UNIVERSITÉ DE SHERBROOKE

Projet personnel

PRÉSENTÉ À :

Gabriel Girard, professeur

PAR

Jimmy Fortin
10 006 129

24 avril 2012

# Préambule

Pour la réalisation de ce travail, j’ai travaillé dans un engin graphique que je suis en train de construire à temps perdu. La structure du projet complet est encore en conception, donc il est normal que plusieurs morceaux soient aux mauvais endroits. Normalement, les morceaux présentés dans la section suivante sont censés être terminés et fonctionnels.

Les solutions ont créé à l’aide de Visual Studio 11 Beta Ultimate, qui est disponible à cette adresse : <http://www.microsoft.com/visualstudio/11/fr-fr>. Les exécutables sont compilés en version 32 bits et toutes les dll sont fournies pour pouvoir être en mesure de les faire fonctionner sur Windows. Visual Studio 11 est nécessaire pour pouvoir effectuer des tests unitaires facilement et rapidement dans C++, puisqu’il donne des outils nativement.

Les fichiers binaires sont disponibles dans le répertoire de partage dans l’archive ***Code.zip***.

# Sections relatives à ce devoir

Le projet est divisé en deux solutions, une pour l’éditeur et une pour l’engin. La solution de l’éditeur est faite en C# et utilise les Sockets fournis dans le Framework.Net. Puisque cette solution n’existait pas avant d’embarquer sur ce travail, tout ce qui est dans cette solution peut être évalué. La solution de l’engin est écrite en C++ et m’a demandé la réécriture d’une classe semblable à celle de C# pour être en mesure de bien effectuer le travail. Les classes pouvant être évaluées sont celles se retrouvant dans les projets **Widecreations.Networking**, **Widecreations.System** et les tests se retrouvent dans les projets se terminant avec Tests.

# Fonctionnalités offertes

Le but de ce projet était d’être en mesure d’établir la communication avec plusieurs clients sur un engin graphique, et ce dans des langages de programmation différents. Pour l’instant, les seules fonctionnalités offertes sont simples, mais montrent bien les possibilités offertes d’un tel système. Grâce à ce système, il est possible de mettre à jour les valeurs de couleurs pour les champs de textes montrant le FPS et la quantité de mémoire.

# Parallélisme

Au niveau de l’éditeur, il a seulement été nécessaire d’utiliser les fonctions de la classe Socket fournies par le Framework .Net qui permettait déjà de faire du parallélisme. Pour ce qui est de l’engin, la librairie SDL a été grandement appréciée puisqu’elle permet nativement d’implanter une gestion de thread multiplateforme. Les exceptions au niveau du Socket n’étaient pas très évidentes à régler puisqu’elles étaient lancées dans des threads totalement différents et dans des méthodes qui ne pouvaient être écoutées du thread principal. Pour pallier ce problème, il a fallu utiliser un système semblable aux événements, puisque dans la classe Socket et CommandServer un pointeur de fonction est gardé et la fonction est exécutée au moment que l’exception est lancée. Si personne n’écoute cet événement, donc que le pointeur est vide, il faut lancer une exception normalement.

# Exécution du programme

Pour être en mesure d’exécuter les deux programmes, il est nécessaire d’être sur l’environnement Windows. Pour ce qui est de l’engin graphique, le code a été testé avec une carte vidéo Nvidia et tous étaient fonctionnels. Au niveau de l’éditeur, il est nécessaire d’avoir le Framework.Net 4.5 d’installés.

Les fichiers binaires sont disponibles dans le répertoire de partage dans l’archive ***Binaires.zip***.

De courtes démonstrations ont été faites dans le répertoire de partage dans l’archive ***Demonstration.zip***.

# Diagramme de classe

