

# Le fichier solides.pro

version 3.02, 15 janvier 2008

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

## 1. En-têtes, initialisations de variables globales

Le fichier solides.pro

```
1: %!  
2: % PostScript prologue for pst-solides3d.tex.  
3: % Version 3.02, 2008/01/15  
4: %  
5: %  
6: /SolidesDict 100 dict def  
7: /SolidesbisDict 100 dict def  
8: SolidesDict begin  
9:  
10: %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
11: %% %% les variables globales gerees par PSTricks %%  
12: %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
13: %% %% les lignes dessous sont a dec commenter si l on veut utiliser le  
14: %% %% fichier solides.pro independamment du package PSTricks  
15: %% /Dobs 20 def  
16: %% /THETA 20 def  
17: %% /PHI 50 def  
18: %% /Decran 30 def  
19: %% /XpointVue {Dobs Cos1Cos2 mul} def  
20: %% /YpointVue {Dobs Sin1Cos2 mul} def  
21: %% /ZpointVue {Dobs Sin2 mul} def  
22: %% /xunit 28.14 def  
23: %% /solidhollow false def  
24: %% /solidbiface false def  
25: %% /xunit 28.45 def  
26: %% /tracelignedeniveau? true def  
27: %% /hauteurlignedeniveau 1 def  
28: %% /couleurlignedeniveau {rouge} def  
29: %% /linewidthlignedeniveau 4 def  
30: %% /solidgrid true def  
31: /aretescachees true def  
32: /defaultsolidmode 2 def  
33: /activationgestioncouleurs true def  
34:  
35:  
36: /fillstyle {} def  
37: /startest false def  
38: /cm {} def  
39: /cm_1 {} def  
40: /yunit {xunit} def  
41: /angle_repere 90 def  
42:  
43: /hadjust 2.5 def  
44: /vadjust 2.5 def  
45:  
46: /pointilles {  
47:   [6.25 3.75] 1.25 setdash  
48: } def  
49: /stockcurrentcpath {} def  
50: /newarrowpath {} def  
51:
```

(\*) [melusine.eu.org/syracuse/pstricks/pst-solides3d/](http://melusine.eu.org/syracuse/pstricks/pst-solides3d/)

(\*\*) [melusine.eu.org/syracuse/bbgraf/](http://melusine.eu.org/syracuse/bbgraf/)

## 2. Déclaration d'une fonte accentuée

Le fichier *solides.pro*

```
52: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53: %% choix d'une fonte accentuée pour le .ps %%
54: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
55: /ReEncode { exch findfont
56: dup length dict begin { 1 index /FID eq {pop pop} {def} ifelse
57: }forall /Encoding ISOLatin1Encoding def currentdict end definefont
58: pop }bind def
59: /Font /Times-Roman /ISOfont ReEncode /ISOfont def
60: %Font findfont 10 scalefont setfont
61:
```

## 3. Définitions des couleurs

Le fichier *solides.pro*

```
62: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63: %% extrait de color.pro pour pouvoir récupérer ses couleurs %%
64: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
65: /GreenYellow{0.15 0 0.69 0 setcmykcolor}def
66: /Yellow{0 0 1 0 setcmykcolor}def
67: /Goldenrod{0 0.10 0.84 0 setcmykcolor}def
68: /Dandelion{0 0.29 0.84 0 setcmykcolor}def
69: /Apricot{0 0.32 0.52 0 setcmykcolor}def
70: /Peach{0 0.50 0.70 0 setcmykcolor}def
71: /Melon{0 0.46 0.50 0 setcmykcolor}def
72: /YellowOrange{0 0.42 1 0 setcmykcolor}def
73: /Orange{0 0.61 0.87 0 setcmykcolor}def
74: /BurntOrange{0 0.51 1 0 setcmykcolor}def
75: /Bittersweet{0 0.75 1 0.24 setcmykcolor}def
76: /RedOrange{0 0.77 0.87 0 setcmykcolor}def
77: /Mahogany{0 0.85 0.87 0.35 setcmykcolor}def
78: /Maroon{0 0.87 0.68 0.32 setcmykcolor}def
79: /BrickRed{0 0.89 0.94 0.28 setcmykcolor}def
80: /Red{0 1 1 0 setcmykcolor}def
81: /OrangeRed{0 1 0.50 0 setcmykcolor}def
82: /RubineRed{0 1 0.13 0 setcmykcolor}def
83: /WildStrawberry{0 0.96 0.39 0 setcmykcolor}def
84: /Salmon{0 0.53 0.38 0 setcmykcolor}def
85: /CarnationPink{0 0.63 0 0 setcmykcolor}def
86: /Magenta{0 1 0 0 setcmykcolor}def
87: /VioletRed{0 0.81 0 0 setcmykcolor}def
88: /Rhodamine{0 0.82 0 0 setcmykcolor}def
89: /Mulberry{0.34 0.90 0 0.02 setcmykcolor}def
90: /RedViolet{0.07 0.90 0 0.34 setcmykcolor}def
91: /Fuchsia{0.47 0.91 0 0.08 setcmykcolor}def
92: /Lavender{0 0.48 0 0 setcmykcolor}def
93: /Thistle{0.12 0.59 0 0 setcmykcolor}def
94: /Orchid{0.32 0.64 0 0 setcmykcolor}def
95: /DarkOrchid{0.40 0.80 0.20 0 setcmykcolor}def
96: /Purple{0.45 0.86 0 0 setcmykcolor}def
97: /Plum{0.50 1 0 0 setcmykcolor}def
98: /Violet{0.79 0.88 0 0 setcmykcolor}def
99: /RoyalPurple{0.75 0.90 0 0 setcmykcolor}def
100: /BlueViolet{0.86 0.91 0 0.04 setcmykcolor}def
101: /Periwinkle{0.57 0.55 0 0 setcmykcolor}def
102: /CadetBlue{0.62 0.57 0.23 0 setcmykcolor}def
103: /CornflowerBlue{0.65 0.13 0 0 setcmykcolor}def
104: /MidnightBlue{0.98 0.13 0 0.43 setcmykcolor}def
105: /NavyBlue{0.94 0.54 0 0 setcmykcolor}def
106: /RoyalBlue{1 0.50 0 0 setcmykcolor}def
107: /Blue{1 1 0 0 setcmykcolor}def
108: /Cerulean{0.94 0.11 0 0 setcmykcolor}def
109: /Cyan{1 0 0 0 setcmykcolor}def
110: /ProcessBlue{0.96 0 0 0 setcmykcolor}def
111: /SkyBlue{0.62 0 0.12 0 setcmykcolor}def
112: /Turquoise{0.85 0 0.20 0 setcmykcolor}def
113: /TealBlue{0.86 0 0.34 0.02 setcmykcolor}def
114: /Aquamarine{0.82 0 0.30 0 setcmykcolor}def
115: /BlueGreen{0.85 0 0.33 0 setcmykcolor}def
116: /Emerald{1 0 0.50 0 setcmykcolor}def
117: /JungleGreen{0.99 0 0.52 0 setcmykcolor}def
```

```

118: /SeaGreen{0.69 0 0.50 0 setcmykcolor}def
119: /Green{1 0 1 0 setcmykcolor}def
120: /ForestGreen{0.91 0 0.88 0.12 setcmykcolor}def
121: /PineGreen{0.92 0 0.59 0.25 setcmykcolor}def
122: /LimeGreen{0.50 0 1 0 setcmykcolor}def
123: /YellowGreen{0.44 0 0.74 0 setcmykcolor}def
124: /SpringGreen{0.26 0 0.76 0 setcmykcolor}def
125: /OliveGreen{0.64 0 0.95 0.40 setcmykcolor}def
126: /RawSienna{0 0.72 1 0.45 setcmykcolor}def
127: /Sepia{0 0.83 1 0.70 setcmykcolor}def
128: /Brown{0 0.81 1 0.60 setcmykcolor}def
129: /Tan{0.14 0.42 0.56 0 setcmykcolor}def
130: /Gray{0 0 0 0.50 setcmykcolor}def
131: /Black{0 0 0 1 setcmykcolor}def
132: /White{0 0 0 0 setcmykcolor}def
133: %% fin de l extrait color.pro
134:
135: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
136: %%%          autres couleurs          %%%
137: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
138:
139: /bleu {0 0 1 setrgbcolor} def
140: /rouge {1 0 0 setrgbcolor} def
141: /vert {0 .5 0 setrgbcolor} def
142: /gris {.4 .4 .4 setrgbcolor} def
143: /jaune {1 1 0 setrgbcolor} def
144: /noir {0 0 0 setrgbcolor} def
145: /blanc {1 1 1 setrgbcolor} def
146: /orange {1 .65 0 setrgbcolor} def
147: /rose {1 .01 .58 setrgbcolor} def
148: /cyan {1 0 0 0 setcmykcolor} def
149: /magenta {0 1 0 0 setcmykcolor} def

```

## 4. Passage coordonnées 3d en coordonnées 2d

```

151: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
152: %%%          definition du point de vue          %%%
153: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
154: %% pour la 3D conventionnelle
155: %% Dony : graphisme scientifique : page 187
156: %% Editeur : Masson
157:
158: %% calcul des coefficients de la matrice
159: %% de transformation
160: /Sin1 {THETA sin} def
161: /Sin2 {PHI sin} def
162: /Cos1 {THETA cos} def
163: /Cos2 {PHI cos} def
164: /Cos1Sin2 {Cos1 Sin2 mul} def
165: /Sin1Sin2 {Sin1 Sin2 mul} def
166: /Cos1Cos2 {Cos1 Cos2 mul} def
167: /Sin1Cos2 {Sin1 Cos2 mul} def
168:
169: /3dto2d {
170: 6 dict begin
171:   /Zcote exch def
172:   /Yordonnee exch def
173:   /Xabscisse exch def
174:   /xObservateur
175:     Xabscisse Sin1 mul neg Yordonnee Cos1 mul add
176:   def
177:   /yObservateur
178:     Xabscisse Cos1Sin2 mul neg Yordonnee Sin1Sin2 mul sub Zcote Cos2
179:     mul add
180:   def
181:   /zObservateur
182:     Xabscisse neg Cos1Cos2 mul Yordonnee Sin1Cos2 mul sub Zcote Sin2
183:     mul sub Dobs add
184:   def
185:   %% maintenant on depose les resultats sur la pile
186:   Decran xObservateur mul zObservateur div cm

```

```

187:   Decran yObservateur mul zObservateur div cm
188: end
189: } def
190:
191: /getpointVue {
192:   XpointVue
193:   YpointVue
194:   ZpointVue
195: } def
196:
197: /GetCamPos {
198:   getpointVue
199: } def
200:

```

## 5. Transcription PStricks → jps

```

201: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
202: %%%          jps modifie pour PStricks          %%%
203: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
204:
205: /solid {continu} def
206: /dashed {pointilles} def
207:

```

## 6. Pour l'option algebraic

```

208: %% les 3 procedures utilisees pour transformer les depots de AlgToPs en nombres
209: /pstrickactionR3 {
210: 3 dict begin
211:   /len@3 exch def
212:   /len@2 exch def
213:   /len@1 exch def
214:   len@1 exec
215:   len@2 exec
216:   len@3 exec
217: end
218: } def
219:
220: /pstrickactionR2 {
221:   exec exch exec exch
222: } def
223:
224: /pstrickactionR {
225:   exec
226: } def
227:

```

## 7. Géométrie 2d basique

```

228: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
229: %%%          geometrie basique          %%%
230: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
231:
232: %% syntaxe~: [x1 y1 ... xn yn] ligne
233: /ligne {
234:   gsave
235:   newpath
236:   dup 0 getp smoveto
237:   ligne_

```

```

238:     starfill
239:     stroke
240: grestore
241: } def
242:
243: %% syntaxe~: [x1 y1 ... xn yn] ligne_
244: /ligne_ {
245:     reversep
246:     aload length 2 idiv
247:     {
248:         slineto
249:     } repeat
250: } def
251:
252: %% syntaxe~: [x1 y1 ... xn yn] polygone
253: /polygone* {
254: 1 dict begin
255:     /startest {true} def
256:     polygone
257: end
258: } def
259:
260: /polygone {
261:     gsave
262:     newpath
263:         aload length 2 idiv
264:         3 copy pop
265:         smoveto
266:         {
267:             slineto
268:         } repeat
269:     closepath
270:     starfill
271:     currentlinewidth 0 eq {} {stroke} ifelse
272:     grestore
273: } def
274:
275: %% syntaxe : x y point
276: /point {
277: gsave
278:     1 setlinecap
279:     newpath
280:         smoveto
281:         0 0 rlineto
282:         5 setlinewidth
283:     stroke
284: grestore
285: } def
286:

```

## 8. Insertion librairie jps

### 8.1 - Le repère utilisateur (repère jps)

```

287: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
288: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
289: %%%                                     %%%
290: %%%             insertion librairie jps             %%%
291: %%%                                     %%%
292: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
293: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
294:
295: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
296: %%%                                     %%%
297: %%%             le repere jps             %%%
298: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
299: %%% ## AAAscale ##
300: %%% les déplacements a l echelle %%%%%%%%%%%%%%%%%%%
301:
302: /v@ct_I {xunit 0} def

```

```

303: /v@ct_J {angle_repere cos yunit mul angle_repere sin yunit mul} def
304:
305: /xscale {} def
306: /yscale {} def
307:
308: /xscale-1 {} def
309: /yscale-1 {} def
310:
311: /gtransform {} def
312: /gtransform-1 {} def
313:
314: /jtoppoint {
315: 2 dict begin
316:   gtransform
317:   /y exch yscale def
318:   /x exch xscale def
319:   v@ct_I x mulv
320:   v@ct_J y mulv
321:   addv
322: end
323: } def
324:
325: /rptojpoint {
326:   xtranslate ytranslate
327:   3 1 roll      %% xA yB yA xB
328:   4 1 roll      %% xB xA yB yA
329:   sub neg 3 1 roll %% yB-yA xB xA
330:   sub neg exch
331:   ptojpoint
332: } def
333:
334: /rptoppoint {
335:   xtranslate ytranslate
336:   3 1 roll      %% xA yB yA xB
337:   4 1 roll      %% xB xA yB yA
338:   sub neg 3 1 roll %% yB-yA xB xA
339:   sub neg exch
340: } def
341:
342: /ptojpoint {
343: 4 dict begin
344:   /Y exch yscale-1 def
345:   /X exch xscale-1 def
346:   /y Y yunit angle_repere sin mul div def
347:   /x X y yunit mul angle_repere cos mul sub xunit div def
348:   x y
349:   gtransform-1
350: end
351: } def
352:
353: /smoveto {
354:   jtoppoint
355:   moveto
356: } def
357:
358: /srmoveto {
359:   jtoppoint
360:   rmoveto
361: } def
362:
363: /slineto {
364:   jtoppoint
365:   lineto
366: } def
367:
368: /srlineto {
369:   jtoppoint
370:   rlineto
371: } def
372:
373: /stranslate {
374:   jtoppoint
375:   translate
376: } def
377:
378: %%%% ### fin insertion ###

```

379 :

## 8.2 - Routines de tests

..... Le fichier solides.pro .....

```

380: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
381: %%%          les tests          %%%
382: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
383:
384: %%% ## isbool ##
385: %% syntaxe : any isbool --> booleen
386: /isbool {
387:     type (booleantype) cvn eq
388: } def
389:
390: %%% ## isarray ##
391: %% syntaxe : any isarray --> booleen
392: /isarray {
393:     type (arraytype) cvn eq
394: } def
395:
396: %%% ## isstring ##
397: %% syntaxe : any isstring --> booleen
398: /isstring {
399:     type (stringtype) cvn eq
400: } def
401:
402: %%% ## isinteger ##
403: %% syntaxe : any isinteger --> booleen
404: /isinteger {
405:     type (integertype) cvn eq
406: } def
407:
408: %%% ## isnum ##
409: %% syntaxe : any isnum --> booleen
410: /isnum {
411:     dup isreal
412:     exch isinteger or
413: } def
414:
415: %%% ## isreal ##
416: %% syntaxe : any isreal --> booleen
417: /isreal {
418:     type (realtype) cvn eq
419: } def
420:
421: %%% ## eq ##
422: %% syntaxe : A B eqp3d --> booleen = true si les points A et B sont identiques
423: /eqp3d {
424:     %% x1 y1 z1 x2 y2 z2
425:     4 -1 roll %% x1 y1 x2 y2 z2 z1
426:     eq {
427:         eqp
428:     } {
429:         pop pop pop pop false
430:     } ifelse
431: } def
432:
433: %% syntaxe : A B eqp --> booleen = true si les points A et B sont identiques
434: /eqp {
435:     3 -1 roll
436:     eq
437:     {
438:         eq
439:         {true}
440:         {false}
441:         ifelse
442:     }
443:     {pop pop false}
444:     ifelse
445: } def
446:

```

```

447: %% syntaxe : z z' eqc --> true si z = z', false sinon
448: /eqc {
449:     eqp
450: } def
451:
452: %%%% ### fin insertion ###
453:

```

### 8.3 - Conversions de types

```

454: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
455: %%%%                                conversions de types                                %%%%
456: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
457:
458: %%%% ### astr2str ###
459: %% syntaxe : array str astr2str --> str
460: %% convertit le contenu de array en chaines de caracteres puis les
461: %% concatene avec str, en inserant un caractere "space" apres chaque
462: %% element du tableau array
463: /astr2str {
464: 5 dict begin
465:   /str exch def
466:   /table exch def
467:   /n table length def
468:   n 0 eq {
469:     str
470:   } {
471:     table 0 n 1 sub getinterval
472:     table n 1 sub get (                               ) cvs
473:     ( ) append
474:     str append
475:     astr2str
476:   } ifelse
477: end
478: } def
479:
480: %%%% ### numstr2array ###
481: %% syntaxe : str numstr2array -> array
482: %% ou str est une chaine de nombres entiers separes par des espaces
483: %% et array est constitue des elements numeriques entiers de string.
484: %% exemple :
485: %% (0 12 4 54) --> [0 12 4 54]
486: /numstr2array {
487: 3 dict begin
488:   /str exch def
489:   /n str length def
490:   /j -1 def
491:   [
492:     0 1 n 1 sub {
493:       /i exch def
494:       /j j 1 add store
495:       str i get
496:       dup 32 eq {
497:         %% c est un espace
498:         /j -1 store
499:         pop
500:       } {
501:         j 1 ge {
502:           exch 10 mul 48 sub add
503:         } {
504:           48 sub
505:         } ifelse
506:       } ifelse
507:     } for
508:   ]
509: end
510: } def
511:
512: %% syntaxe : array numstr2array -> array
513: /arraynumstr2arrayarray {
514:   {numstr2array} apply

```



```
515: } def
516:
517: %%%% ### fin insertion ###
518:
```

## 8.4 - Projection de chaînes de caractères

..... Le fichier solides.pro .....

```
519: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
520: %%%%          macros de projection          %%%%
521: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
522:
523: %%%% ### projtext ###
524: /initpr@jtext {
525: 5 dict begin
526:   dup isbool {
527:     /mybool exch def
528:   } {
529:     /mybool true def
530:   } ifelse
531:   dup isarray {
532:     %% c est un planprojpath
533:     /type_plan_proj true def
534:     /table exch def
535:     /z0 exch def
536:     /y0 exch def
537:     /x0 exch def
538:     0 0
539:   } {
540:     %% c est un solidprojpath
541:     /type_plan_proj false def
542:     %% y a-t-il un str2
543:     dup isstring {
544:       /str2 exch def
545:     } {
546:       /str2 {} def
547:     } ifelse
548:     %% y a-t-il un alpha
549:     2 copy pop issolid {
550:       /alpha 0 def
551:     } {
552:       /alpha exch def
553:     } ifelse
554:     /i exch def
555:     /solid exch def
556:     0 0
557:   } ifelse
558: } def
559: /closepr@jtext {
560:   type_plan_proj {
561:     x0 y0 z0 table mybool projpath
562:   } {
563:     solid i alpha str2 mybool projpath
564:   } ifelse
565:   fill
566:   stroke
567: end
568: } def
569:
570: %% syntaxe : str x0 y0 z0 [normal_vect] ultextp3d --> -
571: %% syntaxe : str x0 y0 z0 [normal_vect] bool ultextp3d --> -
572: %% syntaxe : str1 solid i str2 ultextp3d --> -
573: %% syntaxe : str1 solid i str2 bool ultextp3d --> -
574: %% syntaxe : str1 solid i alpha str2 bool ultextp3d --> -
575: /ultextp3d {initpr@jtext ultext_ closepr@jtext} def
576: /cltextp3d {initpr@jtext cltext_ closepr@jtext} def
577: /bltextp3d {initpr@jtext bltext_ closepr@jtext} def
578: /dltextp3d {initpr@jtext dltext_ closepr@jtext} def
579: /ubtextp3d {initpr@jtext ubtext_ closepr@jtext} def
580: /cbtextp3d {initpr@jtext cbtext_ closepr@jtext} def
581: /bbtextp3d {initpr@jtext bbtext_ closepr@jtext} def
582: /dbtextp3d {initpr@jtext dbtext_ closepr@jtext} def
```

```

583: /uctextp3d {initpr@jtext uctext_ closepr@jtext} def
584: /cctextp3d {initpr@jtext cctext_ closepr@jtext} def
585: /bctextp3d {initpr@jtext bctext_ closepr@jtext} def
586: /dctextp3d {initpr@jtext bctext_ closepr@jtext} def
587: /urtextp3d {initpr@jtext urtext_ closepr@jtext} def
588: /crtextp3d {initpr@jtext crtext_ closepr@jtext} def
589: /brtextp3d {initpr@jtext brtext_ closepr@jtext} def
590: /drtextp3d {initpr@jtext brtext_ closepr@jtext} def
591:

```

## 8.5 - Appliquer une transformation à un chemin

```

592: %%%% ### currentppathtransform ###
593: %% syntaxe : {f} currentppathtransform --> applique la transformation f
594: %% au chemin courant
595: /currentppathtransform {
596: 6 dict begin
597:   /warp exch def
598:   %% pour remplacer 'move'
599:   /warpmove{
600:     2 index {
601:       newpath
602:     } if
603:     warp moveto
604:     pop false
605:   } def
606:
607:   %% pour remplacer 'lineto'
608:   /warpline {
609:     warp lineto
610:   } bind def
611:
612:   %% pour remplacer 'curveto'
613:   /warpcurve {
614:     6 2 roll warp
615:     6 2 roll warp
616:     6 2 roll warp
617:     curveto
618:   } bind def
619:
620:   true
621:   { warpmove } { warpline } { warpcurve } { closepath } pathforall
622:   pop
623: end
624: } def
625:
626: %% syntaxe : {f} currentpathtransform --> applique la transformation f
627: %% au chemin courant
628: /currentpathtransform {
629: 7 dict begin
630:   /transform exch def
631:   /warp {ptojpoint transform} def
632:   %% pour remplacer 'move'
633:   /warpmove{
634:     2 index {
635:       newpath
636:     } if
637:     warp smoveto
638:     pop false
639:   } def
640:
641:   %% pour remplacer 'lineto'
642:   /warpline {
643:     warp slineto
644:   } bind def
645:
646:   %% pour remplacer 'curveto'
647:   /warpcurve {
648:     6 2 roll warp
649:     6 2 roll warp
650:     6 2 roll warp

```

```

651:      scurveto
652:    } bind def
653:
654:      true
655:    { warpmove } { warpline } { warpcurve } { closepath } pathforall
656:    pop
657:  end
658: } def
659:

```

## 8.6 - Base orthonormale à partie de la normale

..... Le fichier solides.pro .....

```

660: %%%% ### normalvect_to_orthobase ###
661: %% syntaxe : [normal_vect] normalvect_to_orthobase
662: %% --> imI imJ imK
663: /normalvect_to_orthobase {
664: 4 dict begin
665:   dup length 3 eq {
666:     aload pop normalize3d /normal_vect defpoint3d
667:     normal_vect -1 0 0 eqp3d {
668:       /imageI {0 -1 0} def
669:       /imageK {-1 0 0} def
670:       /imageJ {0 0 1} def
671:     } {
672:       %% on calcule 1 image de la base (I,J,K)
673:       /imageJ {normal_vect 1 0 0 vectprod3d normalize3d} def
674:       /imageK {normal_vect} def
675:       /imageI {imageJ imageK vectprod3d} def
676:       1 0 0 imageK angle3d 0 eq {
677:         0 1 0 normal_vect vectprod3d /imageI defpoint3d
678:         /imageJ {0 1 0} def
679:         normal_vect /imageK defpoint3d
680:       } if
681:     } ifelse
682:   } {
683:     dup length 6 eq {
684:       aload pop
685:       normalize3d /imageK defpoint3d
686:       normalize3d /imageI defpoint3d
687:       imageK imageI vectprod3d /imageJ defpoint3d
688:     } {
689:       dup length 7 eq {
690:         aload pop
691:         /alpha exch 2 div def
692:         normalize3d /imageK defpoint3d
693:         normalize3d /imageI defpoint3d
694:         imageK imageI vectprod3d /imageJ defpoint3d
695:         %% et ensuite, on fait tourner la base autour de imageK
696:         imageI alpha cos mulv3d
697:         imageJ alpha sin mulv3d
698:         addv3d
699:
700:         imageI alpha sin neg mulv3d
701:         imageJ alpha cos mulv3d
702:         addv3d
703:
704:         /imageJ defpoint3d
705:         /imageI defpoint3d
706:       } {
707:         %% length = 4
708:         aload pop
709:         /alpha exch def
710:         normalize3d /normal_vect defpoint3d
711:
712:         normal_vect -1 0 0 eqp3d {
713:           /imageI {0 -1 0} def
714:           /imageK {-1 0 0} def
715:           /imageJ {0 0 1} def
716:         } {
717:           %% on calcule 1 image de la base (I,J,K)
718:           /imageJ {normal_vect 1 0 0 vectprod3d normalize3d} def

```

```

719:         /imageK {normal_vect} def
720:         /imageI {imageJ imageK vectprod3d} def
721:         1 0 0 imageK angle3d 0 eq {
722:             0 1 0 normal_vect vectprod3d /imageI defpoint3d
723:             /imageJ {0 1 0} def
724:             normal_vect /imageK defpoint3d
725:         } if
726:     } ifelse
727: } ifelse
728:
729: %% et ensuite, on fait tourner la base autour de imageK
730: imageI alpha cos mulv3d
731: imageJ alpha sin mulv3d
732: addv3d
733:
734: imageI alpha sin neg mulv3d
735: imageJ alpha cos mulv3d
736: addv3d
737:
738:     /imageJ defpoint3d
739:     /imageI defpoint3d
740: } ifelse
741: } ifelse
742: imageI
743: imageJ
744: imageK
745: end
746: } def
747:

```

## 8.7 - Projection d'un chemin

```

748: %%%% ### projpath ###
749: %% syntaxe : x y z [normal] projpath --> planprojpath
750: %% syntaxe : x y z [normal] bool projpath --> planprojpath
751: %% syntaxe : solid i projpath --> solidprojpath
752: %% syntaxe : solid i bool projpath --> solidprojpath
753: %% syntaxe : solid i str bool projpath --> solidprojpath
754: %% syntaxe : solid i alpha str bool projpath --> solidprojpath
755: /projpath {
756: 2 dict begin
757:     dup isbool {
758:         /mybool exch def
759:     } {
760:         /mybool true def
761:     } ifelse
762:     dup isarray {
763:         mybool planprojpath
764:     } {
765:         mybool solidprojpath
766:     } ifelse
767: end
768: } def
769:
770: %% syntaxe : solid i str bool solidprojpath --> -
771: %% ou
772: %% syntaxe : solid i alpha str bool solidprojpath --> -
773: %% projette le chemin courant sur la face i du solide, apres
774: %% eventuellement une rotation d angle alpha autour de la normale
775: %% bool : pour savoir si on tient compte de la visibilite
776: /solidprojpath {
777: 5 dict begin
778:     /visibility exch def
779:     dup isstring {
780:         /option exch def
781:     } if
782:     2 copy pop
783:     issolid {
784:         /alpha 0 def
785:     } {
786:         /alpha exch def

```

```

787:   } ifelse
788:   /i exch def
789:   /solid exch def
790:   solid issolid not {
791:     (Error : mauvais type d argument dans solidprojpath) ==
792:   } if
793:   /n solid solidnombrefaces def
794:   i n 1 sub le {
795:     visibility not solid i solidfacevisible? or {
796:       currentdict /option known {
797:         option cvx exec
798:       } {
799:         solid i solidcentreface
800:       } ifelse
801:     [
802:       solid 0 i solidgetsommetface
803:       solid 1 i solidgetsommetface
804:       vecteur3d normalize3d
805:       solid i solidnormaleface alpha
806:     ] false planprojpath
807:   } {
808:     newpath 0 0 smoveto
809:   } ifelse
810: } {
811:   (Error : indice trop grand dans solidprojpath) ==
812:   quit
813: } ifelse
814: end
815: } def
816:
817: %% syntaxe : x y z [normal] bool planprojpath
818: /planprojpath {
819: 6 dict begin
820:   /visibility exch def
821:   %% on calcule l image de la base (I,J,K)
822:   normalvect_to_orthobase
823:   /imageK defpoint3d
824:   /imageJ defpoint3d
825:   /imageI defpoint3d
826:   /z exch def
827:   /y exch def
828:   /x exch def
829:
830:   visibility not x y z imageK planvisible? or {
831:     {ptojpoint 0
832:      imageI
833:      imageJ
834:      imageK
835:      transformpoint3d
836:      x y z addv3d
837:      3dto2d jtoppoint} currentppathtransform
838:   } {
839:     newpath
840:   } ifelse
841: end
842: } def
843:
844: %%%% ### fin insertion ###
845:

```

## 8.8 - Courbes de fonctions

```

846: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
847: %%%          fonctions numeriques          %%%
848: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
849:
850: %%%% ### courbeparam ###
851: /setresolution {
852:   /resolution exch def
853: } def
854: /resolution 200 def

```

```

855:
856: /courbe_dic 2 dict def
857: courbe_dic /X {} put
858: courbe_dic /Y {} put
859:
860: %% syntaxe : tmin tmax C@urbeparam_
861: /C@urbeparam_ {
862: 6 dict begin
863:   /tmax@ exch def
864:   /tmin@ exch def
865:   /t tmin@ def
866:   /dt tmax@ tmin@ sub resolution 1 sub div def
867:   tmin@ courbe_dic /X get exec
868:   pstrickactionR
869:   tmin@ courbe_dic /Y get exec
870:   pstrickactionR
871:   smoveto
872:   resolution 1 sub
873:   {
874:     t courbe_dic /X get exec
875:     pstrickactionR
876:     t courbe_dic /Y get exec
877:     pstrickactionR
878:     slineto
879:
880:     /t t dt add store           %% on incremente
881:   }
882:   repeat
883:     tmax@ courbe_dic /X get exec
884:     pstrickactionR
885:     tmax@ courbe_dic /Y get exec
886:     pstrickactionR
887:     slineto
888:   end
889: } def
890:
891: %% syntaxe : tmin tmax {X} {Y} Courbeparam_
892: /Courbeparam_ {
893:   courbe_dic exch /Y exch put
894:   courbe_dic exch /X exch put
895:   C@urbeparam_
896: } def
897:
898: %% syntaxe : {X} {Y} courbeparam_
899: /courbeparam_ {
900:   tmin tmax
901:   4 -1 roll
902:   4 -1 roll
903:   Courbeparam_
904: } def
905:
906: %% syntaxe : tmin tmax {X} {Y} Courbeparam
907: /Courbeparam {
908:   gsave
909:   6 dict begin
910:     dup isstring
911:     {
912:       /option exch def
913:     }
914:     if
915:     courbe_dic exch /Y exch put
916:     courbe_dic exch /X exch put
917:     /tmax exch def
918:     /tmin exch def
919:
920:     newpath
921:     tmin courbe_dic /X get exec
922:     pstrickactionR
923:     tmin courbe_dic /Y get exec
924:     pstrickactionR
925:     smoveto           %% on commence le chemin
926:     tmin tmax C@urbeparam_
927:     starfill
928:
929:     stockcurrentcpath
930:     newarrowpath

```

```

931:   currentdict /option known
932:   {
933:     /dt tmax tmin sub resolution 1 sub div def
934:     tmin dt add courbe_dic /X get exec
935:     tmin dt add courbe_dic /Y get exec
936:     tmin courbe_dic /X get exec
937:     tmin courbe_dic /Y get exec
938:     arrowpath0
939:     tmax dt sub courbe_dic /X get exec
940:     tmax dt sub courbe_dic /Y get exec
941:     tmax courbe_dic /X get exec
942:     tmax courbe_dic /Y get exec
943:     currentdict /dt undef
944:     arrowpath1
945:     option
946:     gere_arrowhead
947:   }
948:   if
949:
950:   currentlinewidth 0 eq {} {stroke} ifelse
951:
952: end
953: grestore
954: } def
955:
956: %% syntaxe : {X} {Y} courbeparam
957: /courbeparam {
958:   dup isstring
959:   {
960:     tmin tmax
961:     5 -1 roll
962:     5 -1 roll
963:     5 -1 roll
964:   }
965:   {
966:     tmin tmax
967:     4 -1 roll
968:     4 -1 roll
969:   }
970:   ifelse
971:   Courbeparam
972: } def
973:
974: %% syntaxe : tmin tmax {X} {Y} Courbeparam*
975: /Courbeparam* {
976: 1 dict begin
977:   /startest {true} def
978:   Courbeparam
979: end
980: } def
981:
982: %% syntaxe : {X} {Y} courbeparam*
983: /courbeparam* {
984: 1 dict begin
985:   /startest {true} def
986:   courbeparam
987: end
988: } def
989:
990: %%%% ### courbe ###
991: %% syntaxe : {f} courbe
992: /courbe {
993:   dup isstring   %% y a-t-il une option de fin de ligne ?
994:   {
995:     xmin xmax
996:     {}
997:     5 -1 roll
998:     5 -1 roll
999:   }
1000:  {
1001:     xmin xmax
1002:     {}
1003:     4 -1 roll
1004:   }
1005:   ifelse
1006:   Courbeparam

```

```

1007: } def
1008:
1009: %% syntaxe : mini maxi {f} Courbe
1010: /Courbe {
1011:     dup isstring {
1012:         {}
1013:         3 -1 roll
1014:         3 -1 roll
1015:     } {
1016:         {}
1017:         2 -1 roll
1018:     } ifelse
1019:     Courbeparam
1020: } def
1021:
1022: %% syntaxe : {f} courbe_
1023: /courbe_ {
1024:     xmin xmax
1025:     {}
1026:     4 -1 roll
1027:     Courbeparam_
1028: } def
1029:
1030: %% syntaxe : mini maxi {f} Courbe_
1031: /Courbe_ {
1032:     {}
1033:     2 -1 roll
1034:     Courbeparam_
1035: } def
1036:
1037: %% syntaxe : mini maxi {f} Courbe*
1038: /Courbe* {
1039: 1 dict begin
1040:     /startest {true} def
1041:     Courbe
1042: end
1043: } def
1044:
1045: %% syntaxe : {f} courbe*
1046: /courbe* {
1047: 1 dict begin
1048:     /startest {true} def
1049:     courbe
1050: end
1051: } def
1052:
1053: %%%% ### courbeR2 ###
1054: %% syntaxe : tmin tmax C@urbeR2_
1055: /C@urbeR2_ {
1056: 6 dict begin
1057:     /tmax@ exch def
1058:     /tmin@ exch def
1059:     /t tmin@ def
1060:     /dt tmax@ tmin@ sub resolution 1 sub div def
1061:     tmin@ courbe_dic /X get exec
1062:     pstrickactionR2
1063:     smoveto
1064:     /t t dt add store
1065:     resolution 2 sub
1066:     {
1067:         t courbe_dic /X get exec
1068:         pstrickactionR2
1069:         slineto
1070:         /t t dt add store           %% on incremente
1071:     }
1072:     repeat
1073:     tmax@ courbe_dic /X get exec
1074:     pstrickactionR2
1075:     slineto
1076: end
1077: } def
1078:
1079: %% syntaxe : tmin tmax {X} CourbeR2_
1080: /CourbeR2_ {
1081:     courbe_dic exch /X exch put
1082:     C@urbeR2_

```



```

1083: } def
1084:
1085: %% syntaxe : {X} courbeR2_
1086: /courbeR2_ {
1087:   tmin tmax
1088:   3 -1 roll
1089:   3 -1 roll
1090:   CourbeR2_
1091: } def
1092:
1093: %% syntaxe : tmin tmax {X} CourbeR2
1094: /CourbeR2+ {
1095: 2 dict begin
1096:   /slneto {} def
1097:   /smoveto {} def
1098:   CourbeR2
1099: end
1100: } bind def
1101:
1102: /CourbeR2 {
1103: gsave
1104: 6 dict begin
1105:   dup isstring
1106:   {
1107:     /option exch def
1108:   }
1109:   if
1110:   courbe_dic exch /X exch put
1111:   /tmax exch def
1112:   /tmin exch def
1113:
1114:   newpath
1115:   tmin tmax C@urbeR2_
1116:   starfill
1117:   currentlinewidth 0 eq {} {stroke} ifelse
1118:
1119: end
1120: grestore
1121: } def
1122:
1123: %% syntaxe : {X} courbeR2
1124: /courbeR2 {
1125:   tmin tmax
1126:   3 -1 roll
1127:   CourbeR2
1128: } def
1129:
1130: %% syntaxe : tmin tmax {X} CourbeR2*
1131: /CourbeR2* {
1132: 1 dict begin
1133:   /startest {true} def
1134:   CourbeR2
1135: end
1136: } def
1137:
1138: %% syntaxe : {X} {Y} courbeR2*
1139: /courbeR2* {
1140: 1 dict begin
1141:   /startest {true} def
1142:   courbeR2
1143: end
1144: } def
1145:
1146: %%%% ### courbeR3 ###
1147: %% syntaxe : t1 t2 {f} (option) CourbeR3
1148: /CourbeR3 {
1149: 2 dict begin
1150:   dup isstring {
1151:     /option exch def
1152:   } if
1153:   /lafonction exch def
1154:   {lafonction 3dto2d}
1155:   currentdict /option known
1156:   {option}
1157:   if
1158:   CourbeR2

```

```

1159: end
1160: } def
1161:
1162: %% syntaxe : {f} (option) CourbeR3
1163: /courbeR3 {
1164:     tmin tmax 3 -1 roll CourbeR3
1165: } def
1166:
1167: %%%% ### fin insertion ###
1168:

```

## 8.9 - Constantes et fonctions mathématiques

```

1169: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1170: %%%%      fonctions et constantes mathematiques      %%%%
1171: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1172:
1173: %%%% ### math ###
1174: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1175:
1176: /pi 3.14159 def
1177: /e 2.71828 def
1178:
1179: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1180:
1181: /rd {180 pi div mul} def          %% transforme des rd en degres
1182: /deg {pi mul 180 div} def         %% transforme des degres en rd
1183: /log {ln 10 ln div} def
1184: /Exp {e exch exp} def
1185: /Cos {rd cos} def
1186: /Sin {rd sin} def
1187: /tan {dup sin exch cos div} def
1188: /cotan {dup cos exch sin div} def
1189: /Tan {dup Sin exch Cos div} def
1190: /Cotan {dup Cos exch Sin div} def
1191: /coTan {Cotan} def
1192: /arctan {
1193:     dup 0 ge
1194:         {1 atan}
1195:         {neg 1 atan neg}
1196: ifelse
1197: } def
1198: /Arctan {arctan deg} def
1199: /arccos {
1200:     dup
1201:     dup mul neg 1 add sqrt
1202:     exch
1203:     atan
1204: } def
1205: /Arccos {arccos deg} def
1206: /arcsin {
1207:     dup 1 eq {
1208:         90
1209:     } {
1210:         dup
1211:         dup mul neg 1 add sqrt
1212:         atan
1213:         dup 90 lt
1214:         {}
1215:         {360 sub}
1216:     ifelse
1217:     } ifelse
1218: } def
1219: /Arcsin {arcsin deg} def
1220: /cosh {dup Exp exch neg Exp add 2 div} def
1221: /sinh {dup Exp exch neg Exp sub 2 div} def
1222: /tanh {dup sinh exch cosh div} def
1223: /cotanh {dup cosh exch sinh div} def
1224: /argcosh {dup dup mul 1 sub sqrt add ln} def
1225: /argsinh {dup dup mul 1 add sqrt add ln} def
1226: /argtanh {

```

```

1227:   setxvar
1228:   x 1 add
1229:   1 x sub
1230:   div
1231:   ln
1232:   2 div
1233: } def
1234: /factorielle {
1235:   dup 0 eq
1236:   {pop 1}
1237:   {dup 1 sub factorielle mul}
1238:   ifelse
1239: } def
1240: /Gauss {
1241: 3 dict begin
1242:   /sigma exch def
1243:   /m exch def
1244:   /x exch def
1245:   x m sub dup mul sigma dup mul 2 mul div neg Exp
1246:   2 pi mul sigma dup mul mul sqrt div
1247: end
1248: } def
1249:
1250: %%%% ### max ###
1251: /max {
1252:   2 copy
1253:   lt {exch} if
1254:   pop
1255: } def
1256:
1257: %%%% ### min ###
1258: /min {
1259:   2 copy
1260:   gt {exch} if
1261:   pop
1262: } def
1263:

```

## 8.10 - Divers

```

1264: %%%% ### setcolor ###
1265: %% syntaxe : tableau setcolor
1266: /setcolor {
1267:   dup length 4 eq
1268:   {aload pop setcmykcolor}
1269:   {aload pop setrgbcolor}
1270:   ifelse
1271: } def
1272:
1273: %%%% ### in ###
1274: %% cherche si un elt donne appartient au tableau donne
1275: %% rque : utilise 3 variables locales
1276: %% syntaxe : elt array in --> index boolean
1277: /in {
1278: 3 dict begin
1279:   /liste exch def
1280:   /elt exch def
1281:   /i 0 def
1282:   0 false           %% la reponse a priori
1283:   liste length {
1284:     liste i get elt eq {
1285:       pop pop       %% en enleve la reponse
1286:       i true        %% pour mettre la bonne
1287:       exit
1288:     } if
1289:     /i i 1 add store
1290:   } repeat
1291: end
1292: } def
1293:
1294: %%%% ### starfill ###

```

```

1295: %% la procedure pour les objets "star"
1296: %% si c est "star" on fait le fillstyle, sinon non
1297: /starfill {
1298:   startest {
1299:     gsave
1300:     clip
1301:     fillstyle
1302:     grestore
1303:   /startest false def
1304:   } if
1305: } def
1306:
1307: %%%% ### addv ###
1308: %% syntaxe : u v addv --> u+v
1309: /addv {
1310:   3 1 roll    %% xA yA xB yB
1311:   4 1 roll    %% xA yB yA xB
1312:   add 3 1 roll %% yB+yA xB xA
1313:   add exch
1314: } def
1315:
1316: %%%% ### continu ###
1317: /continu {
1318:   [] 0 setdash
1319: } def
1320:
1321: %%%% ### trigospherique ###
1322: %% passage spherique --> cartesiennes
1323: %% les formules de passage ont été récupérées ici :
1324: %% http://fr.wikipedia.org/wiki/Coordonn%C3%A9es\_polaires
1325: %% syntaxe : r theta phi rtp2xyz -> x y z
1326: /rtp2xyz {
1327: 6 dict begin
1328:   /phi exch def
1329:   /theta exch def
1330:   /r exch def
1331:   /x phi sin theta cos mul r mul def
1332:   /y phi sin theta sin mul r mul def
1333:   /z phi cos r mul def
1334:   x y z
1335: end
1336: } def
1337:
1338: %% trace d'un arc sur une sphere de centre 0
1339: %% syntaxe : r thetal phi1 r theta2 phi2 arcspherique
1340: /arcspherique {
1341: 9 dict begin
1342:   dup isstring {
1343:     /option exch def
1344:   } if
1345:   /phi2 exch def
1346:   /theta2 exch def
1347:   pop
1348:   /phil exch def
1349:   /thetal exch def
1350:   /r exch def
1351:   /n 12 def
1352:
1353:   1 thetal phi1 rtp2xyz /u defpoint3d
1354:   1 theta2 phi2 rtp2xyz /v defpoint3d
1355:   u v vectprod3d u vectprod3d dupp3d norme3d 1 exch div mulv3d /w defpoint3d
1356:
1357:   /sinalpha u v vectprod3d norme3d def
1358:   /cosalpha u v scalprod3d def
1359:   /alpha sinalpha cosalpha atan def
1360:   /n 12 def
1361:   /pas alpha n div def
1362:
1363:   gsave
1364:   /t pas neg def
1365:   [
1366:     n 1 add {
1367:       /t t pas add store
1368:       u t cos r mul mulv3d
1369:       w t sin r mul mulv3d
1370:       addv3d

```

```

1371:         } repeat
1372:     ]
1373:     currentdict /option known {
1374:         option
1375:     } if
1376:     ligne3d
1377:     grestore
1378: end
1379: } def
1380:
1381: %% trace d'un arc sur une sphere de centre O
1382: %% syntaxe : r theta1 phi1 r theta2 phi2 arcspherique
1383: /arcspherique_ {
1384: 8 dict begin
1385:   /phi2 excl def
1386:   /theta2 excl def
1387:   pop
1388:   /phi1 excl def
1389:   /theta1 excl def
1390:   /r excl def
1391:   /n 12 def
1392:
1393:   1 theta1 phi1 rtp2xyz /u defpoint3d
1394:   1 theta2 phi2 rtp2xyz /v defpoint3d
1395:   u v vectprod3d u vectprod3d dupp3d norme3d 1 excl div mulv3d /w defpoint3d
1396:
1397:   /sinalpha u v vectprod3d norme3d def
1398:   /cosalpha u v scalprod3d def
1399:   /alpha sinalpha cosalpha atan def
1400:   /n 12 def
1401:   /pas alpha n div def
1402:
1403:   /t pas neg def
1404:   [
1405:     n 1 add {
1406:       /t t pas add store
1407:       u t cos r mul mulv3d
1408:       w t sin r mul mulv3d
1409:       addv3d
1410:     } repeat
1411:   ] ligne3d_
1412: end
1413: } def
1414:
1415: %% trace d'une geodesique sur une sphere de centre O
1416: %% syntaxe : r theta1 phi1 r theta2 phi2 geodesique_sphere
1417: /geodesique_sphere {
1418: 13 dict begin
1419:   /phi2 excl def
1420:   /theta2 excl def
1421:   pop
1422:   /phi1 excl def
1423:   /theta1 excl def
1424:   /r excl def
1425:   /n 360 def
1426:
1427:   1 theta1 phi1 rtp2xyz /u defpoint3d
1428:   1 theta2 phi2 rtp2xyz /v defpoint3d
1429:   u v vectprod3d u vectprod3d dupp3d norme3d 1 excl div mulv3d /w defpoint3d
1430:
1431:   /sinalpha u v vectprod3d norme3d def
1432:   /cosalpha u v scalprod3d def
1433:   /alpha sinalpha cosalpha atan def
1434:   /pas 360 n div def
1435:
1436:   gsave
1437:   /t pas neg def
1438:   [
1439:     n 1 add {
1440:       /t t pas add store
1441:       u t cos r mul mulv3d
1442:       w t sin r mul mulv3d
1443:       addv3d
1444:     } repeat
1445:   ] ligne3d
1446:   grestore

```

```

1447: end
1448: } def
1449:
1450:
1451: %% syntaxe : A B C trianglespherique --> trace le rtriangle ABC
1452: %% (coordonnees spheriques)
1453: /trianglespherique* {
1454: 1 dict begin
1455:   /startest {true} def
1456:   trianglespherique
1457: end
1458: } def
1459:
1460: /trianglespherique {
1461: 10 dict begin
1462:   /C defpoint3d
1463:   /B defpoint3d
1464:   /A defpoint3d
1465:   gsave
1466:   newpath
1467:     A rtp2xyz 3dto2d smoveto
1468:     A B arcspherique_
1469:     B C arcspherique_
1470:     C A arcspherique_
1471:   closepath
1472:   starfill
1473:   currentlinewidth 0 eq {} {stroke} ifelse
1474:   grestore
1475: end
1476: } def
1477:
1478: %%%% ### fin insertion ###
1479:

```

## 8.11 - Routines sur les tableaux

```

1480: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1481: %%%                                operations sur les tableaux                                %%%
1482: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1483:
1484: %%%% ### duparray ###
1485: /duparray {
1486: 1 dict begin
1487:   /table exch def
1488:   table
1489:   [ table aload pop ]
1490: end
1491: } def
1492:
1493: %%%% ### append ###
1494: %% syntaxe : string1 string2 append --> concatene les 2 chaines ou fusionne 2 tableaux
1495: /append {
1496: 3 dict begin
1497:   dup isarray {
1498:     /tab2 exch def
1499:     /tab1 exch def
1500:     [ tab1 aload pop tab2 aload pop ]
1501:   } {
1502:     /str2 exch def
1503:     /str1 exch def
1504:     /result str1 length str2 length add string def
1505:     str1 result copy pop
1506:     result str1 length str2 putinterval
1507:     result
1508:   } ifelse
1509: end
1510: } def
1511:
1512: %%%% ### rollparray ###
1513: %% syntaxe : array n rollparray -> array
1514: %% opere une rotation de n sur les couplets du tableau array

```

```

1515: /rollparray {
1516: 4 dict begin
1517:   /k exch def
1518:   /table exch def
1519:   /n table length def
1520:   k 0 eq {
1521:     table
1522:   } {
1523:     k 0 ge {
1524:       [ table aload pop 2 {n 1 roll} repeat ]
1525:       k 1 sub
1526:     } {
1527:       [ table aload pop 2 {n -1 roll} repeat ]
1528:       k 1 add
1529:     } ifelse
1530:     rollparray
1531:   } ifelse
1532: end
1533: } def
1534:
1535: %%%% ### bubblesort ###
1536: %% syntaxe : array bubblesort --> array2 trie par ordre croissant
1537: %% code de Bill Casselman
1538: %% http://www.math.ubc.ca/people/faculty/cass/graphics/text/www/
1539: /bubblesort {
1540: 4 dict begin
1541:   /a exch def
1542:   /n a length 1 sub def
1543:   n 0 gt {
1544:     % at this point only the n+1 items in the bottom of a remain to
1545:     % the sorted largest item in that blocks is to be moved up into
1546:     % position n
1547:     n {
1548:       0 1 n 1 sub {
1549:         /i exch def
1550:         a i get a i 1 add get gt {
1551:           % if a[i] > a[i+1] swap a[i] and a[i+1]
1552:           a i 1 add
1553:           a i get
1554:           a i a i 1 add get
1555:           % set new a[i] = old a[i+1]
1556:           put
1557:           % set new a[i+1] = old a[i]
1558:           put
1559:         } if
1560:       } for
1561:     } /n n 1 sub def
1562:   } repeat
1563: } if
1564: a
1565: end
1566: } def
1567:
1568: %% syntaxe : array1 doublebubblesort --> array2 array3, array3 est
1569: %% trie par ordre croissant et array2 correspond a la position des
1570: %% indices de depart, ie si array1 = [3 2 4 1], alors array2 = [3 1 0 2]
1571: %% code de Bill Casselman, modifie par jpv, 15/08/2006
1572: %% http://www.math.ubc.ca/people/faculty/cass/graphics/text/www/
1573: /doublebubblesort {
1574: 5 dict begin
1575:   /table exch def
1576:   /n table length 1 sub def
1577:   /indices [ 0 1 n {} for ] def
1578:   n 0 gt {
1579:     % at this point only the n+1 items in the bottom of a remain to
1580:     % the sorted largest item in that blocks is to be moved up into
1581:     % position n
1582:     n {
1583:       0 1 n 1 sub {
1584:         /i exch def
1585:         table i get table i 1 add get gt {
1586:           % if a[i] > a[i+1] swap a[i] and a[i+1]
1587:           table i 1 add
1588:           table i get
1589:           table i table i 1 add get
1590:           % set new a[i] = old a[i+1]

```

```

1591:          put
1592:          % set new a[i+1] = old a[i]
1593:          put
1594:
1595:          indices i 1 add
1596:          indices i get
1597:          indices i indices i 1 add get
1598:          % set new a[i] = old a[i+1]
1599:          put
1600:          % set new a[i+1] = old a[i]
1601:          put
1602:        } if
1603:      } for
1604:    /n n 1 sub def
1605:  } repeat
1606: } if
1607: indices table
1608: end
1609: } def
1610:
1611: %%%% ### quicksort ###
1612: % src : http://www.math.ubc.ca/~cass/graphics/text/www/code/sort.inc
1613: % code de Bill Casselman, modifie par jpv, 18/10/2007
1614:
1615: /qsortdict 8 dict def
1616:
1617: qsortdict begin
1618:
1619: % args: /comp a L R x
1620: % effect: effects a partition into two pieces [L j] [i R]
1621: %   leaves i j on stack
1622:
1623: /partition { 8 dict begin
1624: /x exch def
1625: /j exch def
1626: /i exch def
1627: /a exch def
1628: load /comp exch def
1629: {
1630:   {
1631:     a i get x comp exec not {
1632:       exit
1633:     } if
1634:     /i i 1 add def
1635:   } loop
1636:   {
1637:     x a j get comp exec not {
1638:       exit
1639:     } if
1640:     /j j 1 sub def
1641:   } loop
1642:
1643:   i j le {
1644:     % swap a[i] a[j]
1645:     a j a i get
1646:     a i a j get
1647:     put put
1648:     indices j indices i get
1649:     indices i indices j get
1650:     put put
1651:     /i i 1 add def
1652:     /j j 1 sub def
1653:   } if
1654:   i j gt {
1655:     exit
1656:   } if
1657: } loop
1658: i j
1659: end } def
1660:
1661: % args: /comp a L R
1662: % effect: sorts a[L .. R] according to comp
1663:
1664: /subsort {
1665: % /c a L R
1666: [ 3 1 roll ] 3 copy

```



```

1667: % /c a [L R] /c a [L R]
1668: aload aload pop
1669: % /c a [L R] /c a L R L R
1670: add 2 idiv
1671: % /c a [L R] /c a L R (L+R)/2
1672: 3 index exch get
1673: % /c a [L R] /c a L R x
1674: partition
1675: % /c a [L R] i j
1676: % if j > L subsort(a, L, j)
1677: dup
1678: % /c a [L R] i j j
1679: 3 index 0 get gt {
1680:   % /c a [L R] i j
1681:   5 copy
1682:   % /c a [L R] i j /c a [L R] i j
1683:   exch pop
1684:   % /c a [L R] i j /c a [L R] j
1685:   exch 0 get exch
1686:   % ... /c a L j
1687:   subsort
1688: } if
1689: % /c a [L R] i j
1690: pop dup
1691: % /c a [L R] i i
1692: % if i < R subsort(a, i, R)
1693: 2 index 1 get lt {
1694:   % /c a [L R] i
1695:   exch 1 get
1696:   % /c a i R
1697:   subsort
1698: }{
1699:   4 { pop } repeat
1700: } ifelse
1701: } def
1702:
1703: end
1704:
1705: % args: /comp a
1706: % effect: sorts the array a
1707: % comp returns truth of x < y for entries in a
1708:
1709: /quicksort { qsortdict begin
1710: dup length 1 gt {
1711: % /comp a
1712: dup
1713: % /comp a a
1714: length 1 sub
1715: % /comp a n-1
1716: 0 exch subsort
1717: } {
1718: pop pop
1719: } ifelse
1720: end } def
1721:
1722: % -----
1723:
1724: %% fin du code de Bill Casselman
1725:
1726: %% syntaxe : array1 doublebubblesort --> array2 array3, array3 est
1727: %% trie par ordre croissant et array2 correspond a la position des
1728: %% indices de depart, ie si array1 = [3 2 4 1], alors array2 = [3 1 0 2]
1729: %% code de Bill Casselman, modifie par jpv, 18/10/2007
1730: %% http://www.math.ubc.ca/people/faculty/cass/graphics/text/www/
1731: /doublequicksort {
1732: qsortdict begin
1733:   /comp exch
1734:   /a exch def
1735:   a dup length /n exch def
1736:   /indices [0 1 n 1 sub {} for ] def
1737:   dup length 1 gt {
1738:     % /comp a
1739:     dup
1740:     % /comp a a
1741:     length 1 sub
1742:     % /comp a n-1

```

```

1743:      0 exch subsort
1744:    } {
1745:      pop pop
1746:    } ifelse
1747:    indices a
1748:  end
1749: } def
1750:
1751: /comp {lt} def
1752:
1753: %%%% ### apply ###
1754: %% syntaxe : [x1 ... xn] (f) apply --> [f(x1) ... f(xn)]
1755: /apply {
1756: 3 dict begin
1757:   dup isstring
1758:   {/fonction exch cvx def}
1759:   {/fonction exch def}
1760:   ifelse
1761:   /liste exch def
1762:   /@i 0 def
1763:   [
1764:   liste length {
1765:     liste @i get fonction
1766:     /@i @i 1 add store
1767:   } repeat
1768:   counttomark
1769:   0 eq
1770:   {pop}
1771:   {1}
1772:   ifelse
1773: end
1774: } def
1775:
1776: %% syntaxe : [x1 ... xn] (f) papply
1777: /papply {
1778: 3 dict begin
1779:   dup isstring
1780:   {/fonction exch cvx def}
1781:   {/fonction exch def}
1782:   ifelse
1783:   /liste exch def
1784:   /@i 0 def
1785:   [
1786:   liste length 2 idiv {
1787:     liste @i get
1788:     liste @i 1 add get
1789:     fonction
1790:     /@i @i 2 add store
1791:   } repeat
1792:   counttomark
1793:   0 eq
1794:   {pop}
1795:   {1}
1796:   ifelse
1797: end
1798: } def
1799:
1800: %% syntaxe : [x1 ... xn] (f) capply
1801: /capply {
1802: 3 dict begin
1803:   dup isstring
1804:   {/fonction exch cvx def}
1805:   {/fonction exch def}
1806:   ifelse
1807:   /liste exch def
1808:   /@i 0 def
1809:   [
1810:   liste length 3 idiv {
1811:     liste @i get
1812:     liste @i 1 add get
1813:     liste @i 2 add get
1814:     fonction
1815:     /@i @i 3 add store
1816:   } repeat
1817:   counttomark
1818:   0 eq

```

```

1819:     {pop}
1820:     {1}
1821:   ifelse
1822: end
1823: } def
1824:
1825: %%%% ### reverse ###
1826: %% syntaxe : array reverse --> inverse l ordre des items dans
1827: %% le tableau
1828: /reverse {
1829: 3 dict begin
1830:   /le_tableau exch def
1831:   /n le_tableau length def
1832:   /i n 1 sub def
1833:   [
1834:     n {
1835:       le_tableau i get
1836:       /i i 1 sub store
1837:     } repeat
1838:   ]
1839: end
1840: } def
1841:
1842: %% syntaxe : array_points reversep --> inverse l ordre des points dans
1843: %% le tableau
1844: /reversep {
1845: 3 dict begin
1846:   /le_tableau exch def
1847:   /n le_tableau length 2 idiv def
1848:   /i n 1 sub def
1849:   [
1850:     n {
1851:       le_tableau i getp
1852:       /i i 1 sub store
1853:     } repeat
1854:   ]
1855: end
1856: } def
1857:
1858: %%%% ### get ###
1859: %% syntaxe : array_points n getp --> le n-ieme point du tableau de
1860: %% points array_points
1861: /getp {
1862:   2 copy
1863:   2 mul get
1864:   3 1 roll
1865:   2 mul 1 add get
1866: } def
1867:
1868: %%%% ### fin insertion ###
1869:

```

## 8.12 - Matrices

```

1870: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1871: %%%%                               matrices                               %%%%
1872: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1873:
1874: %%%% ### linear ###
1875: %% syntaxe : M i j any --> depose any dans M en a_ij
1876: /put_ij {
1877: 5 dict begin
1878:   /a exch def
1879:   /j exch def
1880:   /i exch def
1881:   /M exch def
1882:   /L M i get_Li def
1883:   L j a put
1884:   M i L put_Li
1885: end
1886: } def

```

```

1887:
1888: %% syntaxe : M i j get_ij --> le coeff c_ij
1889: /get_ij {
1890:   3 1 roll   %% j M i
1891:   get_Li    %% j L_i
1892:   exch get
1893: } def
1894:
1895: %% syntaxe : M i L put_Li --> remplace dans M la ligne Li par L
1896: /put_Li {
1897:   put
1898: } def
1899:
1900: %% syntaxe : M i get_Li --> la ligne Li de M
1901: /get_Li {
1902:   get
1903: } def
1904:
1905: %%%% ### fin insertion ###

```

### 8.13 - Routines pour le calcul 3d

```

1906:
1907: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1908: %%%%          geometrie 3d (calculs)          %%%%
1909: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1910:
1911: %%%% ### dupp3d ###
1912: %% duplique le vecteur 3d
1913: /dupp3d { %% x y z
1914:   3 copy
1915: } def
1916: /dupv3d {dupp3d} def
1917:
1918: %%%% ### angle3d ###
1919: %% syntaxe : vect1 vect2 angle3d
1920: /angle3d {
1921: 4 dict begin
1922:   normalize3d /vect2 defpoint3d
1923:   normalize3d /vect1 defpoint3d
1924:   /cosalpha vect1 vect2 scalprod3d def
1925:   /sinalpha vect1 vect2 vectprod3d norme3d def
1926:   sinalpha cosalpha atan
1927: end
1928: } def
1929:
1930: %%%% ### transformpoint3d ###
1931: %% syntaxe : x y z a11 a21 a31 a12 a22 a32 a13 a23 a33
1932: %%   transformpoint3d -> X Y Z
1933: /transformpoint3d {
1934: 12 dict begin
1935:   /a33 exch def
1936:   /a23 exch def
1937:   /a13 exch def
1938:   /a32 exch def
1939:   /a22 exch def
1940:   /a12 exch def
1941:   /a31 exch def
1942:   /a21 exch def
1943:   /a11 exch def
1944:   /z   exch def
1945:   /y   exch def
1946:   /x   exch def
1947:   a11 x mul a12 y mul add a13 z mul add
1948:   a21 x mul a22 y mul add a23 z mul add
1949:   a31 x mul a32 y mul add a33 z mul add
1950: end
1951: } def
1952:
1953: %%%% ### normalize3d ###
1954: %% rend le vecteur 3d unitaire. Ne fait rien si u=0

```

```

1955: /unitaire3d { %% x y z
1956: 2 dict begin
1957:   /u defpoint3d
1958:   /norme u norme3d def
1959:   norme 0 eq
1960:   {u}
1961:   {u 1 norme div mulv3d
1962:   } ifelse
1963: end
1964: } def
1965: /normalize3d {unitaire3d} def
1966:
1967: %%%% ### mulv ###
1968: %% syntaxe : u a mulv --> au
1969: /mulv { %% xA, yA, a
1970:   dup %% xA, yA, a, a
1971:   3 1 roll %% xA, a, yA, a
1972:   mul 3 1 roll %% ayA, xA, a
1973:   mul exch
1974: } def
1975:
1976: %%%% ### geom3d ###
1977: %% syntaxe : A k1 B k2 barycentre3d -> G, barycentre du systeme
1978: %% [(A, k1) (B, k2)]
1979: /barycentre3d {
1980: 4 dict begin
1981:   /k2 exch def
1982:   /B defpoint3d
1983:   /k1 exch def
1984:   /A defpoint3d
1985:   A k1 mulv3d
1986:   B k2 mulv3d
1987:   addv3d
1988:   1 k1 k2 add div mulv3d
1989: end
1990: } def
1991:
1992: %% syntaxe : array isobarycentre3d --> G
1993: /isobarycentre3d {
1994: 2 dict begin
1995:   /table exch def
1996:   /n table length 3 idiv def
1997:   table 0 getp3d
1998:   1 1 n 1 sub {
1999:     table exch getp3d
2000:     addv3d
2001:   } for
2002:   1 n div mulv3d
2003: end
2004: } def
2005:
2006: %% syntaxe : M A alpha hompoint3d -> le point M' tel que AM' = alpha AM
2007: /hompoint3d {
2008: 3 dict begin
2009:   /alpha exch def
2010:   /A defpoint3d
2011:   /M defpoint3d
2012:   A M vecteur3d alpha mulv3d A addv3d
2013: end
2014: } def
2015:
2016: %% syntaxe : M A sympoint3d -> le point M' tel que AM' = -AM
2017: /sympoint3d {
2018: 2 dict begin
2019:   /A defpoint3d
2020:   /M defpoint3d
2021:   A M vecteur3d -1 mulv3d A addv3d
2022: end
2023: } def
2024:
2025: %% syntaxe : A u translatepoint3d --> B image de A par la translation de vecteur u
2026: /translatepoint3d {
2027:   addv3d
2028: } def
2029:
2030: /scaleOpoint3d {

```

```

2031: 6 dict begin
2032:   /k3 exch def
2033:   /k2 exch def
2034:   /k1 exch def
2035:   /z exch def
2036:   /y exch def
2037:   /x exch def
2038:   k1 x mul
2039:   k2 y mul
2040:   k3 z mul
2041: end
2042: } def
2043:
2044: % syntaxe : M alpha_x alpha_y alpha_z rotateOpoint3d --> M'
2045: /rotateOpoint3d {
2046: 21 dict begin
2047:   /RotZ exch def
2048:   /RotY exch def
2049:   /RotX exch def
2050:   /Zpoint exch def
2051:   /Ypoint exch def
2052:   /Xpoint exch def
2053:   /c1 {RotX cos} bind def
2054:   /c2 {RotY cos} bind def
2055:   /c3 {RotZ cos} bind def
2056:   /s1 {RotX sin} bind def
2057:   /s2 {RotY sin} bind def
2058:   /s3 {RotZ sin} bind def
2059:   /M11 {c2 c3 mul} bind def
2060:   /M12 {c3 s1 mul s2 mul c1 s3 mul sub} bind def
2061:   /M13 {c1 c3 mul s2 mul s1 s3 mul add} bind def
2062:   /M21 {c2 s3 mul} bind def
2063:   /M22 {s1 s2 mul s3 mul c1 c3 mul add} bind def
2064:   /M23 {s3 s2 mul c1 mul c3 s1 mul sub} bind def
2065:   /M31 {s2 neg} bind def
2066:   /M32 {s1 c2 mul} bind def
2067:   /M33 {c1 c2 mul} bind def
2068:   M11 Xpoint mul M12 Ypoint mul add M13 Zpoint mul add
2069:   M21 Xpoint mul M22 Ypoint mul add M23 Zpoint mul add
2070:   M31 Xpoint mul M32 Ypoint mul add M33 Zpoint mul add
2071: end
2072: } def
2073:
2074: %%%% ### vecteur3d ###
2075: %% creation du vecteur AB a partir de A et B
2076: /vecteur3d { %% xA yA zA xB yB zB
2077: 6 dict begin
2078:   /zB exch def
2079:   /yB exch def
2080:   /xB exch def
2081:   /zA exch def
2082:   /yA exch def
2083:   /xA exch def
2084:   xB xA sub
2085:   yB yA sub
2086:   zB zA sub
2087: end
2088: }def
2089:
2090: %%%% ### vectprod3d ###
2091: %% produit vectoriel de deux vecteurs 3d
2092: /vectprod3d { %% x1 y1 z1 x2 y2 z2
2093: 6 dict begin
2094:   /zp exch def
2095:   /yp exch def
2096:   /xp exch def
2097:   /z exch def
2098:   /y exch def
2099:   /x exch def
2100:   y zp mul z yp mul sub
2101:   z xp mul x zp mul sub
2102:   x yp mul y xp mul sub
2103: end
2104: } def
2105:
2106: %%%% ### scalprod3d ###

```

```

2107: %% produit scalaire de deux vecteurs 3d
2108: /scalprod3d { %% x1 y1 z1 x2 y2 z2
2109: 6 dict begin
2110:   /zp exch def
2111:   /yp exch def
2112:   /xp exch def
2113:   /z exch def
2114:   /y exch def
2115:   /x exch def
2116:   x xp mul y yp mul add z zp mul add
2117: end
2118: } def
2119:
2120: %%%% ### papply3d ###
2121: %% syntaxe : [A1 ... An] (f) papply3d --> [f(A1) ... f(An)]
2122: /papply3d {
2123: 3 dict begin
2124:   /fonction exch def
2125:   /liste exch def
2126:   /i 0 def
2127:   [
2128:     liste length 3 idiv {
2129:       liste i get
2130:       liste i 1 add get
2131:       liste i 2 add get
2132:       fonction
2133:       /i i 3 add store
2134:     } repeat
2135:     counttomark
2136:     0 eq
2137:     {pop}
2138:     {1}
2139:     ifelse
2140:   end
2141: } def
2142:
2143: %%%% ### defpoint3d ###
2144: %% creation du point A a partir de xA yA yB et du nom /A
2145: /defpoint3d { %% xA yA zA /nom
2146: 1 dict begin
2147:   /memo exch def
2148:   [ 4 1 roll ] cvx memo exch
2149: end def
2150: }def
2151:
2152: %%%% ### distance3d ###
2153: /distance3d { %% A B
2154:   vecteur3d norme3d
2155: } def
2156:
2157: %%%% ### get3d ###
2158: /getp3d { %% [tableau de points 3d] i --> donne le ieme point du tableau
2159:   2 copy 2 copy
2160:   3 mul get
2161:   5 1 roll
2162:   3 mul 1 add get
2163:   3 1 roll
2164:   3 mul 2 add get
2165: } def
2166:
2167: %%%% ### norme3d ###
2168: %% norme d un vecteur 3d
2169: /norme3d { %% x y z
2170: 3 dict begin
2171:   /z exch def
2172:   /y exch def
2173:   /x exch def
2174:   x dup mul y dup mul add z dup mul add sqrt
2175: end
2176: } def
2177:
2178: %%%% ### mulv3d ###
2179: %% (scalaire)*(vecteur 3d) Attention : dans l autre sens !
2180: /mulv3d { %% x y z lambda
2181: 4 dict begin
2182:   /lambda exch def

```

```

2183: /z exch def
2184: /y exch def
2185: /x exch def
2186: x lambda mul
2187: y lambda mul
2188: z lambda mul
2189: end
2190: } def
2191:
2192: %%%% ### adv3d ###
2193: %% addition de deux vecteurs 3d
2194: /adv3d { %% x1 y1 z1 x2 y2 z2
2195: 6 dict begin
2196: /zp exch def
2197: /yp exch def
2198: /xp exch def
2199: /z exch def
2200: /y exch def
2201: /x exch def
2202: x xp add
2203: y yp add
2204: z zp add
2205: end
2206: } def
2207:
2208: %%%% ### milieu3d ###
2209: /milieu3d { %% A B --> I le milieu de [AB]
2210: adv3d 0.5 mulv3d
2211: } def
2212:
2213: %%%% ### fin insertion ###
2214:

```

## 8.14 - Routines pour le dessin 3d

```

2215: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2216: %%% geometrie 3d (dessins) %%%
2217: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2218:
2219: %%%% ### point3d ###
2220: /point3d { %% A
2221: 3dto2d point
2222: } def
2223:
2224: /points3d { %% tableau de points3d
2225: tab3dto2d points
2226: } def
2227:
2228: %%%% ### ligne3d ###
2229: %% [tableau de points3d] option --> trace la ligne brisee
2230: /ligne3d {
2231: 1 dict begin
2232: dup isstring
2233: {/option exch def}
2234: if
2235: tab3dto2d
2236: currentdict /option known
2237: {option}
2238: if
2239: ligne
2240: end
2241: } def
2242:
2243: %% [tableau de points3d] option --> trace la ligne brisee
2244: /ligne3d_ {
2245: 1 dict begin
2246: dup isstring
2247: {/option exch def}
2248: if
2249: tab3dto2d
2250: currentdict /option known

```



```

2251:      {option}
2252:      if
2253:      ligne_
2254: end
2255: } def
2256:
2257: %%%% ### tab3dto2d ###
2258: %% transforme un tableau de points 3d en tableau de points 2d
2259: /tab3dto2d {
2260: 2 dict begin
2261:   /T exch def
2262:   /n T length def
2263:   [ T aload pop
2264:   n 1 sub -1 n 3 idiv 2 mul
2265:   { 1 dict begin
2266:     /i exch def
2267:     3dto2d i 2 roll
2268:     end } for ]
2269: end
2270: } def
2271:
2272: %%%% ### polygone3d ###
2273: /polygone3d { %% tableau de points3d
2274:   tab3dto2d polygone
2275: } def
2276:
2277: /polygone3d* { %% tableau de points3d
2278:   tab3dto2d polygone*
2279: } def
2280:
2281: %%%% ### fin insertion ###
2282:

```

## 8.15 - Gestion des chemins définis par des chaînes de caractères

```

2283: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2284: %%%                                gestion du texte                                %%%
2285: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2286:
2287: %%%% ### pathtext ###
2288: %% syntaxe : string x y initp@thtext
2289: /initp@thtext {
2290: 7 dict begin
2291:   /y exch def
2292:   /x exch def
2293:   /str exch def
2294:   str 0 0 show_dim
2295:   /wy exch def
2296:   /wx exch def
2297:   /lly exch def
2298:   /llx exch def
2299:   pop pop pop
2300:   newpath
2301:   x y smoveto
2302: } def
2303: /closep@thtext {
2304:   str true charpath
2305: end
2306: } def
2307:
2308: %% syntaxe : string x y cctext_
2309: /cctext_ {
2310:   initp@thtext
2311:   llx wx add lly wy add -.5 mulv rmoveto
2312:   closep@thtext
2313: } def
2314:
2315: /brtext_ {
2316:   initp@thtext
2317:   hadjust 0 rmoveto
2318:   llx neg 0 rmoveto

```

```

2319:   closep@thtext
2320: } def
2321:
2322: /bbtext_ {
2323:   initp@thtext
2324:   0 0 rmoveto
2325:   0 0 rmoveto
2326:   closep@thtext
2327: } def
2328:
2329: /bltext_ {
2330:   initp@thtext
2331:   hadjust neg 0 rmoveto
2332:   wx neg 0 rmoveto
2333:   closep@thtext
2334: } def
2335:
2336: /bctext_ {
2337:   initp@thtext
2338:   0 0 rmoveto
2339:   wx llx add -.5 mul 0 rmoveto
2340:   closep@thtext
2341: } def
2342:
2343: /ubtext_ {
2344:   initp@thtext
2345:   0 vadjust rmoveto
2346:   0 lly neg rmoveto
2347:   closep@thtext
2348: } def
2349:
2350: /urtext_ {
2351:   initp@thtext
2352:   hadjust vadjust rmoveto
2353:   llx neg lly neg rmoveto
2354:   closep@thtext
2355: } def
2356:
2357: /ultext_ {
2358:   initp@thtext
2359:   hadjust neg vadjust rmoveto
2360:   wx neg lly neg rmoveto
2361:   closep@thtext
2362: } def
2363:
2364: /uctext_ {
2365:   initp@thtext
2366:   0 vadjust rmoveto
2367:   llx wx add -.5 mul lly neg rmoveto
2368:   closep@thtext
2369: } def
2370:
2371: /drtext_ {
2372:   initp@thtext
2373:   hadjust vadjust neg rmoveto
2374:   llx neg wy neg rmoveto
2375:   closep@thtext
2376: } def
2377:
2378: /dbtext_ {
2379:   initp@thtext
2380:   0 vadjust neg rmoveto
2381:   0 wy neg rmoveto
2382:   closep@thtext
2383: } def
2384:
2385: /dltext_ {
2386:   initp@thtext
2387:   hadjust neg vadjust neg rmoveto
2388:   wx neg wy neg rmoveto
2389:   closep@thtext
2390: } def
2391:
2392: /dctext_ {
2393:   initp@thtext
2394:   0 vadjust neg rmoveto

```

```

2395:    llx wx add -2 div wy neg rmoveto
2396:    closep@thtext
2397: } def
2398:
2399: /crtext_ {
2400:    initp@thtext
2401:    hadjust 0 rmoveto
2402:    llx neg lly wy add -2 div rmoveto
2403:    closep@thtext
2404: } def
2405:
2406: /cbtext_ {
2407:    initp@thtext
2408:    0 0 rmoveto
2409:    0 lly wy add -2 div rmoveto
2410:    closep@thtext
2411: } def
2412:
2413: /cltext_ {
2414:    initp@thtext
2415:    hadjust neg 0 rmoveto
2416:    wx neg lly wy add -2 div rmoveto
2417:    closep@thtext
2418: } def
2419:
2420: /cctext_ {
2421:    initp@thtext
2422:    0 0 rmoveto
2423:    llx wx add lly wy add -.5 mulv rmoveto
2424:    closep@thtext
2425: } def
2426:
2427: %%%% ### fin insertion ###
2428:

```

## 8.16 - Routines pour le calcul sur le type solid

```

2429: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2430: %%%%                               bibliotheque sur les solides                               %%%%
2431: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2432:
2433: %%%% ### solide ###
2434: %% solid = [Sommets Faces Colors_Faces InOut_Table]
2435: /solidgetpointstable {
2436:    0 get
2437: } def
2438:
2439: /solidgetfaces {
2440:    1 get
2441: } def
2442:
2443: /solidgetface {
2444: 1 dict begin
2445:    /i exch def
2446:    solidgetfaces i get
2447: end
2448: } def
2449:
2450: /solidgetfcolors {
2451:    2 get
2452: } def
2453:
2454: %% syntaxe : solid i solidgetfcolor --> str
2455: /solidgetfcolor {
2456: 1 dict begin
2457:    /i exch def
2458:    solidgetfcolors i get
2459: end
2460: } def
2461:
2462: %% syntaxe : solid i str solidputfcolor --> -

```

```

2463: /solidputfcolor {
2464: 2 dict begin
2465:   /str exch def
2466:   /i exch def
2467:   solidgetfcolors i str put
2468: end
2469: } def
2470:
2471: /solidgetinouttable {
2472:   3 get
2473: } def
2474:
2475: /solidputpointstable {
2476:   0 exch put
2477: } def
2478:
2479: /solidputfaces {
2480:   1 exch put
2481: } def
2482:
2483: /solidputfcolors {
2484:   2 exch put
2485: } def
2486:
2487: /solidputinouttable {
2488:   3 exch put
2489: } def
2490:
2491: %% syntaxe : any issolid --> booleen, vrai si any est de type solid
2492: /issolid {
2493: 1 dict begin
2494:   /candidat exch def
2495:   candidat isarray {
2496:     candidat length 4 eq {
2497:       candidat 0 get isarray
2498:       candidat 1 get isarray and
2499:       candidat 2 get isarray and
2500:       candidat 3 get isarray and
2501:     } {
2502:       false
2503:     } ifelse
2504:   } {
2505:     false
2506:   } ifelse
2507: end
2508: } def
2509:
2510: /dupsolid {
2511: 5 dict begin
2512:   /solid exch def
2513:   /S solid solidgetpointstable def
2514:   /F solid solidgetfaces def
2515:   /FC solid solidgetfcolors def
2516:   /IO solid solidgetinouttable def
2517:   solid
2518:   [
2519:     S duparray exch pop
2520:     F duparray exch pop
2521:     FC duparray exch pop
2522:     IO duparray exch pop
2523:   ]
2524: end
2525: } def
2526:
2527: %% syntaxe : solid array solidputinfaces --> -
2528: /solidputinfaces {
2529: 4 dict begin
2530:   /facesinternes exch def
2531:   /solid exch def
2532:   /n2 facesinternes length def
2533:   /IO solid solidgetinouttable def
2534:   /facesexternes solid solidgetoutfaces def
2535:   /n1 facesexternes length def
2536:   solid
2537:   [facesexternes aload pop facesinternes aload pop]
2538:   solidputfaces

```

```

2539:      IO 0 0 put
2540:      IO 1 n1 1 sub put
2541:      IO 2 n1 put
2542:      IO 3 n1 n2 add 1 sub put
2543: end
2544: } def
2545:
2546: %% syntaxe : solid array solidputoutfaces --> -
2547: /solidputoutfaces {
2548: 4 dict begin
2549:   /facesexternes exch def
2550:   /solid exch def
2551:   /n1 facesexternes length def
2552:   /IO solid solidgetinouttable def
2553:   /facesinternes solid solidgetinfaces def
2554:   /n2 facesinternes length def
2555:   solid
2556:   [facesexternes aload pop facesinternes aload pop]
2557:   solidputfaces
2558:   IO 0 0 put
2559:   IO 1 n1 1 sub put
2560:   IO 2 n1 put
2561:   IO 3 n1 n2 add 1 sub put
2562: end
2563: } def
2564:
2565: %% syntaxe : solid array solidputoutfaces --> -
2566: /solidputoutfaces {
2567: 4 dict begin
2568:   /facesexternes exch def
2569:   /solid exch def
2570:   /n1 facesexternes length def
2571:   /IO solid solidgetinouttable def
2572:   /facesinternes solid solidgetinfaces def
2573:   /n2 facesinternes length def
2574:   solid
2575:   [facesexternes aload pop facesinternes aload pop]
2576:   solidputfaces
2577:   IO 0 0 put
2578:   IO 1 n1 1 sub put
2579:   IO 2 n1 put
2580:   IO 3 n1 n2 add 1 sub put
2581: end
2582: } def
2583:
2584: /solidnombreinfaces {
2585: 1 dict begin
2586:   /solid exch def
2587:   solid solidwithinfaces {
2588:     /IO solid solidgetinouttable def
2589:     IO 3 get IO 2 get sub 1 add
2590:   } {
2591:     0
2592:   } ifelse
2593: end
2594: } def
2595:
2596: /solidnombreoutfaces {
2597: 1 dict begin
2598:   /solid exch def
2599:   /IO solid solidgetinouttable def
2600:   IO 1 get IO 0 get sub 1 add
2601: end
2602: } def
2603:
2604: %% syntaxe : solid solidgetinfaces --> array
2605: /solidgetinfaces {
2606: 4 dict begin
2607:   /solid exch def
2608:   solid issolid not {
2609:     (Error : mauvais type d argument dans solidgetinfaces) ==
2610:     quit
2611:   } if
2612:   solid solidwithinfaces {
2613:     /IO solid solidgetinouttable def
2614:     /F solid solidgetfaces def

```

```

2615:      /n1 IO 2 get def
2616:      /n2 IO 3 get def
2617:      /n n2 n1 sub 1 add def
2618:      F n1 n getinterval
2619:    } {
2620:      []
2621:    } ifelse
2622:  end
2623: } def
2624:
2625: %% syntaxe : solid solidgetoutfaces --> array
2626: /solidgetoutfaces {
2627: 4 dict begin
2628:   /solid exch def
2629:   solid issolid not {
2630:     (Error : mauvais type d argument dans solidgetoutfaces) ==
2631:     quit
2632:   } if
2633:   /IO solid solidgetinouttable def
2634:   /F solid solidgetfaces def
2635:   /n1 IO 0 get def
2636:   /n2 IO 1 get def
2637:   /n n2 n1 sub 1 add def
2638:   F n1 n getinterval
2639: end
2640: } def
2641:
2642: %%%% ### fin insertion ###
2643:
2644: %% /tracelignedeniveau? false def
2645: %% /hauteurlignedeniveau 1 def
2646: %% /couleurlignedeniveau {rouge} def
2647: %% /linewidthlignedeniveau 4 def
2648: %%
2649: %% /solidgrid true def
2650: %% /aretescachees true def
2651: %% /defaultsolidmode 2 def
2652:
2653: %%%% ### newsolid ###
2654: %% syntaxe : newsolid --> depose le solide nul sur la pile
2655: /newsolid {
2656:   [] [] generesolid
2657: } def
2658:
2659: %%%% ### generesolid ###
2660: /generesolid {
2661: 2 dict begin
2662:   /F exch def
2663:   /S exch def
2664:   [S F [F length {}] repeat] [0 F length 1 sub -1 -1]]
2665: end
2666: } def
2667:
2668: %%%% ### nullsolid ###
2669: %% syntaxe : solide nullsolid -> booleen, vrai si le solide est nul
2670: /nullsolid {
2671: 1 dict begin
2672:   /candidat exch def
2673:   candidat issolid not {
2674:     (Error type argument dans "nullsolid") ==
2675:     quit
2676:   } if
2677:   candidat solidgetpointstable length 0 eq {
2678:     true
2679:   } {
2680:     false
2681:   } ifelse
2682: end
2683: } def
2684:
2685: %%%% ### solidnombreoutfaces ###
2686: /solidnombreoutfaces {
2687: 4 dict begin
2688:   /solid exch def
2689:   solid issolid not {
2690:     (Error : mauvais type d argument dans solidnombreoutfaces) ==

```

```

2691:     quit
2692:   } if
2693:   solid nullsolid {
2694:     0
2695:   } {
2696:     /IO solid solidgetinouttable def
2697:     IO 1 get
2698:     IO 0 get sub
2699:     1 add
2700:   } ifelse
2701: end
2702: } def
2703:
2704: %%%% ### solidnombreinfaces ###
2705: /solidnombreinfaces {
2706: 4 dict begin
2707:   /solid exch def
2708:   solid issolid not {
2709:     (Error : mauvais type d argument dans solidnombreinfaces) ==
2710:     quit
2711:   } if
2712:   solid solidwithinfaces {
2713:     /IO solid solidgetinouttable def
2714:     IO 3 get
2715:     IO 2 get sub
2716:     1 add
2717:   } {
2718:     0
2719:   } ifelse
2720: end
2721: } def
2722:
2723: %%%% ### solidtests ###
2724: %% syntaxe : solid solidwithinfaces --> bool, true si le solide est vide
2725: /solidwithinfaces {
2726: 2 dict begin
2727:   /solid exch def
2728:   solid issolid not {
2729:     (Error : mauvais type d argument dans solidwithinfaces) ==
2730:     quit
2731:   } if
2732:   /table solid solidgetinouttable def
2733:   table 2 get -1 ne {
2734:     true
2735:   } {
2736:     false
2737:   } ifelse
2738: end
2739: } def
2740:
2741: %%%% ### solidgetsommet ###
2742: %% syntaxe : solid i j solidgetsommetface --> sommet i de la face j
2743: /solidgetsommetface {
2744: 6 dict begin
2745:   /j exch def
2746:   /i exch def
2747:   /solid exch def
2748:   solid issolid not {
2749:     (Error : mauvais type d argument dans solidgetsommetface) ==
2750:     quit
2751:   } if
2752:   /table_faces solid solidgetfaces def
2753:   /table_sommets solid solidgetpointstable def
2754:   /k table_faces j get i get def
2755:   table_sommets k getp3d
2756: end
2757: } def
2758:
2759: %% syntaxe : solid i solidgetsommetsface --> array, tableau des
2760: %% sommets de la face i du solide
2761: /solidgetsommetsface {
2762: 6 dict begin
2763:   /i exch def
2764:   /solid exch def
2765:   solid issolid not {
2766:     (Error : mauvais type d argument dans solidgetsommetsface) ==

```

```

2767:     quit
2768:   } if
2769:   /table_faces solid solidgetfaces def
2770:   /table_sommets solid solidgetpointstable def
2771:   /table_indices table_faces i get def
2772:   [
2773:     0 1 table_indices length 1 sub {
2774:       /j exch def
2775:       table_sommets table_indices j get getp3d
2776:     } for
2777:   ]
2778: end
2779: } def
2780:
2781: %% syntaxe : solid i solidgetsommet --> sommet i du solide
2782: /solidgetsommet {
2783: 3 dict begin
2784:   /i exch def
2785:   /solid exch def
2786:   solid issolid not {
2787:     (Error : mauvais type d argument dans solidgetsommet) ==
2788:     quit
2789:   } if
2790:   /table_sommets solid solidgetpointstable def
2791:   table_sommets i getp3d
2792: end
2793: } def
2794:
2795: %%%% ### solidcentreface ###
2796: %% syntaxe : solid i solidcentreface --> M
2797: /solidcentreface {
2798:   solidgetsommetsface isobarycentre3d
2799: } def
2800:
2801: %%%% ### solidnombre ###
2802: /solidnombresommets {
2803:   solidgetpointstable length 3 idiv
2804: } def
2805:
2806: /solidfacenombresommets {
2807:   solidgetface length
2808: } def
2809:
2810: /solidnombrefaces {
2811:   solidgetfaces length
2812: } def
2813:
2814: %%%% ### solidshowsommets ###
2815: /solidshowsommets {
2816: 8 dict begin
2817:   dup issolid not {
2818:     %% on a un argument
2819:     /option exch def
2820:   } if
2821:   /sol exch def
2822:   /n sol solidnombresommets def
2823:   /m sol solidnombrefaces def
2824:   currentdict /option known not {
2825:     /option [0 1 n 1 sub {} for] def
2826:   } if
2827:   0 1 option length 1 sub {
2828:     /k exch def
2829:     option k get /i exch def      %% indice du sommet examine
2830:     sol i solidgetsommet point3d
2831:   } for
2832: end
2833: } def
2834:
2835: %%%% ### solidnumssommets ###
2836: /solidnumssommets {
2837: 8 dict begin
2838:   Font findfont 10 scalefont setfont
2839:   dup issolid not {
2840:     %% on a un argument
2841:     /option exch def
2842:   } if

```



```

2843: /sol exch def
2844: /n sol solidnombresommets def
2845: /m sol solidnombrefaces def
2846: currentdict /option known not {
2847: /option [0 1 n 1 sub {}] for] def
2848: } if
2849: /result [
2850: n {false} repeat
2851: ] def
2852: 0 1 option length 1 sub {
2853: /k exch def
2854: option k get /i exch def %% indice du sommet examine
2855: 0 1 m 1 sub {
2856: /j exch def %% indice de la face examinee
2857: i sol j solidgetface in exch pop {
2858: %% le sommet i est dans la face j
2859: exit
2860: } if
2861: } for
2862: %% le sommet i est dans la face j
2863: sol j solidcentreface /G defpoint3d
2864: sol i solidgetsommet /S defpoint3d
2865: i ( ) cvs
2866: G S vecteur3d normalize3d
2867: 15 dup ptobjpoint pop
2868: mulv3d
2869: S addv3d
2870: 3dto2d cctext
2871: } for
2872: end
2873: } def
2874:
2875: %%%% ### gestionsolidmode ###
2876: %% table = [ [vars] [mode0] [mode1] [mode2] [mode3] [mode4] ]
2877: /gestionsolidmode {
2878: 5 dict begin
2879: /table exch def
2880: dup xcheck {
2881: /mode exch def
2882: } {
2883: dup isarray {
2884: /tableaffectation exch def
2885: /mode -1 def
2886: } {
2887: /mode defaultsolidmode def
2888: } ifelse
2889: } ifelse
2890: /vars table 0 get def
2891: /nbvars vars length def
2892: mode 0 ge {
2893: /tableaffectation table mode 1 add 5 min get def
2894: } if
2895: 0 1 nbvars 1 sub {
2896: /i exch def
2897: vars i get
2898: tableaffectation i get
2899: } for
2900: nbvars
2901: end
2902: {def} repeat
2903: } def
2904:
2905: %%%% ### solidfuz ###
2906: %% syntaxe : solid1 solid2 solidfuz -> solid
2907: /solidfuz {
2908: 5 dict begin
2909: /solid2 exch def
2910: /solid1 exch def
2911: /S1 solid1 solidgetpointstable def
2912: /S2 solid2 solidgetpointstable def
2913: /n S1 length 3 idiv def
2914:
2915: %% les sommets
2916: /S S1 S2 append def
2917:
2918: %% les faces internes et leurs couleurs

```

```

2919: /FI1 solid1 solidgetinfaces def
2920: /FIC1 solid1 solidgetincolors def
2921: solid2 solidnombreinfaces 0 eq {
2922: /FI2 [] def
2923: /FIC2 [] def
2924: } {
2925: /FI2 solid2 solidgetinfaces {{n add} apply} apply def
2926: /FIC2 solid2 solidgetincolors def
2927: } ifelse
2928: /FI [FI1 aload pop FI2 aload pop] def
2929: /FIC [FIC1 aload pop FIC2 aload pop] def
2930:
2931: %% les faces externes et leurs couleurs
2932: /FO1 solid1 solidgetoutfaces def
2933: /FOC1 solid1 solidgetoutcolors def
2934: /FO2 solid2 solidgetoutfaces {{n add} apply} apply def
2935: /FOC2 solid2 solidgetoutcolors def
2936: /FO [FO1 aload pop FO2 aload pop] def
2937: /FOC [FOC1 aload pop FOC2 aload pop] def
2938:
2939: /F [FO aload pop FI aload pop] def
2940: /FC [FOC aload pop FIC aload pop] def
2941: /IO [0 FO length 1 sub dup 1 add dup FI length add 1 sub] def
2942:
2943: S F generesolid
2944: dup FC solidputfcolors
2945: dup IO solidputinouttable
2946: end
2947: } def
2948:
2949: %%%% ### solidnormaleface ###
2950: %% syntaxe : solid i solidnormaleface --> u, vecteur normale a la
2951: %% face d indice i du solide
2952: /solidnormaleface {
2953: 4 dict begin
2954: /i exch def
2955: /solid exch def
2956: solid issolid not {
2957: (Error : mauvais type d argument dans solidgetsommetface) ==
2958: quit
2959: } if
2960: %% solid 0 i solidgetsommetface /G defpoint3d
2961: %% G
2962: %% solid 1 i solidgetsommetface
2963: %% vecteur3d
2964: %% G
2965: %% solid 2 i solidgetsommetface
2966: %% vecteur3d
2967:
2968: /n solid i solidfacenombressommets def
2969: solid i solidcentreface /G defpoint3d
2970: %% debug %% G 3dto2d point
2971: G
2972: solid 0 i solidgetsommetface
2973: /A defpoint3d
2974: % gsave bleu A point3d grestore
2975: A
2976: vecteur3d normalize3d
2977: G
2978: solid 1 i solidgetsommetface
2979: /A defpoint3d
2980: % gsave orange A point3d grestore
2981: A
2982: vecteur3d normalize3d
2983: vectprod3d
2984: /resultat defpoint3d
2985: resultat normalize3d
2986: end
2987: } def
2988:
2989: %%%% ### solidtransform ###
2990: %% syntaxe : solid1 {f} solidtransform --> solid2, solid2 est le
2991: %% transforme de solid1 par la transformation f : R^3 -> R^3
2992: /solidtransform {
2993: 3 dict begin
2994: /f exch def

```

```

2995: /solid exch def
2996: solid issolid not {
2997:   (Error : mauvais type d argument dans solidtransform) ==
2998:   quit
2999: } if
3000: /les_sommets
3001:   solid solidgetpointstable {f} papply3d
3002: def
3003:   solid les_sommets solidputpointstable
3004:   solid
3005: end
3006: } def
3007:
3008: %%%% ### solidputcolor ###
3009: %% syntaxe : solid i string solidputfcolor
3010: /solidputfcolor {
3011: 3 dict begin
3012:   /str exch def
3013:   /i exch def
3014:   /solid exch def
3015:   /FC solid solidgetfcolors def
3016:   i FC length lt {
3017:     FC i str put
3018:   } if
3019: end
3020: } def
3021:
3022: %% syntaxe : solid solidgetincolors --> array
3023: /solidgetincolors {
3024: 3 dict begin
3025:   /solid exch def
3026:   solid issolid not {
3027:     (Error : mauvais type d argument dans solidgetincolors) ==
3028:     quit
3029:   } if
3030:   solid solidwithinfaces {
3031:     /fcol solid solidgetfcolors def
3032:     /IO solid solidgetinouttable def
3033:     /n1 IO 2 get def
3034:     /n2 IO 3 get def
3035:     /n n2 n1 sub 1 add def
3036:     fcol n1 n getinterval
3037:   } {
3038:     []
3039:   } ifelse
3040: end
3041: } def
3042:
3043: %% syntaxe : solid solidgetoutcolors --> array
3044: /solidgetoutcolors {
3045: 3 dict begin
3046:   /solid exch def
3047:   solid issolid not {
3048:     (Error : mauvais type d argument dans solidgetoutcolors) ==
3049:     quit
3050:   } if
3051:   /fcol solid solidgetfcolors def
3052:   /IO solid solidgetinouttable def
3053:   /n1 IO 0 get def
3054:   /n2 IO 1 get def
3055:   /n n2 n1 sub 1 add def
3056:   fcol n1 n getinterval
3057: end
3058: } def
3059:
3060: %% syntaxe : solid array solidputincolors --> -
3061: /solidputincolors {
3062: 4 dict begin
3063:   /newcolorstable exch def
3064:   /solid exch def
3065:   solid issolid not {
3066:     (Error : mauvais type d argument dans solidputincolors) ==
3067:     quit
3068:   } if
3069:   /n newcolorstable length def
3070:   n solid solidnombreinfaces ne {

```

```

3071:      (Error : mauvaise longueur de tableau dans solidputincolors) ==
3072:      quit
3073:    } if
3074:    n 0 ne {
3075:      /FC solid solidgetfcolors def
3076:      /IO solid solidgetinouttable def
3077:      /nl IO 2 get def
3078:      FC nl newcolorstable putinterval
3079:    } if
3080:  end
3081: } def
3082:
3083: %% syntaxe : solid array solidputoutcolors --> -
3084: /solidputoutcolors {
3085: 4 dict begin
3086:   /newcolorstable exch def
3087:   /solid exch def
3088:   solid issolid not {
3089:     (Error : mauvais type d argument dans solidputoutcolors) ==
3090:     quit
3091:   } if
3092:   /n newcolorstable length def
3093:   n solid solidnombreoutfaces ne {
3094:     (Error : mauvaise longueur de tableau dans solidputoutcolors) ==
3095:     quit
3096:   } if
3097:   n 0 ne {
3098:     /FC solid solidgetfcolors def
3099:     /IO solid solidgetinouttable def
3100:     /nl IO 0 get def
3101:     FC nl newcolorstable putinterval
3102:   } if
3103: end
3104: } def
3105:
3106: %% syntaxe : solid str outputcolors
3107: /outputcolors {
3108: 5 dict begin
3109:   /color exch def
3110:   /solid exch def
3111:   solid issolid not {
3112:     (Error : mauvais type d argument dans inoutputcolors) ==
3113:     quit
3114:   } if
3115:   /n solid solidnombreoutfaces def
3116:   solid [ n {color} repeat ] solidputoutcolors
3117: end
3118: } def
3119:
3120: %% syntaxe : solid str inputcolors
3121: /inputcolors {
3122: 5 dict begin
3123:   /color exch def
3124:   /solid exch def
3125:   solid issolid not {
3126:     (Error : mauvais type d argument dans inoutputcolors) ==
3127:     quit
3128:   } if
3129:   /n solid solidnombreinfaces def
3130:   solid [ n {color} repeat ] solidputincolors
3131: end
3132: } def
3133:
3134: %% syntaxe : solid str1 str2 inoutputcolors
3135: /inoutputcolors {
3136: 5 dict begin
3137:   /colout exch def
3138:   /colin exch def
3139:   /solid exch def
3140:   solid colin inputcolors
3141:   solid colout outputcolors
3142: end
3143: } def
3144:
3145: %%%% ### solidputhuecolors ###
3146: %% syntaxe : solid table solidputhuecolors --> -

```

```

3147: /solidputhuecolors {
3148: 1 dict begin
3149:   2 copy pop
3150:   solidgetinouttable /IO exch def
3151:   IO 0 get
3152:   IO 1 get
3153:   s@lidpuehec@l@rs
3154: end
3155: } def
3156:
3157: /solidputinhuecolors {
3158: 2 dict begin
3159:   /table exch def
3160:   /solid exch def
3161:   solid solidgetinouttable /IO exch def
3162:   solid solidwithinfaces {
3163:     solid table
3164:     IO 2 get
3165:     IO 3 get
3166:     s@lidpuehec@l@rs
3167:   } if
3168: end
3169: } def
3170:
3171: /solidputinouthuecolors {
3172: 1 dict begin
3173:   2 copy pop
3174:   solidgetinouttable /IO exch def
3175:   IO 0 get
3176:   IO 3 get IO 1 get max
3177:   s@lidpuehec@l@rs
3178: end
3179: } def
3180:
3181: %% syntaxe : solid table n1 n2 s@lidpuehec@l@rs --> -
3182: %% affecte les couleurs des faces d indice n1 a n2 du solid solid, par
3183: %% un degrade defini par la table.
3184: /s@lidpuehec@l@rs {
3185: 9 dict begin
3186:   /n2 exch def
3187:   /n1 exch def
3188:   /table exch def
3189:   /solid exch def
3190:   /n n2 n1 sub def
3191:
3192:   table length 2 eq {
3193:     /a0 table 0 get def
3194:     /a1 table 1 get def
3195:     a1 isstring {
3196:       /lacouleurdepart {
3197:         gsave
3198:           [a0 cvx exec] length 0 eq {
3199:             a0 cvx exec currentrgbcolor
3200:           } {
3201:             a0 cvx exec
3202:           } ifelse
3203:         grestore
3204:       } def
3205:       /lacouleurarrivee {
3206:         gsave
3207:           [a1 cvx exec] length 0 eq {
3208:             a1 cvx exec currentrgbcolor
3209:           } {
3210:             a1 cvx exec
3211:           } ifelse
3212:         grestore
3213:       } def
3214:       /table [lacouleurdepart lacouleurarrivee] def
3215:     } {
3216:       /A {a0 i a1 a0 sub mul n 1 sub div add} def
3217:       /B {1} def
3218:       /C {1} def
3219:       /D {} def
3220:       /espacedecouleurs (sethsbcolor) def
3221:     } ifelse
3222:   } if

```

```

3223 :
3224 :   table length 4 eq {
3225 :     /a0 table 0 get def
3226 :     /a1 table 1 get def
3227 :     /A {a0 i a1 a0 sub mul n 1 sub div add} def
3228 :     /B table 2 get def
3229 :     /C table 3 get def
3230 :     /D {} def
3231 :     /espacedecouleurs (sethsbcolor) def
3232 :   } if
3233 :
3234 :   table length 6 eq {
3235 :     /a0 table 0 get def
3236 :     /b0 table 1 get def
3237 :     /c0 table 2 get def
3238 :     /a1 table 3 get def
3239 :     /b1 table 4 get def
3240 :     /c1 table 5 get def
3241 :     /A {a0 i a1 a0 sub mul n 1 sub div add} def
3242 :     /B {b0 i b1 b0 sub mul n 1 sub div add} def
3243 :     /C {c0 i c1 c0 sub mul n 1 sub div add} def
3244 :     /D {} def
3245 :     /espacedecouleurs (setrgbcolor) def
3246 :   } if
3247 :
3248 :   table length 7 eq {
3249 :     /a0 table 0 get def
3250 :     /b0 table 1 get def
3251 :     /c0 table 2 get def
3252 :     /a1 table 3 get def
3253 :     /b1 table 4 get def
3254 :     /c1 table 5 get def
3255 :     /A {a0 i a1 a0 sub mul n 1 sub div add} def
3256 :     /B {b0 i b1 b0 sub mul n 1 sub div add} def
3257 :     /C {c0 i c1 c0 sub mul n 1 sub div add} def
3258 :     /D {} def
3259 :     /espacedecouleurs (sethsbcolor) def
3260 :   } if
3261 :
3262 :   table length 8 eq {
3263 :     /a0 table 0 get def
3264 :     /b0 table 1 get def
3265 :     /c0 table 2 get def
3266 :     /d0 table 3 get def
3267 :     /a1 table 4 get def
3268 :     /b1 table 5 get def
3269 :     /c1 table 6 get def
3270 :     /d1 table 7 get def
3271 :     /A {a0 i a1 a0 sub mul n 1 sub div add} def
3272 :     /B {b0 i b1 b0 sub mul n 1 sub div add} def
3273 :     /C {c0 i c1 c0 sub mul n 1 sub div add} def
3274 :     /D {d0 i d1 d0 sub mul n 1 sub div add} def
3275 :     /espacedecouleurs (setcmykcolor) def
3276 :   } if
3277 :
3278 :   n1 1 n2 {
3279 :     /i exch def
3280 :     solid i
3281 :     [A B C D] espacedecouleurs astr2str
3282 :     solidputfcolor
3283 :   } for
3284 :
3285 : end
3286 : } def
3287 :
3288 : %%%% ### solidrmface ###
3289 : %% syntaxe : solid i solidrmface -> -
3290 : /solidrmface {
3291 : 5 dict begin
3292 :   /i exch def
3293 :   /solid exch def
3294 :   solid issolid not {
3295 :     (Error : mauvais type d argument dans solidrmface) ==
3296 :     quit
3297 :   } if
3298 :   %% on enleve la face

```

```

3299:   /F solid solidgetfaces def
3300:   F length 1 sub i lt {
3301:     (Error : indice trop grand dans solidrmface) ==
3302:     quit
3303:   } if
3304:   [
3305:     0 1 F length 1 sub {
3306:       /j exch def
3307:       i j ne {
3308:         F j get
3309:       } if
3310:     } for
3311:   ]
3312:   /NF exch def
3313:   solid NF solidputfaces
3314:   %% on enleve la couleur correspondante
3315:   /FC solid solidgetfcolors def
3316:   [
3317:     0 1 FC length 1 sub {
3318:       /j exch def
3319:       i j ne {
3320:         FC j get
3321:       } if
3322:     } for
3323:   ]
3324:   /NFC exch def
3325:   solid NFC solidputfcolors
3326:   %% on ajuste la table inout
3327:   /IO solid solidgetinouttable def
3328:   solid i solidisoutface {
3329:     IO 1 IO 1 get 1 sub put
3330:     solid solidwithinfaces {
3331:       IO 2 IO 2 get 1 sub put
3332:       IO 3 IO 3 get 1 sub put
3333:     } if
3334:   } if
3335:   solid i solidisinface {
3336:     IO 1 IO 1 get 1 sub put
3337:     IO 2 IO 2 get 1 sub put
3338:     IO 3 IO 3 get 1 sub put
3339:   } if
3340:   solid IO solidputinouttable
3341: end
3342: } def
3343:
3344: %% syntaxe : solid table solidrmfaces --> -
3345: /solidrmfaces {
3346: 2 dict begin
3347:   /table exch bubblesort reverse def
3348:   /solid exch def
3349:   table {solid exch solidrmface} apply
3350: end
3351: } def
3352:
3353: %%%% ### videsolid ###
3354: %% syntaxe : solid videsolid -> -
3355: /videsolid {
3356: 5 dict begin
3357:   /solid exch def
3358:   solid issolid not {
3359:     (Error : mauvais type d argument dans videsolid) ==
3360:     quit
3361:   } if
3362:   solid solidwithinfaces not {
3363:     /IO solid solidgetinouttable def
3364:     /FE solid solidgetfaces def
3365:     /n FE length def
3366:     IO 2 n put
3367:     IO 3 2 n mul 1 sub put
3368:     %% on inverse chaque face
3369:     /FI FE {reverse} apply def
3370:     solid FE FI append solidputfaces
3371:     %% et on rajoute autant de couleurs vides que de faces
3372:     /FEC solid solidgetfcolors def
3373:     /FIC [FI length {}] repeat] def
3374:     solid FEC FIC append solidputfcolors

```

```

3375:         solid IO solidputinouttable
3376:     } if
3377: end
3378: } def
3379:
3380: %%%% ### solidnumfaces ###
3381: %% syntaxe : solid array solidnumfaces
3382: %% syntaxe : solid array bool solidnumfaces
3383: %% array, le tableau des indices des faces a numeroter, est optionnel
3384: %% si bool=true, on ne numerote que les faces visibles
3385: /solidnumfaces {
3386: 5 dict begin
3387:     dup isbool {
3388:         /bool exch def
3389:     } {
3390:         /bool true def
3391:     } ifelse
3392:     setTimes
3393:     dup issolid not {
3394:         %% on a un argument
3395:         /option exch def
3396:     } if
3397:     /sol exch def
3398:     /n sol solidnombrefaces def
3399:     currentdict /option known not {
3400:         /option [0 1 n 1 sub {} for] def
3401:     } if
3402:
3403:     0 1 option length 1 sub {
3404:         /i exch def
3405:         /j option i get def
3406:         j (      ) cvs sol j bool cctextp3d
3407:     } for
3408: end
3409: } def
3410:
3411: %%%% ### creusesolid ###
3412: %% syntaxe : solid creusesolid -> -
3413: /creusesolid {
3414: 5 dict begin
3415:     /solid exch def
3416:     solid issolid not {
3417:         (Error : mauvais type d argument dans creusesolid) ==
3418:         quit
3419:     } if
3420:     %% on enleve le fond et le chapeau
3421:     solid 1 solidrmface
3422:     solid 0 solidrmface
3423:     %% on inverse chaque face
3424:     solid videsolid
3425: end
3426: } def
3427:
3428: %%%% ### fin insertion ###
3429:

```

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

```

3430: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3431: %%%          dessin des solides          %%%
3432: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3433:
3434: %%%% ### solidisinface ###
3435: %% syntaxe : solid i solidisinface --> bool
3436: %% true si i est l indice d une face interne, false sinon
3437: /solidisinface {
3438: 4 dict begin
3439:     /i exch def
3440:     solidgetinouttable /IO exch def
3441:     /n1 IO 2 get def
3442:     /n2 IO 3 get def

```



```

3443:     n1 i le
3444:     i n2 le and
3445: end
3446: } def
3447:
3448: %%%% ### solidisoutface ###
3449: %% syntaxe : solid i solidisoutface --> bool
3450: %% true si i est l indice d une face externe, false sinon
3451: /solidisoutface {
3452: 4 dict begin
3453:   /i exch def
3454:   solidgetinouttable /IO exch def
3455:   /n1 IO 0 get def
3456:   /n2 IO 1 get def
3457:   n1 i le
3458:   i n2 le and
3459: end
3460: } def
3461:
3462: %%%% ### planvisible ###
3463: %% syntaxe : A k planvisible? --> true si le plan est visible
3464: /planvisible? {
3465: 4 dict begin
3466:   /normale_plan defpoint3d
3467:   /origine defpoint3d
3468:   /ligne_de_vue {
3469:     origine
3470:     GetCamPos
3471:     vecteur3d
3472:   } def
3473:   ligne_de_vue normale_plan scalprod3d 0 gt
3474: end
3475: } def
3476:
3477: %%%% ### drawsolid ###
3478: %% syntaxe : solid i solidfacevisible? --> true si la face est visible
3479: /solidfacevisible? {
3480: 4 dict begin
3481:   /i exch def
3482:   /solid exch def
3483:   solid issolid not {
3484:     (Error : mauvais type d argument dans solidgetsommetface) ==
3485:     quit
3486:   } if
3487:   solid i solidgetface length 2 le {
3488:     true
3489:   } {
3490:     /ligne_de_vue {
3491:       solid i solidcentreface
3492:       GetCamPos
3493:       vecteur3d
3494:     } def
3495:
3496:     /normale_face {
3497:       solid i solidnormaleface
3498:     } def
3499:     ligne_de_vue normale_face scalprod3d 0 gt
3500:   } ifelse
3501: end
3502: } def
3503:
3504: %% syntaxe : solid i affectecouleursolid_facei --> si la couleur de
3505: %% la face i est definie, affecte fillstyle a cette couleur
3506: /affectecouleursolid_facei {
3507: 3 dict begin
3508:   /i exch def
3509:   /solid exch def
3510:   solid solidgetfcolors /FC exch def
3511:   FC length 1 sub i ge {
3512:     FC i get length 1 ge {
3513:       /fillstyle FC i get ( fill) append cvx
3514:       true
3515:     } {
3516:       false
3517:     } ifelse
3518:   } {

```

```

3519:         false
3520:     } ifelse
3521: end
3522: {def} if
3523: } def
3524:
3525: %% syntaxe : A solid i dessinefacecachee
3526: /dessinefacecachee {
3527: 6 dict begin
3528:     /i exch def
3529:     /solid exch def
3530:     solid issolid not {
3531:         (Error : mauvais type d argument dans dessinefacecachee) ==
3532:         quit
3533:     } if
3534:     /A exch def
3535:
3536:     /F solid solidgetfaces def
3537:     /S solid solidgetpointstable def
3538:
3539:     solid i solidfacevisible? not {
3540:         %% face cachee => on prend chacune des aretes de la face et on
3541:         %% regarde si elle est deja dessinee.
3542:         4 dict begin
3543:             /n F i get length def %% nb de sommets de la face
3544:             0 1 n 1 sub {
3545:                 /k exch def
3546:                 /k1 F i k get_ij def           %% indice sommet1
3547:                 /k2 F i k 1 add n mod get_ij def %% indice sommet2
3548:                 A k1 k2 get_ij not {
3549:                     gsave
3550:                         currentlinewidth .5 mul setlinewidth
3551:                         pointilles
3552:                         [S k1 getp3d
3553:                         S k2 getp3d] ligne3d
3554:                         A k1 k2 true put_ij
3555:                         A k2 k1 true put_ij
3556:                     grestore
3557:                 } if
3558:             } for
3559:         end
3560:     } if
3561: end
3562: } def
3563:
3564: %% syntaxe : A solid i dessinefacevisible
3565: /dessinefacevisible {
3566: 7 dict begin
3567:     /i exch def
3568:     /solid exch def
3569:     /A exch def
3570:     solid issolid not {
3571:         (Error : mauvais type d argument dans dessinefacevisible) ==
3572:         quit
3573:     } if
3574:     /F solid solidgetfaces def
3575:     /S solid solidgetpointstable def
3576:
3577:     solid i solidfacevisible? {
3578:         /n F i get length def %% nb de sommets de la face
3579:
3580:         startest {
3581:             %% choix de la couleur
3582:             /lightcolor where {
3583:                 pop
3584:                 /coeff
3585:                 lightintensity
3586:                 solid i solidnormaleface normalize3d
3587:                 solid i solidcentreface lightsrc vecteur3d normalize3d
3588:                 scalprod3d mul
3589:                 0 max 1 min
3590:             def
3591:             /fillstyle {
3592:                 lightcolor {coeff mul} apply setcolor fill
3593:             } def
3594:             solidgrid not {

```

```

3595:         lightcolor {coeff mul} apply setcolor
3596:     } if
3597: } {
3598:     /lightsrc where {
3599:         pop
3600:         /coeff
3601:         lightintensity
3602:         solid i solidnormaleface normalize3d
3603:         solid i solidcentreface lightsrc vecteur3d normalize3d
3604:         scalprod3d mul
3605:         0 max 1 min
3606:     def
3607:     /lacouleur [
3608:         gsave
3609:         solid solidgetfcolors i get cvx exec currentrgbcolor
3610:         grestore
3611:     ] def
3612:     /fillstyle {
3613:         lacouleur {coeff mul} apply setcolor fill
3614:     } def
3615:     solidgrid not {
3616:         lacouleur {coeff mul} apply setcolor
3617:     } if
3618: } {
3619: %         solid F i get length affectecouleursolid_ncotes
3620:         solid i affectecouleursolid_facei
3621:     } ifelse
3622: } ifelse
3623: } if
3624:
3625:
3626: /face_a_dessiner [ %% face visible : F [i]
3627:     0 1 n 1 sub {
3628:         /j exch def
3629:         solid j i solidgetsommetface
3630:     } for
3631: ] def
3632: face_a_dessiner polygone3d
3633: /lignedeniveau [] def
3634:
3635: %% trace de la ligne de niveau
3636: tracelignedeniveau? {
3637:     gsave
3638:         linewidthlignedeniveau setlinewidth
3639:         couleurlignedeniveau
3640:         0 1 n 1 sub {
3641:             /j exch def
3642:             face_a_dessiner j getp3d
3643:             face_a_dessiner j 1 add n mod getp3d
3644:             hauteurlignedeniveau segment_inter_planz {
3645:                 1 dict begin
3646:                     /table exch def
3647:                     /lignedeniveau [
3648:                         lignedeniveau aload pop
3649:                         table 0 getp3d
3650:                         table length 4 ge {
3651:                             table 1 getp3d
3652:                         } if
3653:                     ] store
3654:                 end
3655:             } if
3656:         } for
3657:         lignedeniveau length 4 ge
3658:         {lignedeniveau ligne3d}
3659:         if
3660:     grestore
3661: } if
3662:
3663: %% on marque les aretes
3664: 0 1 n 1 sub {
3665:     /j exch def
3666:     /k1 F i j get_ij def %% indice sommet1
3667:     /k2 F i j 1 add n mod get_ij def %% indice sommet2
3668:     A k1 k2 true put_ij
3669:     A k2 k1 true put_ij
3670: } for

```

```

3671:   } if
3672: end
3673: } def
3674:
3675: /drawsolid* {
3676: 1 dict begin
3677:   /startest {true} def
3678:   drawsolid
3679: end
3680: } def
3681:
3682: /peintrealgorithme false def
3683:
3684: /drawsolid** {
3685: 2 dict begin
3686:   /aretescachees false def
3687:   /peintrealgorithme true def
3688:   drawsolid*
3689: end
3690: } def
3691:
3692: %% syntaxe : solid drawsolid
3693: /drawsolid {
3694: 7 dict begin
3695:   /solid exch def
3696:   solid issolid not {
3697:     (Error : mauvais type d argument dans drawsolid) ==
3698:     quit
3699:   } if
3700:   solid nullsolid not {
3701:     solid solidgetfaces
3702:     /F exch def
3703:     solid solidgetpointstable
3704:     /S exch def
3705:     /n S length 3 idiv def
3706:     %% tableau des aretes
3707:     /A [
3708:       n {
3709:         [n {false} repeat]
3710:       } repeat
3711:     ] def
3712:
3713:     peintrealgorithme {
3714:       %% tri des indices des faces par distance decroissante
3715:       [
3716:         0 1 F length 1 sub {
3717:           /i exch def
3718:           solid i solidcentreface
3719:           GetCamPos
3720:           distance3d
3721:         } for
3722:       ] doublequicksort pop reverse
3723:     } {
3724:       [
3725:         0 1 F length 1 sub {
3726:           } for
3727:       ]
3728:     } ifelse
3729:     /ordre exch def
3730:
3731:     0 1 F length 1 sub {
3732:       /k exch def
3733:       /i ordre k get def
3734:       gsave
3735:       A solid i dessinefacevisible
3736:       grestore
3737:     } for
3738:     aretescachees {
3739:       0 1 F length 1 sub {
3740:         /k exch def
3741:         /i ordre k get def
3742:         A solid i dessinefacecachees
3743:       } for
3744:     } if
3745:     %% %% si on veut repasser les traits des faces visibles
3746:     %% 0 1 F length 1 sub {

```

```

3747: %%          /k exch def
3748: %%          /i ordre k get def
3749: %%          gsave
3750: %%          1 dict begin
3751: %%              /startest false def
3752: %%              A solid i dessinefacevisible
3753: %%          end
3754: %%          grestore
3755: %%          } for
3756: %%      } if
3757: %%  end
3758: %% } def
3759:
3760: %%%% ### segment_inter_planz ###
3761: %% syntaxe : A B k segment_inter_planz --> array true ou false
3762: /segment_inter_planz {
3763: 4 dict begin
3764:   /k exch def
3765:   /B defpoint3d
3766:   /A defpoint3d
3767:   A /zA exch def pop pop
3768:   B /zB exch def pop pop
3769:   zA k sub zB k sub mul dup 0 gt {
3770:     %% pas d intersection
3771:     pop
3772:     false
3773:   } {
3774:     0 eq {
3775:       %% intersection en A ou en B
3776:       [
3777:         zA k eq {A} if
3778:         zB k eq {B} if
3779:       ] true
3780:     } {
3781:       %% intersection entre A et B
3782:       [
3783:         A B vecteur3d
3784:         k zA sub zB zA sub div mulv3d
3785:         A addv3d
3786:       ] true
3787:     } ifelse
3788:   } ifelse
3789: end
3790: } def
3791:
3792: %%%% ### fin insertion ###
3793:

```

## 8.18 - Le cube tronqué

```

3794: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3795: %%%%          operations sur des solides particuliers          %%%%
3796: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3797:
3798: %%%% ### tronquecube ###
3799: %% syntaxe : solid n tronque_cube --> solid (tronque)
3800: /tronque_cube {
3801: 6 dict begin
3802:   /d exch def
3803:   /solid exch def
3804:   solid issolid not {
3805:     (Error : mauvais type d argument dans tronque_cube) ==
3806:     quit
3807:   } if
3808:   solid solidgetpointstable
3809:   /s exch def
3810:   /co [
3811:     3 4 1 % 1 3 4 % voisins du sommet 0
3812:     0 5 2 % 0 2 5 %           de 1
3813:     1 6 3 % 1 3 6 %           de 2
3814:     2 7 0 % 0 2 7 %           de 3

```

```

3815:      7 0 5 % 0 5 7 %           de 4
3816:      4 1 6 % 1 4 6 %           de 5
3817:      5 2 7 % 2 5 7 %           de 6
3818:      6 3 4 % 3 4 6 %           de 7
3819:    ] def
3820:
3821:    /dd {d 1 sub} bind def
3822:    /i 0 def
3823:    /les_sommets [ % les coordonnees des sommets du cube tronque
3824:      0 3 21 {
3825:        /j exch def
3826:        %% sommet d indice i = A1
3827:        solid i solidgetsommet /A1 defpoint3d
3828:
3829:        %% k = indice du sommet voisin no 1
3830:        co j get /k exch def
3831:        %% sommet d indice k = A2
3832:        solid k solidgetsommet /A2 defpoint3d
3833:        %% barycentre {(A1, d) (A2, 1)}
3834:        A1 d A2 1 barycentre3d
3835:
3836:        %% k = indice du sommet voisin no 2
3837:        co j 1 add get /k exch def
3838:        %% sommet d indice k = A2
3839:        solid k solidgetsommet /A2 defpoint3d
3840:        %% barycentre {(A1, d) (A2, 1)}
3841:        A1 d A2 1 barycentre3d
3842:
3843:        %% k = indice du sommet voisin no 2
3844:        co j 2 add get /k exch def
3845:        %% sommet d indice k = A2
3846:        solid k solidgetsommet /A2 defpoint3d
3847:        %% barycentre {(A1, d) (A2, 1)}
3848:        A1 d A2 1 barycentre3d
3849:
3850:        /i i 1 add store
3851:      } for
3852:    ] def
3853:
3854:    /les_faces [
3855:      [11 10 22 23 12 13 1 0]
3856:      [2 1 13 14 15 16 4 3]
3857:      [8 7 19 20 21 22 10 9]
3858:      [14 12 23 21 20 18 17 15]
3859:      [3 5 6 8 9 11 0 2]
3860:      [5 4 16 17 18 19 7 6]
3861:      [0 1 2]
3862:      [3 4 5]
3863:      [6 7 8]
3864:      [9 10 11]
3865:      [12 14 13]
3866:      [15 17 16]
3867:      [18 20 19]
3868:      [21 23 22]
3869:    ] def
3870:
3871:    solid les_sommets solidputpointstable
3872:    solid les_faces solidputfaces
3873:    solid dup solidgetfcolors [8 {} repeat] append solidputfcolors
3874:    solid
3875:  end
3876: } def
3877:
3878: %%%% ### fin insertion ###
3879:

```

## 8.19 - Les solides prédéfinis

Le fichier *solides.pro*

```
3880: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3881: %%%          quelques solides precalcules          %%%
3882: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3883:
3884: %%% ## newface ##
3885: %% syntaxe : array newmonoface -> solid
3886: %% ou array = tableau de points 2d
3887: /newmonoface {
3888: 4 dict begin
3889:   /table exch def
3890:   /n table length 2 idiv def
3891:   /S table {0} papply def
3892:
3893:   /F [
3894:     [0 1 n 1 sub {} for]
3895:   ] def
3896:   S F generesolid
3897: end
3898: } def
3899:
3900: %% syntaxe : array newbiface -> solid
3901: %% ou array = tableau de points 2d
3902: /newbiface {
3903:   newmonoface
3904:   dup videsolid
3905: } def
3906:
3907: %%% ## newpolreg ##
3908: %% syntaxe : r n newpolreg --> solid
3909: /newpolreg {
3910: 5 dict begin
3911:   /n exch def
3912:   /r exch def
3913:   /S [
3914:     0 360 n div 360 360 n div sub {
3915:       /theta exch def
3916:       theta cos r mul
3917:       theta sin r mul
3918:       0
3919:     } for
3920:   ] def
3921:   /F [
3922:     [0 1 n 1 sub {} for]
3923:   ] def
3924:
3925:   S F generesolid
3926:   dup videsolid
3927: end
3928: } def
3929:
3930: %%% ## newgrille ##
3931: %% syntaxe : xmin xmax ymin ymax [dx dy] newgrille -> solid
3932: %% syntaxe : xmin xmax ymin ymax [nx ny] newgrille -> solid
3933: %% syntaxe : xmin xmax ymin ymax {mode} newgrille -> solid
3934: %% syntaxe : xmin xmax ymin ymax newgrille -> solid
3935: /newgrille {
3936: 10 dict begin
3937:   [[/nx /ny] [1 1] [1. 1.] [1. 1.] [1. 1.] [.5 .5]] gestionsolidmode
3938:   %% ny nb d etages en y
3939:   %% nx nb d etages en x
3940:   [nx ny] {0} newsurfaceparametree
3941: end
3942: } def
3943:
3944: /newsurface {
3945:   true newsurfaceparametree
3946: } def
3947:
3948: /newsurfaceparametree {
3949: 10 dict begin
3950:   dup isbool {
3951:     pop /surfz true def
3952:   } {
3953:     /surfz false def
```

```

3954:   } ifelse
3955:   /f_surface exch def
3956:   [[/nx /ny] [2 2] [4 4] [1. 1.] [1. 1.] [.25 .25]] gestionsolidmode
3957:   %% ny nb d etages en y
3958:   %% nx nb d etages en x
3959:   /ymax exch def
3960:   /ymin exch def
3961:   /xmax exch def
3962:   /xmin exch def
3963:
3964:   nx isinteger not {
3965:     %% alors nx est un dx
3966:     /nx xmax xmin sub nx div cvi store
3967:   } if
3968:   ny isinteger not {
3969:     %% alors ny est un dy
3970:     /ny ymax ymin sub ny div cvi store
3971:   } if
3972:   /dy ymax ymin sub ny div def %% le pas sur y
3973:   /dx xmax xmin sub nx div def %% le pas sur x
3974:
3975:   /S [
3976:     0 1 nx {
3977:       /i exch def
3978:       0 1 ny {
3979:         /j exch def
3980:         /u xmin i dx mul add def
3981:         /v ymin j dy mul add def
3982:         u v
3983:         surfz {2 copy} if
3984:         f_surface
3985:         pstrickactionR3
3986:       } for
3987:     } for
3988:   ] def
3989:
3990:   /F [
3991:     0 1 nx 1 sub {
3992:       /i exch def
3993:       0 1 ny 1 sub {
3994:         /j exch def
3995:         [
3996:           j 1 add      i ny 1 add mul add
3997:           j            i ny 1 add mul add
3998:           j ny 1 add add i ny 1 add mul add
3999:           j ny 2 add add i ny 1 add mul add
4000:         ]
4001:       } for
4002:     } for
4003:     %% 0 1 0 {%nx 1 sub {
4004:     %% /i exch def
4005:     %% 0 1 0 {%ny 2 sub {
4006:     %% /j exch def
4007:     %% [
4008:     %% j 1 add      %% i ny mul add
4009:     %% j            %% i ny mul add
4010:     %% ny 1 add j add  %% i ny mul add
4011:     %% ny 2 add j add  %% i ny mul add
4012:     %% ]
4013:     %% } for
4014:     %% } for
4015:   ] def
4016:   S F generesolid
4017:   dup videsolid
4018: end
4019: } def
4020:
4021: %%%% ### newgrillecirculaire ###
4022: %% syntaxe : r option newgrillecirculaire -> solid
4023: /newgrillecirculaire {
4024: 6 dict begin
4025:   [[/K /N] [6 6] [6 8] [10 8] [16 12] [16 36]] gestionsolidmode
4026:
4027:   %% N = nb de meridians (diviseur de 360 = 2^4 * 3^2 * 5)
4028:   %% K = nb d horizontales (diviseur de 160 = 2^5 * 5)
4029:

```



```

4030: /r exch def
4031: /F [
4032:   %% 1er etage
4033:   1 1 N {
4034:     /i exch def
4035:     [0 i i N mod 1 add]
4036:   } for
4037:   %% etages suivants
4038:   0 1 K 2 sub {
4039:     /j exch def
4040:     1 1 N {
4041:       /i exch def
4042:       [i      j N mul add
4043:        i N add j N mul add
4044:        i N mod N add 1 add j N mul add
4045:        i N mod 1 add j N mul add]
4046:     } for
4047:   } for
4048: ] def
4049:
4050: %% tableau des sommets
4051: /S [
4052:   0 0 0
4053:   1 1 K {
4054:     /j exch def
4055:     1 1 N {
4056:       /i exch def
4057:       /theta i 360 mul N div def
4058:       theta cos r j mul K div mul
4059:       theta sin r j mul K div mul
4060:       2 copy exch atan 90 div
4061:     } for
4062:   } for
4063: ] def
4064:
4065: S F generesolid
4066: end
4067: } def
4068:
4069: %%%% ### newruban ###
4070: %% syntaxe : array h u [n] newruban -> solid d axe (0, u), de maillage vertical n
4071: %% syntaxe : array h u newruban -> solid d axe (0, u),
4072: %% syntaxe : array h newruban -> solid d axe (0, k),
4073: %% ou array tableau de points 2d
4074: /newruban {
4075: 7 dict begin
4076:   %% N = nb d etages
4077:   [[/N] [1] [1] [1] [3] [4]] gestionsolidmode
4078:   2 copy pop isarray {
4079:     /u {0 0 1} def
4080:   } {
4081:     /u defpoint3d
4082:   } ifelse
4083:   u 0 eq {
4084:     (Error : 3eme composante nulle dans le vecteur pour newruban) ==
4085:     quit
4086:   } if
4087:   pop pop
4088: /h exch def
4089: /table exch def
4090: %% n = indice du dernier point
4091: /n table length 2 idiv 1 sub def
4092: %% vecteur de translation
4093: u
4094: h u norme3d div
4095: mulv3d /v defpoint3d
4096:
4097: %% tableau des sommets
4098: /S [
4099:   0 1 N {
4100:     /j exch def
4101:     0 1 n {
4102:       /i exch def
4103:       table i getp
4104:       0
4105:       v N j sub N div mulv addv3d

```

```

4106:         } for
4107:     } for
4108: ] def
4109:
4110: /F [
4111:     %% faces etage
4112:     1 1 N {
4113:         /j exch def
4114:         1 1 n {
4115:             /i exch def
4116:             [i                j 1 sub n 1 add mul add
4117:             i 1 sub          j 1 sub n 1 add mul add
4118:             n 1 add i add 1 sub j 1 sub n 1 add mul add
4119:             n 1 add i add      j 1 sub n 1 add mul add]
4120:         } for
4121:     } for
4122: ] def
4123:
4124:     S F generesolid
4125:     dup videsolid
4126: end
4127: } def
4128:
4129: %%%% ### newicosaedre ###
4130: /newicosaedre {
4131: 3 dict begin
4132:     /a exch def
4133:     /S [
4134:         0.8944271 0          0.4472137
4135:         0.2763932 0.8506507 0.4472137
4136:         -0.7236067 0.5257311 0.4472137
4137:         -0.7236067 -0.5257311 0.4472137
4138:         0.2763932 -0.8506507 0.4472137
4139:         0          0          1
4140:         0          0          -1
4141:         -0.8944271 0          -0.4472137
4142:         -0.2763932 -0.8506507 -0.4472137
4143:         0.7236067 -0.5257311 -0.4472137
4144:         0.7236067 0.5257311 -0.4472137
4145:         -0.2763932 0.8506507 -0.4472137
4146:     ] {a mulv3d} papply3d def
4147:
4148:     /F [
4149:         [0 1 5] %% 1 2 6 ]
4150:         [1 2 5] %% 2 3 6 ]
4151:         [2 3 5] %% 3 4 6 ]
4152:         [3 4 5] %% 4 5 6 ]
4153:         [4 0 5] %% 5 1 6 ]
4154:         [9 0 4] %% 10 1 5 ]
4155:         [0 9 10] %% 1 10 11]
4156:         [10 1 0] %% 11 2 1 ]
4157:         [1 10 11] %% 2 11 12]
4158:         [11 2 1] %% 12 3 2 ]
4159:         [2 11 7] %% 3 12 8 ]
4160:         [2 7 3] %% 3 8 4 ]
4161:         [3 7 8] %% 4 8 9 ]
4162:         [3 8 4] %% 4 9 5 ]
4163:         [4 8 9] %% 5 9 10 ]
4164:         [6 7 11] %% 7 8 12 ]
4165:         [6 8 7] %% 7 9 8 ]
4166:         [6 9 8] %% 7 10 9 ]
4167:         [6 10 9] %% 7 11 10]
4168:         [6 11 10] %% 7 12 11]
4169:     ] def
4170:
4171:     S F generesolid
4172: end
4173: } def
4174:
4175: %%%% ### newdodecaedre ###
4176: /newdodecaedre {
4177: 3 dict begin
4178:     /a exch def
4179:     /S [
4180:         0          0.607062 0.7946545
4181:         -0.5773503 0.1875925 0.7946545

```

```

4182:      -0.3568221 -0.4911235 0.7946545
4183:      0.3568221 -0.4911235 0.7946545
4184:      0.5773503 0.1875925 0.7946545
4185:      0          0.982247  0.1875925
4186:     -0.9341724 0.303531  0.1875925
4187:     -0.5773503 -0.7946645 0.1875925
4188:      0.5773503 -0.7946645 0.1875925
4189:      0.9341724 0.303531  0.1875925
4190:      0          -0.982247 -0.1875925
4191:      0.9341724 -0.303531 -0.1875925
4192:      0.5773503 0.7946545 -0.1875925
4193:     -0.5773503 0.7946545 -0.1875925
4194:     -0.9341724 -0.303531 -0.1875925
4195:     -0.5773503 -0.1875925 -0.7946545
4196:     -0.3568221 0.4911235 -0.7946545
4197:      0.3568221 0.4911235 -0.7946545
4198:      0.5773503 -0.1875925 -0.7946545
4199:      0          -0.607062 -0.7946545
4200:      ] {a mulv3d} papply3d def
4201:
4202:      /F [
4203:        [0 1 2 3 4]
4204:        [4 3 8 11 9]
4205:        [4 9 12 5 0]
4206:        [0 5 13 6 1]
4207:        [1 6 14 7 2]
4208:        [2 7 10 8 3]
4209:        [10 19 18 11 8]
4210:        [11 18 17 12 9]
4211:        [12 17 16 13 5]
4212:        [13 16 15 14 6]
4213:        [14 15 19 10 7]
4214:        [15 16 17 18 19]
4215:      ] def
4216:      S F generesolid
4217: end
4218: } def
4219:
4220: %%%% ### newoctaedre ###
4221: /newoctaedre {
4222: 3 dict begin
4223:   /a exch def
4224:   %%Sommets
4225:   /S [
4226:     0 0 1
4227:     1 0 0
4228:     0 1 0
4229:     -1 0 0
4230:     0 -1 0
4231:     0 0 -1
4232:   ] {a mulv3d} papply3d def
4233:
4234:   /F [
4235:     [0 4 1]
4236:     [1 2 0]
4237:     [0 2 3]
4238:     [3 4 0]
4239:     [1 5 2]
4240:     [2 5 3]
4241:     [3 5 4]
4242:     [4 5 1]
4243:   ] def
4244:
4245:   S F generesolid
4246: end
4247: } def
4248:
4249: %%%% ### newtetraedre ###
4250: /newtetraedre {
4251: 3 dict begin
4252:   /r exch def
4253:   %%Tetraedre
4254:   /S [
4255:     0          0          1
4256:     -0.4714045 -0.8164965 -1 3 div
4257:     0.942809  0          -1 3 div

```

```

4258:      -0.4714045 0.8164965 -1 3 div
4259:    ] {r mulv3d} papply3d def
4260:
4261:    /F [
4262:      [0 1 2]
4263:      [0 2 3]
4264:      [0 3 1]
4265:      [1 3 2]
4266:    ] def
4267:
4268:    S F generesolid
4269:  end
4270: } def
4271:
4272: %%%% ### newcube ###
4273: /newcube {
4274: 3 dict begin
4275:   [[/n] [1] [1] [1] [3] [4]] gestionsolidmode
4276:   /a exch 2 div def
4277:
4278:   n 1 le {
4279:     /F [
4280:       [0 1 2 3]
4281:       [0 4 5 1]
4282:       [1 5 6 2]
4283:       [2 6 7 3]
4284:       [0 3 7 4]
4285:       [4 7 6 5]
4286:     ] def
4287:
4288:     %% tableau des sommets
4289:     /S [
4290:       1 1 1 %% 0
4291:       -1 1 1 %% 1
4292:       -1 -1 1 %% 2
4293:       1 -1 1 %% 3
4294:       1 1 -1 %% 4
4295:       -1 1 -1 %% 5
4296:       -1 -1 -1 %% 6
4297:       1 -1 -1 %% 7
4298:     ] {a mulv3d} papply3d def
4299:     S F generesolid
4300:   } {
4301:     /dl 2 n div def
4302:     /N n dup mul n add 4 mul def
4303:     /nl n 1 sub dup mul def %% nb sommets centre d une face
4304:
4305:     %% tableau des sommets
4306:     /S1 [
4307:     0 1 n 1 sub {
4308:       /j exch def
4309:       0 1 n {
4310:         /i exch def
4311:         -1 i dl mul add
4312:         -1 j dl mul add
4313:       }
4314:     } for
4315:   } for
4316: ] def
4317:
4318: /S2 S1 {-90 0 0 rotateOpoint3d} papply3d def
4319: /S3 S2 {-90 0 0 rotateOpoint3d} papply3d def
4320: /S4 S3 {-90 0 0 rotateOpoint3d} papply3d def
4321:
4322: /S5 [
4323: 1 1 n 1 sub {
4324:   /j exch def
4325:   1 1 n 1 sub {
4326:     /i exch def
4327:     1
4328:     -1 i dl mul add
4329:     -1 j dl mul add
4330:   } for
4331: } for
4332: ] def
4333:

```

```

4334:      /S6 [
4335:      1 1 n 1 sub {
4336:          /j exch def
4337:          1 1 n 1 sub {
4338:              /i exch def
4339:              -1
4340:              -1 i dl mul add
4341:              -1 j dl mul add
4342:          } for
4343:      } for
4344:      ] def
4345:
4346:      %% tableau des faces
4347:      /F1 [
4348:      0 1 n 1 sub {
4349:          /j exch def
4350:          0 1 n 1 sub {
4351:              /i exch def
4352:              [
4353:                  i n 1 add j mul add
4354:                  dup 1 add
4355:                  dup n 1 add add
4356:                  dup 1 sub
4357:              ]
4358:          } for
4359:      } for
4360:      ] def
4361:
4362:      %% syntaxe : i sommettourgauche --> l indice du i-eme sommet du tour
4363:      %% de la face gauche (en commençant par l indice 0). ATTENTION :
4364:      %% utilise la variable globale n = nb d etages
4365:      /sometttourgauche {
4366:          1 dict begin
4367:          /i exch def
4368:          i 4 n mul ge {
4369:              i
4370:              (Error: indice trop grand dans sommettourgauche) ==
4371:              exit
4372:          } if
4373:          n n 1 add i mul add
4374:          end
4375:      } def
4376:
4377:      %% syntaxe : i sommetcentregauche --> l indice du i-eme sommet du centre
4378:      %% de la face gauche (en commençant par l indice 0). ATTENTION :
4379:      %% utilise les variables globales n = nb d etages, et N = nb sommets
4380:      %% des 4 leres faces
4381:      /somettcentregauche {
4382:          1 dict begin
4383:          /i exch def
4384:          i n 1 sub dup mul ge {
4385:              i
4386:              (Error: indice trop grand dans sommetcentregauche) ==
4387:              exit
4388:          } if
4389:          N i add
4390:          end
4391:      } def
4392:
4393:      /F5 [
4394:      %%%% la face gauche %%%%
4395:      %% le coin superieur gauche
4396:      [
4397:          1 sommettourgauche
4398:          0 sommettourgauche
4399:          n 4 mul 1 sub sommettourgauche
4400:          n1 n 1 sub sub sommetcentregauche
4401:      ]
4402:
4403:      %% la bande superieure (i from 1 to n-2)
4404:      1 1 n 2 sub {
4405:          /i exch def
4406:          [
4407:              i 1 add sommettourgauche
4408:              i sommettourgauche
4409:              n1 n sub i add sommetcentregauche

```

```

4410:         n1 n sub i 1 add add sommetcentregauche
4411:     ]
4412: } for
4413:
4414: %% le coin superieur droit
4415: [
4416:     n sommettourgauche
4417:     n 1 sub sommettourgauche
4418:     n1 1 sub sommetcentregauche
4419:     n 1 add sommettourgauche
4420: ]
4421:
4422: %% la descente gauche
4423: %% j from 1 to n-2
4424: 1 1 n 2 sub {
4425:     /j exch def
4426:     [
4427:         n1 n 1 sub j mul sub sommetcentregauche
4428:         n 4 mul j sub sommettourgauche
4429:         n 4 mul j 1 add sub sommettourgauche
4430:         n1 n 1 sub j 1 add mul sub sommetcentregauche
4431:     ]
4432: } for
4433:
4434: %% les bandes centrales (j from 1 to n-2 et i from 1 to n-2)
4435: 1 1 n 2 sub {
4436:     /j exch def
4437:     1 1 n 2 sub {
4438:         /i exch def
4439:         [
4440:             n1 i n 1 sub j 1 sub mul add sub sommetcentregauche
4441:             n1 i 1 add n 1 sub j 1 sub mul add sub sommetcentregauche
4442:             n1 i 1 add n 1 sub j mul add sub sommetcentregauche
4443:             n1 i n 1 sub j mul add sub sommetcentregauche
4444:         ]
4445:     } for
4446: } for
4447:
4448: %% la descente droite
4449: 1 1 n 2 sub {
4450:     /j exch def
4451:     [
4452:         n j add sommettourgauche
4453:         n1 1 sub j 1 sub n 1 sub mul sub sommetcentregauche
4454:         n1 1 sub j n 1 sub mul sub sommetcentregauche
4455:         n j 1 add add sommettourgauche
4456:     ]
4457: } for
4458:
4459: %% le coin inferieur gauche
4460: [
4461:     0 sommetcentregauche
4462:     n 3 mul 1 add sommettourgauche
4463:     n 3 mul sommettourgauche
4464:     n 3 mul 1 sub sommettourgauche
4465: ]
4466:
4467: %% la bande inferieure (i from 1 to n-2)
4468: 1 1 n 2 sub {
4469:     /i exch def
4470:     [
4471:         i sommetcentregauche
4472:         i 1 sub sommetcentregauche
4473:         n 3 mul i sub sommettourgauche
4474:         n 3 mul i sub 1 sub sommettourgauche
4475:     ]
4476: } for
4477:
4478: %% le coin inferieur droit
4479: [
4480:     n 2 mul 1 sub sommettourgauche
4481:     n 2 sub sommetcentregauche
4482:     n 2 mul 1 add sommettourgauche
4483:     n 2 mul sommettourgauche
4484: ]
4485: ] def

```

```

4486:
4487:   %% syntaxe : i sommettourdroit --> l indice du i-eme sommet du tour
4488:   %% de la face droit (en commençant par l indice 0). ATTENTION :
4489:   %% utilise la variable globale n = nb d etages
4490:   /sometttourdroit {
4491:   l dict begin
4492:   /i exch def
4493:   i 4 n mul ge {
4494:     i
4495:     (Error: indice trop grand dans sommettourdroit) ==
4496:     exit
4497:   } if
4498:   n 1 add i mul
4499:   end
4500:   } def
4501:
4502:   %% syntaxe : i sommetcentredroit --> l indice du i-eme sommet du centre
4503:   %% de la face droit (en commençant par l indice 0). ATTENTION :
4504:   %% utilise les variables globales n = nb d etages, et N = nb sommets
4505:   %% des 4 leres faces
4506:   /somettcentredroit {
4507:   l dict begin
4508:   /i exch def
4509:   i n 1 sub dup mul ge {
4510:     i
4511:     (Error: indice trop grand dans sommetcentredroit) ==
4512:     exit
4513:   } if
4514:   N n1 add i add
4515:   end
4516:   } def
4517:
4518:   /F6 [
4519:   %% coin superieur droit
4520:   [
4521:     0 sommettourdroit
4522:     1 sommettourdroit
4523:     n1 n 1 sub sub sommetcentredroit
4524:     4 n mul 1 sub sommettourdroit
4525:   ]
4526:   %% coin superieur gauche
4527:   [
4528:     n 1 sub sommettourdroit
4529:     n sommettourdroit
4530:     n 1 add sommettourdroit
4531:     n1 1 sub sommetcentredroit
4532:   ]
4533:   %% coin inferieur gauche
4534:   [
4535:     n 2 sub sommetcentredroit
4536:     2 n mul 1 sub sommettourdroit
4537:     2 n mul sommettourdroit
4538:     2 n mul 1 add sommettourdroit
4539:   ]
4540:   %% coin inferieur droit
4541:   [
4542:     3 n mul 1 add sommettourdroit
4543:     0 sommetcentredroit
4544:     3 n mul 1 sub sommettourdroit
4545:     3 n mul sommettourdroit
4546:   ]
4547:   %% bande superieure
4548:   1 1 n 2 sub {
4549:     /i exch def
4550:     [
4551:       i sommettourdroit
4552:       i 1 add sommettourdroit
4553:       n 1 sub n 2 sub mul i add sommetcentredroit
4554:       n 1 sub n 2 sub mul i 1 sub add sommetcentredroit
4555:     ]
4556:   } for
4557:   %% bande inferieure
4558:   1 1 n 2 sub {
4559:     /i exch def
4560:     [
4561:       i 1 sub sommetcentredroit

```

```

4562:         i sommetcentredroit
4563:         3 n mul 1 sub i sub sommettourdroit
4564:         3 n mul i sub sommettourdroit
4565:     ]
4566: } for
4567: %% descente gauche
4568: 1 1 n 2 sub {
4569:     /i exch def
4570:     [
4571:         n1 1 sub i 1 sub n 1 sub mul sub sommetcentredroit
4572:         n i add sommettourdroit
4573:         n i 1 add add sommettourdroit
4574:         n1 1 sub i n 1 sub mul sub sommetcentredroit
4575:     ]
4576: } for
4577: %% descente droite
4578: 1 1 n 2 sub {
4579:     /i exch def
4580:     [
4581:         4 n mul i sub sommettourdroit
4582:         n 1 sub n 1 sub i sub mul sommetcentredroit
4583:         n 1 sub n 2 sub i sub mul sommetcentredroit
4584:         4 n mul i sub 1 sub sommettourdroit
4585:     ]
4586: } for
4587: %% bandes interieures
4588: 1 1 n 2 sub {
4589:     /j exch def
4590:     1 1 n 2 sub {
4591:         /i exch def
4592:         [
4593:             n 1 sub j mul i 1 sub add sommetcentredroit
4594:             n 1 sub j mul i add sommetcentredroit
4595:             n 1 sub j 1 sub mul i add sommetcentredroit
4596:             n 1 sub j 1 sub mul i 1 sub add sommetcentredroit
4597:         ]
4598:     } for
4599: } for
4600:
4601: ] def
4602:
4603: /F2 F1 {{n dup mul n add add} apply} apply def
4604: /F3 F2 {{n dup mul n add add} apply} apply def
4605: /F4 F3 {{n dup mul n add add} apply} apply def
4606:
4607:
4608: S1 S2 append S3 append S4 append S5 append S6 append {a mulv3d} papply3d
4609: F1 F2 append F3 append F4 append {{N mod} apply} apply F5 append F6 append
4610: generesolid
4611: } ifelse
4612: end
4613: } def
4614:
4615: %%%% ### newparallelepiped ###
4616: % 14 octobre 2006
4617: /newparallelepiped {
4618: 2 dict begin
4619:     /c exch 2 div def
4620:     /b exch 2 div def
4621:     /a exch 2 div def
4622:     /F [
4623:         [0 1 2 3]
4624:         [0 4 5 1]
4625:         [1 5 6 2]
4626:         [2 6 7 3]
4627:         [0 3 7 4]
4628:         [4 7 6 5]
4629:     ] def
4630:
4631:     %% tableau des sommets
4632:     /S [
4633:         a      b      c %% 0
4634:         a neg b      c %% 1
4635:         a neg b neg c %% 2
4636:         a      b neg c %% 3
4637:         a      b      c neg %% 4

```



```

4638:      a neg b      c neg %% 5
4639:      a neg b neg c neg %% 6
4640:      a      b neg c neg %% 7
4641:    ] def
4642:    S F generesolid
4643:  } def
4644:
4645:  %%%% ### newcylindre ###
4646:  %% syntaxe : z0 r0 z1 newcylindre -> solide
4647:  /newcylindre {
4648:    dup xcheck {
4649:      2 index exch
4650:    } {
4651:      dup isarray {
4652:        2 index exch
4653:      } {
4654:        1 index
4655:      } ifelse
4656:    } ifelse
4657:    newtronccone
4658:  } def
4659:
4660:  %% syntaxe : z0 r0 z1 newcylindrecreux -> solide
4661:  /newcylindrecreux {
4662:    newcylindre
4663:    dup creusesolid
4664:  } def
4665:
4666:  %%%% ### newtronccone ###
4667:  %% syntaxe : z0 r0 z1 r1 newtronccone -> solid
4668:  /newtronccone {
4669:    11 dict begin
4670:      [[/n /N] [1 6] [1 8] [1 10] [3 12] [5 18]] gestionsolidmode
4671:
4672:      /r1 exch def
4673:      /z1 exch def
4674:      /r0 exch def
4675:      /z0 exch def
4676:      /dz z1 z0 sub n div def
4677:      /dr r1 r0 sub n div def
4678:
4679:      /FE [
4680:        [0 1 N 1 sub {} for]
4681:        [n 1 add N mul 1 sub -1 n N mul {} for]
4682:
4683:        0 1 n 1 sub {
4684:          /k exch def
4685:          k N mul 1 add 1 k 1 add N mul 1 sub {
4686:            /i exch def
4687:            [i i 1 sub N i add 1 sub N i add]
4688:          } for
4689:          [k N mul k 1 add N mul 1 sub k 2 add N mul 1 sub k 1 add N mul]
4690:        } for
4691:
4692:      ] def
4693:
4694:      %% tableau des sommets
4695:      /S [
4696:        n -1 0 {
4697:          /k exch def
4698:          0 1 N 1 sub {
4699:            /i exch def
4700:            360 N idiv i mul cos r0 dr k mul add mul
4701:            360 N idiv i mul sin r0 dr k mul add mul
4702:            z0 dz k mul add
4703:          } for
4704:        } for
4705:      ] def
4706:    S FE generesolid
4707:  end
4708: } def
4709:
4710: %% syntaxe : z0 r0 z1 r1 newtroncconecreux -> solid
4711: /newtroncconecreux {
4712:   newtronccone
4713:   dup creusesolid

```

```

4714: } def
4715:
4716: %%%% ### newcone ###
4717: %% syntaxe : z0 r0 z1 newcone -> solid
4718: /newcone {
4719: 11 dict begin
4720:   [ [/n /N] [1 6] [1 8] [1 10] [3 12] [5 18] ] gestionsolidmode
4721:
4722:   /z1 exch def
4723:   /r0 exch def
4724:   /z0 exch def
4725:   /dz z1 z0 sub n div def
4726:   /dr r0 n div def
4727:
4728:   /F [
4729:     %% la base
4730:     [N 1 sub -1 0 {}] for]
4731:     %% le dernier etage
4732:     n 1 sub N mul 1 add 1 n N mul 1 sub {
4733:       /i exch def
4734:       [i 1 sub i n N mul]
4735:     } for
4736:     [n N mul 1 sub n 1 sub N mul n N mul]
4737:     %% les autres etages
4738:     0 1 n 2 sub {
4739:       /j exch def
4740:       0 N j mul add 1 N N j mul add 2 sub {
4741:         /i exch def
4742:         [i i 1 add dup N add dup 1 sub]
4743:       } for
4744:       [N N j mul add 1 sub N j mul dup N add dup N add 1 sub]
4745:     } for
4746:   ] def
4747:
4748:   %% tableau des sommets
4749:   /S [
4750:     %% etage no j (in [1; n])
4751:     0 1 n 1 sub {
4752:       /j exch def
4753:       0 1 N 1 sub {
4754:         /i exch def
4755:         360 N idiv i mul cos r0 dr j mul sub mul
4756:         360 N idiv i mul sin r0 dr j mul sub mul
4757:         z0 dz j mul add
4758:       } for
4759:     } for
4760:     0 0 z1
4761:   ] def
4762:   S F generesolid
4763: end
4764: } def
4765:
4766: %% syntaxe : z0 r0 z1 newconecreux -> solid
4767: /newconecreux {
4768:   newcone
4769:   dup 0 solidrmface
4770:   dup videsolid
4771: } def
4772:
4773: %%%% ### newtore ###
4774: %% syntaxe : r R newtore -> solid
4775: /newtore {
4776: 10 dict begin
4777:   [[/n1 /n2] [4 5] [6 10] [8 12] [9 18] [18 36]] gestionsolidmode
4778:   /n2 n2 3 max store
4779:   /n1 n1 2 max store
4780:   /R exch def
4781:   /r exch def
4782:   /S [
4783:     0 1 n1 1 sub {
4784:       /i exch def
4785:       360 n1 div i mul cos r mul R add
4786:       360 n1 div i mul sin r mul
4787:     } for
4788:   ]
4789:   def

```

```

4790:   S [n2] newanneau
4791: end
4792: } def
4793:
4794: %%%% ### newprisme ###
4795: /newprismedroit {
4796:   [[/N] [1] [1] [1] [3] [6]] gestionsolidmode
4797:   0 0 1 [N] newprisme
4798: } def
4799:
4800: %% syntaxe : array N z0 z1 u newprisme -> solid d axe (O, u),
4801: %% ou array tableau de points 2d
4802: /newprisme {
4803: 7 dict begin
4804:   [[/N] [1] [1] [1] [3] [6]] gestionsolidmode
4805:   dup 0 eq {
4806:     (Error : 3eme composante nulle dans le vecteur pour newprisme) ==
4807:     quit
4808:   } if
4809:   /u defpoint3d
4810:   /z1 exch def
4811:   /z0 exch def
4812:   %% N = nb d etages
4813:   /table exch def
4814:   %% n = indice du dernier point
4815:   /n table length 2 idiv 1 sub def
4816:   %% vecteur de translation
4817:   u
4818:   z1 z0 sub u norme3d div
4819:   mulv3d /v defpoint3d
4820:
4821:   %% tableau des sommets
4822:   /S [
4823:     0 1 N {
4824:       /j exch def
4825:       0 1 n {
4826:         /i exch def
4827:         table i getp
4828:         z0
4829:         v N j sub N div mulv addv3d
4830:       } for
4831:     } for
4832:   ] def
4833:
4834:   /F [
4835:     %% face superieure
4836:     [0 1 n {} for]
4837:     %% base
4838:     [N 1 add n 1 add mul 1 sub -1 N n 1 add mul {} for]
4839:     %% faces etage
4840:     1 1 N {
4841:       /j exch def
4842:       1 1 n {
4843:         /i exch def
4844:         [i                j 1 sub n 1 add mul add
4845:          i 1 sub          j 1 sub n 1 add mul add
4846:          n 1 add i add 1 sub j 1 sub n 1 add mul add
4847:          n 1 add i add    j 1 sub n 1 add mul add]
4848:       } for
4849:       [0                j 1 sub n 1 add mul add
4850:        n                j 1 sub n 1 add mul add
4851:        2 n mul 1 add    j 1 sub n 1 add mul add
4852:        n 1 add          j 1 sub n 1 add mul add]
4853:     } for
4854:   ] def
4855:
4856:   S F generesolid
4857: end
4858: } def
4859:
4860: %%%% ### newsphere ###
4861: %% syntaxe : r option newsphere -> solid
4862: /newsphere {
4863: 2 dict begin
4864:   [[/K /N] [6 6] [8 8] [10 12] [16 12] [16 36]] gestionsolidmode
4865:   -90 90 [K N] newcalottesphere

```

```

4866: end
4867: } def
4868:
4869: %% syntaxe : r phi theta option newcalottesphere -> solid
4870: /newcalottesphere {
4871: 6 dict begin
4872:   [[/K /N] [6 6] [8 8] [10 12] [16 12] [16 36]] gestionsolidmode
4873:
4874:   %% test de beta (ex-theta)
4875:   dup 90 eq {
4876:     /beta exch def
4877:     /idebut 1 def
4878:   } {
4879:     /beta exch 80 min -80 max def
4880:     /idebut 0 def
4881:   } ifelse
4882:   %% test de alpha (ex-phi)
4883:   dup -90 eq {
4884:     /alpha exch def
4885:   } {
4886:     /alpha exch beta min -80 max def
4887:   } ifelse
4888:   /r exch def
4889:   beta 90 eq {
4890:     alpha -90 eq {
4891:       /ifin K def
4892:       /db alpha beta sub K 1 add div def
4893:     } {
4894:       /ifin K def
4895:       /db alpha beta sub K div def
4896:     } ifelse
4897:   } {
4898:     alpha -90 eq {
4899:       /ifin K 1 sub def
4900:       /db alpha beta sub K div def
4901:     } {
4902:       /ifin K 1 sub def
4903:       /db alpha beta sub K 1 sub div def
4904:     } ifelse
4905:   } ifelse
4906:
4907:   %% nombre de sommets -2
4908:   /nb N K mul def
4909:
4910:   %% tableau des sommets
4911:   /S [
4912:     idebut 1 ifin {
4913:       /j exch def
4914:       /phi beta j db mul add def
4915:       phi cos r mul /r_tmp exch def
4916:       0 1 N 1 sub {
4917:         /i exch def
4918:         360 N idiv i mul cos r_tmp mul
4919:         360 N idiv i mul sin r_tmp mul
4920:         phi sin r mul
4921:       } for
4922:     } for
4923:     0 0 r neg
4924:     0 0 r
4925:   ] def
4926:
4927:   /F [
4928:     %% calotte inferieure
4929:     alpha -90 eq {
4930:       1 1 N 1 sub {
4931:         /i exch def
4932:         [
4933:           nb
4934:           nb i sub
4935:           nb i 1 add sub
4936:         ]
4937:       } for
4938:       [nb nb N sub nb 1 sub]
4939:     } {
4940:       [nb 1 sub -1 nb N sub {} for ]
4941:     } ifelse

```

```

4942:
4943:   %% calotte superieure
4944:   beta 90 eq {
4945:     0 1 N 1 sub {
4946:       /i exch def
4947:       [i i 1 add N mod N K mul 1 add]
4948:     } for
4949:   } {
4950:     [0 1 N 1 sub {} for]
4951:   } ifelse
4952:
4953:   1 1 K 1 sub {
4954:     /j exch def
4955:     [
4956:       j N mul
4957:       j N mul 1 add
4958:       j 1 sub N mul 1 add
4959:       j 1 sub N mul
4960:     ]
4961:     N 2 sub {dup {1 add} apply} repeat
4962:     [
4963:       j 1 add N mul 1 sub
4964:       j N mul
4965:       j 1 sub N mul
4966:       j N mul 1 sub
4967:     ]
4968:   } for
4969: ] def
4970:
4971: S F generesolid
4972: end
4973: } def
4974:
4975: %% syntaxe : r phi theta option newcalottespherecrease -> solid
4976: /newcalottespherecrease {
4977: 6 dict begin
4978:   [[/K /N] [6 6] [8 8] [10 12] [16 12] [16 36]] gestionsolidmode
4979:
4980:   %% test de beta (ex-theta)
4981:   dup 90 eq {
4982:     /beta exch def
4983:     /idebut 1 def
4984:   } {
4985:     /beta exch 80 min -80 max def
4986:     /idebut 0 def
4987:   } ifelse
4988:   %% test de alpha (ex-phi)
4989:   dup -90 eq {
4990:     /alpha exch def
4991:   } {
4992:     /alpha exch beta min -80 max def
4993:   } ifelse
4994:   /r exch def
4995:   beta 90 eq {
4996:     alpha -90 eq {
4997:       /ifin K def
4998:       /db alpha beta sub K 1 add div def
4999:     } {
5000:       /ifin K def
5001:       /db alpha beta sub K div def
5002:     } ifelse
5003:   } {
5004:     alpha -90 eq {
5005:       /ifin K 1 sub def
5006:       /db alpha beta sub K div def
5007:     } {
5008:       /ifin K 1 sub def
5009:       /db alpha beta sub K 1 sub div def
5010:     } ifelse
5011:   } ifelse
5012:
5013:   %% nombre de sommets -2
5014:   /nb N K mul def
5015:
5016:   %% tableau des sommets
5017:   /S [

```

```

5018:      idebut 1 ifin {
5019:          /j exch def
5020:          /phi beta j db mul add def
5021:          phi cos r mul /r_tmp exch def
5022:          0 1 N 1 sub {
5023:              /i exch def
5024:              360 N idiv i mul cos r_tmp mul
5025:              360 N idiv i mul sin r_tmp mul
5026:              phi sin r mul
5027:          } for
5028:      } for
5029:      0 0 r neg
5030:      0 0 r
5031:  ] def
5032:
5033:  /F [
5034:      %% calotte inferieure
5035:      alpha -90 eq {
5036:          1 1 N 1 sub {
5037:              /i exch def
5038:              [
5039:                  nb
5040:                  nb i sub
5041:                  nb i 1 add sub
5042:              ]
5043:          } for
5044:          [nb nb N sub nb 1 sub]
5045:      } {
5046:  %      [nb 1 sub -1 nb N sub {} for ]
5047:  } ifelse
5048:
5049:      %% calotte superieure
5050:      beta 90 eq {
5051:          0 1 N 1 sub {
5052:              /i exch def
5053:              [i i 1 add N mod N K mul 1 add]
5054:          } for
5055:      } {
5056:  %      [0 1 N 1 sub {} for]
5057:  } ifelse
5058:
5059:      1 1 K 1 sub {
5060:          /j exch def
5061:          [
5062:              j N mul
5063:              j N mul 1 add
5064:              j 1 sub N mul 1 add
5065:              j 1 sub N mul
5066:          ]
5067:          N 2 sub {dup {1 add} apply} repeat
5068:          [
5069:              j 1 add N mul 1 sub
5070:              j N mul
5071:              j 1 sub N mul
5072:              j N mul 1 sub
5073:          ]
5074:      } for
5075:  ] def
5076:
5077:  S F generesolid
5078:  dup videsolid
5079: end
5080: } def
5081:
5082: %%%% ### newanneau ###
5083: %% syntaxe : array n newanneau --> solid
5084: %% syntaxe : array {mode} newanneau --> solid
5085: %% ou array est un tableau de points de R^2 et n un nombre entier positif
5086: /newanneau {
5087: 10 dict begin
5088:   dup isnum {
5089:     /n exch def
5090:     [n]
5091:   } if
5092:   [[/n2] [6] [12] [24] [32] [36]] gestionsolidmode
5093:   /n2 n2 3 max store

```

```

5094:    %% on plonge la section dans R^3 par projection sur yOz
5095:    /S1 exch {0 3 1 roll} papply def
5096:    %% nombre de sommets
5097:    /n1 S1 length 3 idiv def
5098:
5099:    /S S1
5100:        n2 {
5101:            duparray
5102:            {0 0 360 n2 div rotateOpoint3d} papply3d
5103:        } repeat
5104:        n2 {append} repeat
5105:    def
5106:
5107:    /F [
5108:        0 1 n2 1 sub {
5109:            /j exch def
5110:            n1 j mul 1 j 1 add n1 mul 2 sub {
5111:                /i exch def
5112:                [i 1 add i dup n1 add i n1 1 add add]
5113:            } for
5114:            [n1 j mul j 1 add n1 mul 1 sub j 2 add n1 mul 1 sub j 1 add n1 mul]
5115:        } for
5116:    ] def
5117:
5118:    S F generesolid
5119: end
5120: } def
5121:
5122: %%%% ### newvecteur ###
5123: %% syntaxe : x y z newvecteur
5124: /newvecteur {
5125: 4 dict begin
5126:   /A defpoint3d
5127:   %%Sommets
5128:   /S [0 0 0 A] def
5129:   /F [
5130:     [0 1]
5131:   ] def
5132:   S F generesolid
5133: %%   /axe exch def
5134:   [ A ]
5135:   normalvect_to_orthobase
5136:   /imK defpoint3d
5137:   /imJ defpoint3d
5138:   /imI defpoint3d
5139:
5140:   A norme3d /z exch .3 sub def
5141:   0 .1 .3 [1 8] newcone
5142:   dup (noir) outputcolors
5143:   {0 0 z translatepoint3d} solidtransform
5144:   {imI imJ imK transformpoint3d} solidtransform
5145:   solidfuz
5146: end
5147: } def
5148:
5149: %%%% ### newobjfile ###
5150: /newobjfile {
5151: 3 dict begin
5152:   /objfilename exch def
5153:   /v {} def
5154:   /ok true def
5155:   /f {
5156:     ok {
5157:       %% lere fois
5158:       ] %% ferme les sommets
5159:       [ [ %% ouvre les faces
5160:         /ok false store
5161:       ] {
5162:         %% les autres fois
5163:         ] %% ferme la face
5164:         [ %% ouvre la nouvelle
5165:       } ifelse
5166:     } def
5167:     [ 0 0 0
5168:     objfilename run
5169:     ] ]

```

```

5170:      /F exch def
5171:      /S exch def
5172:
5173:      S F generesolid
5174:      %   dup videsolid
5175:    end
5176:  } def
5177:
5178:  %%%% ### fin insertion ###
5179:
5180:  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5181:  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5182:  %%%%                                     %%%%
5183:  %%%%           fin insertion librairie jps           %%%%
5184:  %%%%                                     %%%%
5185:  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5186:  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5187:

```

## 9. Gestion des chaînes de caractère

```

5188:  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5189:  %%%%           gestion de chaine de caracteres           %%%%
5190:  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5191:
5192:  /Times-Roman findfont
5193:  dup length dict begin
5194:    {
5195:      1 index /FID ne
5196:      {def}
5197:      {pop pop}
5198:      ifelse
5199:    } forall
5200:    /Encoding ISOLatin1Encoding def
5201:    currentdict
5202:  end
5203:  /Times-Roman-ISOLatin1 exch definefont pop
5204:
5205:  /setTimesRoman {
5206:    /Times-Roman-ISOLatin1 findfont
5207:    fontsize scalefont
5208:    setfont
5209:  } def
5210:
5211:  /setTimes {
5212:    setTimesRoman
5213:  } def
5214:
5215:  %% syntaxe : string x y cctext
5216:  /cctext {
5217:    5 dict begin
5218:      /y exch def
5219:      /x exch def
5220:      /str exch def
5221:      str stringwidth
5222:      /wy exch def
5223:      /wx exch def
5224:      gsave
5225:        x y smoveto
5226:        wx -2 div wy -2 div rmoveto
5227:        str show
5228:      grestore
5229:    end
5230:  } def
5231:
5232:  %% syntaxe : str x y show_dim --> str x y llx lly wx wy
5233:  %% attention, doit laisser la pile intacte
5234:  /show_dim {
5235:    3 copy pop pop
5236:    newpath
5237:    0 0 moveto

```



```
5238:      true charpath flattenpath pathbbox
5239:      closepath
5240:      newpath
5241:    } def
5242:
```

## 10. Interfaçage avec PSTricks

### 10.1 - Interface pour la macro psSolid

..... Le fichier solides.pro .....

```
5243: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5244: %%%      procedures pour PSTricks      %%%
5245: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5246:
5247: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5248: %%%      procedures pour \psSolid      %%%
5249: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5250:
5251: /all (all) def
5252:
5253: /draw {drawsolid} def
5254: /draw* {drawsolid*} def
5255: /draw** {drawsolid**} def
5256: /none {pop} def
5257:
5258: /gere_pstricks_color_inout {
5259:   gsave
5260:     dup [fillincolor] (setrgbcolor) astr2str
5261:     [fillcolor] (setrgbcolor) astr2str inoutoutputcolors
5262:   grestore
5263: } def
5264:
5265: /gere_pstricks_color_out {
5266:   gsave
5267:     dup [fillcolor] (setrgbcolor) astr2str outputcolors
5268:   grestore
5269: } def
5270:
5271: /gere_pstricks_opt {
5272: %   /CourbeR2 {CourbeR2+} def
5273:   linecolor
5274:   solidlinewidth setlinewidth
5275:   RotX 0 ne RotY 0 ne or RotZ 0 ne or {
5276:     {RotX RotY RotZ rotateOpoint3d} solidtransform
5277:   } if
5278:   CX 0 ne CY 0 ne or CZ 0 ne or {
5279:     {CX CY CZ translatepoint3d} solidtransform
5280:   } if
5281:   /rmfaces rmfaces bubblesort reverse store
5282:   0 1 rmfaces length 1 sub {
5283:     /i exch def
5284:     dup rmfaces i get solidrmface
5285:   } for
5286:   solidhollow {
5287:     dup videsolid
5288:   } if
5289:   activationgestioncouleurs {
5290:     dup solidwithinfaces {
5291:       gere_pstricks_color_inout
5292:     } {
5293:       gere_pstricks_color_out
5294:     } ifelse
5295:   } if
5296:
5297:   0 1 fcol length 2 idiv 1 sub {
5298:     /i exch def
5299:     dup fcol 2 i mul get fcol 2 i mul 1 add get solidputfcolor
5300:   } for
5301:   tx@Dict /pst-transformoption known {
5302:     dup {pst-transformoption} solidtransform
```

```

5303:   } if
5304:   solidinouthue length 0 gt {
5305:     dup solidinouthue solidputinouthuecolors
5306:   } {
5307:     solidhue length 0 gt {
5308:       dup solidhue solidputhuecolors
5309:     } if
5310:     solidinhue length 0 gt {
5311:       dup solidinhue solidputinhuecolors
5312:     } if
5313:   } ifelse
5314:   dup action
5315:   noir
5316:   solidnumf length 0 ne {
5317:     solidnumf 0 get isstring {
5318:       dup projectionsifacevisible solidnumfaces
5319:     } {
5320:       dup solidnumf projectionsifacevisible solidnumfaces
5321:     } ifelse
5322:   } if
5323:   solidshow length 0 ne {
5324:     solidshow 0 get isstring {
5325:       dup solidshowsummits
5326:     } {
5327:       dup solidshow solidshowsummits
5328:     } ifelse
5329:   } if
5330:   solidnum length 0 ne {
5331:     solidnum 0 get isstring {
5332:       dup solidnumsummits
5333:     } {
5334:       dup solidnum solidnumsummits
5335:     } ifelse
5336:   } {
5337:     %% pop
5338:   } ifelse
5339:   tx@Dict /solidname known {
5340:     solidname exch bind def
5341:     tx@Dict /solidname undef
5342:   } {
5343:     pop
5344:   } ifelse
5345: } def
5346:
5347: /pst-octahedron {
5348:   a newoctaedre
5349:   gere_pstricks_opt
5350: } def
5351:
5352: /pst-dodecahedron {
5353:   a newdodecaedre
5354:   gere_pstricks_opt
5355: } def
5356:
5357: /pst-icosahedron {
5358:   a newicosaedre
5359:   gere_pstricks_opt
5360: } def
5361:
5362: /pst-cube {
5363:   a
5364:   ngrid length 1 eq {
5365:     ngrid
5366:   } {
5367:     {Mode}
5368:   } ifelse
5369:   newcube
5370:   %% solidhollow {
5371:   %%   dup videsolid
5372:   %%   } if
5373:   gere_pstricks_opt
5374: } def
5375:
5376: /pst-parallelepiped {
5377:   a b c
5378:   newparallelepiped

```

```
5379:   gere_pstricks_opt
5380: } def
5381:
5382: /pst-tetrahedron {
5383:   r newtetraedre
5384:   gere_pstricks_opt
5385: } def
5386:
5387: /pst-tore {
5388:   r0 r1
5389:   ngrid length 2 eq {
5390:     ngrid
5391:   } {
5392:     {Mode}
5393:   } ifelse
5394:   newtore
5395:   gere_pstricks_opt
5396: } def
5397:
5398: /pst-sphere {
5399:   % rayon
5400:   % mode
5401:   % r {Mode} newsphere
5402:   r
5403:   ngrid length 2 eq {
5404:     ngrid
5405:   } {
5406:     {Mode}
5407:   } ifelse
5408:   newsphere
5409:   gere_pstricks_opt
5410: } def
5411:
5412: /pst-cylindre {
5413:   % rayon
5414:   % mode
5415:   0 r h
5416:   ngrid length 2 eq {
5417:     ngrid
5418:   } {
5419:     {Mode}
5420:   } ifelse
5421:   newcylindre
5422:   solidhollow {
5423:     dup creusesolid
5424:   } if
5425:   gere_pstricks_opt
5426: } def
5427:
5428: /pst-cylindrecreux {
5429:   % rayon
5430:   % mode
5431:   0 r h
5432:   ngrid length 2 eq {
5433:     ngrid
5434:   } {
5435:     {Mode}
5436:   } ifelse
5437:   newcylindre
5438:   dup creusesolid
5439:   gere_pstricks_opt
5440: } def
5441:
5442: /pst-cone {
5443:   % rayon
5444:   % mode
5445:   0 r h
5446:   ngrid length 2 eq {
5447:     ngrid
5448:   } {
5449:     {Mode}
5450:   } ifelse
5451:   solidhollow {
5452:     newcone
5453:   } {
5454:     newcone
```

```

5455:     } ifelse
5456:     gere_pstricks_opt
5457: } def
5458:
5459: /pst-tronccone {
5460:     % rayon
5461:     % mode
5462:     0 r0 h r1
5463:     ngrid length 2 eq {
5464:         ngrid
5465:     } {
5466:         {Mode}
5467:     } ifelse
5468:     solidhollow {
5469:         newtroncconecreux
5470:     } {
5471:         newtronccone
5472:     } ifelse
5473:     gere_pstricks_opt
5474: } def
5475:
5476: /pst-troncconecreux {
5477:     % rayon
5478:     % mode
5479:     0 r0 h r1
5480:     ngrid length 2 eq {
5481:         ngrid
5482:     } {
5483:         {Mode}
5484:     } ifelse
5485:     newtroncconecreux
5486:     gere_pstricks_opt
5487: } def
5488:
5489: /pst-cone {
5490:     % rayon
5491:     % mode
5492:     0 r h
5493:     ngrid length 2 eq {
5494:         ngrid
5495:     } {
5496:         {Mode}
5497:     } ifelse
5498:     newcone
5499:     gere_pstricks_opt
5500: } def
5501:
5502: /pst-anneau {
5503:     [ section ]
5504:     ngrid length 1 ge {
5505:         [ngrid 0 get]
5506:     } {
5507:         [24]
5508:     } ifelse
5509:     newanneau
5510:     gere_pstricks_opt
5511: } def
5512:
5513:
5514: /pst-prisme {
5515:     % tableau des points de la base
5516:     % h hauteur du prisme
5517:     % axe : vecteur direction de l axe
5518:     base decal rollparray
5519:     0 h axe
5520:     ngrid length 1 ge {
5521:         [ngrid 0 get]
5522:     } if
5523:     newprisme
5524:     solidhollow {
5525:         dup creusesolid
5526:     } if
5527:     gere_pstricks_opt
5528: } def
5529:
5530: /pst-prismecreux {

```

```

5531: % tableau des points de la base
5532: % h hauteur du prisme
5533: % axe : vecteur direction de l axe
5534: base
5535: 0 h axe
5536: ngrid length 1 ge {
5537:   [ngrid 0 get]
5538: } if
5539: newprisme
5540: dup creusesolid
5541: gere_pstricks_opt
5542: } def
5543:
5544: /pst-grille {
5545:   base aload pop
5546:   ngrid length 2 ge {
5547:     [ngrid 0 get ngrid 1 get]
5548:   } {
5549:     ngrid length 1 eq {
5550:       [ngrid 0 get dup]
5551:     } if
5552:   } ifelse
5553:   newgrille
5554:   gere_pstricks_opt
5555: } def
5556:
5557: %% syntaxe : array N h u newruban -> solid d axe (O, u),
5558: /pst-ruban {
5559:   % tableau des points de la base
5560:   % h hauteur du prisme
5561:   % axe : vecteur direction de l axe
5562:   base
5563:   h axe
5564:   ngrid length 1 ge {
5565:     [ngrid 0 get]
5566:   } if
5567:   newruban
5568:   gere_pstricks_opt
5569: } def
5570:
5571: %% syntaxe : r phi option newcalottesphere -> solid
5572: /pst-calottesphere {
5573:   % rayon
5574:   % mode
5575:   % r phi theta option newcalottesphere
5576:   r
5577:   phi theta
5578:   ngrid length 2 eq {
5579:     ngrid
5580:   } {
5581:     {Mode}
5582:   } ifelse
5583:   solidhollow {
5584:     newcalottespherecrease
5585:   } {
5586:     newcalottesphere
5587:   } ifelse
5588:   gere_pstricks_opt
5589: } def
5590:
5591: %% syntaxe : r phi option newcalottesphere -> solid
5592: /pst-calottespherecrease {
5593:   % rayon
5594:   % mode
5595:   % r phi theta option newcalottespherecrease
5596:   r
5597:   phi theta
5598:   ngrid length 2 eq {
5599:     ngrid
5600:   } {
5601:     {Mode}
5602:   } ifelse
5603:   newcalottespherecrease
5604:   gere_pstricks_opt
5605: } def
5606:

```

```

5607: /pointtest{2 2 2} def
5608:
5609: /pst-face {
5610:   % tableau des points de la base
5611:   % h hauteur du prisme
5612:   % axe : vecteur direction de l axe
5613:   base
5614:   solidbiface {
5615:     newbiface
5616:   } {
5617:     newmonoface
5618:   } ifelse
5619:   gere_pstricks_opt
5620: } def
5621:
5622: /pst-surface {
5623:   base
5624:   base aload pop
5625:   ngrid length 2 ge {
5626:     [ngrid 0 get ngrid 1 get]
5627:   } {
5628:     ngrid length 1 eq {
5629:       [ngrid 0 get dup]
5630:     } ifelse
5631:   } ifelse
5632:   {f} newsurface
5633:   solidbiface {
5634:     dup videsolid
5635:   } if
5636:   gere_pstricks_opt
5637: } def
5638:
5639: /pst-polygoneregulier {
5640:   r ngrid 0 get
5641:   newpolreg
5642:   solidbiface {
5643:     } {
5644:     dup 1 solidrmface
5645:   } ifelse
5646:   gere_pstricks_opt
5647: } def
5648:
5649: /pst-fusion {
5650: 1 dict begin
5651:   /activationgestioncouleurs false def
5652:   /n base length def
5653:   base aload pop n 1 sub {solidfuz} repeat
5654:   gere_pstricks_opt
5655: end
5656: } def
5657:
5658: /pst-new {
5659:   sommets faces
5660:   generesolid
5661: %%   solidhollow {
5662: %%     dup videsolid
5663: %%   } if
5664:   gere_pstricks_opt
5665: } def
5666:
5667: /pst-courbe {
5668:   solidlinewidth setlinewidth
5669:   range aload pop {function} Courber3
5670: } def
5671:
5672: /pst-surfaceparametree {
5673:   base aload pop
5674:   ngrid length 2 ge {
5675:     [ngrid 0 get ngrid 1 get]
5676:   } {
5677:     ngrid length 1 eq {
5678:       [ngrid 0 get dup]
5679:     } if
5680:   } ifelse
5681:   { function } newsurfaceparametree
5682:   dup videsolid

```

```

5683:   gere_pstricks_opt
5684: } def
5685:
5686: /pst-vecteur {
5687: gsave
5688:   solidlinewidth setlinewidth
5689:   1 setlinejoin
5690:   1 setlinecap
5691:   linecolor
5692:   linestyle
5693:   args newvecteur
5694:   dup
5695:     [linecolor currentrgbcolor] ( ) astr2str (setrgbcolor) append
5696:   outputcolors
5697:   gere_pstricks_opt
5698: grestore
5699: } def
5700:
5701: /pst-ligne {
5702:   newpath
5703:     base 0 get
5704:     base 1 get
5705:     base 2 get
5706:     3dto2d smoveto
5707:     base ligne3d_
5708: } def
5709:
5710: /pst-objfile {
5711:   solidfilename newobjfile
5712:   % dup {1 1 div mulv3d} solidtransform
5713:   %% solidhollow {
5714:   %%   dup videsolid
5715:   %% } if
5716:   gere_pstricks_opt
5717: } def
5718:

```

## 10.2 - Interface pour la macro psProjection

```

5719: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5720: %%%      procedures pour \psProjection      %%%
5721: %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5722:
5723: /gere_pstricks_proj_opt {
5724:   /solidprojname where {
5725:     /solidprojname get noface phi
5726:     xorigine 0 eq
5727:     yorigine 0 eq and
5728:     zorigine 0 eq and
5729:     xorigine isinteger not and
5730:     yorigine isinteger not and
5731:     yorigine isinteger not and {
5732:       } {
5733:         [xorigine yorigine zorigine] (           ) astr2str
5734:       } ifelse
5735:     projectionsifacevisible solidprojpath
5736:   } {
5737:     xorigine yorigine zorigine [ normale ] projectionsifacevisible planprojpath
5738:   } ifelse
5739: } def
5740:
5741: /proj-pst-chemin {
5742:   solidlinewidth setlinewidth
5743:   newpath
5744:     path
5745:     linecolor
5746:     gere_pstricks_proj_opt
5747: } def
5748:
5749: /proj-pst-courbeR2 {
5750:   solidlinewidth setlinewidth

```

```

5751:   newpath
5752:     linecolor
5753:     range aload pop { function } CourbeR2_
5754:     gere_pstricks_proj_opt
5755: } def
5756:
5757: /proj-pst-courbe {
5758:   solidlinewidth setlinewidth
5759:   newpath
5760:     linecolor
5761:     range aload pop {} { function } Courbeparam_
5762:     gere_pstricks_proj_opt
5763: } def
5764:
5765: /proj-pst-texte {
5766: 2 dict begin
5767:   setTimes
5768:   solidlinewidth setlinewidth
5769:   newpath
5770:   linecolor
5771:   texte 0 0
5772:   pos (text_) append cvx exec
5773:   gere_pstricks_proj_opt
5774: fill
5775: end
5776: } def
5777:
5778: /pst-trigospherique {
5779: 3 dict begin
5780: gsave
5781:   solidlinewidth setlinewidth
5782:   linecolor
5783:   linestyle
5784:   args definition
5785: grestore
5786: end
5787: } def
5788:
5789: % END solides.pro
5790:

```