POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

INF2705 Infographie

Travail pratique 2

Détails et de réalisme des scènes graphiques

Département de génie informatique et génie logiciel
Polytechnique Montréal
Tristan Rioux, Automne 2025

Table des matières

1 Description globale

1.1 But

Le but de TP est de permettre à l'étudiant de mettre en pratique la matière reliée au texture, au test de stencil et à l'illumination. Il sera en mesure d'utiliser des textures afin d'ajouter des détails aux objets. Il sera aussi capable de configurer et d'utiliser le test de stencil. Pour finir, l'ajout de calculs d'illumination permettra d'ajouter plus de réalisme à l'ensemble de la scène. Le travail fera l'utilisation des fonctions d'OpenGL `glGenTextures`, `glBindTexture`, `glTexImage2D` et `glTexParameteri`, puis `glStencilOp`, `glStencilFunc`, `glStencilMask`.

1.2 Travail demandé

Le tp2 est une continuation du tp1. Il faudra reprendre à partir de votre ancien code ou utiliser la solution partielle offerte.

Un ensemble de modèles, textures et codes supplémentaires est disponible.

Il faut remplacer les modèles, qui contiennent maintenant les attributs de coordonnées de texture et de normales.

Pour le code, certains fichiers sont à ajouter à votre projet directement (textures, shaders, shader_program, uniform_buffer, model_data), alors que d'autres contiennent du code à ajouter dans certains fichiers actuelles (car, main, model).

Il faudra adapter légèrement le code pour être en mesure d'utiliser la classe nouvelle classe de ShaderProgram.

De plus, une implémentation de UniformBuffer est disponible pour faciliter l'envois des informations de lumière au shader. Vous n'avez pas besoin de vous soucier de cette classe.

Vous pouvez retirer tout ce qui avait rapport au polygone de la partie 1 du tp1.

Partie 1 : textures

Maintenant que la scène 3D est peuplée de modèles 3D, on remarque assez facilement les limitations de ce qu'on peut représenter avec seulement des couleurs sur chaque sommet. Pour corriger cette aspect, on peut ajouter des textures afin d'avoir un meilleur contrôle sur ce qui est dessiné sur la surface de nos triangles.

Pour se faire, il va falloir charger les textures de chaques modèles et les appliquer sur ceux-ci lorsqu'il sera temps de les dessiner avec la méthode `use`. Une classe `Texture2D` contenant du code pour le chargement des images vous est fournis. Vous devez complété son constructeur pour charger l'image

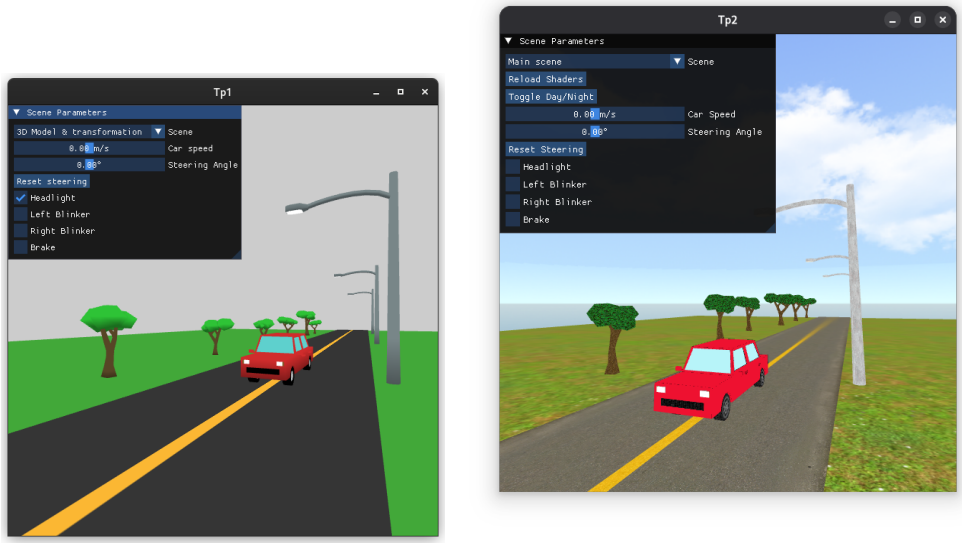

FIGURE 1 – Comparaison de avant et après l'ajout des textures

en tant que texture, ainsi que les autres méthodes pour utiliser la texture.

Utiliser les méthodes `setFiltering`, `setWrap` et `enableMipmap` à l'extérieur du constructeur (dans la méthode init dans le main.cpp).

On veut seulement que les textures se répètent sur les objets suivant : le sol, de la route, les arbres et les lampadaires.

On veut que les textures aient un fini lisse, à l'exception des arbres, des lumières de lampadaire et des fenêtres de la voiture, où on veut voir les pixels de façon plus définie.

Pour les modèles 3D, l'attribut de coordonnée de texture est déjà défini dans les modèles. Le chargement de celle-ci et la configuration de l'attribut a été fait pour vous dans la classe Model.

Pour ce faire, vous devez remplir dans l'ordre les sections suivantes :

- La classe `Texture2D` ;
- Initialiser les textures dans `main.cpp` ;
- Remplacer tous vos utilisation de shader par celui de `CelShading` ;
- Modifier `CelShading` pour faire le dessin avec seulement des textures pour commencer ;
- Dans `main.cpp`, ajouter les utilisations de texture avant chaque commande de dessin ;

Ceci devrait être suffisant pour avoir les textures sur les objets.

Pour dessiner le ciel, le concept de cubemap est utilisé. Il est recommandé de consulter le site LearnOpenGL sur le sujet.

Pour les vitres de l'automobile, un effet de mélange de couleur sera fait. Utiliser le facteur classique : le facteur source sera l'alpha source (a_src) et le facteur destination sera un moins l'alpha source ($1-a\_src$). N'oublier pas d'utiliser la transparence dans la couleur finale du fragment (dans le shader).

Partie 2 : effet de contour avec stencil

Maintenant que la scène contient plus de détail avec les textures, on voudrait la rendre plus stylée en lui donnant un aspect "cartoon". Ces dessins sont très distinct avec le tracé de contour noir, ainsi qu'un petit nombre de nuances de couleur pour les d'ombrages. Cela sera un bon moment de mettre en pratique les connaissances de test de stencil afin de réaliser l'effet de contour.

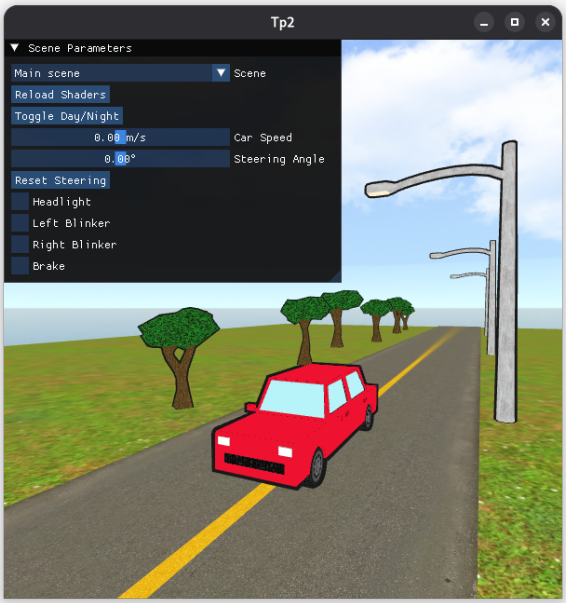

FIGURE 2 – Effet de contour sur les modèles 3D

Avant toute chose, il faut mentionné ici que l'effet n'est probablement pas le meilleur pour avoir un résultat de bonne qualité, mais celui-ci sera tout de même formateur.

Plusieurs méthodes sont faisables et autorisées dans notre cas, mais la façon la plus direct d'arriver au résultat souhaité est la suivante :

- Dessiner l'objet en couleur, mais aussi dans le tampon de stencil lorsque seulement le test de profondeur passe. On ne teste pas cette objet avec le stencil.
- On change le teste pour ne rien écrire dans le tampon de stencil.
- On change le teste pour réussir lorsque le tampon de stencil n'a pas eu d'écriture.
- On dessine une seconde fois le même objet, mais avec un autre shader, permettant de l'agrandir.
- On restaure l'état après avoir fini de dessiner tous les objets qui avaient besoin du test, et on désactive le test pour les objets qui n'en ont pas de besoin.

Il faudra faire attention à l'ordre de dessinage surtout pour les vitres.

Partie 3 : illumination

Pour finaliser le tout, on voudrait ajouter plus de réalisme dans notre scène avec les calculs d'illumation.

FIGURE 3 – Illumination de la scène, de jour et de nuit

Les calculs devraient être fait dans le référentiel de la vue.

2 Exigences

2.1 Exigences fonctionnelles

Partie 1 :

E1. TBD [1 pt]

Partie 2 :

E2. TBD [1 pt]

Partie 3 :

E3. TBD [1 pt]

2.2 Exigences non fonctionnelles

De façon générale, le code que vous ajouterez sera de bonne qualité. Évitez les énoncés superflus (qui montrent que vous ne comprenez pas bien ce que vous faites !), les commentaires erronés ou simplement absents (lorsque nécessaire), les mauvaises indentations, etc. Retirer les commentaires de TODOs dans la remise. [2 pts]

2.3 Remise

Créer une archive nommée « **INF2705_remise_TPn.zip** » que vous déposerez ensuite dans Moodle. (Moodle ajoute automatiquement vos matricules ou le numéro de votre groupe au nom du fichier remis.)

Ce fichier `zip` contient tout le code source du TP que vous avez modifié (`main.cpp`, `models.cpp`, `car.cpp`, `car.hpp`, `textures.hpp`, `shaders/*.glsl`).