Documentation sur le fichier geometriesyr16.mp

Christophe Poulain

16 septembre 2005

Table des matières

Ce fichier est une nouvelle évolution de la série `geometriesyr`. Après la version 15 qui a apportée *le dessin à main levée*[1], cette nouvelle version apporte, outre quelques macros supplémentaires de dessin, un module qui permet LA GÉOMÉTRIE SPATIALE.

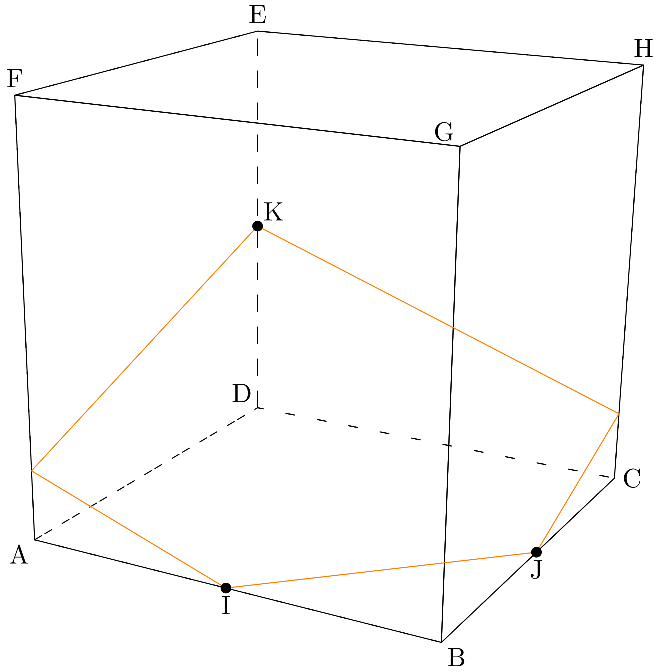


FIG. 1 – Section du cube $ABCDEFGH$ par le plan IJK

Cette idée fait bien évidemment suite à celle sur le tracé à main levée. De quel autre type de tracés avais-je besoin ? Les quelques solides étudiés en collège, l'utilisation de théorèmes « fondamentaux » dans ces solides,…tout ça m'a amené à me pencher sur le problème.

Je ne souhaitais pas devoir régler tout moi même, ou alors utiliser un subterfuge pour obtenir, finalement, une figure qui ne respecte pas forcément les réglès de représentation de la géométrie spatiale. D'ailleurs, voici un autre exemple obtenu avec `geometriesyr16.mp`.

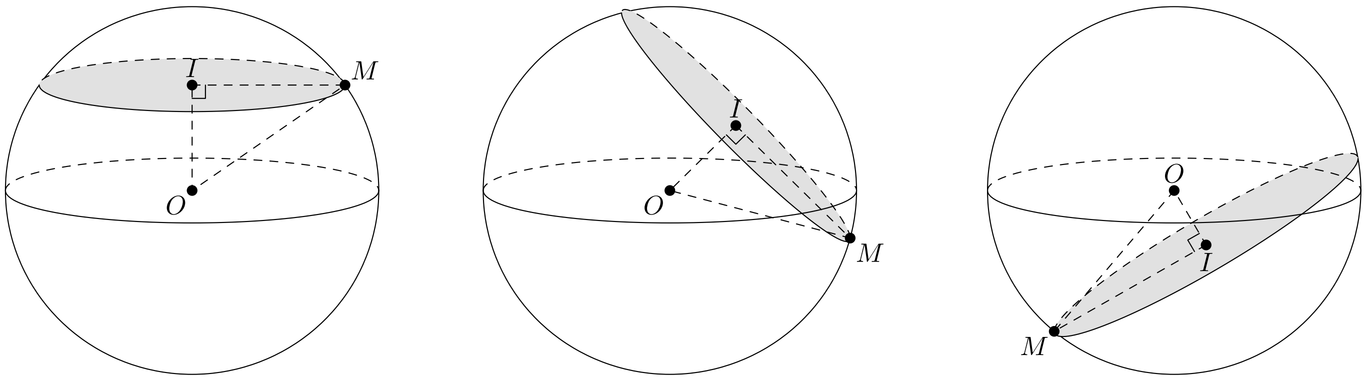


FIG. 2 – Sections d'une sphère par un plan

Mais quelles règles respecter ? Je me suis alors rappelé que MANUEL LUQUE avait conseillé de lire [1]. Lecture très instructive car elle m'a permi de mettre en place les diverses projections sur l'écran (projection perspective ou parallèle) et de commencer à créer quelques images[2]. Mais il manquait quelque chose : les intersections entre droites, entre droites et plans, entre plans. J'ai alors utilisé le

[1]À ce sujet, on pourra lire la documentation à la page *www.melusine.eu.org/syracuse/poulecl/macros/*

[2]On pourra aller voir les différents albums sur la géométrie dans l'espace à la page *www.melusine.eu.org/syracuse/poulecl/albums/*.

module de géométrie dans l'espace de Denis Roegel. Je n'ai eu qu'à l'adapter à ma programmation des méthodes de R.Dony. Je n'ai donc que peu de mérites ; tout avait déjà été fait.

Et pour finir cette partie, un exemple qui fera plaisir à Jean-Michel Sarlat.

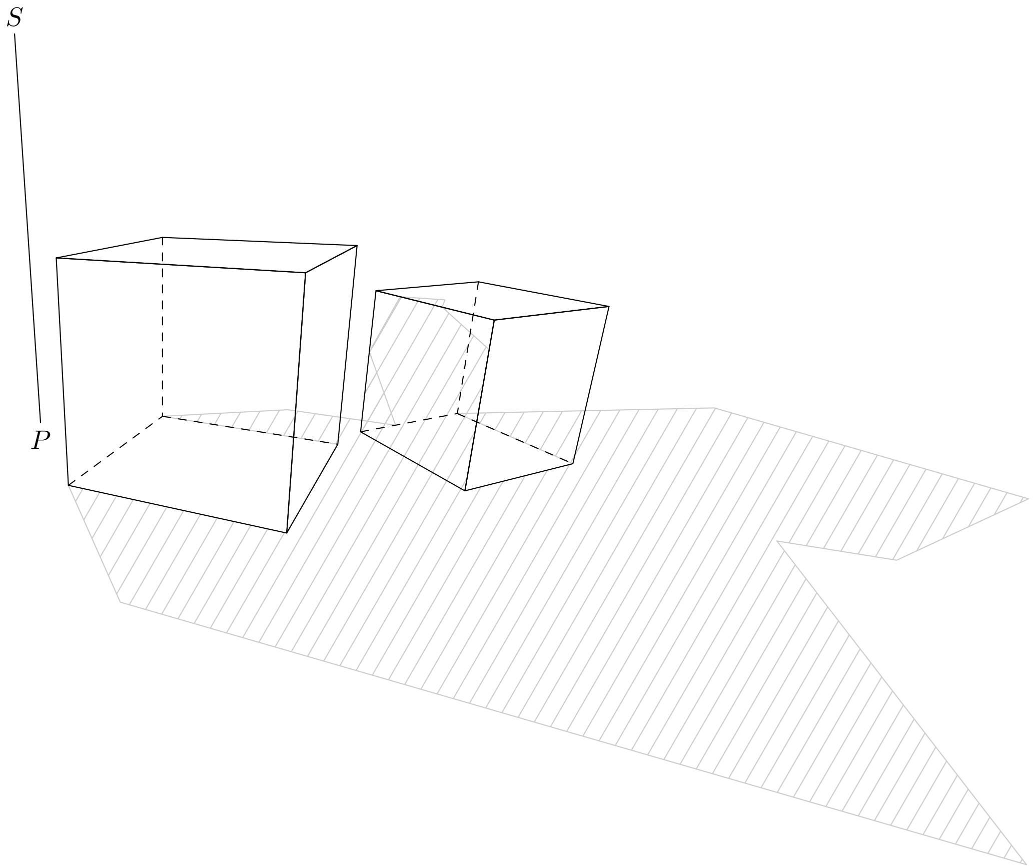


Fig. 3 – Ombre portée sur deux cubes.

1 Détails du fichier

Le but secondaire était que le tracé spatial s'inclue naturellement dans l'écriture des autres types de tracés faits avec `geometriesyr15.mp`. Donc il est fait appel aux fichiers :

– `constantes.mp` qui, comme son nom l'indique, contient les différents paramètres nécessaires aux tracés (couleurs, unités de longueur,...).
  Les constantes disponibles sont π, e, c la conversion d'un radian en degrés.
  Les couleurs disponibles sont rouge, bleu, vert, kaki, blanc, noir, orange, violet, rose, ciel, orangevif, jaune et gris. Elles s'utilisent avec ces noms francisés.

– `papiers2.mp` qui permet de produire[3] différents types de papiers (millimétré, 5×5, isométrique, ...)

– `donymodule.mp` qui contient toutes les routines nécessaires aux différents calculs à effectuer dans le cadre du tracé spatial.

et à plusieurs fichiers de base de METAPOST.

1.1 Environnement général de la figure.

Aux macros `figure(xa,xb,ya,yb);` - `fin;` - et `figuremainlevee(xa,xb,ya,yb)` - `finespace;` vient s'ajouter maintenant `figureespace(xa,xb,ya,yb);` - `finespace;`. Cela permet, comme les

[3]Très peu utile dans le cas d'un tracé spatial

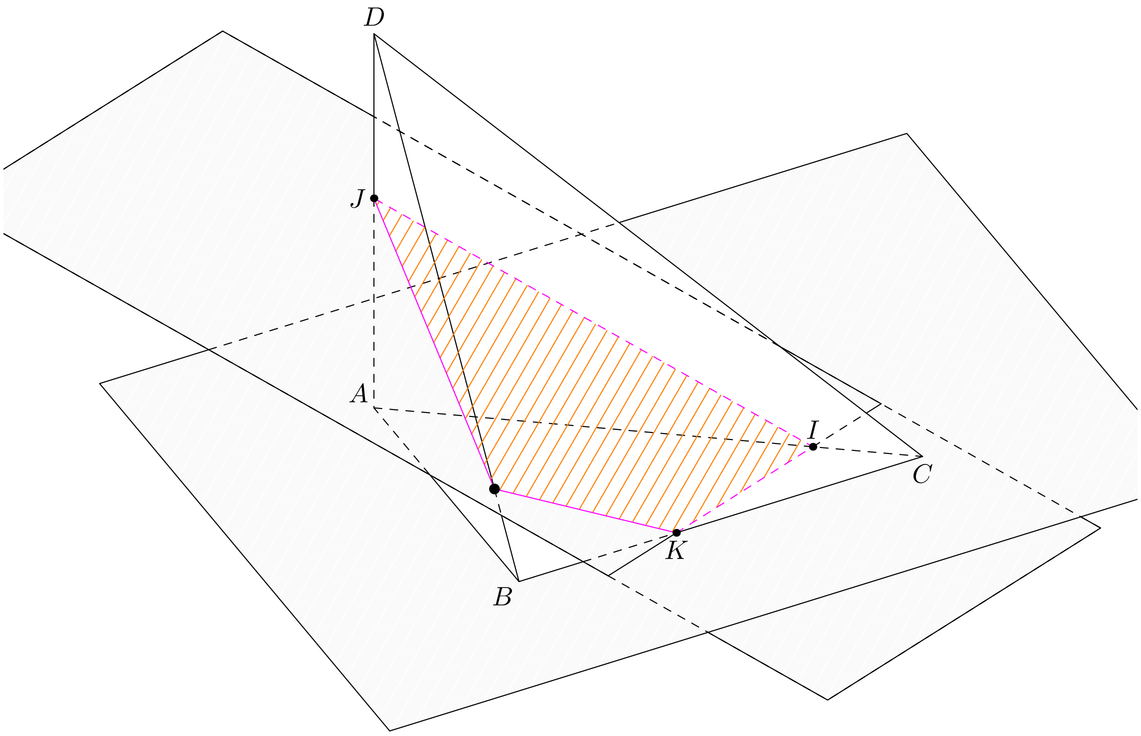


FIG. 4 – Section d'un tetraèdre par un plan.

autres d'inclure le schéma à l'intérieur d'un cadre[4] dont le sommet inférieur gauche a pour coordonnées (x_a, y_a) et le sommet supérieur droit (x_b, y_b). J'ai conservé la possibilité de pouvoir moduler, au sein du même fichier, plusieurs types de tracés différents et éventuellement alternés.

Par contre, dans le cadre du tracé spatial, les « points mathématiques » doivent être définis comme des « couleurs METAPOST ». Par exemple `color A; A=(0,0,0);` ; point A dont on accèdera aux composantes avec les commandes `redpart(A)`, `redpart(B)`, `redpart(C)`.

Une autre différence est, *avant le moindre tracé*, l'appel à la macro `Initialisation(5,30,20,500)` ; elle définit la position de l'oeil de l'observateur. En peu de mots, cet exemple signifie (5,30,20) représente les coordonnées sphériques de l'observateur dans le repère objet ; 500 représente la distance à l'écran[5].

1.2 Affichage.

Les macros ou paramètres d'affichage (`marque_p`, `pointe(A,B,C)`, `nomme.pos(A)`)[6] sont compatibles avec le tracé spatial.

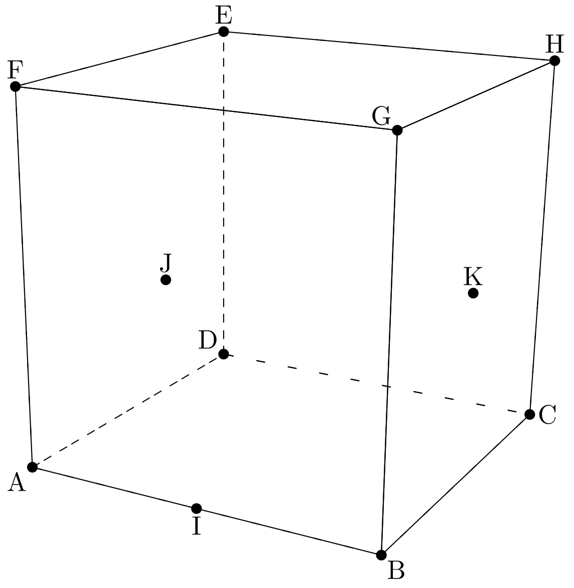
1.3 Points.

Comme déjà signalé, les « points »doivent être définis comme des « couleurs METAPOST ». On peut alors utiliser la commande `iso(A,B,C,...)` pour obtenir l'isobarycentre des points de l'espace A, B, C...Les autres macros[7] n'ont pas été adaptées au tracé spatial.

On dispose de définitions propres à la 3^{e} dimension :

Intersection de deux droites par la macro `IntersectionDroite(A,B,C,D)` pour obtenir l'intersection des droites (AB) et (CD).

[4]Je dois avouer que dans le cas d'un tracé spatial, les dimensions du cadre sont, *a priori*, difficiles à envisager.
[5]Pour plus de détail, voir [1]
[6]Ce sont les positions classiques de METAPOST : `llft`, `lft`, `ulft`, `top`, `urt`, `rt`, `lrt`, `bot`
[7]`CentreCercleC(A,B,C)`, `Orthocentre(A,B,C)` et `CentreCercleI(A,B,C)`

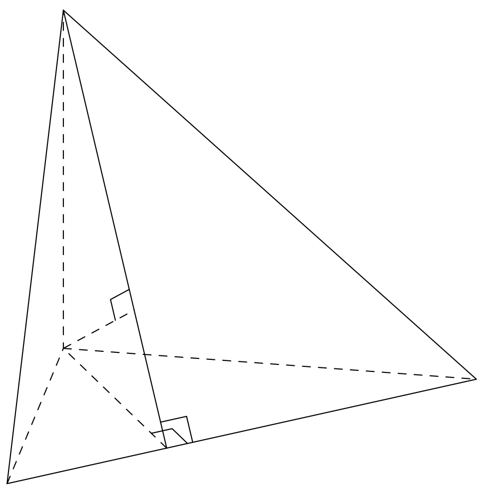



```
figureespace(-10u,-10u,10u,10u);
Initialisation(5,30,20,600);
color A,B,C,D,E,F,G,H,I,J,K;
trace Cube(A,B,C,D,E,F,G,H);
nomme.llft(A);nomme.lrt(B);
nomme.rt(C);nomme.ulft(D);
nomme.top(E);nomme.top(F);
nomme.ulft(G);nomme.top(H);
I=iso(A,B);
J=iso(A,D,E);
K=iso(B,C,H,G);
marque_p:="plein";
nomme.bot(I);nomme.top(J);
nomme.top(K);
finespace;
```

FIG. 5 – Isobarycentre de différents ensembles de points.

Intersection d'un plan et d'une droite par la macro `IntersectionPlanDroite(A,B,C,D,E)` qui construit le point d'intersection (s'il existe[8]) du plan (ABC) avec la droite (DE).

Projection orthogonale d'un point sur un plan avec la macro `ProjectionsurPlan(A,B,C,D)` pour obtenir le projeté orthogonal de A sur le plan (BCD).



```
figureespace(-10u,-10u,10u,10u);
Initialisation(5,10,25,500);
color A,B,C,O,H,K;
O=(0,0,0);
A=(1,0,0);
B=(0,1.5,0);
C=(0,0,1.25);
H=ProjectionsurPlan(O,A,B,C);
K=IntersectionDroite(C,H,A,B);
trace polygone(A,B,C);
trace segment(C,K);
trace chemin(A,O,C) dashed evenly;
trace segment(O,B) dashed evenly;
trace chemin(K,O,H) dashed evenly;
trace codeperp(O,H,C,8);
trace codeperp(C,K,B,10);
trace codeperp(O,K,B,8);
finespace;
```

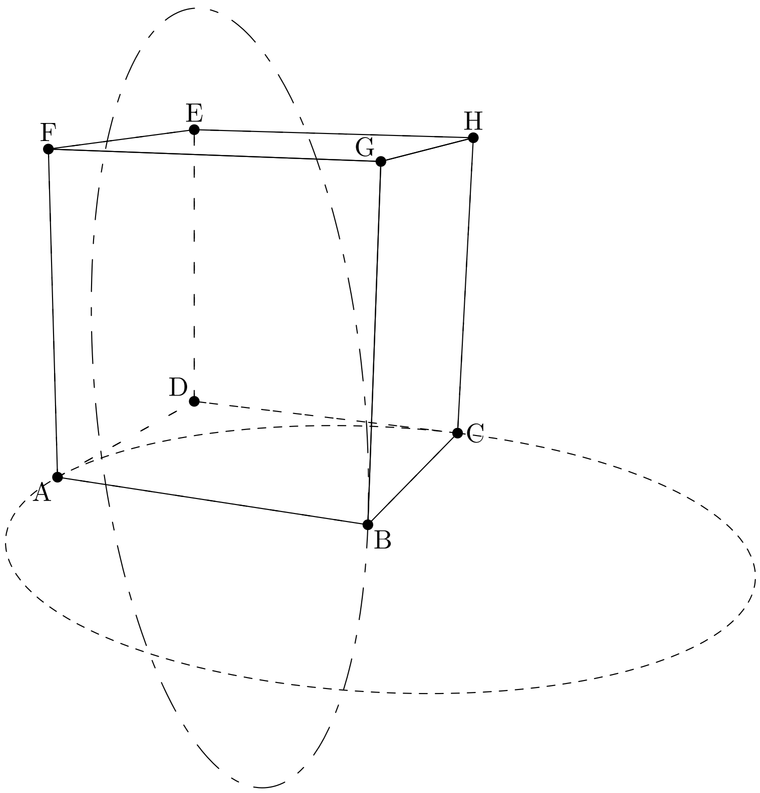
FIG. 6 – Projection d'un point sur un plan. Tétraèdre trirectangle.

1.4 Cercles.

Pour un dessin de géométrie dans l'espace, la seule façon d'obtenir un cercle avec ce package est d'utiliser la macro `cercles(A,B,O,C,D)`. Cela correspond au tracé du cercle de centre A et passant

[8]Un test est fait.

par B *dans le plan* (OCD) où $\overrightarrow{OC}$ et $\overrightarrow{OD}$ sont des vecteurs *orthogonaux*. Ils seront ensuite normés par la commande.




```
figureespace(-10u,-10u,10u,10u);
Initialisation(5,30,20,600);
color A,B,C,D,E,F,G,H;
trace Cube(A,B,C,D,E,F,G,H);
nomme.llft(A); nomme.lrt(B);
nomme.rt(C); nomme.ulft(D);
nomme.top(E); nomme.top(F);
nomme.ulft(G); nomme.top(H);
trace cercles(B,A,B,C,A)
 dashed evenly;
trace cercles(D,B,D,B,E)
 dashed dashpattern(on 12bp
 off6bp on3bp off6bp);
finespace;
```

FIG. 7 – Quelques cercles dans un cube.

Les arcs de cercles n'ont pas encore été implantés.

1.5 Droites.

Les constructions classiques sont disponibles :

segment(A,B) construit le segment $[AB]$.

droite(A,B) construit la droite (AB).

demidroite(A,B) construit la demi-droite $[AB)$.

On dispose aussi d'une construction propre à la géométrie spatiale :

```
IntersectionPlanPlan(A,B,C,D,E,F)
```

qui, je crois, porte bien son nom pour obtenir l'intersection (lorsqu'elle existe[9]) des plans (ABC) et (DEF). On pourra se faire une idée avec les figures 1 et 4.
Voici d'ailleurs le code source de la figure 1.

```
figureespace(-5u,-5u,6.5u,6u);
Initialisation(5,30,20,700);
color A,B,C,D,E,F,G,H,I,J,K;
trace Cube(A,B,C,D,E,F,G,H);
nomme.llft(A); nomme.lrt(B);
nomme.rt(C); nomme.ulft(D);
```

[9]Un test est fait

```
nomme.top(E); nomme.top(F);
nomme.ulft(G); nomme.top(H);
I=1/2[A,B]; J=1/2[B,C]; K=1/2[E,D];
path cc;
cc=buildcycle(
IntersectionPlanPlan(I,J,K,A,B,C),IntersectionPlanPlan(J,K,I,B,C,H),
IntersectionPlanPlan(K,J,I,C,D,E),IntersectionPlanPlan(J,K,I,F,E,D),
IntersectionPlanPlan(J,K,I,A,F,B)
);
trace cc withcolor orange;
marque_p:="plein";
nomme.bot(I); nomme.bot(J); nomme.urt(K);
finespace;
```

1.6 Transformations.

On peut utiliser l'addition de couleurs pour la translation et une commande telle `2[A,O]` pour obtenir le symétrique de A par rapport au point O.

1.7 Différents types de codage.

Voici la liste des codages disponibles, ils sont tous à utiliser avec la commande `trace`. Ce ne sont pas les seuls mais, actuellement, ce sont les seuls compatibles avec le tracé spatial.

Angle droit : `codeperp(A,B,C,5)` produit un angle droit en B avec pour « épaisseur » 5.

2 longueurs égales : `codesegments(A,B,C,D,n)` code les segments $[AB]$ et $[CD]$ avec le codage d'ordre n : 1, 2, 3 pour autant de traits, 4 pour la croix et 5 pour le cercle.

Longueurs égales

> `Codelongueur(A,B,C,D,...,n)` code les segments $[AB]$, $[CD]$,...avec le codage d'ordre n : 1, 2, 3 pour autant de traits, 4 pour la croix et 5 pour le cercle. Le nombre de lettres présentes dans l'appel de la macro doit être pair. L'ordre dans l'appel n'a pas d'importance, on aurait pu taper `Codelongueur(n,A,B,C,D,...)`

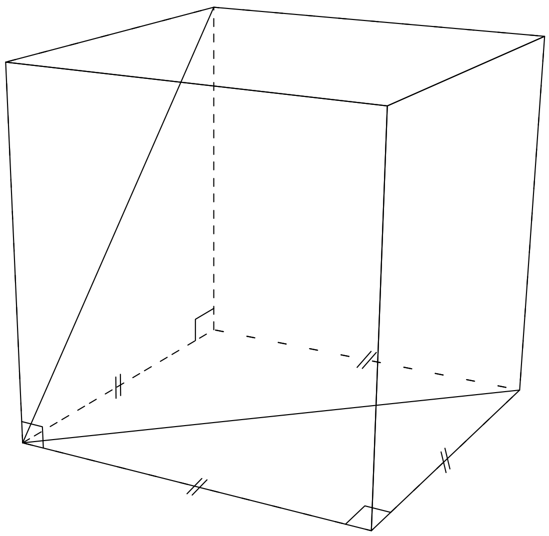


FIG. 8 – Différents codages dans un cube.

1.8 Sucres

L'hachurage[10] est toujours présent et seule la macro `appelation(A,B,2mm,btex nom etex)` a été adaptée.

Il existe aussi la macro `Cube(A,B,C,D,E,F,G,H)` dont il a déjà été fait usage dans les exemples précédents.

2 Développements

Il faudrait implanter les coordonnées sphériques, les arcs de cercles, quelques solides (pyramide et pavé), le dessin de plan, et certainement d'autres choses auxquelles je ne pense pas.

[10] Il ne dépend pas des points

Table des figures

Références

[1] Raymond DONY. *Graphisme dans le plan et l'espace avec Turbo Pascal*. Masson.