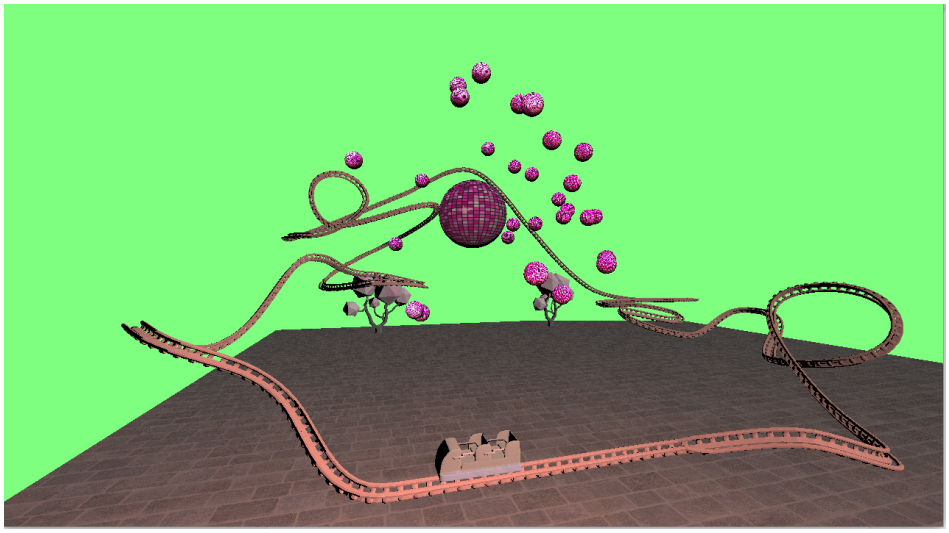
E5-INF-5204C

CHEN Emilie
LU Arnold

Rapport synthèse d'images :

Roller Coaster OpenGL



VENCESLAS Biri

Université Gustave Eiffel
ESIEE PARIS

Sommaire :

1) Introduction

Dans le cadre de l'unité INF_5204C, nous devons réaliser un projet de synthèse d'images avec **OpenGL**, une API destinée au rendu par rastérisation. Nous allons au travers de notre code transformer les primitives 3D tels que les points, lignes et triangles en fragments grâce au pipeline programmable. L'avantage d'OpenGL réside dans l'exploitation des **GPUs**, permettant d'avoir accès à la puissance des processeurs exclusivement dédiés au calcul parallèle.

L'objectif principal de ce projet est de créer une **application graphique** qui aura pour thème le **roller coaster** (montagnes russes) mais reste libre dans sa mise en œuvre.

2) Manuel d'utilisation

a) Compilation

Remarque : le projet se nomme "TP9"

Pré-requis : Système Linux, Cmake

1) Après avoir téléchargé le code, ouvrir un terminal et aller dans le répertoire projet.
2) Créer un répertoire "build" et effectuer la commande "cmake ../"
3) Finalement, lancer un "make", aller dans le répertoire "bin" et lancer "./TP9_exe" afin d'exécuter le programme.

b) Touches utilitaires

Dans notre application, nous avons implémenté une **caméra mobile, pilotée par l'utilisateur**. La caméra permet à l'utilisateur de se déplacer dans toutes les directions et de tourner sur lui-même (à l'image des jeux à la première personne).

Souris : Rotation de la caméra

Touche Z : bouger vers l'avant

Touche S : bouger vers l'arrière

Touche Q : bouger vers la gauche

Touche D : bouger vers la droite

3) Spécification techniques

Une des contraintes de ce projet était d'implémenter l'application en **C/C++** tout en compilant sous Linux. Nous utilisons bien sûr comme bibliothèque graphique **OpenGL** (version 3+) et les bibliothèques vues en TD (**GLFW , GLM**) pour tout ce qui est gestion des fenêtres graphiques, interface homme machine, accès à des classes, …).

Pour ce qui est du chargement de modèles 3D pré-faits, nous utilisons la bibliothèque **Assimp**. Assimp est capable d'importer des dizaines de formats de fichiers de modèles différents (et d'exporter vers certains également) en chargeant toutes les données du modèle dans les structures de données généralisées de cette dernière.

L'application sera disponible sur **GitHub** via ce lien :
https://github.com/arnoldlu2000/OpenGL_rollercoaster

4) Fonctionnalités implémentées

Notre application :

- met en place une caméra mobile (**Freefly Camera**) avec les touches Z, Q, S, D pour se déplacer
- contient 2 objets texturés : la **boule disco tournante** (avec ses "**lunes rotatives**") au centre et le **sol en pierre** sur lequel repose le roller coaster
- modélise un **sol**, contenu dans une texture d'environnement

- possède un modèle d'illumination à 3 lumières : **1 lumière directionnelle** et **2 lumières ponctuelles tournantes** venant de la boule disco principale (pour donner un effet de boule disco)
- présente un parcours de rollercoaster contenant plusieurs virages et 2 loopings (boucles)
- présente un **wagonnet** qui se déplace sur une **partie rectiligne** du roller coaster

5) Evolution de l'application

Nous avons un peu modifié le modèle de Phong afin de pouvoir rajouter une lumière ambiante. Nous utilisons notamment une texture (sampler2D) à la place d'un Kdiffuse (vec3).

Nous avons aussi ajouté des décors (le wagonnet, le roller coaster et des arbres) via des chargements de modèles 3D pré-faits avec la bibliothèque Assimp. Nous avons suivi un tutoriel de chargement de modèles 3D disponible sur ce site : https://learnopengl.com/Model-Loading/Assimp

Nous avons créé et texturé (UV-mapping) ces modèles 3D au préalable sur Blender afin de les importer dans notre application OpenGL. Nous manipulons et importons des "Wavefront objects" (.obj), un format de données simple qui représente la géométrie 3D (la position de chaque vertex, la position (U, V) de chaque coordonnée de texture, les normales de vertex, …)

L'idée derrière ce travail est de créer une classe C++ qui représente un modèle 3D dans son intégralité, comprenant plusieurs meshs (Un mesh est un objet tridimensionnel constitué de sommets, d'arêtes et de faces organisés en polygones) et textures. Ainsi, on pourra alors manipuler un modèle et recueillir les données directement en passant par l'objet modèle C++.