TP Synthèse d'images: Courbes et surfaces paramétriques
CPE

durée - 4h

4ETI, 2011-2012

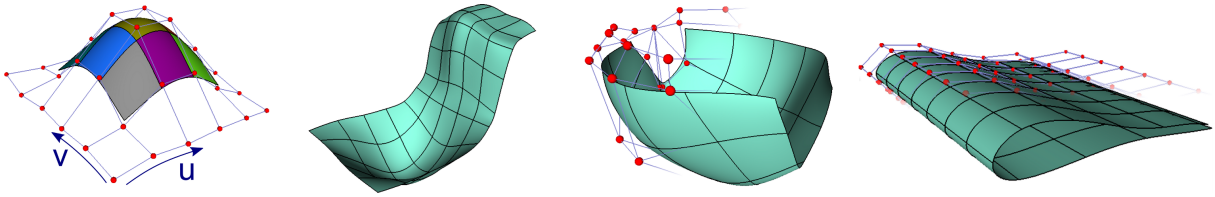

FIGURE 1 – Exemples de résultats de modélisation pour fonctions paramétriques de type Spline.

1 But du TP

Le courbes et surfaces paramétriques permettent de modéliser des objets lisses C^2. Les surfaces obtenus par raccords de *carreaux* (ou *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 courbes et surfaces paramétriques. Dans une première partie, le cas des courbes de l'espace sera étudié dans avec l'interpolation linéaire puis des courbes de Bézier. Dans une seconde partie, nous étendons ce type d'interpolation au cas des surfaces. On s'interessera en particuliers aux carreaux de Bézier ainsi qu'aux carreaux BSplines et à leurs raccordements.

2 Courbes 1D

2.1 Interpolation linéaire

Soit l'interpolation linéaire entre deux positions de l'espace A et B. On rappel que l'on a dans ce cas l'interpolation

$$f(t) = (1-t)\,A + t\,B\,,$$

avec $t \in [0,1]$.

Question 1 *Ecrivez en language Matlab ou Octave une fonction qui prend comme paramètre* $y0 \in \mathbb{R}$, $y1 \in \mathbb{R}$, $t \in \mathbb{R}$ *et renvoie l'interpolation linéaire au paramètre* t *entre* y_0 *et* y_1.

Question 2 *Utilisez cette fonction pour afficher un echantillonnage du segment interpolant deux positions de l'espace. Notez que l'on réalisera 3 appels successifs sur les coordonnées* (x, y, z).

Question 3 *Réalisez cette fois l'echantillonnage d'une courbe constituée de 8 points de l'espace à l'aide de cette fonction. Lorsque le nombre déchantillon augmente, converge t-on vers une courbe lisse ?*

2.2 Courbes de Bezier

Considérons les 4 premiers points de controle de la question précédente que l'on désignera par A, B, C, D.

Interessons nous à leur coordonnées x que l'on désignera par (x_A, x_B, x_C, x_D).

On rappelle que la courbe de Bézier de degré 3 dont le polygone de controle est (x_A, x_B, x_C, x_D) peut s'exprimer par

$$x(t) = (1-t)^3\, x_A + 3(1-t)^2 t\, x_B + 3(1-t) t^2\, x_C + t^3\, x_D$$

$$x(t) = \mathbf{t}\ \mathrm{M}\ \mathbf{p^T}\ ,$$

avec $\mathbf{t}$ le vecteur ligne des paramètres $(t^3\ t^2\ t\ 1)$, et $\mathbf{p}$ le vecteur ligne des données $(x_A\ x_B\ x_C\ x_D)$, et M une matrice 4×4.

Question 4 *Donnez l'expression de la matrice* M.

Question 5 *Étant donnée 4 valeurs x_A, x_B, x_C et x_D, ainsi qu'un paramètre d'interpolation $t \in [0, 1]$, implémentez en language Matlab ou Octave une fonction qui renvoit la valeur $x(t)$ ainsi définie.*

Question 6 *Appliquez cette fonction sur les coordonnées x, y et z de votre polygone A, B, C, D. Affichez le résultat en affichant un point par échantillon ainsi qu'en reliant vos échantillons par des segments.*

Question 7 *Tracez les segments $[AB]$ et $[BC]$. Vérifiez les propriétés des courbes de Bézier.*

Question 8 *Appliquez cette fonction pour générer une courbe à partir de votre polygone à 8 points. Lorsque le nombre déchantillon augmente, converge t-on vers une courbe lisse ? Détaillez.*

3 Surfaces 2D

On s'interesse désormais à des surfaces, 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 :

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

3.1 Prise en main de l'environnement

Utilisez désormais le 1er programme fourni.

Question 9 *Compilez et executez 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.

L'organisation des dossiers est illustrée en fig 2.

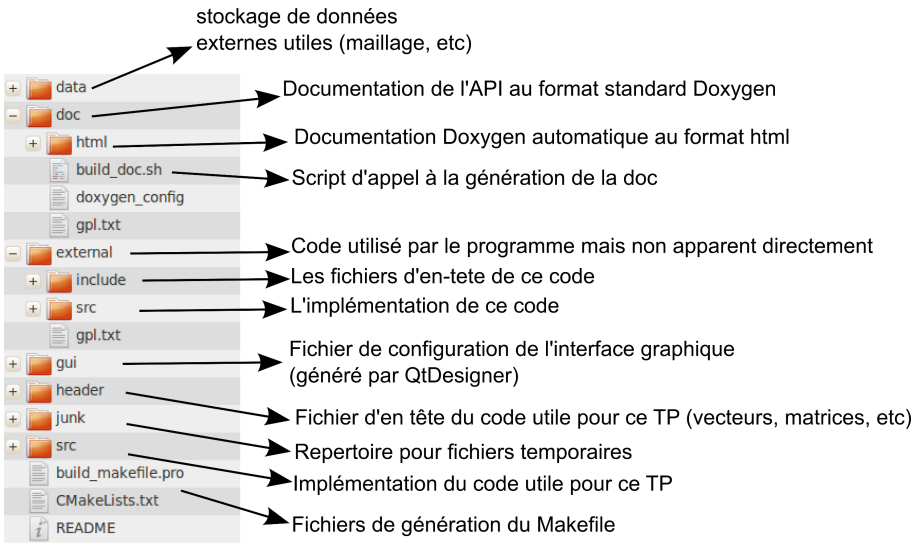

FIGURE 2 – Organisation des repertoires du code proposé.

Vous disposez de classes de vecteurs de $\mathbb{R}^3$ et $\mathbb{R}^4$ (v3 et v4). Ainsi que de matrices 3×3 et 4×4 (matrix3, et matrix4) qui possèdent leurs différents opérateurs redéfinis.

La méthode *scene : :load model* permet de charger la scène 3D avant d'afficher celle-ci.

Question 10 *Afin de prendre en main les classes de bases qui vous sont fournies, implémentez et commentez ce que réalise les lignes de la figure 3.*

```
void scene::load_model()
{
    v3 p1(1,2,3);
    v3 p2(-1,2.2,5.5);

    std::cout<<p1+p2<<std::endl;

    matrix3 M( 1.0 , 0.5 , 0.7,
               2.1 ,-3.2 , 0.4,
              -5.2 , 1.4 ,-2.5);

    v3 y=M*v3(1,5,6);
    std::cout<<y.x()<<","<<y[1]<<","<<y.dot(0,0,1)<<std::endl;
    std::cout<<y.dot(p2-p1)<<std::endl;


    exit(0);
```

FIGURE 3 – Exemple d'utilisation des classes de bases.

3.2 Évaluateur

Le code vous propose une structure d'évaluation de Bezier ou de Spline par le biais de la classe *cpe : :evaluator spline*. Cette classe redéfinie 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. 4.

```
//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 4 – 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.

Question 11 *Observez la classe d'évaluation et visualisez son utilisation dans la méthode* scene : :load model *pour afficher un carreau de Bézier.*

3.3 Interpolation d'un carreau de Bézier

Question 12 *Modifiez la classe d'évaluation pour que celle-ci calcule la valeur d'un carreau (ou patch) de Bézier, et non l'interpolation linéaire.*

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

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

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

Question 13 *Stockez dans un vecteur les coordonnées de la courbe définie par la relation :*

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

Question 14 *Complétez les fonctions de calculs de dérivées aux coordonnées (u, v) afin de pouvoir calculer des normales signifiantes. Comprenez et expliquez le principe du calcul de la normale à la surface.*

Vérifiez que votre programme fournie un résultat visuel comparable à celui de la figure 5.

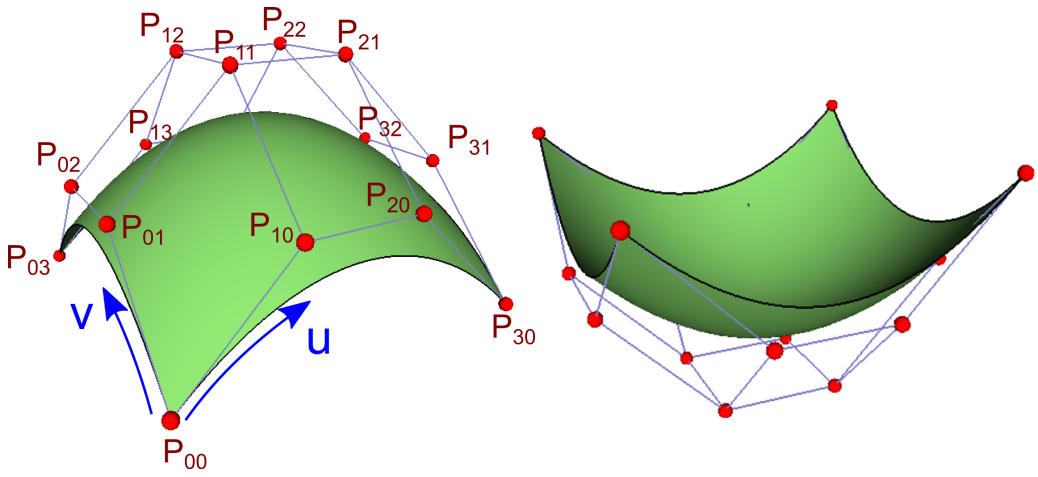

FIGURE 5 – Exemple de résultat de surface de Bézier.

Question 15 *Modifiez les coordonnées de vos points de contrôles et vérifiez le comportement de votre surface de Bézier.*

Question 16 *Construisez un second carreau de coordonnées (3 nouvelles matrices 4×4). 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.

3.4 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 controle.

Question 17 *Compilez et executez le second programme. Complétez la fonction d'évaluation par votre code de calcul de surface de Bézier. Modifiez les sommets de contrôle à la main en les selectionnant à l'aide du* clic gauche+touche CTRL.

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. 6).

Dans le cas où les paramètres (u, v) sont choisis pour évoluer de manière homogène sur la surface, les surfaces B-Splines peuvent s'exprimer de manière similaire à l'eq. 1. Cette fois, on considèrera par contre la matrice de poids M telle que

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

Question 18 *Implémentez l'interpolation par les fonctions de bases de type BSplines. Observez le résultat, que constatez vous.*

Question 19 *Ajoutez manuellement une ligne dans votre grille de contrôle. Observez le résultat obtenu, que pouvez vous dire ?*

Question 20 *Observez la manière dont sont construite les surfaces en calculant les sommets patchs à patchs. Trouvez et comprenez la méthode de sélection des patchs les uns derrière les autres.*

Question 21 *Dupliquez manuellement les sommets du bord de votre polygone de contrôle. Observez le résultat obtenu. Dupliquez désormais 2x ces même sommets, qu'obtenez vous ?*

Question 22 *Testez les boutons prévus par l'interface graphique pour ajouter ou retrancher des lignes de la grille de contrôle. Modifiez manuellement certains sommets et visualisez les propriétés de contrôle locale des B-Splines. (Pensez qu'un bouton vous permet de colorer séparément chaque carreau différement).*

Question 23 *Quelles équations satisfont les courbes tracées en noire sur la surface ? Pourrait-on définir des courbes intermédiaires entre ces carreaux ? quels serait leurs équations ?*

La grille de contrôle possède une méthode *build meshgrid()* qui construit une grille ordonnées suivant x et y de manière similaire à l'appel Matlab/Octave.

Question 24 *Testez cet appel pour générez une surface plane de* 10×10 *éléments dont vous viendrez en déformer certains manuellement.*

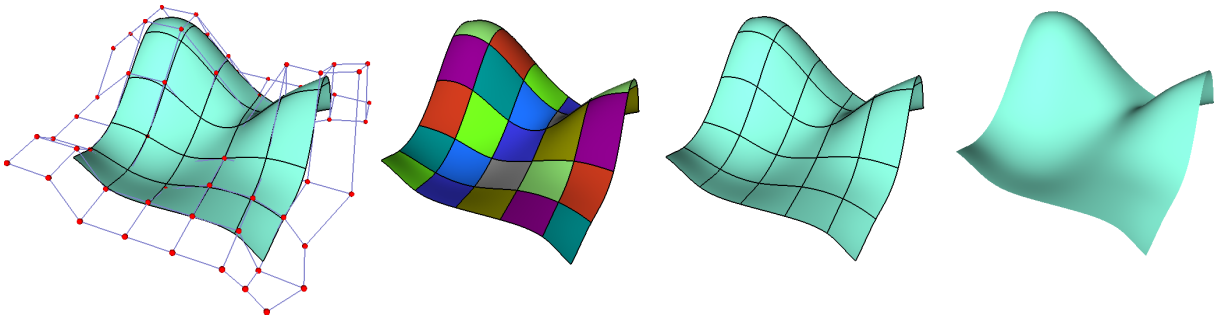

FIGURE 6 – Exemple de résultat de surface BSpline. De gauche à droite : Visualisation de la surface+grille de contrôle ; Visualisation des différents carreaux séparément ; Visualisation de la surface et des courbes de séparation des carreaux ; Visualisation de la surface uniquement.

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

Question 25 *Complétez la méthode* build sphere *qui viens répartir uniformément vos points de contrôle sur une sphere (voir fig. 7). Testez le résultat.*

Question 26 *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.*

4 Pour aller plus loin

Question 27 *Optimisez l'efficacité du calcul de l'évaluateur. Pouvez vous factoriser certaines calculs ? Pouvez vous paralléliser des calculs* [1].

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

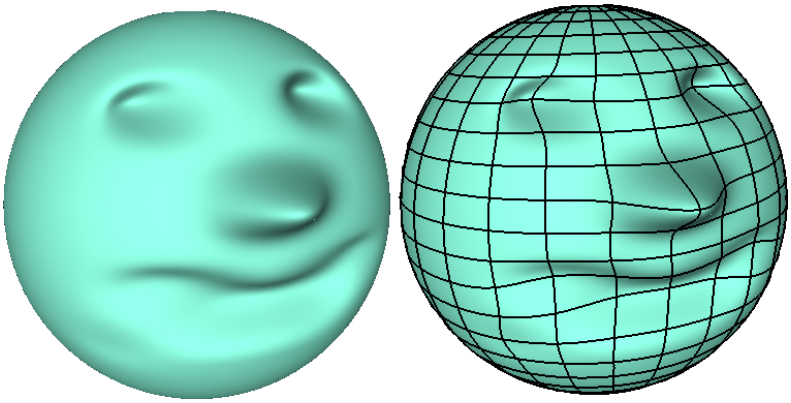

FIGURE 7 – Exemple de modélisation sur une grille de contrôle initialisée sur une sphère.

Question 28 *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 29 *Implémentez des surfaces animées (dont le polygone de contrôle varie au cours du temps).*

Question 30 *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.*