Le fichier *solides.pro*

6 Septembre 2006

1. Variables locales, dictionnaires

La langage postcript stocke ses variables dans des *dictionnaires* sous la forme de paires clé/valeur. Ainsi le code **`/A 2 def`** associe-t-il la clé */A* au nombre 2 dans le dictionnaire courant.

L'ouverture d'un nouveau dictionnaire, permet de « surcharger » un nom de variable. Le rejet de ce nouveau dictionnaire (avec un **`end`**) permettra de retrouver l'ancienne valeur de la variable surchargée : c'est la localisation des variables.

Un petit code en exemple :

```
/a 2 def
%% ici a vaut 2
1 dict begin
   /a 3 def
   %% ici a vaut 3
end
%% ici a vaut 2
```

Utiliser des variables locales permet d'éviter les conflits de noms de variables entre différentes parties d'un code.

2. la prodédure de dessin

Il y a 3 procédures de dessin : **`drawsolid`**, **`drawsolid*`** et **`drawsolid**`**.

2.1 - la prodédure `drawsolid*`

Elle se contente de positionner le switch *startest* à *true* pour indiquer que l'on veut le coloriage des faces, puis elle passe la main à *drawsolid*.

la prodédure **`drawsolid*`**

```
1245 : /drawsolid* {
1246 : 1 dict begin
1247 :    /startest {true} def
1248 :    drawsolid
1249 : end
1250 : } def
```

2.2 - la prodédure `drawsolid**`

Elle positionne le switch *aretescachees* à *false* pour indiquer que l'on ne veut pas le tracé des arêtes cahées, puis elle positionne le switch *peintrealgorithme* à *true*, puis elle passe la main à *drawsolid∗*.

la prodédure **`drawsolid**`**

```
1254 : /drawsolid** {
1255 : 2 dict begin
1256 :    /aretescachees false def
1257 :    /peintrealgorithme true def
1258 :    drawsolid*
1259 : end
1260 : } def
```

2.3 - la prodédure `drawsolid`

Le listing complet :

la prodédure **drawsolid**

```
1319 : %% syntaxe : solid drawsolid
1320 : /drawsolid {
1321 : 7 dict begin
1322 :    /solid exch def
1323 :    solid issolid not {
1324 :       (Error : mauvais type d'argument dans drawsolid) ==
1325 :       quit
1326 :    } if
1327 :    solid emptysolid not {
1328 :       solid solidgetfaces
1329 :       /F exch def
1330 :       solid solidgetpointstable
1331 :       /S exch def
1332 :       /n S length 3 idiv def
1333 :       %% tableau des aretes
1334 :       /A [
1335 :          n {
1336 :             [n {false} repeat]
1337 :          } repeat
1338 :       ] def
1339 :
1340 :       peintrealgorithme {
1341 :          %% tri des indices des faces par distance decroissante
1342 :          [
1343 :             0 1 F length 1 sub {
1344 :                /i exch def
1345 :                solid i solidcentreface
1346 :                getpointVue
1347 :                distance3d
1348 :             } for
1349 :          ] doublebubblesort pop reverse
1350 :       } {
1351 :          [
1352 :             0 1 F length 1 sub {
1353 :             } for
1354 :          ]
1355 :       } ifelse
1356 :       /ordre exch def
1357 :
1358 :       0 1 F length 1 sub {
1359 :          /k exch def
1360 :          /i ordre k get def
1361 :          gsave
1362 :          A solid i dessinefacevisible
1363 :          grestore
1364 :       } for
1365 :       aretescachees {
1366 :          0 1 F length 1 sub {
1367 :             /k exch def
1368 :             /i ordre k get def
1369 :             A solid i dessinefacecachee
1370 :          } for
1371 :       } if
1372 : %%       %% si on veut repasser les traits des faces visibles
1373 : %%       0 1 F length 1 sub {
1374 : %%          /k exch def
1375 : %%          /i ordre k get def
1376 : %%          gsave
1377 : %%          1 dict begin
1378 : %%             /startest false def
1379 : %%             A solid i dessinefacevisible
1380 : %%          end
1381 : %%          grestore
1382 : %%       } for
1383 :    } if
1384 : end
```

Et les explications :

partiel

```
1322 :    /solid exch def
1323 :    solid issolid not {
1324 :       (Error : mauvais type d'argument dans drawsolid) ==
```

partiel (suite)

```
1325 :        quit
1326 :     } if
```

on stocke le solide passé en paramètre dans *solid*, puis on teste de le type de cet objet. Si ce n'est pas un type *solid*, on envoie un message d'erreur.

partiel

```
1327 :     solid emptysolid not {
```

si le solide est vide, on n'essaie pas de le dessiner.

partiel

```
1328 :        solid solidgetfaces
1329 :        /F exch def
1330 :        solid solidgetpointstable
1331 :        /S exch def
1332 :        /n S length 3 idiv def
1333 :        %% tableau des aretes
1334 :        /A [
1335 :           n {
1336 :              [n {false} repeat]
1337 :           } repeat
1338 :        ] def
```

on stocke dans F le tableau des faces du solide, et en S le tableau des sommets. L'entier n est égal au nombre de sommets du solide, et enfin A est une matrice (n, n) de booléens, initialisée à $false$ pour tous ses éléments. Cette matrice nous indiquera les arêtes déjà tracées : $A[i, j] = false \Longrightarrow$ l'arête (i, j) n'est pas tracée.

partiel

```
1340 :        peintrealgorithme {
1341 :           %% tri des indices des faces par distance decroissante
1342 :           [
1343 :              0 1 F length 1 sub {
1344 :                 /i exch def
1345 :                 solid i solidcentreface
1346 :                 getpointVue
1347 :                 distance3d
1348 :              } for
1349 :           ] doublebubblesort pop reverse
1350 :        } {
1351 :           [
1352 :              0 1 F length 1 sub {
1353 :              } for
1354 :           ]
1355 :        } ifelse
1356 :        /ordre exch def
```

Maintenant on va définir l'ordre dans lequel on dessine les faces. Le tableau *ordre* défini ligne 1356 contiendra les indices des faces à dessiner, dans l'ordre convenable.

Si l'algorithme du peintre n'est pas utilisé (lignes $1351 - -1354$); l'ordre est tout simplement $0, 1, 2, \ldots, m-1$, où m est le nombre de faces du solide (la variable m n'est pas déclarée, c'est seulement la taille de F, i.e. **`F length`**).

Sinon, on crée un tableau des distances du centre des faces au point de vue (lignes $1342 - -$début 1344), puis on trie avec *doublebubblesort* (tri à bulle) qui dépose sur la pile le tableau des indices puis le tableau des distances triées par ordre croissant. On enlève ce dernier avec un **`pop`** (ligne 1349), puis on inverse le tableau des indices avec **`reverse`** (on veut dessiner suivant les distances décroissantes).

partiel

```
1358 :       0 1 F length 1 sub {
1359 :          /k exch def
1360 :          /i ordre k get def
1361 :          gsave
1362 :          A solid i dessinefacevisible
1363 :          grestore
1364 :       } for
```

Pour chacune des faces, dans l'ordre du tableau *ordre*, on appelle la procédure *dessinefacevisible*, qui dessine la face *i* si elle est visible, et qui ne fait rien sinon. La matrice *A* est passée en paramètre : si la procédure trace une arête, elle positionnera à *true* l'élément correspondant dans la matrice des arêtes.

partiel

```
1365 :       aretescachees {
1366 :          0 1 F length 1 sub {
1367 :             /k exch def
1368 :             /i ordre k get def
1369 :             A solid i dessinefacecachee
1370 :          } for
1371 :       } if
```

Si le switch *aretescahees* est à *true*, alors on recommence, mais cette fois en appelant la procédure *dessinefacecachee*.

partiel

```
1372 : %%       %% si on veut repasser les traits des faces visibles
1373 : %%       0 1 F length 1 sub {
1374 : %%          /k exch def
1375 : %%          /i ordre k get def
1376 : %%          gsave
1377 : %%          1 dict begin
1378 : %%             /startest false def
1379 : %%             A solid i dessinefacevisible
1380 : %%          end
1381 : %%          grestore
1382 : %%       } for
```

Enfin, commentée, on pourrait éventuellement repasser une dernière fois les traits des faces visibles avec cette portion de code où on positionne *startest* à *false* pour ne pas avoir le coloriage des faces.

3. Dessin des faces cachées

Le listing complet de la procédure:

la prodédure de dessin des faces cachées

```
1206 : %% syntaxe : A solid i dessinefacecachee
1207 : /dessinefacecachee {
1208 : 6 dict begin
1209 :    /i exch def
1210 :    /solid exch def
1211 :    solid issolid not {
1212 :       (Error : mauvais type d'argument dans dessinefacecachee) ==
1213 :       quit
1214 :    } if
1215 :    /A exch def
1216 :
1217 :    /F solid solidgetfaces def
1218 :    /S solid solidgetpointstable def
1219 :
1220 :    solid i solidfacevisible? not {
1221 :       %% face cachee => on prend chacune des aretes de la face et on
1222 :       %% regarde si elle est deja dessinee.
1223 :       4 dict begin
1224 :          /n F i get length def %% nb de sommets de la face
1225 :          0 1 n 1 sub {
```

la prodédure de dessin des faces cachées (suite)

```
1226 :           /k exch def
1227 :              /k1 F i k get_ij def              %% indice sommet1
1228 :              /k2 F i k 1 add n mod get_ij def  %% indice sommet2
1229 :              A k1 k2 get_ij not {
1230 :                 gsave
1231 :                    currentlinewidth .5 mul setlinewidth
1232 :                    pointilles
1233 :                    [S k1 getp3d
1234 :                    S k2 getp3d] ligne3d
1235 :                    A k1 k2 true put_ij
1236 :                    A k2 k1 true put_ij
1237 :                 grestore
1238 :              } if
1239 :           } for
1240 :        end
1241 :     } if
1242 : end
1243 : } def
```

et les explications :

partiel

```
1206 : %% syntaxe : A solid i dessinefacecachee
1207 : /dessinefacecachee {
1208 : 6 dict begin
1209 :    /i exch def
1210 :    /solid exch def
1211 :    solid issolid not {
1212 :       (Error : mauvais type d'argument dans dessinefacecachee) ==
1213 :       quit
1214 :    } if
1215 :    /A exch def
1216 :
1217 :    /F solid solidgetfaces def
1218 :    /S solid solidgetpointstable def
```

On commence par stocker les arguments : le numéro *i* de la face, le solide *solid* et la matrice *A* des arêtes. On teste le type de *solid* et on envoie éventuellement un message d'erreur.

Sinon on extrait les tableaux des faces puis des sommets et on les stocke respectivement dans *F* et *S*

partiel

```
1220 :    solid i solidfacevisible? not {
```

Si la face *i* est visible, on ne fait rien.

partiel

```
1221 :       %% face cachee => on prend chacune des aretes de la face et on
1222 :       %% regarde si elle est deja dessinee.
1223 :       4 dict begin
1224 :          /n F i get length def %% nb de sommets de la face
1225 :          0 1 n 1 sub {
1226 :          /k exch def
1227 :             /k1 F i k get_ij def              %% indice sommet1
1228 :             /k2 F i k 1 add n mod get_ij def  %% indice sommet2
1229 :             A k1 k2 get_ij not {
1230 :                gsave
1231 :                   currentlinewidth .5 mul setlinewidth
1232 :                   pointilles
1233 :                   [S k1 getp3d
1234 :                   S k2 getp3d] ligne3d
1235 :                   A k1 k2 true put_ij
1236 :                   A k2 k1 true put_ij
1237 :                grestore
1238 :             } if
1239 :          } for
1240 :       end
```

Sinon, pour chacune des arêtes de la face, on vérifie (ligne 1229) qu'elle n'est pas dessinée, et dans le cas contraire on la dessine (lignes 1233 – –1234) puis on fait le changement adéquat dans la matrice des arêtes (lignes 1235 – –1236).

4. Dessin des faces visibles et choix des couleurs

Le listing complet de la procédure:

la prodédure de dessin des faces visibles

```
1130 : %% syntaxe : A solid i dessinefacevisible
1131 : /dessinefacevisible {
1132 : 6 dict begin
1133 :    /i exch def
1134 :    /solid exch def
1135 :    /A exch def
1136 :    solid issolid not {
1137 :       (Error : mauvais type d'argument dans dessinefacevisible) ==
1138 :       quit
1139 :    } if
1140 :    /F solid solidgetfaces def
1141 :    /S solid solidgetpointstable def
1142 :
1143 :    solid i solidfacevisible? {
1144 :       /n F i get length def
1145 :
1146 :       startest {
1147 :          %% choix de la couleur
1148 :          /lightcolor where {
1149 :             pop
1150 :             /coeff
1151 :                lightintensity
1152 :                solid i solidnormaleface normalize3d
1153 :                solid i solidcentreface lightsrc vecteur3d normalize3d
1154 :                scalprod3d mul
1155 :                0 max 1 min
1156 :             def
1157 :             /fillstyle {
1158 :                 lightcolor {coeff mul} apply setcolor fill
1159 :             } def
1160 :             lightcolor {coeff mul} apply setcolor
1161 :          } {
1162 :             /lightsrc where {
1163 :                pop
1164 :                /coeff
1165 :                   lightintensity
1166 :                   solid i solidnormaleface normalize3d
1167 :                   solid i solidcentreface lightsrc vecteur3d normalize3d
1168 :                   scalprod3d mul
1169 :                   0 max 1 min
1170 :                def
1171 :                /lacouleur [
1172 :                   gsave
1173 :                      solid solidgetfcolors i get cvx exec currentrgbcolor
1174 :                   grestore
1175 :                ] def
1176 :                /fillstyle {
1177 :                   lacouleur {coeff mul} apply setcolor fill
1178 :                } def
1179 : %              lacouleur {coeff mul} apply setcolor
1180 :             } {
1181 :                solid F i get length affectecouleursolid_ncotes
1182 :                solid i affectecouleursolid_facei
1183 :             } ifelse
1184 :
1185 :          } ifelse
1186 :       } if
1187 :
1188 :       [  %% face visible : F [i]
1189 :          0 1 n 1 sub {
1190 :             /j exch def
1191 :             solid j i solidgetsommetface
1192 :          } for
1193 :       ] polygone3d
1194 :       %% on marque les aretes
1195 :       0 1 n 1 sub {
1196 :          /j exch def
1197 :          /k1 F i j get_ij def              %% indice sommet1
```

la prodédure de dessin des faces visibles (suite)

```
1198 :          /k2 F i j 1 add n mod get_ij def  %% indice sommet2
1199 :          A k1 k2 true put_ij
1200 :          A k2 k1 true put_ij
1201 :       } for
1202 :    } if
1203 : end
1204 : } def
```

et les explications :

partiel

```
1130 : %% syntaxe : A solid i dessinefacevisible
1131 : /dessinefacevisible {
1132 : 6 dict begin
1133 :    /i exch def
1134 :    /solid exch def
1135 :    /A exch def
1136 :    solid issolid not {
1137 :       (Error : mauvais type d'argument dans dessinefacevisible) ==
1138 :       quit
1139 :    } if
1140 :    /F solid solidgetfaces def
1141 :    /S solid solidgetpointstable def
```

On commence par stocker les arguments : le numéro *i* de la face, le solide *solid* et la matrice *A* des arêtes. On teste le type de *solid* et on envoie éventuellement un message d'erreur.

Sinon on extrait les tableaux des faces puis des sommets et on les stocke respectivement dans *F* et *S*

partiel

```
1143 :    solid i solidfacevisible? {
```

Si la face *i* n'est pas visible, on ne fait rien.

partiel

```
1146 :       startest {
1147 :          %% choix de la couleur
1148 :          /lightcolor where {
1149 :             pop
1150 :             /coeff
1151 :                lightintensity
1152 :                solid i solidnormaleface normalize3d
1153 :                solid i solidcentreface lightsrc vecteur3d normalize3d
1154 :                scalprod3d mul
1155 :                0 max 1 min
1156 :             def
1157 :             /fillstyle {
1158 :                 lightcolor {coeff mul} apply setcolor fill
1159 :             } def
1160 :             lightcolor {coeff mul} apply setcolor
1161 :          } {
1162 :             /lightsrc where {
1163 :                pop
1164 :                /coeff
1165 :                   lightintensity
1166 :                   solid i solidnormaleface normalize3d
1167 :                   solid i solidcentreface lightsrc vecteur3d normalize3d
1168 :                   scalprod3d mul
1169 :                   0 max 1 min
1170 :                def
1171 :                /lacouleur [
1172 :                   gsave
1173 :                      solid solidgetfcolors i get cvx exec currentrgbcolor
1174 :                   grestore
1175 :                ] def
1176 :                /fillstyle {
```

partiel (suite)

```
1177 :                    lacouleur {coeff mul} apply setcolor fill
1178 :                 } def
1179 : %               lacouleur {coeff mul} apply setcolor
1180 :              } {
1181 :                 solid F i get length affectecouleursolid_ncotes
1182 :                 solid i affectecouleursolid_facei
1183 :              } ifelse
1184 :
1185 :           } ifelse
```

Si l'on doit colorier les faces, il faut choisir la couleur. On distingue 2 cas : soit la variable *lightcolor* (couleur de la source de lumière) est définie (lignes 1149 – –1160), auquel cas on l'utilise, soit non (lignes 1162 – –1183). Dans ce dernier cas, on a 2 sous-cas : soit la variable *lightsrc* (position de la source lumineuse) est définie (lignes 1163 – –1179) auquel cas on calcule le coefficient d'intensité à appliquer en fonction de l'angle entre la normale de la face et le rayon lumineux, soit non (lignes 1181 – –1182) et on utilise la couleur de la face sans changement d'intensité.

La ligne 1160 fait que les trait du maillage sont de la couleur des faces dans le cas d'une source lumineuse ponctuelle colorée. Ils sont donc invisibles.

La ligne correspondante dans le cas d'une source lumineuse ponctuelle blanche est la ligne commentée 1179.