Représentation des courbes et des surfaces

Contenu

Les courbes et les surfaces sont présentes dans la majorité dans des applications faisant intervenir des images. La plupart des scènes réelles ou virtuelles sont composées, partiellement ou totalement, de courbes ou de surfaces.

Le besoin de représenter les courbes et les surface se présentent dans deux situations principales :

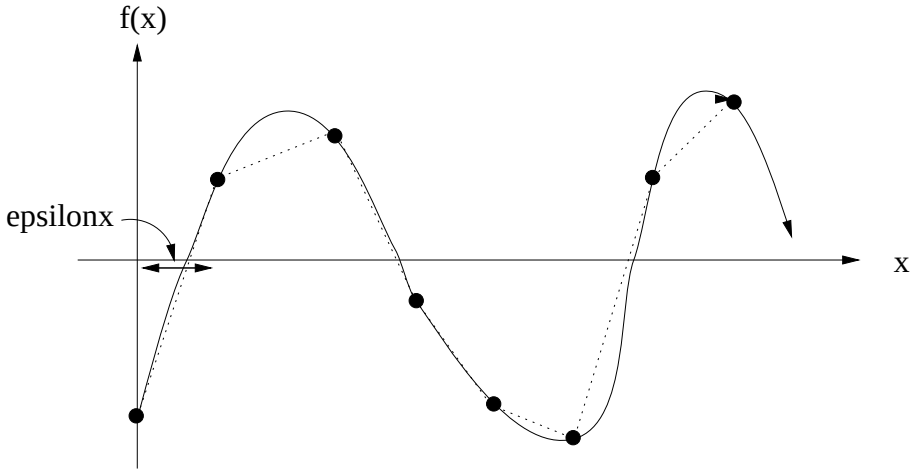
1. Modélisation d'un objet virtuel, à partir d'un modeleur et sur la base d'une description mathématique ou de manière interactive.
2. Modélisation d'un objet existant à partir de données issus de mesure, de capteurs ou d'images, en général des points de l'espace. L'objet est modélisé par approximation par des plans, des sphères ou d'autres modèles mathématiques simples.

Dans cette partie nous allons aborder le problème de la représentation de ces primitives, problème qui se présente aussi bien en synthèse d'images qu'en traitement d'images.

1 Modèles polygonaux

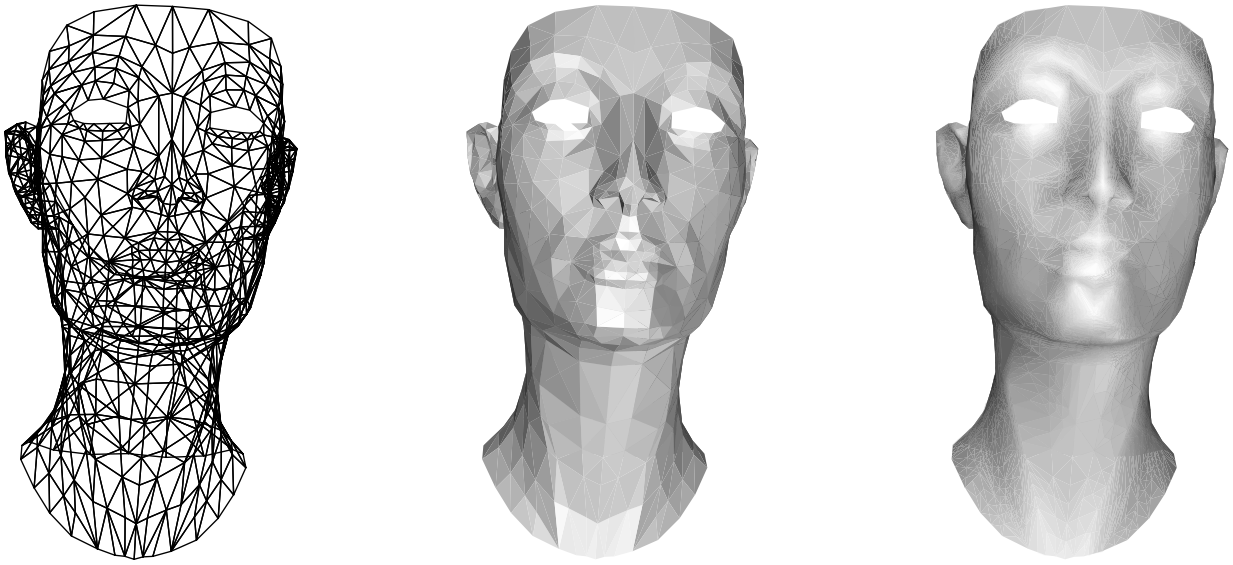
Ce type de représentation consiste à ne stocker qu'un ensemble discret de point de la courbe (surface) et à approximer la courbe (surface) entre ces points, par des segments de droites (facettes polygonales). Ces représentations sont extrêmement répandues en raison de leurs simplicité.

Exemple : approximation polygonale d'une courbe



```
Algorithme :
pour x de 0 a xmax, epsilonx
        tracer(x,f(x), x+epsilonx, f(x+epsilonx, valeur))
finpour
```

Dans le cas de surface, on parle de *maillage polygonale* : une collection de sommets, côtés et polygone telle qu'un côté est partagé par deux polygones au plus. Le problème qui se pose au programmeur est ici de choisir une représentation adéquate, pour le stockage externe (dans un fichier) et interne (en mémoire). Quelles informations stockées et comment ?



(a) (b) (c)

Figure 1: *Exemple de modèle polygonal (facettes triangulaires) : (b) ombrage plat, (c) ombrage de Gouraud.*

Plusieurs représentations sont possibles, le choix se fait en fonction des besoins et considérant les coûts en espace (représentations externes et internes) et en temps (représentations internes). Les opérations typiques sur les maillages sont :

- trouver les côtés incidents à un sommet,
- trouver les voisins d'un sommet,

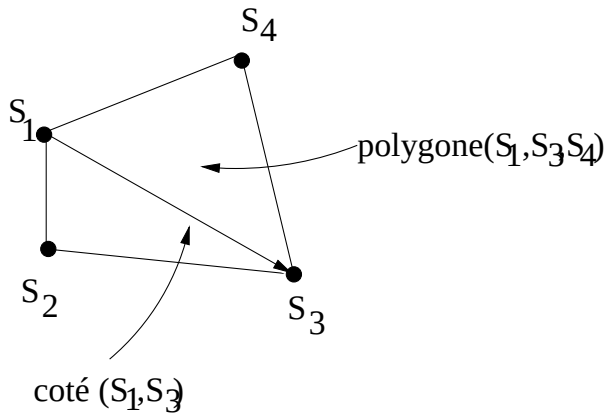
- trouver les polygones partageant un côté ou un sommet,
- trouver les côtés d'un polygone,
- ...

L'idéal consiste a stocker en interne le maximum d'informations pour éviter des calculs répétitifs, mais cela nécessite un espace mémoire important.

☞ Compromis entre coût mémoire et coût en temps pour les représentations internes.

1.1 Représentations internes

Représentation explicite



Un polygone est représenté par une structure :

$$P = \{(x_1, y_1, z_1), (x_3, y_3, z_3), (x_4, y_4, z_4)\},$$

où deux sommets successifs représentent un côté, ainsi que le dernier est le premier sommet.

☞ Mauvaise utilisation de l'espace : les sommets partagés sont dupliqués,

☞ Difficulté pour établir des relations (côtés incidents, ...).

Pointeurs sur une liste de sommets

Les sommets sont stockes dans une table :

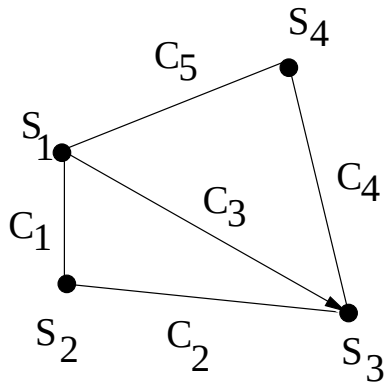
$$S = \{S_1, S_2, ..., S_n\} = \{(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n)\},$$

Un polygone est ensuite défini par une liste d'indices (ou de pointeurs) sur des sommets :

$$P = \{(1,2,3),(1,3,4),...\}.$$

- ☞ Les sommets ne sont stockes qu'une fois.
- ☞ difficulté pour établir des relations.

Pointeurs sur une liste de côtés



Les sommets sont toujours stockes dans une liste :

$$S = \{S_1, S_2, ..., S_n\} = \{(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n)\}.$$

Les côtés sont eux aussi stockes dans une table et sont définis par deux indices (pointeurs) sur des sommets et deux indices (pointeurs) sur des polygones :

$$C = \{C_1, C_2, C_3, ...\} = \{(\&S_1, \&S_2, \&P_1, -), (\&S_2, \&S_3, \&P_1, -), (\&S_1, \&S_3, \&P_1, P_2), ...\},$$

et enfin les polygones qui sont définis par une liste d'indices (pointeurs) sur des côtés :

$$P = \{P_1, P_2, ...\} = \{(\&C_1, \&C_2, \&C_3), (\&C_3, \&C_4, \&C_5), ...\}.$$

- ☞ utilisation optimale de l'espace mémoire : seuls des indices sont manipulés,
- ☞ les polygones partageant un côté sont identifiés,
- ☞ difficile de connaître les voisins d'un sommets donné, les côtés incidents à un sommet, ...
- ☞ stockage d'informations supplémentaires si nécessaire.

1.2 Représentations externes

Il s'agit ici de stocker, dans un fichier, le nombre d'informations minimum permettant de reconstituer la surface. Ils existent différent formats de fichiers existants pour cela, par exemple le format *off* utilisé par le logiciel *geomview* :

```
OFF              #Debut du fichier
Nb_sommets Nb_polygones Nb_cotes   #nombres entiers
#commentaire
#les coordonnées des sommets (nombres réels)
x1 y1 z1
x2 y2 z2
.
.
#pour chaque polygone : nb_sommets, indice de chaque sommet
#                       dans la liste precedente
3 2 1 3
4 20 12 11 65
.
.
```

- ☞ format minimal mais efficace et simple à gérer.
- ☞ il est possible de rajouter des informations pour chaque sommet : normale à la surface, couleurs.

Un autre format qui est un standard, VRML (Virtual Reality Modeling Language). Les sommets et les polygones sont aussi stockes sous forme de liste :

```
 Coordinate3 {
              point   [ -2.250000 3.110000 -0.350000,
                        -2.170000 3.070000 -0.520000,
                        ...
                      ]
             }
 IndexedFaceSet {
               coordIndex [ 0, 1, 2, 2, -1,
                            3, 2, 4, 4, -1,
                            ...
                          ]
                }
```

- Nécessite en plus la description des propriétés de la surface.
- Format complet, beaucoup de possibilités : animation, ... mais plus difficile à gérer (lecture, écriture).
- Format de description de scènes 3D.

2 Courbes paramétriques

Les approximations polygonales sont des approximations linéaires par morceaux (1er degré). Elles nécessitent un nombre de points importants pour obtenir une bonne précision. Une autre approche consiste donc à utiliser des modèles d'ordre supérieurs pour représenter les courbes et surfaces. Les représentations paramétriques répondent a ce besoin et apportent un gain en espace mémoire ainsi qu'en possibilité de manipulations.

Les courbes paramétriques :

$$\begin{cases} x = x(t) \\ y = y(t) \\ z = z(t) \end{cases}$$

On utilise en général des courbes de degré 3 (cubiques). Dans le contexte du traitement ou de la synthèse d'images, nous cherchons à interpoler ou approximer une courbe à partir d'un nombre fini de points. On utilise pour cela les courbes de la forme :

$$P(t) = (x(t), y(t), z(t)) = \sum_{i=0}^{n} f_i(t)\ P_i,$$

où $f_i(t)$ sont des fonctions de pondérations, $P_i = (x_i, y_i, z_i)$ sont les points de données. Cette définition englobe une famille de courbes dont en particulier :

1. les courbes de Béziers (approximation),
2. les B-splines (approximation),
3. les Catmull-Rom (interpolation).

2.1 Les courbes de Béziers

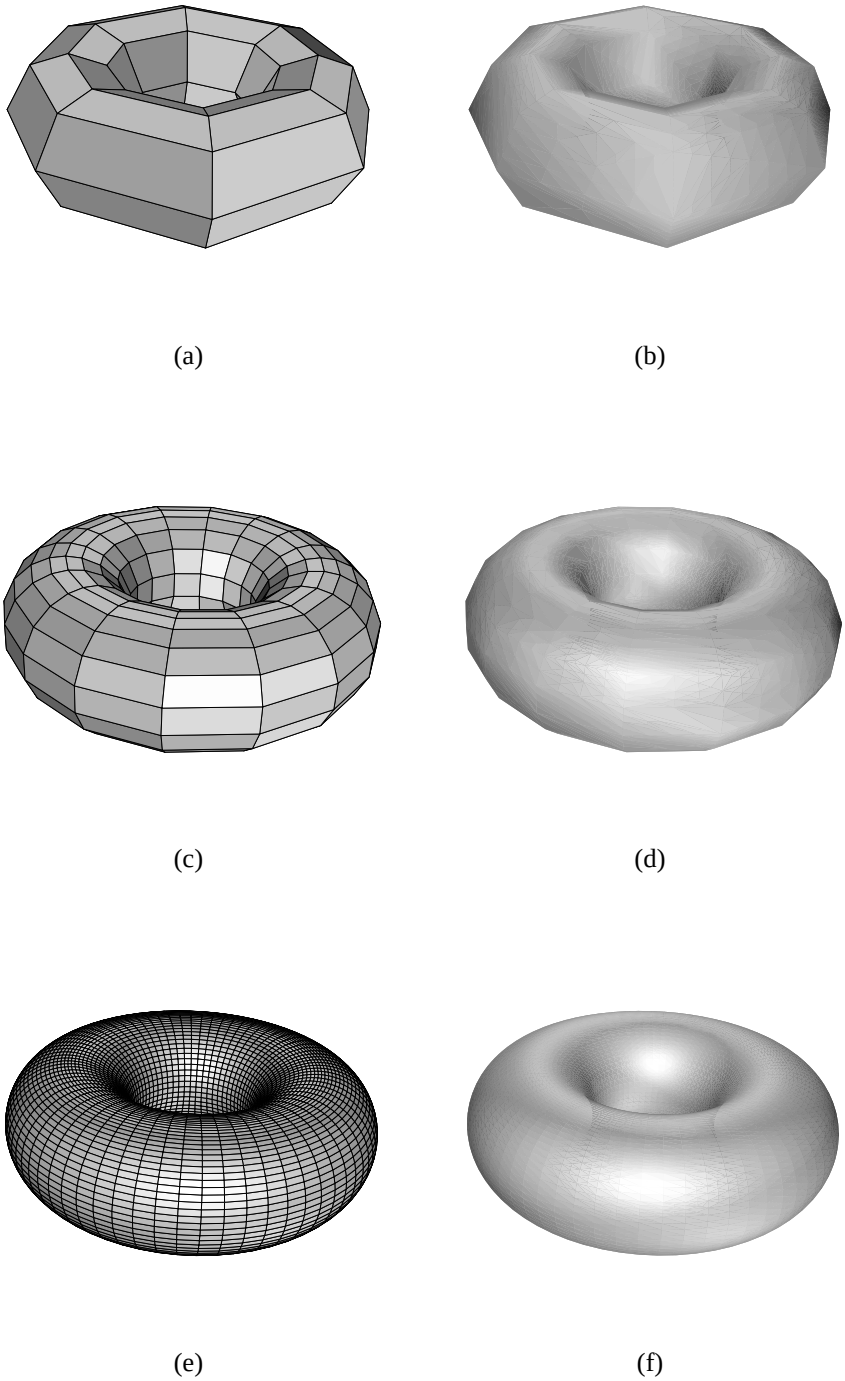
Du nom d'un ingénieur de Renault (Pierre Bézier) qui développa ce modèle de courbes pour la conception de carrosseries de voitures.
Définition :

$$P(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i,$$

avec :

$$B_{i,n} = C_i^n t^i (1-t)^{n-i}, \quad C_i^n = \frac{n!}{i!(n-i)!}.$$

où :



(a) (b)

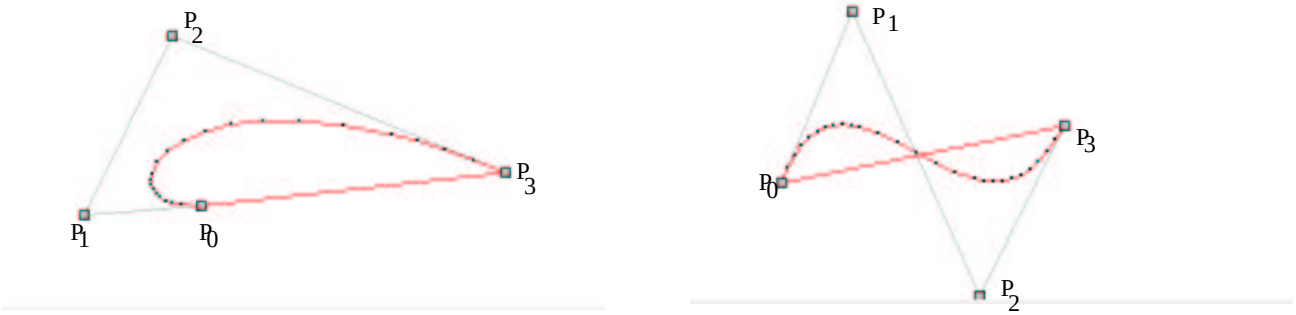
(c) (d)

(e) (f)

Figure 2: Le tore par surface de Beziers et à différents niveaux de subdivisions : (a)-(b) : patchs de 5x5, (c)-(d) : patchs de 10x10, (e)-(f) : patch de 40x40.

- $B_{i,n}$ sont les polynômes de Bernstein,
- P_i sont les points contrôle,
- n est le nombre de points de contrôle utilisés (degré de la courbe + 1).
- le paramètre t varie de 0 à 1.

Exemples : 4 points de contrôles, degré de l'approximation 3 -> courbes de Bézier cubiques



(a) (b)

☞ la courbe ne passe pas, en général, par les points de contrôle à l'exception du premier et du dernier.

☞ la courbe est toujours contenue dans l'enveloppe convexe des points de contrôle.

☞ une courbe fermée peut-être générée en prenant le même point comme premier et dernier point de contrôle.

☞ pour deux points de contrôle, l'approximation correspond à une interpolation linéaire.

$$P(t) = (1-t)P_0 + tP_1,$$

☞ pour trois points de contrôle :

$$P(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2,$$

☞ pour quatre points de contrôle (Bézier cubiques) :

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3,$$

soit sous forme matricielle :

$$P(t) = (t^3, t^2, t, 1) \cdot \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

$$P(t) = T^t \cdot M_{bezier} \cdot P.$$

Exemple d'algorithme de dessin : Bézier de degré 3, subdivision de 100.

```
/* Points 2D*/
type struct {
 float x;
 float y;
} XY;

/* Fonction de calcul */
XY Bezier3(XY p1, XY p2,XY p3,XY p4,double t)
{
   double tm1,tm13,t3;
   XY p;

   t1 = 1 - t;
   t13 = t1 * t1 * t1;
   t3 = t * t * t;

   p.x = t13*p1.x + 3*t*t1*t1*p2.x + 3*t*t*t1*p3.x + t3*p4.x;
   p.y = t13*p1.y + 3*t*t1*t1*p2.y + 3*t*t*t1*p3.y + t3*p4.y;

   return(p);
}
/* Dessin */
void main()
{
  double t;
  XY p, p_prec;
```

```
  p_prec = p1;

  for(i = 0; i < 100 ; i++){
    t = i / 100.0;
    p = Bezier4(p1,p2,p3,t);
    DrawLine(p_prec.x, p_prec.y,p.x,p.y);
    p_prec = p;
  }
}
```

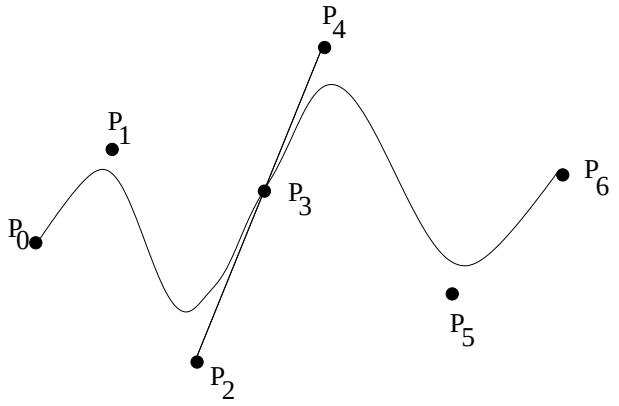
Pour n points de contrôle, n grand
Pour tracer des courbes avec beaucoup de points de contrôle, on utilise en général des Béziers cubiques que l'on colle bout à bout. Le problème est alors de vérifier la continuité aux extrémités des segments de courbes collés.

Tangente en P_3 d'une Bézier cubique :

$$P'(1) = 3(P_3 - P_2).$$

Donc pour assurer une continuité C^1 entre deux segments de courbes de Bézier, il faut que :

$$(P_3 - P_2) = (P_4 - P_3),$$



2.2 Les B-splines

Les B-splines cubiques (splines = lattes de jardinier) sont des courbes polynomiales cubiques de continuité C^2. Elles approximent un ensemble de points de contrôle P_i, $i \in [0, m]$ avec une courbe constituée de $m-2$ segments de courbes polynomiaux Q_i, $i \in [3, m]$. Chaque segment de courbes est fonction d'un paramètre t variant de t_i a t_{i+1} et de 4 points de contrôle. Le segment Q_i est donc défini par :

$$P_{i-3}, P_{i-2}, P_{i-1}, P_i$$

pour t variant de t_i a t_{i+1}.

☞ Un point de contrôle affecte 4 segments de courbes.

On distingue deux types de courbes B-splines :

1. Les B-splines uniformes.
2. Les B-splines non-uniformes.

Les B-splines cubiques uniformes

Les paramètres t_i sont uniformément répartis sur l'ensemble des points de contrôle, à savoir :

$$t_i = t_{i-1} + 1.$$

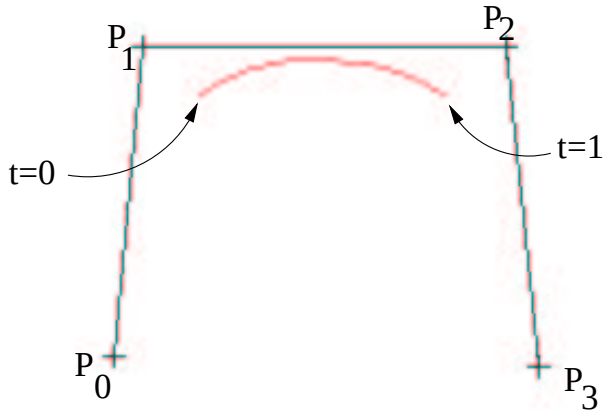
donc t varie de 1 pour un segment de courbe et on effectue le changement de variable $t = t - t_i$. La position d'un point P est alors définie par :

$$P(t) = \frac{1}{6}[(1-t)^3 P_{i-3} + (3t^3 - 6t^2 + 4)P_{i-2} + (-3t^3 + 3t^2 + 3t + 1)P_{i-1} + t^3 P_i],$$

pour : $0 \leq t < 1$.
Soit sous forme matricielle :

$$P(t) = (t^3, t^2, t, 1) \cdot \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{pmatrix}$$

$$P(t) = T^t \cdot M_{bspline} \cdot P.$$



Continuité aux points extrémités :

- Dérivée première :
  1. en $t = 0$: $\frac{d}{dt}P(t) = (P_{i-1} - P_{i-3})/2$,
  2. en $t = 1$: $\frac{d}{dt}P(t) = (P_i - P_{i-2})/2$.
- Dérivée seconde :
  1. en $t = 0$: $\frac{d^2}{dt^2}P(t) = P_{i-3} - 2P_{i-2} + P_{i-1}$,
  2. en $t = 1$: $\frac{d^2}{dt^2}P(t) = P_{i-2} - 2P_{i-1} + P_i$.

Propriétés des B-splines uniformes :

- ☞ Contrôle local : chaque segment est défini par quatre points de contrôle,
- ☞ La courbe approxime et n'interpole pas les points de contrôle. En jouant sur la multiplicité des points de contrôle, on peut attirer la courbe vers les points de contrôle jusqu'à l'interpolation pour une multiplicité de trois.
- ☞ La courbe est contenue dans l'enveloppe convexe des points de contrôle.
- ☞ La courbe est invariante par transformation affine.
- ☞ La courbe est de continuité C^2 partout.

Les B-splines cubiques non-uniformes

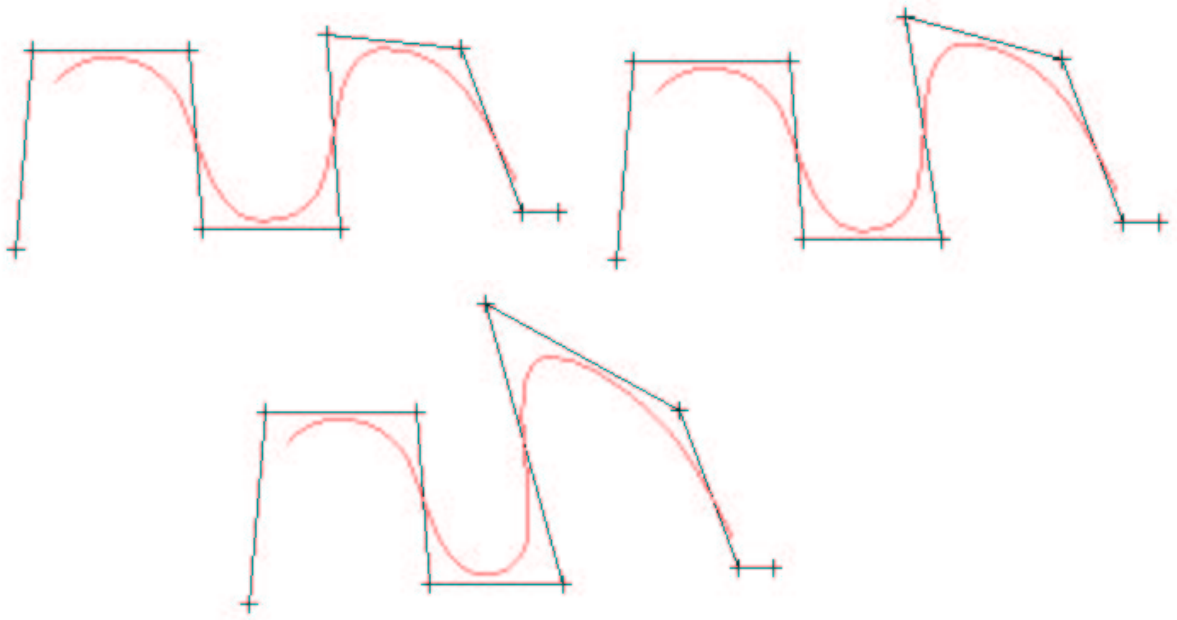


Figure 3: Contrôle local de la courbe.

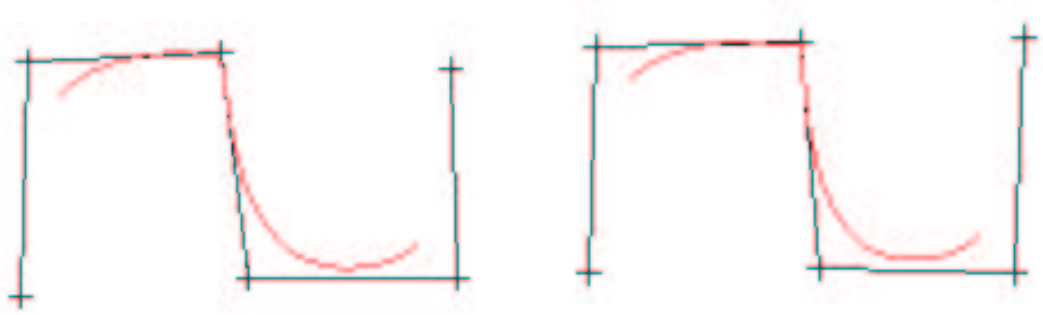


Figure 4: Différentes multiplicités des points de contrôle.

Les paramètres t_i ne sont pas nécessairement répartis de manière uniforme. La position d'un point P de la courbe est définie par :

$$P(t) = N_{i-3,4}(t)P_{i-3} + N_{i-2,4}(t)P_{i-2} + N_{i-1,4}(t)P_{i-1} + N_{i,4}(t)P_i,$$

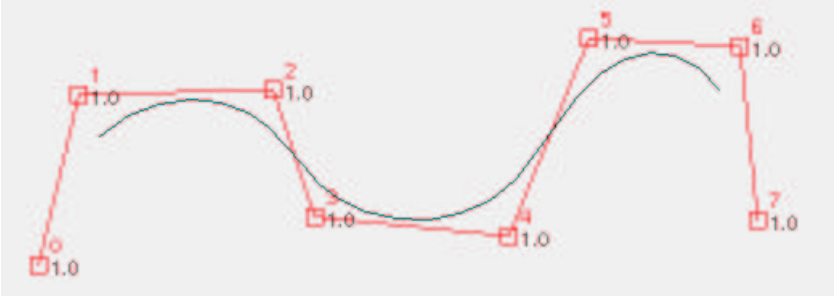
$$3 \leq i \leq m, \quad t_i \leq t < t_{i+1},$$

avec :

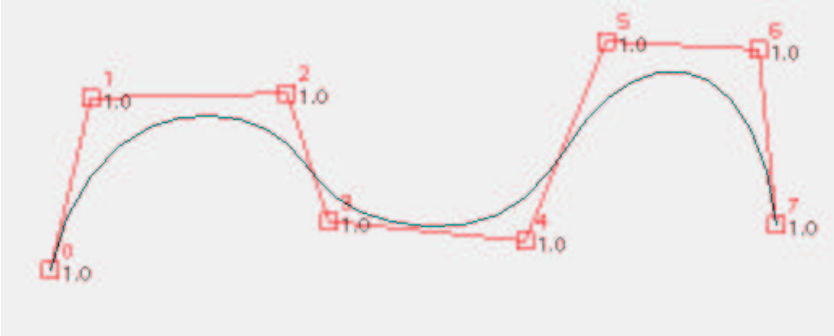
$$N_{i,0} = \begin{cases} 1, & t_i \leq t < t_{i+1}, \\ 0, & \text{sinon} \end{cases}$$

$$N_{i,k} = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t).$$

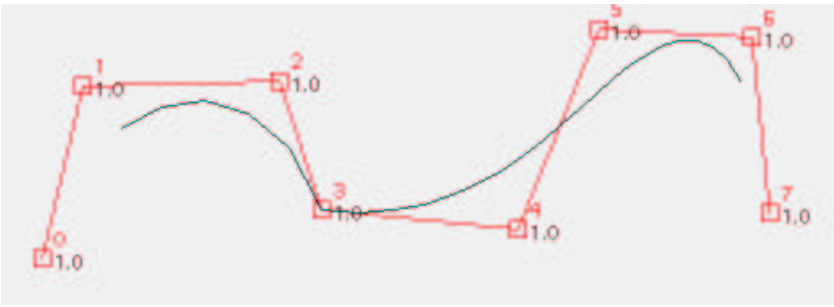
- ☞ Nécessite 4 paramètres t_i de plus que le nombre m de points de contrôle : t_0, t_{m+4}.
- ☞ Contrôle local de la courbe par les paramètres. En particulier en donnant la même valeur à plusieurs paramètres successifs.
  1. Trois paramètres successifs de même valeurs forcent l'interpolation ($t_i = t_{i+1} = t_{i+2}$ forcent l'interpolation par P_{i-1}).
  2. Quatre paramètres successifs de même valeurs provoquent une rupture de la courbe.
- ☞ Pour la suite de paramètres : $[0, 0, 0, 1, 2, 3, 4, 4, 4]$, on retrouve les segments de courbes de Bsplines uniformes.
- ☞ Pour la suite de paramètres : $[0, 0, 0, 0, 1, 1, 1, 1]$, on retrouve les segments de courbes de Bézier.



(a) paramètres (0 0 0 1 2 3 4 5 6 7 7 7)



(b) paramètres (0 0 0 0 2 3 4 5 7 7 7 7)



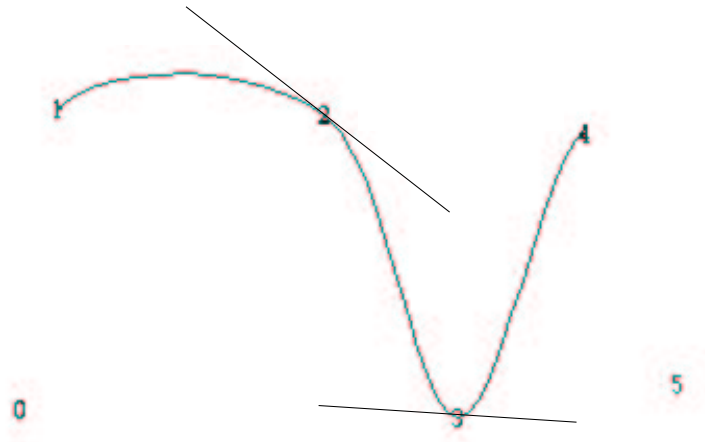
(c) paramètres (0 0 0 1 2 2 2 5 6 7 7 7)

2.3 Autres splines

Il existent plusieurs familles de courbes splines, citons en particulier les **Catmull-Rom** qui sont des courbes d'interpolations des points de contrôle. Les Catmull-Rom interpolent les points P_1, P_{m-1} d'une séquence de points P_0, P_m. La courbe est déterminée par :

$$P(t) = (t^3, t^2, t, 1) \cdot \frac{1}{2} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{pmatrix}$$

$$P(t) = T^t \cdot M_{catmullrom} \cdot P.$$



☞ La tangente à la courbe en un point de contrôle P_i est parallèle a la droite (P_{i-1}, P_{i+1}).

Citons aussi les β**splines** cubiques uniformes, définies de manière similaire sous forme matricielle :

$$P(t) = (t^3, t^2, t, 1) \cdot \frac{1}{2} \begin{pmatrix} -2\beta_1^3 & 2(\beta_2 + \beta_1^3 + \beta_1^2 + \beta_1) & -2(\beta_2 + \beta_1^3 + \beta_1^2 + \beta_1) & 2 \\ 6\beta_1^3 & -3(\beta_2 + 2\beta_1^3 + 2\beta_1^2) & 3(\beta_2 + 2\beta_1^2) & 0 \\ -6\beta_1^3 & 6(\beta_1^3 - \beta_1) & 6\beta_1 & 0 \\ 2\beta_1^3 & \beta_2 + 4(\beta_2^2 + \beta_1) & 2 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{pmatrix},$$

ces courbes permettent un contrôle local plus important au travers des deux paramètres β_1 et β_2. A noter que pour $\beta_1 = 1$ et $\beta_2 = 0$, on retrouve les B-splines cubiques.

2.4 Courbes polynomiales cubiques rationnelles

Les segments de courbes rationnelles cubiques sont des rapports de polynômes :

$$x(t) = X(t)/W(t),\ y(t) = Y(t)/W(t),\ z(t) = Z(t)/W(t).$$

où $X(t), Y(t), Z(t), W(t)$ sont des courbes polynomiales cubiques dont les points de contrôle sont définis en coordonnées homogènes. L'ensemble des définitions précédemment introduites s'appliquent ici en rajoutant une troisième (2D) ou quatrième (3D) coordonnées aux points traités. On peut donc calculer des Bézier rationnelles ou des B-splines rationnelles non-uniformes (NURBS). L'intérêt des courbes rationnelles est double :

1. les courbes rationnelles sont invariantes par rotation, changement d'échelle, translation et projection perspective. Ce qui veut dire qu'une transformation peut-être appliquée aux points de contrôle uniquement.

2. Un deuxième intérêt est que les splines rationnelles (quadratiques) permettent de tracer précisément les coniques sans avoir besoin de nombreux points de contrôle.

3 Surfaces paramétriques

Le raisonnement développé sur les courbes peut s'appliquer aux surfaces. Cette fois-ci deux paramètres interviennent pour la définition d'une surface et les surfaces sont déterminées à partir de grilles (patchs) de points de contrôle.

Exemple : **les surfaces de Bézier**

$$P(s,t) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} B_{i,n}(s) B_{j,m}(t)$$

avec :

- $(m+1)(n+1)$ points de contrôle P_{ij},

- $s, t \in [0,1]$.

Dans le cas de surface de Bézier cubiques, les grilles sont formées de seize points de contrôle et la détermination de la surface peut se faire à l'aide de l'expression

:

$$P(s,t) = (s^3, s^2, s, 1) \cdot \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_0 & P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 & P_7 \\ P_8 & P_9 & P_{10} & P_{11} \\ P_{12} & P_{13} & P_{14} & P_{15} \end{pmatrix} \cdot \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

$$P(s,t) = (s^3, s^2, s, 1) \cdot M_{bezier} \cdot P \cdot M_{bezier}^t \cdot \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

En remplaçant dans l'expression ci-dessus la matrice de Bézier par la matrice B-spline précédemment définie, on obtient une surface B-spline cubique uniforme. On peut de la même manière effectuer une interpolation surfacique à l'aide de la matrice des Catmull-Rom.

- ☞ L'ensemble des définitions sur les courbes s'appliquent aux surfaces.
- ☞ Les surfaces de Bézier interpolent les quatre points extrémités.
- ☞ Les problèmes de continuité sont les mêmes que dans le cas des courbes.

Exercices :

1. Calculez la normale à une surface paramétriques bicubiques.
2. Soit quatre points du plan $P_1, .., P_4$, on cherche à déterminer les quatre points de contrôle tels que le segment de courbes associé a ces points de contrôle (Bézier, B-spline) interpole les quatre points donnés.
   Proposez une solution pour un paramétrage uniforme : $t_{P_1} = 0, t_{P_2} = 1, t_{P_3} = 2, t_{P_4} = 3$.

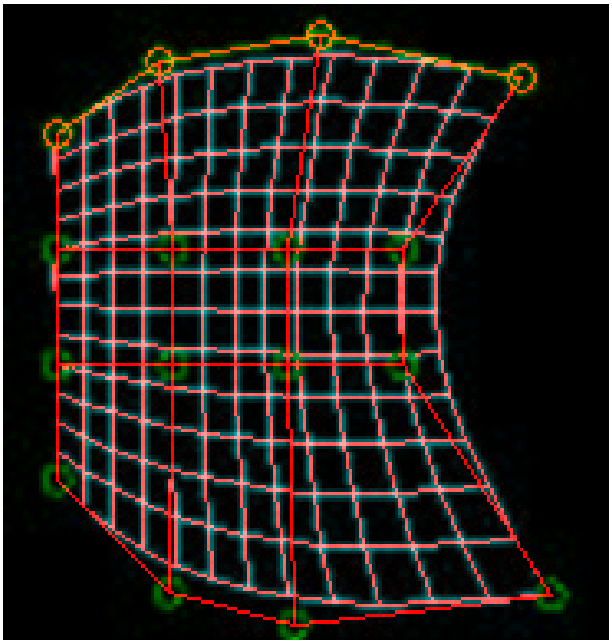


Figure 5: Patch de Bézier

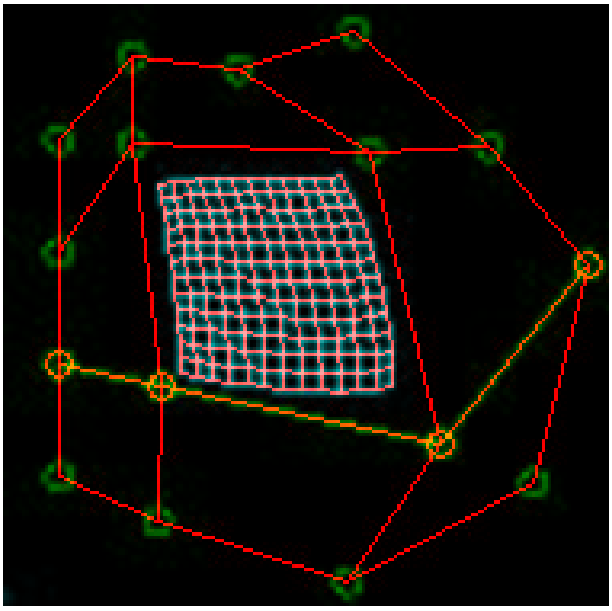


Figure 6: Patch de Bspline

4 Modèles fractals

Pour modéliser des environnements naturels tels que les montagnes, les feuillages ou un ciel nuageux, il est nécessaire de disposer de modèles autres que ceux présentés précédemment. Ces derniers sont, en effet, inefficaces pour représenter des motifs répétés un grand nombre de fois comme c'est le cas pour les décors cités. Les modèles fractals apportent une solution efficace à ce problème.

Le terme fractal, issu du latin *frangere* et originalement proposé par MandelBrot [1982], est utilisé par la communaute graphique pour décrire tout modèle irrégulier et fragmenté pour lequel les irrégularités sont conservées (on parle d'*auto-similarité*). Il s'agit généralement de fonctions récursives utilisant un motif initial et un motif de remplacement. Un objet est dit fractal lorsque le processus de remplacement est appliqué à l'infini pour la création de l'objet.

4.1 Courbes fractales

Pour générer une courbe fractale, on remplace à l'étape i les segments du motif de l'étape (i-1) par le motif de remplacement. Lorsque le processus est répété à l'infini, la courbe est dite auto-similaire : la courbe entière est similaire à une sous-partie d'elle même.

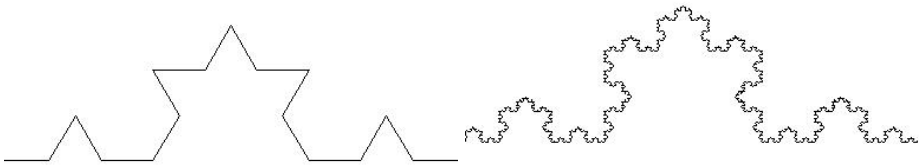


Figure 7: Le flocon de von Koch. La dimension fractale d de la courbe est definit par : $n^{1/d} = f$, où n est le nombre de parties du motif ($n = 4$ ici) et f est le facteur de diminution du motif ($f = 3$ ici). La dimension est donc : $d = \log 4 / \log 3 = 1.26$.

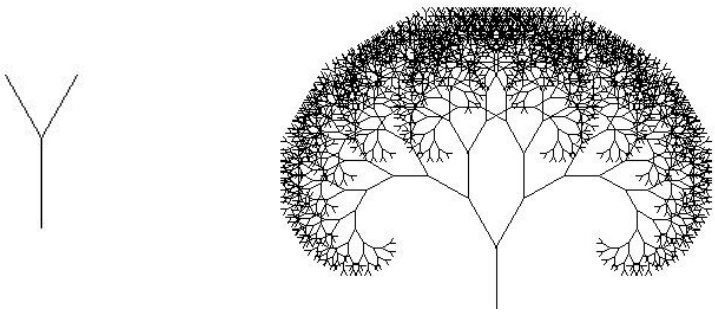


Figure 8: Autre exemple de courbe fractale. (dimension ?)

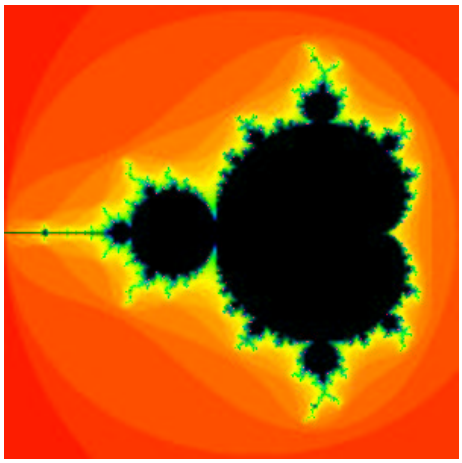


Figure 9: Approximation de l'ensemble de Mandelbrot (en noir dans la figure) : ce sont les points (x, y) pour lesquels la suite complexe : $s_k = s_{k-1}^2 + x + y \times i$, $s_0 = 0$, ne diverge pas, en module, à l'infini.

4.2 Surfaces fractales

Les surfaces sont générées de manière similaires aux courbes. Par exemple, l'algorithme de Fournier-Russel-Carpenter permet de générer des montagnes fractales. Le principe est, à partir d'un triangle initial, de subdiviser chaque segment du triangle en deux puis de modifier la hauteur du point milieu de chaque segment de manière aléatoire (voir figure 10).

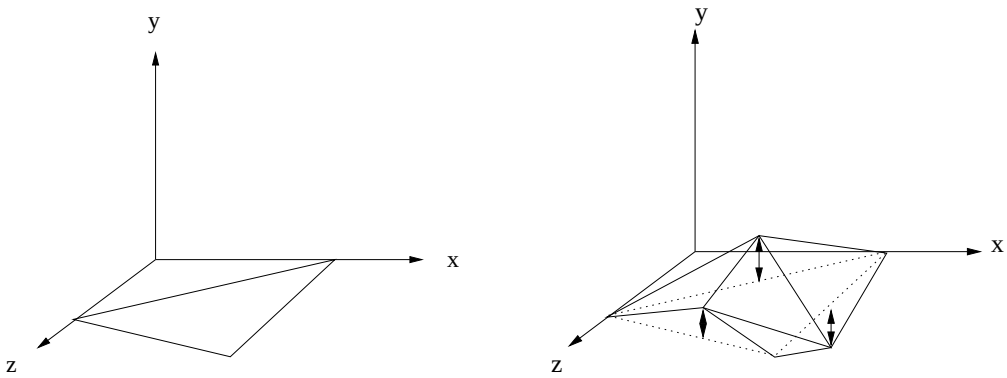


Figure 10: Montagnes fractales : chaque segment est subdivisé en deux et la hauteur du point milieu de chaque segment est modifiée de manière aléatoire.

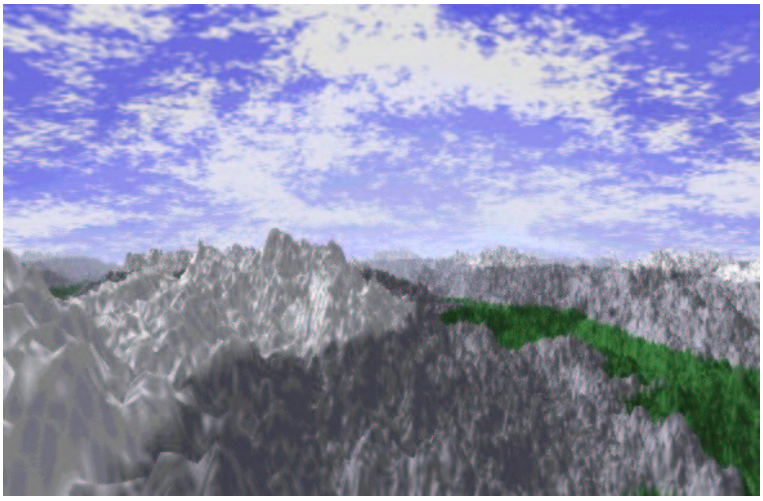


Figure 11: Exemple de modélisation par fractales.

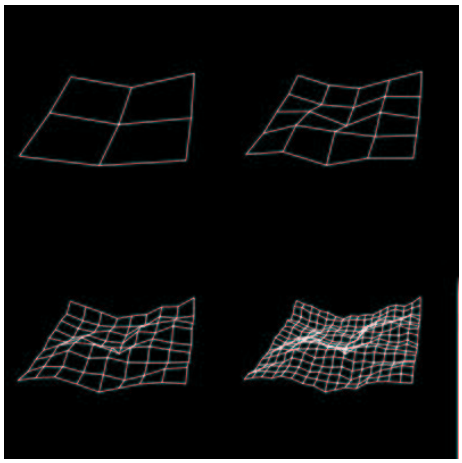


Figure 12: Le motif utilisé pour la modélisation du ciel.

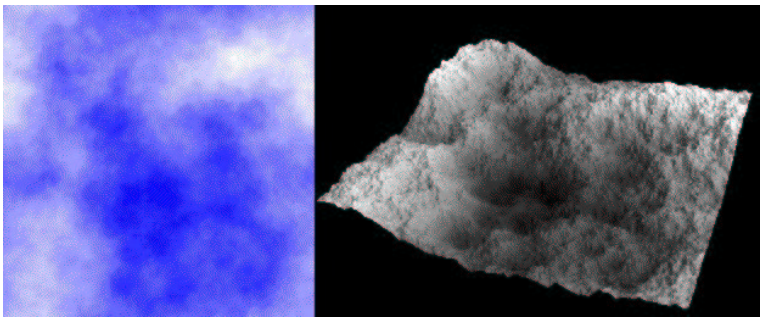


Figure 13: La surface générée pour réaliser le ciel et l’image résultat.