
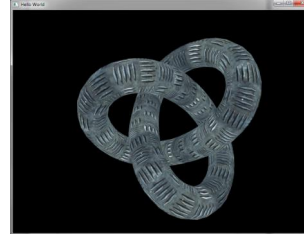
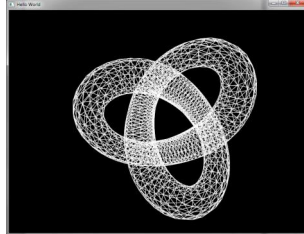


TP02 – Système de coordonnées et textures

 Nous allons afficher dans ce TP un nœud en 3D, dans un premier temps, sans texture en fil de fer, puis avec une texture plaquée.



Exercice 0 Préparation de votre environnement de travail

0.1 Mise en place de la solution


 Créez une solution Visual Studio comme lors des précédents TP.

Nous utiliserons des textures dans ce TP, nous avons donc besoin de charger des images. Le chargement s'effectuera à l'aide de la bibliothèque FreeImage disponible dans le fichier Ressources.zip. Par ailleurs, nos objets sont affichés en 3D, nous aurons donc besoin de réaliser des changements de repère à travers des calculs matriciels. Nous utiliserons pour cela la bibliothèque GLM disponible également dans le fichier Ressources.zip.


 Ajoutez les dépendances suivantes à votre projet Visual Studio

 Répertoires Include


- ➔ \$(SolutionDir)..\..\Ressources\FreeImage\Wrapper\FreeImagePlus\dist\x64
- ➔ \$(SolutionDir)..\..\Ressources\FreeImage\Dist\x64
- ➔ \$(SolutionDir)..\..\Ressources\glm

 Répertoires de bibliothèques

- ➔ \$(SolutionDir)..\..\Ressources\FreeImage\Wrapper\FreeImagePlus\dist\x64

 Entrée de l'éditeur de liens

- ➔ FreeImagePlus.lib

 Pour s'exécuter, votre application aura alors besoin des fichiers FreeImage.dll et FreeImagePlus.dll que vous trouverez dans le fichier Ressources.zip et que vous copierez dans le répertoire Debug de sortie de votre compilation.






Exercice 1 Chargement et affichage du nœud

La forme du nœud est relativement complexe. Heureusement, les positions et coordonnées de textures des vertex ont préalablement été calculées pour vous et stockées dans un fichier au format texte : knot.mesh. Mieux encore, une classe de chargement de ce type de fichier a été développée

pour vous par votre meilleur professeur. Il s'agit de la classe CMeshLoader définie par les fichiers MeshLoader.h et MeshLoader.cpp.

1.1 Utilisation de la classe CMeshLoader



La classe CMeshLoader, que vous pouvez bien entendu visiter, dispose des membres suivants :

-  `bool loadFromFile(const std::string& strFileName)`
 - ➡ Charge le maillage depuis le chemin de fichier passé en paramètre
-  `const GLfloat* getVBO() const`
-  `size_t getVBOSize() const`
 - ➡ Retourne respectivement un pointeur et la taille du Vertex Buffer Object associé au maillage chargé.
-  `const GLfloat* getEBO() const`
-  `size_t getEBOSize() const`
 - ➡ Retourne respectivement un pointeur et la taille de l'Element Buffer Object associé au maillage chargé.









Pour la spécification des attributs de vertex, la classe recueille 4 types d'attributs : la position, la normale, la couleur et les coordonnées de texture. La taille et la position de ces attributs est obtenue respectivement par les méthodes `getNb*Components()`, `get*Offset()` et `getStride()`. Ces valeurs sont exprimées en nombre de flottants, les valeurs « d'offset » et de « stride » doivent donc être multipliées par la taille d'un flottant (`sizeof(GLfloat)`) pour permettre la spécification des attributs de vertex pour le VAO.

Pour l'affichage des triangles, le type de construction des primitives OpenGL est retourné par la méthode `getPrimitivesType()`. Typiquement pour le nœud, cette méthode retourne la valeur `GL_TRIANGLES`.

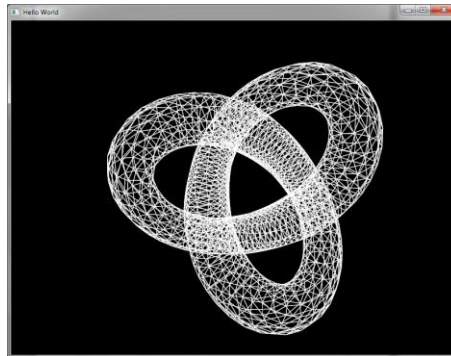
Enfin, des fonctions d'informations permettent de recueillir la boîte englobante du maillage chargé. Il s'agit de la fonction `getBoundingBox(...)` et des fonctions `get{X|Y|Z}{min|max}`.

-  En utilisant cette classe, chargez le maillage du nœud depuis le fichier `knot.mesh`.
-  Spécifiez l'organisation des attributs de vertex dans le VAO.
 - ➡ L'attribut qui nous intéresse pour le moment est uniquement la position des vertex.





1.2 Organisation de la scène

-  En utilisant la bibliothèque GLM, spécifiez la scène suivante comme vu en cours :
 -  Nœud en position de modélisation (position inchangée)
 -  Caméra en position $(0, 0, \text{taille})$ regardant le centre du nœud. La valeur *taille* correspond à la diagonale de la boîte englobante du nœud.
 -  Projection perspective d'angle de champs de vu égal à 45°
 - ➡ Cette projection doit se faire en respectant le ratio d'aspect de la fenêtre, même si la dimension de la fenêtre est modifiée.
-  Implémentez le vertex shader en conséquence
 -  Pensez au transfert des matrices en tant qu'uniform
-  Implémentez le fragment shader de façon à afficher simplement une couleur blanche pour tous les fragments.
-  Affichez le résultat en fil de fer

Vous devriez obtenir la capture suivante :










1.3 Animation et interactions

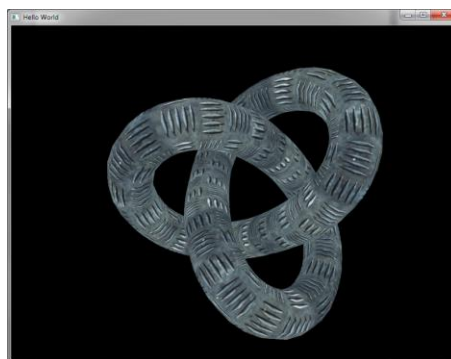
-  Faites en sorte que la molette de la souris impacte l'angle de champs de vision
-  Faites en sorte que le nœud tourne sur lui-même
 -  C'est la matrice de modélisation qui doit être modifiée par la rotation, puisque c'est le nœud qui tourne (ce n'est pas la caméra)
-  Faites en sorte que l'appuie sur la touche 'w' active / désactive le mode fil de fer

Exercice 2 Texturage du nœud


2.1 Plaquage de la texture

La classe CMeshLoader vous a permis de retrouver des coordonnées de texture pour chaque vertex.

-  Spécifiez l'organisation des attributs de vertex dans le VAO.
 -  Les attributs qui nous intéressent maintenant sont la position et les coordonnées de texture des vertex.
-  Chargez la texture « MtlPlat2.jpg » à l'aide de la bibliothèque FreeImage.
 -  Transférez-là vers le GPU
 -  Activez le filtrage linéaire pour les mipmaps
-  Modifiez le fragment shader en conséquence pour afficher la texture.
-  Exécutez votre programme, vous devriez obtenir la capture suivante :



2.2 Interactions

-  Faites en sorte que l'appuie sur la touche 't' active / désactive le plaquage de la texture. Lorsque la texture est désactivée, la couleur des fragments doit être blanche.