

# TP Synthèse d'images: Courbes et surfaces paramétriques

## CPE

durée - 4h

4ETI, 2013

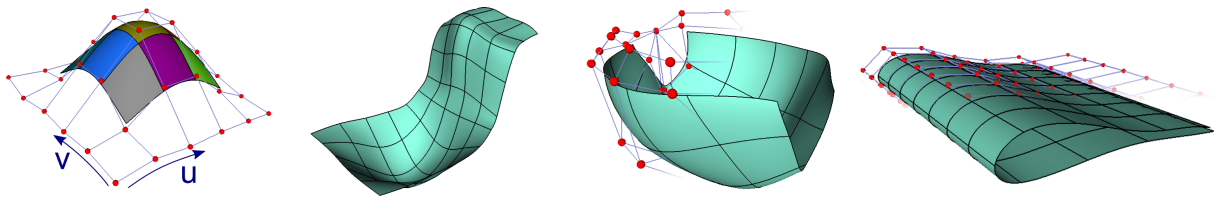


FIGURE 1 – Exemples de résultats de modélisation pour fonctions paramétriques de type Spline.

## 1 But du TP

Les courbes et surfaces paramétriques permettent de modéliser des objets lisses  $C^1$ , ou  $C^2$ . Les surfaces obtenues par raccords de *carreaux* (ou *patches*) sont à la base des logiciels de CAO et de modélisation d'objets manufacturés (voir fig 1).

Le but de ce TP est de se familiariser avec la modélisation de surfaces paramétriques. On s'intéressera en particulier aux carreaux de Bézier ainsi qu'aux carreaux BSplines et à leurs raccordements.

## 2 Prise en main de l'environnement

Utilisez le 1er programme fourni.

**Question 1** Compilez et exécutez le programme en suivant les instructions du fichier README.

Une documentation *Doxygen* du code est disponible dans le repertoire `doc/html/index.html`.

**Note.** Vous n'avez pas besoin de comprendre l'ensemble de ce code pour ce module. Vous pouvez cependant prendre connaissance d'un code d'affichage utilisant la librairie OpenGL pour la partie 3D, et Qt pour la partie fenêtre.

Vous disposez de classes de vecteurs de  $\mathbb{R}^3$  et  $\mathbb{R}^4$  (`vec3` et `vec4`). Ainsi que de matrices  $3 \times 3$  et  $4 \times 4$  (`matrix3`, et `matrix4`) qui possèdent leurs différents opérateurs redéfinis. Les classes de matrices  $4 \times 1$  et  $1 \times 4$  sont également définies et permettent de modéliser des vecteurs lignes/colonnes.

Le fichier `scene` est à compléter. Il est formé de deux parties. La première partie `load_model` permet de charger la scène 3D avant d'afficher celle-ci. La seconde partie `draw_scene` contient les commandes d'affichages des objets visibles à l'écran.

## 3 Évaluateur

On s'intéresse à des surfaces paramétriques, c'est à dire que l'on travail sur des fonctions dépendantes de 2 paramètres  $(u, v)$ . De manière générale, on pourra écrire les coordonnées de la surface comme des

fonctions :

$$\begin{cases} x = f_1(u, v); \\ y = f_2(u, v); \\ z = f_3(u, v); \end{cases}$$

Le code vous propose une structure d'évaluation de fonctions (ici Bezier ou Spline) par le biais de la classe `evaluator_spline`. Cette classe redéfinit l'opérateur `()` en prenant deux arguments : les paramètres  $u$  et  $v$  de la surface, et retourne la valeur correspondante de l'interpolation.

*Note* : On appelle ce type de classe *foncteur* qui a pour rôle de remplacer l'utilisation de pointeurs de fonctions.

On pourra alors écrire ce type de code illustré en fig. 2. Notez que l'interpolation se réalise de manière 1D. Dans le cas d'une surface possédant 3 composantes, on devra donc appeler cette fonction 3 fois.

```
//la matrice 4x4 des donnees
matrix4 matrice_de_donnee(0,1,1,0,
                          1,2,2,1,
                          1,2,2,1,
                          0,1,1,0);

//la classe d'évaluation
evaluator_spline evaluateur(matrice_de_donnee);

//parametres dans [0,1]
double u=0.1;
double v=0.1;

//evaluation de la Bezier aux coordonnées paramétriques (u,v)
double f=evaluateur(u,v);
```

FIGURE 2 – Exemple d'utilisation de la classe d'évaluation.

La classe d'évaluation est actuellement complétée par une interpolation bilinéaire entre les 4 valeurs extrémales de la matrice de données (variable `P` de la classe d'évaluation).

**Question 2** *Observez la classe d'évaluation `evaluator_spline` et visualisez son utilisation dans la méthode `scene::load_model` pour afficher une surface paramétrique.*

## 4 Interpolation d'un carreau de Bézier

On rappelle que l'interpolation bi-dimensionnelle aux coordonnées  $(u, v)$  par des polynomes de degrés 3 peut se mettre sous la forme

$$f(u, v) = \mathbf{u} \mathbf{M} \mathbf{P} \mathbf{M}^T \mathbf{v}^T, \quad (1)$$

avec  $\mathbf{u}$  le vecteur ligne  $(u^3 \ u^2 \ u \ 1)$ ,  $\mathbf{v}$  le vecteur ligne  $(v^3 \ v^2 \ v \ 1)$ ,  $\mathbf{P}$  la matrice  $4 \times 4$  des données (=les coordonnées du polygone de contrôle), et  $\mathbf{M}$  la matrice  $4 \times 4$  de pondération fonction de la base choisie.

**Question 3** *Modifiez la classe d'évaluation (opérateur parenthèse) pour que celle-ci calcule la valeur d'un carreau (ou patch) de Bézier, et non l'interpolation linéaire. Quels propriétés sont vérifiées par cette surface.*

Notez que vous avez à disposition des classes de matrices  $4 \times 4$ , et des vecteurs lignes et colonnes (`matrix1x4` et `matrix4x1`).

Une fois l'opération réalisée, on peut noter que l'illumination du carreau de Bézier n'est pas correcte. En effet, pour permettre à OpenGL de réaliser l'illumination de Phong, il est nécessaire de lui indiquer les normales de la surface aux sommets calculés.

**Question 4** Soit  $f$  la fonction correspondant à la surface 3D. Donnez la relation entre la normale  $n(u, v)$  d'un sommet aux coordonnées locales  $(u, v)$ , et les dérivées partielles  $\frac{\partial f}{\partial u}(u, v)$  et  $\frac{\partial f}{\partial v}(u, v)$  des coordonnées 3D par rapport à  $u$  et  $v$ .

Les deux méthodes `diff_u(u, v)` et `diff_v(u, v)` renvoient les dérivées partielles mentionnées. Notez leur utilisation dans le calcul de la normale dans le fichier `scene`.

**Question 5** Complétez ces deux méthodes, et observez que l'illumination de la surface est corrigée.

Vérifiez que votre programme fourni un résultat visuel comparable à celui de la figure 3.

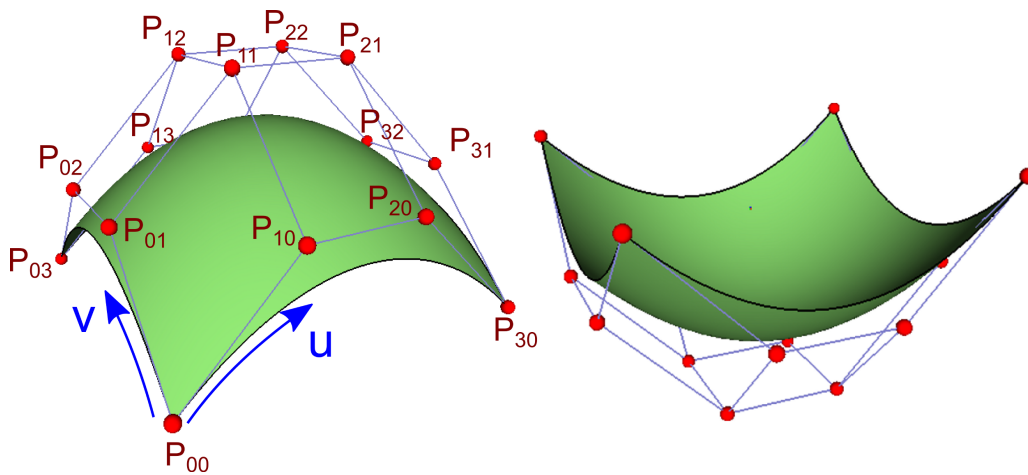


FIGURE 3 – Exemple de résultat de surface de Bézier.

**Question 6** Modifiez les coordonnées de vos points de contrôles et vérifiez le comportement de votre surface de Bézier.

Soit la courbe  $C$  définie par la relation

$$\forall t \in [0, 1], \begin{cases} x = f_1(0.5, t) \\ y = f_2(0.5, t) \\ z = f_3(0.5, t) \end{cases}$$

**Question 7** Calculez et stockez dans un vecteur (de type `std::vector<vec3>`) les coordonnées de la courbe  $C$ . Affichez cette courbe en noire.

Notez que l'appel `opengl_drawer::draw_curve()` permet d'afficher visuellement un vecteur de type `std::vector<vec3>`. Notez également que l'appel à `glColor3f(r, g, b)` est une commande OpenGL permettant d'affecter la couleur  $(r, g, b)$  à tous les affichages qui suivent cet appel.

**Question 8** Construisez un second carreau de coordonnées (3 nouvelles matrices  $4 \times 4$ ). Construisez la surface associée et affichez la (vous définirez pour cela un second objet de type `surf`). Faites en sorte que la surface résultant des deux carreaux soit la plus continue possible. Quantifiez le degré de continuité que vous pouvez obtenir en fonction des contraintes que vous devez appliquer.

## 5 Raccordement lisse et B-Splines

Prenez désormais le second programme qui vous est fourni. Celui-ci, plus complet, permet de gérer des grilles de tailles quelconques ainsi que de déplacer manuellement des sommets de la grille de contrôle.

**Question 9** Compilez et exécutez le second programme.

**Question 10** Complétez la fonction d'évaluation par votre code de calcul de surface de Bézier (dans la classe `evaluator_spline`). Manipulez les sommets de contrôle à la main en les sélectionnant à l'aide du clic gauche+touche CTRL, observez la déformation de la surface de Bézier.

Pour obtenir des surfaces  $C^2$  (deux fois dérivables, dont la dérivée seconde est continue) à partir d'une grille de contrôle quelconque on s'intéresse aux surfaces de type B-Spline (voir fig. 4).

**Question 11** Implémentez l'interpolation par les fonctions de bases de type BSplines Naturelle. Observez le résultat, que constatez vous.

**Question 12** Ajoutez manuellement une ligne dans votre grille de contrôle (boutons de l'interface graphique). Observez le résultat obtenu, que pouvez vous dire ?

**Question 13** Observez la manière dont sont construite les surfaces en calculant les sommets patches à patches (fonction `build_spline_surface` dans le fichier `scene`). Trouvez et comprenez la méthode de sélection des patches les uns derrière les autres.

**Question 14** Dupliquez les sommets du bord de votre polygone de contrôle (bouton de l'interface disponible). Observez le résultat obtenu. Dupliquez désormais 2x ces mêmes sommets, qu'obtenez vous ?

**Question 15** Quelles équations satisfont les courbes tracées en noires sur la surface ? Pourrait-on définir des courbes intermédiaires entre ces carreaux ? quels serait leurs équations ?

La grille de contrôle possède la méthode `build_meshgrid()` qui construit une grille ordonnées suivant  $x$  et  $y$  de manière similaire à l'appel Matlab/Octave.

**Question 16** Testez et modifiez la génération de la grille de contrôle de manière à générer une surface plane de  $10 \times 10$  éléments dont vous viendrez en déformer certains manuellement.

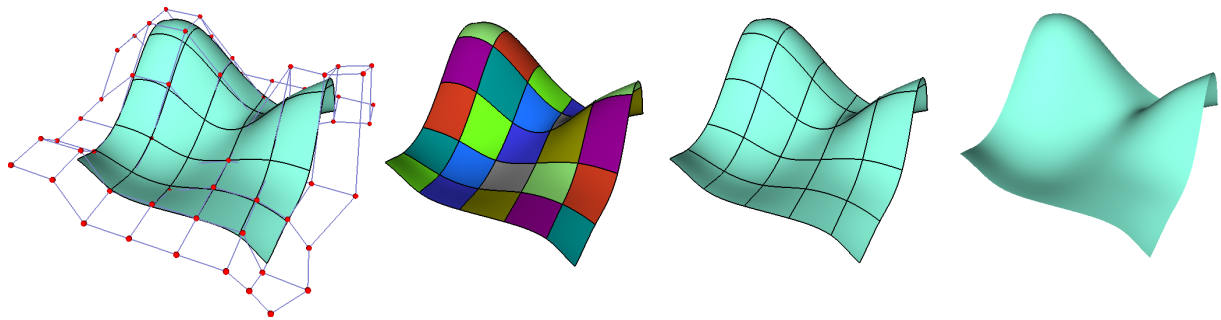


FIGURE 4 – Exemple de résultat de surface BSpline. De gauche à droite : Visualisation de la surface+grille de contrôle ; Visualisation des différents carreaux séparément ; Visualisation de la surface et des courbes de séparation des carreaux ; Visualisation de la surface uniquement.

Il est également possible de donner une structure tridimensionnelle à votre grille de contrôle pour modéliser des formes avancées.

**Question 17** Complétez la méthode `grid::build_sphere()` qui viens répartir uniformément vos points de contrôle sur une sphere (voir fig. 5). Testez le résultat.

**Question 18** Modélisez une forme avancée lisse à l'aide de ce programme. ex. Véhicule, aile d'avion, etc. Notez que vous disposez d'une méthode de sauvegarde et de chargement de grille de contrôle.

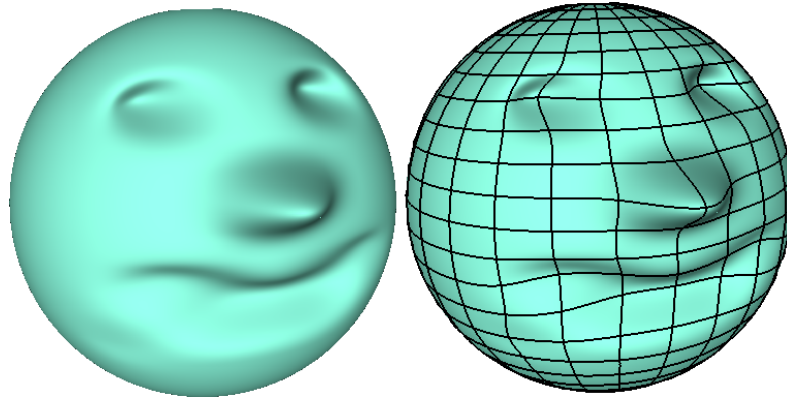


FIGURE 5 – Exemple de modélisation sur une grille de contrôle initialisée sur une sphère.

## 6 Pour aller plus loin

**Question 19** Optimisez l'efficacité du calcul de l'évaluateur. Pouvez vous factoriser certaines calculs ? Pouvez vous paralléliser des calculs<sup>1</sup>.

**Question 20** Proposez une modification sur le parcours de votre grille lors du calcul des carreaux de surface afin de prendre en compte des surface périodique suivant  $u$ , suivant  $v$ , ou suivant  $u$  et  $v$  en même temps. Appliquez cela à des formes de topologie cylindrique ou sphérique.

**Question 21** Implémentez des surfaces animées (dont le polygone de contrôle varie au cours du temps).

**Question 22** Généralisez votre surface aux cas de paramétrage non homogène (vecteur de noeuds), à l'élévation en degré, ou bien généralisez celle-ci aux cas des NURBS.

---

1. Pensez aux threads, et/ou visitez le site d'OpenMP : <http://openmp.org/wp/>