

Conférence Microsoft Nouveautés de Visual C++ 2008TR1, MFC et autres avancées

par Loïc Joly

Date de publication : 23/04/2008

Dernière mise à jour : 19/6/2008

Ce document est un compte rendu d'une conférence organisée le 23/04/2008 par Microsoft dans le cadre des Mercredis du développement.
Le conférencier était Boris Jabes, qui travaille chez Microsoft à l'amélioration de l'IDE de Visual C++.

I - Introduction.....	3
II - Visual C++ 2008.....	4
II-A - Manifestes compatibles Vista.....	4
II-B - Enregistrement COM.....	4
II-C - Nouvelles fonctions MFC.....	4
II-D - Nouvelles options de compilation.....	5
II-E - Interop.....	5
III - Le feature pack.....	6
III-A - Les MFC.....	6
III-B - TR1.....	6
IV - Visual C++ 10 ?.....	8
IV-A - La fin des .ncb.....	8
IV-B - Le parallélisme.....	8
IV-C - Ms-Build.....	8
IV-D - Divers.....	8

I - Introduction

Ce document est un compte rendu d'une conférence organisée le 23/04/2008 par Microsoft dans le cadre des Mercredis du développement. Le conférencier était Boris Jabes, qui travaille chez Microsoft à l'amélioration de l'IDE de Visual C++. Comme on dit dans ce genre de cas, les points intéressants de ce document sont de lui, les erreurs sont de moi.

Ce document est destiné à des gens connaissant un minimum le C++, ainsi que les versions actuelles de l'environnement de développement Visual Studio, dans sa partie Visual C++. La conférence portait exclusivement sur les aspects C++ de l'outil, et passe entièrement sous silence les autres parties. N'utilisant pas moi même les MFC, je resterai assez vague sur les nouveautés apportées dans ce domaine.

A mon sens, l'un des buts principaux de cette conférence était de rassurer les développeurs C++ que ce langage restait un langage majeur sous la plate-forme Windows, d'ailleurs largement utilisé en interne chez Microsoft pour développer le cœur de Windows, d'Office, des autres compilateurs...

La conférence était filmée, et est disponible en [webcast Microsoft](#).

II - Visual C++ 2008

La première partie était une présentation des nouveautés de Visual C++ 2008, déjà disponibles. Ces nouveautés sont somme toute assez peu nombreuses, surtout par rapport à l'intention annoncée par Microsoft de conserver le C++ comme un langage de premier choix. C'est surtout lié à des contraintes de planning (surtout que les équipes C++ ont visiblement pendant un temps dû tourner au ralenti), et les plus grandes nouveautés seront présentées dans les sections suivantes.

Les principales modifications ont un leitmotiv principal : Supporter Vista.

II-A - Manifestes compatibles Vista

Afin de gérer la sécurité, Vista prend en compte une option supplémentaire dans le manifeste d'un programme. Cette option peut prendre trois valeurs :

- **En mode normal** : Dans ce cas, le programme déclare qu'il n'a pas besoin de privilèges spéciaux pour s'exécuter.
- **En mode privilégié** : Dans ce cas, le programme déclare qu'il a besoin de privilèges spéciaux pour s'exécuter. Au lancement du programme, Vista demandera une confirmation à l'utilisateur, plus éventuellement un mot de passe administrateur.
- **Non positionnée** : C'est le mode de compatibilité pour les anciens programmes. Le programme est alors exécuté comme en mode privilégié, mais dans un bac à sable, qui fait que les modifications qu'il croit effectuer au système le sont dans une copie virtuelle de celui ci, qui sera détruite quand on quittera le programme.

Bien qu'il soit possible de modifier manuellement cette option dans Visual C++ 2005, la version 2008 offre une option de projet pour aboutir au même résultat.

II-B - Enregistrement COM

Toujours en termes de sécurité, Vista demande confirmation pour l'enregistrement de serveur COM au niveau système. Quand on est en cours de développement, et que l'on enregistre des serveurs toutes les 2 minutes à chaque compilation, ça peut devenir assez désagréable. Visual Studio 2008 transforme de manière transparente cet enregistrement depuis l'environnement de développement en enregistrement au niveau utilisateur, qui lui n'a pas besoin de privilèges spéciaux.

II-C - Nouvelles fonctions MFC

Ces fonctions se résument à :

- La gestion de menus déroulants "furtifs", masqués par défaut, et apparaissant quand l'utilisateur en a besoin. Ces menus ne sont pas compatibles avec une application MDI.
- Des gros boutons à cliquer avec une mise en forme riche du texte remplaçant des boutons radios pour effectuer un choix parmi N en un seul click
- Des boutons à cliquer ayant l'apparence de liens internet
- Un contrôle de saisie d'url acceptant en plus des IPV4 classiques les IPV6 et les noms type DNS, avec validation de la saisie
- Un nouvel éditeur d'icônes

II-D - Nouvelles options de compilation

De nouvelles options de compilation de code, principalement destinées à en rendre le craquage plus difficile ont été rendues disponibles.

`/DYNAMICBASE` permet de faire que d'une exécution à l'autre, les DLL soient chargées en des emplacements différents, ce qui fait qu'une attaque dans une zone mémoire spécifique n'a qu'une chance sur 256 d'avoir l'effet souhaité.

`/NXCOMPAT` place le processeur dans un mode où seul du code présent dans une zone mémoire prédéfinie est exécutable, empêchant d'exécuter du code dans la section de données.

`/MP`, dans un autre registre, a pour but de permettre la compilation dans plusieurs threads (c'était déjà possible dans Visual Studio 2005 au niveau du projet, cette fois c'est supporté au niveau du fichier) (1).

II-E - Interop

Un dernier volet d'améliorations concerne l'interopérabilité entre du code C++ et du code managé. Le présentateur est allé assez rapidement sur ces points, peu de monde dans l'audience utilisant du code managé. Ces fonctions s'articulent autour de trois points :

Tout d'abord une bibliothèque permettant de transformer les objets depuis du code managé vers du code non managé, et vice versa, à l'aide des templates `marshal_as`. A noter que la transformation dans le premier sens gère automatiquement la notion de pinning nécessaire à la transformation, avec la notion de `marshal_context`. Tant que l'objet de type `marshal_context` qui sert à traduire des données managées est vivant, ces données sont verrouillées à une adresse mémoire fixe.

En second lieu, ce qui se nomme STL/CLR, ou STL.NET, une série de conteneurs managés, reprenant les conteneurs de la bibliothèque standard STL du C++, et permettant principalement de :

- Wrapper des conteneurs standards pour les exposer dans du code managé, sans pour autant devoir copier les éléments uns à uns vers une collection managée comme il était courant de le faire préalablement.
- Utiliser les mêmes algorithmes que la bibliothèque standard sur des conteneurs managés. À titre personnel, le `remove_if` était une fonction cruellement manquante des conteneurs .NET.

Finalement, un point qui devrait bien faciliter la vie des développeurs, une modification du système de build afin de corriger un problème de recompilation excessive. A la base, les assembly .NET ne séparent pas comme peut le faire le C++ l'interface de l'implémentation du code. Ce qui a pour conséquence que, dans la version 2005, la moindre modification d'une assembly, même dans son implémentation, provoque la recompilation de toutes les assemblies qui en dépendent. Ce qui est un moindre mal dans le monde C# où les temps de compilation sont rapides devient un réel calvaire en C++. La modification apportée consiste à extraire d'une assembly une pseudo-assembly d'interface, qui sera normalement assez stable, et à dépendre de celle-ci au lieu de dépendre de l'assembly globale.

III - Le feature pack

Ce **feature pack**, téléchargeable gratuitement sur le site de Microsoft présente une mise à jour importante de deux bibliothèques livrées avec le compilateur.

III-A - Les MFC

Ce pack contient une ribambelle de nouveaux contrôles utilisables avec les MFC. L'objectif principal est de pouvoir donner à une application MFC un look&feel semblable à celui d'une application moderne (lire Microsoft Office) avec un coût de développement réduit. On nous a indiqué que cette version doublait pratiquement la taille du code source des MFC.

Parmi les nouvelles options disponibles, on peut noter :

- La gestion de documents MDI à l'aide de tabs, et non plus de fenêtres, un peu comme dans Visual studio.
- La gestion de bandeau offrant les mêmes fonctions que celui d'Office 2007.
- La gestion d'un menu rond (je ne sais pas son nom officiel) comme celui d'Office 2007.
- La gestion de feuille de style permettant de relooker son application aisément

Le point que je trouve un peu dommage, c'est qu'il ne s'agit là que d'ajouts. L'architecture même des MFC, qui est un peu vieillotte, peu orientée objet, statique, n'a pas été mise à jour. L'objectif principal est la compatibilité avec le code existant, et malgré le souhait de se rapprocher d'une architecture plus moderne (à la Qt, par exemple), aucune solution permettant de respecter cette contrainte n'a été proposée pour l'instant.

III-B - TR1

La norme C++ est sortie en 98, et la seconde version de la norme, nommée C++0x est en cours de finalisation. Entre temps, le comité d'évolution du C++ a sorti, un document, nommé TR1 (technical report 1), indiquant certaines extensions au C++ (dans sa partie bibliothèque) qui feront selon toute probabilité partie de C++0x. On peut voir cette extension comme une version 1.1 de la norme.

Le TR1 est composé de trois parties, des évolutions destinées à la compatibilité avec le C99, une série de fonctions mathématiques, et des bibliothèques d'intérêt général, issues de bibliothèques développées dans le cadre de boost. C'est cette dernière partie, jugée comme étant la plus directement utile au plus grand nombre, que Microsoft a introduit dans le feature pack. Les autres parties sont prévues pour la prochaine version du compilateur.

Je ne vais pas trop entrer dans le détail de ces bibliothèques qui mériteraient pour la plupart un article dédié (certaines en ont déjà un, de l'époque où elles faisaient partie de boost), mais juste en faire un résumé rapide :

- **shared_ptr, weak_ptr** : Pointeurs intelligents implémentant le concept de pointeurs se partageant, par comptage de référence, une même zone mémoire.
- **mem_fn, function, bind** : Support de base de la programmation fonctionnelle, objets fonctionnels, adaptateurs permettant d'obtenir un functor à partir d'un autre ayant des arguments différents...
- **regex** : Moteur d'évaluation d'expressions régulières.
- **type_traits** : Système d'aide à la méta-programmation permettant d'obtenir des informations sur les capacités d'un type.
- **random** : Génération de nombres aléatoires.
- **tuple** : Extension de pair à un nombre variable d'éléments, sorte de struc anonyme.
- **array** : Tableau de taille fixe, à allocation sur la pile, mais plus sur d'utilisation qu'un tableau à la C, et compatible STL.
- **unordered_map, unordered_set...** : Version à base de hash table des conteneurs classiques map, set... (qui eux sont basés sur un arbre).

Ces éléments étant déjà disponibles dans boost, y a-t-il intérêt à basculer vers ceux fournis avec le compilateur ? A priori oui, pour trois raisons : Le fait qu'elles soient justement fournies avec le compilateur va probablement faciliter leur acceptation dans certains milieux. De plus, certaines synergies d'optimisation sont possible à partir du moment où toute la bibliothèque est fournie par une même entité (exemple : un `vector<shared_ptr<T>>` pourra être spécialisé pour éviter d'avoir à modifier le comptage de référence quand on copie le pointeur dans le cadre d'une opération de réallocation). Enfin, on peut espérer une bonne intégration dans l'IDE, et en particulier la visualisation des objets dans le debugger.

IV - Visual C++ 10 ?

IV-A - La fin des .ncb

Dans cette dernière partie de la présentation (où, si j'ai bien compris, la caméra été coupée), Boris Jabes nous a présenté ce sur quoi travaillait l'équipe C++ de Microsoft en ce moment, afin de préparer la version 10 de Visual studio (qui devrait sortir en 2009 ou 2010, en même temps que les prochaines versions de Windows et d'Office). Bien entendu, les informations que je reprends ici ne sont qu'à titre indicatif, afin de deviner à quoi pourra ressembler la prochaine version du produit.

Tout d'abord, le point principal est le fait que l'IDE est difficilement utilisable pour des gros projets, à cause de lenteurs excessives. De plus, les outils d'édition offrent une productivité limitée en C++ par rapport à ce qu'ils offrent dans d'autres langages comme C#, en matière d'aide à la saisie et au refactoring de code. Une des limitations de l'éditeur actuel est qu'il utilise le format .ncb pour décrire une analyse dynamique du code servant par exemple à IntelliSense, et que ce format est fort ancien (une trentaine d'années je crois) et peu flexible. Il n'a pas non plus été conçu initialement pour de très gros projets.

L'équipe a donc décidé de faire table rase sur ce format, et a réécrit cette gestion à l'aide d'une base de donnée de type SQL, directement stockée dans un fichier (sans serveur), qui permet d'avoir un format extensible, tout en ayant une empreinte mémoire gérable par le client. De plus, le parseur utilisé actuellement pour faire l'analyse de code est différent de celui du compilateur, ce qui fait que toute une partie du code n'est pas reconnue correctement. Dans la nouvelle version, c'est le front-end du compilateur lui même qui sera utilisé. Espérons que le résultat sera à la hauteur !

IV-B - Le parallélisme

L'utilisation du parallélisme pour mieux utiliser les processeurs multi-cœurs actuels est un sujet d'actualité, et un point de passage obligé pour l'augmentation des performances du code. Sans préciser les détails, on nous a indiqué que c'était une priorité de faciliter ce genre de code dans la prochaine version du compilateur, probablement par l'intermédiaire d'une bibliothèque dédiée.

IV-C - Ms-Build

Par rapport aux autres outils de Visual Studio, Visual C++ fait un peu bande à part en terme de processus de build. Il devrait dès la prochaine version être complètement intégré à Ms-Build, le processus d'actualité chez Microsoft. Parmi les avantages annoncés, on peut noter la possibilité de commander depuis l'environnement Visual C++ 10 les compilateurs des versions antérieures, afin de faciliter la transition, ou même des compilateurs autres, le processus étant ouvert.

IV-D - Divers

Voici en vrac quelques points qui ont été abordés dans la riche séance de questions réponses.

Support du C++98, et en particulier de export : Scoop : Il n'est pas impossible, que d'ici une version ou deux...

Support du C++0x dans la version 10 : La version 10 devrait sortir avant le C++0x, il est donc évident qu'elle ne pourra pas le supporter. En termes de bibliothèque, tant qu'elles ne demandent pas de points spécifiques du langage, tout est possible. En termes de langage, les lambdas pourraient être intégrés en premier point. En revanche, auto, les concepts ou les variadic templates n'auront selon toute probabilité pas le temps d'être intégrés dans la version 10, il faudra attendre la version suivante.

Installation du produit : Elle devrait être améliorée et accélérée grandement. En interne, une simple installation par xcopy est utilisée actuellement.

Stabilité des fichiers pour le contrôle de source : Les fichiers de projet devraient de part l'introduction dans Ms-Build être plus stables par rapport à une petite modification (le bug du 8.0/8,0 selon la langue de la machine et peut-être même déjà corrigé). Les fichiers de code généré (par exemple par l'éditeur de formulaires .NET ne sont pas du ressort de l'équipe C++, et celle-ci ne promet rien.

L'inspection de variables au débogage : Le système d'inspection de variable devrait être fusionné avec celui de des langages .NET, ou en tout cas être géré par la même équipe. On peut donc espérer une plus grande configurabilité de ce côté.

La génération de code machine : Le back-end ne devrait pas trop évoluer en version 10, mais en version 11, le projet **Phoenix** devrait être pleinement utilisé dans le back-end de génération de code, offrant de nouvelles possibilités en terme d'optimisation et d'ouverture de la génération de code.

1 : Cette fonction existait déjà dans 2005, mais encore instable, elle n'était pas documentée.