TP 1 et 2 d'OpenGL
découverte des méthodes d'affichage et modélisation d'un personnage

Dans ce (double) TP, nous allons découvrir les fonctions de base d'OpenGL, notamment en ce qui concerne l'affichage. Puis nous allons manipuler des primitives de base (sphère, cylindre, cube, et cône) afin de modéliser un personnage (probablement un robot, mais chacun peut apporter sa touche artistique) qui nous servira dans les TPs suivants.

1. Prise en main et tests

Recuperez le code source fourni (http://artis.imag.fr/~Pascal.Barla/opengl/progTP_1_2.tar), compilez (avec make) puis exécutez. Nous allons examiner progressivement ce programme dans la suite et modifier quelques fonctionalites afin de se rendre compte de leur impact sur le rendu final.

Le contrôle de la camera se fait par les flèches du clavier (Page Up/Down incluses). Ces touches sont considerees comme spéciales par GLUT (c'est pourquoi elles sont gerees dans la fonction *specialKey()*). Testez leur comportement, puis inspectez le code. Quel sont les degrés de liberté lies a chaque touche ?

Par défaut, la gestion de l'éclairage est desactivee dans ce programme, ce qui nous permet de mieux comprendre son fonctionnement. Commencez par modifier les décisions prises dans la fonction *initScene()* : couleur de fond, modèle de shading, mode d'affichage des polygones, taille des points et lignes, etc. Observez le résultat. Examinez ensuite les fonctions qui ont permis de créer les primitives.

Decommentez ensuite la ligne *glEnable(GL_LIGHTING);* et testez en manipulant la position de la lumière au moyen des touches F1 et F2. Quel est le mouvement de la lumière, programme dans la fonction *setLight()* ? Modifiez également les paramètres ambient et diffuse de la lumière et observez a nouveau les résultats. Quelle est la différence entre ambient et diffuse ?
Remarque: a chaque mouvement de camera (dans la méthode *setCamera()*), nous repositionnons la lumière car cette dernière est par défaut liée a la camera, alors que l'on veut qu'elle soit liée a la scène.

Enfin, la fonction *drawPrimitives()* se charge de positionner et dessiner les primitives cube, sphère, cylindre et cône. Modifiez les valeurs des paramètres de translation et observez les résultats. Que pouvez vous en déduire sur l'enchaînement des transformations ?

2. Modélisation hiérarchique

Nous allons maintenant nous servir des primitives que nous savons créer (et afficher) et les assembler afin d'obtenir un personnage/robot. On s'intéresse donc a créer son buste, sa tête, bras et avant-bras, jambes et pieds. Vous pouvez bien sur personnaliser votre création (comme ajouter des bras...), même s'il est mieux de commencer par un personnage simple.

La bonne approche pour modéliser un personnage est d'utiliser une **hiérarchie** de transformations. On commence par positionner le corps tout entier (qui correspond traditionnellement au placement du buste), puis on peut ensuite positionner les bras, les jambes et la tête de manière relative au buste. De la sorte, lorsque l'on décide de déplacer le corps, tous les membres suivent (ils restent a la même position relative). On applique la même approche hiérarchique pour les bras, les jambes, la tête, etc.

Afin d'implémenter “intuitivement” cette hiérarchie en OpenGL, on la représente d'abord par l'arbre suivant (sur un personnage simple dont vous pouvez prendre inspiration):

Corps
|_ Buste (Cylindre)
|_ Jambe Gauche/Droite
  |_ Jambe (Cube)
  |_ Pied (Cube)
|_ Bras Gauche/Droit
  |_ Bras (Cube)
  |_ Avant-bras (Cube)
|_ Tête
  |_ Visage (Sphère)
  |_ Yeux (Cylindres)

A chaque noeud de la hiérarchie (symbole |_), on a besoin de sauvegarder le positionnement courant. Toutes les transformations qui suivent seront ainsi relatives a ce positionnement. En OpenGL, on utilise les commandes *glPushMatrix()* et *glPopMatrix()*, qui respectivement empilent et dépilent la matrice de positionnement courante. Afin de positionner un objet, on utilise les commandes *glTranslate(), glScale()* et *glRotate()* (utilisez la commande *man* pour savoir comment les utiliser). Les feuilles de l'arbre correspondent aux appels de primitives. Il ne vous reste plus qu'a créer votre personnage a vous !

3. Modélisation procedurale

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

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 a dire sur une sphère) de manière dynamique lors du rendu. Il nous faut pour cela ajouter une gestion d'evenements (par exemple gérer l'évènement “touche '+' / '-' enfoncée” avec la fonction *commonKey()*), et ajouter ou supprimer en conséquence des primitives de type cheveux (cône ou cylindre). L'apparence de notre personnage dépend donc d'un paramètre “nombre de cheveux”, et une procédure gère la création dynamique de la “chevelure” a partir de ce paramètre ; on parle donc de modélisation procedurale (simple).

Pour mettre en oeuvre ce mécanisme, on utilise comme précédemment 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 a dire avec une seule rangée de cheveux, puis ajouter une gestion du nombre de rangées.