Qt OpenGL - Introduction

Ce tutoriel présente l'utilisation du processeur graphique (GPU) pour créer des applications de rendu 3D et de calculs parallèles multiplateformes sous Qt. Deux API libres et multiplateformes seront utilisées afin de communiquer avec le processeur graphique : OpenGL pour la partie rendu 3D et OpenCL pour le calcul. Un des points-clés de ce tutoriel est la recherche d'optimisation des performances. Différentes méthodes de programmation seront proposées : partant de versions qu'on pourrait qualifier de “naïves” c'est-à-dire simples mais non optimisées, le souci de performances nous poussera à l'utilisation de techniques d'optimisation permettant souvent d'améliorer considérablement les performances. Un exemple récurrent est utilisé tout au long du tutoriel : la génération de terrain.

Contenu du tutoriel et objectifs

La recherche de performances dans les domaines du rendu 3D ou du calcul lourd a toujours été une préoccupation majeure pour les développeurs. Tout au long de ce tutoriel, nous allons étudier les différents moyens de réaliser des applications sous Qt utilisant au mieux la capacité de calcul mise à notre disposition par l'intermédiaire du processeur graphique. S'appuyant sur un problème classique de programmation 3D, le rendu de terrain, nous allons explorer les différentes techniques permettant d'optimiser l'utilisation de ces ressources de calcul.

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 sur le site de NVIDIA, n'utilisent pas de fonction de Qt spécifique et ne seront donc pas détaillées.

Dans la dernière partie, la bibliothèque QtOpenCL, provenant de Qt Labs, sera utilisée pour les calculs parallèles sur GPU. Le code présenté sera basé sur un exemple simple et didactique : le calcul des vecteurs normaux à des surfaces. Nous aborderons en particulier les techniques de profiling et de débogage et les benchmarks.

Environnement de développement

Pour pouvoir compiler le code présenté dans ce tutoriel, il faut pour commencer, disposer d'une version de Qt fonctionnelle. Certaines fonctionnalités, par exemple les buffers, nécessitent la version 4.7 de Qt. Pour la partie GPGPU, il faut également installer le module QtOpenCL, téléchargeable sur le site Qt Labs, ainsi que les pilotes de développement de la carte graphique, téléchargeable sur le site des constructeurs (par exemple, le CUDA pour les cartes NVIDIA). Le lecteur se référera aux différents manuels fournis pour leurs installations.

Prérequis

Ce tutoriel s'adresse en premier lieu aux développeurs ayant les bases de la programmation avec Qt. Avoir des notions en programmation 3D ou en GPGPU facilitera la compréhension mais n'est pas indispensable. Certaines notions de base ne seront pas présentées, le lecteur devra donc se référer aux autres tutoriels proposés par la rubrique Qt de Developpez, sur la programmation Qt en général et plus spécialement aux tutoriels Intégration d'OpenGL dans une interface Qt, Qt Graphics et performance - Le moteur de rendu OpenGL et Utiliser OpenCL avec Qt. Le lecteur pourra également consulter les tutoriels de la rubrique Jeux de Developpez, en particulier les tutoriels sur CUDA et sur la génération de terrain.

OpenGL, Qt