Les courbes et surfaces lissées

Les courbes paramétriques cubiques

Définition

Courbe paramétrique cubique de R^3: Courbe définie par le système d'équations paramétriques cubiques suivant:

$$Q(t) = \begin{cases} x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) = a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) = a_z t^3 + b_z t^2 + c_z t + d_z \end{cases} \qquad t \in R \quad (1)$$

Pour obtenir des segments de courbe, on considère que t appartient à un intervalle [min,max] (fréquemment [0.0,1.0]).

Reformulation mathématique

Soit le vecteur $T = (t^3, t^2, t, 1)$, (1) s'écrit sous la forme

$$Q(t) = (x(t),y(t),z(t)) = T.C \text{ avec } C = \begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{pmatrix} \text{ et } t \in R.$$

Les courbes de cette famille ont pour propriétés d'être:

- continues,
- de dérivés premières en t continues (dérivation de chaque composante de T),
- de dérivés secondes en t continues (double dérivation de chaque composante de T).

Quatre coefficients numériques constants sont nécessaires pour définir chacune des trois équations paramétriques.
-> Quatre contraintes mathématiques sont imposées pour définir l'équation selon chacun des trois axes.

RETOUR

On transforme la matrice C donnée précédemment en la matrice produit M.G où M est une matrice 4x4 (appelée matrice de base) et G est un vecteur colonne de quatre contraintes géométriques (appelé vecteur géométrie).

$Q(t) = (x(t), y(t), z(t)) = T.C = T.M.G =$

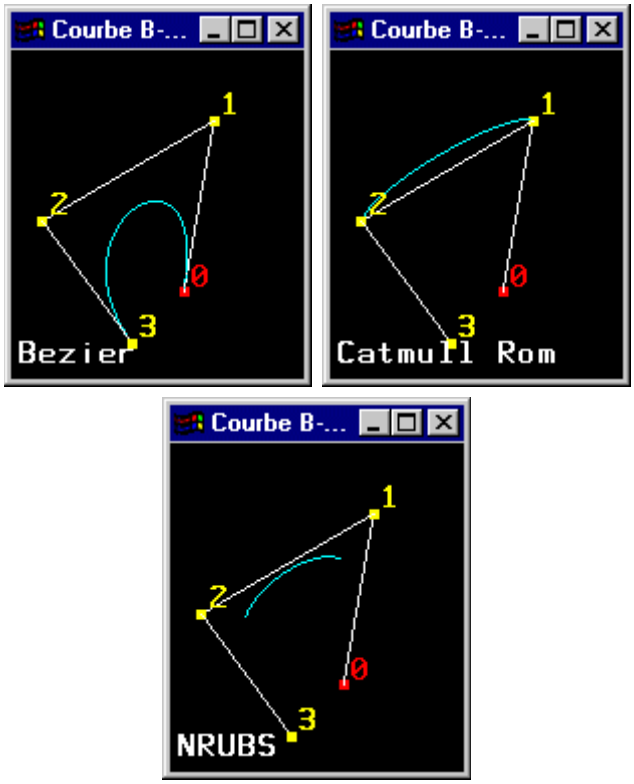
$$\begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix} \begin{pmatrix} g_{1x} & g_{1y} & g_{1z} \\ g_{2x} & g_{2y} & g_{2z} \\ g_{3x} & g_{3y} & g_{3z} \\ g_{4x} & g_{4y} & g_{4z} \end{pmatrix} \quad (2)$$

La matrice G permet de représenter 4 contraintes géométriques s'appliquant à la courbe. Il s'agit habituellement de la position de 4 points dans l'espace de représentation. La courbe générée évoluera "entre" ces quatre points.

La matrice M permet d'attribuer des "poids" respectifs à chacun des coefficients t^i s'appliquant à une contrainte géométrique, permettant ainsi de définir la "forme" de la courbe.

De bons choix de matrice M permettront d'obtenir des tracés de courbe approximant la ligne polygonale à 4 sommets définie par les quatre points présentant différentes caractéristiques.

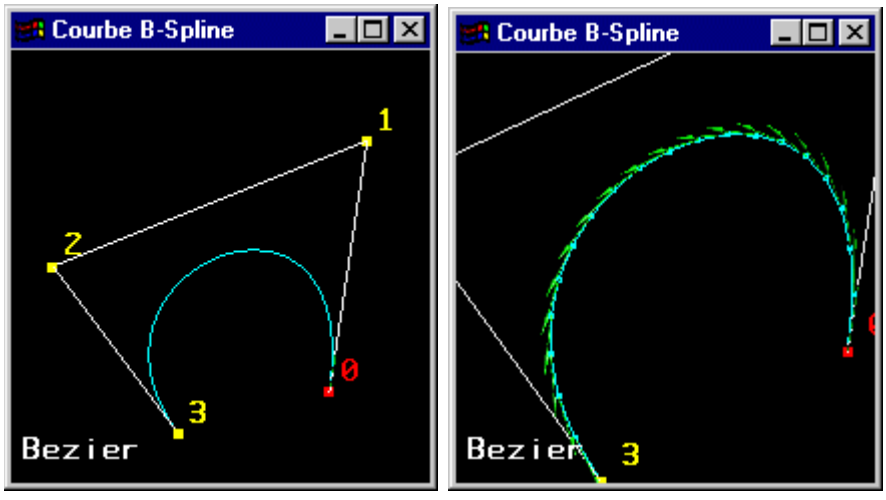
Trois types de courbes paramétriques cubiques



Le programme GLUt

Exécutable GLUt

Tangente à une courbe paramétrique cubique



Le programme GLUt

Exécutable GLUt

Propriétés, applications

Les courbes paramétriques cubiques sont souvent utilisées pour programmer des fonctions de lissage de lignes brisées.

Les propriétés de continuité, de continues dérivabilités première et seconde sont primordiales dans ce cadre car elles assurent la possibilité de se déplacer uniformément sur tous les points de la courbe.

Elles assurent une vitesse de déplacement continue (pas de déplacement instantané) et une accélération du déplacement continue (pas de saute de vitesse instantanée) quand on fait varier t uniformément.

Ces propriétés sont importantes en CFAO (domaine d'origine de ces outils mathématiques).

Les courbes paramétriques cubiques possèdent aussi la propriété de ne pas être déformées par translation et rotation de leurs sommets de définition.
En revanche, elles sont modifiées par changement des paramètres de mise en perspective.

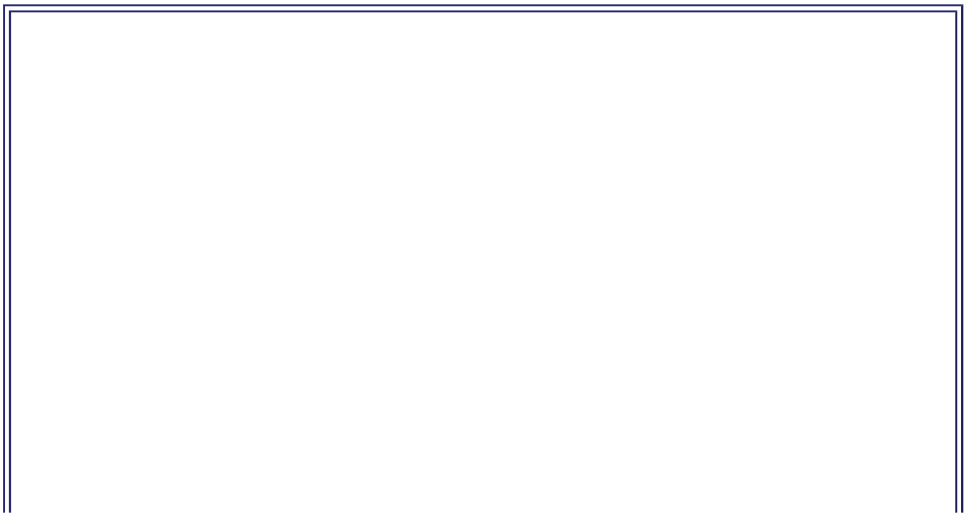
Problème: Ces courbes sont définies sur 4 points de contrôle. Que faire si on en a moins (3) ou plus (5, 6, 7 ou encore plus)?

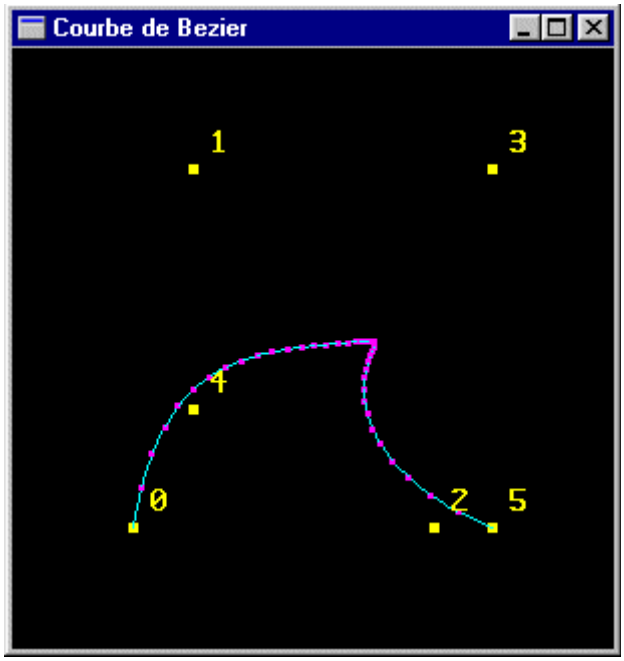
Les courbes de Bézier

Définition

Si $P_0, P_1, P_2, P_3, \ldots, P_n$ sont les n+1 points d'une ligne brisée. La courbe de Bézier définie par ces points est:

$C(t) = \sum_{i=0}^{n} C_i^n t^i (1-t)^{n-i} P_i$ avec le coefficient t compris entre 0.0 et 1.0. On rappelle que $C_i^n = \frac{n!}{i!(n-i)!}$.





Le programme Aux	Le programme GLUt
Exécutable Aux	Exécutable GLUt

Propriétés

Une telle courbe

- commence par le premier point, finit par le dernier point et ne passe pas obligatoirement par les autres points,
- est tangente au segment P_0,P_1 en P_0 et au segment P_n,P_{n-1} en P_n,
- n'est pas déformée par translation et rotation,
- est déformée par changement des paramètres de mise en perspective.

Problèmes

Quoique pouvant être optimisé, le calcul des C_i^n n'est pas économique en temps.

Les deux mises à la puissance en t et en 1-t ne sont pas économiques en temps.

La courbe est intégralement modifiée quand on modifie la position d'un seul point de contrôle. Ceci a pour conséquence l'obligation de la réévaluer intégralement. Il n'y a donc pas de contrôle local sur la forme de la courbe.

Lorsqu'une courbe de Bézier est tracée avec beaucoup de sommets, les sommets intermédiaires sont peu "attractifs". La courbe reste donc loin d'eux. Il n'y a donc pas de fort contrôle de la forme de la courbe.

Algorithme

```
struct coord_3D {
  GLfloat x,y,z ; } ;
struct polygone {
  int n ;
  coord_3D *p ; } ;

void pixel(float x,float y,float z) {
  glVertex3f(x,y,z);
}

double power(double v,int p) {
  return(( !v && !p ) ?
    1.:
    pow(v,(double) p)) ;
}

void bezier(polygone *p,int n) {
  int i,j ;
  float t,mt ;
  float *cn,x,y,z,fac ;
  cn =(float *) calloc(p->n,
                       sizeof(float));
  cn[0] = 1 ;
  cn[1] =(float) (p->n-1) ;
  for ( i = 2 ; i < p->n ; i++ )
    cn[i] = cn[i-1] * (p->n - i) / i ;
  for ( i = 0 ; i < n ; i++ ) {
    t =(float) i/(n-1) ;
    mt = 1-t ;
    x = y = z = 0.0F ;
    for ( j = 0 ; j < p->n ; j++ ) {
      fac = cn[j]*(float) power(t,j)*
            (float) power(mt,p->n-1-j);
      x += fac * p->p[j].x ;
      y += fac * p->p[j].y ;
      z += fac * p->p[j].z ; }
    pixel(x,y,z) ; }
  free(cn) ;
}
```

Cas particulier

Si n = 3 (4 sommets), C(t) peut être présentée sous la forme d'une courbe paramétrique cubique définie par les 4 points de contrôle. Les 4 contraintes géométriques G_1, G_2, G_3 et G_4 sont alors les positions de ces 4 points de contrôle.

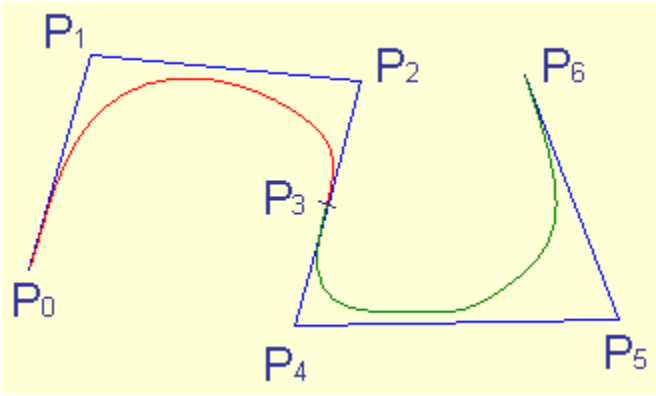
On a alors:

$C(t) = (x(t), y(t), z(t))$

$$= \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_{0x} & P_{0y} & P_{0z} \\ P_{1x} & P_{1y} & P_{1z} \\ P_{2x} & P_{2y} & P_{2z} \\ P_{3x} & P_{3y} & P_{3z} \end{pmatrix}$$

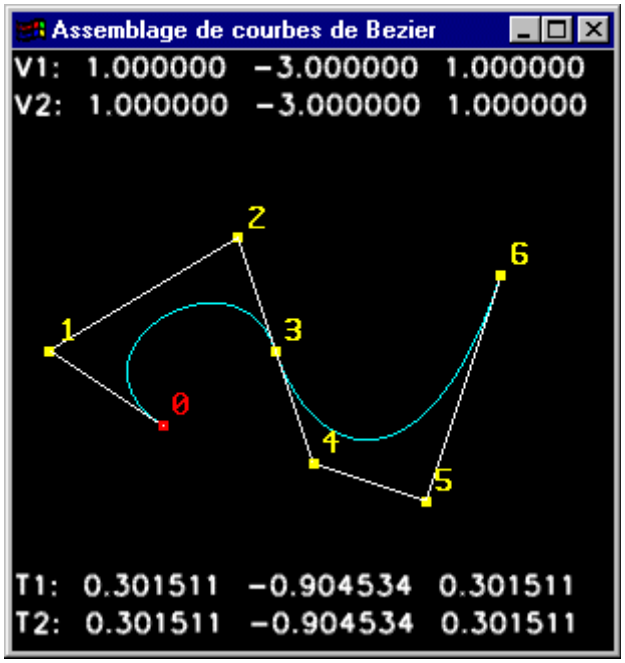
Problème: Acquérir un contrôle local et obtenir un bon lissage avec beaucoup de points de contrôle.

Solution: Relier deux courbes de Bézier définies sur quatre sommets tels que le dernier sommet définissant la première courbe soit aussi le premier sommet de la seconde. C'est à dire tracer la courbe P_i avec $0 <= i <= 6$ (soit au total 7 sommets différents).



On s'assure que P_2P_3 et P_3P_4 sont colinéaires. Cette propriété assure la continue dérivabilité au point P_3.

Si on veut que la dérivé seconde soit elle aussi continue au point P_3, les distances entre (P_2, P_3) et (P_3,P_4) devront être identiques.



Le programme GLUt

Exécutable GLUt

Application: Tracé de courbes lissées quand le nombre de points de contrôle n'est pas imposé. Avantage : Contrôle local, forme "bien" lissée et contrôlée, ajout possible de points de contrôle.

Les courbes B-Splines non rationnelles uniformes

Problème

Lisser une ligne polygonale quelconque au moyen d'une courbe paramétrique cubique telle que la modification de la position d'un seul sommet ne modifie pas l'intégralité de la courbe.
-> Seule la partie modifiée aura à être recalculée.

Définition

Soit une ligne polygonale de n+1 sommets P_i, 0 <= i <= n avec n >= 3.

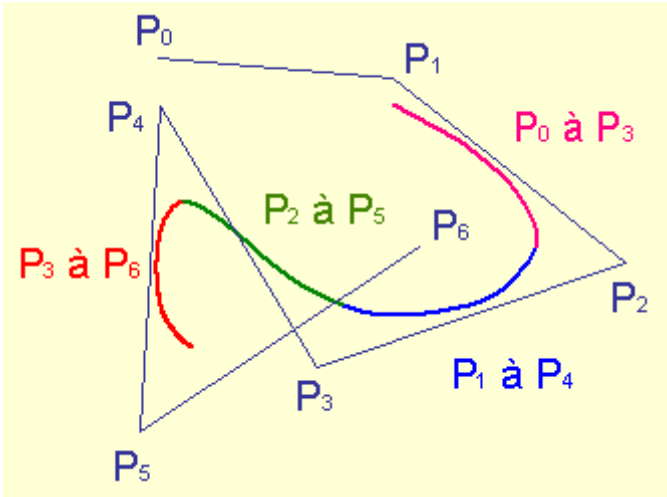
On définit les n-2 vecteurs géométriques $G_i = \begin{pmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{pmatrix}$ avec

0 <= i <= n-3.

Ces n-2 vecteurs géométriques permettent la génération de n-2 morceaux de courbe jointifs qui une fois réunis formeront la courbe approximant le polygone P.

Pour chaque morceau la valeur de t variera obligatoirement uniformément entre 0.0 et 1.0
-> B-Spline uniforme.

La matrice de base utilisée est $\frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}$



Les morceaux de courbe se joignent obligatoirement en leurs extrémités. En effet, soient les deux morceaux de courbe correspondant à G_i et G_{i+1}, le premier se finit à t = 1.0, le second commence à t = 0.0.

En ces points, on calcule que les coordonnées obtenues sont identiques avec pour valeur $\frac{P_{i+1} + 4P_{i+2} + P_{i+3}}{6}$. Cette position est quelque part dans le triangle défini par les trois sommets, plus proche de P_{i+2} que des 2 autres car il s'agit du barycentre des points P_{i+1}, P_{i+2} et p_{i+3} affectés des poids 1.0, 4.0 et 1.0.

Chaque morceau de spline dépend de 4 points de contrôle. Chaque point de contrôle intervient au maximum 4 morceaux de splines.

-> On n'a pas l'obligation de recalculer toute la courbe si un seul point est modifié, mais seulement 4 morceaux au maximum.

Propriétés

Les courbes B-Splines NRU:

- ne passent pas par les sommets de la ligne brisée, y compris le premier et le dernier,
- respectent la continuité, la continue dérivabilité première et la continue dérivabilité seconde.

Une solution simple pour passer par le premier et le dernier point consiste à tripler ces points.

Algorithme

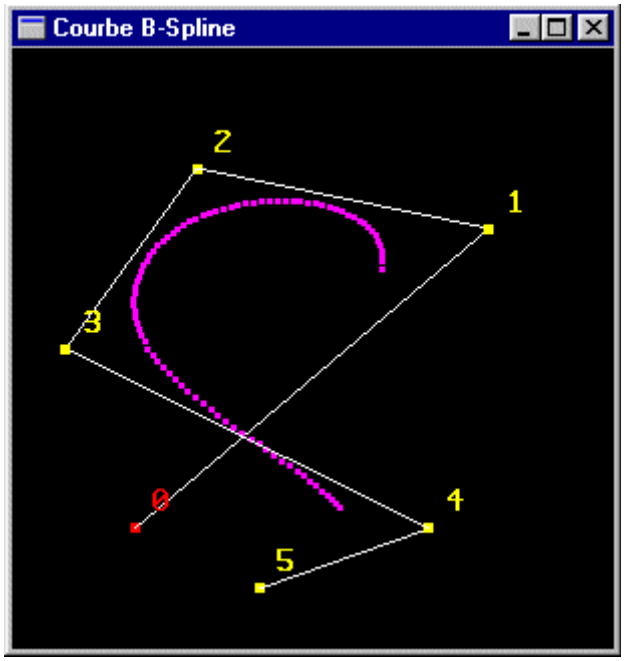
```
struct coord_3D {
  GLfloat x,y,z ; } ;
struct polygone {
  int n ;
  coord_3D *p ; } ;
typedef float matrice[4][4] ;

void pixel(float x,float y,float z) {
  glVertex3f(x,y,z);
}

void lisse(coord_3D *p,int n,matrice m) {
  int j,k ;
  float tt[4],ttt[4],x,y,z ;
  for ( int i = 0 ; i < n ; i++ ) {
    float t =(float) i/(n-1) ;
    tt[0] = t*t*t ;
    tt[1] = t*t ;
    tt[2] = t ;
    tt[3] = 1 ;
    for ( j = 0 ; j < 4 ; j++ ) {
      ttt[j] = 0.0 ;
      for ( k = 0 ; k < 4 ; k++ )
        ttt[j] += tt[k] * m[k][j] ; }
    x = y = z = 0 ;
    for ( j = 0 ; j < 4 ; j++ ) {
      x += ttt[j] * p[j].x ;
      y += ttt[j] * p[j].y ;
      z += ttt[j] * p[j].z ; }
    pixel(x,y,z) ; }
```

```
}

void b_spline(struct polygone *p,
              matrice m,
              int n) {
  for ( int i = 0 ; i < p->n-3 ; i++ )
    lisse(&p->p[i],n,m) ;
}
```



Le programme Aux	Le programme GLUt
Exécutable Aux	Exécutable GLUt

Les courbes Splines de Catmull-Rom

Problème

Lisser une ligne polygonale par une courbe passant par chacun des sommets de cette ligne.

-> Utilisation des courbes Splines de Catmull-Rom.

Définition

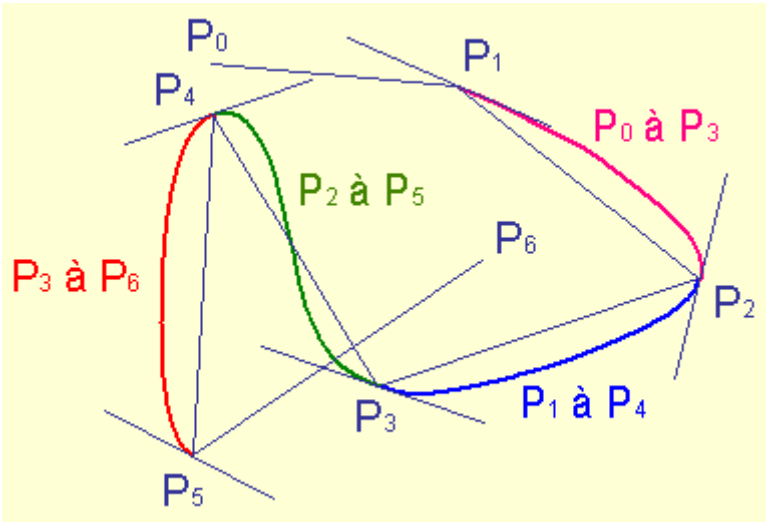
On lisse une ligne polygonale de n+1 sommets P_i, 0 <= i <= n

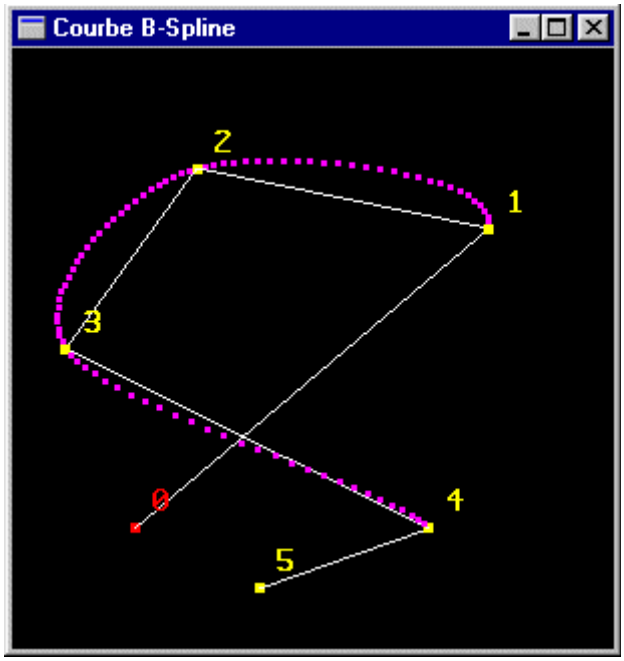
avec n >= 3. Soient les n-2 $G_i = \begin{pmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{pmatrix}$, 0 <= i <= n-3.

$$\frac{1}{2}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix}$$ est la matrice de base utilisée.

Pour le vecteur géométrique G_i, la courbe commence pour t = 0.0 en P_{i+1}, et finit pour t = 1.0 en P_{i+2}.

-> Tous les morceaux se rejoignent en leurs extrémités pour des intervalles [0.0,1.0]. Ces extrémités sont les points de contrôle.





Le programme Aux	Le programme GLUt
Exécutable Aux	Exécutable GLUt

Propriétés

Les courbes splines ainsi définies:

- passent par tous les sommets de la ligne brisée, à l'exception du premier et du dernier,
- commencent au deuxième point et finissent à l'avant dernier,
- sont tangentes au point P_i à la direction donnée par le segment P_{i-1},P_{i+1},
- respectent la continuité, les continues dérivabilités première et seconde.

Une solution simple pour passer par le premier et le dernier point consiste à doubler ces points.

Les courbes B-Splines non uniformes non rationnelles

Ces courbes sont mathématiquement équivalentes aux B-Splines uniformes non rationnelles à la seule différence que les intervalles de variabilité pour les morceaux de courbe ne sont pas forcément tous [0.0,1.0].

Cette modification entraîne une plus grande complexité car il est plus difficile de s'assurer des propriétés caractéristiques qui nous sont nécessaires (dérivabilités). Obtenir cette assurance a fréquemment pour conséquence la duplication de points de contrôle ou l'apparition de nouveaux points de contrôle.

L'intérêt de ces B-Splines est par exemple la possibilité d'ajouter des points de contrôle sans modifier la courbe initiale.

Les courbes B-Splines non uniformes rationnelles: les NURBS

Une B-Spline non rationnelle est calculée au moyen de points de contrôle. Si elle est visualisée en perspective, la même courbe devra être modélisée en cas de modification des paramètres de mise en perspective.
Il n'est pas possible de placer les points de contrôle en perspective et de calculer la B-Spline ensuite. On devra d'abord calculer la B-Spline pour ensuite faire la mise en perspective sur l'ensemble des points obtenus.

Faute de cet ordre de manipulation, la B-Spline obtenue n'est pas invariante par modification des paramètres de mise en perspective avec pour conséquence des problèmes de visualisation.
Cet ordre de calcul est inefficace en temps de calcul car le nombre de sommets de la courbe lissée est plus grand que le nombre de points de contrôle.

Solution: Utiliser les B-Splines non-uniformes rationnelles.

Définition mathématique

Courbe NURBS: $Q(t) = \left(\frac{x(t)}{w(t)} \quad \frac{y(t)}{w(t)} \quad \frac{z(t)}{w(t)} \right)$ avec x(t), y(t), z(t) et w(t) des équations paramétriques cubiques représentant des coordonnées homogènes.

Les surfaces paramétriques bicubiques

Les surfaces paramétriques bicubiques sont une généralisation aux surfaces des courbes paramétriques cubiques.

Définition

Soit une courbe cubique Q(s) = S.M.G (s et S sont équivalents à t et T). Le vecteur géométrique G est constant.

Si les composantes de ce vecteur sont elles-mêmes des cubiques. On obtient:

$Q(s,t) = S.M.G(t) = S.M.\begin{pmatrix} G_1(t) \\ G_2(t) \\ G_3(t) \\ G_4(t) \end{pmatrix}$ avec t compris entre 0.0 et 1.0.

Les G_i sont des cubiques.

-> $G_i(t) = T.M'.P_i$ avec $P_i = (P_{i1},P_{i2},P_{i3},P_{i4})^T$,

-> $G_i(t) = (T.M'.P_i)^T = P_i^T.M'^T.T^T$

$= (P_{i1},P_{i2},P_{i3},P_{i4}).M'^T.T^T.$

Chaque point P_{ij} est un point de contrôle de Gi(t).

-> 16 points P_{ij}

$Q(s,t) = S.M.\begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{pmatrix}.M'^{T}.T^{T}$ avec s et t compris entre 0.0 et 1.0.

Propriétés

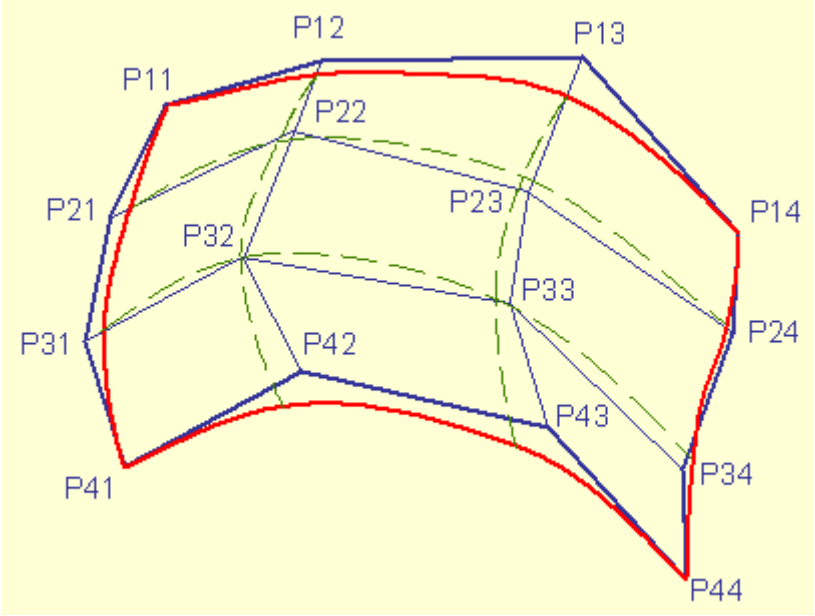
La modification d'un seul des 16 points de contrôle entraîne la modification de l'ensemble de la surface.

Le choix des matrices M et M' permet d'adapter les caractéristiques de la surface tracée aux spécifications désirées.
Ces matrices pourront être celles décrites précédemment dans le chapitre (Bézier, NRU, Catmull-Rom) ou d'autres matrices.
Elles pourront être identiques ou bien différentes l'une de l'autre pour obtenir certaines caractéristiques selon un axe et des caractéristiques différentes selon l'autre axe.

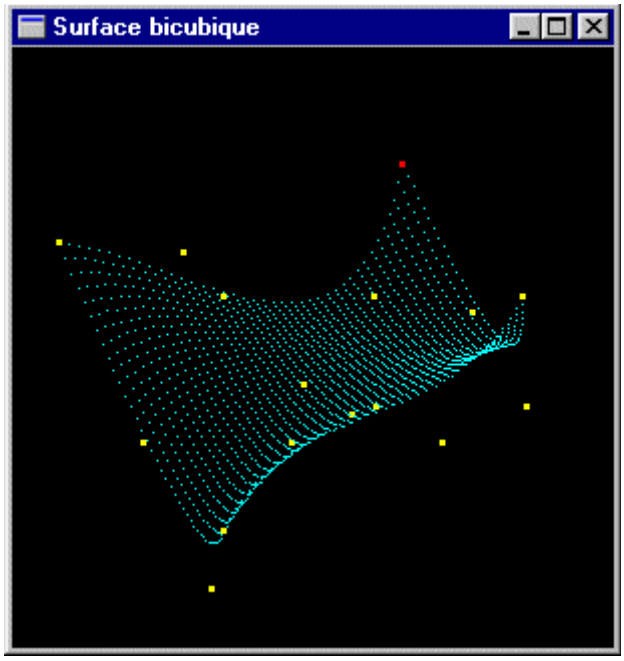
Interprétation intuitive

Une interprétation intuitive de ces surfaces consiste à considérer qu'une telle surface est générée à partir d'une matrice carrée de 4x4 points de contrôle.

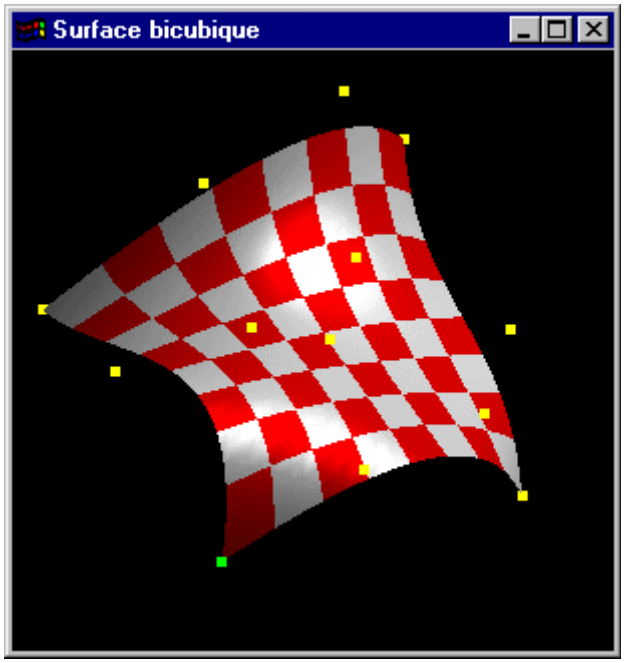
Les 4 lignes de cette matrice permettent de générer m séries de 4 points de contrôle, chacune de ces m séries permettant de générer n points modélisant la surface, soit donc au total m*n points. Ces m*n points définissent un maillage quadrangulaire traçable.



Surface obtenue avec M et M' matrices de Bézier.

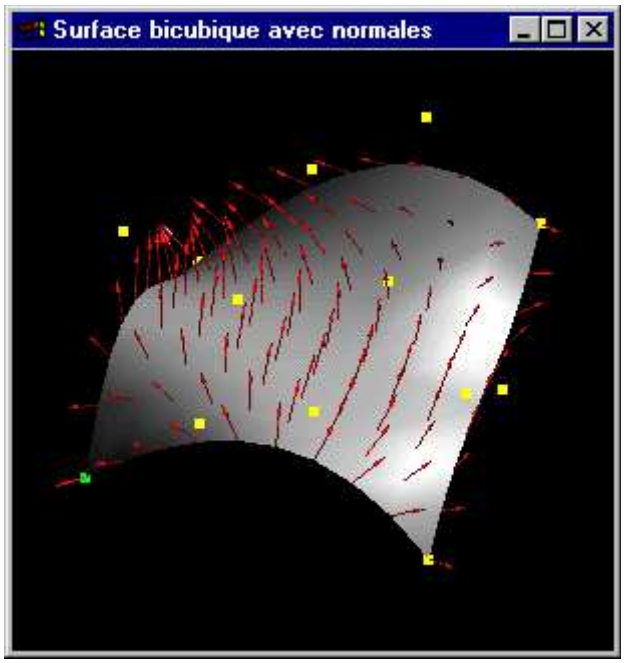


Le programme Aux	Le programme GLUt
Exécutable Aux	Exécutable GLUt



Le programme GLUt

Exécutable GLUt



Le programme GLUt

Exécutable GLUt

Les surfaces quadriques

Définition

On appelle surface quadrique une surface d'équation:

$f(x,y,z) = ax^2 + by^2 + cz^2 + dxy + eyz + fxz + gx + hy + iz + j = 0.0$

Cette famille de surfaces est très vaste.

Il est possible de représenter des plans, des sphères, des paraboloïdes, des ellipsoïdes, ...

Les raisons qui pourraient faire utiliser les surfaces quadriques sont multiples:

- Calcul facile de la normale en un point à la surface
- Test facile de la présence d'un point sur la surface
- Calcul facile d'une des coordonnées si les deux autres sont connues
- Calcul possible de l'intersection d'une surface quadrique avec une autre surface quadrique

Exemples

- Plans: a = b = c = d = e = f = 0.0
  -> $f(x,y,z) = gx + hy + iz + j = 0.0$

- Ellipsoïdes: d = e = f = g = h = i = 0, j <= 0.0
  -> $f(x,y,z) = ax^2 + by^2 + cz^2 + j = 0.0$

Conclusion

Les surfaces et courbes lissées ont été initialement conçues comme outils de CFAO pour la modélisation d'objets.

Plus récemment, l'informatique graphique s'est emparée de ces objets mathématiques comme outils de modélisation pour représenter les objets des scènes autrement que par ensembles de facettes ou en tant qu'objets canoniques volumiques de type sphère, cube, ...

Les outils de haut niveau en modélisation pour la synthèse d'images font appel aux NURBS (Non Uniform Rational B-Splines) qui sont une évolution des NRUBs pour permettre un affichage en perspective.