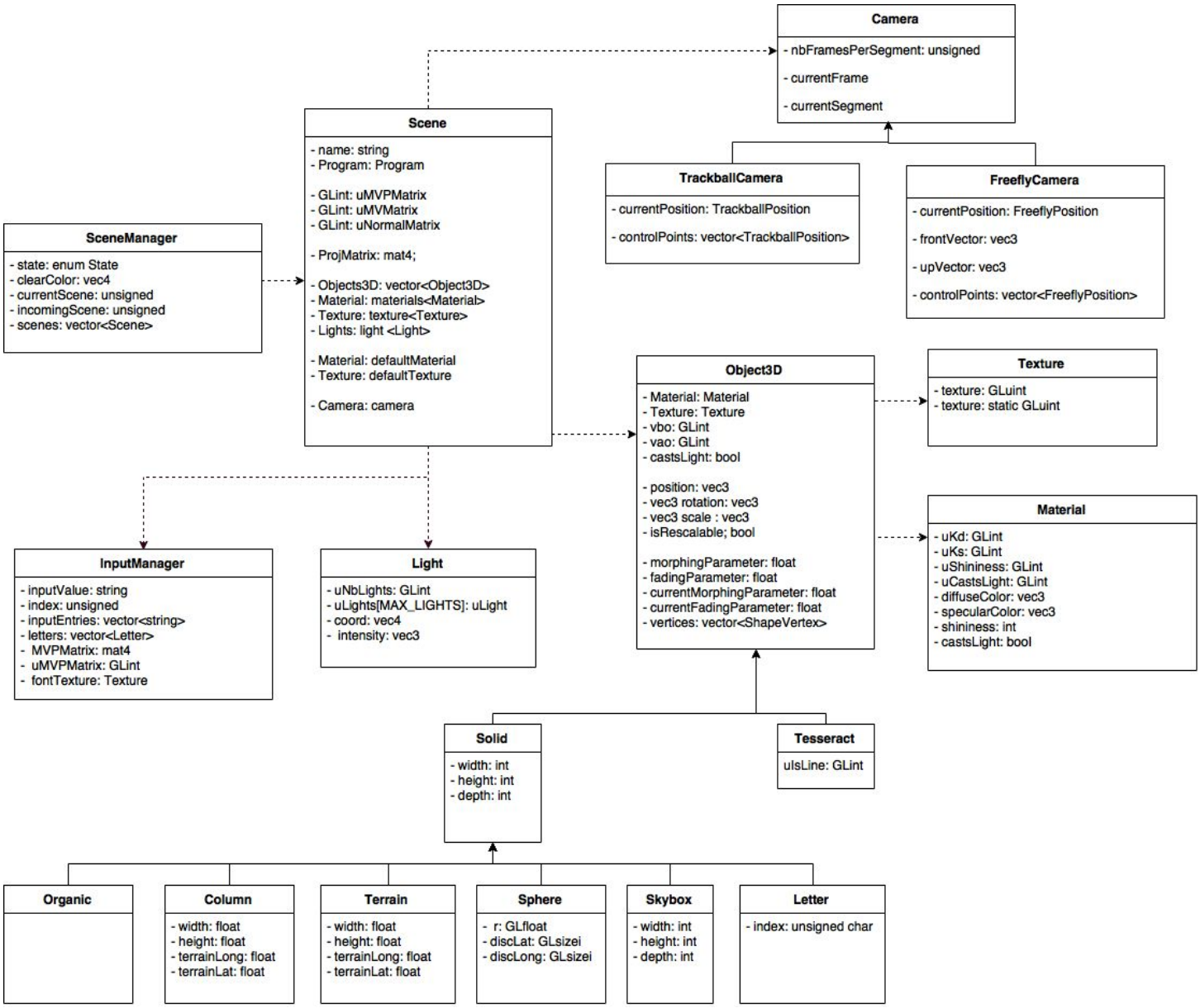
Vertxt

Projet OpenGL/C++

Robin Jungers
Charlotte Nortier
Jules Tantot

1. Schéma de données



2. Problèmes rencontrés

Les points critiques du projet se sont articulés autour de différents axes :

→ *Structure*
Hiérarchie générale.
Quels design patterns sont adaptés, ou non.
Polymorphisme: comment et où définir des comportements généraux ?

→ *Algorithmique*
Transitions séquentielles, en parallèle des boucles d'affichage.

→ *Mathématiques*
Calcul de normales sur des maillages.
Calcul de nombres aléatoires continus selon plusieurs dimensions; bruit de Perlin.
Génération de structures aléatoires cohérentes.

→ *Technique*
Gestion des entrées texte multi-plateforme.
Gestion de la mémoire ré-allouée pour la mise à jour des vertex.

3. Résultats

En termes de performances, on observe de faibles baisses avec un nombre de points important. Les transitions, puisqu'elle demandent de re-calculer l'ensemble des points et des normales systématiquement, implique davantage de ralentissement. Ceci est toutefois contenu aux états de transitions, et pas généralisé pour chaque tour de boucle d'affichage.

Le rendu des textures, quand elles sont nombreuses, semble plus lourd. On l'observe assez clairement sur la scène "Basalt". Le calcul de densité atmosphérique (de brume) est ici contenu aux shaders, et n'a pas d'impact sur les performances.

Le moteur de rendu que nous avons développé, et sur lequel toute l'application repose, est polyvalent et semble bien fonctionner pour les tâches qu'on lui soumet.

En terme de rendu graphique, le résultat répond globalement bien à l'idée d'origine; malgré tout, à titre d'amélioration, il aurait été intéressant de proposer un plus large panel de shaders, accentuant encore la charte graphique de chaque scène.