Introduction à OpenGL et Qt 5.4

Ce tutoriel présente l'utilisation du processeur graphique (GPU) pour créer des applications de rendu 3D avec Qt et OpenGL. Ce tutoriel se base sur un exemple d'affichage d'une map en 3D.

L'objectif est de présenter en détail les fonctionnalités offertes par Qt permettant de travailler avec les GPU. Le module QtOpenGL, intégré dans le framework, nous permettra de réaliser des rendus 3D. Dans un premier temps, le rendu sera réalisé en utilisant le pipeline fixe et des fonctions qui sont maintenant dépréciées. Dans une deuxième étape, nous présenterons l'utilisation du pipeline programmable, des buffers et des shaders. Cela permettra au lecteur de pouvoir plus facilement adapter le code trouvé sur Internet (qui utilise souvent du code déprécié) en code moderne. Nous implémenterons différentes fonctionnalités classiques de la 3D : la gestion de l'éclairage, les ombres et les textures. Les techniques avancées de 3D, par exemple celles présentées dans les livres GPU Gems, n'utilisent pas de fonction de Qt spécifique et ne seront donc pas détaillées.

Ce tutoriel s'adresse en premier lieu aux développeurs ayant les bases de la programmation avec Qt. Avoir des notions en programmation 3D facilitera la compréhension, mais n'est pas indispensable. Certaines notions de base ne seront pas présentées, le lecteur devra donc se référer aux tutoriels que l'on peut trouver sur internet, comme par exemple :

- Développez vos applications 3D avec OpenGL 3.3
- Traduction des tutoriels Opengl-tutorial.org
- Traduction des tutoriels Ogldev

Pensez a vérifier quelle version d'OpenGL est utilisée dans les tutoriels que vous lisez, prenez au moins du OpenGL 2.

Sommaire

- Introduction à OpenGL et Qt 5.4
- Support d'OpenGL dans Qt
- Qt OpenGL - Générer un terrain
- Qt OpenGL - Envoyer des données au processeur graphique
- Qt OpenGL - Utilisation du pipeline programmable
- Qt OpenGL - Ajouter des lumières et des textures
- Qt OpenGL - Réaliser un rendu offscreen
- Qt OpenGL - Overpainting : dessiner en 2D avec QPainter sur une scène 3D
- Qt OpenGL - Gestion des extensions avec QGLContext::getProcAddress()
- Qt OpenGL - Annexes

OpenGL, Qt