POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

INF2705 Infographie

Travail pratique 3

Rendu de scènes complexes

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 les notions d'approximation et d'utilisation courbes, de shaders de géométrie et tessellation, puis de calcul parallèle sur la carte graphique. Il sera en mesure de dessiner une approximation de courbe bezier, de générer du gazon sur n'importe quelle surface et de faire des simulations de particules sur la carte graphique.

Le travail fera l'utilisation des fonctions d'OpenGL `glPatchParameteri`, `glDispatchCompute`, puis l'utilisation des shaders de géométrie, tessellation et de calcul.

1.2 Travail demandé

Le tp3 est une fois de plus une continuation du tp2. Il faudra reprendre à partir de votre ancien code du tp2 ou tp1, libre au choix. Les éléments du tp2 ne sont pas requis au fonctionnement des éléments du tp3, mais rend le rendu plus agréable. Seulement les nouvelles parties de code seront évaluées.

Quelques fichiers et fonctions de bases sont fournis.

Partie 1 : Spline bézier

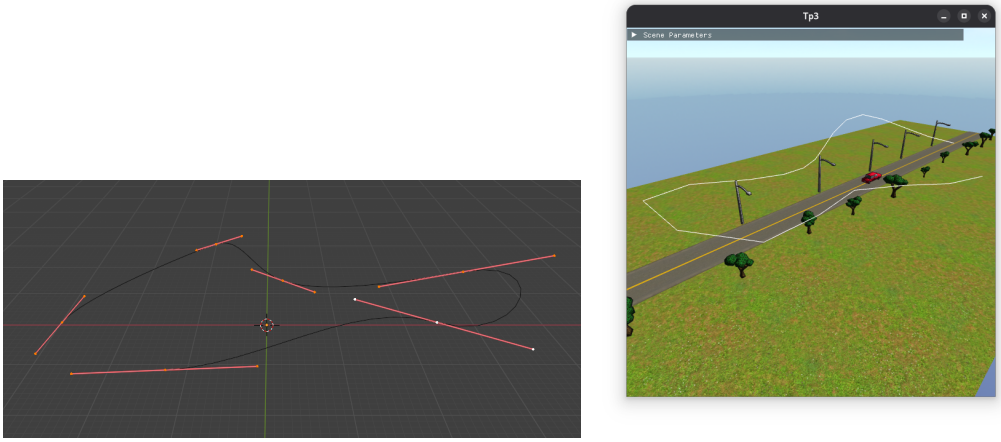

FIGURE 1 – Spline bezier

Bien que les triangles puissent nous permettre de dessiner la majorité des objets, ils ne sont pas adaptés pour représenter toutes les formes. Les courbes sont un bon exemple.

Dans un premier lieu, on fera le dessin d'une spline bézier. Cette spline est constitué de 5 courbes béziers à interpolation cubique (donc 4 points de contrôle par courbe).

Pour ce faire, le code de chargement de la courbe ressemblera beaucoups à celui de la partie 1 du tp1. Il est recommandé de procéder comme suit :

- Allouer suffisamment d'espace pour les sommets des courbes pour la durée totale du programme à l'initialisation. Les points faisant parties de deux courbes peuvent être dédoublé sans problème.
- Faire le calcul d'évaluation d'un point sur la courbe.
- Approximer une courbe selon le nombre de division (algorithme au choix).
- Approximer chaque courbe pour former la spline.
- Envoyer les données à jour à la carte graphique.
- Dessiner la courbe d'une couleur uniforme.

N'importe quel shader pourrait être utiliser pour le dessin de celle-ci (un nouveau très simple ou ancien). Pour que l'illumination n'ait pas d'impact dessus, le matériel `bezierMat` peut être utilisé avec le shader `CelShading`.

Une fois qu'on est certain du calcul pour approximer cette spline, on pourra animer la caméra pour suivre une trajectoire sur la spline.

Il sera pertinent d'utiliser une matrice de vue qui fixe toujours la voiture. Par la suite, on pourra déplacer la caméra selon la trajectoire.

Partie 2 : effet de gazon procédural

Bien que la scène soit peuplée d'arbres et de lampadaires, le reste de la surface du sol semble très vide. Ajoutons-y du gazon !

FIGURE 2 – Résultat final de l'effet de gazon procédural

Pour ce faire, il faudra définir une région sur laquel générer notre gazon. À l'initialisation, on créera un mesh de plusieurs carrées constitué de triangles. Ceux-ci seront nos patches en entrée pour le shader de tessellation.

Pour commencer, il est fortement recommander de déboguer cette partie en utilisant un affichage de fil de fer afin de bien voir les divisions de la tessellation (`glPolygonMode`).

Avec le shader de contrôle de tessellation, on pourra définir le niveau de division de la patch traitée en fonction de la distance par rapport à la caméra.

Une fois qu'on sera capable de voir les divisions correctements sur le long des arêtes, on changera le mode de primitive du shader de tessellation pour générer des points. Ces points seront l'entrée du shader de géométrie, qui pourra les convertir en brin d'herbes.

Au niveau du code, il faudra faire les modifications suivantes :

- Création du nouveau shader pour le gazon (n'oublier pas j'ajouté l'appel à `reload` dans `drawFrame`).
- À l'initialisation, générer les patches triangulaires de façon à avoir des carrées sur la surface du sol
- Dessin des patches, n'oublier pas d'envoyer les matrices nécessaires.
- Réalisation des shaders pour voir et déboguer les divisions.
- Modification des shaders pour générer le gazon.

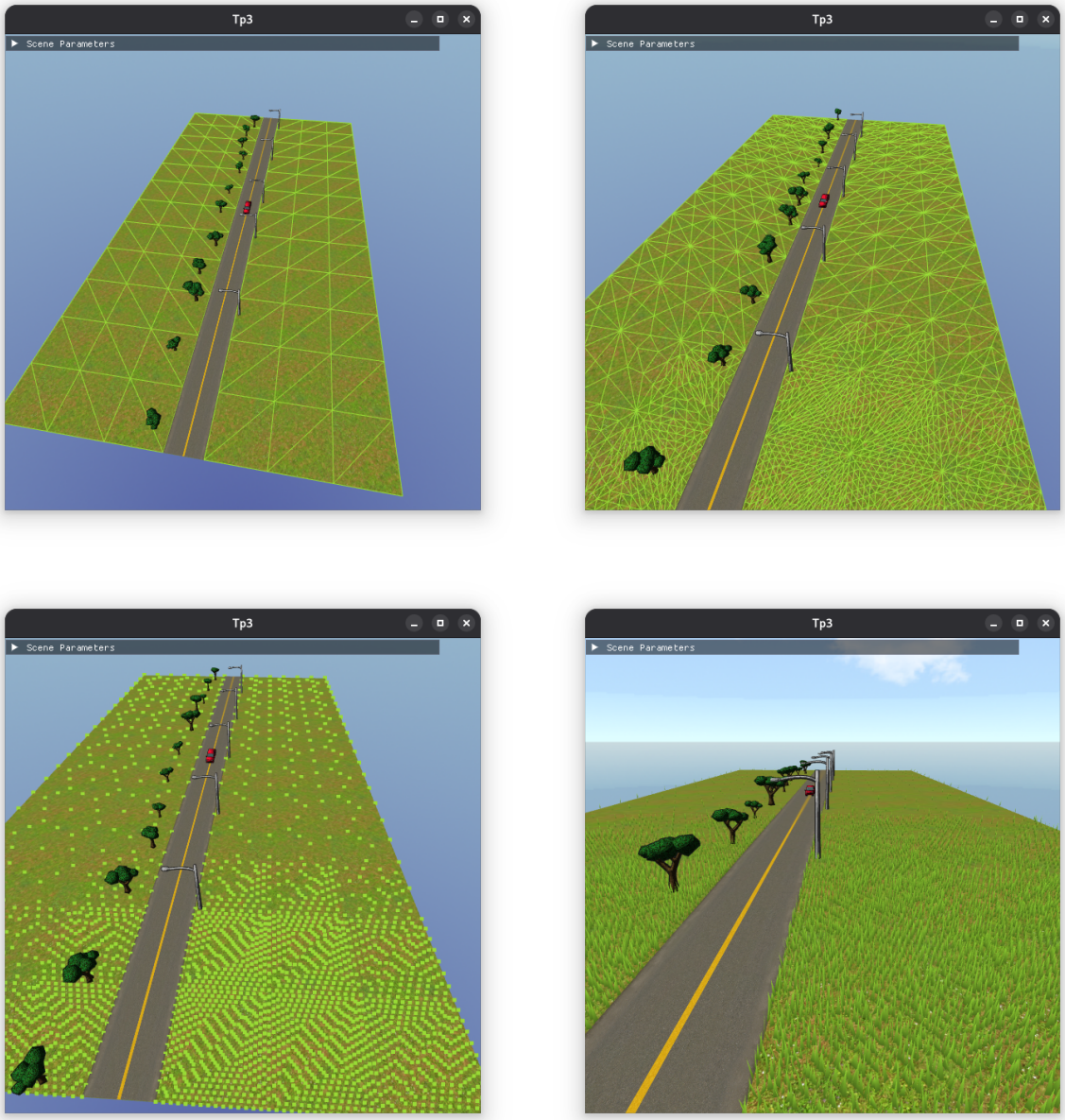

FIGURE 3 – Développement de l'effet de gazon

Partie 3 : Effet de particules

À venir.

2 Exigences

2.1 Exigences fonctionnelles

Partie 1 :

E1. TBD

Partie 2 :

E2. TBD

Partie 3 :

E3. TBD

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`, `model_data.hpp`, `shaders.cpp`, `shaders.hpp`, `car.cpp`, `car.hpp`, `textures.cpp`, `shaders/*.glsl`).