Rapport final de TIPE

PEREIRA Romain

4 Juin 2017

Sommaire

Préambule

Les objectifs du MCOT ont été mené à termes : j'ai étudié en détail les champs de hauteurs, et été en mesure de faire une synthèse d'image rapide. Cependant, un léger infléchissement a eu lieu quant à l'étude spatiale et temporelle de la rastérisation. En effet, ces problématiques sont apparues être trop vastes pour être traitées entièrement dans le cadre de mon TIPE. Ainsi, j'ai décidé de me limiter à une présentation rapide du procédé dans mon étude.

1 Introduction

Les algorithmes de rendu graphique en temps réel consistent à générer des images numériques, qui sont calculées et affichées suffisament rapidement (de l'ordre des 10ms/image) pour faire l'illusion de continuité sur l'œil humain. Mon travail s'est orienté sur l'étude de grille uniforme bidimensionnelle surelevées, générée à partir de champs de hauteurs. On peut ainsi synthétiser l'image d'un terrain rapidement grâce à une rastérisation. Ce TIPE décrira le processus de rendu, la façon dont on crée des champs des hauteurs, et les structures de données optimales pour les stocker en mémoire.

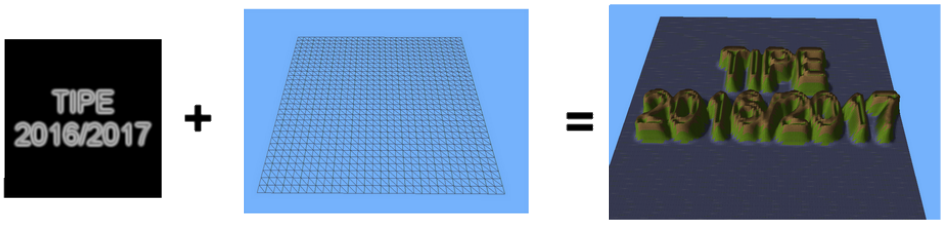


Figure 1: *Champ de hauteur + grille uniforme = terrain*

2 Corps principal

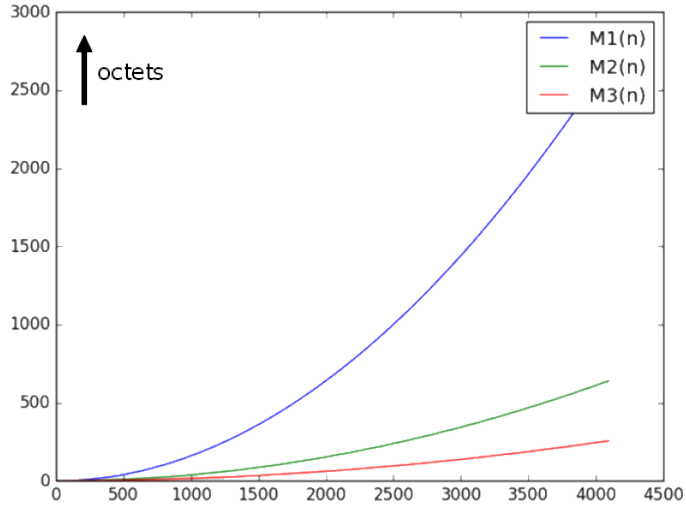
2.1 Modalités d'action

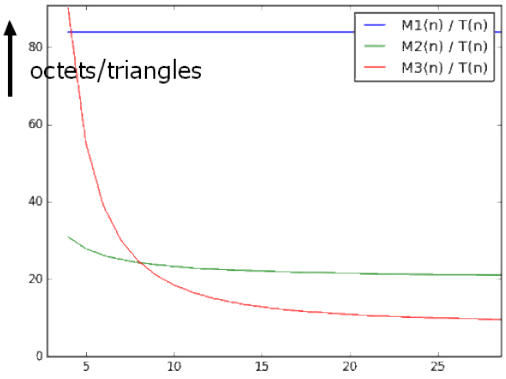
J"étudie la théorie de deux méthodes de rendu graphiques : le lancer de rayon et la rastérisation. La 1ère méthode se base sur l'optique géométrique (retour inverse de la lumière), alors que la 2ème sur de la géometrie projective à laquelle on ajoute des modèles d'éclairage approximatif. [3] Dans la synthèse d'image numérique, les objets sont représentés virtuellement sous forme d'une liste de vecteurs, qui forment un maillage surfacique de l'objet à partir de primitives géometriques (triangles, carrés, sphères...).

Il m'a donc fallu être en mesure de générer de tels objets. Je me suis interessé aux champs de hauteurs [4], qui sont des champs scalaires à 2 coordonnées : à une position « (x, z) » , elle associe une hauteur « y ». Afin de mettre en pratique ces études, j'ai créé un programme capable de travailler avec les champs de hauteurs. J'ai généré des champs de hauteur discrets (à partir d'image BMP), et continus (via l'interpolation de fonctions de bruits [1]). Ils m'ont permis de génerer des surfaces de type terrain, que j'ai par la suite triangulariser. Finalement, les terrains sont envoyés dans une rastérisation pour être rendu graphiquement.

2.2 Restitution des résultats

Le lancer de rayon permet d'obtenir des images plus fidèles à la réalité, mais est plus lent que la rastérisation. Donc pour faire de la synthèse d'image en temps réel, la rastérisation est plus adapté.
La complexité spatiale des champs de hauteurs m'a conduit à élaborer et comparer 3 méthodes de représentation des données du terrain. Le résultat de cette étude est donnée ci dessous.





format	M_1	M_2	M_3
octet / vecteur	$S=28$	$S=28$	$S=16$
octets / triangles	84	20	8

Figure 2: *Utilisation mémoire d'un terrain carré avec n vecteurs par coté*

J'ai decomposé l'étude de la complexité temporelle en deux sous-parties.

• Il y a premièrement un 'temps de préparation', qui consiste à envoyer les données sur le serveur de calcul (carte graphique), et à initialiser la parallélisation. Ce temps dépend considérablement de

l'architecture du serveur (nombre de coeurs de calculs à disposition, bande passante...) La bande passante peut varier de 1 à 100 GO/s selon l'architecture... ce qui représente 1Mo/ms à 1Go/ms.
• La deuxième sous-partie est l'execution de la rastérisation. Elle peut être interpreté comme le nombre de 'triangle par secondes' que le serveur est capable de calculer. Ce rapport n'est pas absolument déterminable, car il dépends de l'architecture de calcul. Sur une carte graphique moderne, qui ne serait sollicitéé que par un seul programme de calcul, il est de l'ordre de 10 à 1000 millions de triangles par secondes.

2.3 Analyse - Exploitation - Discussion

Le temps de calcul théorique précis n'a pas pu être entierement determiné. En effet, de nombreux facteurs entre en compte, nottement: la mémoire disponible, la capacité de parallélisation du serveur de calcul, la bande passante...

J'ai donc cherché à optimiser l'espace mémoire requis car la complexité spatiale reste très étroitement lié aux performances temporelles. En effet, pour n=4096, il aurait fallu 2.5Go pour stocker les données, ce qui represente dans un cas moyen (bande passante de 50Go/s <=> 0.5Go/ms), un temps de préparation de l'ordre des 5ms. (avec 33 538 050 triangles formant le maillage, la contrainte des 10ms/image inenvisageable.) Bien que le rendu puisse contenir quelques millions de triangles, le 'temps de préparation' est non negligeable devant le temps d'execution de la rastérisation (car cette méthode se prête à une parallélisation [2])
Avec le dernier format de stockage, ce temps chute à 0.5ms. Je juge la complexité asymptotique de ce dernier format comme optimale pour des grilles uniformes.

$$C(n) = S\star\ n^2 + o(n^2)$$

Figure 3: *Utilisation mémoire d'un terrain carré n vecteurs (de S octets) par coté*

En effet, n^2 correspond au nombre total de vecteurs (indépendemment du maillage que l'on en fait), et 'S' la taille d'un vecteur. Il est nécessaire d'avoir tous les vecteurs en mémoire au moins une fois afin de faire le rendu, on ne peut donc théoriquement pas faire mieux, asymptotiquement et avec des grilles uniformes.
On peut cependant imaginer un maillage non-uniforme : en combinant des triangles similaires (i.e vastes étendues planes), on pourrait réduire sensiblement le nombre de triangle. En revanche, le travail d'optimisation spatiale qui a été effectué ne s'applique pas sur des grilles non-uniformes, c'est pourquoi je ne m'etendrais pas sur leur cas dans ce TIPE.

De plus, j'ai mis en place une technique d'optimisation : le 'Frustum Culling'. [5]
Ce procédé consiste à déterminer, avant l'envoie des données sur le serveur de calcul, quels objets sont dans le champ de vision (et seront donc bien projeté sur le plan de l'image). Cette technique permet d'économiser de la mémoire à transférer, et des vecteurs à projeter.

Une autre optimisation qui n'a pas été mis en oeuvre, mais qui aurait pu donner lieu à un gain de performance, est l'implémentation d'un système 'niveau de détail'. Cela consiste à réduire le nombre de triangles des terrains eloigné du plan de projection.

3 Conclusion générale

Mon TIPE a donc été centré sur la synthèse d'images d'un terrain virtuel, utilisant la rastérisation. En réduisant la mémoire nécessaire, j'ai pu réduire le temps de rendu. Après l'implémentation du programme, les techniques mises en oeuvre décrite dans ce document m'ont permis d'avoir un temps de rendu variant de 10 à 20 ms, pour un terrain de 100 000 à 2 000 000 de triangles. J'obtiens donc une simulation fluide pour l'oeil.

References

[1] Adrian BIAGIOLI, *Understanding Perlin noise*,
https://flafla2.github.io/2014/08/09/perlinnoise.html,
Date d'accès: 9 Août 2014.

[2] NVIDIA, *What is gpu computing*,
http://www.nvidia.com/object/what-is-gpu-computing.htmll,
Date d'accès: ????

[3] The University of Texas at Austin, *The Phong illumination model*,
http://www.cs.utexas.edu/~bajaj/graphics2012/cs354/lectures/lect14.pdfl,
Date d'accès: Printemps 2013.

[4] Wikipédia, *Champ de hauteur*,
https://fr.wikipedia.org/wiki/Champ_de_hauteur,
Date d'accès: 11 Août 2016.

[5] _____, *Détermination des surfaces cachées*,
https://fr.wikipedia.org/wiki/Détermination_des_surfaces_cachées,
Date d'accès: 22 Janvier 2017.