



Université Clermont 1  
M1 Master Informatique et Systèmes  
Année universitaire 2011-2012

## Synthèse d'images 3D avec OpenGL

# TP n° 1 Premiers pas avec *glut* et *OpenGL*

### Objectifs :

Le but de ce TP est de se familiariser avec les commandes de base de *glut*, et de pratiquer les fonctions `gluPerspective` et `gluLookAt` pour se familiariser avec leurs paramètres. On réalisera aussi le chargement de fichier de scènes 3D avec la librairie *assimp*.

Pour compiler, on utilisera simplement GCC dans la console avec les librairies `glut`, `GL` et `GLU` (contenant OpenGL et les utilitaires associés).

```
gcc -lglut -lGL -lGLU exemple.c -o exemple
```

Pour installer ces librairies chez vous instalez simplement le paquet `freeglut3-dev`.

Sous Ubuntu Oneiric 32 bits le linker de gcc (version 4.6.1) ne marche pas aux dernières nouvelles. Utilisez `gcc-4.4` au lieu du gcc par défaut :

```
sudo apt-get install gcc-4.4  
gcc-4.4 -lglut -lGL -lGLU exemple.c -o exemple
```

**Exercice 1** Créer un programme *glut* avec création d'une fenêtre graphique. Tester.

**Exercice 2** Ajouter au programme un événement d'affichage avec comme code l'effaçage de l'écran.

**Exercice 3** Ajouter au programme des fonctions de gestion des événements suivants :

- Enfoncement d'une touche normale au clavier ;
- Enfoncement d'une touche spéciale au clavier ;
- Enfoncement ou relachement d'un bouton de souris ;
- Mouvement de la souris.

Pour l'instant, on fera un code minimal pour que les fonctions compilent.

**Exercice 4** Faire une fonction de redimensionnement avec réglage de la transformation 2D par `glViewport` pour que le centre de la fenêtre ait pour coordonnées (0,0), sans changement d'échelle. Régler la caméra avec un angle d'ouverture en  $\gamma$  égal à 50 degrés.

**Exercice 5** Modifier la fonction d'affichage pour qu'elle affiche une théière de taille 5. On fera un appel à `gluLookAt` de manière que la théière soit visible et apparaisse verticale dans le champs de vision. Modifier les paramètres de `gluLookAt` en essayant d'imaginer la position de la caméra.

**Exercice 6** Définir des variables globale `posx`, `posy`, `posz` pour la position de la caméra, et des variables `centrex`, `centrey`, `centrez` pour le point de la direction de visée.

**Exercice 7** Créer un mouvement (en positif et en négatif) des coordonnées  $x$   $y$  et  $z$  de la position de la caméra lors de la pression des touches  $x$ ,  $X$ ,  $y$ ,  $Y$ , et  $z$ ,  $Z$ .

**Exercice 8** Créer un mouvement des coordonnées  $x$  et  $z$  du centre de la vue lors d'un mouvement de la souris avec le bouton droit enfoncé.

**Exercice 9** Créer un mouvement de la coordonnée  $y$  de la position de la caméra lors d'un enfoncement des touches `haut` et `bas` du clavier.

**Exercice 10** Créer un mouvement de la coordonnée  $y$  du centre de la vue lors d'un enfoncement des touches `gauche` et `droit` du clavier.

**Exercice 11** Créer un mouvement d'éloignement ou de rapprochement de la position de la caméra lorsque l'utilisateur enfonce les touches `e` et `r`. On fera (coordonnée par coordonnée) :  
$$\text{nouveau\_pos} = \text{centre} + 1.1 * (\text{ancien\_pos} - \text{centre})$$
pour l'éloignement, et la transformation inverse pour le rapprochement.

**Exercice 12** Augmenter l'angle d'ouverture de la caméra lorsque l'utilisateur presse la touche `A`. Diminuer l'angle d'ouverture de la caméra lorsque l'utilisateur presse la touche `a`.

**Exercice 13** La librairie `assimp` est une librairie Open Source (disponible sur SourceForge.net) qui permet de charger en mémoire dans une structure de donnée les données d'une scène 3D dans différents formats. Les objets  $y$  sont représentés sous forme de maillage. Pour compiler, on utilisera les options suivantes :

```
gcc -I /usr/local/include/assimp/ -lglut -lGL -lGLU -lassimp exemple.c
```

Dans le menu build/environnement d'Eclipse, Ajouter le chemin `/usr/local/lib` dans la variable d'environnement `LD_LIBRARY_PATH` qui indique les répertoires contenant les librairies dynamiques :

```
LD_LIBRARY_PATH=/usr/local/lib/:LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

Dans votre code C, ajoutez les incluses suivants :

```
#include <assimp.h>          // C++ importer interface assimp
#include <aiScene.h>         // Structure de données de scène
#include <aiPostProcess.h> // Post processing flags
```

Téléchargez les fichiers de scène 3D sur <http://www.malgouyres.fr> rubrique "Enseignement-OpenGL".

Observez les structures de données dans les headers suivants du répertoire `/usr/local/include/assimp`

```
aiScene.h
aiMesh.h
```

Observez la documentation de la fonction `aiImportFile` dans `assimp.h`. Utilisez cette fonction pour charger une scène au format 3DS. On pourra par exemple utiliser les flags suivants :

```
aiProcess_FindDegenerates |
aiProcess_Triangulate |
aiProcess_SortByPType |
```

Voir la documentation de tous les flags dans `aiPostProcess.h`.

On pourra aussi se référer à la documentation doxygen disponible sur [assimp.sourceforge.net](http://assimp.sourceforge.net).

Réalisez enfin l'affichage wireframe de la scène précédemment chargée en parcourant le maillage.

### Complément : installer assimp sur son ordinateur Ubuntu ou Debian

Téléchargez le source de la librairie (l'archive `assimp--2.0.863-sdk.zip` ou version postérieure) sur [sourceforge.net](http://sourceforge.net), puis faire les commandes suivantes, (en adaptant avec un peu de bon sens si besoin!).

```
sudo apt-get install libboost-all-dev
unzip assimp--2.0.863-sdk.zip
cd assimp--2.0.863-sdk
mkdir ../lib
cmake .
# si problèmes, installer le compilateur ou les librairies manquantes
# (paquets de la forme lib*-dev) et refaire "cmake ."
make
sudo make install
```