SUJET DE PROJET EN MIGS2, DIJON, 2014--2015

RESOLUTION DE CONTRAINTES SUR DES COURBES DE SUBDIVISION

Thématique : Modélisation géométrique, Infographie

Encadrant : Dominique MICHELUCCI (dmichel@u-bourgogne.fr)

Pour 1 ou 2 étudiants. Langage de programmation : libre.

Contexte scientifique :

La modélisation géométrique a atteint une maturité au niveau des outils mathématiques permettant de créer les modèles à partir de surfaces dites à pôles (Bézier, B-splines, NURBS). Les manques sont essentiellement sur des méthodes de plus haut niveau permettant de concevoir les objets en termes de propriétés et/ou de contraintes correspondant aux métiers des utilisateurs.

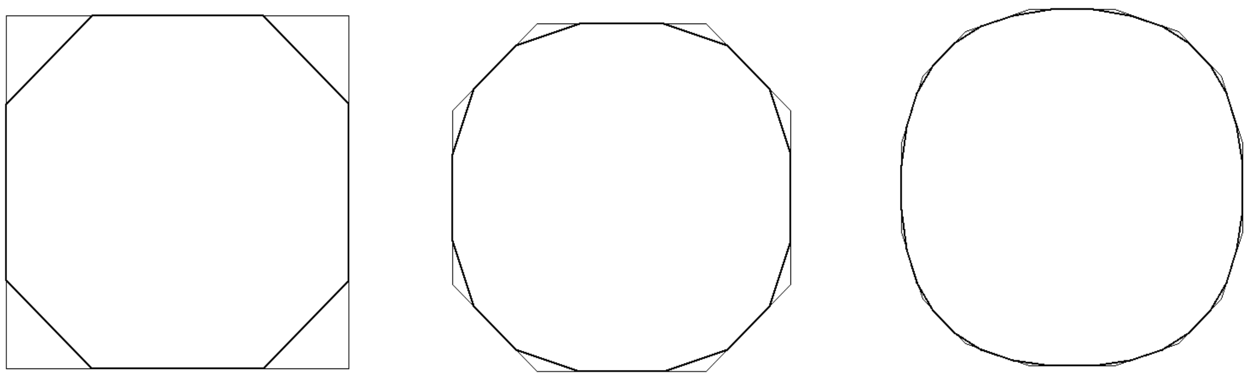
Par ailleurs, le modèle mathématique s'appuyant sur les surfaces de subdivision a totalement envahi le monde de l'animation et du cinéma mais pas celui de la Conception Assistée par Ordinateur (CAO). Or les complémentarités avec les surfaces à pôles sont fortes. Concevoir des modèles de subdivision à partir de contraintes est aussi un défi nouveau.

Enfin, envisager de mêler dans cette même approche les surfaces à pôles et les surfaces de subdivision est totalement nouveau et permettrait d'avoir des modèles d'objet tirant parti des deux modèles mathématiques sous-jacents.

Description du sujet :

L'étudiant programmera la résolution de contraintes géométriques sur des courbes 2D subdivisées par la méthode de Chaikin (voir Figure page suivante).

La subdivision de Chaikin coupe au quart et au trois quart chaque segment d'un polygone P= (P_i, i=0...n-1) ; le polygone résultant est Q=(Q_i, i=0..2n-1), où $Q_{2k}=(3P_k+P_{k+1})/4$ et $Q_{2k+1}=(P_k+3P_{k+1})/4$. Chaque étape de subdivision double donc le nombre de sommets et d'arêtes du polygone. Le polygone initial P génère une suite de polygones Q, R, S, T, U... qui après 5 ou 6 étapes est visuellement indiscernable d'une courbe lisse. Certains sommets du polygone initial peuvent être doublés $P_i=P_{i+1}$), et persisteront dans les subdivisions ultérieures. L'équation paramétrique de la courbe limite peut être calculée dans certains cas, mais le projet consiste à ne pas utiliser cette équation. En effet, cette équation est plus difficile à calculer en 3D, et nous souhaitons généraliser les méthodes mises au point lors de ce projet à des problèmes 3D.



Ci dessus, les 3 premières étapes de la subdivision de Chaikin à partir d'un polygone carré initial. Chaque image montre un polygone et sa subdivision.

Voici les contraintes géométriques possibles : la courbe limite, ou le polygone après un nombre fixé de subdivisions, doit passer par un point donné, ou bien être tangent à une droite donnée (soit en un point d'indice connu, soit en un point d'indice inconnu), ou bien être tangent à un autre polygone de Chaikin, ou à une courbe donnée par une équation paramétrique (exemple : $x=\cos t$, $y= \sin t$) ou par une équation implicite (exemple : $x^2+y^2-1=0$). Résoudre les contraintes signifie trouver les coordonnées des sommets étiquetés inconnus du polygone initial, ou des polygones initiaux.

Un polygone initial sera disponible dans tous les cas. Il doit être visuellement proche de la solution. Le polygone initial et les contraintes seront donnés, au choix, par un bout de programme, ou un fichier texte de données, ou une petite interface graphique ad hoc. Attention : l'interface graphique n'est pas le sujet du stage.

Bien sûr, les contraintes aussi devront être connues. Elles seront spécifiées comme le polygone initial, par un bout de programme, ou... etc.

Pour résoudre les contraintes, le stagiaire n'utilisera pas une méthode qui exige de poser explicitement un système d'équations.

Il suffit en effet de pouvoir mesurer à quel point un polygone donné, obtenu après quelques étapes de subdivision, satisfait les contraintes spécifiées. Par exemple, il est facile de mesurer la distance entre le polygone et un point par lequel il est censé passer, ou la distance à une droite donnée, à laquelle il doit être tangent. C'est juste un peu plus compliqué pour des tangences entre courbes.

Le stagiaire testera diverses méthodes de résolution. Nous suggérons de minimiser la somme des carrés des équations. Citons quelques méthodes possibles :

- des méthodes de descente de gradient (plus forte pente, ou BFGS), disponibles par exemple dans la librairie GSL (GNU Scientific Library). Ces méthodes ont besoin de pouvoir calculer la fonction à minimiser, et son vecteur gradient.

- la méthode du simplexe de Nelder-Mead, disponible dans la librairie GSL. Elle n'a pas besoin du gradient. Elle nécessite seulement une fonction à minimiser, ou un pointeur sur la fonction à minimiser, F(X,U). Les X et U sont des tableaux de valeurs ; l'optimiseur a le droit de modifier les valeurs des inconnues X ; il n'a pas le droit de modifier les valeurs des paramètres, U.

- la méthode de Torczon, une amélioration de la précédente. Elle n'a pas besoin du gradient elle non plus.

- La méthode par homotopie. Présentons la dans le cas où les contraintes sont des points par lesquels la courbe doit passer. Les points de passage Pi sont projetés sur la courbe initiale, en Qi. Puis on définit Mi(t) = (1-t)Qi + t Pi pour t variant entre 0 et 1. La solution pour t=0 est connue : c'est le polygone initial. La variable t est incrémenté de 0.05, et on fait passer la courbe par les Mi(t + 0.05) en prenant comme solution initiale la solution précédente. Quand t vaut 1, les Mi(1) sont les Pi. Pour chaque étape, une méthode de plus forte pente devrait marcher. Intuitivement, on prend les points Qi sur la courbe les plus près des points de passage Pi, puis on les tire doucement pour qu'ilsviennent sur les Pi, et au fur et à mesure, le polygone est mis à jour.

- l'optimisation par essaim (« swarm optimization ») dite aussi optimisation particulaire ou PSO (« particle system optimization »). L'optimisation par essaim fournira de bons polygones initiaux, à partir desquels la méthode BFGS, qui reconstruit approximativement le hessien à partir des valeurs de la fonction, aura de bonnes chances de converger.

Le gradient sera évalué numériquement, pour les méthodes qui en ont besoin.

Ces méthodes de résolution ou d'optimisation sont utilisables même quand la fonction optimisée est une boîte noire. Ce cas se produit de plus en plus souvent dans l'industrie (par exemple, la boîte noire est un programme de simulation). Elles sont bien documentées sur internet, et sont souvent disponibles dans des librairies.

Remarque : les inconnues sont les coordonnées du polygone de contrôle. Il peut être commode de distinguer inconnues (le solveur a le droit de modifier les valeurs des inconnues) et paramètres (le solveur n'a pas le droit de modifier les valeurs des paramètres). Eventuellement, certains sommets du polygone peuvent être fixés, autrement dit considérés comme des paramètres, et le solveur n'aura pas le droit de les modifier. On peut aussi chercher la solution qui minimise la somme des déplacements des sommets du polygone de contrôle.

Compétences attendues :

Ce stage est un projet de recherche. Il est nécessaire de savoir programmer en C, ou C++, ou autre. L'étudiant doit être à l'aise avec les équations paramétriques ou implicites des courbes 2D les plus courantes, et être capable de les approcher par des polygones. Il doit pouvoir programmer le calcul du point d'un polygone le plus proche d'un point donné ou d'une droite donnée, ou la distance entre deux polygones. Il doit savoir programmer ou réutiliser des méthodes bien documentées comme BFGS ou PSO. Il doit faire preuve d'autonomie dans sa recherche bibliographique et dans le choix des algorithmes qu'il utilisera et programmera.