IMN501 **Animation et rendu temps réel**

Travail pratique #3

But:

Se familiariser avec le nuanceur de vertex et le nuanceur de fragment.

Directives:

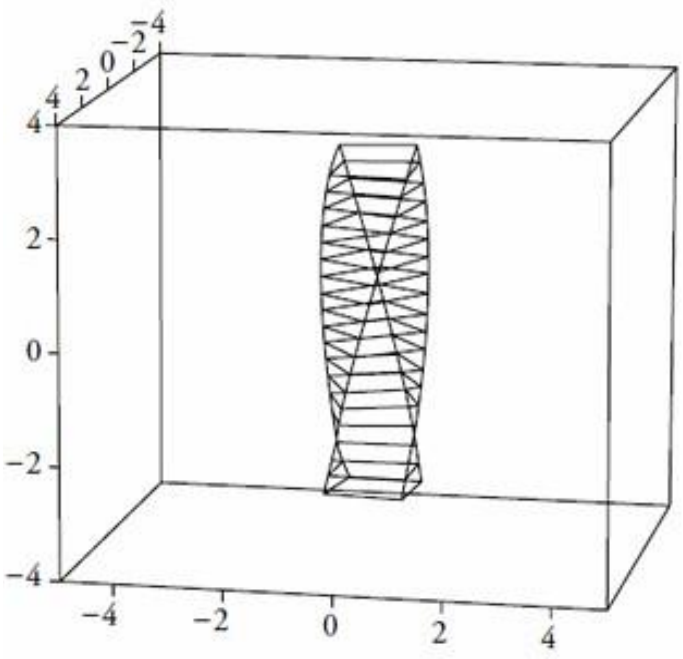
Le tp3 consiste à modifier un tutoriel de opengl-tutorial, http://www.opengl-tutorial.org/ . Il y une version anglaise et une version française pour les explications du tutoriel, malheureusement la version française n'est pas toujours à jour. Donc référez-vous à la version anglaise en premier. La version française pourra cependant vous aider à la compréhension si vous n'êtes pas très à l'aise en anglais.

Commencez par lire attentivement le tutoriel 1 pour avoir les détails de l'installation. Choisissez la plate-forme que vous voulez mais vous devriez quand même vérifier que tout fonctionne bien sous Windows avec Visual Studio 2017. Pour l'installation, vous pouvez aussi utiliser le CMake inclut dans Visual Studio 2017. Par la suite, faites les tutoriels que vous voulez mais dans ce travail, vous allez modifier le tutoriel 8.

Pour ce tp, il n'y a évidemment pas de section « insérer le code ici » et cela fera partie du travail de figurer les endroits où insérer le code.

Nuanceur de vertex :

Vous devrez déformer le maillage avec une des méthodes que l'on a vue en classe : transformation géométrique affine qui varie selon la position dans l'espace, soit de tordre la scène autour d'un axe, ici Z :



k = twist factor

$$x' = x \cdot \cos(k \cdot z) - y \cdot \sin(k \cdot z)$$
$$y' = x \cdot \sin(k \cdot z) + y \cdot \cos(k \cdot z)$$
$$z' = z$$

Par contre, la déformation se fera autour de l'axe Y. Référez-vous aux matrices de rotations 3D autour d'un axe canonique pour figurer la nouvelle formule.

Pour avoir une illumination correcte, n'oubliez-pas de transformer aussi les normales à la surface !

Le k = twist factor, ne sera pas statique, il variera dans le temps. Vous devrez le passer en paramètre comme « uniform float » au nuanceur de vertex et il sera calculer sur CPU, i.e. dans le programme C++ (tutorial08.cpp). Nommez la variable « twist » dans le programme C++.

Vous pourriez utiliser un compteur de nombre de fois que vous afficher la scène pour simuler le temps écoulé. Cette façon de faire n'est pas standard car l'animation serait plus ou moins rapide selon la vitesse de la machine. Pour procéder de la façon correcte, vous devez utiliser le temps réel écoulé. La fonction de GLFW double glfwGetTime(void) retourne le temps écoulé en secondes depuis que GLFW a été initialisé. Le twist, entre chaque affichage, aura la valeur de :

```
twist = 2.0f * sin(0.5f * (float) glfwGetTime());
```

Nuanceur de fragment :

Au lieu d'utiliser la texture bitmap, vous aller procéder avec une texture procédurale 2D de type damier. Voici la formule pour le noir et blanc (0 noir et 1 blanc).

```
mod(floor(100.0 * U)+ floor(100.0 * V), 2.0)
```

Où U et V sont les coordonnées de texture 2D. Vous devez l'adapter pour avoir des cases jaunes et bleues.

Remise pour **dimanche le 22 mars** avant minuit. Remettez les fichiers modifiés par turnin web. Les fichiers en question : **tutorial08.cpp**, **StandardShading.fragmentshader** et **StandardShading.vertexshader.** N'oubliez pas d'inscrire **vos noms** au début du fichier tutorial08.cpp.

Bon travail !