TP2 - Hiver 2018

IMN 428

Infographie

Objectifs

1. Utiliser quelques fonctions de base d'OpenGL
2. Utiliser les fonctions d'OpenGL liées aux transformations affines 3D.
3. Afficher des objets 3D
4. Se familiariser avec les shaders

Description

À l'aide du code fournit (fichiers tp2.cpp, shader.h, shader.cpp, shader.frag, shader.vert) vous devez implanter les transformations affines suivantes : la translation, la rotation, le changement d'échelle et le cisaillement. Les transformations seront appliquées à un cube, un cône et un système d'axes. Les endroits où vous devez ajouter du code sont clairement indiqués par l'étiquette «AJOUTER DU CODE ICI !». Vous devez également fournir le code pour dessiner les trois axes du système. Chaque axe doit être une ligne **d'épaisseur 4 et de longueur 20**, munie d'une pointe (flèche). Cette pointe est un tétraèdre et doit être affichée à l'aide de la fonction *DrawTetrahedron()* fournie avec le code. L'axe X doit être **rouge**, l'axe Y **vert** et l'axe Z **bleu**. Il est à noter que la caméra est centrée à l'origine du monde (0,0,0) et que le système d'axes est situé à l'origine de l'objet sélectionné (cône ou cube).

Les fonctions initCone(), initCube(), initTetrahedron() et initLineAxis() ont pour but de créer les vertex et les polygones associés à ces objets et de les copier sur le GPU. Les fonctions drawCone(), drawCube(), drawTetrahedron() et drawLineAxis() ont pour but de copier sur le GPU les matrices model, view et projection ainsi que la couleur associées à chacun de ces objets et à lancer l'opération de rendu à l'aide de la commande "glDrawArrays".

Description de l'interface

L'interface actuelle est très simple. Le bouton de droite permet de sélectionner le mode de transformation. Une fois le mode sélectionné, la transformation peut être appliquée en enfonçant le bouton de gauche et en déplaçant la souris verticalement (avant-arrière). Le terme "Rotation X" signifie une rotation autour de l'axe X.

Vous devez implanter les transformations suivantes : translation selon l'axe X, Y ou Z, rotation autour de l'axe X, Y ou Z, changement d'échelle en X, Y ou Z, changement d'échelle uniforme (même facteur pour X, Y et Z) ainsi que le cisaillement suivant l'axe X,Y ou Z. Vous devez également ajouter la possibilité de faire bouger indépendamment le cône et le cube (*Move Cube Vs Move Cone*).

Pour implanter les transformations, vous utiliserez les fonctions suivantes de la bibliothèque glm :

. glm : :rotate

. glm : :translate

. glm : :scale.

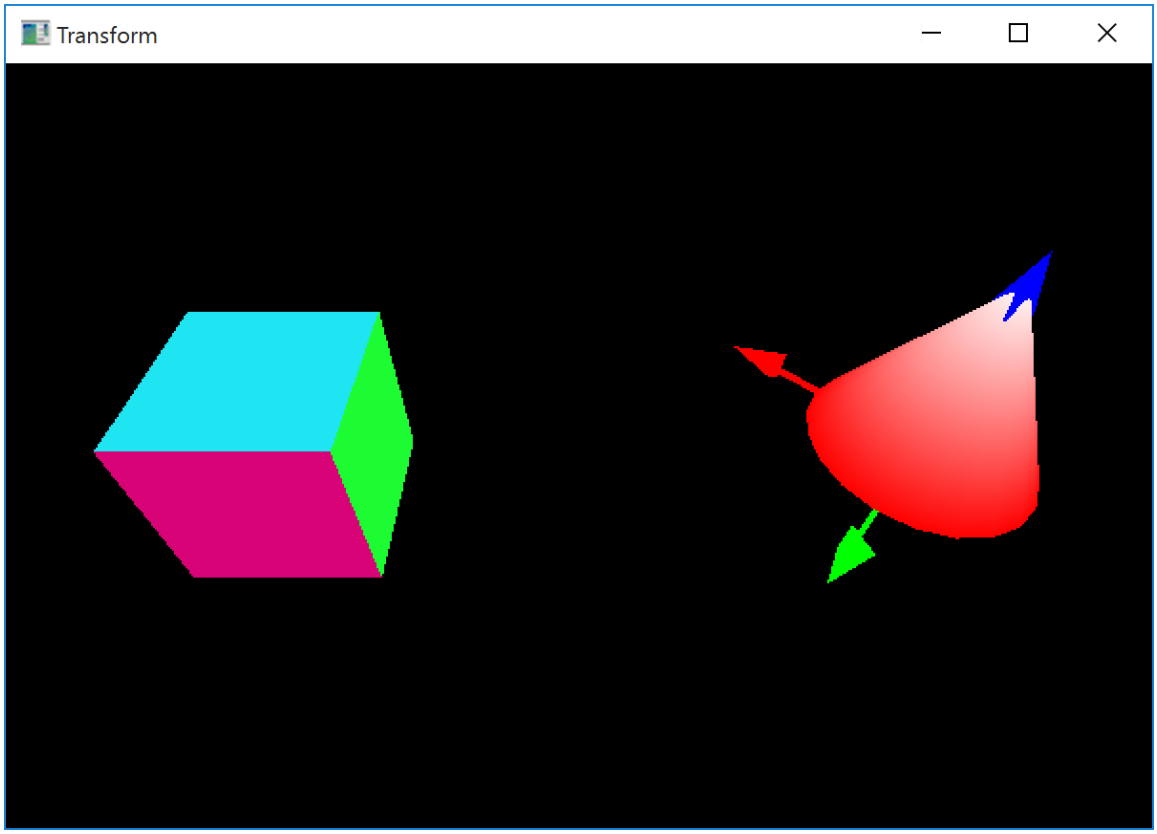

FIGURE 1 – Capture d'écran du tp2 avec bonus

Pour le cisaillement, vous devez vous-même faire la matrice de transformation et la multiplier à la matrice model de l'objet sélectionné.

À noter que la position et l'orientation des deux objets sont respectivement contenues dans les matrices "model" 4x4 globales **cubeModelMatrix** et **coneModelMatrix**. Ces matrices contiennent l'ensemble des transformations ayant été appliquées aux deux objets. La matrice model **axisModelMatrix** est également utilisée pour positionner les axes et les pointes de flèches.

N'oubliez pas d'inscrire vos noms en commentaire au début du fichier .cpp remis !

BONUS 5%

Vous devez rédiger le code C++ ainsi que GLSL (fragment et/ou vertex shader) vous permettant d'afficher le cube avec des faces de couleurs aléatoires ainsi que le cone avec une pointe blanche et une base rouge avec un dégradé entre les deux (voir figure 1).

Évaluation

Ce travail doit être fait par **équipe de TROIS**. Au moment de soumettre votre travail, assurez-vous que votre code compile bien sous *Microsoft Visual Studio 2015*. Utilisez le système **turnin web** pour soumettre votre travail.