HAI719I - Programmation 3D
TP3 Transformations

Noura Faraj *noura.faraj@umontpellier.fr*

Objectif

Le but de ce TP est de se familiariser avec les transformations 3D représentées sous forme matricielle.

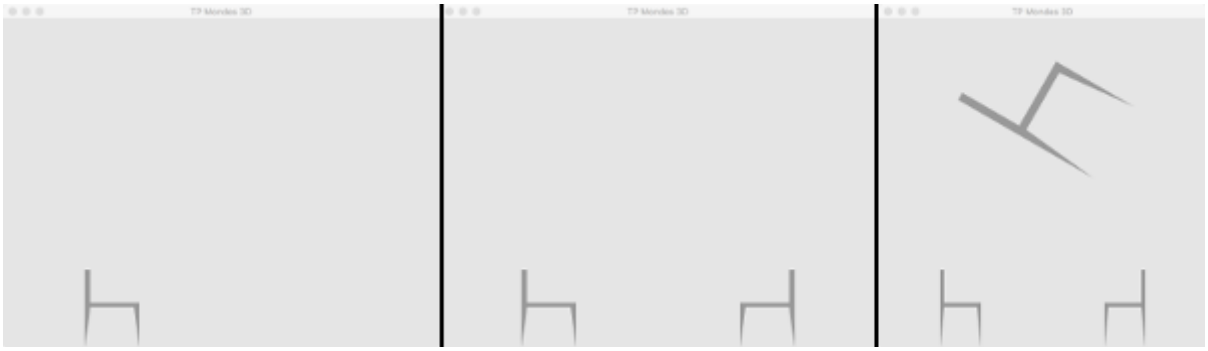
Décompresser le fichier, dans le dossier résultant :
mkdir build
cd build
cmake ..
make -j
./launch-TP4_transformations.sh

Question 1

1. Ajoutez une variable uniforme de type mat4 au vertex shader représentant la matrice de transformation à appliquer aux sommets. Construire une matrice identité.

2. Faites-en sorte que les effets de zoom et de translation du TP précédent (avec contrôle au clavier) fonctionnent à nouveaux via cette matrice de transformation.

3. Pour cette petite série d'exercices, modifiez les valeurs de la matrice à la main (dans le code C++, commentez le contrôle au clavier).



   a. Réduisez la dimension de la chaise par 2 et la poser sur le sol et légèrement à gauche.
   b. Rajoutez une chaise en face.
   c. Rajoutez une troisième chaise dont la seule transformation sera une rotation contrôlée au clavier.
   d. Faites-en sorte que la chaise tourne autour de son centre de gravité en (0,0.5,0). Vous pourrez construire plusieurs transformations que vous combinerez dans le bon ordre.

4. Changer le model chargé par suzanne.off

   a. Appliquez-lui une rotation 3D d'un angle contrôlé au clavier, et autour d'un axe quelconque via `glm::rotate(matrice, angle_in_degrees,rotation_axis)`.
   b. Calculez et appliquez une rotation de telle sorte que l'axe vertical du personnage (0,1,0) soit aligné avec le vecteur (1,1,1) du repère monde.

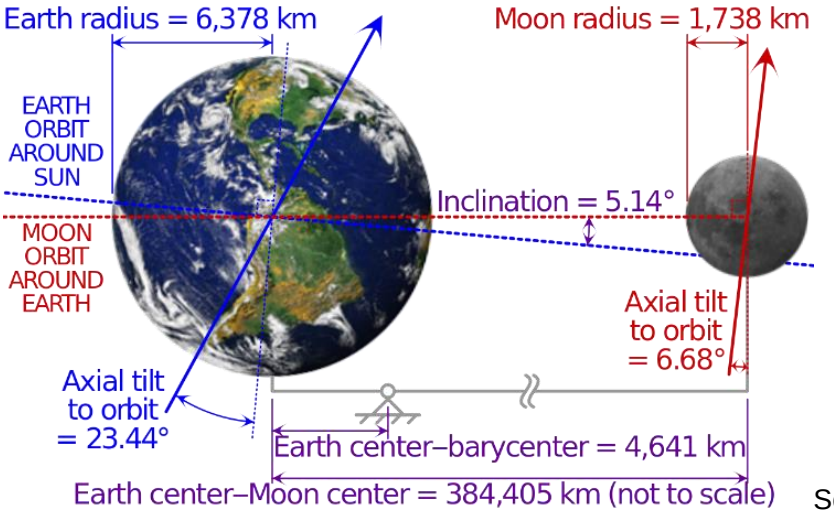
Question 3

1. Pour passer d'un espace 2D à une "vraie" vue 3D, il nous faut tout d'abord définir une **matrice de projection** et l'exploiter dans nos shaders. Pour cela, créer une matrice Vous pouvez contrôler la pyramide de vision avec la méthode `glm::perspective(...)`.
2. La seconde étape consiste à contrôler la position et orientation de la caméra via une **matrice de vue**. Pour cela implémentez la méthode `glm::lookAt(...)` dont le but est de définir la matrice `ViewMatrix` de transformation du repère monde au repère de la caméra à partir d'une position d'observation `position`, une cible à viser `target` et une orientation verticale `up` (utiliser les variables `camera_position, camera_target, camera_up`). Pensez à mettre à jour vos shaders pour exploiter cette matrice supplémentaire (vos shaders doivent maintenant prendre en compte 3 matrices : la matrice *objet*, la matrice de *vue*, et la matrice de *perspective*). Testez en créant une vue de 3/4.
3. Vous pouvez zommer et dé-zoomer en utilisant les touches Z et S. Inspirez-vous de ce code pour ajouter les déplacements latéraux.

Question 4

Mettre en œuvre un mini système solaire animé. Il inclura au moins la terre, qui devra tourner sur son axe et être en révolution autour du soleil. Vous ajouterez ensuite notre lune en révolution autour de la terre.

1. Commencez par un système solaire minimaliste avec une sphère (sphere.off) au centre de la scène représentant le soleil et une seconde sphère, plus petite, tournant autour du soleil à une distance fixe. *Conseil : afin de mieux visualiser les rotations, utilisez le mode de rendu en fils de fer* `glPolygonMode(GL_FRONT_AND_BACK, GL_LINE).`
2. Faites en sorte que la terre tourne sur elle-même autour d'un second axe. Modifiez l'axe de la terre afin que son axe de rotation soit incliné (d'environ 23 degrés) comme c'est le cas en réalité. Rappel : vous pouvez utiliser la classe `glm::rotate` pour construire une rotation autour d'un axe arbitraire.
3. Sur le même principe, ajoutez une lune tournant autour de la terre.



Source : Wikipedia

Vous pouvez continuer en ajoutant d'autres planètes et leurs lunes pour former le système solaire complet.