

OpenGL - TP 1 et 2

Découverte d'OpenGL et modélisation d'un personnage

Dans ce TP, nous allons découvrir et manipuler les fonctions de base d'OpenGL. Nous utiliserons ensuite des primitives de haut-niveau (sphère, cylindre, cube et cône) afin de modéliser un personnage (chacun pouvant ajouter sa "touche artistique"). Ce personnage nous servira dans les TPs suivants. A la fin de ce TP, vous aurez à rendre (par mail à estelle.duveau@inria.fr au plus tard le vendredi 17 octobre 2008 en précisant le binôme créé) un bref compte rendu répondant aux questions ainsi que votre code.

Pour rappel, la documentation d'OpenGL peut se trouver sur le site d'OpenGL.

1 Prise en main et tests

Récupérez le code fourni sur la page web des TPs :

<http://www-evasion.imag.fr/Membres/Estelle.Duveau/SIA.html>,

compilez (`make`) puis exécutez (`tp1`).

Nous allons dans la suite examiner progressivement ce programme et modifier quelques fonctionnalités afin de nous rendre compte de leur impact sur le rendu final.

1.1 Manipulation de la caméra

Le contrôle de la caméra se fait par les flèches du clavier (*left*, *right*, *up*, *down*, *page up*, *page down*). Testez leur comportement.

Ces touches sont considérées comme spéciales par GLUT. C'est pourquoi elles sont gérées par la fonction `glutSpecialFunc(specialKey)` et non par `glutKeyboardFunc(commonKey)` (cf: `main`). Inspectez le code de la fonction `specialKey()` qui gère ces touches spéciales et ses conséquences dans `setCamera()`. Quels sont les degrés de liberté liés à chaque touche?

1.2 Manipulation de l'affichage

Par défaut, la gestion de l'éclairage est désactivée dans ce programme, ce qui nous permettra de mieux comprendre l'influence des différents paramètres. Modifiez les décisions prises dans `initScene()` (couleur de fond, modèle de shading, taille des primitives, ...) et observez à chaque fois le résultat dans les différents modes d'affichage des polygones.

1.3 Manipulation de l'éclairage

Activez la gestion de l'éclairage en décommentant `glEnable(GL_LIGHTING)` dans l'initialisation de la scène. Le contrôle de la position de la lumière se fait par les flèches du clavier (*F1*, *F2*). Testez leur comportement.

Ces touches sont également considérées comme spéciales par GLUT. Inspectez de nouveau le code de la fonction `specialKey()` et ses conséquences dans `setLight()`. Quel est le mouvement de la lumière programmé dans cette fonction?

Modifiez les paramètres `ambient` et `diffus` de la lumière et observez les résultats. Quelle est la différence entre ces deux paramètres?

Remarque : A chaque mouvement de caméra (dans la méthode `setCamera()`), nous repositionnons la lumière car cette dernière est par défaut liée à la caméra alors que nous voulons qu'elle soit liée à la scène.

1.4 Manipulation de la scène

Quelle est la fonction appelée lors du rendu de la scène?

La fonction `drawPrimitives()` se charge de positionner et de dessiner les primitives que nous allons utiliser : cube, sphère, cylindre et cône. Examinez les fonctions qui permettent de créer ces quatre primitives. Quel est, brièvement, le processus de création de chacune des primitives?

Modifiez les valeurs des paramètres de translation dans `drawPrimitives()` et observez les résultats. Que pouvez vous en déduire sur l'enchaînement des transformations?

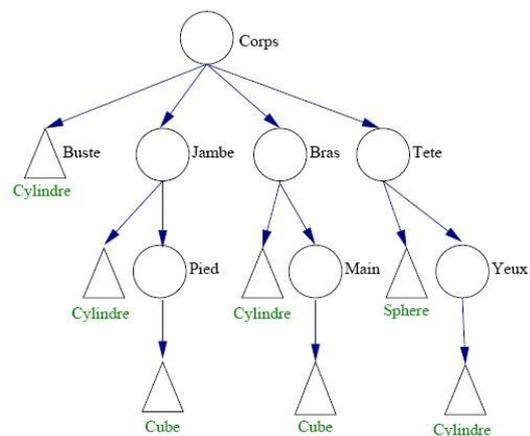
2 Modélisation hiérarchique

Nous allons maintenant assembler ces primitives afin d'obtenir un robot. Nous voulons donc créer (au minimum) son buste, sa tête, ses bras et avant-bras, ses jambes et ses pieds. Vous pouvez bien sûr personnaliser votre création (en ajoutant par exemple, un bras, une queue, ...) mais il est préférable de commencer par un personnage simple quitte à le raffiner ensuite.

2.1 Hiérarchie des repères

La bonne approche pour modéliser un personnage est d'utiliser une **hiérarchie** de repères (donc de transformations en OpenGL). On choisit tout d'abord le positionnement du corps tout entier (que l'on choisit traditionnellement comme étant défini par le placement du buste). On positionne ensuite les bras, les jambes et la tête de manière **relative** au buste. De la sorte, lorsque l'on déplace le corps, tous les membres suivent : ils restent à la même position relative (*cf* : 1.4).

Avant d'implémenter toute hiérarchie en OpenGL, on la représente tout d'abord par un arbre. Dans notre cas, la hiérarchie de base est représentée par l'arbre ci-contre où les disques correspondent aux noeuds de la hiérarchie et où les triangles représentent les formes géométriques dessinées.



2.2 Traduction en OpenGL

A chaque noeud de la hiérarchie, on a besoin de sauvegarder le positionnement courant (`glPushMatrix` et `glPopMatrix()`). Toutes les transformations qui suivent seront relatives à ce positionnement. Afin de positionner un objet, on utilise en général les commandes `glTranslate(...)`, `glScale(...)` et `glRotate(...)`. Remplacez le dessin des primitives (`drawPrimitives()`) par le dessin du personnage (`drawRobot()`) et modélisez le personnage.

3 Modélisation procédurale

Pour finir, nous allons ajouter des détails à notre personnage. Nous allons commencer par lui rajouter des cheveux mais, une fois que vous aurez compris le principe, vous pourrez rajouter d'autres types de détails.

3.1 Création dynamique de chevelure

Nous allons donc raffiner le noeud *tête* de notre hiérarchie. Le but est de pouvoir ajouter des cheveux sur la surface de la tête (c'est-à-dire sur une sphère) de manière **dynamique** lors du rendu. Il nous faut pour cela ajouter une gestion d'évènements et ajouter ou supprimer des cheveux en conséquence.

L'apparence de notre personnage dépend donc d'un paramètre *nombre de cheveux* et une procédure gère la création dynamique de la chevelure à partir de ce paramètre. On parle donc de **modélisation procédurale**.

3.2 Traduction en OpenGL

Un cheveu sera modélisé par une primitive (cône ou cylindre...). Gérez le paramètre *nombre de cheveux* avec l'évènement *touche '+' / '-' enfoncée* grâce à la méthode `commonKey()`. Les primitives-cheveux seront ajoutées ou supprimées en conséquence. Comme précédemment, il faut utiliser les commandes de positionnement et d'empilage/dépilage des matrices de transformation mais cette fois-ci en considérant le paramètre *nombre de cheveux*.

Conseil : Commencez par créer une coupe "punk", c'est-à-dire avec une seule rangée de cheveux, puis ajouter une gestion du nombre de rangées.