Exercices du module OpenGL de base

Contact: jean.lefeuvre@telecom-paristech.fr

1. Prérequis

La configuration présentée ici a été testée sous Windows et MacOS X et devrait fonctionner sans souci sous Linux.

Vous utiliserez la librairie lwjgl, surcouche Java autour d'OpenGL. Cette librairie est disponible en ligne : http://lwjgl.org/

Vous devez avoir un JDK récent sur votre machine (Java 6 ou 7). La librairie oepnGL utilisée peut travailler en mode 32 bits ou en mode 64 bits.

2. Installer lwjgl

Télécharger les binaires la dernière version (3.1) de la librairie sur https://www.lwjgl.org/download.

Suivez les instructions d'installation disponibles en ligne https://www.lwjgl.org/guide

Si vous n'utilisez pas la méthode ci-dessus, vous pouvez alternativement configurer directement votre poste de travail en modifiant vos variables d'environnement.

Rappel : Les variables d'environnement sous Windows se modifient via:

click droit sur "Ordinateur"->propriétés->Paramètres systèmes avancés->Variables d'environnement.

Les variables sont séparées par un point-virgule ';'.

Votre classpath devra comporter au moins

- lwjgl.jar, lwjgl-opengl.jar, lwjgl-glfw.jar
- lwjgl-glfw-natives-$PLATFORM.jar et lwjgl-natives-$PLATFORM.jar

3. Commencer vos travaux

Vous pouvez maintenant commencer à manipuler OpenGL.

Pour Windows/Linux/Mac, utilisez vos outils Java préférés pour exécuter le code du fichier *http://perso.telecom-paristech.fr/~lefeuvre/PACT/OpenGL/PACTBaseGL.java*. Le programme créé affiche un simple triangle dans une fenêtre OpenGL.

Dans le cadre de PACT, un certain nombre d'exercices de bases vous sont demandés pour la prise en main d'OpenGL, comme indiqué dans le code source. Pour chaque phase complétée, vous enverrez un courriel à votre expert pour le prévenir, et mettrez votre GIT PACT à jour.

Pour toute information complémentaire sur LWJGL, vous pourrez consulter leur documentation en ligne :

https://javadoc.lwjgl.org/

Inspirez-vous aussi du guide d'introduction à OpenGL, aussi appelé « Red Book » :

http://www.glprogramming.com/red/

Et n'oubliez pas toutes les ressources bibliographiques en place sur le site de PACT :

http://pact.enst.fr/composants/blocs-fonctionnels/la-3d/synthese-3d/

Vous aurez en plus de vos exercices une petite bibliographie à effectuer tout au long du module, que vous enverrez à votre expert avant l'évaluation finale du module.

Les tutoriaux ne répondront pas forcément aux questions que vous vous posez, n'hésitez surtout pas à contacter votre expert si vous avez besoin d'aide.

4. Liste des exercices

Cette liste est aussi décrite dans le fichier *GLBaseModule.java*

Phase 0: Biblio OpenGL

Cette biblio se fera tout au long de votre module, et comprendra de manière succincte:

- Explication du modèle OpenGL pour le fonctionnement du processeur graphique: quels sont les blocs fonctionnels du GPU et quelles sont leurs fonctions,
- Principes de base de OpenGL: Représentation des données et des transformations géométriques: Matrices, vertex, lignes/triangles/..., projection et modèle de caméra,
- Principes de base de OpenGL: Texture et système de coordonnées,
- Principes de base de OpenGL: Modèle de lumière,
- Principes de base de OpenGL: Expliquez les différences entre *DisplayList*, *VeretxArray* et *VertexBufferObject*

Phase 1: Manipulation de base en OpenGL

Exercice 1.1:

- utilisez une projection perspective au lieu d'une projection orthogonale
- expliquez la différence entre les modes de projections

Exercice 1.2:

- Dessinez un carré
- Dessinez un carré à l'aide d'un "éventail" de triangles (Triangle Fan), puis à l'aide d'un "ruban" (Triangle Strip)

- Dessinez un cube au lieu d'un triangle, avec une couleur par face, puis une sphère ou tout autre objet

Exercice 1.3:

Modifiez cette routine pour afficher votre scène d'au moins deux points de vues en même temps, en partageant votre zone d'affichage en plusieurs zones via glViewport. Vous devrez rendre votre scène une fois par point de vue. Ceci vous sera utile dans votre projet pour gérer des zones d'affichages différentes, pour débugger votre contenu ou afficher des barres d'outils indépendantes de votre transformation de point de vue

Exercice 1.4:

Sans changer votre caméra, faites tourner un cube sur lui-même

- Manuellement via le clavier (flèches de direction)
- de manière automatique en faisant un tour par seconde

Vous ferez pour cela un contrôle de la vitesse de rafraichissement en décidant de la fréquence des trames affichées. Ceci est classiquement appelé "frame rate", "FPS" ou "Frame Per Second". Les jeux utilisent classiquement des fréquences de trames entre 30 et 60Hz.

Votre rotation se fera à l'aide en modifiant la matrice de modèle courante via une matrice de rotation.

Exercice 1.5:

Sans changer les coordonnées de vos objets, animer le point de vue de votre caméra virtuelle :

- manuellement via le clavier (flèches de direction)
- de manière automatique en faisant un tour par seconde

Phase 2: Manipulation de l'aspect des objets

Exercice 2.1:

Chargez une image et dessinez l'image sous forme d'un rectangle. **ATTENTION :** vous ferez attention à ne pas déformer l'image.

Vous pourrez vous inspirer de l'exemple suivant :

http://www.java-gaming.org/topics/lwjgl-stb-bindings/36153/view.html

Exercice 2.2:

Chargez 3 images différentes et utilisez les pour remplir les faces d'un cube.

Exercice 2.3:

Vous effectuerez dans un premier temps un petit travail bibliographique sur le modèle de lumière en OpenGL. Puis vous effectuerez les tests suivants, en faisant particulièrement attention à vos définitions de normales !

- Placez une lumière dans votre scène (sans texture) éclairant un cube placé au centre de la scène,
- modifiez les paramètres de couleurs (souvent appelé « material colors» car utile pour modéliser l'apparence de la matière) pour changer l'apparence de votre cube,
- faites tourner votre cube ou votre lumière pour visualiser les changements liés à l'éclairage.

Exercice 2.4:

Sans modifier vos paramètres de lumières (source de lumière et couleurs de matériel de l'objet), animez légèrement la normale de chaque face du cube (une légère rotation autour de la vraie normale) et observez le résultat.

Phase 3: Vers plus de puissance et de souplesse!!

Les exercices que vous avez effectués jusqu'ici utilise le modèle d'origine d'OpenGL, souvent appelé "fixed pipeline": vous ne pouvez pas modifier la manière dont votre carte graphique dessine un pixel. Vous allez dans cette phase étudier les GPU modernes qui permettent ce genre de manipulation.

Exercice 3.1:

En partant de votre cube, créez un VertexBufferObject (ou VBO) pour dessiner votre cube de couleur uniforme.

- définissez l'ensemble des triangles composants le cube dans un VBO, et dessinez le cube en utilisant la fonction glDrawArrays,
- définissez l'ensemble des points du cube dans un VBO, créez un VBO contenant les index définissant les triangles et dessinez le cube en utilisant la fonction glDrawElements,

Exercice 3.2:

En partant de l'exercice précédent :

- Ajoutez une couleur par face ou par sommet à votre cube.
- Ajoutez une texture par face à votre cube.
- Ajoutez une lumière à votre scène, et spécifiez les normales à votre cube.

Exercice 3.3:

En partant de votre cube précédent (VBO) et en vous référant à un tutoriel en ligne (prenez celui de lwjgl ou d'android par exemple), créez un "programme" openGL utilisant un *vertex shader* et un *fragment shader* très simples pour dessiner votre objet.

Exercice 3.4:

En partant de l'exemple précédent, animer votre cube SANS CHANGER LE VBO pour:

- modifier très légèrement les coins du cube dans le vertex shader à partir d'un paramètre externe (lu depuis le shader mais défini dans votre programme java),
- modifier très légèrement les couleurs du cube dans le fragment shader à partir d'un paramètre externe (lu depuis le shader mais défini dans votre programme java)
- modifier très légèrement les couleurs du cube dans le fragment shader à partir d'un paramètre externe (lu depuis le shader mais défini dans votre programme java)
- ou d'un paramètre calculé dans le vertex shader, comme un vecteur normal altéré

Exercice 3.5:

En partant de l'exemple 3.3, utiliser les VBO pour ajouter une texture à votre cube (le tutoriel de lwjgl à ce sujet est très bien).