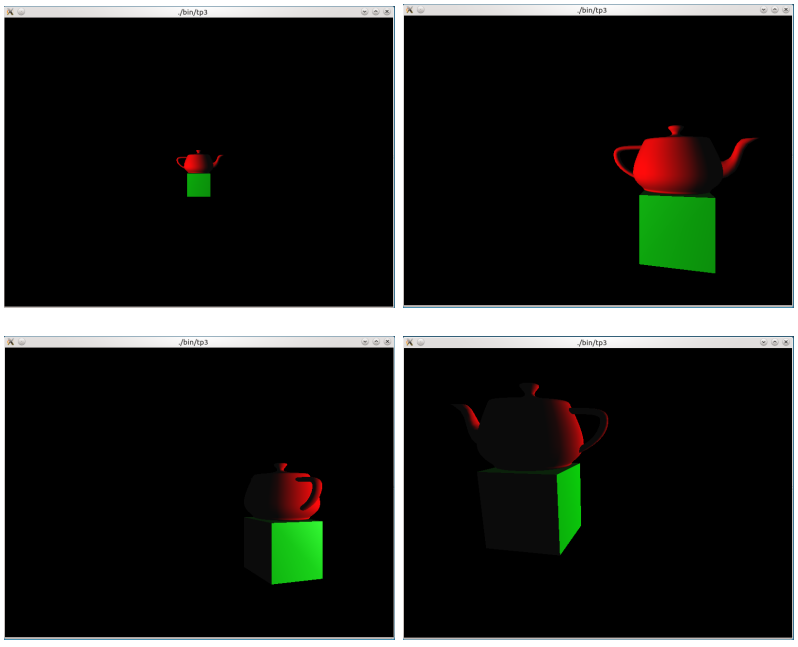
TPs L2 Informatique AII3D : découverte de la synthèse d'images

Placement de la caméra en OpenGL

Ce TP a pour but de comprendre comment placer (et déplacer) la caméra en OpenGL. Nous allons créer une navigation de type *walk*, c'est à dire que l'on va pouvoir se déplacer dans la scène comme un utilisateur qui peut marcher dans le vide.

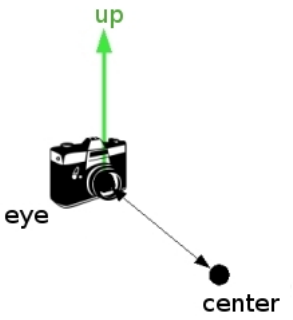


Rappels

- Pour pouvoir déplacer la caméra et placer le point de vue n'importe où dans la scène, nous utilisons une sorte de caméra virtuelle avec l'appel de la fonction :

  gluLookAt(eyeX,eyeY,eyeZ,centerX,centerY,centerZ,upX,upY,upZ)

  eyeX, *eyeY* et *eyeZ* définissent la position de la caméra.
  centerX, *centerY* et *centerZ* définissent la position du point que fixe la caméra (le point correspondant se trouvera au centre de la fenêtre d'affichage).
  upX, *upY* et *upZ* définissent le vecteur vertical.

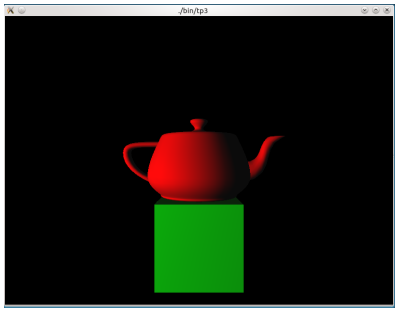


  Placer la caméra revient à déplacer tout le monde pour qu'il soit centré sur la caméra et orienté selon l'axe du regard. Cela influe donc logiquement sur la matrice *GL_MODELVIEW* et vous ne serez pas étonnés donc que l'appel de la caméra soit juste après la réinitialisation de *GL_MODELVIEW*.

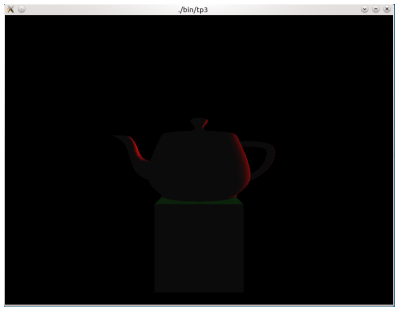
  (Pour plus de détails, vous pouvez consulter la doc d'openGL).

- **Exemples :**

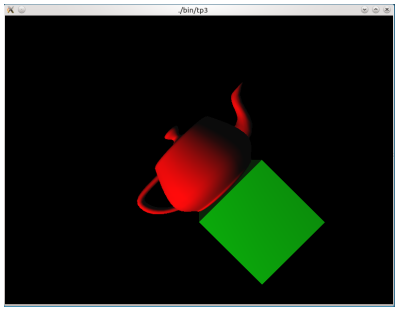
  Dans notre scène initiale, la théirère est placée en *(0.,0.,-5)* grâce une translation :



L'appel à la fonction *gluLookAt(0., 0., -10., 0., 0., 0., 0., 1., 0.);* permet de voir l'arrière de la théière :



Changer la verticale (les paramètres *vertX*, *vertY* et *vertZ*) et donc appeler la fonction *gluLookAt(0., 0., 0., 0., 0., -5., 1., 1., 0.);* permet de pencher la scène :



Introduction

- Commencez par réccupérer les fichiers *camera.h* et *camera.c*. Si vous ne l'avez pas fait lors du TP précédent, réccupérez également les fichiers *vector4.h* et *vector4.c*. N'oubliez pas d'ajouter tous les *include* nécessaires, et de faire les modifications appropriées dans votre *Makefile*.

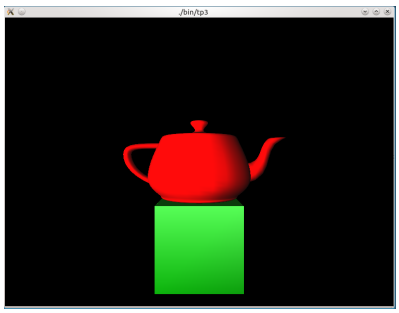
- Le type *tCamera* (avec le type du pointeur associé *pCamera*) est une structure ayant pour champs :
  - le centre du repère caméra (= position de la caméra) : *Vector4 O*,
  - les 3 vecteurs du repère caméra : *Vector4 X*, *Vector4 Y*, *Vector4 Z*

- La fonction *pCamera creerCamera(Vector4 eye, Vector4 vise, Vector4 up)* prend en paramètres les 3 paramètres du *gluLookat(...)* et calcule le repère caméra correspondant. Créez un objet *pCamera* dans votre programme principal à partir des paramètres suivants : *eye = 0.,0.,0.,1.*, *vise = 0.,0.,-1.,1.* et *up = 0.,1.,0.5,0.*.

- Les fonctions *void eyeCamera (pCamera c, Vector4 eye)*, *void viseCamera (pCamera c, Vector4 vise)*, *void upCamera (pCamera c, Vector4 up)* calculent les paramètres du *gluLookat(...)* à partir des attributs O,X,Y,Z de la caméra.

- A ce niveau, il est intéressant de noter le comportement d'une source de lumière par rapport à la caméra. Une source de lumière positionnée après le *gluLookAt()* ne tiendra pas compte de la position de la caméra. Sa position est définie par rapport au repère scène. Par contre, une source de lumière dont la position est définie juste avant le *gluLookAt()* est positionnée par rapport au repère de la caméra (sa position est alors relative à celle de la caméra). Positionnez correctement la lumière *GL_LIGHT0* pour qu'elle soit fixe dans la scène (elle ne doit pas bouger si on déplace la caméra).

  Utilisez la touche 1 pour allumer/éteindre la lumière *GL_LIGHT1*. *GL_LIGHT1* doit être un spot de lumière blanche, avec un angle d'ouverture de 40 degrés et un coefficient d'atténuation de 20. Ce spot positionné à la position de la caméra et éclairant droit devant, doit se déplacer avec la caméra. C'est un peu comme une lampe de poche attachée à la caméra pour éclairer ce que l'on a devant nous dans la scène.



Gestion des évènements clavier

Il va falloir développer les fonctionnalités suivantes :

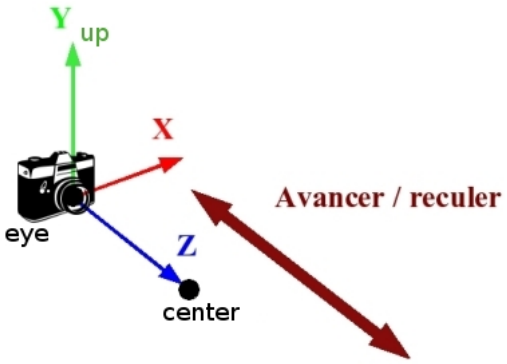
- *avancer/reculer* : la caméra se déplace vers l'avant ou l'arrière dans sa direction Z.
- *tourner sur la gauche/droite* : par rotation autour de l'axe verical Y de la caméra.
- *s'orienter vers le haut/bas* : par rotation autour de l'axe horizontal X de la caméra.

Nous allons utiliser les flèches du clavier pour avancer/reculer et gauche/droite, ainsi que les touches h et b pour haut/bas.

- Pour accéder aux flèches du clavier, ajoutez la fonction *glutSpecialFunc (arrow)* (par exemple juste sous la fonction *glutKeyboardFunc (key)* du *main* de votre programme).
- Écrire la fonction *void arrow (int key, int mousex, int mousey)* sur le même modèle que la fonction *void key (void)*. Le test *switch* est effectué sur la variable *key* et les flèches gauche, droite, bas, haut sont prises en compte avec respectivement *case GLUT_KEY_LEFT*, *case GLUT_KEY_RIGHT*, *case GLUT_KEY_DOWN*, *case GLUT_KEY_UP*.

Avancer/Reculer

Commencez par faire le déplacement avant/arrière de la camera à l'aide de la fonction void *avanceCamera (pCamera c, float pas)*. Attention, le déplacement s'effectue dans la direction du vecteur Z de la caméra. Pour vous aider, vous avez à disposition les fonctions du fichier *Vector4.h* pour modifier les valeurs du O,X, Y et Z de la caméra.

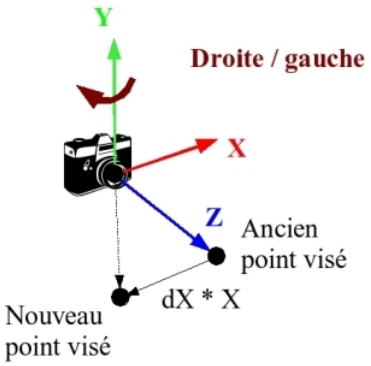


Rotation gauche/droite, Rotation haut/bas

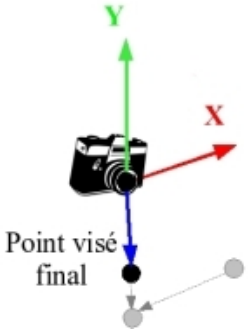
Regardons maintenant les rotations :

La rotation autour de l'axe Y va être effectuée en suivant la méthode suivante :

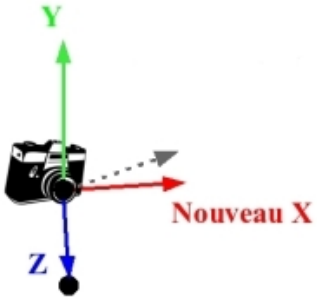
- Déplacez le point visé (qui vaut toujours $O+Z$) dans la direction du vecteur X. La taille du déplacement correspond à un pas dX pour une pression sur une touche gauche ou droite (le point visé est déplacé de $dX * X$).



- Calculez le nouveau vecteur unitaire Z correspondant à la nouvelle position du point visé.



- Calculez le nouveau vecteur X.



Sur le même principe, effectuez la rotation haut/bas.