UFR IEEA

Programmation 3D

2006–2007
Licence/Master
Fabrice AUBERT

OpenGL - TP 02

Reprenez `tp02.zip` sous `http://www.lifl.fr/~aubert/p3d/tp02.zip`. Compilez et exécutez : le programme ouvre seulement une fenêtre graphique. Toutes les initialisations nécessaires sont effectuées (callbacks, effacement d'écran, depth buffer,...).

Remarques générales :

- Vous trouverez les procédures spécifiques aux tracés à la fin du source.
- La projection est perspective.
- Le calcul d'éclairement est activé en `glColorMaterial` (il vous suffit de faire un `glColor` pour donner la caractéristique courante de réflexion diffuse des objets).
- Vous trouverez des "callbacks" aux messages provenant du clavier (voir par exemple `myKey` qui quitte le programme si appui sur "ESC"), et de la souris.
- On utilise `trackball.cpp` pour pouvoir tourner la scène avec la souris en maintenant le bouton gauche appuyé. La fonctionnalité est déjà intégrée dans les tps (instructions commençant par `tb` dans les squelettes) : **utilisez** le trackball lors de l'exécution de vos programmes pour mieux visualiser vos scènes.
- Ligne de compilation :

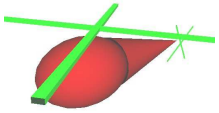
  make

Exercice 1 : Hélicoptère

Q 1. Dans `myDisplay`, activez l'appel à `tracerHelico`.

Q 2. Dans `tracerHelico`, faire un hélicoptère hiérarchiquement (décomposez avec `tracerCockpit`, `tracerRotor`, `tracerPale`, etc ; utilisez les `push-pop` !). Utilisez `glColor3f` pour donner de la couleur. L'hélicoptère est très simple (vous devez faire cet exercice en 20 minutes maxi) :

- Une sphère un peu «allongée» pour le cockpit.
- Un cône pour la queue.
- Deux cubes "allongés" pour le rotor principal (que vous ferez tourner lors de l'animation).
- Un rotor plus petit (tracé avec la même procédure que le rotor principal) pour le rotor de queue (et qui doit également tourner).



Faites tourner l'hélicoptère sur lui-même.

Exercice 2 : Eclairement

On travaille dans `tracerTores` (appelez `tracerTores` dans `myDisplay`). Vous devez voir deux tores en rotation (1 bleu et 1 vert). Changez éventuellement l'incrément de rotation dans `myIdle` si la rotation est trop lente.

Q 1. Pour avoir plus de liberté sur les caractéristiques d'éclairement et des matériaux, **désactivez** le GL_COLOR_MATERIAL (les propriétés diffuses ne seront plus spécifiées par `glColor3f`). Supprimez les deux lignes correspondantes dans `myInit`, puis compilez et exécutez (vous devez voir deux tores en "gris", ce qui correspond aux propriétés de matériel par défaut).

Notez que les `glColor3f` n'ont plus aucune influence (nous sommes en `GL_LIGHTING` sans `GL_COLOR_MATERIAL`). Supprimez les dans `tracerTores`.

Q 2. Dans `myInit` est appelée la fonction `initSource` avec `GL_LIGHT0`. Dans `initSource` sont initialisés tous les coefficients $(rouge, vert, bleu)$ d'une source lumineuse. **Constatez** comment on spécifie des propriétés liées à la source par `glLight`).

Appelez `materielRouge` avant d'afficher le premier tore. Visualisez (les **deux** tores doivent être rouge).

Remarque : dans `materielRouge` sont modifiées toutes les caractéristiques du matériel courant (reflexion diffuse, ambiante, spéculaire ; constatez l'aspect brillant donné par le spéculaire).

Q 3. Appelez `materielBleuVert` juste avant d'afficher le deuxième tore. Visualisez.

Mettez en commentaire l'appel à `materielRouge` du premier tore. Visualisez. Expliquez pourquoi le premier tore apparait maintenant en bleu-vert.

Retenez qu'il n'existe qu'un seul matériel qui est appliqué à tous les tracés. Les valeurs courantes de ce matériel peuvent être modifiées (ici le premier tore subit un matériel gris lors de la toute première image puis il subit le bleu-vert pour les images suivantes).

Remettez l'appel à `materielRouge` pour le premier tore.

Q 4. Modifiez le coefficient diffus dans `initSource` pour le mettre completement à vert (rouge et bleu à zéro). Visualisez (la lumière est verte : seules les composantes vertes des matériaux seront visibles ; cf formulations du cours).

Q 5. Dans initSource, remettez le coefficient diffus à completement blanc (tout à 0.8).

Q 6. Dans `tracerTores`, appelez `sourceLocal(GL_LIGHT0)` lorsque le repère courant est au centre du premier tore (constatez que la position spécifiée dans `sourceLocal` est l'origine du repère courant c'est-à-dire que la position de la source subit la `MODELVIEW`. Il suffit donc d'appeler `sourceLocal` au moment où le repère courant est celui du premier tore pour la placer au centre de ce tore).

Visualisez et observez que l'éclairement subit par les deux tores a changé, et correspond à la source placée au centre du premier tore (un cube blanc est tracé explicitement dans `sourceLocal` pour vérifiez la position de la source ; remarquez que ce cube ne subit pas l'éclairement car il est désactivé lors de son tracé).

Constatez qu'OpenGL ne gère pas les ombres portées : le premier tore ne génère pas d'ombre sur le second tore.

Q 7. Ajoutez un appel `sourceLocal(GL_LIGHT0)` entre les tracés des deux tores (entre le glPop et le glPush. Visualisez.

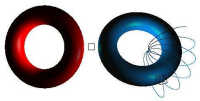
Constatez que nous n'avons pas deux sources lumineuses simultanées : le premier tore est tracé en subissant l'éclairement de `LIGHT0` placée en son centre, puis le second tore est tracé avec toujours `LIGHT0` mais qui a été déplacée entre les deux tracés (le tore rouge n'est pas éclairé par la source placée au milieu des 2 tores, puisqu'il est déjà tracé...).

Q 8. Supprimez les `sourceLocal` dans `tracerTores`. Visualisez et constatez que l'éclairement correspond à une source `LIGHT0` qui éclaire dans la direction Z.

Q 9. Activez une seconde source lumineuse avec `glEnable(GL_LIGHT1)` (faites le dans `myInit`) et appelez `initSource(GL_LIGHT1)`. Placez là au début de `tracerTores` avec `sourceLocal(GL_LIGHT1)`.

Visualisez. Constatez qu'on a bien deux sources simultanées (notez par exemple les deux spécularités sur chacun des objets).

Q 10.



Animez la source `GL_LIGHT0` pour qu'elle possède un mouvement hélicoidale autour de la section du **deuxième** tore (**Remarque :** vous devez faire ce mouvement au début de `tracerTores` ; le premier tore ne subira effectivement pas `GL_LIGHT0` si vous la placez après que son tracé soit fait. Ceci vous oblige donc à reproduire les changements de repère qui permettent d'arriver au centre du tore 2).

Q 11. Créez-vous un matériel `monMateriel()` selon votre sens artitistique et appliquez le sur l'un des tores (ne vous éternisez pas sur cette question et pensez à passer à l'exercice qui suit).

<u>Exercice 3</u> : Elimination faces arrières

Q 1. Enlevez l'activation d'éclairement dans `myInit` (enlevez le `glEnable(GL_LIGHTING)`).

Q 2. Appelez `tracerTetraedre` depuis `myDisplay`
Visualisez. Vous devez voir les faces rouge, verte, bleue et jaune d'un tétraèdre.

Q 3. Activez l'élimination des faces arrières (par `glEnable(GL_CULL_FACE)` dans `myInit()`). Visualisez. **Comprenez** le défaut constaté (disparition de la face bleue).

Q 4. Corrigez pour que toutes les faces soient orientées directes lorqu'elles sont vues de l'extérieur. Vérifiez alors que vous n'avez plus de défauts d'affichages.

Q 5. Supprimez la face rouge du tétraèdre (les 3 glVertex correspondant). Visualisez. Expliquez pourquoi on ne voit pas l'intérieur.

Q 6. Enlevez l'activation du culling et vérifiez qu'on voit bien l'intérieur à présent.
Retenez que l'activation du culling est pertinent uniquement pour les objets fermés (i.e. l'objet est le bord d'un volume) avec des faces correctement orientées.