Visiteurs

alain marty

MODELISATION DE
SURFACES

SOMMAIRE

PRESENTATION 01

1 MODELES GEOMETRIQUES DES SURFACES 02
DEFINITION PAR EQUATION PARAMETRIQUE 02
DEFINITION PAR POINTS DE CONTROLE 03
DEFINITION PAR CONTRAINTES PHYSIQUES 05

2 MODELES DE REPRESENTATION 08
LE RENDU : DIFFUS, AMBIANT, SPECULAIRE 08
LES TRANSFORMATIONS : REPERE, POINT 09

3 LANGAGE 12
PRESENTATION 12
LES OBJETS 12
LA FAMILLE 13
L'ARBRE GENEALOGIQUE 14
LISTINGS 15
PROGRAMME PRINCIPAL 15
PROCEDURE UNE_FACETTE 15
PROCEDURE SURFACE_BEZIER 16
PROCEDURE SURFACE_MINIMALE 17

CONCLUSION 18

Alain Marty
Ingénieur ECL
Architecte DPLG
Enseignant à l'EALR
G•R•R•S•L•A

JUIN 1993

PRESENTATION

*

Le Groupe Recherche et Réalisation de Structures Légères pour l'Architecture (GRRSLA) dont le responsable est le Professeur René MOTRO, est basé dans les locaux de l'Ecole d'Architecture du Languedoc Roussillon (EALR), et s'est donné comme champ de recherche l'étude de la conception de systèmes constructifs.

L'un des champs d'investigation, "Architecture et Infographie", s'appuie déjà sur un langage de formes répondant à certains critères géométriques et mécaniques, la Formex Algebre créée par le Space Structures Research Centre de Guildford en Grande Bretagne.

Pour les divers besoins de la recherche et de leurs retombées sur l'enseignement des structures à l'EALR ou à l'Université de Montpellier, d'autres langages ont été analysés comme le fameux RenderMan Interface de la société PIXAR ou le Generative Modeling for Computer Graphics & CAD de John M. Snyder, mais celà n'a pas du tout semblé exclusif d'un langage "maison" plus modeste et certainement moins puissant, mais qui présenterait l'avantage d'être totalement ouvert (pas de boite noire), d'être écrit au contact de l'utilisateur (adaptable aux divers domaines abordés par le groupe) et si possible assez simple et clair pour devenir un support pédagogique à l'intention de futurs architectes des formes.

Et dans tous les cas, si comme le dit un certain BL WHORF: " Les langages modélisent nos pensées et déterminent ce à quoi nous pouvons penser", alors il y a toujours intérêt à réfléchir sur le choix d'une syntaxe et d'un vocabulaire appropriés pour décrire les structures et/ou les formes avec ou sans l'aide de l'outil informatique.

La recherche d'un langage de formes répondant à certains critères géométriques et/ou mécaniques passe par la définition d'une typologie de ces formes. Il nous a semblé intéressant de prendre

comme critère principal de cette typologie ce qui caractérise au fond les conditions de manipulation de ces formes à l'aide d'un curseur 3D et nous avons naturellement été conduits à définir les trois types primaires suivants :

- le premier type n'accepte aucune manipulation (une sphère n'est plus une sphère si l'on déplace l'un de ses points);
- le deuxième type accepte un contrôle limité de ses points par l'intermédiaire de polygones ou de polyèdres de contrôle (une surface de Bézier reste une bicubique si l'on déplace un point de son polyèdre de contrôle);
- le troisième type accepte la manipulation de n'importe lequel de ses points mais en propage les modifications à tous les autres points suivant une loi donnée (une membrane fine élastique suit tous les déplacements des points que l'on manipule tout en distribuant les autres points suivant la loi de Laplace).

Le premier type correspond aux formes définies directement par une équation paramétrique (les quadriques par exemple), le deuxième type correspond aux formes définies par points de contrôle constituant autant de poignées agissant indirectement sur la forme, le troisième type correspond aux formes définies par la donnée de contraintes (conditions aux limites) et d'un mode de réaction.

Dans ce qui suit, nous présentons donc ces trois types dans une première partie, nous proposons ensuite rapidement les quelques rappels sur le rendu et les transformations qui nous paraissent indispensables à la compréhension des exemples montrés, et nous terminons par la présentation de l'état actuel du langage et l'application sous forme de listings à la modélisation des trois types de surface.

Ce travail est un exemple de l'état actuel de la recherche entreprise; il a été réalisé en vue de servir de base à un cours sur la modélisation des surfaces qui a été donné dans le cadre d'une formation Doctorale en Mécanique des Matériaux, Structure, Génie des Procédés à l'Université Montpellier II, ainsi qu'à un cours donné dans le cadre d'un Certificat Optionnel sur les Structures Légères (prof. R MOTRO) à l'EALR.

1 MODELES GEOMETRIQUES DES SURFACES

Une surface peut être définie comme le lieu d'un ensemble de points P(x,y,z) dont les trois coordonnées sont fonctions de deux variables indépendantes s et t:

```
            x := x(s,t);
P(s,t)      y := y(s,t);
            z := z(s,t);
```

Nous choisirons ici d'étudier trois types particuliers de définition du point courant d'une surface P(s,t) sur un intervalle continu de définition: s,t = [0,1].

Toutes ces surfaces seront homéomorphes au carré unité, c'est à dire applicables sur un carré de dimension unitaire après déformation élastique (et après découpage des bords si la surface est fermée).

La discrétisation de l'intervalle continu se fera de la façon suivante:

s:=i/iMax et
t:=j/jMax avec i,j = [0..iMax,0..jMax].

1) Le premier type suppose la donnée d'un fonction vectorielle paramétrique P(s,t);

2) Le deuxième type suppose la donnée d'un tableau de points P[i,j], dits points de contrôle, sur l'intervalle de définition i,j dans [0..MC,0..NC] et d'une fonction P(s,t) reliant le point P à ce tableau de points sur l'intervalle de définition s,t=[0,1], réalisant ainsi une sorte de lissage, interpolant les points de contrôle ou non;

3) Le troisième type suppose la donnée d'une équation différentielle:

f(P,P',P",...) = 0

sur l'intervalle de définition continu: s,t=[0,1]ou d'une équation aux différences finies sur un intervalle discret: i,j=[0..iMax,0..jMax], et la donnée de contraintes aux limites sur P[s,t] pour certaines valeurs de s et de t.

11 DEFINITION PAR UNE EQUATION PARAMETRIQUE

Les surfaces définies par une équation paramétrique sont nombreuses: les quadriques en sont une famille bien connue. Nous ne rappelerons ici que le cas de la sphère et auparavant celui du cercle pour préciser les notations

* *

fig. 1: une portion de sphère fig. 2: la même surface ondulée et tordue

111 CERCLE

Un cercle d'axe Oz, de rayon R, centré sur l'origine peut être défini par:

```
            x := R*cos(2*pi*s);
P(s)        y := R*sin(2*pi*s);
```

```
            z := 0;             avec s dans [0,1];
```

112 SPHERE

Dans le cas de la sphère, la définition pourrait être la suivante:

```
            x := R*cos(2*pi*s)*cos(pi*t);
    P(s,t)  y := R*cos(2*pi*s)*sin(pi*t);
            z := R*sin(2*pi*s);     avec s,t dans [0,1];
```

Mais cet exemple conduit à une sphère complète (du pôle Sud au pôle Nord sur une rotation de 360°), et il est souvent préférable de limiter la surface à créer à une portion non fermée : en effectuant le changement de variable suivant:

```
    s := (1-s)*phiMin+s*phiMax      donc s est dans [phiMin, phiMax]
    t := t*thetaMax             et t est dans [0, thetaMax]
```

la surface générée sera ouverte et reliera le méridien phiMin au méridien phiMax sur une rotation de 0 à thetaMax autour de l'axe Oz (figure 1). On peut ainsi réaliser quantité de formes par des assemblages de portions de sphère bien choisies.

De plus, il est possible de perturber la surface en appliquant sur une ou plusieurs coordonnée une transformation comme par exemple la transformation suivante:

```
    x := x + A*cos(T*s);            avec s dans [0,1];
```

ce qui provoquera sur x une ondulation sinusoïdale d'amplitude A et de période T, et pourra enrichir l'aspect de la surface comme le montre la figure 2 .

12 DEFINITION PAR POINTS DE CONTROLE

L'étude des courbes et des surfaces définies par points de contrôle est relativement récente: les astronomes ont été les premiers à étudier de façon intensive le problème de l'interpolation d'une série de points expérimentaux; LAGRANGE est à l'origine d'une méthode classique d'interpolation, mais l'application en est lourde et les courbes résultantes présentent des oscillations "parasites" assez génantes.

Plus près de nous, Bézier, deCastelJau et Coons, pour ne citer que les plus connus, ont été d'ardents promoteurs de nouvelles approches plus pratiques et très adaptées aux calculs sur ordinateurs. Nous ne nous intéresserons ici qu'aux bases élémentaires d'une approche de la modélisation des surfaces qui a fait et fera encore couler beaucoup d'encre.

```
    *           *
```

fig. 3: un arc de parabole défini par 3 points fig. 4: une cubique définie par 4 points

121 DEUX POINTS: UN SEGMENT

On écrit qu'un point P appartient au segment formé par deux points P0 et P1 généralement sous la forme suivante:

```
    P := P0 + s*P0P1    avec s dans [0,1],
```

mais l'écriture suivante plus symétrique par rapport à P0 et P1, et dite écriture barycentrique, (fig. 3):

```
    P := (1-s)*P0 + s*P1    avec s dans [0,1]
```

est bien plus adaptée aux extrapolations qui suivent. P0 et P1 sont deux points affectés des poids (1-s) et (s), dont la somme vaut 1; ces poids marquent le pouvoir d'attraction, la quantité d'influence, le contrôle, que chaque point a sur P en fonction de la valeur de s. Et on imagine que P pourrait être contrôlé par plus de deux points..

122 TROIS POINTS: UNE PARABOLE

Dans le cas de plusieurs points [P0.. PN], il existe plusieurs possibilités : on pourrait rechercher une fonction P(s) interpolant ces points, (c'est le cas du simple segment) mais ce problème est loin d'être trivial et conduit comme il a été dit à une certaine instabilité des résultats (oscillations parasites, etc..); une autre approche consiste à prendre une fonction polynomiale de degré convenable et à en exprimer les coefficients en fonction de la suite de points Pi; nous analyserons un cas particulier de cette approche, peut être le plus constructif et sans doute plus intuitif, basé sur l'algorithme de "de Casteljau" prolongeant naturellement le cas précédant à deux points. Nous n'étudierons que les cas à 3 et à 4 points.

Prenons tout d'abord le cas de trois points. L'analyse de la figure 3 conduit aux relations suivantes: P0, P1 et P2 sont les points donnés (dits points de contrôle)

```
    Q0 est positionné à s sur P0P1  soit: Q0 := (1-s)*P0 + s*P1;
    Q1 est positionné à s sur P1P2  soit:   Q1 := (1-s)*P1 + s*P2;
    P est positionné à s sur Q0Q1       soit:   P := (1-s)*Q0 + s*Q1;
```

ce qui, en éliminant les points intermédiaires Q0 et Q1, conduit à l'expression de P en fonction de P0, P1 et P2:

```
P := (1-s)*(1-s)*P0 + 2*(1-s)*s*P1 + s*s*P2;
```

qui est un polynome du second degré définissant un arc de parabole dans l'espace. La parabole a la propriété d'être tangente aux vecteurs P0P1 et P1P2 en P0 et P2. Cette expression est simple et conduit à des cacluls rapides; certains logiciels (comme MACDRAW) ne connaissent que cette expression pour lisser un polygone de N points, en calculant les milieux des segments successifs et en traçant des arcs de parabole passant par ces points et tirés par les sommets pour être tangents aux cotés du polygone en ces points. Le problème est qu'un arc de parabole est plan et, dans le cas de la 3D, une courbe gauche constituée par une série d'arcs successifs de parabole présentera des défauts de continuité en torsion, ce qui est au moins mauvais pour les bras des robots pilotés par une telle courbe... Il est donc nécessaire d'aller plus loin...

123 QUATRE POINTS: UNE CUBIQUE

En appliquant la même construction récursive au cas de 4 points [P0..P3], on est conduit à une courbe d'équation:

```
P := (1-s)*(1-s)*(1-s)*P0 + 3*(1-s)*(1-s)*s*P1 + 3*(1-s)*s*s*P2 + s*s*s*P3;
```

qui est l'expression d'un polygone cubique, plus connu sous le nom de courbe de Bézier du 3ème degré (fig. 4). Les poids affectant chaque point de contrôle ont une somme unitaire. Cette expression peut facilement être écrite pour le cas général de N+1 points, mais ce n'est pas notre problème dans le cadre de cet article.

124 QUATRE POINTS: UNE FACETTE GAUCHE

La donnée de 4 points peut aussi conduire à une surface.

*

fig. 5: construction d'une facette gauche

Un tableau de 4 points [P00,P10,P01,P11] nous permet d'engendrer deux segments P00P10 et P01P11 dont les points courants Q0 et Q1 situés à la valeur s s'expriment comme suit :

```
Q0 := (1-s)*P00 + s*P10;
Q1 := (1-s)*P01 + s*P11;
```

et dans l'intervalle t=[0,1], le segment Q0Q1 engendre lui-même une surface dont le point courant s'écrit :

```
P := (1-t)*Q0 + t*Q1;
```

ou bien, en éliminant les points intermédiaires Q0 et Q1, et exprimant P en fonction des 4 points P00,P10,P01,P11:

```
P := (1-s)*(1-t)*P00 + (1-s)*t*P10 + s*(1-t)*P01 + s*t*P11;
    avec s,t dans[0,1]
```

Ceci est l'expression bilinéaire d'une surface appelée également paraboloïde hyperbolique (fig. 5); totalement cohérente avec les données fournies, cette expression est bien plus facile à manipuler que l'expression classique dérivée de l'équation générale des quadriques.

125 BICUBIQUE

* *

fig. 6: construction d'une bicubique fig. 7: une surface de Bézier

La facette gauche était construite à partir de segments et on en imagine sans trop de peine une généralisation au cas d'une surface qui serait construite à partir de courbes de Bézier, appelée bien sur "surface de Bézier": la donnée d'un tableau de 4*4 = 16 points conduit à la génération de 4 courbes de Bézier (dans un sens ou dans l'autre) et le parcours de chaque courbe à avancement égal (même valeur de la variable s entre 0 et 1) positionne 4 points (un sur chaque courbe) conduisant eux-mêmes à une courbe de Bézier transversale: on génère donc ainsi une surface par glissement d'une courbe variable sur 4 courbes fixes (fig. 6) et (fig. 7).

Comme pour les courbes polynomiales, le nombre de points de contrôle peut être augmenté de façon théoriquement illimitée, mais en pratique, au delà de 16 points de contrôle, et donc au delà d'une bicubique, on a coutume de choisir un autre type de construction : on décompose la courbe (surface) en une suite de cubiques (bicubique) dont les coefficients sont calculés pour assurer automatiquement la continuité de l'ensemble des points, de leurs tangentes et de la courbure: ce sont les courbes (surfaces) Splines et au delà les courbes (surfaces) définies par des polygones à

coefficients rationnels et à intervalles variables, les NURBS, qui sortent du cadre de cet article.

13 DEFINITION PAR CONTRAINTES

Certaines courbes ou surfaces ne sont connues que par les inter-relations qui peuvent exister entre les points, et les contraintes extérieures: par exemple la trajectoire d'un corps pesant, la courbe d'équilibre d'une chaine, la surface d'une lame de savon glissant le long de plans jusqu'à l'équilibre. En général la courbe ou la surface finale n'est pas connue, et il est nécessaire de partir d'une forme initiale proche d'une solution possible et de laisser se "relaxer" cette forme dans une série d'itérations à la recherche d'un minimum.

131 COURBE FUNICULAIRE

Soit une suite de points P[i] avec i=[0..iMax] reliés par des cables et soumis à un champ de force F[i]; comme illustré sur la figure 8 , et en choisissant convenablement les unités, l'équilibre de chaque point se traduit par la relation vectorielle suivante:

(P[i-1] - P[i]) + (P[i+1] - P[i]) + F[i] = 0,

qui amène à l'expression de P[i] en fonction des deux points voisins :

```
P[i] := (P[i-1] + P[i+1] + F[i])/2;
```

Les points P[i] étant initialisés à une valeur a priori quelconque dans l'intervalle i= [0..iMax], et à des valeurs fixées [P0.. PN] pour certaines valeurs de i, un calcul itératif amène les points P[i] à des valeurs stables définissant un polygone appelé "polygone funiculaire". Quand cette suite de points est simplement tenu aux deux extrémités, est soumise à son poids propre et lorsque iMax tend vers l'infini, le polygone tend vers une courbe appelée chainette. L'équation différentielle de la chainette s'écrit :

d2P(s) / ds2 + F(s) = 0 avec s dans [0,1] et P[0] = P0 et P[1] = P1

qui est l'expression continue de l'équation aux différences finies trouvée plus haut.

* *

fig. 8: courbe funiculaire fig. 9: surface funiculaire

132 SURFACE FUNICULAIRE

Dans le cas d'une surface, l'équation différentielle (aux dérivées partielles) devient l'équation de Laplace :

```
d2P(s,t)/ds2 + d2P(s,t)/dt2 + F(s,t) = 0    avec s,t dans [0,1],
```

et P[s,t] fixé à des valeurs données pour certaines valeurs de s et t.

L'équation aux différences finies s'écrit (fig. 9):

```
(P[i-1,j] - P[i,j]) + (P[i+1,j] - P[i,j]) +
(P[i,j-1] - P[i,j]) + (P[i,j+1] - P[i,j]) + F[i]=0
```

qui amène à l'expression de P[i,j] en fontion des 4 points en croix et de F:

```
P[i] := (P[i-1,j] + P[i+1,j] + P[i,j-1] + P[i,j+1] + F[i])/4;
```

qui est manifestement une simple généralisation de l'expression du polygone funiculaire trouvée plus haut: en l'absence de force, le point P est tout simplement situé en position médiane par rapport aux quatre points voisins en croix.

On démontre que les surfaces funiculaires soumises à un champ de force nul sont des surfaces d'aire minimale.

On peut affecter d'un poids chaque point et ainsi simuler des surfaces élastiques non isotropes, et en particulier la présence de ralingues dans une toile.

133 DEUX EXEMPLES

Les deux exemples représentent des surfaces minimales.

La première surface est obtenue en fixant une facette sur 3 cotés à la base du cube de référence de µ3D, en fixant un segment en position centrale de cette surface en position haute, en laissant libre le troisième coté, en soumettant la surface à une force verticale dirigée vers le bas, et en laissant bien sur le tout se relaxer.

Le mathématicien français GERGONNE a posé en 1816 le problème suivant: trouver la surface minimale dont deux cotés sont fixés sur deux diagonales opposées d'un cube et le partageant en deux volumes égaux; les deux autres cotés glissent donc librement sur les autres faces du cube. Le

deuxième exemple en montre une illustration dans le cas où on laisse les deux autres cotés libres de glisser. La surface de départ est encore une facette mais cette fois-ci elle est gauche.

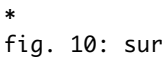

fig. 10: surface minimale

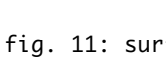

fig. 11: surface de Gergonne

14 COMMENTAIRE SUR LES TROIS TYPES

Le premier type de surface présente peu de maniabilité: l'équation est en général définie en fonction de quelques paramètres contrôlant l'aspect de la surface dans sa globalité, et il est dificile sinon impossible d'en déplacer un point sans dénaturer la forme de façon incontrôlable.

Le contrôle du second type de surface est par définition un contrôle par points, la fonction de lissage assurant automatiquement la continuité de la surface, de ses tangentes et de ses courbures. En modifiant la position des points on peut s'approcher du tracé recherché sans dénaturer la surface, la précision de l'approche dépendant du nombre de points.

Le contrôle du troisième type présente des analogies avec le précédent: il suffit de manipuler les courbes limitant la surface, de positionner les points singuliers, de préciser les modes de contrainte (point fixe ou glissant sur un plan, etc...), de fixer le champ de force appliqué en chaque point (éventuellemet nul) et de laisser le tout se mettre en place, se relaxer...

Le premier type est assimilable à une surface rigide, indéformable; le deuxième type correspond à une surface présentant une certaine souplesse, mais aussi une certaine rigidité, une certaine résistance à la flexion; le troisième type n'a aucune résistance à la flexion et se positionne au mieux de son aise dans l'espace en prenant tout juste appui sur les limites imposées.

Les trois approches sont combinables et l'on peut envisager par exemple une surface bicubique définie à partir d'un polyèdre de 16 sommets dont les points seraient à l'origine sur une sphère, seraient ensuite déplacés et fixés arbitrairement, et enfin soumis à une légère brise de Nord-Ouest...

Une fois définie une surface par ses caractéristiques géométriques, il reste à en préciser les attributs physiques, à la positionner dans une scène et à éclairer le tout. C'est l'objet de la section suivante.

2 MODELES DE REPRESENTATION

Une fois qu'une surface a été définie, il s'agit de la représenter, et il faut donc décider du mode de rendu des points la constituant et de la façon de les projeter sur l'écran.

Le rendu demande le calcul de l'interaction entre la ou les sources lumineuses et chaque point de la surface, considéré comme un élément de matière orienté dans l'espace. La projection sur l'écran demande le calcul de la suite de transformations permettant de passer du repère de définition de la surface au repère lié à l'observateur.

21 LE RENDU

Il y a plusieurs rendus possibles pour représenter une surface : le plus simple consiste à tracer le maillage construit sur le tableau de points, mais cette représentation est limitée et inefficace dans le cas de surfaces complexes.

Représenter une surface pleine consiste à calculer en chaque point de la surface vu depuis le point d'observation à travers la grille des pixels de l'écran, le vecteur couleur (dont les composantes sont habituellement le rouge, le vert et le bleu) résultant de l'action de toutes les sources lumineuses sur l'élément de surface en ce point.

211 AMBIANT, DIFFUS, SPECULAIRE

On peut décrire la couleur d'un point comme la somme de trois reflets issus de l'action de la ou des sources lumineuses sur une surface:

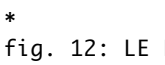

fig. 12: LE DIFFUS

fig. 13: LE SPECULAIRE

1) L'AMBIANT qui est lié à l'apport considéré en première approximation comme constant de la lumière ambiante, somme de toutes les interactions entre tous les objets lumineux ou pas; sous ce seul apport, la surface sera montrée uniformément colorée sans qu'il soit possible de discerner le volume; le calcul effectif de cet apport dans toute sa complexité fait l'objet d'une technique complexe et lourde en calculs appelée "radiosité";

Dans notre hypothèse, le reflet ambiant est constant.

2) LE DIFFUS qui est lié à l'éclairement provenant de chaque source lumineuse, et qui révèle la couleur de la surface et celà d'autant plus que l'élément de surface est plus orienté vers la source: la réflection est au maximum quand l'angle entre la normale et la direction de la source lumineuse est nul, et nulle quand cet angle est au delà de 90°; cet apport lumineux contribue à bien révéler les variations d'orientation de la surface et donc le volume de l'objet;
Le reflet diffus dépend des positions relatives des souces lumineuses et de la surface.

3) LE SPECULAIRE qui est, comme le reflet diffus, lié à l'éclairement provenant de chaque source lumineuse, mais qui est situé dans le cône de réflection spéculaire dont l'axe est le rayon réfléchi, symétrique du rayon lumineux incident par rapport à la normale à l'élement de surface en ce point: le reflet est d'autant plus important que l'observateur se trouve proche de la direction du rayon réfléchi, l'angle du cône dépendant de la rugosité de la surface, la dimension de la tache réfléchie étant infiniment petite si la surface est un miroir parfait, ou étendue (diffusée) à toute la surface dans le cas contraire; on se limitera au calcul de l'apport des seules sources lumineuse sans inter-réflections, ce calcul effectif dans toute sa complexité faisant l'objet d'une technique lourde en calculs appelée "lancé de rayon";
Le reflet spéculaire dépend des positions relatives des souces lumineuses, de l'observateur et de la surface.

Nous donnons ci-dessous l'expression de la couleur réfléchie par un point dans la direction de l'observateur, le point étant soumis à une seule source lumineuse. Le calcul des reflets diffus et spéculaire supposant la connaissance de la normale en chaque point, on déterminera tout d'abord celle-ci.

212 CALCUL DE LA NORMALE
Une surface étant définie par un tableau de points P[i,j] dans un intervalle de définition i,j = [0..iMax,0..jMax], on peut définir les tangentes en P[i,j] comme étant les vecteurs:

```
dTi[i,j] := (P[i+1,j] - P[i-1,j])/2;
dTj[i,j] := (P[i,j+1] - P[i,j-1])/2;
```

et la normale N[i,j] comme étant le produit vectoriel entre ces deux vecteurs (normalisé ensuite au vecteur unité):

```
N[i,j] := produit_vectoriel(dTi[i,j], dTj[i,j]);
```

213 CALCUL DE LA COULEUR
Soient , en omettant les [i,j] pour une lecture plus facile:
V le vecteur unitaire issu du point P et dirigé vers l'observateur,
L le vecteur unitaire issu du point P et dirigé vers la source lumineuse
R le vecteur symétrique de L par rapport à la normale;

alors le vecteur couleur C en chaque point P peut s'écrire sous la forme:

```
C :=    Ka*C_ambiant + Kd*C_diffus + Ks*C_spéculaire;
```

avec Ka, Kd, Ks comme coefficients de restitution des apports (en principe compris entre 0 et 1), et:

```
C_ambiant    := couleur_ambiante;
C_diffus     := couleur_surface * produit_scalaire(N, L);
C_speculaire := couleur_lumiere * puissance(produit_scalaire(V, R),n);
            et avec R := - L + 2*produit_scalaire(N, L)*N
```

le produit scalaire exprimant l'angle (le cosinus) entre deux vecteurs unitaires, le facteur n modulant la dimension de la tache spéculaire (un petit n étale la tache, un grand la diminue), la dernière ligne donnant R en fonction de L et de N.

Connaissant la couleur des points de la surface, on peut alors représenter ce point sur l'écran... à condition qu'il ne soit pas caché par un ou plusieurs autres. Si plusieurs points se projettent sous le même pixel, seul le point le plus proche de l'observateur est pris en compte (on supposera ici que les surfaces ne sont pas transparentes); celà impose qu'à chaque pixel de l'écran corresponde, en plus de la mémoire consacrée à la couleur (RGB), une mémoire des profondeurs initialisée à l'infini en début de calcul et actualisée lors du calcul de chaque pixel.
La profondeur est en fait la distance entre l'observateur et le point considéré; dans le cas où l'observateur est à l'infini, le cône de projection devient un cylindre de projection et la distance est simplement la différence entre les "z" des deux points 3D situés au même endroit sur l'écran de projection et il suffit de mémoriser les "z" dans un .. zBuffer. On peut toujours se ramener au cas de l'observateur à l'infini en appliquant à la scène une transformation appelée la perspective canonique que nous aborderons un peu plus loin.

22 TRANSFORMATIONS DU REPERE ET DU POINT

221 PRINCIPE

En général les surfaces sont définies dans le repère le mieux adapté à leur description: l'équation de la sphère définie en début d'article est ainsi plus facile à écrire pour une sphère d'axe Oz et centrée à l'origine du repère.

En général le repère associé à la fenêtre est centré sur la fenêtre, l'axe Ox est horizontal vers la droite, l'axe Oy vertical orienté vers le haut, et l'axe Oz étant perpendiculaire à l'écran en général orienté dans le sens du regard.

Un premier problème est que décrit dans un tel repère, un cube est montré sous le forme de sa base carrée, un cylindre sous la forme d'un cercle, etc... ; un second problème est que les objets sont rarement au même endroit, ni dans la même orientation les uns par rapport aux autres.

Il faut donc disposer d'opérateurs pouvant agir sur un repère courant, sorte de curseur tridimensionnel désignant le lieu et l'orientation du repère par rapport auquel chaque surface est à décrire, et d'opérateurs transformant chaque point depuis le repère courant jusqu'au repère associé à la fenêtre.

L'idée est d'associer une matrice au repère courant formé des vecteurs i, j, k, ainsi qu'à chaque opérateur de transformation :

- translation(tx,ty,tz);
- homothetie(hx,hy,hz);
- rotation(angle,axe); avec axe = [Ox,Oy,Oz],

et

- projection(focale);

le tout pouvant être combiné selon les règles de l'algèbre linéaire et enfin appliqué aux coordonnées de chaque point P(x,y,z).

Il se trouve qu'on peut facilement écrire sous forme d'une matrice 3x3 les transformations d'homothétie et de rotation mais pas la translation ni la projection; il faut pour celà passer à la dimension supérieure (4). L'idée est donc d'écrire toutes les transformations dans un espace à quatre dimensions, et les appliquer à des points à quatre dimensions dont il s'agira de trouver les correspondants dans notre espace à trois dimensions. Sans entrer dans le détail, il faut savoir que la correspondance s'écrit:

P4(x,y,z,w) --> P3(x/w,y/w,z/w)

* *

fig. 14: avant la perspective canonique... fig.15: ... et après !

222 LES TRANSFORMATIONS

Nous donnons sans démonstration les expressions des matrices de transformation appliquée au repère initial:

ijk initial =
```
1   0   0   0
0   1   0   0
0   0   1   0
0   0   0   1
```

Translation(tx,ty,tz) =
```
1   0   0   0
0   1   0   0
0   0   1   0
tx  ty  tz  1
```

Homothétie(hx,hy,hz) =
```
hx  0   0   0
0   hy  0   0
0   0   hz  0
0   0   0   1
```

Rotation(a,Ox) =
```
1   0   0   0
0   cosa    -sina   0
0   sina    cosa    0
0   0   0   1
```

Rotation(a,Oy) =
```
cosa    0   sina    0
0   1   0   0
-sina   0   cosa    0
0   0   0   1
```

Rotation(a,Ox) =
```
cosa    -sina   0   0
sina    cosa    0   0
0   0   1   0
0   0   0   1
```

Perspective(foc) =
```
1   0   0   0
0   1   0   0
0   0   -1/foc  0
```

```
                    0   0   -1/foc  1
```

Cette dernière expression est en fait le produit d'une perspective canonique (non inversible car de déterminant nul) et d'une translation T(0,0,foc) aboutissant à une matrice inversible: toutes les autres étant également inversibles, il est facile de conserver la matrice inverse de la matrice de base, matrice qui sera la clé des opérations de contrôle interactif à la souris d'un point 3D.

Une perspective canonique (fig 14) et (fig 15) est une transformation associant à un point 3D situé au delà du plan de projection et dans la pyramide ayant l'observateur pour sommet un autre point 3D situé dans un cube unité. On reste dans l'espace à 3 dimensions, mais les objets sont déformés de telle sorte qu'une simple projection orthogonale aboutira au même résultat que la projection conique, ce qui sera bien utile pour gérer le zBuffer vu plus haut...

Appliquer une transformation au repère courant reviendra à multiplier les matrices associées; et transformer un point 3D en son équivalent sur le repère associé à l'écran reviendra à appliquer la matrice du repère courant au point 4D P(x,y,z,1) correspondant au point 3D P(x,y,z); ce produit conduira normalement à un point 4D P(x,y,z,w) pour lequel w sera différent de 1 et l'on reviendra enfin aux trois dimensions en divisant toutes les coordonnées par la dernière P(x/w,y/w,z/w,1), correspondant au point 3D P(x/w,y/w,z/w), et en adaptant les deux dimensions en x et en y à l'échelle (à la résolution) de la fenêtre de travail.

223 UTILISATION

Cette mécanique sera simplement mise en jeu par un ensemble d'instructions du genre suivant:

```
begin
    initialiser_repere;     { ijk = matrice_unité}
    perspective(focale);        { ijk = ijk • matrice_perspective}
    translation(tx,ty,tz);      { ijk = ijk • matrice_translation}
    etc...
    p := point3D(x,y,z);        {}
    p := coordonnees_ecran(p);  {(x',y',z') = (x,y,z) • ijk}
    p := coordonnees_fenetre(p);    {(p[1],p[2],z) = (x',y',z') • matrice_fenetre}
    moveTo(p[1],p[2]);      {}
    etc...
end;
```

Ce qui nous amène à la présentation du langage effectivement utilisé pour mettre en oeuvre tous les concepts précédents.

3 LE LANGAGE

La syntaxe et le vocabulaire proposés pour manipuler les concepts présentés plus haut sont maintenant brièvement exposés et enfin appliqués aux trois types de surface.

31 PRESENTATION

Le programme µ3D est écrit en Think Pascal sur une machine Macintosh. Le portage sur langage C et/ou sur d'autres environnements (MSDOS, UNIX,..) suppose la réécriture de l'interface seule qui reste secondaire, et ne concerne pas les concepts 3D fondamentaux.

Il est articulé en modules (Units) séparés contenant la déclaration et le corps des différents objets le composant:

- application, menu, fenêtre,
- rendu, repère,
- point, courbe, surface,
- objets (sphère, cube, ph, bézier,...),
- shaders (sable, ondulation,...),
- scènes, animations,...
- et enfin le corps principal du programme .

Le programme est destiné à être utilisé à plusieurs niveaux, depuis la production de scènes et d'animations à partir de la simple utilisation des objets géométriques prédéfinis, jusqu'à l'écriture de nouveaux objets plus ou moins complexes liés par exemple à l'étude du comportement de structure tendues soumises à des cas de charge complexes et dynamiques, et même jusqu'à la réalisation d'applications complètes et autonomes.

Dans cet esprit, c'est autant la recherche de la qualité de l'écriture de µ3D que le résultat à atteindre qui a été le souci principal de la démarche de base. Dans ce travail, le modèle permanent, le guide à travers la forêt touffue des concepts de l'infographie a été le langage proposé par la société américaine PIXAR pour décrire de façon standard et structurée les scènes infographiques; le logiciel Renderman Interface, produit sur ce langage, constituant un laboratoire d'approfondissement et d'application.

Le programme µ3D est autant un travail d'accompagnement pour une maitrise effective des concepts infographiques, qu'un travail de prolongement et d'application d'outils du type Renderman

à des sujets plus particuliers, comme la modélisation de surfaces devant répondre à des critères de forme et / ou de structure.

32 OBJETS

Pascal est un beau langage bien structuré disposant de nombreuses structures de données et de contrôle, et notamment une extension objet. Grâce à ce type d'extension, tout ou presque tout peut être construit sous une forme cohérente, compacte, autonome, réutilisable, hierarchisable. Un objet est capable de réagir à des messages et de modifier ses caractéristiques ou celles d'autres objets. On définit tout d'abord une famille de classes (sortes de modèles de réaction) à partir desquels on va créer et faire interagir un ensemble d'objets.

321 PRINCIPE

Supposons que nous souhaitions dessiner un objet quelconque (OBJ).

Il faut d'abord en choisir le modèle, disons le modèle t_objet (pour type_objet), définir le repère dans lequel il sera dessiné et le rendu de surface qu'il recevra. Supposons qu'il existe déjà un objet gérant tout ce qui concerne les repères (REP) et un objet gérant tout ce qui concerne le rendu (REN), c'est à dire la position des sources lumineuses, la définition des états de surface, etc.

On prononcera alors une phrase du genre:

- créer OBJ de type t_objet,
- donner à OBJ les valeurs des paramètres publics (par exemple le rayon d'une sphère),
- donner l'ordre à OBJ de calculer les paramètres privés (par exemple les normales),
- demander à l'objet REP de positionner le repère courant dans l'espace,
- demander à l'objet REN de préparer la source lumineuse et la couleur courante,
- enfin de demander OBJ de se dessiner dans les conditions ainsi définies,
- et ceci fait demander à OBJ de s'auto-détruire.

L'élément de programme, limité à la définition de l'objet, s'écrira comme suit:

```
var
    OBJ : t_objet;
begin
    New(OBJ);                   { création}
    OBJ.debut(20, 20;               { maillage de 21x21 points}
    OBJ.parametres(un, deux, trois);    { définition externe de l'objet}
    OBJ.calculer;               { définition interne}
    REP.rotation(30,Oz);            { rotation du repère courant}
    REN.couleur(1,0,0);         { la surface sera rouge}
    OBJ. dessiner;              { ordre de se dessiner}
    OBJ.fin;                    { au revoir }
end;
```

Parler dans le langage µ3D revient donc, suivant les niveaux d'approche, à choisir et mettre en scène des objets prédéfinis et de les faire se représenter suivant des choix de repère et de rendu dont on n'a pas à connaitre le détail; ou bien à un niveau supérieur, à créer des classes d'objets qui seront utilisées par d'autres.

322 LA FAMILLE

On distinguera les objets incontournables et présents tout au long de la vie du programme, de ceux qui sont créés, utilisés et détruits suivant les besoins pour produire une scène ou une animation. En page 14 est donné le BROWSER du programme, tableau représentant l'arbre généalogique de tous les objets du programme.

3221 OBJETS GLOBAUX

La liste des objets dits "globaux" est à l'heure actuelle la suivante:

APP l'application
MEN les menus
FEN les fenêtres
REP le repère courant
REN l'état graphique courant
SHA le shader matière courant
DEF le shader déformation courant

Les deux derniers objets (shader pour rendu) sont des filtres destinés à modifier les autres objets: par exemple les shaders ondulation et sable pourraient être appliqués à l'objet plan facette pour produire une dune sableuse.

Les concepts à la base des objets REN et REP ont été donnés au début de cette seconde partie, et les objets MEN, FEN et APP présentent un intérêt secondaire pour cet article, quoique essentiel pour le programme (il faut au moins une fenêtre pour visualiser tout ce beau monde, et une application et des menus pour organiser le tout pendant la vie du programme.

3222 OBJETS LOCAUX

Les objets dits “locaux” sont en fait les différentes courbes et surfaces héritant des propriétés contenues dans les classes t_courbe et t_surface. C’est le cas par exemple des sphères, des facettes, des surfaces splines, etc... Chacune de ces surfaces sait se calculer, se tracer, appeler des filtres de déformation, des modes de rendu et de projection, et même sait s’appliquer des relaxations élastiques à la manière d’une surface minimale.

Il est enfin également possible de contrôler interactivement à la souris chacun de ces objets, soit globalement (translations rotations) soit ponctuellement, en intercalant simplement dans un listing analogue à celui du début de cette page un ordre du type: OBJ.controle;

323 LES SCENES

Comme les objets, les scènes sont articulées en blocs comprenant des instructions de début et de fin pouvant être imbriquées. On peut ainsi empiler et dépiler les états successifs du repère courant et du rendu courant. On peut enfin construire des suites de scènes produisant des images à enregistrer destinées à être incorporées dans un fichier MOOVIE et à être repassées à grande vitesse grace à l’extension système QuickTime.

Les listings commentés illustrant tout ce qui a été dit sur les trois exemples de surface étudiés au début de l’article, sont donnés dans les pages qui suivent: le listing est présenté dans la colonne de gauche, et chaque ligne est commentée en regard dans la colonne de droite.

arbre généalogique des objets composant le programme µ3D

listing 1
listing 2
listing 3

CONCLUSION

Dans ce qui précède, nous avons étudié trois types de surface, allant de la surface mathématique abstraite dont la forme prédéterminée est totalement indéformable, à la surface naturelle dont la forme se soumet avec grâce au principe de la recherche du minimum, en passant par la surface “flexible” et manipulable par des “poignées”, surface ancrée dans le monde de l’industrie, surface “opérationnelle”, surface “artefact”.

Chacune de ces surfaces peut être obtenue par déformation souple d’un carré unité, chacune peut être échantillonnée par un tableau de points, chacune peut être représentable en utilisant les mêmes processus, les mêmes choix de rendu. Ces surfaces appartiennent à une famille dont l’arbre généalogique est donné en page 14, et qui comprend d’autres surfaces comme les surfaces de révolution, les surfaces tubulaires, et des surfaces étranges comme les surfaces de Moëbius ou comme celle d’un ticket de Métro plié en 4 dont les assemblages conduisent à des noeuds intéressants pour les structures réticulées. Toutes ces surfaces peuvent se calculer, subir des déformations apportées par les objets t_deformation, recevoir des textures apportées par les objets t_shader, peuvent être contrôlées interactivement par l’utilisateur, peuvent se relaxer entre les contraintes qu’il a définies, peuvent être animées le long de chemins rectilignes, circulaires, cubiques ou splines...

Le langage permettant de mettre en oeuvre tous ces objets est un simple ensemble de données et de procédures assemblées en objets cachant les aspects “de bas niveau” et tentant d’exprimer de mieux en mieux les concepts de base en infographie, en modélisation des surfaces, en algèbre vectorielle, en math et en physique pour tout dire... c’est à dire là où le passage est absolument obligatoire quelle que soit la qualité d’un logiciel pour passer de la consommation à la création!

Ce langage travaille dans un environnement de développement comme le proposent les Think Pascal ou les Turbo Pascal, avec un bon éditeur de texte, un bon compilateur réduisant les temps d’attente à presque rien, un bon “débogueur” toujours à l’affût des incohérences. Le surensemble est réduit au minimum, il n’y a pas un étage intermédiaire à apprendre, seul doit être connu le langage de base dans ses aspects fondamentaux, les boucles, les tests, le typage de données; seule doit être connue la logique de base du langage qui peut alors être du Pascal, du C, ou d’autres à venir.

Ce langage est ouvert: l’arbre généalogique est extensible en objets, en shaders de déformation et de texture; il est possible d’enrichir les objets de données et de fonctions complémentaires sans remettre en question l’ensemble. Les recherches actuelles sont orientées vers la définition d’une typologie des surfaces suivant un principe de composition de courbes génératrices: par exemple pour engendrer la surface d’une banane, ou de pétales de fleur, ou des pales d’une turbine à eau, sans réinventer “la roue” à chaque fois et dans un langage si possible clair, lisible, et pourquoi pas élégant.

Entre les logiciels prêts à produire de l’image de synthèse et les cours magistraux abstraits sur l’infographie, le programme µ3D est en somme une tentative de couplage entre la programmation d’un ensemble cohérent de primitives de plus en plus abstraites, et la structuration d’un langage

construit sur ces primitives qui soit de plus en plus simple et de plus en plus évident, l'ensemble devenant pour différents profils de chercheurs un véritable outil d'exploration, de simulation et au delà de création.

alain marty le 5 JUIN 1993

BIBLIOGRAPHIE

La bibliographie qui suit liste les principaux ouvrages qui ont été utilisés à des titres divers dans cette recherche.

MATHEMATIQUES
1 La Géométrie Analytique, André Delachet, 1047 coll "Que Sais-je",PUF
2 Mathématiques et CAO 5, Méthode matricielle, S.A.Coons, Hermes
3 Géométrie Différentielle, J.Lelong-Ferrand, Masson
4 Courbes et Surfaces, André Delachet, 564 coll "Que Sais-je",PUF
5 Les Géométries, L.Godeaux, 206, Armand Colin
6 La Géométrie Contemporaine, André Delachet, 401 coll "Que Sais-je",PUF
7 La Géométrie Projective, André Delachet, 1103 coll "Que Sais-je",PUF
8 Calcul Vectoriel et Tensoriel, André Delachet, 418 coll "Que Sais-je",PUF
9 Eléments de Calcul Tensoriel,A.Lichnerowitz, 259, Armand Colin

STRUCTURES
1 Les Voiles minces en béton armé, A.Paduart, Presses univ Bruxelles, Eyrolles
2 Le voile autoportant, P.Conil, Eyrolles
3 Calcul des Structures en Informatique, J.F.Auboin, 7 Eyrolles

LANGAGES INFORMATIQUES
1 Manuel de Think Pascal 4 et du C C++, Symantec
2 Scientific Programming with MacPascal, R.E.Crandall & M.M.Colgrove, Wiley Press
3 Macintosh Program. PRIMER ThinkPascal, I et II, D.Mark & C.Reed, Addison Wesley
4 Structuration des données en informatique, B.Ibrahim & C.Pellegrini, Dunod Informatique
5 Pascal Procedures Understanding Graf3D, B.Scott, Mac Tutor Vol3 N°3
6 Programming QuickDraw , D A Surovell, F M Hall, K Othmer, Addison Wesley

INFOGRAPHIE
1 Three-Dimensional Computer Graphics, Alan Watt, Addison Wesley
2 Fundamentals of interactive Computer Graphics, Foley&VanDam,Addison Wesley
3 Principles of Interactive Computer Graphics, Newman&Sproul, Mc Graw Hill
4 Programmation en Infographie-principes et exercices, L.Ammeraal, Masson
5 Interactive Computer Graphics, P.Burger & D.Gillies, Addison Wesley
6 Mathematical Elements for Computer Graphics, ?, ?
7 Algorithmes pour l'Infographie, David F.Rogers, Mc Graw Hill
8 Graphiques en Turbo-Pascal, R.Dony, Masson
9 Computer Graphics, Edward Angel, Addison Wesley
10 Images de Synthèse, M.Bret, Bo-Pré Dunod Informatique
11 L'image numérique, JM.Nasr, LED Initiation Eyrolles
12 L'image numérique II, JM.Nasr, LED Initiation Eyrolles
13 Generative Modeling for Computers and CAD, JM SNYDER, Academic Press
14 Advanced Graphics Programming Using C / C++, Loren Heiny , JohnWiley & Sons

LOGICIELS
1 Exception, J.Herrando, Montpellier
2 The Renderman Companion, Steve Upstill / Pixar, Addison Wesley
3 MacRenderman Developer's Stuff, Pixar,

DIVERS
1 Monsieur Tompkins au Pays des Merveilles, GAMOW

LISTINGS

Les objets repère (REP) et rendu (REN) sont créés au lancement du programme et sont disponibles à tout moment pour recevoir les instructions de modification du repère courant ou du rendu courant.

LE PROGRAMME PRINCIPAL

```
PROGRAMME µ3D;
USES
    Primitives, Objets, Shaders;
BEGIN
    debut_3D;
    REP.perspective(1.5);
    REP.rotation(-60, Ox);
    REP.rotation(-30, Oz);
    axes(0.5);
    cube(0.5, 0.5, 0.5);
    ampoule(    1,
                        0.03,
                        point3D(0, -0.5, 0.5),
                        couleur(1, 1, 1));
    BEGIN
            une_facette;            {1}
            une_Sbezier;            {2}
            un_voile_mince;     {3}
    END;
    fin_3D;
END.
```

Entre le BEGIN et le END; en milieu du programme sont appelés 3 procédures décrites dans les lignes suivantes.

UNE SURFACE SIMPLE

```
PROCEDURE une_facette;      {1}
VAR
    POLY: t_facette;
BEGIN
    New(POLY);
    WITH POLY DO BEGIN
        debut(20, 20);
        geometrie(
                            point3D(-0.5, -0.5, -0.5),
                            point3D(0.5, -0.5, -0.5),
                            point3D(-0.5, 0.5, 0.5),
                            point3D(0.5, 0.5, -0.5));
        calculer_points;
        tracer;
        fin;
    END;
END;
```

COMMENTAIRES

déclaration du programme principal
déclaration des fichiers d'objets utilisés utilisés par ce programme
début du bloc programme
création des objets du programme
l'objet REP, créé à l'initialisation, va répondre aux 3 transformations demandées et positionner le repère ijk courant
cet appel montre le repère courant
cet appel montre un cube unité filaire
création d'une source lumineuse n°1,
qui sera une sphère de rayon 0.03,
placée au point 3D: x=0, y=-0.5, z=0.5,

```
et de couleur blanche: R=G=B=1
debut de la scène à décrire:
    une facette gauche (PH)
    une surface de Bézier
    un voile mince élastique
fin de la scène à décrire
destruction des objets du programme
fin du bloc programme
```

```
_________________________________________
déclaration de l'objet une_facette
déclaration des variables
POLY sera un objet de type facette
début de la procédure
création de l'objet POLY en mémoire
pour cet objet POLY faire
création d'un polyèdre de 400 facettes
transmission des paramètres à utiliser:
    le point 00
    le point 10
    le point 01
    le point 11
calcul du tableau de 21x21 points
tracé du maillage entre ces points
destruction du tableau et de lobjet

fin de la procédure
    LA SURFACE DE BEZIER


    Définir une surface de Bézier c'est d'abord positionner dans l'espace un tableau de 4x4 points.
Pour celà on crée une surface intermédiaire plane (ou non) de 16 points, on la mofifie à la souris,
puis on donne le résultat à la surface qui en tirera les informations pour calculer le maillage
final.
    On donne à l'objet rendu (REN) les informations sur la couleur et les coefficients de
réflexion, au shader SABLE la quantité dosant la densité de points noirs simulant le sable, à
l'objet repère (REP) le type de transformation désiré.
    On demande à la surface de se dessiner puis de se supprimer. On détruit enfin le polyèdre des
points de contrôle.
```

```
_________________________________________
PROCEDURE une_SBezier;      {2}
VAR
    POLY: t_facette;
    SURF: t_SBezier;
BEGIN
    New(POLY);
    POLY.debut(3, 3);
    POLY.geometrie(
                point3D(-0.5, -0.5, -0.5),
                point3D(0.5, -0.5, -0.5),
                point3D(-0.5, 0.5, 0.5),
                point3D(0.5, 0.5, -0.5));
    POLY.calculer_points;
    POLY.controler;
    POLY.tracer;
    BEGIN
        New(SURF);
        SURF.debut(30,30);
        SURF.geometrie(POLY);
    SURF.calculer_points;
    REN.debut_attributs;
        REN.attributs(
```

```
                                  couleur(1, 0, 0),
                                KaKdKsKr(0.1,1,5,0.05),
                                  NOT TEST_VISIBILITE);
            SABLE.parametres(0.3);

            REP.debut_transformation;
                REP.rotation(30,0z);
                SURF.dessiner(SABLE);
            REP.fin_transformation;
        REN.fin_attributs;
        SURF.fin;
    END;
    POLY.fin;
END;
    COMMENTAIRES
```

A noter que le polyèdre de contrôle peut être n'importe quelle surface définie sur un maillage de 4x4. Par exemple un hyperboloide de révolution ou une fonction aléatoire à lisser.

A noter également qu'une courbe de Bézier est construite de la même façon (sur un polygone de contrôle) et qu'elle peut servir de génératrice à une surface de révolution ou même de directrice et de génératrice à un tubage.

Toute surface peut être déformée interactivement globalement ou ponctuellement: il suffit d'incorporer le message:
SURF.controle
à l'endroit voulu, le programme reprenant le cours après la frappe de la touche 'q'.

```
_____________________________________
début de la procédure surface de Bézier
déclaration des variables
POLY sera le polyèdre de contrôle
SURF sera une surface de Bézier

création de l'objet POLY en mémoire
création d'un tableau de 4x4 points3D
transmission des paramètres à utiliser:
    le point 00
    le point 10
    le point 01
    le point 11
calcul du tableau de 16 points3D
on controle les points à la souris  tracé du maillage entre ces points

création de l'objet SURF en mémoire
création d'une surface de 900 facettes
transmission des 16 points de contrôle
calcul du tableau de 31x31 points
conservation de l'état du RENdu courant
transmission des attributs à l'objet REN
    couleur de la surface = rouge
    les 4 coefficients de réflection
    représenter aussi les faces arrières
SABLE est un objet générant sur la surface des points noirs aléatoires
conservation du REPère courant
application d'une rotation d'axe Oz
dessin de la surface avec matière sable
restauration du REPère précédent
restauration de l'état du RENdu préced
destruction de la surface de Bézier

destruction du polyèdre de contrôle
```

LE PROBLEME DE GERGONNE

Le mathématicien français GERGONNE a posé en 1816 le problème suivant: trouver la surface minimale dont deux cotés sont fixés sur deux diagonales opposées d'un cube et le partageant en deux volumes égaux; les deux autres cotés glissent donc librement sur les autres faces du cube.

```
_____________________________________
PROCEDURE gergonne;     {3)
VAR
    SURF: t_facette;
    i,j: integer;
```

```
BEGIN
    New(SURF);
    WITH SURF DO BEGIN
        debut(20, 20);
    geometrie(
                            point3D(-0.5, -0.5, -0.5),
                            point3D(0.5, 0.5, -0.5),
                            point3D(0.5, -0.5, 0.5),
                            point3D(-0.5, 0.5, 0.5));
        calculer_points;

    intervalle(iMax, jMax);
        FOR i := 0 TO iMax DO
            contraindre_point(i,0,
                                        point3D(-1,-1,-1));
        FOR i := 0 TO iMax DO
            contraindre_point(i,jMax,
                                        point3D(-1,-1,-1));
        FOR j := 0 TO jMax DO
            contraindre_point(0,j,
                                        point3D(0,-1,0));
        FOR j := 0 TO jMax DO
            contraindre_point(iMax,j,
                                        point3D(0,-1,0));

    WHILE NOT Button DO
        BEGIN
            laplace(force3D(0, 0, 0));
            FEN.effacer;
            cube(0.5, 0.5, 0.5);
            tracer;
        END;

    FOR i := 1 TO 36 DO
        BEGIN
            REN.debut_scene('TENTE.', i);
            REP.rotation(10, Oz);
                dessiner(SABLE);
            REN.fin_scene(ENREGISTRER);
        END;

    fin;
    END;

END;
```

COMMENTAIRES

On part simplement d'une facette gauche (paraboloïde hyperbolique) s'appuyant sur 4 cotés rectilignes, surface définissable par 4 points de contrôle et assez proche de la solution. Le nombre d'itérations permettant d'aboutir à une surface stable est de l'ordre de 15.

---

début de la procédure
déclaration des variables
SURF sera la facette de départ
i et j sont des compteurs

création de l'objet SURF en mémoire
on s'adressera toujours à SURF..
création d'un tableau de 21x21 points3D
transmission des paramètres à utiliser:
    le point 00 est en bas du cube
    le point 10 ""
    le point 01 est en haut du cube
    le point 11 ""
calcul du tableau de points3D

on rappelle les valeurs iMax et jMax
• définition des contraintes aux limites
pour le coté j=0 de la facette on bloque tous les points en x, y et z

idem pour le coté opposé j=jMax

pour le coté i=0 on laissera glisser les points sur le plan Oxz en ne bloquant que y, et en laissant libres x et y

idem pour le coté opposé i=iMax

• calcul de tous les autres points
tant que le bouton n'est pas appuyé
relaxer un à un chaque point avec une force appliquée nulle
effacer la fenêtre
tracer le cube de référence
tracer le maillage
et recommencer

faire 36 images

s'appelant TENTE.1, TENTE.2,.. TENTE.36
avec une rotation / Oz de 360°/36=10°
dessiner suivant la matière sable
enregistrer chaque image

destruction du polyèdre de contrôle