GMINT317 - Moteurs de jeux – TP5

Textures et GPU

Rémi Ronfard remi.ronfard@inria.fr https://team.inria.fr/imagine/team/

# Objectifs

Dans ce TP, nous allons nous intéresser à étudier les possibilités offertes par le GPU. Ainsi, ces travaux vont vous demander de vous familiariser avec différentes techniques :

* Pour habiller vos objets 3D
* Pour appliquer des transformation sur un objet
* Pour améliorer le tendu de votre application précédente
* Améliorer le gestionnaire de ressources pour supporter :
  + les textures
  + les shaders
* Garder un rendu temps réel

**Bonus :**

* Réaliser des calculs parallèles en utilisant le GPU
* Evaluer les performances de vos primitives en CPU vs GPU
* Utiliser le GPU pour réaliser les calculs d’interaction entre les particules

(cf TP précédent : pluie, neige, eau)

* Réaliser un shader pour modéliser l’écoulement
* Améliorer le gestionnaire de ressources pour le rendre dynamique

# Gestionnaire de version

Cloner le quatrième TP :

git clone http://lange@www.lange.xyz/GMIN317/TP/groupe<#>/tp5.git

Au plus tard la semaine prochaine, vous devrez rendre un compte rendu de ce TP sur votre espace GIT, ainsi que votre code source.

# Texture d’un triangle

Dans cet exercice, nous allons reprendre le projet suivant :

http://qt-project.org/doc/qt-5/qtgui-openglwindow-example.html

Ici, nous allons texturer notre triangle, d’abord avec des couleurs, puis avec une image de votre choix.

Puis nous allons utiliser le filtrage de couleur pour modifier le rendu de la texture, jouer également avec la transparence.

Ensuite, modéliser une pyramide, et une surface plane qui représente le sol, faite léviter la pyramide et la faire tourner sur elle-même.

Texturer le sol ainsi que la pyramide (chaque face de la pyramide devra avoir une texture différente).

**Shader sur le triangle**

Toujours sur notre pyramide, ajouter à votre scène une sphère et mettre en place différents shaders :

* Un éclairage diffus
* Une méthode de : *spherical environment mapping*
* Une méthode de *normal mapping*
* Et une déformation de la géométrie

**Texture et Shader sur le terrain**

Maintenant nous allons reprendre les travaux précédents.

Texturer votre terrain à partir de couleurs (adapté à l’altitude) si vous ne l’avez pas encore réaliser. Maintenant, appliquer une texture unique pour toute les fenêtre (adapté au terrain toujours).

Pour chaque saison (donc fenêtre) appliqué une coloration sur la texture.

Ajouter un shader *normal mapping,* pour donner un rendu accidenté.

Ajouter un shader cartoon pour le rendu de votre scène.

Veuillez bien à conserver l’aspect temps réel de votre application.

# Compte rendu

Présenter toutes vos nouvelles fonctionnalités

Expliquer les points que vous n’êtes pas arriver à réaliser et pourquoi.

Expliquer votre démarche de développement.

Présenter votre structure de données.

Expliquer comment vous vous y prendriez pour les parties bonus.

# Bonus

* Réaliser des calculs parallèles en utilisant le GPU

Ré-implémenter votre librairie mathématique en utilisant openCL.

* Evaluer les performances de vos primitives en CPU vs GPU
* Utiliser le GPU pour réaliser les calculs d’interaction entre les particules ou les objets

Dans le TP précédent, nous avons mis en place une chute de pluie, et neige. Vous pouvez maintenant utiliser le GPU pour calculer ces chutes.

Si vous avez également mis en place l’accumulation de particules, vous pouvez également vous servir du GPU afin de réaliser une simulation réaliste de fluide, notamment pour les rivières.

* Réaliser un shader pour modéliser l’écoulement

Pour l’accumulation d’eau sous forme de rivière, utiliser un shader pour donner un rendu d’eau.

* Améliorer le gestionnaire de ressources pour le rendre dynamique

Permettre de charger dynamique différents données : shaders, textures, objets..