SIRE – TD

Programmation Graphique : Premiers pas (3)

Le but de ce TD, le dernier de la série de 3, est de continuer l'exploration du *pipeline* et des ses concepts pour la synthèse d'image sur carte graphique, et donc leur programmation. Dans cette séance, nous explorerons les méthodes optimisant le transfert d'information entre le CPU et le GPU et l'utilisation des textures. Faisant suite au TP précédent, il se basera sur le code écrit au TP2 et modifié suivant les instructions.

Encore une fois , pour toute documentation, profitez d'Internet et de tout ce qui s'y trouve ! Creusez, fouillez, apprenez à trouver la solution par vous-même ! Sinon, vous pouvez toujours consulter les pages concernant OpenGL.

http://www.opengl.org/resources/

1. Utilisation des VBOS

L'affichage actuel du terrain est réalisé via des appels aux fonction `glVertexAttribPointer`, qui indique à `OpenGL` ou sont placés les buffers de données, coté mémoire `CPU`. Ces données sont ainsi transmises à chaque frame de la `RAM` vers la `G-RAM`.

`OpenGL` propose un mécanisme permettant d'envoyer une seule fois un buffer sur la `G-RAM`, puis d'en demander l'affichage lors du rendu : ce sont les `VBO(Vertex Buffer Object)`.

Ajoutez dans votre classe `Relief3D` une méthode `updateVBO()` réalisant le transfert des buffers sur la carte. Cette méthode sera appelée à la fin de chaque appel à la méthode `Generate`, et se chargera d'envoyer sur la carte le contenu de chaque buffer (positions, normales, indices).

Mots clefs : `glBindBuffer`, `glBufferData`
Une fois cela effectué, modifiez votre fonction d'affichage pour utiliser les buffers stockés sur la carte et non coté CPU. Controllez les gains de performance.

2. Les textures – Vertex Shader

En synthèse d'image, l'utilisation de textures permet d'améliorer considérablement l'apparence graphique des applications. L'objectif de cet exercice est de remplacer les couleurs unies utilisées actuellement par des textures.

Dans un premier temps, utilisez la classe `Texture` pour charger la texture `htexture.pnm`. (classe et fichier texture disponibles dans l'archive `tp3.tar.gz`).

Rendez cette texture accessible à votre vertex shader via un `Sampler1D`, et remplacez le calcul des couleurs par une lecture dans la texture en fonction de la coordonnées en z des sommets.

Que constatez vous pour les transitions eau/sable, foret neige ?

3. Les textures – Fragment Shader

Modifiez votre code pour donner au fragment une couleur correspondant à un accès à la texture à une coordonnée en Z interpolé en fonction des sommets voisins.

Quel changement observez vous ?