TP Cartes Graphiques - GLSL

Sébastien Barbier

7 janvier 2009

1 Pour commencer

Télécharger l'archive `tp_glsl.zip` à l'adresse suivante :
`http ://evasion.inrialpes.fr/Membres/Sebastien.Barbier/Enseignement/`.

Vous y trouverez aussi sur cette page un ensemble de liens utiles comme les dépendances aux bibliothèques, des éditeurs de shaders et des déboggueurs *OpenGL*.

A l'intérieur vous trouverez deux dossiers contentant deux solutions *Visual C++* qui vous permettront de vous familiariser avec les shaders `GLSL`.

- fragment_demo.vcproj
- geo_demo.vcproj

2 Vertex et Fragment Shaders

Compilez le projet *fragment_ demo.vcproj* et faites le tourner. Ouvrez ensuite le fichier main_fragment.cpp et comprenez-le. Faites-en de même avec les shaders sandbox.vs et sandbox.fs.

Puis, essayez l'ensemble des shaders fournis. Modifiez le programme principal afin de les compiler et de les executer. Ensuite, ouvrez les sources et comprenez-le.

N'hésitez pas à faire des modifications par vous-même pour regarder ce que cela modifie. Modifiez par exemple la variable `gl_FragColor`.

Vous pouvez aussi utiliser le *shader designer de TyphoonLabs* pour faire tourner d'autres shaders

3 Geometry Shader

Compiler le projet *geo_ demo.vcproj* et faites les tourner. Ensuite, ouvrez le fichier principal main_geo.cpp et identifiez les lignes d'appel spécifiques à la compilation du geometry shader. Puis, ouvrez les fichiers de shaders associés et comprenez-les.

Ceci réalisé, imaginons que le shader ne fonctionne pas. Nous pouvons soit faire tourner le déboggueur `GLSLDevil` soit écrire un autre shader pour afficher les normales par sommets. Essayez les deux solutions si vous avez le temps.

Dans le second cas, on souhaite écrire un autre geometry shader que l'on appelera `debug_normal` qui transformera les triangles du maillage d'entrée en points, un point par triangle, par exemple en son barycentre. Le point résultant sera coloré en fonction de la valeur de la normale et du point de vue qui permet d'extraire les silhouettes. Nous pourrions imaginer une valeur sombre pour les valeurs négatives (gris-noir) et des valeurs claires (rouge, vert, bleu selon les directions) pour les valeurs positives. L'idée étant de rapidement trouver les triangles visibles et ce qui ne le sont pas et ainsi retrouver les positions possibles des silhouettes.