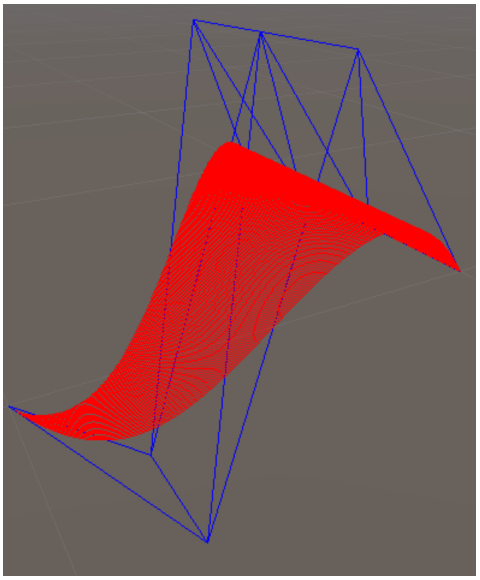
TP2 - Interpolation Approximation

Approximation Polynomiale

Julien Desvergnes, Géraldine Morin



Ce TP illustre la partie de cours sur les courbes et surfaces de Bézier et l'approximation polynomiale.

Table des matières

1 Rappel : Installation et présentation rapide de Unity

1.1 Installation

Pour installer Unity3D, rendez-vous sur le site `https://unity3d.com/fr/get-unity/download` et téléchargez le UnityHub. Une fois ceci fait, il faut télécharger un éditeur Unity (en gros une version). La version à choisir est 2019.3.5f1 si elle est disponible. Si ce n'est pas le cas prenez simplement la plus récente.

Pour vérifier que tout s'est bien passé, essayez de créer un nouveau projet en faisant :

$$Project >> New >> 3D$$

1.2 Prise en main de l'interface

Je vous encourage à essayer de jouer un peu avec Unity, c'est à dire créer des objets, les déplacer, utiliser la caméra, faire des petites bidouilles. Cela vous permettra de ne pas être perdus lors des premiers TP.

Vous trouverez une petite vidéo qui résume le fonctionnement de l'interface ci-dessous.

`http://www.unity3d-dev.com/tuto/debutant/les-bases-unity-3d/interface.php`

2 Préparation pour le TP2

Il faut récupérer les sources sous Moodle : `TP2.zip`

Une fois cela fait, dans UnityHub, aller dans la section Projects et faire ADD puis choisir TP2 comme racine.

Suite à cela le projet devrait être importé dans Unity, vous pourrez alors le lancer.

3 Descriptif du projet

3.1 Scène

Dans la scène, il y a trois sections :

UI : La partie UI (pour user interface) regroupe des éléments liés à de l'affichage textuel. **Vous n'aurez pas à modifier cette partie**

TP2 : C'est ici qu'il y a les éléments spécifiques au TP2. Des détails seront présentés dans la suite.

Setup : Des données de mise en place, caméra, lumière, etc. **Vous n'aurez pas à modifier cette partie**

3.1.1 Exo-Bernstein

C'est cet objet qui sera à modifier dans le cadre de l'exercice des polynomes de Bernstein

3.1.2 Exo-DC

C'est cet objet qui sera a modifier dans le cadre de les exercices traitants de l'algorithme de De Casteljau.

3.2 Assets » Scripts

C'est dans cette section que vous trouverez les fichiers à modifier.
Les fichiers à modifier sont
— `Bernstein.cs`,
— `DeCasteljauEvaluation.cs`,
— `DeCasteljauSubdivision.cs`,
— `CalculHodographe.cs`,
— `SurfaceBezier.cs`

4 Travail à faire

4.1 La base de Bernstein

Tracer les n fonctions de la base de Bernstein d'un degré n choisi. Que se passe-t-il quand on augmente le degré n ? (attention, ce sont des courbes fonctionnelles et non paramétriques).

En pratique : Compléter le code de la fonction `buildPolysBernstein` et de la fonction `buildEchantilonnage` (fichier `Bernstein.cs`)

Tester ses modifications : Dans l'éditeur, s'assurer que le bouton *Gizmos* est bien coché (dans la barre de scène), sélectionner l'objet Exo-Bernstein dans la hiérarchie. Une fois ces opérations effectuées, cliquer sur play. Vous pouvez utiliser le slider pour modifier le nombre de polynômes (=degré +1) de Bernstein.

4.2 Algorithme de De Casteljau - évaluation

Étant donnés $n+1$ points $P_i = ((x_i, y_i))_{i=0}^n$ du plan, dessinez la courbe de Bézier *P(t)* correspondante et son polygone de contrôle en utilisant l'algorithme de De Casteljau pour l'évaluation. Que se passe-t-il quand le degré de la courbe augmente ? (faire le lien avec la question précédente).

En pratique : Compléter le code de la fonction `buildEchantilonnage` et `DeCasteljau` (fichier `DeCasteljauEvaluation.cs`) Vérifiez que l'échantillonnage reste suffisamment précis pour une fonction de degré élevé (sinon, vous pouvez changer la signature de la fonction `buildEchantilonnage` pour qu'elle prenne le degré en paramètre)

Tester ses modifications : Dans l'éditeur, s'assurer que le bouton *Gizmos* est bien coché (dans la barre de scène), sélectionner l'objet Exo-DC**E** dans la hiérarchie. Pour placer des points, il suffit de cliquer avec la souris, pour lancer l'approximation, appuyez sur entrée.

4.3 Algorithme de de Casteljau - subdivision

Étant donnés $n+1$ points $P_i = ((x_i, y_i))_{i=0}^n$ du plan, dessinez le polygone de contrôle décrivant la courbe approximante après n subdivisions.

En pratique Il suffit de coder ici la méthode `DeCasteljauSub` dans le fichier `DeCasteljauSubdivisio`

Tester ses modifications : Dans l'éditeur, s'assurer que le bouton Gizmos est bien coché (dans la barre de scène), sélectionner l'objet Exo-DC**S** dans la hiérarchie. Pour placer des points, il suffit de cliquer avec la souris, pour lancer l'approximation, appuyez sur entrée.

4.4 Bonus : Hodographe

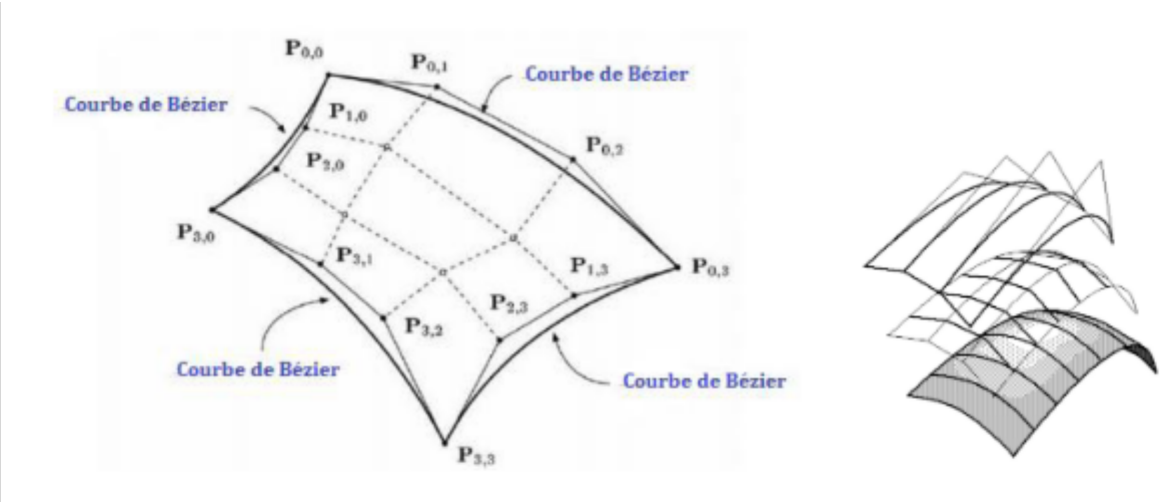
On appelle hodographe d'une courbe de Bézier de degré n, la courbe de Bézier de degré *n-1* représentant la dérivée de la courbe. Dessinez la courbe hodographe d'une courbe de Bézier, c'est à dire dont les vecteurs de base sont les vecteurs $P_{i+1} - P_i$. Peut-on lier la régularité de la paramétrisation d'une courbe avec son hodographe ?

En pratique : Coder les deux méthodes `DeCasteljauSub` et Hodographe du fichier `CalculHodographe.cs`

Tester ses modifications : Dans l'éditeur, s'assurer que le bouton Gizmos est bien coché (dans la barre de scène), sélectionner l'objet Exo-DC**S** dans la hiérarchie. Pour placer des points, il suffit de cliquer avec la souris, pour lancer l'approximation, appuyez sur entrée.

4.5 Surfaces de Bézier : produit tensoriel ou Bézier triangulaire

Objectif : Programmer un des deux modèles de surfaces de Bézier.



Cas du produit tensoriel On considère une surface paramétrique S :

$$S : \mathcal{R}^2 \to \mathcal{R}^3, (u, v) \to S(u, v)$$

$$S(u,v) = \sum_{j=0}^{n}\sum_{i=0}^{m} P_{ij} B_i^m(u) B_j^n(v) = \sum_{j=0}^{m} B_j^n(v) P_j(u)$$

On traite des courbes de de contrôle $P_j(u)$ plutôt que de points de contrôle P_j

Attention à faire si vous choisissez les surfaces en PT !

- Créer un nouvel objet vide dans la section **Exo-Surface** intitulé SurfaceBezierPT. (Clic droit -> Create Empty. S'assurer qu'il est en (0,0,0)).
- Ajouter un composant sur l'objet depuis l'inspecteur, appelé SurfaceBezierPT. (Add Component)
- Copier Coller le code de la classe Interpolateur de surface du TP1. (Uniquement l'intérieur de la classe)
- Ajouter un **using System ;** en début de fichier.
- Associer au champs particle l'objet Point qui se trouve dans Assets»Prefabs.
- Cocher la case Auto Generated Grid

Normalement, la grille de point à approximer devrait s'afficher si vous lancez le jeu.

Cas Bézier triangulaire En 1D : t varie sur un segment [0,1] : t = 0(1-t) + 1t

En 2D : étant donné que u + v + w = 1, (u,v) varie dans un triangle défini par les points (0,0), (0,1) et (1,0) et on définit les coordonnées barycentriques par :

$$(u, v) = (1 - u - v)(0, 0) + u(1, 0) + v(0, 1)$$

$$S(u,v) = \sum_{i+j+k=n, i,j,k>=0} P_{ij} B_{ijk}^n(u, v)$$

avec $B^n_{ijk}(u,v) = \binom{ijk}{n}(1-u-v)^k u^j v^i$

et $\binom{ijk}{n} = \frac{n!}{i!j!k!} = \frac{n!}{i!j!(n-i-j}!$ et $k = n-i-j$