UJF - M1 Informatique 2005-2006 : Infographie
TP5 : Modèle de caméra - Perspective

4 novembre 2005

Aujourd'hui nous allons étudier le modèle de caméra perspective en openGL.

1 Définition de la caméra perspective

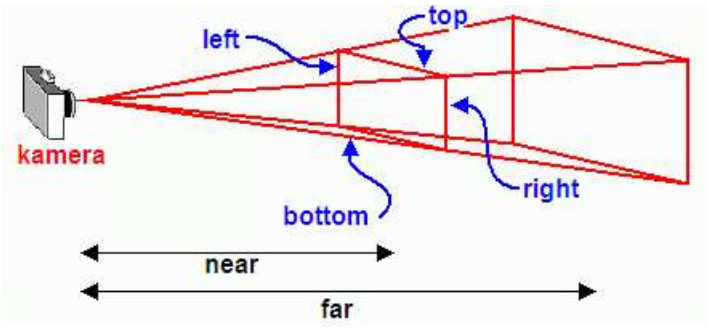
Pour l'instant nous ne nous intéresserons qu'aux parties du programme définies par "`type == CUBE`".

La procédure display du le fichier `main.cpp` est appellée à chaque fois que le fenêtre openGL doit être redessinée, c'est-à-dire en permanence dans notre cas (cf. `glutIdleFunc` dans la procédure `main`).

Après avoir effacé le contenu de la fenêtre précédente, elle appelle la fonction definirCamera(w/h) et placerCamera :

- `definirCamera( aspect )` : définit la caméra perspective en se plaçant dans la **matrice de projection** puis en donnant les paramètres de la caméra : les 6 plans du **frustum**. Puis elle repasse dans la matrice `MODEL_VIEW`.
- `placerCamera( z )` : définit simplement la position initiale de la caméra, en la déplaçant sur les Z.

Voici un schéma du frustum en openGL :



Le paramètre left de la procédure `glFrustum`, par exemple, définit la position en X dans le repère de la caméra du plan gauche du frustum (à une distance `znear` du centre optique).

Il faut voir le plan near comme le plan image, c'est-à-dire la pellicule dans un appareil photo.

1.1 Aspect

Le programme dessine en fils de fer un cube centré à l'origine (après placement de la caméra en arrière) et de côté 2 (donc qui va de (-1,-1,-1) à (1,1,1)).

1. Redimensionnez la fenêtre, qu'observez vous et pourquoi ?
2. Lorsque vous redimensionnez la fenêtre, c'est la procédure reshape du `main.cpp` qui est appellée. Sachant que le paramètre `aspect` de la procédure `definirCamera` représente le rapport entre la largeur et la hauteur de la fenêtre, comment remédier à ce "problème" ? Quels paramètres pour `glFrustum` faut-il changer ?

1.2 Clipping et perspective

Voici l'utilisation des touches du clavier (cf. procédure `keyboard` appelée après chaque appui de touche) :

- z et Z : réduit et augmente la variable `zCamera`, la position dans l'espace du centre optique de la caméra selon l'axe Z (cf. placerCamera)
- n et N : réduit et augmente la variable `znear`, la position du plan near
- f et F : réduit et augmente la variable `zfar`, la position du plan far

Rappelez vous que ce que vous observez dans votre fenêtre openGL est le "contenu" de la projection de la scène sur le plan near coupé par les plans gauche, droite, bas et haut.

1. Rapprochez le plan far, qu'observez vous, pourquoi et comment s'appelle ce phénomène ?
2. Eloignez le plan near jusq'à la valeur 2 (cf. sortie du programme sur le terminal), où se trouve t-il dans le repère monde ?
3. Rapprochez le plan near au plus près et rapprochez le centre optique de la caméra. Observez vous le même cube qu'initialement et pourquoi (faites le schéma de la projection dans ce cas et le cas initial) ?

2 Visualisation du frustum

Nous désirons maintenant visualiser la caméra de la première partie (que nous nomerons virtuelle) depuis un autre point de vue (en fait une autre caméra) sous une forme similaire au schéma du frustum présenté plus haut. Nous allons utliser le programme avec le second type : `NAVIGATION` ("`-t 1`" ou la touche 't' pour changer de type en cours d'éxécution). Ce type éxécute le programme comme nous l'avons toujours fait dans les TP précédents, c'est-à-dire avec la possibilité de tourner autours de la scène avec la souris.

Le but de cette partie est de réussir à placer correctement et de la manière la plus simple la caméra virtuelle dans l'espace.

1. Il est facile de repérer le centre optique de la caméra virtuelle dans l'espace, ce n'est qu'un point. Quelles sont ses coordonnées ? Pour dessiner un point,

vous pouvez utiliser le rendu `GL_POINTS` (n'oubliez pas le S). Pour changer le taille du point : **avant** `glBegin` utilisez la procédure `glPointSize(int)`.

2. Dessinez le reste du frustum en fonction des autres paramètres de la caméra virtuelle (znear et zfar).
3. Utilisez les touches (qui modifient toujours ces paramètres) pour modifier la caméra virtuelle et vérifiez en repassant dans le point de vue de cette dernière (en changeant de "type") que vous obtenez bien le résultat escompté (largeur de la boîte, profondeur, distance focale, etc.).

2.1 Caméra totalement manipulable et caméra orthographique

Ajoutez quelques objets (teapot, sphere) à différents endroits.

1. Essayez la projection orthographique en décommentant les lignes contenant "`glOrtho`" dans la procédure `definirCamera`. Qu'observez vous dans les 2 types différents ? Et pourquoi le zoom (bouton du milieu) ne "fonctionne t-il plus" en type navigation ?
2. Otez le mode projection orthographique. Ajoutez d'autres variables pour mieux manipuler la caméra virtuelle (bouger en X et en Y, rotations), vérifiez la correction de votre méthode par la visualisation de la caméra virtuelle.