TP Courbes et surfaces paramétriques
CPE

durée: 4h

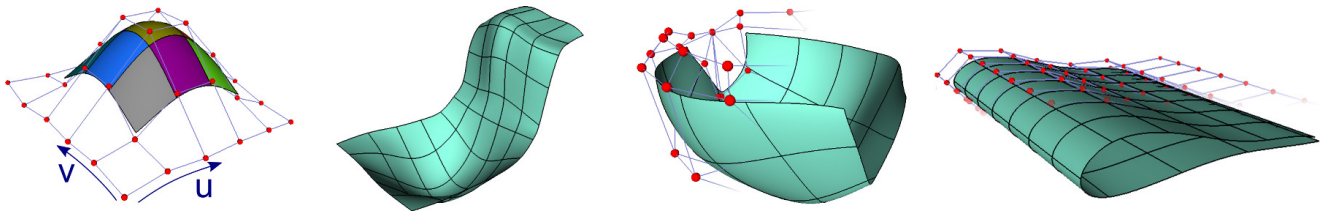

Figure 1: Exemples de modélisation de fonctions paramétriques de type Spline.

Le courbes et surfaces paramétriques permettent de modéliser des objets lisses C^1, ou C^2. Les surfaces obtenues par raccords de *carreaux* (en anglais *patchs*) sont à la bases 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.

1- Fonctionnement actuel du programme

Question 1 *Assurez vous que le programme fourni compile et s'exécute correctement (c.a.d une scène 3D s'affiche à l'écran).*

Question 2 *Observez le fichier* `local/interface/scene.cpp`. *Notez la fonction de chargement de la scène ainsi que la fonction d'affichage. Les autres fonctions servant principalement à la manipulation du polygone de contrôle. Notez qu'il est possible de sélectionner/désélectionner ainsi que de translater les sommets du polygone de contrôle à l'aide de la touche Ctrl + clic gauche/droit.*

Le fonctionnement de l'affichage d'une surface de Bézier est globalement le suivant:

1. La classe `mesh_array_opengl` contient un ensemble de VBO contenant chacun une surface. Ces VBO sont mis à jour lors d'une déformation (liée à la translation d'un sommet du polygone de contrôle).

2. La classe `spline_mesh_surface` contient un ensemble de classes `mesh`. Chaque maillage représentant un carreau Spline. La classe est organisée pour contenir un ensemble de surfaces

organisées sous forme d'une grille 2D. C'est à partir de cette classe que se réalise la mise à jour du `mesh_array_opengl`. Les maillages stockés dans cette classe sont construit en ayant la connaissance d'un polygone de contrôle (classe `polygon_grid`) ainsi qu'un générateur d'un carreau de surface (classe `spline_patch`).

3. `polygon_grid` Contient un ensemble de sommets organisés suivant une grille 2D. Cette structure permet de manipuler directement les sommets du polygone de contrôle. Elle permet en particulier d'ajouter/retirer une ligne ou une colonne entière de sommets, ainsi que d'extraire des sous-grille 4x4 permettant de réaliser le calcul d'un carreau de surface (`polygon_patch`).

4. `polygon_patch` Est une grille de contrôle de 4x4 sommets. Cette structure est utilisée directement par la classe `spline_patch` pour le calcul d'une surface spline sur un carreau.

5. `spline_patch` permet le calcul d'une surface spline bi-cubique étant donné un carreau de contrôle. Elle permet le calcul de la position de la spline mais également de la normale aux coordonnées (u,v). La spline est calculée sur les composantes x, y, et z par le biais d'un évaluateur (`spline_evaluator`).

6. `spline_evaluator` est une classe implémentant la formule d'une spline de manière scalaire. Ainsi ont appellera un évaluateur pour chaque composante (x,y,z). Cette classe stocke le polygone de controle 4x4 comme une matrice.

Question 3 *Dans le fichier* `lib/spline/spline_evaluator.cpp`*, observez la classe d'évaluation (opérateur parenthèse). Quelle type de surface est actuellement implémentée ? Lors de l'exécution du programme, quels sommets faut il translater pour déformer cette surface ?*

2- Carreaux de Bézier

On rappelle que l'on peut définir une surface de Bézier suivant la forme matricielle suivante

$$S(u,v) = U^t \, \mathrm{M}^t \, \mathrm{P} \, \mathrm{M} \, V \ ,$$

avec $U^t = (u^3, u^2, u, 1)$, $V^t = (v^3, v^2, v, 1)$, P étant la matrice des points de contrôles (ici les composantes x, y, ou z), et

$$M = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Question 4 *Écrivez (sur une feuille) la matrice P utilisée respectivement pour les coordonnées x, y, et z au moment du lancement du programme. Soyez sur de comprendre d'où viennent ces nombres et ce qu'ils représentent.*

Question 5 *Implémentez le calcul d'une surface de Bézier à la place de la surface actuelle (notez que vous disposez de* `mat4`*,* `mat4x1` *et* `mat1x4` *pour vous permettre d'implémenter le calcul de manière matricielle).*

Question 6 *Vérifiez visuellement en lançant votre programme et en manipulant vos sommets de contrôle que votre surface est bien un carreau de Bézier (quels critères prenez vous en compte ?).*

Notez que pour l'instant, nous nous contenterons de générer un seul carreau de Bézier (les boutons d'ajout de lignes/colonnes sur le polygone de contrôle ne sont pas applicables avec les surface de Bézier).

Notez également que l'illumination de votre surface n'est pas correcte. Votre surface semble illuminée comme un plan car sa normale est toujours considérée comme constante.

Question 7 *Quelle relation mathématique existe-il entre la normale* $n(u, v)$ *et la fonction* S *ainsi que les paramètres* (u, v).

Question 8 *Remplacez dans le code le calcul des dérivées* `diff_u` *et* `diff_v` *par celui correspondant à un carreau de Bézier.*
Complétez la fonction `normal` *dans le fichier* `lib/spline/spline_patch.cpp` *afin d'obtenir un vrai calcul de normale. N'oubliez pas qu'une normale doit avoir une norme valant* 1.
Observez que cette fois, votre surface est correctement éclairée (voir Figure 2).

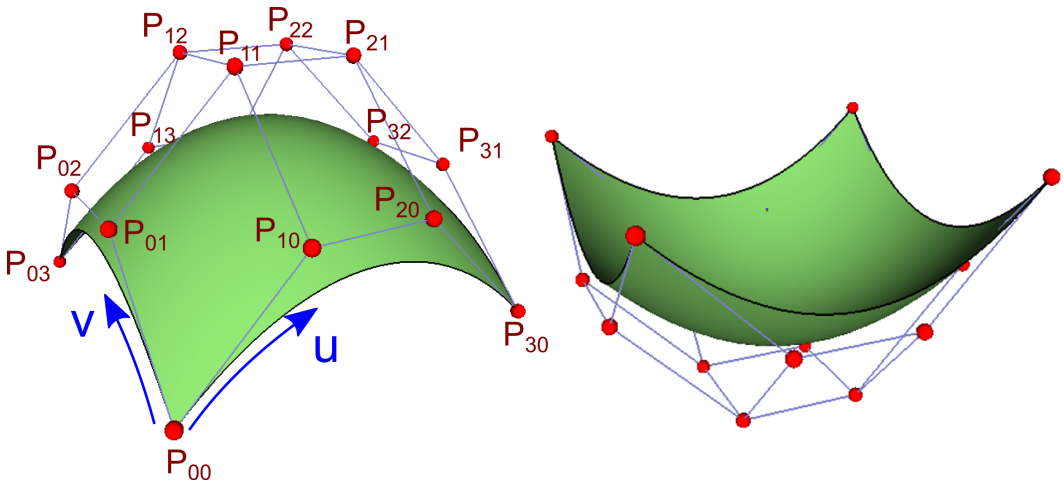

Figure 2: Exemple d'un patch de Bézier.

3- Carreau de Spline

Afin de pouvoir raccorder aisément plusieurs carreaux les uns à coté de autres en donnant l'impression d'une surface globale lisse, nous allons implémenter les fonctions B-Splines à la place des Bézier.

On rappelle que dans le cas d'une B-Spline uniforme, la formule de calcul de $S(u, v)$ reste valable, mais la matrice M est désormais la suivante:

$$M = \frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Question 9 *Mettez à jour votre code (surface et dérivée) pour implémenter une surface B-Spline dans votre évaluateur.*

Question 10 *Observez à l'exécution que votre surface correspond bien à un carreau de B-Spline (quels critères considérez vous ?).*

Question 11 *Utilisez désormais l'ajout de lignes et de colonnes du polygone de contrôle. Observez que vous puissiez définir une surface lisse comportant plusieurs carreaux (voir Figure 3).*

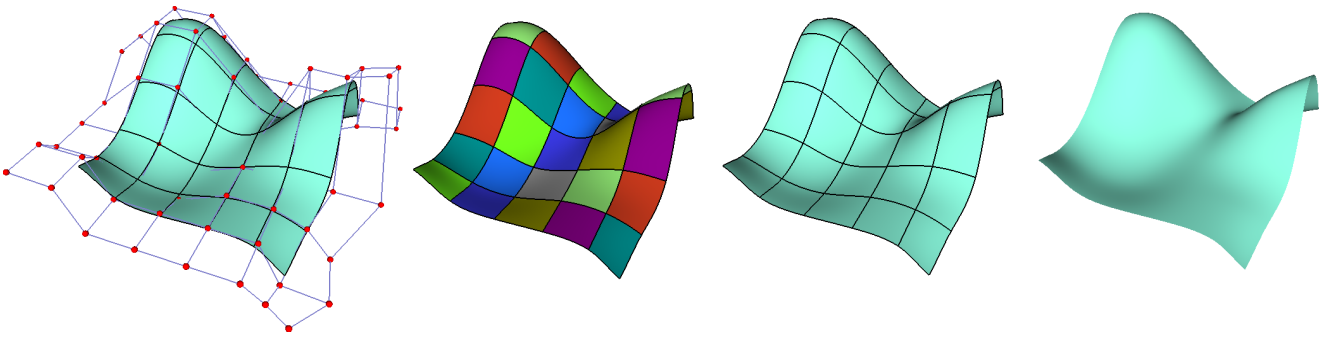

Figure 3: Exemple de résultat de surface B-Spline. De gauche à droite: Visualisation de la surface et du polygone de contrôle; Visualisation des différents carreaux séparément; Visualisation de la surface et des courbes de séparation des carreaux; Visualisation de la surface uniquement.

4- Extension à des formes variés

Question 12 *Que se passe t-il lorsque vous dupliquez (une fois, puis deux fois) les sommets du bords de votre polygone de contrôle ?*

Question 13 *Dans la classe* `polygon_grid`*, implémentez la création de différents type de grille de contrôle (cylindrique, sphérique, etc). Notez qu'il n'est pas forcément trivial de faire en sorte que les extrémités de la surface Spline se raccordent sans artefacts visuels.*

Question 14 *Modélisez une forme avancée lisse à l'aide de ce programme (ex. Figure 4,1, véhicule, aile d'avion, etc.) Notez que vous disposez d'une fonction de sauvegarde et de chargement de la grille de contrôle.*

5- Pour aller plus loin

Question 15 *Implémentez des surfaces animées (dont le polygone de contrôle varie au cours du temps).*

Question 16 *Généralisez votre surface aux cas de paramétrages non homogènes (vecteurs de noeuds), à l'élévation en degré, ou bien encore aux cas des NURBS.*

Question 17 *Implémentez le calcul des carreaux de Bézier sur GPU (quelles données doit-on passer au shader?).*

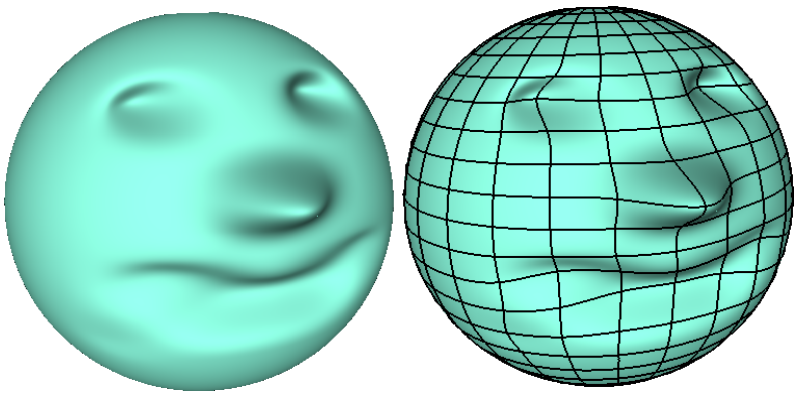

Figure 4: Exemple de modélisation sur une grille de contrôle initialisée à partir d'une sphère.