Étude et Implémentation GPU des carreaux de Gregory à N-cotés

Travail d'Étude et de Recherche
—
Rapport Intermédiaire

Maxime Hohl

Encadrants
—
Dominique Bechmann
Sylvain Thery

29 février 2020

1 Objectif et Contexte

1.1 Objectif

1.1.1 Carreaux de Gregory à N-cotés

Les carreaux de Gregory est une méthode permettant le représentation de surfaces via des points de contrôles (cf. Figure 1). Cela permet de définir un objet uniquement via des points de contrôles et de ne discrétiser la surface qu'au dernier moment, ce qui permet potentiellement d'augmenter la précision des objets sans augmenter la place en mémoire nécessaire.

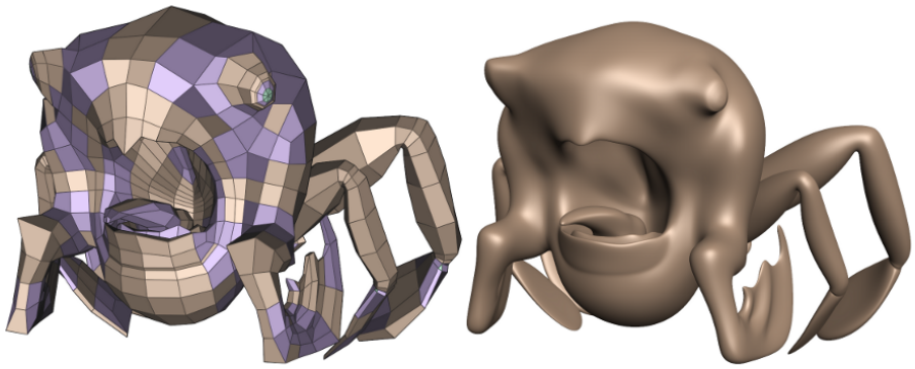

Figure 1 – À gauche un maillage de contrôle et à droite la surface générée grâce à ce maillage. (Images tirées de la source [3])

1.1.2 Implantation GPU

L'implantation de l'affichage des carreaux de Gregory à N-cotés doit se faire sur GPU grâce à OpenGL et la tesselation. La tesselation est apparue en version 4.0 d'OpenGL[1] et est découpée en deux sous-*shaders* : le *tesselation control shader* et le *tesselation evaluation shader*.

Le programme sera développé est C++14, et on utilisera la version la plus basse possible d'OpenGL (≥ 4.0) pour essayer de garder le maximum de compatibilité avec les différents *hardware* et systèmes (par exemple Mac OS X ne supporte qu'OpenGL ≤ 4.1[2]).

1.2 Contexte

La représentation 3D d'objets se fait usuellement directement grâce à des géométrie discrètes de type maillages triangulaires, car ils sont plus simple à représenter et tout les GPU modernes sont optimisés pour leur utilisation. Mais plus le modèle est précis, plus la mémoire nécessaire est importante et plus le temps de calcul pour leur manipulation (animation, etc.) augmente. Et une géométrie discrète ne permet pas toujours d'avoir une surface visuellement continue.

L'utilisation d'une représentation avec points de contrôle tel que les surfaces et Bézier et les carreaux de Gregory permettent de drastiquement réduire la taille nécessaire pour les modèles, on passe de milliers, voir centaines de milliers de *vertices* à quelques centaines de points de contrôle. Le problème de continuité peut aussi être résolue, étant donné que l'on peut augmenter à la volée la précision de la surface.

Les carreaux de Gregory permettent de fortement réduire le nombre de points de contrôle par rapport aux surfaces de Bézier[3].

2 Tâches effectués

Jusque là, les courbes de dimension N et les surfaces rectangulaires de Bézier de dimension $N \times M$ sur GPU ont était implantées. C'est un premier pas qui servira de base de code pour l'implantation des carreaux de Gregory et qui a permis d'apprendre le fonctionnement des *tesselation shaders* (Voir les Figures 2 et 3 pour des exemples de résultats).

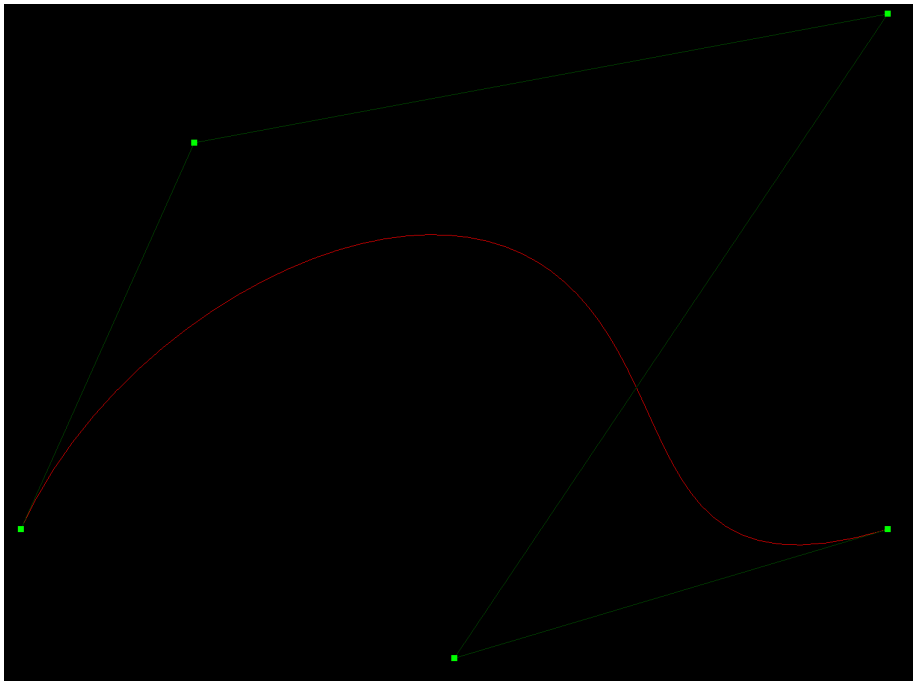

Figure 2 – Courbe de Bézier ($N = 4$)

3 Tâches encore à réaliser

Pour ce qui est des courbes et des surfaces de Bézier, il reste à implanter la possibilité de déplacer les points de contrôle via la souris. Pour ce qui est des surfaces de Bézier, il

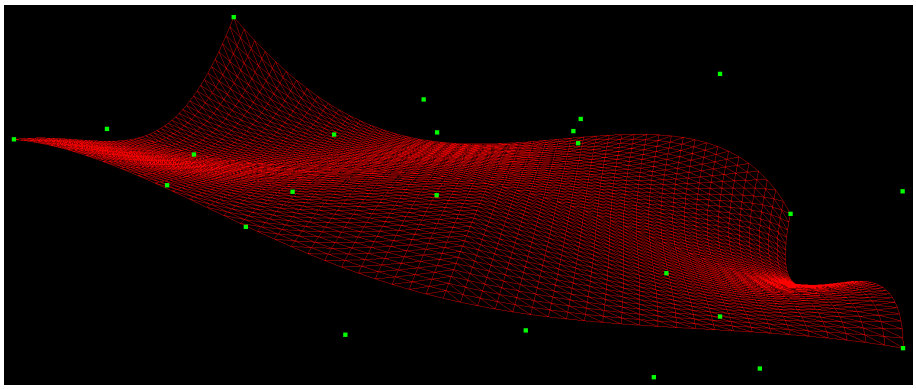

Figure 3 – Surface rectangulaire de Bézier ($N = 5, M = 3$)

manque le calcul des normales et l'implantation d'un modèle de Phong.

Une fois ces deux fonctionnalités programmés, il faudra implanter les carreaux de Gregory en eux mêmes.

Références

[1] OpengGL Wiki, *Tessellation*, https://www.khronos.org/opengl/wiki/Tessellation.

[2] Apple Developer, *OpenGL Capabilities Tables*, 2017, https://developer.apple.com/opengl/OpenGL-Capabilities-Tables.pdf.

[3] *Approximating Subdivision Surfaces with Gregory Patches for Hardware Tessellation*, C. Loop, S. Schaefer, T. Ni, I. Castaño, 2009, http://faculty.cs.tamu.edu/schaefer/research/greg.pdf.

[4] *Multisided generalisations of Gregory patches*, G.J. Hettinga, J. Kosinka, 2018

[5] *A Multi-sided Bézier Patch with a Simple Control Structure*, T. Várady, P. Salvi, G. Karikó, 2016