Projet C++ / OpenGL

Synthèse d'Images Imac 2

laurent.noel.c2ba@gmail.com
chatelet.guillaume@gmail.com

IM(A)ineCraft

Objectif principal

Créer un minecraft simplifié en C++/OpenGL 3.

Fonctionnalités minimales requises

- Univers en 3D composé de cubes alignés
- Contrôle d'un personnage en vue à la premiere personne
- Gestion minimale de la physique: le joueur doit se déplacer sur les cubes. Il peut également sauter.
- Possibilité pour le joueur de modifier l'univers en ajoutant ou en détruisant des cubes
- Description de l'univers à l'aide d'un fichier (chargement et sauvegarde de l'univers)
    - choisissez un format pour stocker votre univers, ce peut être XML, JSON, ou un format binaire custom. Dans tous les cas, motivez votre choix.
- Cubes de différents types

    - destructibles ou non
    - apparences variées définis selon une texture
- Possibilité d'ajouter des torches émettant de la lumière sur les faces des cubes (animation des flammes gérée avec la technique du billboarding). Lumière gérée avec des point lights, sans gestion des ombres, en nombre limité.
- Temps réel (au moins 30 images par seconde)

Contraintes techniques

- Compilation pour linux avec g++
- Utilisation de OpenGL 3+ (shaders, pas de pipeline fixe)
- Utilisation de la SDL pour la gestion des évenements et du fenêtrage
- Utilisation du mécanisme de rendu instancié en OpenGL 3+ (pour les cubes)
- Possibilité de définir facilement de nouveaux types de cubes sans modifier l'architecture existante (utilisation d'une interface, polymorphisme)
- Gestion des ressource (aussi bien coté GPU que CPU) en utilisant RAII
    - définition
    - note de TD
- Gestion de la mémoire
    - Le programme ne doit pas perdre de mémoire (à vérifier avec Valgrind)
    - N'utilisez donc l'allocation dynamique que lorsque c'est nécessaire
- Séparation moteur de jeu / moteur graphique claire
- Par groupe de 5 personnes maximum

Compétences mises en jeu

- Maitrise des concepts de base du C++
    - classes, RAII, polymorphisme, éventuellement templates
    - gestion de la mémoire
- Maitrise des concepts OpenGL appris durant l'année
    - VBOs, VAOs, shaders, caméra FPS, textures, lumières
- Intégration de techniques OpenGL plus avancées non rencontrées en TD (billboard, rendu instancié)
- Architecture d'un moteur de jeu simple

Idées d'améliorations

- effets sonores ou musique
- originalité de l'univers
- mise en scène / cinématique
- génération aléatoire d'univers
- rendu des ombres en utilisant la technique des shadows maps

- possibilité d'avoir un nombre illimité (ou presque !) de lumières (en utilisant la technique du deferred shading par exemple)
- intégration d'énnemis cubiques (comme dans le vrai Minecraft)
- gestion poussée du crafting
- création d'un t-shirt "Im(a)ineCraft" pour l'occasion (en offrir un à chacun des deux chargés de TD au moment de la soutenance)

Quelques conseils pour la gestion du projet:

Bien que le projet paraisse assez conséquent, la partie obligatoire peut normalement être codée rapidement en suivant quelques bonnes pratiques:

- Réfléchissez tous ensemble un minimum à votre architecture avant de vous jeter dans le code. Mettez vous dans une salle pendant une heure et faites un brainstorming au tableau. Organisez ensuite les idées pour avoir une idée globale de l'architecture du programme.
- Ne restez pas pour autant plusieurs jours sur cette partie: l'objectif reste d'avoir un résultat respectant les fonctionnalités, pas d'obtenir un code super souple mais ne proposant rien au final. Adoptez des petits cycles de la forme analyse - spécification - développement - tests.
- Définissez rapidement le rôle de chacun en fonction des affinités/compétences.
- Les phases d'analyse doivent vous permettre de réflechir ensemble à l'architecture globale et à la manière dont les modules intéragissent. C'est également le moment de se répartir les taches efficacement. Les taches doivent être suffisamment courtes pour pouvoir être intégrées et testées rapidement.
- Utilisez un outil de gestion de version (GIT par exemple) pour le partage du code. Le site https://bitbucket.org/ propose des repositories gratuits et privés.
- Ecrivez une petite surcouche à OpenGL (à la manière de **imac2gl3** en TD) que vous étofferez au fur à mesure du développement. Elle pourra contenir des classes utilitaires simples permettant de simplifier la gestion des ressources (RAII) ou l'utilisation de techniques récurrentes.

Le rendu instancié:

Le rendu instancié est une technique OpenGL permettant de dessiner en un seul appel le même objet un grand nombre de fois en changeant simplement de transformation. Cela permet de booster les performances du programme. C'est de plus très adapté à un Minecraft puisqu'on ne dessine quasiment qu'un seul type d'objet: le cube. Il vous est imposé d'utiliser cette technique mais ce n'est pas la priorité: commencez par coder votre moteur sans cela et une fois que les choses fonctionnent assez bien, intégrez la solution. Voici un lien vers un tutorial sur le rendu instancié afin de vous aider un peu: http://ogldev.atspace.co.uk/www/tutorial33/tutorial33.html.

L’objectif pour vous est d’apprendre à intégrer une technique que vous n’avez pas vu en TD dans un projet de plus grosse envergure que les exercices de TD.