtextp3d {initpr@jtext bbtext_ closepr@jtext} def
/dbtextp3d {initpr@jtext bbtext_ closepr@jtext} def
```

Le fichier solides.pro (suite)

```
/uctextp3d {initpr@jtext uctext_ closepr@jtext} def
/cctextp3d {initpr@jtext cctext_ closepr@jtext} def
/bctextp3d {initpr@jtext bctext_ closepr@jtext} def
/dctextp3d {initpr@jtext bctext_ closepr@jtext} def
/urtextp3d {initpr@jtext urtext_ closepr@jtext} def
/crtextp3d {initpr@jtext crtext_ closepr@jtext} def
/brtextp3d {initpr@jtext brtext_ closepr@jtext} def
/drtextp3d {initpr@jtext brtext_ closepr@jtext} def

```

8.5 - Appliquer une transformation à un chemin

Le fichier solides.pro

```
%%%%% ### currentppathtransform ###
%% syntaxe : {f} currentppathtransform --> applique la transformation f
%% au chemin courant
/currentppathtransform {
6 dict begin
   /warp exch def
   %% pour remplacer 'move'
   /warpmove{
      2 index {
        newpath
      } if
      warp moveto
      pop false
   } def

   %% pour remplacer 'lineto'
   /warpline {
      warp lineto
   } bind def

   %% pour remplacer 'curveto'
   /warpcurve {
      6 2 roll warp
      6 2  roll warp
      6 2 roll warp
      curveto
   }  bind def

   true
   { warpmove } {  warpline } { warpcurve } { closepath } pathforall
   pop
end
} def

%% syntaxe : {f} currentpathtransform --> applique la transformation f
%% au chemin courant
/currentpathtransform {
7 dict begin
   /transform exch def
   /warp {ptojpoint transform} def
   %% pour remplacer 'move'
   /warpmove{
      2 index {
        newpath
      } if
      warp smoveto
      pop false
   } def

   %% pour remplacer 'lineto'
   /warpline {
      warp slineto
   } bind def

   %% pour remplacer 'curveto'
   /warpcurve {
      6 2 roll warp
      6 2  roll warp
      6 2 roll warp
```

Le fichier solides.pro (suite)

```
      scurveto
   }  bind def

   true
   { warpmove } {  warpline } { warpcurve } { closepath } pathforall
   pop
end
} def

```

8.6 - Base orthonormale à partie de la normale

Le fichier solides.pro

```
%%%%% ### normalvect_to_orthobase ###
%% syntaxe : [normal_vect] normalvect_to_orthobase
%%    --> imI imJ imK
/normalvect_to_orthobase {
4 dict begin
   dup length 3 eq {
      aload pop normalize3d /normal_vect defpoint3d
      normal_vect -1 0 0 eqp3d {
         /imageI {0 -1 0} def
         /imageK {-1 0 0} def
         /imageJ {0 0 1} def
      } {
         %% on calcule l image de la base (I,J,K)
         /imageJ {normal_vect 1 0 0 vectprod3d normalize3d} def
         /imageK {normal_vect} def
         /imageI {imageJ imageK vectprod3d} def
         1 0 0 imageK angle3d 0 eq {
            0 1 0 normal_vect vectprod3d /imageI defpoint3d
            /imageJ {0 1 0} def
            normal_vect /imageK defpoint3d
         } if
      } ifelse
   } {
      dup length 6 eq {
         aload pop
         normalize3d /imageK defpoint3d
         normalize3d /imageI defpoint3d
         imageK imageI vectprod3d /imageJ defpoint3d
      } {
         dup length 7 eq {
            aload pop
            /alpha exch 2 div def
            normalize3d /imageK defpoint3d
            normalize3d /imageI defpoint3d
            imageK imageI vectprod3d /imageJ defpoint3d
            %% et ensuite, on fait tourner la base autour de imageK
            imageI alpha cos mulv3d
            imageJ alpha sin mulv3d
            addv3d

            imageI alpha sin neg mulv3d
            imageJ alpha cos mulv3d
            addv3d

            /imageJ defpoint3d
            /imageI defpoint3d
         } {
            %% length = 4
            aload pop
            /alpha exch def
            normalize3d /normal_vect defpoint3d

            normal_vect -1 0 0 eqp3d {
               /imageI {0 -1 0} def
               /imageK {-1 0 0} def
               /imageJ {0 0 1} def
            } {
               %% on calcule l image de la base (I,J,K)
               /imageJ {normal_vect 1 0 0 vectprod3d normalize3d} def
```

Le fichier solides.pro (suite)

```
               /imageK {normal_vect} def
               /imageI {imageJ imageK vectprod3d} def
               1 0 0 imageK angle3d 0 eq {
                  0 1 0 normal_vect vectprod3d /imageI defpoint3d
                  /imageJ {0 1 0} def
                  normal_vect /imageK defpoint3d
               } if
            } ifelse
         } ifelse

         %% et ensuite, on fait tourner la base autour de imageK
         imageI alpha cos mulv3d
         imageJ alpha sin mulv3d
         addv3d

         imageI alpha sin neg mulv3d
         imageJ alpha cos mulv3d
         addv3d

         /imageJ defpoint3d
         /imageI defpoint3d
      } ifelse
   } ifelse
   imageI
   imageJ
   imageK
end
} def

```

8.7 - Projection d’un chemin

Le fichier solides.pro

```
%%%%% ### projpath ###
%% syntaxe : x y z [normal] projpath --> planprojpath
%% syntaxe : x y z [normal] bool projpath --> planprojpath
%% syntaxe : solid i projpath --> solidprojpath
%% syntaxe : solid i bool projpath --> solidprojpath
%% syntaxe : solid i str bool projpath --> solidprojpath
%% syntaxe : solid i alpha str bool projpath --> solidprojpath
/projpath {
2 dict begin
   dup isbool {
      /mybool exch def
   } {
      /mybool true def
   } ifelse
   dup isarray {
      mybool planprojpath
   } {
      mybool solidprojpath
   } ifelse
end
} def

%% syntaxe : solid i str bool solidprojpath --> -
%% ou
%% syntaxe : solid i alpha str bool solidprojpath --> -
%% projette le chemin courant sur la face i du solide, apres
%% eventuellement une rotation d angle alpha autour de la normale
%% bool : pour savoir si on tient compte de la visibilite
/solidprojpath {
5 dict begin
   /visibility exch def
   dup isstring {
      /option exch def
   } if
   2 copy pop
   issolid {
      /alpha 0 def
   } {
      /alpha exch def
   } ifelse
   /i exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidprojpath) ==
   } if
   /n solid solidnombrefaces def
   i n 1 sub le {
      visibility not solid i solidfacevisible? or {
         currentdict /option known {
            option cvx exec
         } {
            solid i solidcentreface
         } ifelse
         [
            solid 0 i solidgetsommetface
            solid 1 i solidgetsommetface
            vecteur3d normalize3d
            solid i solidnormaleface alpha
         ] false planprojpath
      } {
         newpath 0 0 smoveto
      } ifelse
   } {
      (Error : indice trop grand dans solidprojpath) ==
      quit
   } ifelse
end
} def

%% syntaxe : x y z [normal] bool planprojpath
/planprojpath {
6 dict begin
   /visibility exch def
   %% on calcule l image de la base (I,J,K)
   normalvect_to_orthobase
   /imageK defpoint3d
   /imageJ defpoint3d
   /imageI defpoint3d
   /z exch def
   /y exch def
   /x exch def

   visibility not x y z imageK planvisible? or {
      {ptojpoint 0
      imageI
      imageJ
      imageK
      transformpoint3d
      x y z addv3d
      3dto2d jtoppoint} currentppathtransform
   } {
      newpath
   } ifelse
end
} def

%%%%% ### fin insertion ###

```

8.8 - Courbes de fonctions

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%          fonctions numeriques                      %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### courbeparam ###
/setresolution {
   /resolution exch def
} def
/resolution 200 def
```

```

/courbe_dic 2 dict def
courbe_dic /X {} put
courbe_dic /Y {} put

%% syntaxe : tmin tmax C@urbeparam_
 /C@urbeparam_ {
6 dict begin
   /tmax@ exch def
   /tmin@ exch def
   /t tmin@ def
   /dt tmax@ tmin@ sub resolution 1 sub div def
   tmin@ courbe_dic /X get exec
   pstrickactionR
   tmin@ courbe_dic /Y get exec
   pstrickactionR
   smoveto
   resolution 1 sub
   {
      t courbe_dic /X get exec
      pstrickactionR
      t courbe_dic /Y get exec
      pstrickactionR
      slineto

      /t t dt add store                      %% on incremente
   }
   repeat
   tmax@ courbe_dic /X get exec
   pstrickactionR
   tmax@ courbe_dic /Y get exec
   pstrickactionR
   slineto
end
} def

%% syntaxe : tmin tmax {X} {Y} Courbeparam_
/Courbeparam_ {
   courbe_dic exch /Y exch put
   courbe_dic exch /X exch put
   C@urbeparam_
} def

%% syntaxe : {X} {Y} courbeparam_
/courbeparam_ {
   tmin tmax
   4 -1 roll
   4 -1 roll
   Courbeparam_
} def

%% syntaxe : tmin tmax {X} {Y} Courbeparam
/Courbeparam {
gsave
6 dict begin
   dup isstring
      {
         /option exch def
      }
   if
   courbe_dic exch /Y exch put
   courbe_dic exch /X exch put
   /tmax exch def
   /tmin exch def

   newpath
      tmin courbe_dic /X get exec
      pstrickactionR
      tmin courbe_dic /Y get exec
      pstrickactionR
      smoveto                        %% on commence le chemin
      tmin tmax C@urbeparam_
      starfill

   stockcurrentcpath
   newarrowpath
   currentdict /option known
      {
         /dt tmax tmin sub resolution 1 sub div def
         tmin dt add courbe_dic /X get exec
         tmin dt add courbe_dic /Y get exec
         tmin courbe_dic /X get exec
         tmin courbe_dic /Y get exec
         arrowpath0
         tmax dt sub courbe_dic /X get exec
         tmax dt sub courbe_dic /Y get exec
         tmax courbe_dic /X get exec
         tmax courbe_dic /Y get exec
         currentdict /dt undef
         arrowpath1
         option
         gere_arrowhead
      }
   if

   currentlinewidth 0 eq {} {stroke} ifelse

end
grestore
} def

%% syntaxe : {X} {Y} courbeparam
/courbeparam {
   dup isstring
      {
         tmin tmax
         5 -1 roll
         5 -1 roll
         5 -1 roll
      }
      {
         tmin tmax
         4 -1 roll
         4 -1 roll
      }
   ifelse
   Courbeparam
} def

%% syntaxe : tmin tmax {X} {Y} Courbeparam*
/Courbeparam* {
1 dict begin
   /startest {true} def
   Courbeparam
end
} def

%% syntaxe : {X} {Y} courbeparam*
/courbeparam* {
1 dict begin
   /startest {true} def
   courbeparam
end
} def

%%%%% ### courbe ###
%% syntaxe : {f} courbe
/courbe {
   dup isstring   %% y a-t-il une option de fin de ligne ?
      {
         xmin xmax
         {}
         5 -1 roll
         5 -1 roll
      }
      {
         xmin xmax
         {}
         4 -1 roll
      }
   ifelse
   Courbeparam
```

```
} def

%% syntaxe : mini maxi {f} Courbe
/Courbe {
   dup isstring {
      {}
      3 -1 roll
      3 -1 roll
   } {
      {}
      2 -1 roll
   } ifelse
   Courbeparam
} def

%% syntaxe : {f} courbe_
/courbe_ {
   xmin xmax
   {}
   4 -1 roll
   Courbeparam_
} def

%% syntaxe : mini maxi {f} Courbe_
/Courbe_ {
   {}
   2 -1 roll
   Courbeparam_
} def

%% syntaxe : mini maxi {f} Courbe*
/Courbe* {
1 dict begin
   /startest {true} def
   Courbe
end
} def

%% syntaxe : {f} courbe*
/courbe* {
1 dict begin
   /startest {true} def
   courbe
end
} def

%%%%% ### courbeR2 ###
%% syntaxe : tmin tmax C@urbeR2_
 /C@urbeR2_ {
6 dict begin
   /tmax@ exch def
   /tmin@ exch def
   /t tmin@ def
   /dt tmax@ tmin@ sub resolution 1 sub div def
   tmin@ courbe_dic /X get exec
   pstrickactionR2
   smoveto
   /t t dt add store
   resolution 2 sub
   {
      t courbe_dic /X get exec
      pstrickactionR2
      slineto
      /t t dt add store                      %% on incremente
   }
   repeat
   tmax@ courbe_dic /X get exec
   pstrickactionR2
   slineto
end
} def

%% syntaxe : tmin tmax {X} CourbeR2_
/CourbeR2_ {
   courbe_dic exch /X exch put
   C@urbeR2_
```

```
} def

%% syntaxe : {X} courbeR2_
/courbeR2_ {
   tmin tmax
   3 -1 roll
   3 -1 roll
   CourbeR2_
} def

%% syntaxe : tmin tmax {X} CourbeR2
/CourbeR2+ {
2 dict begin
   /slineto {} def
   /smoveto {} def
   CourbeR2
end
} bind def

/CourbeR2 {
gsave
6 dict begin
   dup isstring
      {
         /option exch def
      }
   if
   courbe_dic exch /X exch put
   /tmax exch def
   /tmin exch def

   newpath
      tmin tmax C@urbeR2_
      starfill
   currentlinewidth 0 eq {} {stroke} ifelse

end
grestore
} def

%% syntaxe : {X} courbeR2
/courbeR2 {
   tmin tmax
   3 -1 roll
   CourbeR2
} def

%% syntaxe : tmin tmax {X} CourbeR2*
/CourbeR2* {
1 dict begin
   /startest {true} def
   CourbeR2
end
} def

%% syntaxe : {X} {Y} courbeR2*
/courbeR2* {
1 dict begin
   /startest {true} def
   courbeR2
end
} def

%%%%% ### courbeR3 ###
%% syntaxe : t1 t2 {f} (option) CourbeR3
/CourbeR3 {
2 dict begin
   dup isstring {
      /option exch def
   } if
   /lafonction exch def
   {lafonction 3dto2d}
   currentdict /option known
      {option}
   if
  CourbeR2
```

```
end
} def

%% syntaxe : {f} (option) CourbeR3
/courbeR3 {
   tmin tmax 3 -1 roll CourbeR3
} def

%%%%% ### fin insertion ###

```

8.9 - Constantes et fonctions mathématiques

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%      fonctions et constantes mathematiques         %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### math ###
%%%%%%%%%%% constantes mathematiques %%%%%%%%%%%%%%

/pi 3.14159 def
/e 2.71828 def

%%%%%%%%%%% fonctions mathematiques %%%%%%%%%%%%%%%

/rd {180 pi div mul} def        %% transforme des rd en degres
/deg {pi mul 180 div} def       %% transforme des degres en rd
/log {ln 10 ln div} def
/Exp {e exch exp} def
/Cos {rd cos} def
/Sin {rd sin} def
/tan {dup sin exch cos div} def
/cotan {dup cos exch sin div} def
/Tan {dup Sin exch Cos div} def
/Cotan {dup Cos exch Sin div} def
/coTan {Cotan} def
/arctan {
dup 0 ge
   {1 atan}
   {neg 1 atan neg}
ifelse
} def
/Arctan {arctan deg} def
/arccos {
   dup
   dup mul neg 1 add sqrt
   exch
   atan
} def
/Arccos {arccos deg} def
/arcsin {
   dup 1 eq {
      90
   } {
      dup
      dup mul neg 1 add sqrt
      atan
      dup 90 lt
         {}
         {360 sub}
      ifelse
   } ifelse
} def
/Arcsin {arcsin deg} def
/cosh {dup Exp exch neg Exp add 2 div} def
/sinh {dup Exp exch neg Exp sub 2 div} def
/tanh {dup sinh exch cosh div} def
/cotanh {dup cosh exch sinh div} def
/argcosh {dup dup mul 1 sub sqrt add ln} def
/argsinh {dup dup mul 1 add sqrt add ln} def
/argtanh {
   setxvar
   x 1 add
   1 x sub
   div
   ln
   2 div
} def
/factorielle {
      dup 0 eq
         {pop 1}
         {dup 1 sub factorielle mul}
      ifelse
} def
/Gauss {
3 dict begin
   /sigma exch def
   /m exch def
   /x exch def
   x m sub dup mul sigma dup mul 2 mul div neg Exp
   2 pi mul sigma dup mul mul sqrt div
end
} def

%%%%% ### max ###
/max {
   2 copy
   lt {exch} if
   pop
} def

%%%%% ### min ###
/min {
   2 copy
   gt {exch} if
   pop
} def

```

8.10 - Divers

Le fichier solides.pro

```
%%%%% ### setcolor ###
%% syntaxe : tableau setcolor
/setcolor {
   dup length 4 eq
      {aload pop setcmykcolor}
      {aload pop setrgbcolor}
   ifelse
} def

%%%%% ### in ###
%% cherche si un elt donne appartient au tableau donne
%% rque : utilise 3 variables locales
%% syntaxe : elt array in --> index boolean
/in {
3 dict begin
   /liste exch def
   /elt exch def
   /i 0 def
   0 false                      %% la reponse a priori
   liste length {
      liste i get elt eq {
         pop pop                %% en enleve la reponse
         i true                 %% pour mettre la bonne
         exit
      } if
      /i i 1 add store
   } repeat
end
} def

%%%%% ### starfill ###
```

```
%% la procedure pour les objets "star"
%% si c est "star" on fait le fillstyle, sinon non
/starfill {
   startest {
      gsave
         clip
         fillstyle
      grestore
      /startest false def
   } if
} def

%%%%% ### addv ###
%% syntaxe : u v addv --> u+v
/addv {         %% xA yA xB yB
   3 1 roll     %% xA yB yA xB
   4 1 roll     %% xB xA yB yA
   add 3 1 roll %% yB+yA xB xA
   add exch
} def

%%%%% ### continu ###
/continu {
   [] 0 setdash
} def

%%%%% ### trigospherique ###
%% passage spherique --> cartesiennes
%% les formules de passage ont été récupérées ici :
%%    http://fr.wikipedia.org/wiki/Coordonn%C3%A9es_polaires
%% syntaxe : r theta phi rtp2xyz -> x y z
/rtp2xyz {
6 dict begin
   /phi exch def
   /theta exch def
   /r exch def
   /x phi sin theta cos mul r mul def
   /y phi sin theta sin mul r mul def
   /z phi cos r mul def
   x y z
end
} def

%% trace d'un arc sur une sphere de centre O
%% syntaxe : r theta1 phi1 r theta2 phi2 arcspherique
/arcspherique {
9 dict begin
   dup isstring {
      /option exch def
   } if
   /phi2 exch def
   /theta2 exch def
   pop
   /phi1 exch def
   /theta1 exch def
   /r exch def
   /n 12 def

   1 theta1 phi1 rtp2xyz /u defpoint3d
   1 theta2 phi2 rtp2xyz /v defpoint3d
   u v vectprod3d u vectprod3d dupp3d norme3d 1 exch div mulv3d /w defpoint3d

   /sinalpha u v vectprod3d norme3d def
   /cosalpha u v scalprod3d def
   /alpha sinalpha cosalpha atan def
   /n 12 def
   /pas alpha n div def

   gsave
      /t pas neg def
      [
         n 1 add {
            /t  t pas add store
            u t cos r mul mulv3d
            w t sin r mul mulv3d
            addv3d
         } repeat
      ]
      currentdict /option known {
         option
      } if
      ligne3d
   grestore
end
} def

%% trace d'un arc sur une sphere de centre O
%% syntaxe : r theta1 phi1 r theta2 phi2 arcspherique
/arcspherique_ {
8 dict begin
   /phi2 exch def
   /theta2 exch def
   pop
   /phi1 exch def
   /theta1 exch def
   /r exch def
   /n 12 def

   1 theta1 phi1 rtp2xyz /u defpoint3d
   1 theta2 phi2 rtp2xyz /v defpoint3d
   u v vectprod3d u vectprod3d dupp3d norme3d 1 exch div mulv3d /w defpoint3d

   /sinalpha u v vectprod3d norme3d def
   /cosalpha u v scalprod3d def
   /alpha sinalpha cosalpha atan def
   /n 12 def
   /pas alpha n div def

   /t pas neg def
   [
      n 1 add {
         /t  t pas add store
         u t cos r mul mulv3d
         w t sin r mul mulv3d
         addv3d
      } repeat
   ] ligne3d_
end
} def

%% trace d'une geodesique sur une sphere de centre O
%% syntaxe : r theta1 phi1 r theta2 phi2 geodesique_sphere
/geodesique_sphere {
13 dict begin
   /phi2 exch def
   /theta2 exch def
   pop
   /phi1 exch def
   /theta1 exch def
   /r exch def
   /n 360 def

   1 theta1 phi1 rtp2xyz /u defpoint3d
   1 theta2 phi2 rtp2xyz /v defpoint3d
   u v vectprod3d u vectprod3d dupp3d norme3d 1 exch div mulv3d /w defpoint3d

   /sinalpha u v vectprod3d norme3d def
   /cosalpha u v scalprod3d def
   /alpha sinalpha cosalpha atan def
   /pas 360 n div def

   gsave
      /t pas neg def
      [
         n 1 add {
            /t  t pas add store
            u t cos r mul mulv3d
            w t sin r mul mulv3d
            addv3d
         } repeat
      ] ligne3d
   grestore
```

```
end
} def


%% syntaxe : A B C trianglespherique --> trace le rtiangle ABC
%% (coordonnees spheriques)
/trianglespherique* {
1 dict begin
   /startest {true} def
   trianglespherique
end
} def

/trianglespherique {
10 dict begin
   /C defpoint3d
   /B defpoint3d
   /A defpoint3d
   gsave
   newpath
      A rtp2xyz 3dto2d smoveto
      A B arcspherique_
      B C arcspherique_
      C A arcspherique_
   closepath
   starfill
   currentlinewidth 0 eq {} {stroke} ifelse
   grestore
end
} def

%%%%% ### fin insertion ###

```

8.11 - Routines sur les tableaux

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%         operations sur les tableaux                %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### duparray ###
/duparray {
1 dict begin
   /table exch def
   table
   [ table aload pop ]
end
} def

%%%%% ### append ###
%% syntaxe : string1 string2 append --> concatene les 2 chaines ou fusionne 2 tableaux
/append {
3 dict begin
   dup isarray {
      /tab2 exch def
      /tab1 exch def
      [ tab1 aload pop tab2 aload pop ]
   } {
      /str2 exch def
      /str1 exch def
      /result str1 length str2 length add string def
      str1 result copy pop
      result str1 length str2 putinterval
      result
   } ifelse
end
} def

%%%%% ### rollparray ###
%% syntaxe : array n rollparray -> array
%% opere une rotation de n sur les couplets du tableau array
```

```
/rollparray {
4 dict begin
   /k exch def
   /table exch def
   /n table length def
   k 0 eq {
       table
   } {
       k 0 ge {
          [ table aload pop 2 {n 1 roll} repeat ]
           k 1 sub
       } {
          [ table aload pop 2 {n -1 roll} repeat ]
           k 1 add
       } ifelse
       rollparray
   } ifelse
end
} def

%%%%% ### bubblesort ###
%% syntaxe : array bubblesort --> array2 trie par ordre croissant
%% code de Bill Casselman
%% http://www.math.ubc.ca/people/faculty/cass/graphics/text/www/
/bubblesort {
4 dict begin
   /a exch def
   /n a length 1 sub def
   n 0 gt {
      % at this point only the n+1 items in the bottom of a remain to
      % the sorted largest item in that blocks is to be moved up into
      % position n
      n {
         0 1 n 1 sub {
            /i exch def
            a i get a i 1 add get gt {
               % if a[i] > a[i+1] swap a[i] and a[i+1]
               a i 1 add
               a i get
               a i a i 1 add get
               % set new a[i] = old a[i+1]
               put
               % set new a[i+1] = old a[i]
               put
            } if
         } for
         /n n 1 sub def
      } repeat
   } if
   a
end
} def

%% syntaxe : array1 doublebubblesort --> array2 array3, array3 est
%% trie par ordre croissant et array2 correspond a la position des
%% indices de depart, ie si array1 = [3 2 4 1], alors array2 = [3 1 0 2]
%% code de Bill Casselman, modifie par jpv, 15/08/2006
%% http://www.math.ubc.ca/people/faculty/cass/graphics/text/www/
/doublebubblesort {
5 dict begin
   /table exch def
   /n table length 1 sub def
   /indices [ 0 1 n {} for ] def
   n 0 gt {
      % at this point only the n+1 items in the bottom of a remain to
      % the sorted largest item in that blocks is to be moved up into
      % position n
      n {
         0 1 n 1 sub {
            /i exch def
            table i get table i 1 add get gt {
               % if a[i] > a[i+1] swap a[i] and a[i+1]
               table i 1 add
               table i get
               table i table i 1 add get
               % set new a[i] = old a[i+1]
               put
               % set new a[i+1] = old a[i]
               put

               indices i 1 add
               indices i get
               indices i indices i 1 add get
               % set new a[i] = old a[i+1]
               put
               % set new a[i+1] = old a[i]
               put
            } if
         } for
         /n n 1 sub def
      } repeat
   } if
   indices table
end
} def

%%%%% ### quicksort ###
%% src : http://www.math.ubc.ca/~cass/graphics/text/www/code/sort.inc
%% code de Bill Casselman, modifie par jpv, 18/10/2007

/qsortdict 8 dict def

qsortdict begin

% args: /comp a L R x
% effect: effects a partition into two pieces [L j] [i R]
%     leaves i j on stack

/partition { 8 dict begin
/x exch def
/j exch def
/i exch def
/a exch def
load /comp exch def
{
  {
    a i get x comp exec not {
      exit
    } if
    /i i 1 add def
  } loop
  {
    x a j get comp exec not {
      exit
    } if
    /j j 1 sub def
  } loop

  i j le {
    % swap a[i] a[j]
    a j a i get
    a i a j get
    put put
    indices j indices i get
    indices i indices j get
    put put
    /i i 1 add def
    /j j 1 sub def
  } if
  i j gt {
    exit
  } if
} loop
i j
end } def

% args: /comp a L R
% effect: sorts a[L .. R] according to comp

/subsort {
% /c a L R
[ 3 1 roll ] 3 copy
```

```
% /c a [L R] /c a [L R]
aload aload pop
% /c a [L R] /c a L R L R
add 2 idiv
% /c a [L R] /c a L R (L+R)/2
3 index exch get
% /c a [L R] /c a L R x
partition
% /c a [L R] i j
% if j > L subsort(a, L, j)
dup
% /c a [L R] i j j
3 index 0 get gt {
  % /c a [L R] i j
  5 copy
  % /c a [L R] i j /c a [L R] i j
  exch pop
  % /c a [L R] i j /c a [L R] j
  exch 0 get exch
  % ... /c a L j
  subsort
} if
% /c a [L R] i j
pop dup
% /c a [L R] i i
% if i < R subsort(a, i, R)
2 index 1 get lt {
  % /c a [L R] i
  exch 1 get
  % /c a i R
  subsort
}{
  4 { pop } repeat
} ifelse
} def

end

% args: /comp a
% effect: sorts the array a
% comp returns truth of x < y for entries in a

/quicksort { qsortdict begin
dup length 1 gt {
% /comp a
dup
% /comp a a
length 1 sub
% /comp a n-1
0 exch subsort
} {
pop pop
} ifelse
end } def

% ----------------------------------------

%% fin du code de Bill Casselman

%% syntaxe : array1 doublebubblesort --> array2 array3, array3 est
%% trie par ordre croissant et array2 correspond a la position des
%% indices de depart, ie si array1 = [3 2 4 1], alors array2 = [3 1 0 2]
%% code de Bill Casselman, modifie par jpv, 18/10/2007
%% http://www.math.ubc.ca/people/faculty/cass/graphics/text/www/
/doublequicksort {
qsortdict begin
   /comp exch
   /a exch def
   a dup length /n exch def
   /indices [0 1 n 1 sub {} for ] def
   dup length 1 gt {
      % /comp a
      dup
      % /comp a a
      length 1 sub
      % /comp a n-1
      0 exch subsort
   } {
      pop pop
   } ifelse
   indices a
end
} def

/comp {lt} def

%%%%% ### apply ###
%% syntaxe : [x1 ... xn] (f) apply --> [f(x1) ... f(xn)]
/apply {
3 dict begin
   dup isstring
      {/fonction exch cvx def}
      {/fonction exch def}
   ifelse
   /liste exch def
   /@i 0 def
   [
   liste length {
      liste @i get fonction
      /@i @i 1 add store
   } repeat
   counttomark
   0 eq
      {pop}
      {]}
   ifelse
end
} def

%% syntaxe : [x1 ... xn] (f) papply
/papply {
3 dict begin
   dup isstring
      {/fonction exch cvx def}
      {/fonction exch def}
   ifelse
   /liste exch def
   /@i 0 def
   [
   liste length 2 idiv {
      liste @i get
      liste @i 1 add get
      fonction
      /@i @i 2 add store
   } repeat
   counttomark
   0 eq
      {pop}
      {]}
   ifelse
end
} def

%% syntaxe : [x1 ... xn] (f) capply
/capply {
3 dict begin
   dup isstring
      {/fonction exch cvx def}
      {/fonction exch def}
   ifelse
   /liste exch def
   /@i 0 def
   [
   liste length 3 idiv {
      liste @i get
      liste @i 1 add get
      liste @i 2 add get
      fonction
      /@i @i 3 add store
   } repeat
   counttomark
   0 eq
```

Le fichier solides.pro (suite)

```
      {pop}
      {[]}
   ifelse
end
} def

%%%%% ### reverse ###
%% syntaxe : array reverse --> inverse l ordre des items dans
%% le tableau
/reverse {
3 dict begin
   /le_tableau exch def
   /n le_tableau length def
   /i n 1 sub def
   [
      n {
         le_tableau i get
         /i i 1 sub store
      } repeat
   ]
end
} def

%% syntaxe : array_points reversep --> inverse l ordre des points dans
%% le tableau
/reversep {
3 dict begin
   /le_tableau exch def
   /n le_tableau length 2 idiv def
   /i n 1 sub def
   [
      n {
         le_tableau i getp
         /i i 1 sub store
      } repeat
   ]
end
} def

%%%%% ### get ###
%% syntaxe : array_points n getp --> le n-ieme point du tableau de
%% points array_points
/getp {
   2 copy
   2 mul get
   3 1 roll
   2 mul 1 add get
} def

%%%%% ### fin insertion ###

```

8.12 - Matrices

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%             matrices                              %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### linear ###
%% syntaxe : M i j any --> depose any dans M en a_ij
/put_ij {
5 dict begin
   /a exch def
   /j exch def
   /i exch def
   /M exch def
   /L M i get_Li def
   L j a put
   M i L put_Li
end
} def
```

```

%% syntaxe : M i j get_ij --> le coeff c_ij
/get_ij {
   3 1 roll   %% j M i
   get_Li     %% j L_i
   exch get
} def

%% syntaxe : M i L put_Li --> remplace dans M la ligne Li par L
/put_Li {
   put
} def

%% syntaxe : M i get_Li --> la ligne Li de M
/get_Li {
   get
} def

%%%%% ### fin insertion ###
```

8.13 - Routines pour le calcul 3d

Le fichier solides.pro

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%          geometrie 3d (calculs)                    %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### dupp3d ###
%% duplique le vecteur 3d
/dupp3d { %% x y z
        3 copy
} def
/dupv3d {dupp3d} def

%%%%% ### angle3d ###
%% syntaxe : vect1 vect2 angle3d
/angle3d {
4 dict begin
   normalize3d /vect2 defpoint3d
   normalize3d /vect1 defpoint3d
   /cosalpha vect1 vect2 scalprod3d def
   /sinalpha vect1 vect2 vectprod3d norme3d def
   sinalpha cosalpha atan
end
} def

%%%%% ### transformpoint3d ###
%% syntaxe : x y z a11 a21 a31 a12 a22 a32 a13 a23 a33
%%    transformpoint3d -> X Y Z
/transformpoint3d {
12 dict begin
   /a33 exch def
   /a23 exch def
   /a13 exch def
   /a32 exch def
   /a22 exch def
   /a12 exch def
   /a31 exch def
   /a21 exch def
   /a11 exch def
   /z   exch def
   /y   exch def
   /x   exch def
   a11 x mul a12 y mul add a13 z mul add
   a21 x mul a22 y mul add a23 z mul add
   a31 x mul a32 y mul add a33 z mul add
end
} def

%%%%% ### normalize3d ###
%% rend le vecteur 3d unitaire. Ne fait rien si u=0
```

```
/unitaire3d { %% x y z
2 dict begin
   /u defpoint3d
   /norme u norme3d def
   norme 0 eq
     {u}
     {u 1 norme div mulv3d
   } ifelse
end
} def
/normalize3d {unitaire3d} def

%%%%% ### mulv ###
%% syntaxe : u a mulv --> au
/mulv {   %% xA, yA, a
   dup          %% xA, yA, a, a
   3 1 roll     %% xA, a, yA, a
   mul 3 1 roll %% ayA, xA, a
   mul exch
} def

%%%%% ### geom3d ###
%% syntaxe : A k1 B k2 barycentre3d -> G, barycentre du systeme
%% [(A, k1) (B, k2)]
/barycentre3d {
4 dict begin
   /k2 exch def
   /B defpoint3d
   /k1 exch def
   /A defpoint3d
   A k1 mulv3d
   B k2 mulv3d
   addv3d
   1 k1 k2 add div mulv3d
end
} def

%% syntaxe : array isobarycentre3d --> G
/isobarycentre3d {
2 dict begin
   /table exch def
   /n table length 3 idiv def
   table 0 getp3d
   1 1 n 1 sub {
       table exch getp3d
       addv3d
   } for
   1 n div mulv3d
end
} def

%% syntaxe : M A alpha hompoint3d -> le point M' tel que AM' = alpha AM
/hompoint3d {
3 dict begin
   /alpha exch def
   /A defpoint3d
   /M defpoint3d
   A M vecteur3d alpha mulv3d A addv3d
end
} def

%% syntaxe : M A sympoint3d -> le point M' tel que AM' = -AM
/sympoint3d {
2 dict begin
   /A defpoint3d
   /M defpoint3d
   A M vecteur3d -1 mulv3d A addv3d
end
} def

%% syntaxe : A u translatepoint3d --> B image de A par la translation de vecteur u
/translatepoint3d {
   addv3d
} def

/scaleOpoint3d {
```

```
6 dict begin
   /k3 exch def
   /k2 exch def
   /k1 exch def
   /z exch def
   /y exch def
   /x exch def
   k1 x mul
   k2 y mul
   k3 z mul
end
} def

% syntaxe : M alpha_x alpha_y alpha_z rotateOpoint3d --> M’
/rotateOpoint3d {
21 dict begin
   /RotZ exch def
   /RotY exch def
   /RotX exch def
   /Zpoint exch def
   /Ypoint exch def
   /Xpoint exch def
   /c1 {RotX cos} bind def
   /c2 {RotY cos} bind def
   /c3 {RotZ cos} bind def
   /s1 {RotX sin} bind def
   /s2 {RotY sin} bind def
   /s3 {RotZ sin} bind def
   /M11 {c2 c3 mul} bind def
   /M12 {c3 s1 mul s2 mul c1 s3 mul sub} bind def
   /M13 {c1 c3 mul s2 mul s1 s3 mul add} bind def
   /M21 {c2 s3 mul} bind def
   /M22 {s1 s2 mul s3 mul c1 c3 mul add} bind def
   /M23 {s3 s2 mul c1 mul c3 s1 mul sub} bind def
   /M31 {s2 neg} bind def
   /M32 {s1 c2 mul} bind def
   /M33 {c1 c2 mul} bind def
   M11 Xpoint mul M12 Ypoint mul add M13 Zpoint mul add
   M21 Xpoint mul M22 Ypoint mul add M23 Zpoint mul add
   M31 Xpoint mul M32 Ypoint mul add M33 Zpoint mul add
end
} def

%%%%% ### vecteur3d ###
%% creation du vecteur AB a partir de A et B
/vecteur3d { %% xA yA zA xB yB zB
6 dict begin
   /zB exch def
   /yB exch def
   /xB exch def
   /zA exch def
   /yA exch def
   /xA exch def
   xB xA sub
   yB yA sub
   zB zA sub
end
}def

%%%%% ### vectprod3d ###
%% produit vectoriel de deux vecteurs 3d
/vectprod3d { %% x1 y1 z1 x2 y2 z2
6 dict begin
   /zp exch def
   /yp exch def
   /xp exch def
   /z exch def
   /y exch def
   /x exch def
   y zp mul z yp mul sub
   z xp mul x zp mul sub
   x yp mul y xp mul sub
end
} def

%%%%% ### scalprod3d ###
```

```
%% produit scalaire de deux vecteurs 3d
/scalprod3d { %% x1 y1 z1 x2 y2 z2
6 dict begin
   /zp exch def
   /yp exch def
   /xp exch def
   /z exch def
   /y exch def
   /x exch def
   x xp mul y yp mul add z zp mul add
end
} def

%%%%% ### papply3d ###
%% syntaxe : [A1 ... An] (f) papply3d --> [f(A1) ... f(An)]
/papply3d {
3 dict begin
   /fonction exch def
   /liste exch def
   /i 0 def
   [
   liste length 3 idiv {
      liste i get
      liste i 1 add get
      liste i 2 add get
      fonction
      /i i 3 add store
   } repeat
   counttomark
   0 eq
      {pop}
      {]}
   ifelse
end
} def

%%%%% ### defpoint3d ###
%% creation du point A a partir de xA yA yB et du nom /A
/defpoint3d { %% xA yA zA /nom
1 dict begin
   /memo exch def
   [ 4 1 roll ] cvx memo exch
end def
}def

%%%%% ### distance3d ###
/distance3d { %% A B
   vecteur3d norme3d
} def

%%%%% ### get3d ###
/getp3d { %% [tableau de points 3d] i --> donne le ieme point du tableau
   2 copy 2 copy
   3 mul get
   5 1 roll
   3 mul 1 add get
   3 1 roll
   3 mul 2 add get
} def

%%%%% ### norme3d ###
%% norme d un vecteur 3d
/norme3d { %% x y z
3 dict begin
   /z exch def
   /y exch def
   /x exch def
   x dup mul y dup mul add z dup mul add sqrt
end
} def

%%%%% ### mulv3d ###
%% (scalaire)*(vecteur 3d) Attention : dans l autre sens !
/mulv3d { %% x y z lambda
4 dict begin
   /lambda exch def
   /z exch def
   /y exch def
   /x exch def
   x lambda mul
   y lambda mul
   z lambda mul
end
} def

%%%%% ### addv3d ###
%% addition de deux vecteurs 3d
/addv3d { %% x1 y1 z1 x2 y2 z2
6 dict begin
   /zp exch def
   /yp exch def
   /xp exch def
   /z exch def
   /y exch def
   /x exch def
   x xp add
   y yp add
   z zp add
end
} def

%%%%% ### milieu3d ###
/milieu3d { %% A B --> I le milieu de [AB]
   addv3d 0.5 mulv3d
} def

%%%%% ### fin insertion ###

```

8.14 - Routines pour le dessin 3d

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%          geometrie 3d (dessins)                    %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### point3d ###
/point3d { %% A
   3dto2d point
} def

/points3d { %% tableau de points3d
   tab3dto2d points
} def

%%%%% ### ligne3d ###
%% [tableau de points3d] option --> trace la ligne brisee
/ligne3d {
1 dict begin
   dup isstring
      {/option exch def}
   if
   tab3dto2d
   currentdict /option known
      {option}
   if
   ligne
end
} def

%% [tableau de points3d] option --> trace la ligne brisee
/ligne3d_ {
1 dict begin
   dup isstring
      {/option exch def}
   if
   tab3dto2d
   currentdict /option known
```

```
      {option}
   if
   ligne_
end
} def

%%%%% ### tab3dto2d ###
%% transforme un tableau de points 3d en tableau de points 2d
/tab3dto2d {
2 dict begin
   /T exch def
   /n T length def
   [ T aload pop
   n 1 sub -1 n 3 idiv 2 mul
   { 1 dict begin
   /i exch def
   3dto2d i 2 roll
   end } for ]
end
} def

%%%%% ### polygone3d ###
/polygone3d { %% tableau de points3d
   tab3dto2d polygone
} def

/polygone3d* { %% tableau de points3d
   tab3dto2d polygone*
} def

%%%%% ### fin insertion ###

```

8.15 - Gestion des chemins définis par des chaînes de caractères

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                 gestion du texte                   %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### pathtext ###
%% syntaxe : string x y initp@thtext
 /initp@thtext {
7 dict begin
   /y exch def
   /x exch def
   /str exch def
   str 0 0 show_dim
   /wy exch def
   /wx exch def
   /lly exch def
   /llx exch def
   pop pop pop
   newpath
      x y  smoveto
} def
 /closep@thtext {
      str true charpath
end
} def

%% syntaxe : string x y cctext_
/cctext_ {
   initp@thtext
   llx wx add lly wy add -.5 mulv rmoveto
   closep@thtext
} def

/brtext_ {
   initp@thtext
   hadjust 0 rmoveto
   llx neg 0 rmoveto
   closep@thtext
} def

/bbtext_ {
   initp@thtext
   0 0 rmoveto
   0 0 rmoveto
   closep@thtext
} def

/bltext_ {
   initp@thtext
   hadjust neg 0 rmoveto
   wx neg 0 rmoveto
   closep@thtext
} def

/bctext_ {
   initp@thtext
   0 0 rmoveto
   wx llx add -.5 mul 0 rmoveto
   closep@thtext
} def

/ubtext_ {
   initp@thtext
   0 vadjust rmoveto
   0 lly neg rmoveto
   closep@thtext
} def

/urtext_ {
   initp@thtext
   hadjust vadjust rmoveto
   llx neg lly neg rmoveto
   closep@thtext
} def

/ultext_ {
   initp@thtext
   hadjust neg vadjust rmoveto
   wx neg lly neg rmoveto
   closep@thtext
} def

/uctext_ {
   initp@thtext
   0 vadjust rmoveto
   llx wx add -.5 mul lly neg rmoveto
   closep@thtext
} def

/drtext_ {
   initp@thtext
   hadjust vadjust neg rmoveto
   llx neg wy neg rmoveto
   closep@thtext
} def

/dbtext_ {
   initp@thtext
   0 vadjust neg rmoveto
   0 wy neg rmoveto
   closep@thtext
} def

/dltext_ {
   initp@thtext
   hadjust neg vadjust neg rmoveto
   wx neg wy neg rmoveto
   closep@thtext
} def

/dctext_ {
   initp@thtext
   0 vadjust neg rmoveto
   llx wx add -2 div wy neg rmoveto
   closep@thtext
} def

/crtext_ {
   initp@thtext
   hadjust 0 rmoveto
   llx neg lly wy add -2 div rmoveto
   closep@thtext
} def

/cbtext_ {
   initp@thtext
   0 0 rmoveto
   0 lly wy add -2 div rmoveto
   closep@thtext
} def

/cltext_ {
   initp@thtext
   hadjust neg 0 rmoveto
   wx neg lly wy add -2 div rmoveto
   closep@thtext
} def

/cctext_ {
   initp@thtext
   0 0 rmoveto
   llx wx add lly wy add -.5 mulv rmoveto
   closep@thtext
} def

%%%%% ### fin insertion ###

```

8.16 - Routines pour le calcul sur le type solid

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%             bibliotheque sur les solides           %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### solide ###
%% solid = [Sommets Faces Colors_Faces InOut_Table]
/solidgetpointstable {
   0 get
} def

/solidgetfaces {
   1 get
} def

/solidgetface {
1 dict begin
   /i exch def
   solidgetfaces i get
end
} def

/solidgetfcolors {
   2 get
} def

%% syntaxe : solid i solidgetfcolor --> str
/solidgetfcolor {
1 dict begin
   /i exch def
   solidgetfcolors i get
end
} def

%% syntaxe : solid i str solidputfcolor --> -
```

```
/solidputfcolor {
2 dict begin
   /str exch def
   /i exch def
   solidgetfcolors i str put
end
} def

/solidgetinouttable {
   3 get
} def

/solidputpointstable {
   0 exch put
} def

/solidputfaces {
   1 exch put
} def

/solidputfcolors {
   2 exch put
} def

/solidputinouttable {
   3 exch put
} def

%% syntaxe : any issolid --> booleen, vrai si any est de type solid
/issolid {
1 dict begin
   /candidat exch def
   candidat isarray {
      candidat length 4 eq {
         candidat 0 get isarray
         candidat 1 get isarray and
         candidat 2 get isarray and
         candidat 3 get isarray and
      } {
         false
      } ifelse
   } {
      false
   } ifelse
end
} def

/dupsolid {
5 dict begin
   /solid exch def
   /S solid solidgetpointstable def
   /F solid solidgetfaces def
   /FC solid solidgetfcolors def
   /IO solid solidgetinouttable def
   solid
   [
      S duparray exch pop
      F duparray exch pop
      FC duparray exch pop
      IO duparray exch pop
   ]
end
} def

%% syntaxe : solid array solidputinfaces --> -
/solidputinfaces {
4 dict begin
   /facesinternes exch def
   /solid exch def
   /n2 facesinternes length def
   /IO solid solidgetinouttable def
   /facesexternes solid solidgetoutfaces def
   /n1 facesexternes length def
   solid
      [facesexternes aload pop facesinternes aload pop]
      solidputfaces
   IO 0 0 put
   IO 1 n1 1 sub put
   IO 2 n1 put
   IO 3 n1 n2 add 1 sub put
end
} def

%% syntaxe : solid array solidputoutfaces --> -
/solidputoutfaces {
4 dict begin
   /facesexternes exch def
   /solid exch def
   /n1 facesexternes length def
   /IO solid solidgetinouttable def
   /facesinternes solid solidgetinfaces def
   /n2 facesinternes length def
   solid
      [facesexternes aload pop facesinternes aload pop]
      solidputfaces
   IO 0 0 put
   IO 1 n1 1 sub put
   IO 2 n1 put
   IO 3 n1 n2 add 1 sub put
end
} def

%% syntaxe : solid array solidputoutfaces --> -
/solidputoutfaces {
4 dict begin
   /facesexternes exch def
   /solid exch def
   /n1 facesexternes length def
   /IO solid solidgetinouttable def
   /facesinternes solid solidgetinfaces def
   /n2 facesinternes length def
   solid
      [facesexternes aload pop facesinternes aload pop]
      solidputfaces
   IO 0 0 put
   IO 1 n1 1 sub put
   IO 2 n1 put
   IO 3 n1 n2 add 1 sub put
end
} def

/solidnombreinfaces {
1 dict begin
   /solid exch def
   solid solidwithinfaces {
      /IO solid solidgetinouttable def
      IO 3 get IO 2 get sub 1 add
   } {
      0
   } ifelse
end
} def

/solidnombreoutfaces {
1 dict begin
   /solid exch def
   /IO solid solidgetinouttable def
   IO 1 get IO 0 get sub 1 add
end
} def

%% syntaxe : solid solidgetinfaces --> array
/solidgetinfaces {
4 dict begin
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidgetinfaces) ==
      quit
   } if
   solid solidwithinfaces {
      /IO solid solidgetinouttable def
      /F solid solidgetfaces def
      /n1 IO 2 get def
      /n2 IO 3 get def
      /n n2 n1 sub 1 add def
      F n1 n getinterval
   } {
      []
   } ifelse
end
} def

%% syntaxe : solid solidgetoutfaces --> array
/solidgetoutfaces {
4 dict begin
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidgetoutfaces) ==
      quit
   } if
   /IO solid solidgetinouttable def
   /F solid solidgetfaces def
   /n1 IO 0 get def
   /n2 IO 1 get def
   /n n2 n1 sub 1 add def
   F n1 n getinterval
end
} def

%%%%% ### fin insertion ###

%% /tracelignedeniveau? false def
%% /hauteurlignedeniveau 1 def
%% /couleurlignedeniveau {rouge} def
%% /linewidthlignedeniveau 4 def
%%
%% /solidgrid true def
%% /aretescachees true def
%% /defaultsolidmode 2 def

%%%%% ### newsolid ###
%% syntaxe : newsolid --> depose le solide nul sur la pile
/newsolid {
   [] [] generesolid
} def

%%%%% ### generesolid ###
/generesolid {
2 dict begin
   /F exch def
   /S exch def
   [S F [F length {()} repeat] [0 F length 1 sub -1 -1]]
end
} def

%%%%% ### nullsolid ###
%% syntaxe : solide nullsolid -> booleen, vrai si le solide est nul
/nullsolid {
1 dict begin
   /candidat exch def
   candidat issolid not {
      (Error type argument dans "nullsolid") ==
      quit
   } if
   candidat solidgetpointstable length 0 eq {
      true
   } {
      false
   } ifelse
end
} def

%%%%% ### solidnombreoutfaces ###
/solidnombreoutfaces {
4 dict begin
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidnombreoutfaces) ==
      quit
   } if
   solid nullsolid {
      0
   } {
      /IO solid solidgetinouttable def
      IO 1 get
      IO 0 get sub
      1 add
   } ifelse
end
} def

%%%%% ### solidnombreinfaces ###
/solidnombreinfaces {
4 dict begin
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidnombreinfaces) ==
      quit
   } if
   solid solidwithinfaces {
      /IO solid solidgetinouttable def
      IO 3 get
      IO 2 get sub
      1 add
   } {
      0
   } ifelse
end
} def

%%%%% ### solidtests ###
%% syntaxe : solid solidwithinfaces --> bool, true si le solide est vide
/solidwithinfaces {
2 dict begin
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidwithinfaces) ==
      quit
   } if
   /table solid solidgetinouttable def
   table 2 get -1 ne {
      true
   } {
      false
   } ifelse
end
} def

%%%%% ### solidgetsommet ###
%% syntaxe : solid i j solidgetsommetface --> sommet i de la face j
/solidgetsommetface {
6 dict begin
   /j exch def
   /i exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidgetsommetface) ==
      quit
   } if
   /table_faces solid solidgetfaces def
   /table_sommets solid solidgetpointstable def
   /k table_faces j get i get def
   table_sommets k getp3d
end
} def

%% syntaxe : solid i solidgetsommetsface --> array, tableau des
%% sommets de la face i du solide
/solidgetsommetsface {
6 dict begin
   /i exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidgetsommetsface) ==
      quit
   } if
   /table_faces solid solidgetfaces def
   /table_sommets solid solidgetpointstable def
   /table_indices table_faces i get def
   [
      0 1 table_indices length 1 sub {
         /j exch def
         table_sommets table_indices j get getp3d
      } for
   ]
end
} def

%% syntaxe : solid i solidgetsommet --> sommet i du solide
/solidgetsommet {
3 dict begin
   /i exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidgetsommet) ==
      quit
   } if
   /table_sommets solid solidgetpointstable def
   table_sommets i getp3d
end
} def

%%%%% ### solidcentreface ###
%% syntaxe : solid i solidcentreface --> M
/solidcentreface {
   solidgetsommetsface isobarycentre3d
} def

%%%%% ### solidnombre ###
/solidnombresommets {
   solidgetpointstable length 3 idiv
} def

/solidfacenombresommets {
   solidgetface length
} def

/solidnombrefaces {
   solidgetfaces length
} def

%%%%% ### solidshowsommets ###
/solidshowsommets {
8 dict begin
   dup issolid not {
      %% on a un argument
      /option exch def
   } if
   /sol exch def
   /n sol solidnombresommets def
   /m sol solidnombrefaces def
   currentdict /option known not {
      /option [0 1 n 1 sub {} for] def
   } if
   0 1 option length 1 sub {
      /k exch def
      option k get /i exch def       %% indice du sommet examine
      sol i solidgetsommet point3d
   } for
end
} def

%%%%% ### solidnumsommets ###
/solidnumsommets {
8 dict begin
   Font findfont 10 scalefont setfont
   dup issolid not {
      %% on a un argument
      /option exch def
   } if
   /sol exch def
   /n sol solidnombresommets def
   /m sol solidnombrefaces def
   currentdict /option known not {
      /option [0 1 n 1 sub {} for] def
   } if
   /result [
      n {false} repeat
   ] def
   0 1 option length 1 sub {
      /k exch def
      option k get /i exch def       %% indice du sommet examine
      0 1 m 1 sub {
         /j exch def %% indice de la face examinee
         i sol j solidgetface in exch pop {
            %% le sommet i est dans la face j
            exit
         } if
      } for
      %% le sommet i est dans la face j
      sol j solidcentreface /G defpoint3d
      sol i solidgetsommet /S defpoint3d
      i (   ) cvs
      G S vecteur3d normalize3d
      15 dup ptojpoint pop
      mulv3d
      S addv3d
      3dto2d cctext
   } for
end
} def

%%%%% ### gestionsolidmode ###
%% table = [ [vars] [mode0] [mode1] [mode2] [mode3] [mode4] ]
/gestionsolidmode {
5 dict begin
   /table exch def
   dup xcheck {
      /mode exch def
   } {
      dup isarray {
         /tableaffectation exch def
 /mode -1 def
      } {
         /mode defaultsolidmode def
      } ifelse
   } ifelse
   /vars table 0 get def
   /nbvars vars length def
   mode 0 ge {
      /tableaffectation table mode 1 add 5 min get def
   } if
   0 1 nbvars 1 sub {
      /i exch def
      vars i get
      tableaffectation i get
   } for
   nbvars
end
   {def} repeat
} def

%%%%% ### solidfuz ###
%% syntaxe : solid1 solid2 solidfuz -> solid
/solidfuz {
5 dict begin
   /solid2 exch def
   /solid1 exch def
   /S1 solid1 solidgetpointstable def
   /S2 solid2 solidgetpointstable def
   /n S1 length 3 idiv def

   %% les sommets
   /S S1 S2 append def

   %% les faces internes et leurs couleurs
   /FI1 solid1 solidgetinfaces def
   /FIC1 solid1 solidgetincolors def
   solid2 solidnombreinfaces 0 eq {
      /FI2 [] def
      /FIC2 [] def
   } {
      /FI2 solid2 solidgetinfaces {{n add} apply} apply def
      /FIC2 solid2 solidgetincolors def
   } ifelse
   /FI [FI1 aload pop FI2 aload pop] def
   /FIC [FIC1 aload pop FIC2 aload pop] def

   %% les faces externes et leurs couleurs
   /FO1 solid1 solidgetoutfaces def
   /FOC1 solid1 solidgetoutcolors def
   /FO2 solid2 solidgetoutfaces {{n add} apply} apply def
   /FOC2 solid2 solidgetoutcolors def
   /FO [FO1 aload pop FO2 aload pop] def
   /FOC [FOC1 aload pop FOC2 aload pop] def

   /F [FO aload pop FI aload pop] def
   /FC [FOC aload pop FIC aload pop] def
   /IO [0 FO length 1 sub dup 1 add dup FI length add 1 sub] def

   S F generesolid
   dup FC solidputfcolors
   dup IO solidputinouttable
end
} def

%%%%% ### solidnormaleface ###
%% syntaxe : solid i solidnormaleface --> u, vecteur normale a la
%% face d indice i du solide
/solidnormaleface {
4 dict begin
   /i exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidgetsommetface) ==
      quit
   } if
%%    solid 0 i solidgetsommetface /G defpoint3d
%%    G
%%    solid 1 i solidgetsommetface
%%    vecteur3d
%%    G
%%    solid 2 i solidgetsommetface
%%    vecteur3d

   /n solid i solidfacenombresommets def
   solid i solidcentreface /G defpoint3d
  %% debug %%   G 3dto2d point
   G
   solid 0 i solidgetsommetface
   /A defpoint3d
  %   gsave bleu A point3d grestore
   A
   vecteur3d normalize3d
   G
   solid 1 i solidgetsommetface
   /A defpoint3d
  %   gsave orange A point3d grestore
   A
   vecteur3d normalize3d
   vectprod3d
   /resultat defpoint3d
   resultat normalize3d
end
} def

%%%%% ### solidtransform ###
%% syntaxe : solid1 {f} solidtransform --> solid2, solid2 est le
%% transforme de solid1 par la transformation f : R^3 -> R^3
/solidtransform {
3 dict begin
   /f exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidtransform) ==
      quit
   } if
   /les_sommets
      solid solidgetpointstable {f} papply3d
   def
   solid les_sommets solidputpointstable
   solid
end
} def

%%%%% ### solidputcolor ###
%% syntaxe : solid i string solidputfcolor
/solidputfcolor {
3 dict begin
   /str exch def
   /i exch def
   /solid exch def
   /FC solid solidgetfcolors def
   i FC length lt {
      FC i str put
   } if
end
} def

%% syntaxe : solid solidgetincolors --> array
/solidgetincolors {
3 dict begin
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidgetincolors) ==
      quit
   } if
   solid solidwithinfaces {
      /fcol solid solidgetfcolors def
      /IO solid solidgetinouttable def
      /n1 IO 2 get def
      /n2 IO 3 get def
      /n n2 n1 sub 1 add def
      fcol n1 n getinterval
   } {
      []
   } ifelse
end
} def

%% syntaxe : solid solidgetoutcolors --> array
/solidgetoutcolors {
3 dict begin
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidgetoutcolors) ==
      quit
   } if
   /fcol solid solidgetfcolors def
   /IO solid solidgetinouttable def
   /n1 IO 0 get def
   /n2 IO 1 get def
   /n n2 n1 sub 1 add def
   fcol n1 n getinterval
end
} def

%% syntaxe : solid array solidputincolors --> -
/solidputincolors {
4 dict begin
   /newcolorstable exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidputincolors) ==
      quit
   } if
   /n newcolorstable length def
   n solid solidnombreinfaces ne {
      (Error : mauvaise longueur de tableau dans solidputincolors) ==
      quit
   } if
   n 0 ne {
      /FC solid solidgetfcolors def
      /IO solid solidgetinouttable def
      /n1 IO 2 get def
      FC n1 newcolorstable putinterval
   } if
end
} def

%% syntaxe : solid array solidputoutcolors --> -
/solidputoutcolors {
4 dict begin
   /newcolorstable exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidputoutcolors) ==
      quit
   } if
   /n newcolorstable length def
   n solid solidnombreoutfaces ne {
      (Error : mauvaise longueur de tableau dans solidputoutcolors) ==
      quit
   } if
   n 0 ne {
      /FC solid solidgetfcolors def
      /IO solid solidgetinouttable def
      /n1 IO 0 get def
      FC n1 newcolorstable putinterval
   } if
end
} def

%% syntaxe : solid str outputcolors
/outputcolors {
5 dict begin
   /color exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans inoutputcolors) ==
      quit
   } if
   /n solid solidnombreoutfaces def
   solid [ n {color} repeat ] solidputoutcolors
end
} def

%% syntaxe : solid str inputcolors
/inputcolors {
5 dict begin
   /color exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans inoutputcolors) ==
      quit
   } if
   /n solid solidnombreinfaces def
   solid [ n {color} repeat ] solidputincolors
end
} def

%% syntaxe : solid str1 str2 inoutputcolors
/inoutputcolors {
5 dict begin
   /colout exch def
   /colin exch def
   /solid exch def
   solid colin inputcolors
   solid colout outputcolors
end
} def

%%%%% ### solidputhuecolors ###
%% syntaxe : solid table solidputhuecolors --> -
```

```
/solidputhuecolors {
1 dict begin
   2 copy pop
   solidgetinouttable /IO exch def
   IO 0 get
   IO 1 get
   s@lidputhuec@l@rs
end
} def

/solidputinhuecolors {
2 dict begin
   /table exch def
   /solid exch def
   solid solidgetinouttable /IO exch def
   solid solidwithinfaces {
      solid table
      IO 2 get
      IO 3 get
      s@lidputhuec@l@rs
   } if
end
} def

/solidputinouthuecolors {
1 dict begin
   2 copy pop
   solidgetinouttable /IO exch def
   IO 0 get
   IO 3 get IO 1 get max
   s@lidputhuec@l@rs
end
} def

%% syntaxe : solid table n1 n2 s@lidputhuec@l@rs --> -
%% affecte les couleurs des faces d indice n1 a n2 du solid solid, par
%% un degrade defini par la table.
 /s@lidputhuec@l@rs {
9 dict begin
   /n2 exch def
   /n1 exch def
   /table exch def
   /solid exch def
   /n n2 n1 sub def

   table length 2 eq {
       /a0 table 0 get def
       /a1 table 1 get def
       a1 isstring {
          /lacouleurdepart {
             gsave
                [a0 cvx exec] length 0 eq {
                   a0 cvx exec currentrgbcolor
                } {
                   a0 cvx exec
                } ifelse
             grestore
          } def
          /lacouleurarrivee {
             gsave
                [a1 cvx exec] length 0 eq {
                   a1 cvx exec currentrgbcolor
                } {
                   a1 cvx exec
                } ifelse
             grestore
          } def
          /table [lacouleurdepart lacouleurarrivee] def
       } {
          /A {a0 i a1 a0 sub mul n 1 sub div add} def
          /B {1} def
          /C {1} def
          /D {} def
          /espacedecouleurs (sethsbcolor) def
       } ifelse
   } if

   table length 4 eq {
       /a0 table 0 get def
       /a1 table 1 get def
       /A {a0 i a1 a0 sub mul n 1 sub div add} def
       /B table 2 get def
       /C table 3 get def
       /D {} def
       /espacedecouleurs (sethsbcolor) def
   } if

   table length 6 eq {
       /a0 table 0 get def
       /b0 table 1 get def
       /c0 table 2 get def
       /a1 table 3 get def
       /b1 table 4 get def
       /c1 table 5 get def
       /A {a0 i a1 a0 sub mul n 1 sub div add} def
       /B {b0 i b1 b0 sub mul n 1 sub div add} def
       /C {c0 i c1 c0 sub mul n 1 sub div add} def
       /D {} def
       /espacedecouleurs (setrgbcolor) def
   } if

   table length 7 eq {
       /a0 table 0 get def
       /b0 table 1 get def
       /c0 table 2 get def
       /a1 table 3 get def
       /b1 table 4 get def
       /c1 table 5 get def
       /A {a0 i a1 a0 sub mul n 1 sub div add} def
       /B {b0 i b1 b0 sub mul n 1 sub div add} def
       /C {c0 i c1 c0 sub mul n 1 sub div add} def
       /D {} def
       /espacedecouleurs (sethsbcolor) def
   } if

   table length 8 eq {
       /a0 table 0 get def
       /b0 table 1 get def
       /c0 table 2 get def
       /d0 table 3 get def
       /a1 table 4 get def
       /b1 table 5 get def
       /c1 table 6 get def
       /d1 table 7 get def
       /A {a0 i a1 a0 sub mul n 1 sub div add} def
       /B {b0 i b1 b0 sub mul n 1 sub div add} def
       /C {c0 i c1 c0 sub mul n 1 sub div add} def
       /D {d0 i d1 d0 sub mul n 1 sub div add} def
       /espacedecouleurs (setcmykcolor) def
   } if

   n1 1 n2 {
      /i exch def
      solid i
      [A B C D] espacedecouleurs astr2str
      solidputfcolor
   } for

end
} def

%%%%% ### solidrmface ###
%% syntaxe : solid i solidrmface -> -
/solidrmface {
5 dict begin
   /i exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidrmface) ==
      quit
   } if
   %% on enleve la face
   /F solid solidgetfaces def
   F length 1 sub i lt {
      (Error : indice trop grand dans solidrmface) ==
      quit
   } if
   [
      0 1 F length 1 sub {
         /j exch def
         i j ne {
            F j get
         } if
      } for
   ]
   /NF exch def
   solid NF solidputfaces
   %% on enleve la couleur correspondante
   /FC solid solidgetfcolors def
   [
      0 1 FC length 1 sub {
         /j exch def
         i j ne {
            FC j get
         } if
      } for
   ]
   /NFC exch def
   solid NFC solidputfcolors
   %% on ajuste la table inout
   /IO solid solidgetinouttable def
   solid i solidisoutface {
      IO 1 IO 1 get 1 sub put
      solid solidwithinfaces {
         IO 2 IO 2 get 1 sub put
         IO 3 IO 3 get 1 sub put
      } if
   } if
   solid i solidisinface {
      IO 1 IO 1 get 1 sub put
      IO 2 IO 2 get 1 sub put
      IO 3 IO 3 get 1 sub put
   } if
   solid IO solidputinouttable
end
} def

%% syntaxe : solid table solidrmfaces --> -
/solidrmfaces {
2 dict begin
   /table exch bubblesort reverse def
   /solid exch def
   table {solid exch solidrmface} apply
end
} def

%%%%% ### videsolid ###
%% syntaxe : solid videsolid -> -
/videsolid {
5 dict begin
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans videsolid) ==
      quit
   } if
   solid solidwithinfaces not {
      /IO solid solidgetinouttable def
      /FE solid solidgetfaces def
      /n FE length def
      IO 2 n put
      IO 3 2 n mul 1 sub put
      %% on inverse chaque face
      /FI FE {reverse} apply def
      solid FE FI append solidputfaces
      %% et on rajoute autant de couleurs vides que de faces
      /FEC solid solidgetfcolors def
      /FIC [FI length {()} repeat] def
      solid FEC FIC append solidputfcolors
      solid IO solidputinouttable
   } if
end
} def

%%%%% ### solidnumfaces ###
%% syntaxe : solid array solidnumfaces
%% syntaxe : solid array bool solidnumfaces
%% array, le tableau des indices des faces a numeroter, est optionnel
%% si bool=true, on ne numerote que les faces visibles
/solidnumfaces {
5 dict begin
   dup isbool {
      /bool exch def
   } {
      /bool true def
   } ifelse
   setTimes
   dup issolid not {
      %% on a un argument
      /option exch def
   } if
   /sol exch def
   /n sol solidnombrefaces def
   currentdict /option known not {
      /option [0 1 n 1 sub {} for] def
   } if

   0 1 option length 1 sub {
      /i exch def
      /j option i get def
      j (     ) cvs sol j bool cctextp3d
   } for
end
} def

%%%%% ### creusesolid ###
%% syntaxe : solid creusesolid -> -
/creusesolid {
5 dict begin
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans creusesolid) ==
      quit
   } if
   %% on enleve le fond et le chapeau
   solid 1 solidrmface
   solid 0 solidrmface
   %% on inverse chaque face
   solid videsolid
end
} def

%%%%% ### fin insertion ###

```

8.17 - Routines pour le dessin d'un objet de type solid

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                 dessin des solides                %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### solidisinface ###
%% syntaxe : solid i solidisinface --> bool
%% true si i est l indice d une face interne, false sinon
/solidisinface {
4 dict begin
   /i exch def
   solidgetinouttable /IO exch def
   /n1 IO 2 get def
   /n2 IO 3 get def
   n1 i le
   i n2 le and
end
} def

%%%%% ### solidisoutface ###
%% syntaxe : solid i solidisoutface --> bool
%% true si i est l indice d une face externe, false sinon
/solidisoutface {
4 dict begin
   /i exch def
   solidgetinouttable /IO exch def
   /n1 IO 0 get def
   /n2 IO 1 get def
   n1 i le
   i n2 le and
end
} def

%%%%% ### planvisible ###
%% syntaxe : A k planvisible? --> true si le plan est visible
/planvisible? {
4 dict begin
   /normale_plan defpoint3d
   /origine defpoint3d
   /ligne_de_vue {
      origine
      GetCamPos
      vecteur3d
   } def
   ligne_de_vue normale_plan scalprod3d 0 gt
end
} def

%%%%% ### drawsolid ###
%% syntaxe : solid i solidfacevisible? --> true si la face est visible
/solidfacevisible? {
4 dict begin
   /i exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans solidgetsommetface) ==
      quit
   } if
   solid i solidgetface length 2 le {
      true
   } {
      /ligne_de_vue {
         solid i solidcentreface
         GetCamPos
         vecteur3d
      } def

      /normale_face {
         solid i solidnormaleface
      } def
      ligne_de_vue normale_face scalprod3d 0 gt
   } ifelse
end
} def

%% syntaxe : solid i affectecouleursolid_facei --> si la couleur de
%% la face i est definie, affecte fillstyle a cette couleur
/affectecouleursolid_facei {
3 dict begin
   /i exch def
   /solid exch def
   solid solidgetfcolors /FC exch def
   FC length 1 sub i ge {
      FC i get length 1 ge {
         /fillstyle FC i get ( fill) append cvx
         true
      } {
         false
      } ifelse
   } {
      false
   } ifelse
end
{def} if
} def

%% syntaxe : A solid i dessinefacecachee
/dessinefacecachee {
6 dict begin
   /i exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans dessinefacecachee) ==
      quit
   } if
   /A exch def

   /F solid solidgetfaces def
   /S solid solidgetpointstable def

   solid i solidfacevisible? not {
      %% face cachee => on prend chacune des aretes de la face et on
      %% regarde si elle est deja dessinee.
      4 dict begin
         /n F i get length def %% nb de sommets de la face
         0 1 n 1 sub {
         /k exch def
            /k1 F i k get_ij def              %% indice sommet1
            /k2 F i k 1 add n mod get_ij def  %% indice sommet2
            A k1 k2 get_ij not {
               gsave
                  currentlinewidth .5 mul setlinewidth
                  pointilles
                  [S k1 getp3d
                  S k2 getp3d] ligne3d
                  A k1 k2 true put_ij
                  A k2 k1 true put_ij
               grestore
            } if
         } for
      end
   } if
end
} def

%% syntaxe : A solid i dessinefacevisible
/dessinefacevisible {
7 dict begin
   /i exch def
   /solid exch def
   /A exch def
   solid issolid not {
      (Error : mauvais type d argument dans dessinefacevisible) ==
      quit
   } if
   /F solid solidgetfaces def
   /S solid solidgetpointstable def

   solid i solidfacevisible? {
      /n F i get length def %% nb de sommets de la face

      startest {
         %% choix de la couleur
         /lightcolor where {
            pop
            /coeff
               lightintensity
               solid i solidnormaleface normalize3d
               solid i solidcentreface lightsrc vecteur3d normalize3d
               scalprod3d mul
               0 max 1 min
            def
            /fillstyle {
                lightcolor {coeff mul} apply setcolor fill
            } def
            solidgrid not {
                lightcolor {coeff mul} apply setcolor
             } if
          } {
             /lightsrc where {
                pop
                /coeff
                   lightintensity
                   solid i solidnormaleface normalize3d
                   solid i solidcentreface lightsrc vecteur3d normalize3d
                   scalprod3d mul
                   0 max 1 min
                def
                /lacouleur [
                   gsave
                      solid solidgetfcolors i get cvx exec currentrgbcolor
                   grestore
                ] def
                /fillstyle {
                   lacouleur {coeff mul} apply setcolor fill
                } def
                solidgrid not {
                   lacouleur {coeff mul} apply setcolor
                } if
             } {
%                  solid F i get length affectecouleursolid_ncotes
                solid i affectecouleursolid_facei
             } ifelse

          } ifelse
       } if

       /face_a_dessiner [  %% face visible : F [i]
          0 1 n 1 sub {
             /j exch def
             solid j i solidgetsommetface
          } for
       ] def
       face_a_dessiner polygone3d
       /lignedeniveau [] def

       %% trace de la ligne de niveau
       tracelignedeniveau? {
          gsave
             linewidthlignedeniveau setlinewidth
             couleurlignedeniveau
             0 1 n 1 sub {
                /j exch def
                face_a_dessiner j getp3d
                face_a_dessiner j 1 add n mod getp3d
                hauteurlignedeniveau segment_inter_planz {
                1 dict begin
                   /table exch def
                   /lignedeniveau [
                      lignedeniveau aload pop
                      table 0 getp3d
                      table length 4 ge {
                         table 1 getp3d
                      } if
                   ] store
                end
                } if
             } for
             lignedeniveau length 4 ge
                {lignedeniveau ligne3d}
             if
          grestore
       } if

       %% on marque les aretes
       0 1 n 1 sub {
          /j exch def
          /k1 F i j get_ij def              %% indice sommet1
          /k2 F i j 1 add n mod get_ij def  %% indice sommet2
          A k1 k2 true put_ij
          A k2 k1 true put_ij
       } for
```

```
   } if
end
} def

/drawsolid* {
1 dict begin
   /startest {true} def
   drawsolid
end
} def

/peintrealgorithme false def

/drawsolid** {
2 dict begin
   /aretescachees false def
   /peintrealgorithme true def
   drawsolid*
end
} def

%% syntaxe : solid drawsolid
/drawsolid {
7 dict begin
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans drawsolid) ==
      quit
   } if
   solid nullsolid not {
      solid solidgetfaces
      /F exch def
      solid solidgetpointstable
      /S exch def
      /n S length 3 idiv def
      %% tableau des aretes
      /A [
         n {
            [n {false} repeat]
         } repeat
      ] def

      peintrealgorithme {
         %% tri des indices des faces par distance decroissante
         [
            0 1 F length 1 sub {
               /i exch def
               solid i solidcentreface
               GetCamPos
               distance3d
            } for
         ] doublequicksort pop reverse
      } {
         [
            0 1 F length 1 sub {
            } for
         ]
      } ifelse
      /ordre exch def

      0 1 F length 1 sub {
         /k exch def
         /i ordre k get def
         gsave
         A solid i dessinefacevisible
         grestore
      } for
      aretescachees {
         0 1 F length 1 sub {
            /k exch def
            /i ordre k get def
            A solid i dessinefacecachee
         } for
      } if
%%       %% si on veut repasser les traits des faces visibles
%%       0 1 F length 1 sub {
```

```
%%          /k exch def
%%          /i ordre k get def
%%          gsave
%%          1 dict begin
%%             /startest false def
%%             A solid i dessinefacevisible
%%          end
%%          grestore
%%       } for
   } if
end
} def

%%%%% ### segment_inter_planz ###
%% syntaxe : A B k segment_inter_planz --> array true ou false
/segment_inter_planz {
4 dict begin
   /k exch def
   /B defpoint3d
   /A defpoint3d
   A /zA exch def pop pop
   B /zB exch def pop pop
   zA k sub zB k sub mul dup 0 gt {
      %% pas d intersection
      pop
      false
   } {
      0 eq {
         %% intersection en A ou en B
         [
            zA k eq {A} if
            zB k eq {B} if
         ] true
      } {
         %% intersection entre A et B
         [
            A B vecteur3d
            k zA sub zB zA sub div mulv3d
            A addv3d
         ] true
      } ifelse
   } ifelse
end
} def

%%%%% ### fin insertion ###

```

8.18 - Le cube tronqué

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%     operations sur des solides particuliers        %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### tronquecube ###
%% syntaxe : solid n tronque_cube --> solid (tronque)
/tronque_cube {
6 dict begin
   /d exch def
   /solid exch def
   solid issolid not {
      (Error : mauvais type d argument dans tronque_cube) ==
      quit
   } if
   solid solidgetpointstable
   /S exch def
   /co [
      3 4 1 % 1 3 4 % voisins du sommet 0
      0 5 2 % 0 2 5 %                de 1
      1 6 3 % 1 3 6 %                de 2
      2 7 0 % 0 2 7 %                de 3
      7 0 5 % 0 5 7 %                de 4
      4 1 6 % 1 4 6 %                de 5
      5 2 7 % 2 5 7 %                de 6
      6 3 4 % 3 4 6 %                de 7
   ] def

   /dd {d 1 sub} bind def
   /i 0 def
   /les_sommets [  % les coordonnees des sommets du cube tronque
      0 3 21 {
          /j exch def
          %% sommet d indice i = A1
          solid i solidgetsommet /A1 defpoint3d

          %% k = indice du sommet voisin no 1
          co j get /k exch def
          %% sommet d indice k = A2
          solid k solidgetsommet /A2 defpoint3d
          %% barycentre {(A1, d) (A2, 1)}
          A1 d A2 1 barycentre3d

          %% k = indice du sommet voisin no 2
          co j 1 add get /k exch def
          %% sommet d indice k = A2
          solid k solidgetsommet /A2 defpoint3d
          %% barycentre {(A1, d) (A2, 1)}
          A1 d A2 1 barycentre3d

          %% k = indice du sommet voisin no 2
          co j 2 add get /k exch def
          %% sommet d indice k = A2
          solid k solidgetsommet /A2 defpoint3d
          %% barycentre {(A1, d) (A2, 1)}
          A1 d A2 1 barycentre3d

          /i i 1 add store
      } for
   ] def

   /les_faces [
      [11 10 22 23 12 13 1 0]
      [2 1 13 14 15 16 4 3]
      [8 7 19 20 21 22 10 9]
      [14 12 23 21 20 18 17 15]
      [3 5 6 8 9 11 0 2]
      [5 4 16 17 18 19 7 6]
      [0 1 2]
      [3 4 5]
      [6 7 8]
      [9 10 11]
      [12 14 13]
      [15 17 16]
      [18 20 19]
      [21 23 22]
   ] def

   solid les_sommets solidputpointstable
   solid les_faces solidputfaces
   solid dup solidgetfcolors [8 {()} repeat] append solidputfcolors
   solid
end
} def

%%%%% ### fin insertion ###

```

8.19 - Les solides prédéfinis

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%             quelques solides precalcules           %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% ### newface ###
%% syntaxe : array newmonoface -> solid
%% ou array = tableau de points 2d
/newmonoface {
4 dict begin
   /table exch def
   /n table length 2 idiv def
   /S table {0} papply def

   /F [
       [0 1 n 1 sub {} for]
   ] def
   S F generesolid
end
} def

%% syntaxe : array newbiface -> solid
%% ou array = tableau de points 2d
/newbiface {
   newmonoface
   dup videsolid
} def

%%%%% ### newpolreg ###
%% syntaxe : r n newpolreg --> solid
/newpolreg {
5 dict begin
   /n exch def
   /r exch def
   /S [
       0 360 n div 360 360 n div sub {
           /theta exch def
           theta cos r mul
           theta sin r mul
           0
       } for
   ] def
   /F [
       [0 1 n 1 sub {} for]
   ] def

   S F generesolid
   dup videsolid
end
} def

%%%%% ### newgrille ###
%% syntaxe : xmin xmax ymin ymax [dx dy] newgrille -> solid
%% syntaxe : xmin xmax ymin ymax [nx ny] newgrille -> solid
%% syntaxe : xmin xmax ymin ymax {mode} newgrille -> solid
%% syntaxe : xmin xmax ymin ymax newgrille -> solid
/newgrille {
10 dict begin
   [[/nx /ny] [1 1] [1. 1.] [1. 1.] [1. 1.] [.5 .5]] gestionsolidmode
   %% ny nb d etages en y
   %% nx nb d etages en x
   [nx ny] {0} newsurfaceparametree
end
} def

/newsurface {
   true newsurfaceparametree
} def

/newsurfaceparametree {
10 dict begin
   dup isbool {
      pop /surfz true def
   } {
      /surfz false def
   } ifelse
   /f_surface exch def
   [[/nx /ny] [2 2] [4 4] [1. 1.] [1. 1.] [.25 .25]] gestionsolidmode
   %% ny nb d etages en y
   %% nx nb d etages en x
   /ymax exch def
   /ymin exch def
   /xmax exch def
   /xmin exch def

   nx isinteger not {
       %% alors nx est un dx
       /nx xmax xmin sub nx div cvi store
   } if
   ny isinteger not {
       %% alors ny est un dy
       /ny ymax ymin sub ny div cvi store
   } if
   /dy ymax ymin sub ny div def %% le pas sur y
   /dx xmax xmin sub nx div def %% le pas sur x

   /S [
       0 1 nx {
           /i exch def
           0 1 ny {
               /j exch def
               /u xmin i dx mul add def
               /v ymin j dy mul add def
               u v
       surfz {2 copy} if
       f_surface
               pstrickactionR3
           } for
       } for
   ] def

   /F [
       0 1 nx 1 sub {
          /i exch def
          0 1 ny 1 sub {
             /j exch def
             [
                j 1 add        i ny 1 add mul add
                j              i ny 1 add mul add
                j ny 1 add add i ny 1 add mul add
                j ny 2 add add i ny 1 add mul add
             ]
          } for
       } for
%%       0 1 0 {%nx 1 sub {
%%          /i exch def
%%          0 1 0 {%ny 2 sub {
%%             /j exch def
%%             [
%%             j 1 add        %% i ny mul add
%%             j              %% i ny mul add
%%             ny 1 add j add       %% i ny mul add
%%             ny 2 add j add     %% i ny mul add
%%             ]
%%          } for
%%       } for
   ] def
   S F generesolid
   dup videsolid
end
} def

%%%%% ### newgrillecirculaire ###
%% syntaxe : r option newgrillecirculaire -> solid
/newgrillecirculaire {
6 dict begin
   [[/K /N] [6 6] [6 8] [10 8] [16 12] [16 36]] gestionsolidmode

   %% N = nb de meridiens (diviseur de 360 = 2^4 * 3^2 * 5)
   %% K = nb d horizontales (diviseur de 160 = 2^5 * 5)

   /r exch def
   /F [
       %% 1er etage
       1 1 N {
           /i exch def
           [0 i i N mod 1 add]
       } for
       %% etages suivants
       0 1 K 2 sub {
           /j exch def
           1 1 N {
               /i exch def
               [i      j N mul add
               i N add j N mul add
               i N mod N add 1 add j N mul add
               i N mod 1 add j N mul add]
           } for
      } for
   ] def

   %% tableau des sommets
   /S [
       0 0 0
       1 1 K {
           /j exch def
           1 1 N {
             /i exch def
             /theta i 360 mul N div def
             theta cos r j mul K div mul
             theta sin r j mul K div mul
             2 copy exch atan 90 div
          } for
       } for
   ] def

   S F generesolid
end
} def

%%%%% ### newruban ###
%% syntaxe : array h u [n] newruban -> solid d axe (O, u), de maillage vertical n
%% syntaxe : array h u newruban -> solid d axe (O, u),
%% syntaxe : array h newruban -> solid d axe (O, k),
%% ou array tableau de points 2d
/newruban {
7 dict begin
   %% N = nb d etages
   [[/N] [1] [1] [1] [3] [4]] gestionsolidmode
   2 copy pop isarray {
      /u {0 0 1} def
   } {
      /u defpoint3d
   } ifelse
   u 0 eq {
      (Error : 3eme composante nulle dans le vecteur pour newruban) ==
      quit
   } if
   pop pop
   /h exch def
   /table exch def
   %% n = indice du dernier point
   /n table length 2 idiv 1 sub def
   %% vecteur de translation
   u
   h u norme3d div
   mulv3d /v defpoint3d

   %% tableau des sommets
   /S [
      0 1 N {
         /j exch def
         0 1 n {
             /i exch def
             table i getp
             0
             v N j sub N div mulv addv3d
```

```
            } for
         } for
      ] def

      /F [
         %% faces etage
         1 1 N {
            /j exch def
            1 1 n {
                /i exch def
                [i                  j 1 sub n 1 add mul add
                 i 1 sub            j 1 sub n 1 add mul add
                 n 1 add i add 1 sub j 1 sub n 1 add mul add
                 n 1 add i add      j 1 sub n 1 add mul add]
            } for
        } for
      ] def

      S F generesolid
      dup videsolid
   end
   } def

   %%%%% ### newicosaedre ###
   /newicosaedre {
   3 dict begin
      /a exch def
      /S [
         0.8944271  0              0.4472137
         0.2763932  0.8506507      0.4472137
         -0.7236067 0.5257311      0.4472137
         -0.7236067 -0.5257311     0.4472137
         0.2763932  -0.8506507     0.4472137
         0          0              1
         0          0              -1
         -0.8944271 0              -0.4472137
         -0.2763932 -0.8506507     -0.4472137
         0.7236067  -0.5257311     -0.4472137
         0.7236067  0.5257311      -0.4472137
         -0.2763932 0.8506507      -0.4472137
      ] {a mulv3d} papply3d def

      /F [
         [0 1 5]   %% 1  2 6  ]
         [1 2 5]   %% 2  3 6  ]
         [2 3 5]   %% 3  4 6  ]
         [3 4 5]   %% 4  5 6  ]
         [4 0 5]   %% 5  1 6  ]
         [9 0 4]   %% 10 1 5  ]
         [0 9 10]  %% 1  10 11]
         [10 1 0]  %% 11 2 1  ]
         [1 10 11] %% 2  11 12]
         [11 2 1]  %% 12 3 2  ]
         [2 11 7]  %% 3  12 8 ]
         [2 7 3]   %% 3  8 4  ]
         [3 7 8]   %% 4  8 9  ]
         [3 8 4]   %% 4  9 5  ]
         [4 8 9]   %% 5  9 10 ]
         [6 7 11]  %% 7  8 12 ]
         [6 8 7]   %% 7  9 8  ]
         [6 9 8]   %% 7  10 9 ]
         [6 10 9]  %% 7  11 10]
         [6 11 10] %% 7  12 11]
      ] def

      S F generesolid
   end
   } def

   %%%%% ### newdodecaedre ###
   /newdodecaedre {
   3 dict begin
      /a exch def
      /S [
         0          0.607062   0.7946545
         -0.5773503 0.1875925  0.7946545
      -0.3568221 -0.4911235 0.7946545
      0.3568221  -0.4911235 0.7946545
      0.5773503  0.1875925  0.7946545
      0          0.982247   0.1875925
      -0.9341724 0.303531   0.1875925
      -0.5773503 -0.7946645 0.1875925
      0.5773503  -0.7946645 0.1875925
      0.9341724  0.303531   0.1875925
      0          -0.982247  -0.1875925
      0.9341724  -0.303531  -0.1875925
      0.5773503  0.7946545  -0.1875925
      -0.5773503 0.7946545  -0.1875925
      -0.9341724 -0.303531  -0.1875925
      -0.5773503 -0.1875925 -0.7946545
      -0.3568221 0.4911235  -0.7946545
      0.3568221  0.4911235  -0.7946545
      0.5773503  -0.1875925 -0.7946545
      0          -0.607062  -0.7946545
   ] {a mulv3d} papply3d def

   /F [
      [0 1 2 3 4]
      [4 3 8 11 9]
      [4 9 12 5 0]
      [0 5 13 6 1]
      [1 6 14 7 2]
      [2 7 10 8 3]
      [10 19 18 11 8]
      [11 18 17 12 9]
      [12 17 16 13 5]
      [13 16 15 14 6]
      [14 15 19 10 7]
      [15 16 17 18 19]
   ] def
   S F generesolid
end
} def

%%%%% ### newoctaedre ###
/newoctaedre {
3 dict begin
   /a exch def
   %%Sommets
   /S [
      0  0  1
      1  0  0
      0  1  0
      -1 0  0
      0  -1 0
      0  0  -1
   ] {a mulv3d} papply3d def

   /F [
      [0 4 1]
      [1 2 0]
      [0 2 3]
      [3 4 0]
      [1 5 2]
      [2 5 3]
      [3 5 4]
      [4 5 1]
   ] def

   S F generesolid
end
} def

%%%%% ### newtetraedre ###
/newtetraedre {
3 dict begin
   /r exch def
   %%Tetraedre
   /S [
      0          0          1
      -0.4714045 -0.8164965 -1 3 div
      0.942809   0          -1 3 div
      -0.4714045 0.8164965  -1 3 div
   ] {r mulv3d} papply3d def

   /F [
      [0 1 2]
      [0 2 3]
      [0 3 1]
      [1 3 2]
   ] def

   S F generesolid
end
} def

%%%%% ### newcube ###
/newcube {
3 dict begin
   [[/n] [1] [1] [1] [3] [4]] gestionsolidmode
   /a exch 2 div def

   n 1 le {
      /F [
     [0 1 2 3]
     [0 4 5 1]
     [1 5 6 2]
     [2 6 7 3]
     [0 3 7 4]
     [4 7 6 5]
      ] def

      %% tableau des sommets
      /S [
      1  1  1 %% 0
     -1  1  1 %% 1
     -1 -1  1 %% 2
      1 -1  1 %% 3
      1  1 -1 %% 4
     -1  1 -1 %% 5
     -1 -1 -1 %% 6
      1 -1 -1 %% 7
      ] {a mulv3d} papply3d def
      S F generesolid
   } {
      /dl 2 n div def
      /N n dup mul n add 4 mul def
      /n1 n 1 sub dup mul def %% nb sommets centre d une face

      %% tableau des sommets
      /S1 [
     0 1 n 1 sub {
        /j exch def
        0 1 n {
           /i exch def
           -1 i dl mul add
           -1 j dl mul add
        1
        } for
     } for
      ] def

      /S2 S1 {-90 0 0 rotateOpoint3d} papply3d def
      /S3 S2 {-90 0 0 rotateOpoint3d} papply3d def
      /S4 S3 {-90 0 0 rotateOpoint3d} papply3d def

      /S5 [
     1 1 n 1 sub {
        /j exch def
        1 1 n 1 sub {
           /i exch def
        1
           -1 i dl mul add
           -1 j dl mul add
        } for
     } for
      ] def

```

```
 /S6 [
1 1 n 1 sub {
   /j exch def
   1 1 n 1 sub {
      /i exch def
      -1
      -1 i dl mul add
      -1 j dl mul add
   } for
} for
 ] def

 %% tableau des faces
 /F1 [
0 1 n 1 sub {
   /j exch def
   0 1 n 1 sub {
      /i exch def
      [
     i n 1 add j mul add
     dup 1 add
     dup n 1 add add
     dup 1 sub
      ]
   } for
} for
 ] def

 %% syntaxe : i sommettourgauche --> l indice du i-eme sommet du tour
 %% de la face gauche (en commencant par l indice 0). ATTENTION :
 %% utilise la variable globale n = nb d etages
 /sommettourgauche {
 1 dict begin
/i exch def
i 4 n mul ge {
   i
   (Error: indice trop grand dans sommettourgauche) ==
   exit
} if
n n 1 add i mul add
 end
 } def

 %% syntaxe : i sommetcentregauche --> l indice du i-eme sommet du centre
 %% de la face gauche (en commencant par l indice 0). ATTENTION :
 %% utilise les variables globales n = nb d etages, et N = nb sommets
 %% des 4 1eres faces
 /sommetcentregauche {
 1 dict begin
/i exch def
i n 1 sub dup mul ge {
   i
   (Error: indice trop grand dans sommetcentregauche) ==
   exit
} if
N i add
 end
 } def

 /F5 [
%%%%% la face gauche %%%%%
%% le coin superieur gauche
[
   1 sommettourgauche
   0 sommettourgauche
   n 4 mul 1 sub sommettourgauche
   n1 n 1 sub sub sommetcentregauche
]

%% la bande superieure (i from 1 to n-2)
1 1 n 2 sub {
   /i exch def
   [
      i 1 add sommettourgauche
      i sommettourgauche
      n1 n sub i add sommetcentregauche
```

```
            n1 n sub i 1 add add sommetcentregauche
         ]
      } for

      %% le coin superieur droit
      [
         n sommettourgauche
         n 1 sub sommettourgauche
         n1 1 sub sommetcentregauche
         n 1 add sommettourgauche
      ]

      %% la descente gauche
      %% j from 1 to n-2
      1 1 n 2 sub {
         /j exch def
         [
            n1 n 1 sub j mul sub sommetcentregauche
            n 4 mul j sub sommettourgauche
            n 4 mul j 1 add sub sommettourgauche
            n1 n 1 sub j 1 add mul sub sommetcentregauche
         ]
      } for

      %% les bandes centrales (j from 1 to n-2 et i from 1 to n-2)
      1 1 n 2 sub {
         /j exch def
         1 1 n 2 sub {
            /i exch def
            [
           n1 i n 1 sub j 1 sub mul add sub sommetcentregauche
           n1 i 1 add n 1 sub j 1 sub mul add sub sommetcentregauche
           n1 i 1 add n 1 sub j mul add sub sommetcentregauche
           n1 i n 1 sub j mul add sub sommetcentregauche
            ]
         } for
      } for

      %% la descente droite
      1 1 n 2 sub {
         /j exch def
         [
            n j add sommettourgauche
            n1 1 sub j 1 sub n 1 sub mul sub sommetcentregauche
            n1 1 sub j n 1 sub mul sub sommetcentregauche
            n j 1 add add sommettourgauche
         ]
      } for

      %% le coin inferieur gauche
      [
         0 sommetcentregauche
         n 3 mul 1 add sommettourgauche
         n 3 mul sommettourgauche
         n 3 mul 1 sub sommettourgauche
      ]

      %% la bande inferieure (i from 1 to n-2)
      1 1 n 2 sub {
         /i exch def
         [
            i sommetcentregauche
            i 1 sub sommetcentregauche
            n 3 mul i sub sommettourgauche
            n 3 mul i sub 1 sub sommettourgauche
         ]
      } for

      %% le coin inferieur droit
      [
         n 2 mul 1 sub sommettourgauche
         n 2 sub sommetcentregauche
         n 2 mul 1 add sommettourgauche
         n 2 mul sommettourgauche
      ]
       ] def
```

```

 %% syntaxe : i sommettourdroit --> l indice du i-eme sommet du tour
 %% de la face droit (en commencant par l indice 0). ATTENTION :
 %% utilise la variable globale n = nb d etages
 /sommettourdroit {
 1 dict begin
/i exch def
i 4 n mul ge {
   i
   (Error: indice trop grand dans sommettourdroit) ==
   exit
} if
n 1 add i mul
 end
 } def

 %% syntaxe : i sommetcentredroit --> l indice du i-eme sommet du centre
 %% de la face droit (en commencant par l indice 0). ATTENTION :
 %% utilise les variables globales n = nb d etages, et N = nb sommets
 %% des 4 1eres faces
 /sommetcentredroit {
 1 dict begin
/i exch def
i n 1 sub dup mul ge {
   i
   (Error: indice trop grand dans sommetcentredroit) ==
   exit
} if
N n1 add i add
 end
 } def

 /F6 [
%% coin superieur droit
[
   0 sommettourdroit
   1 sommettourdroit
   n1 n 1 sub sub sommetcentredroit
   4 n mul 1 sub sommettourdroit
]
%% coin superieur gauche
[
   n 1 sub sommettourdroit
   n sommettourdroit
   n 1 add sommettourdroit
   n1 1 sub sommetcentredroit
]
%% coin inferieur gauche
[
   n 2 sub sommetcentredroit
   2 n mul 1 sub sommettourdroit
   2 n mul sommettourdroit
   2 n mul 1 add sommettourdroit
]
%% coin inferieur droit
[
   3 n mul 1 add sommettourdroit
   0 sommetcentredroit
   3 n mul 1 sub sommettourdroit
   3 n mul sommettourdroit
]
%% bande superieure
1 1 n 2 sub {
   /i exch def
   [
      i sommettourdroit
      i 1 add sommettourdroit
      n 1 sub n 2 sub mul i add sommetcentredroit
      n 1 sub n 2 sub mul i 1 sub add sommetcentredroit
   ]
} for
%% bande inferieure
1 1 n 2 sub {
   /i exch def
   [
      i 1 sub sommetcentredroit
```

```
            i sommetcentredroit
            3 n mul 1 sub i sub sommettourdroit
            3 n mul i sub sommettourdroit
         ]
      } for
      %% descente gauche
      1 1 n 2 sub {
         /i exch def
         [
            n1 1 sub i 1 sub n 1 sub mul sub sommetcentredroit
            n i add sommettourdroit
            n i 1 add add sommettourdroit
            n1 1 sub i n 1 sub mul sub sommetcentredroit
         ]
      } for
      %% descente droite
      1 1 n 2 sub {
         /i exch def
         [
            4 n mul i sub sommettourdroit
            n 1 sub n 1 sub i sub mul sommetcentredroit
            n 1 sub n 2 sub i sub mul sommetcentredroit
            4 n mul i sub 1 sub sommettourdroit
         ]
      } for
      %% bandes interieures
      1 1 n 2 sub {
         /j exch def
         1 1 n 2 sub {
            /i exch def
            [
           n 1 sub j mul i 1 sub add sommetcentredroit
           n 1 sub j mul i add sommetcentredroit
           n 1 sub j 1 sub mul i add sommetcentredroit
           n 1 sub j 1 sub mul i 1 sub add sommetcentredroit
            ]
         } for
      } for

       ] def

       /F2 F1 {{n dup mul n add add} apply} apply def
       /F3 F2 {{n dup mul n add add} apply} apply def
       /F4 F3 {{n dup mul n add add} apply} apply def


       S1 S2 append S3 append S4 append S5 append S6 append {a mulv3d} papply3d
       F1 F2 append F3 append F4 append {{N mod} apply} apply F5 append F6 append
       generesolid
   } ifelse
end
} def

%%%%% ### newparallelepiped ###
% 14 octobre 2006
/newparallelepiped {
2 dict begin
   /c exch 2 div def
   /b exch 2 div def
   /a exch 2 div def
   /F [
      [0 1 2 3]
      [0 4 5 1]
      [1 5 6 2]
      [2 6 7 3]
      [0 3 7 4]
      [4 7 6 5]
    ] def

    %% tableau des sommets
    /S [
       a     b     c %% 0
       a neg b     c %% 1
       a neg b neg c %% 2
       a     b neg c %% 3
       a     b     c neg %% 4
```

```
        a neg b     c neg %% 5
        a neg b neg c neg %% 6
        a     b neg c neg %% 7
     ] def
     S F generesolid
} def

%%%%% ### newcylindre ###
%% syntaxe : z0 r0 z1 newcylindre -> solide
/newcylindre {
   dup xcheck {
      2 index exch
   } {
      dup isarray {
         2 index exch
      } {
         1 index
      } ifelse
   } ifelse
   newtronccone
} def

%% syntaxe : z0 r0 z1 newcylindrecreux -> solide
/newcylindrecreux {
   newcylindre
   dup creusesolid
} def

%%%%% ### newtronccone ###
%% syntaxe : z0 r0 z1 r1 newtronccone -> solid
/newtronccone {
11 dict begin
   [[/n /N] [1 6] [1 8] [1 10] [3 12] [5 18]] gestionsolidmode

   /r1 exch def
   /z1 exch def
   /r0 exch def
   /z0 exch def
   /dz z1 z0 sub n div def
   /dr r1 r0 sub n div def

   /FE [
      [0 1 N 1 sub {} for]
      [n 1 add N mul 1 sub -1 n N mul {} for]

      0 1 n 1 sub {
      /k exch def
         k N mul 1 add 1 k 1 add N mul 1 sub {
            /i exch def
            [i i 1 sub N i add 1 sub N i add]
         } for
         [k N mul k 1 add N mul 1 sub k 2 add N mul 1 sub k 1 add N mul]
      } for

   ] def

   %% tableau des sommets
   /S [
      n -1 0 {
         /k exch def
         0 1 N 1 sub {
            /i exch def
            360 N idiv i mul cos r0 dr k mul add mul
            360 N idiv i mul sin r0 dr k mul add mul
            z0 dz k mul add
         } for
      } for
   ] def
   S FE generesolid
end
} def

%% syntaxe : z0 r0 z1 r1 newtroncconecreux -> solid
/newtroncconecreux {
   newtronccone
   dup creusesolid
```

```
} def

%%%%% ### newcone ###
%% syntaxe : z0 r0 z1 newcone -> solid
/newcone {
11 dict begin
   [ [/n /N] [1 6] [1 8] [1 10] [3 12] [5 18] ] gestionsolidmode

   /z1 exch def
   /r0 exch def
   /z0 exch def
   /dz z1 z0 sub n div def
   /dr r0 n div def

   /F [
      %% la base
      [N 1 sub -1 0 {} for]
      %% le dernier etage
      n 1 sub N mul 1 add 1 n N mul 1 sub {
           /i exch def
           [i 1 sub i n N mul]
      } for
      [n N mul 1 sub n 1 sub N mul n N mul]
      %% les autres etages
      0 1 n 2 sub {
         /j exch def
         0 N j mul add 1 N N j mul add 2 sub {
            /i exch def
            [i i 1 add dup N add dup 1 sub]
         } for
         [N N j mul add 1 sub N j mul dup N add dup N add 1 sub]
      } for
   ] def

   %% tableau des sommets
   /S [
      %% etage no j (in [1; n])
      0 1 n 1 sub {
         /j exch def
         0 1 N 1 sub {
             /i exch def
             360 N idiv i mul cos r0 dr j mul sub mul
             360 N idiv i mul sin r0 dr j mul sub mul
             z0 dz j mul add
         } for
      } for
      0 0 z1
   ] def
   S F generesolid
end
} def

%% syntaxe : z0 r0 z1 newconecreux -> solid
/newconecreux {
   newcone
   dup 0 solidrmface
   dup videsolid
} def

%%%%% ### newtore ###
%% syntaxe : r R newtore -> solid
/newtore {
10 dict begin
   [[/n1 /n2] [4 5] [6 10] [8 12] [9 18] [18 36]] gestionsolidmode
   /n2 n2 3 max store
   /n1 n1 2 max store
   /R exch def
   /r exch def
   /S [
         0 1 n1 1 sub {
            /i exch def
            360 n1 div i mul cos r mul R add
            360 n1 div i mul sin r mul
         } for
      ]
   def
   S [n2] newanneau
end
} def

%%%%% ### newprisme ###
/newprismedroit {
   [[/N] [1] [1] [1] [3] [6]] gestionsolidmode
   0 0 1 [N] newprisme
} def

%% syntaxe : array N z0 z1 u newprisme -> solid d axe (O, u),
%% ou array tableau de points 2d
/newprisme {
7 dict begin
   [[/N] [1] [1] [1] [3] [6]] gestionsolidmode
   dup 0 eq {
      (Error : 3eme composante nulle dans le vecteur pour newprisme) ==
      quit
   } if
   /u defpoint3d
   /z1 exch def
   /z0 exch def
   %% N = nb d etages
   /table exch def
   %% n = indice du dernier point
   /n table length 2 idiv 1 sub def
   %% vecteur de translation
   u
   z1 z0 sub u norme3d div
   mulv3d /v defpoint3d

   %% tableau des sommets
   /S [
      0 1 N {
         /j exch def
         0 1 n {
             /i exch def
             table i getp
             z0
             v N j sub N div mulv addv3d
         } for
      } for
   ] def

   /F [
      %% face superieure
      [0 1 n {} for]
      %% base
      [N 1 add n 1 add mul 1 sub -1 N n 1 add mul {} for]
      %% faces etage
      1 1 N {
         /j exch def
         1 1 n {
             /i exch def
             [i                    j 1 sub n 1 add mul add
              i 1 sub              j 1 sub n 1 add mul add
              n 1 add i add 1 sub j 1 sub n 1 add mul add
              n 1 add i add       j 1 sub n 1 add mul add]
         } for
         [0           j 1 sub n 1 add mul add
         n            j 1 sub n 1 add mul add
         2 n mul 1 add j 1 sub n 1 add mul add
         n 1 add      j 1 sub n 1 add mul add]
     } for
   ] def

   S F generesolid
end
} def

%%%%% ### newsphere ###
%% syntaxe : r option newsphere -> solid
/newsphere {
2 dict begin
   [[/K /N] [6 6] [8 8] [10 12] [16 12] [16 36]] gestionsolidmode
   -90 90 [K N] newcalottesphere
```

```
end
} def

%% syntaxe : r phi theta option newcalottesphere -> solid
/newcalottesphere {
6 dict begin
   [[/K /N] [6 6] [8 8] [10 12] [16 12] [16 36]] gestionsolidmode

   %% test de beta (ex-theta)
   dup 90 eq {
      /beta exch def
      /idebut 1 def
   } {
      /beta exch 80 min -80 max def
      /idebut 0 def
   } ifelse
   %% test de alpha (ex-phi)
   dup -90 eq {
      /alpha exch def
   } {
      /alpha exch beta min -80 max def
   } ifelse
   /r exch def
   beta 90 eq {
       alpha -90 eq {
           /ifin K def
          /db alpha beta sub K 1 add div def
       } {
           /ifin K def
          /db alpha beta sub K div def
       } ifelse
   } {
       alpha -90 eq {
          /ifin K 1 sub def
          /db alpha beta sub K div def
       } {
          /ifin K 1 sub def
          /db alpha beta sub K 1 sub div def
       } ifelse
   } ifelse

   %% nombre de sommets -2
   /nb N K mul def

   %% tableau des sommets
   /S [
       idebut 1 ifin {
           /j exch def
           /phi beta j db mul add def
           phi cos r mul /r_tmp exch def
           0 1 N 1 sub {
                /i exch def
                360 N idiv i mul cos r_tmp mul
                360 N idiv i mul sin r_tmp mul
                phi sin r mul
            } for
       } for
      0 0 r neg
      0 0 r
   ] def

   /F [
     %% calotte inferieure
     alpha -90 eq {
         1 1 N 1 sub {
         /i exch def
            [
                nb
                nb i sub
                nb i 1 add sub
            ]
         } for
         [nb nb N sub nb 1 sub]
     } {
        [nb 1 sub -1 nb N sub {} for ]
     } ifelse
```

```

    %% calotte superieure
    beta 90 eq {
        0 1 N 1 sub {
           /i exch def
            [i i 1 add N mod N K mul 1 add]
        } for
     } {
        [0 1 N 1 sub {} for]
     } ifelse

    1 1 K 1 sub {
         /j exch def
      [
          j N mul
          j N mul 1 add
          j 1 sub N mul 1 add
          j 1 sub N mul
      ]
      N 2 sub {dup {1 add} apply} repeat
      [
          j 1 add N mul 1 sub
          j N mul
          j 1 sub N mul
          j N mul 1 sub
      ]
   } for
  ] def

  S F generesolid
end
} def

%% syntaxe : r phi theta option newcalottespherecreuse -> solid
/newcalottespherecreuse {
6 dict begin
  [[/K /N] [6 6] [8 8] [10 12] [16 12] [16 36]] gestionsolidmode

  %% test de beta (ex-theta)
  dup 90 eq {
     /beta exch def
     /idebut 1 def
  } {
     /beta exch 80 min -80 max def
     /idebut 0 def
  } ifelse
  %% test de alpha (ex-phi)
  dup -90 eq {
     /alpha exch def
  } {
     /alpha exch beta min -80 max def
  } ifelse
  /r exch def
  beta 90 eq {
      alpha -90 eq {
          /ifin K def
         /db alpha beta sub K 1 add div def
      } {
          /ifin K def
         /db alpha beta sub K div def
      } ifelse
  } {
      alpha -90 eq {
          /ifin K 1 sub def
         /db alpha beta sub K div def
      } {
          /ifin K 1 sub def
         /db alpha beta sub K 1 sub div def
      } ifelse
  } ifelse

  %% nombre de sommets -2
  /nb N K mul def

  %% tableau des sommets
  /S [
       idebut 1 ifin {
           /j exch def
           /phi beta j db mul add def
           phi cos r mul /r_tmp exch def
           0 1 N 1 sub {
                /i exch def
                360 N idiv i mul cos r_tmp mul
                360 N idiv i mul sin r_tmp mul
                phi sin r mul
            } for
       } for
      0 0 r neg
      0 0 r
   ] def

   /F [
     %% calotte inferieure
     alpha -90 eq {
         1 1 N 1 sub {
         /i exch def
            [
                nb
                nb i sub
                nb i 1 add sub
            ]
         } for
         [nb nb N sub nb 1 sub]
     } {
%        [nb 1 sub -1 nb N sub {} for ]
     } ifelse

     %% calotte superieure
     beta 90 eq {
         0 1 N 1 sub {
            /i exch def
             [i i 1 add N mod N K mul 1 add]
         } for
      } {
%         [0 1 N 1 sub {} for]
      } ifelse

     1 1 K 1 sub {
          /j exch def
       [
           j N mul
           j N mul 1 add
           j 1 sub N mul 1 add
           j 1 sub N mul
       ]
       N 2 sub {dup {1 add} apply} repeat
       [
           j 1 add N mul 1 sub
           j N mul
           j 1 sub N mul
           j N mul 1 sub
       ]
    } for
   ] def

   S F generesolid
   dup videsolid
end
} def

%%%%% ### newanneau ###
%% syntaxe : array n newanneau --> solid
%% syntaxe : array {mode} newanneau --> solid
%% ou array est un tableau de points de Rˆ2 et n un nombre entier positif
/newanneau {
10 dict begin
   dup isnum {
      /n exch def
      [n]
   } if
   [[/n2] [6] [12] [24] [32] [36]] gestionsolidmode
   /n2 n2 3 max store
   %% on plonge la section dans R^3 par projection sur yOz
   /S1 exch {0 3 1 roll} papply def
   %% nombre de sommets
   /n1 S1 length 3 idiv def

   /S S1
      n2 {
         duparray
         {0 0 360 n2 div rotateOpoint3d} papply3d
      } repeat
      n2 {append} repeat
   def

   /F [
      0 1 n2 1 sub {
         /j exch def
         n1 j mul 1 j 1 add n1 mul 2 sub {
            /i exch def
            [i 1 add i dup n1 add i n1 1 add add]
         } for
         [n1 j mul j 1 add n1 mul 1 sub j 2 add n1 mul 1 sub j 1 add n1 mul]
      } for
   ] def

   S F generesolid
end
} def

%%%%% ### newvecteur ###
%% syntaxe : x y z newvecteur
/newvecteur {
4 dict begin
   /A defpoint3d
   %%Sommets
   /S [0 0 0 A] def
   /F [
      [0 1]
   ] def
   S F generesolid
%%   /axe exch def
   [ A ]
   normalvect_to_orthobase
   /imK defpoint3d
   /imJ defpoint3d
   /imI defpoint3d

   A norme3d /z exch .3 sub def
   0 .1 .3 [1 8] newcone
   dup (noir) outputcolors
   {0 0 z translatepoint3d} solidtransform
   {imI imJ imK transformpoint3d} solidtransform
   solidfuz
end
} def

%%%%% ### newobjfile ###
/newobjfile {
3 dict begin
   /objfilename exch def
   /v {} def
   /ok true def
   /f {
       ok {
        %% 1ere fois
           ] %% ferme les sommets
        [ [ %% ouvre les faces
        /ok false store
       } {
        %% les autres fois
           ] %% ferme la face
        [ %% ouvre la nouvelle
       } ifelse
   } def
   [ 0 0 0
   objfilename run
   ]]
   /F exch def
   /S exch def

   S F generesolid
%   dup videsolid
end
} def

%%%%% ### fin insertion ###

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                                                    %%%%
%%%%      fin insertion librairie jps                   %%%%
%%%%                                                    %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

9. Gestion des chaînes de caractère

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%          gestion de chaine de caracteres           %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/Times-Roman findfont
dup length dict begin
   {
   1 index /FID ne
      {def}
      {pop pop}
   ifelse
   } forall
   /Encoding ISOLatin1Encoding def
   currentdict
end
/Times-Roman-ISOLatin1 exch definefont pop

/setTimesRoman {
   /Times-Roman-ISOLatin1 findfont
   fontsize scalefont
   setfont
} def

/setTimes {
   setTimesRoman
} def

%% syntaxe : string x y cctext
/cctext {
5 dict begin
   /y exch def
   /x exch def
   /str exch def
   str stringwidth
   /wy exch def
   /wx exch def
   gsave
      x y smoveto
      wx -2 div wy -2 div rmoveto
      str show
   grestore
end
} def

%% syntaxe : str x y show_dim --> str x y llx lly wx wy
%% attention, doit laisser la pile intacte
/show_dim {
   3 copy pop pop
   newpath
      0 0 moveto
      true charpath flattenpath pathbbox
   closepath
   newpath
} def

```

10. Interfaçage avec PSTricks

10.1 - Interface pour la macro `psSolid`

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%             procedures pour PSTricks              %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%             procedures pour \psSolid              %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/all (all) def

/draw {drawsolid} def
/draw* {drawsolid*} def
/draw** {drawsolid**} def
/none {pop} def

/gere_pstricks_color_inout {
   gsave
      dup  [fillincolor] (setrgbcolor) astr2str
         [fillcolor] (setrgbcolor) astr2str inoutputcolors
   grestore
} def

/gere_pstricks_color_out {
   gsave
      dup  [fillcolor] (setrgbcolor) astr2str outputcolors
   grestore
} def

/gere_pstricks_opt {
%   /CourbeR2 {CourbeR2+} def
   linecolor
   solidlinewidth setlinewidth
   RotX 0 ne RotY 0 ne or RotZ 0 ne or {
      {RotX RotY RotZ rotateOpoint3d} solidtransform
   } if
   CX 0 ne CY 0 ne or CZ 0 ne or {
      {CX CY CZ translatepoint3d} solidtransform
   } if
   /rmfaces rmfaces bubblesort reverse store
   0 1 rmfaces length 1 sub {
      /i exch def
      dup rmfaces i get solidrmface
   } for
   solidhollow {
      dup videsolid
   } if
   activationgestioncouleurs {
      dup solidwithinfaces {
         gere_pstricks_color_inout
      } {
         gere_pstricks_color_out
      } ifelse
   } if

   0 1 fcol length 2 idiv 1 sub {
      /i exch def
      dup fcol 2 i mul get fcol 2 i mul 1 add get solidputfcolor
   } for
   tx@Dict /pst-transformoption known {
      dup {pst-transformoption} solidtransform
   } if
   solidinouthue length 0 gt {
      dup solidinouthue solidputinouthuecolors
   } {
      solidhue length 0 gt {
         dup solidhue solidputhuecolors
      } if
      solidinhue length 0 gt {
         dup solidinhue solidputinhuecolors
      } if
   } ifelse
   dup action
   noir
   solidnumf length 0 ne {
      solidnumf 0 get isstring {
         dup projectionsifacevisible solidnumfaces
      } {
         dup solidnumf projectionsifacevisible solidnumfaces
      } ifelse
   } if
   solidshow length 0 ne {
      solidshow 0 get isstring {
         dup solidshowsommets
      } {
         dup solidshow solidshowsommets
      } ifelse
   } if
   solidnum length 0 ne {
      solidnum 0 get isstring {
         dup solidnumsommets
      } {
         dup solidnum solidnumsommets
      } ifelse
   } {
      %% pop
   } ifelse
   tx@Dict /solidname known {
      solidname exch bind def
      tx@Dict /solidname undef
   } {
      pop
   } ifelse
} def

/pst-octahedron {
   a newoctaedre
   gere_pstricks_opt
} def

/pst-dodecahedron {
   a newdodecaedre
   gere_pstricks_opt
} def

/pst-icosahedron {
   a newicosaedre
   gere_pstricks_opt
} def

/pst-cube {
   a
   ngrid length 1 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   newcube
%%    solidhollow {
%%       dup videsolid
%%    } if
   gere_pstricks_opt
} def

/pst-parallelepiped {
   a b c
   newparallelepiped
   gere_pstricks_opt
} def

/pst-tetrahedron {
   r newtetraedre
   gere_pstricks_opt
} def

/pst-tore {
   r0 r1
   ngrid length 2 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   newtore
   gere_pstricks_opt
} def

/pst-sphere {
   % rayon
   % mode
  %   r {Mode} newsphere
   r
   ngrid length 2 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   newsphere
   gere_pstricks_opt
} def

/pst-cylindre {
   % rayon
   % mode
   0 r h
   ngrid length 2 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   newcylindre
   solidhollow {
      dup creusesolid
   } if
   gere_pstricks_opt
} def

/pst-cylindrecreux {
   % rayon
   % mode
   0 r h
   ngrid length 2 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   newcylindre
   dup creusesolid
   gere_pstricks_opt
} def

/pst-cone {
   % rayon
   % mode
   0 r h
   ngrid length 2 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   solidhollow {
      newconecreux
   } {
      newcone
   } ifelse
   gere_pstricks_opt
} def

/pst-tronccone {
   % rayon
   % mode
   0 r0 h r1
   ngrid length 2 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   solidhollow {
      newtroncconecreux
   } {
      newtronccone
   } ifelse
   gere_pstricks_opt
} def

/pst-troncconecreux {
   % rayon
   % mode
   0 r0 h r1
   ngrid length 2 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   newtroncconecreux
   gere_pstricks_opt
} def

/pst-conecreux {
   % rayon
   % mode
   0 r h
   ngrid length 2 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   newconecreux
   gere_pstricks_opt
} def

/pst-anneau {
   [ section ]
   ngrid length 1 ge {
      [ngrid 0 get]
   } {
      [24]
   } ifelse
   newanneau
   gere_pstricks_opt
} def


/pst-prisme {
   % tableau des points de la base
   % h hauteur du prisme
   % axe : vecteur direction de l axe
   base decal rollparray
   0 h axe
   ngrid length 1 ge {
      [ngrid 0 get]
   } if
   newprisme
   solidhollow {
      dup creusesolid
   } if
   gere_pstricks_opt
} def

/pst-prismecreux {
   % tableau des points de la base
   % h hauteur du prisme
   % axe : vecteur direction de l axe
   base
   0 h axe
   ngrid length 1 ge {
      [ngrid 0 get]
   } if
   newprisme
   dup creusesolid
   gere_pstricks_opt
} def

/pst-grille {
   base aload pop
   ngrid length 2 ge {
      [ngrid 0 get ngrid 1 get]
   } {
      ngrid length 1 eq {
         [ngrid 0 get dup]
      } if
   } ifelse
   newgrille
   gere_pstricks_opt
} def

%% syntaxe : array N h u newruban -> solid d axe (O, u),
/pst-ruban {
   % tableau des points de la base
   % h hauteur du prisme
   % axe : vecteur direction de l axe
   base
   h axe
   ngrid length 1 ge {
      [ngrid 0 get]
   } if
   newruban
   gere_pstricks_opt
} def

%% syntaxe : r phi option newcalottesphere -> solid
/pst-calottesphere {
   % rayon
   % mode
   % r phi theta option newcalottesphere
   r
   phi theta
   ngrid length 2 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   solidhollow {
      newcalottespherecreuse
   } {
      newcalottesphere
   } ifelse
   gere_pstricks_opt
} def

%% syntaxe : r phi option newcalottesphere -> solid
/pst-calottespherecreuse {
   % rayon
   % mode
   % r phi theta option newcalottespherecreuse
   r
   phi theta
   ngrid length 2 eq {
      ngrid
   } {
      {Mode}
   } ifelse
   newcalottespherecreuse
   gere_pstricks_opt
} def

```

```
/pointtest{2 2 2} def

/pst-face {
   % tableau des points de la base
   % h hauteur du prisme
   % axe : vecteur direction de l axe
   base
   solidbiface {
      newbiface
   } {
      newmonoface
   } ifelse
   gere_pstricks_opt
} def

/pst-surface {
   base
   base aload pop
   ngrid length 2 ge {
      [ngrid 0 get ngrid 1 get]
   } {
      ngrid length 1 eq {
         [ngrid 0 get dup]
      } ifelse
   } ifelse
   {f} newsurface
   solidbiface {
      dup videsolid
   } if
   gere_pstricks_opt
} def

/pst-polygoneregulier {
   r ngrid 0 get
   newpolreg
   solidbiface {
   } {
      dup 1 solidrmface
   } ifelse
   gere_pstricks_opt
} def

/pst-fusion {
1 dict begin
   /activationgestioncouleurs false def
   /n base length def
   base aload pop n 1 sub {solidfuz} repeat
   gere_pstricks_opt
end
} def

/pst-new {
   sommets faces
   generesolid
%%    solidhollow {
%%       dup videsolid
%%    } if
   gere_pstricks_opt
} def

/pst-courbe {
   solidlinewidth setlinewidth
   range aload pop {function} CourbeR3
} def

/pst-surfaceparametree {
   base aload pop
   ngrid length 2 ge {
      [ngrid 0 get ngrid 1 get]
   } {
      ngrid length 1 eq {
         [ngrid 0 get dup]
      } if
   } ifelse
   { function } newsurfaceparametree
   dup videsolid
   gere_pstricks_opt
} def

/pst-vecteur {
gsave
   solidlinewidth setlinewidth
   1 setlinejoin
   1 setlinecap
   linecolor
   linestyle
   args newvecteur
   dup
      [linecolor currentrgbcolor] ( ) astr2str (setrgbcolor) append
      outputcolors
   gere_pstricks_opt
grestore
} def

/pst-ligne {
   newpath
      base 0 get
      base 1 get
      base 2 get
      3dto2d smoveto
      base ligne3d_
} def

/pst-objfile {
   solidfilename newobjfile
%   dup {1 1 div mulv3d} solidtransform
%%    solidhollow {
%%       dup videsolid
%%    } if
   gere_pstricks_opt
} def

```

10.2 - Interface pour la macro `psProjection`

Le fichier solides.pro

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%         procedures pour \psProjection              %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/gere_pstricks_proj_opt {
      /solidprojname where {
         /solidprojname get noface phi
         xorigine 0 eq
         yorigine 0 eq and
         zorigine 0 eq and
         xorigine isinteger not and
         yorigine isinteger not and
         yorigine isinteger not and {
         } {
            [xorigine yorigine zorigine] (                 ) astr2str
         } ifelse
         projectionsifacevisible solidprojpath
      } {
         xorigine yorigine zorigine [ normale ] projectionsifacevisible planprojpath
      } ifelse
} def

/proj-pst-chemin {
   solidlinewidth setlinewidth
   newpath
      path
      linecolor
      gere_pstricks_proj_opt
} def

/proj-pst-courbeR2 {
   solidlinewidth setlinewidth
   newpath
      linecolor
      range aload pop { function } CourbeR2_
      gere_pstricks_proj_opt
} def

/proj-pst-courbe {
   solidlinewidth setlinewidth
   newpath
      linecolor
      range aload pop {} { function } Courbeparam_
      gere_pstricks_proj_opt
} def

/proj-pst-texte {
2 dict begin
      setTimes
      solidlinewidth setlinewidth
      newpath
      linecolor
      texte 0 0
      pos (text_) append cvx exec
      gere_pstricks_proj_opt
fill
end
} def

/pst-trigospherique {
3 dict begin
gsave
   solidlinewidth setlinewidth
   linecolor
   linestyle
   args definition
grestore
end
} def

% END solides.pro

```