Rendu graphique: la synthétisation d'une image à 3 dimensions

- Optimisation des algorithmes de synthetisation d'image, afin d'obtenir une image en "temps réel" (~ 15ms/image)
- Contraintes mémoires / temps (choix de structures / d'algorithmes optimales, parallélisation)
- Hasard: génération procédurale de surface afin de tester l'algorithme

- Motivation: le rendu graphique est un sujet qui me passionne. Les algorithmes utilisés associes rigueur, astuces et élégances. De plus, les applications sont nombreuses et passionantes: simulations de modèles physiques / mathématiques (il est parfois intéressant d'avoir un visuel du probleme), médecines (IRM, logiciels de simulations...)

Positionnement thématique

INFORMATIQUE (Informatique Théorique), INFORMATIQUE (Informatique pratique), MATHEMATIQUES (Géométrie).

Mots-clés

Mots-Clés (en français)	**Mots-Clés** (en anglais)
Infographie	*Brensenham line algorithm*
Matricialisation	*Rasterization*
Triangulation	*Phong shading*
Champ de hauteur	*Heightmaps*
Image de synthèse	*Perlin Noise*

Bibliographie commentée

L'infographie **[1]** est une discipline apparue au cours des années 1970, qui regroupent différentes techniques sur la synthèse d'image numérique. Son développement est très étroitement lié à l'augmentation des capacités de calcul des ordinateurs. Dans le cadre du TIPE, je m'intéresse à la rastérisation **[2]**, un procédé infographique utilisé très largement pour le rendu graphique « en temps réel »: chaque image doit avoir un temps de calcul de l'ordre de 10 ms pour tromper la capacité de perception de l'oeil et procurer une sensation d'instantanéité.

La rastérisation qui consiste à transformer une scène virtuelle à 3 dimensions en une image à 2 dimensions (fait à partir de pixels, qui sera physiquement affiché à l'écran). Une scène se représente généralement sous forme d'une liste de vecteurs, où chaque vecteur contient des informations sur un point : une position (x, y, z) dans le repère de la scène, une couleur (r, g, b, a), un vecteur normal (nx, ny, nz), une masse (m)... ou tout autres paramètres utiles au rendu. Ces vecteurs sont indicés afin de former le maillage triangulaire de la scène. **[3]**
Lors de ce procédé, une première étape consiste à faire subir à tous les vecteurs, et ceci en exécution parallèle sur carte graphique **[4]**, des transformations via des applications matricielles (rotations, translations, mises à l'échelle) **[2. section «Transformation Model to World »]**. C'est également pendant cette étape que le calcul de la luminosité et des couleurs est effectué, je

me suis basé sur le modèle d'ombrage de Phong **[5]**.
Ensuite, les vecteurs sont projetés dans la base de l'image plane, pour finalement obtenir une nouvelle liste de vecteur prêt à être dessiné. Lors du tracé, les triangles sont dessinés et remplis via un algorithme se basant sur le tracé de ligne de Brensenham **[6]**

La suite de mon travail s'est orientée sur la création de scènes tridimensionnelles, afin de les injecter dans le procédé de rastérisation. Je me suis interessé aux « Height-maps » **[7]** (ou encore « champ de hauteur »). Ce sont des champs scalaires bi-dimensionnels : pour un point de coordonnées (x, y), la heightmap renvoie un réel correspondant à la hauteur du point. (avec x, y, dans un ensemble dénombrable)

J'ai conçu, et optimisé mon propre programme me permettant de travailler avec les heightmaps :
• La génération procédurale: la heightmap est générée « procéduralement » via l'utilisation de fonctions de bruits cohérents **[8]** (nottament le bruit de Perlin) **[9]**. On peut alors simuler des reliefs montagneux aisément.
La lecture d'heightmaps : stockées sous format image BMP, permettant ainsi de visualiser des heightmaps préexistantes (carte de France, données GPS...)
• Mon programme est donc capable de générer des heightmaps, de les triangulariser, puis de l'injecter dans le procédé de rastérisation, pour finalement obtenir une image plane

La contrainte temporelle est au cœur du problème. Par exemple, dans un simulateur de vol, le pilote a besoin d'une simulation fluide pour être en bonnes conditions d'apprentissage.

Problématique retenue

Effectuer des rendus graphiques est couteux en calculs. Le problème du « temps réel » est au centre du sujet, il s'agit donc de comprendre quelles optimisations algorithmiques entre en jeux pour se plier à la contraite temporelle.

Objectifs du TIPE

1. Géométrie, Informatique théorique, Optique : j'étudie en détail le procédé de rastérisation, en explicitant le modèle mathématique utilisé, en evaluant la complexité (temporelle et spatiale) des algorithmes, et en me basant sur un modèle d'éclairage infographique

2. Simulation numérique : j'implementerai un système de « heightmaps » en C (par soucis d'optimisation), en utilisant la librairie OpenGL 4.0 qui permet de faire de la programmation parrallèle sur carte graphique, et d'accéder aux périphériques de l'ordinateur (souris, clavier)

3. Ouverture professionnelle : j'ai pris contact avec un ex-employé de Thalès, qui y programmait un simulateur de vol

Références bibliographiques

[1] Wikipédia : Infographie : *https://fr.wikipedia.org/wiki/Infographie*

[2] Wikipédia : Rastérisation : *https://fr.wikipedia.org/wiki/Rast%C3%A9risation*

[3] Wikipédia : Maillage triangulaire : *https://fr.wikipedia.org/wiki/Mesh_(objet)*

[4] NVIDIA : Parralélisation : *http://www.nvidia.com/object/what-is-gpu-computing.html*

[5] The University of Texas at Austin : Modèle d'éclairage de Phong : *http://www.cs.utexas.edu/~bajaj/graphics2012/cs354/lectures/lect14.pdf*

[6] Bastian MOLKENTHIN : Brensenham line algorithm : *http://www.sunshine2k.de/coding/java/TriangleRasterization/TriangleRasterization.html*

[7] Wikipédia : Heightmaps : *https://en.wikipedia.org/wiki/Heightmap*

[8] Amit PATEL, redblobgames : Génération procédurale : *http://www.redblobgames.com/maps/terrain-from-noise/*

[9] Adrian BIAGIOLI : Bruit de Perlin : *http://flafla2.github.io/2014/08/09/perlinnoise.html*