

Developpez

Magazine

Edition de Octobre-Novembre 2008.
Numéro 18.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Article C/C++/GTK/Qt



Pointeurs intelligents

Ce document présente un outil indispensable à l'écriture de code correct en C++ : Les pointeurs intelligents

par **Loïc Joly**
Page 20

Article Mac



Apple Event 14 Octobre 2008 : Découvrez les nouveaux portables d'Apple

Apple Event 14 Octobre 2008 : Découvrez les nouveaux portables d'Apple

par **La rédaction Mac**
Page 51

Index

Java	Page 2
PHP	Page 7
(X)HTML/CSS	Page 10
DotNet	Page 13
C/C++/GTK/Qt	Page 20
MS Office	Page 29
SGBD	Page 38
Linux	Page 47
Mac	Page 51
2D/3D/Jeux	Page 60
Liens	Page 67

Editorial

En automne les feuilles tombent, chez Developpez.com nous en profitons pour les remplir de nos meilleurs articles, critiques de livres, questions/réponses sur diverses technologies.

Profitez-en bien !

La rédaction

Les derniers tutoriels et articles

Prise en main d'Ant

Le but principal de cet article est de faire découvrir Ant aux développeurs Java qui ne connaissent pas encore cet outil. Afin de dévoiler les principaux apports d'Ant, nous allons comparer la création d'un projet en ligne de commande classique avec la création du même projet avec Ant.

1. Introduction

Ant est un projet open source de la fondation Apache écrit en Java qui vise le développement d'un logiciel d'automatisation des opérations répétitives tout au long du cycle de développement logiciel. Il est téléchargeable à l'adresse suivante <http://ant.apache.org> ([Lien1](#)).

Ant pourrait être comparé au célèbre outil make sous Unix. Il a été développé pour fournir un outil de construction indépendant de toute plate-forme. Ceci est particulièrement utile pour des projets développés sur et pour plusieurs systèmes ou pour migrer des projets d'un système vers un autre. Il est aussi très efficace pour de petits développements.

Ant repose sur un fichier de configuration XML qui décrit les différentes tâches qui devront être exécutées par l'outil. Ant fournit un certain nombre de tâches courantes qui sont codées sous forme de classes Java. Ces tâches sont donc indépendantes du système sur lequel elles seront exécutées. De plus, il est possible d'ajouter ces propres tâches en écrivant de nouvelles classes Java respectant certaines spécifications.

La popularité d'Ant augmente de jour en jour. Sa souplesse, sa puissance et sa simplicité en ont fait l'un des outils les plus populaires dans le monde de Java.

Les environnements de développement intégrés proposent souvent un outil de construction propriétaire qui son généralement moins souple et moins puissant que Ant. Ainsi des plug-ins ont été développés pour la majorité d'entre eux (JBuilder, Forte, Visual Age ...) pour leur permettre d'utiliser Ant, devenu un standard de facto.

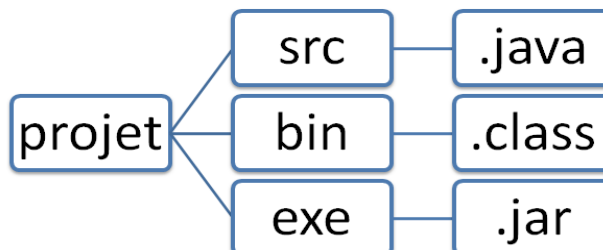
L'IDE Netbeans integre Ant depuis sa version 4.0, Il prévoit un fichier *build.xml* pour tout ses projets.

2. Création du projet par ligne de commande

Dans ce tutoriel, nous voulons séparer les sources des fichiers compilés, donc nos fichiers sources (.java) seront placés dans le répertoire *src*. Tous les fichiers (.class) générés doivent être placés dans le répertoire *bin* et enfin le fichier jar de notre projet ira dans le répertoire *exe*.

Nous avons utilisé la syntaxe de la ligne de commande de windows. Sous Linux le **md** est remplacée par **mkdir** et l'antislash (\) par un simple slash (/). Les autres commandes devraient toutes marcher sans problème.

On aura donc l'arborescence suivante :



2.1. Compilation

On crée seulement le répertoire *src*. Voici la commande le permettant :

```
md src
```

La classe java suivante affiche " Hello world " dans la sortie standard

HelloWorld.java

```
package org.sdf;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Vous devez mettre ce code dans un fichier nommé HelloWorld.java. Ce fichier doit se trouver dans le répertoire *src/org/sdf*. On aura donc ceci :



Maintenant nous allons essayer de compiler et exécuter notre projet.

```
md bin
javac -sourcepath src -d bin\
src\org\sdf\HelloWorld.java
java -cp bin org.sdf.HelloWorld
```

Qui va résulter en ceci :

```
Hello World.
```

Après la compilation du fichier source, on aura l'arborescence suivante :



2.2. Création du fichier jar exécutable

La création du fichier jar n'est pas très difficile. Mais le rendre exécutable nécessite plus d'une étape :

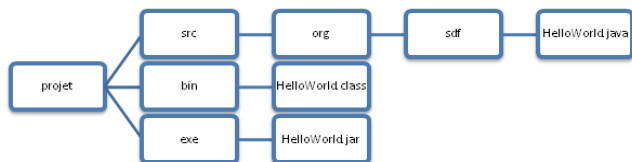
1. Création d'un fichier *manifest* contenant le nom de la class principale.
2. Création du répertoire *exe*.
3. Création du fichier jar.

Tout ceci peut être effectué en ligne de commande comme suit :

```

echo Main-Class: org.sdf.HelloWorld>MonManifest
md exe
jar cfm exe\HelloWorld.jar MonManifest -C bin .
  
```

Après l'exécution de ces trois lignes, on aura l'arborescence de fichiers suivante :



2.3. Exécution du projet

On pourra exécuter notre projet avec la ligne de commande :

```
java -jar exe\HelloWorld.jar
```

3. Création du projet avec Ant

Nous allons maintenant essayer de créer un fichier *build.xml* Ant. Pour notre projet, ce *build.xml* contiendra 4 cibles : clean, compile, jar et run.

3.1. La cible clean

Cette entrée permet de supprimer tous les fichiers générés lors de la compilation et de la création du fichier. En général, C'est une bonne habitude d'avoir une cible clean.

```

<target name="clean">
  <delete dir="bin"/>
  <delete dir="exe"/>
</target>
  
```

Notre cible clean est très simple, il appellera la tâche *delete* deux fois, une fois pour supprimer les fichiers class générés, et une autre pour supprimer le fichier jar.

3.2. La cible compile

Cette entrée permet de créer un répertoire nommé *bin* puis de compiler les fichiers sources du répertoire *src*, et de mettre les fichiers générés dans le répertoire *bin*.

```

<target name="compile">
  <mkdir dir="bin"/>
  <javac srcdir="src" destdir="bin"/>
</target>
  
```

La création du répertoire *bin* se fait avec la tâche *mkdir*. La compilation appelle la tâche *javac* tout simplement.

3.3. La cible jar

Cette entrée permet de créer un répertoire nommé *exe* et de créer le fichier jar exécutable de notre projet dans ce répertoire.

```

<target name="jar">
  <mkdir dir="exe"/>
  <jar destfile="exe/HelloWorld.jar" basedir="bin">
    <manifest>
      <attribute name="Main-Class"
        value="org.sdf.HelloWorld"/>
    </manifest>
  </jar>
</target>
  
```

La création du répertoire *exe* se fait avec la tâche *mkdir*. La création du fichier jar exécutable se fait avec la tâche *jar*. Le fichier *manifest* est créé très facilement avec l'élément *manifest*.

3.4. Exécution du projet

Cette entrée permet d'exécuter le fichier jar exécutable généré lors de l'étape précédente.

```

<target name="run">
  <java jar="exe/HelloWorld.jar" fork="true"/>
</target>
  
```

La tâche *java* permet d'exécuter le fichier jar. **Fork= "true"** permet d'exécuter le projet dans une nouvelle machine virtuelle Java.

3.5. Dépendances entre cibles

L'exécution d'une cible (*target*) peut dépendre de l'exécution d'autres cibles. Dans notre exemple, la cible *jar* dépend de la cible *compile*. Il n'est pas logique de générer un fichier Jar sans que les sources soient compilés.

Ant offre un mécanisme de dépendance qui permet d'ordonner l'exécution des cibles du fichier *build.xml*. Ce mécanisme est implémenté avec l'attribut *depends*.

Exemple :

```

<target name="jar" depends="compile">
  <mkdir dir="exe"/>
  <jar destfile="exe/HelloWorld.jar" basedir="bin">
    <manifest>
      <attribute name="Main-Class"
        value="org.sdf.HelloWorld"/>
    </manifest>
  </jar>
</target>
  
```

L'exécution de cette cible requiert l'exécution de la cible **compile**. Ant se chargera de le faire. Si la cible **compile** dépend elle aussi d'une autre cible, alors Ant se chargera d'exécuter les dépendances récursivement.

3.6. Récapitulatif

Après avoir écrit les principales cibles de notre projet, nous allons maintenant construire notre fichier *build.xml* qui sera

utilisé par Ant pour exécuter les différentes tâches. Le fichier *build.xml* doit avoir comme racine un élément appelé **project**. Cet élément peut avoir les attributs suivants :

Attribut	Description	Obligatoire
name	Le nom du projet	non
default	la cible par défaut à utiliser lorsqu'aucune cible n'est indiquée	non
basedir	le répertoire de base du projet	non

```
<project default="run">

  <target name="clean">
    <delete dir="bin"/>
    <delete dir="exe"/>
  </target>

  <target name="compile" depends="clean">
    <mkdir dir="bin"/>
    <javac srcdir="src" destdir="bin"/>
  </target>

  <target name="jar" depends="compile">
    <mkdir dir="exe"/>
    <jar destfile="exe/HelloWorld.jar" basedir="bin">
      <manifest>
        <attribute name="Main-Class"
value="org.sdf.HelloWorld"/>
      </manifest>
    </jar>
  </target>

  <target name="run" depends="jar">
    <java jar="exe/HelloWorld.jar" fork="true"/>
  </target>

</project>
```

Maintenant, on peut compiler, archiver et exécuter notre projet avec les commandes :

```
ant clean
ant compile
ant jar
ant run
```

Et puisque **run** dépend de **jar** qui dépend de **compile** qui dépend de **clean**, On peut faire tout simplement :

```
ant run
```

Ou mieux encore, puisque la cible par défaut du projet est **run** :

```
ant
```

3.7. Amélioration du fichier *build.xml*

Maintenant que nous avons notre fichier *build.xml*, on peut y effectuer quelques optimisations. On remarque qu'on a utilisé plusieurs fois les mêmes noms de répertoires. Un problème surviendrait si l'on voulait changer le répertoire où seront générés les fichiers compilés : il faudrait changer toutes les occurrences de *bin* dans le fichier.

Pour contourner ce problème, Ant offre le mécanisme de propriétés. Il s'agit de déclarer des propriétés qui ont des valeurs, et de

récupérer leur valeur n'importe où dans notre fichier *build.xml*.

De plus, Ant nous offre la possibilité de garder une trace de l'exécution du fichier *build.xml* par l'emploi de la tâche **echo**. Ceci peut s'avérer très utile dans certains cas.

```
build.xml
<project default="run">

  <property name="src.dir" value="src"/>
  <property name="bin.dir" value="bin"/>
  <property name="jar.dir" value="exe"/>
  <property name="main-class"
value="org.sdf.HelloWorld"/>

  <target name="clean">
    <delete dir="${bin.dir}"/>
    <delete dir="${jar.dir}"/>
    <echo message="nettoyage termine"/>
  </target>

  <target name="compile" depends="clean">
    <mkdir dir="${bin.dir}"/>
    <javac srcdir="${src.dir}" destdir="${bin.dir}"/>
    <echo message="compilation terminée"/>
  </target>

  <target name="jar" depends="compile">
    <mkdir dir="${jar.dir}"/>
    <jar destfile="${jar.dir}/sdf.jar" basedir="${
bin.dir}"/>
    <manifest>
      <attribute name="Main-Class" value="${main-
class}"/>
    </manifest>
  </jar>
  <echo message="Creation du fichier Jar terminée"/>
</target>

  <target name="run" depends="jar">
    <java jar="${jar.dir}/sdf.jar" fork="true"/>
  </target>

</project>
```

Maintenant, si on veut changer le répertoire où seront placés les fichiers compilés, il suffira de changer la valeur de la propriété *bin.dir*.

4. Comparaison entre build.xml et la ligne de commande

Nous avons essayé d'établir un tableau de comparaison entre la compilation, l'archivage et l'exécution en ligne de commande et avec Ant. Ceci est résumé dans le tableau suivant :

Ligne de commande	build.xml
md bin javac -sourcepath src -d bin\ src\org\sdf\HelloWorld. java	<target name="compile"> <mkdir dir="bin"/> <javac srcdir="src" destdir="bin"/> </target>
echo Main-Class: org.sdf.HelloWorld>M onManifest md exe jar cfm exe\HelloWorld.jar MonManifest -C bin .	<target name="jar"> <mkdir dir="exe"/> <jar destfile="exe/HelloWorld.jar " basedir="bin"> <manifest> <attribute name="Main- Class" value="org.sdf.HelloWorld"/ > </manifest> </jar> </target>
java -jar exe\HelloWorld.jar	<target name="run"> <java jar="exe/HelloWorld.jar" fork="true"/> </target>

5. Autres outils

5.1. Maven

Apache Maven est un outil logiciel libre pour la gestion et l'automatisation de production des projets logiciel Java. L'objectif recherché est comparable au système Make sous Unix : produire un logiciel à partir de ses sources, en optimisant les tâches réalisées à cette fin et en garantissant le bon ordre de fabrication.

Il est semblable à l'outil Ant, mais fournit des moyens de configuration plus simples, eux aussi basés sur le format XML. Maven est géré par l'organisation Apache Software Foundation. Précédemment Maven était une branche de l'organisation Jakarta Project.

6. Liens utiles

Introduction à ANT – partie 1 ([Lien2](#))

Introduction à ANT – partie 2 ([Lien3](#))

Developpons en Java : Ant ([Lien4](#))

7. Téléchargement de sources

télécharger ([Lien5](#))

8. Conclusion

En les structurant dans un fichier XML, Ant offre une plus grande lisibilité des opérations qu'on peut exécuter sur un projet. L'intérêt principal est qu'on ne doit plus exécuter les tâches routinières (ennuyantes !!) de compilation, d'archivage et d'exécution à chaque fois qu'on modifie une ligne dans un fichier source du projet : Ant se chargera de faire tout cela pour nous..

Retrouvez l'article de Selim Kebir en ligne : [Lien6](#)

Les livres Java

Harnessing Hibernate

This guide is an ideal introduction to Hibernate, the framework that lets Java developers work with information from a relational database easily and efficiently. Databases are a very different world than Java objects, and with Hibernate, bridging them is significantly easier. This new edition lets you explore the system, from download and configuration through a series of projects that demonstrate how to accomplish a variety of practical goals.

Critique du livre par Pierre Chauvin

Habitué à d'autres formes de persistance et ORM avec le langage Java, Harnessing Hibernate est ma première lecture sur Hibernate. Les chapitres de ce livre gravitent autour d'un **projet central** (le traditionnel outil de gestion de collection de musique), enrichi au fur et à mesure par les fonctionnalités d'Hibernate. Installation des outils (Maven, Ant), mappings et requêtes simples, requêtes Criteria, annotations, intégration avec d'autres bases de données (HSQLDB, MySQL) et outils (Spring, Stripes, IDE Eclipse) : une **vision assez globale** est donnée et vous serez en mesure de construire vos premiers projets avec une couche de persistance reposant sur Hibernate, et en exploitant les primitives simples.

Néanmoins je trouve que l'auteur **consacre beaucoup trop de temps aux outils annexes** comme Maven (le chapitre 12!), qui nécessite à lui seul un livre, et j'aurais préféré qu'il passe

davantage de temps sur des mappings plus complexes, ou encore à la gestion de cache, la gestion des sessions Hibernate. Il demeure bien construit pour un lecteur débutant, mais aura du mal à satisfaire celui-ci dès l'apparition des premiers besoins atypiques et plus complexes. Utile, mais loin d'être indispensable.

Head First Servlets and JSP

Looking to study up for the new J2EE 1.5 Sun Certified Web Component Developer (SCWCD) exam?

This book will get you way up to speed on the technology you'll know it so well, in fact, that you can pass the brand new J2EE 1.5 exam. If that's what you want to do, that is. Maybe you don't care about the exam, but need to use servlets and JSPs in your next project. You're working on a deadline. You're over the legal limit for caffeine. You can't waste your time with a book that makes sense only AFTER you're an expert (or worse, one that puts you to sleep).

Learn how to write servlets and JSPs, what makes a web container tick (and what ticks it off), how to use JSP's Expression Language (EL for short), and how to write deployment descriptors for your web applications. Master the c:out tag, and get a handle on exactly what's changed since the older J2EE 1.4 exam. You don't just pass

the new J2EE 1.5 SCWCD exam, you'll understand this stuff and put it to work immediately.

Head First Servlets and JSP doesn't just give you a bunch of facts to memorize; it drives knowledge straight into your brain. You'll interact with servlets and JSPs in ways that help you learn quickly and deeply. And when you're through with the book, you can take a brand-new mock exam, created specifically to simulate the real test-taking experience.

Critique du livre par Eric Reboisson

Comme l'explique la description du livre, sans Java côté serveur, eBay n'existerait pas...et donc pas possible d'acheter le poster de Farrah Fawcett qui vous plait tant. Voilà l'ambiance dans laquelle vous baignerez à la lecture de "Head First Servlets and JSP" : un contenu décalé (des images et dialogues amusants, etc.) par rapport aux ouvrages informatiques traditionnels, mais qui cache une pédagogie bien étudiée et qui fonctionne (je pense que tous les lecteurs de cette collection confirmeront).

Le but de ce livre est non seulement d'expliquer les techniques de développement orienté web avec Java (Servlets, JSP, taglibs...)

mais surtout une préparation à la certification Sun Certified Web Component Developer Exam (SCWCD) dans sa version pour J2EE 1.5 : chaque chapitre est ainsi ponctué d'un mini examen, et à la fin du livre un dernier pour vous tester véritablement.

Les trois auteurs sont très qualifiés sur le sujet puisque Bryan Basham a participé à la conception de la SCWCD, Bert Bates quant à lui, a oeuvré pour la Sun Certified Business Component Developer et enfin Kathy Sierra est fondatrice de javaranch.com et a participé également à la conception de la SCJP. Que du beau monde donc !

Et les trois auteurs se focalisent uniquement sur les sujets de la certification, vous garantissant la réussite de la certification avec tous les conseils prodigués dans l'ouvrage.

Je ne vais pas trop m'étendre et juste résumer la qualité de l'ouvrage en donnant la note maximale pour un livre qui vous apprendra tous les rudiments du développement web en Java, ou vous aidera à réviser pour passer la SCWCD. Un livre à acheter les yeux fermés !

Retrouvez ces critiques de livre sur la page livres Java : [Lien7](#)

Les derniers tutoriels et articles

APC : un cache d'OPCodes pour PHP

Le cache d'OPCodes permet des économies de traitements divers dans le coeur de PHP. Il augmente la vitesse générale de traitement d'une requête par PHP, et il est souvent la solution d'optimisation la plus simple à mettre en place. Cet article va expliquer comment installer, configurer et gérer un cache OPCodes pour PHP : Alternative PHP Cache (APC).

1. Introduction à APC

Les caches d'OPCodes permettent des économies de travail coté serveur, ce qui se manifeste par une accélération de la réponse des pages PHP pour l'utilisateur final.

Il représente en plus une des solutions les plus simples à mettre en place dans ce but.

APC est "l'Alternative PHP Cache", un cache libre, gratuit et robuste pour mettre en cache et optimiser le code intermédiaire PHP aussi appelé "OPCode".

APC est activement maintenu dans PECL ([Lien8](#)) et offre non seulement un cache OPCode mais aussi un cache utilisateur. Il est configurable et facilement installable.

D'autres solutions de cache existent, vous les trouverez en conclusion. Cet article se base sur une installation de PHP en environnement Apache module (mod_php).

1.1. Installation et configuration

L'installation se fait grâce à la commande `pecl install apc`. Vous pouvez aussi l'installer depuis ses sources ([Lien9](#)) avec les commandes `phpize`, `./configure`, `make` et `make install`.

Dans les sources se trouve aussi un fichier appelé `apc.php`. Ce fichier est à installer sur le serveur et à interroger comme une vulgaire page PHP. Il s'agit du fichier qui permet le monitoring du cache ([Lien11](#)).

Après l'installation du cache, un appel à `phpinfo()` devrait confirmer que celui-ci est bien installé.

APC rajoute des options dans votre fichier `php.ini`, nous les passerons en détail plus tard, assurez vous simplement que celle appelée `apc.enabled` est bien sur la valeur **On**.

Vous pouvez dès lors vérifier le gain de performances sur une application avec un outil comme **ab** (Apache Bench). La directive intéressante à monitorer sera le nombre de requêtes par seconde qu'Apache peut servir. En théorie, ce nombre devrait augmenter de manière plus ou moins significative en fonction de l'application concernée et de l'OS utilisé. Nous allons voir comment fonctionne le cache d'OPCode afin de mieux comprendre ce que l'on est en droit d'attendre de lui.

apc

APC Support	enabled
Version	3.1.0-dev
MMAP Support	Disabled
Locking type	File Locks
Revision	\$Revision: 3.151 \$
Build Date	Nov 8 2007 15:17:30

2. Comment fonctionne un cache d'OPCodes ?

2.1. En quelques mots

Une partie importante du temps de traitement d'une page, servie via HTTP avec un langage de script dynamique comme PHP, provient du fait que ce langage doit transformer un code lisible par un humain, en un code exécutable par une machine virtuelle et le microprocesseur de la machine physique.

Des processus lourds interviennent alors, comme l'analyse syntaxique du code source et sa transformation en un code binaire compréhensible par la machine.

Ceci étant coûteux, beaucoup de langages mettent ce code binaire en cache lors du traitement de la première requête, afin de sauter toutes ces étapes les fois suivantes.

APC fonctionne par interception de la phase de compilation en la remplaçant par une étape de récupération de l'OPCode déjà généré précédemment et mis en mémoire. On peut voir ceci comme un compilateur de code intelligent.

2.2. de l'OPCode ?

Les OPCodes sont des structures de données C qui peuvent être interprétées par la machine virtuelle PHP : Zend Engine. L'extension PHP Vulcan Logic Disassembler (VLD) ([Lien12](#)) permet de prendre connaissance de ce code intermédiaire. Elle est utilisée par les développeurs de PHP afin d'optimiser leurs algorithmes et a été créée par l'un d'entre eux : Derick Rethans.

code source PHP

```
<?php
$output = 'Hello World';
if($_GET['exclaim']) {
    $output .= '!';
} else {
    $output .= '.';
}
echo $output;
?>
```

OPCode PHP

```
1 ASSIGN          !0, 'HELLO+WORLD'
2 FETCH_R GLOBAL $1, '_GET'
3 FETCH_DIM_R     $2, $1, 'exclaim'
4 JMPZ           $2, ->6
5 ASSIGN_CONCAT  !0, '%21'
6 JMP            ->7
7 ASSIGN_CONCAT  !0, '.'
8 ECHO           $0
9 RETURN         1
10 ZEND_HANDLE_EXCEPTION
```

Comme on peut le voir, les OPCodes utilisent toujours plus de place que le code PHP, environ 25 à 33%. Cette remarque est

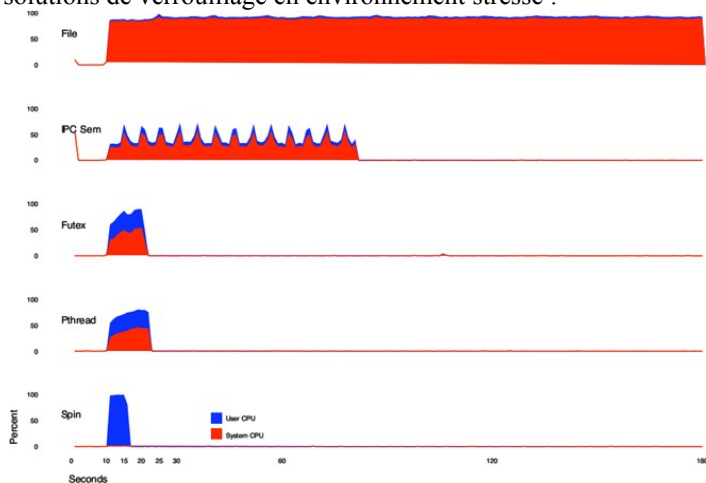
importante lorsqu'il s'agira de configurer le cache, car sa taille sera alors à calculer pour le code intermédiaire et non le code PHP.

2.3. Fonctionnement technique avancé

APC stocke les opcodes dans un segment de mémoire partagée pour accéder rapidement aux données. Cette mémoire étant partagée entre les processus du serveur, un mécanisme de verrou doit être mis en place afin de s'assurer de l'exclusivité des accès modifiant les données.

Ceci crée un goulot d'étranglement de performances important. Les anciennes versions d'APC utilisaient un verrouillage au niveau des fichiers, ce qui était alors très fiable, mais avec un coût en performances d'une importance extrême.

Les versions récentes d'APC utilisent les verrous pthread qui sont standards sur la plupart des matériels. D'autres solutions de verrouillage sont aussi disponibles comme l'Inter-Process Control (IPC), les verrous Linux Futex et plus récemment, les verrous tournants (spin locks) en provenance du projet PostgreSQL. Voici un graphique qui représente un benchmark des différentes solutions de verrouillage en environnement stressé :



Comme le système pthread est très utilisé dans d'autres projets, il se place comme une solution fiable et choisie dans les versions actuelles d'APC comme défaut.

Les verrous tournants ont été portés du projet PostgreSQL et peuvent donner de bien meilleurs résultats que le système pthread. Cependant leur support dans APC demeure expérimental étant donné des problèmes de verrouillage rencontrés avec des signaux d'exécution de PHP. Une solution est en cours d'élaboration, elle devrait proposer un support stable des verrous tournants dans le futur.

2.3.1. Une table géante

APC peut être vu comme une table de partage géante en mémoire, entre tous les processus du serveur web. Comme toutes les tables mémoire, ses performances proviennent directement de la gestion des collisions. La taille de la table devrait être proportionnelle au nombre d'éléments à y stocker.

APC propose pour cela 3 directives de configuration :

1. `apc.shm.size` : Taille du cache en Mo.
2. `apc.num_files_hint` : La valeur du nombre maximum de fichiers à cacher est conseillée.
3. `apc.user_entries_hint` : La valeur du nombre maximum d'entrées utilisateur (cache utilisateur) à stocker est conseillée.

Il n'est pas conseillé de tenter de leurrer APC en doublant (par exemple) ces valeurs. APC double déjà en interne les valeurs spécifiées, et le gain en performances est fortement réduit avec des valeurs non adaptées.

2.3.2. Stat ou pas stat ?

Contrairement à la mémoire vive, les disques durs sont extrêmement lents, et devraient être évités le plus possible dans les solutions de cache. Malgré le fait qu'APC charge les Opcodes depuis la mémoire directement, il accède au disque pour être certain que le fichier source n'a pas été modifié, ce qui déclencherait une procédure de mise à jour du code en cache. Sans cette vérification, un changement du fichier source n'aurait aucun effet sur le serveur. Si cependant, la mise à jour des fichiers sources est rare, l'étape de vérification sur disque (stat) peut être évitée. Ceci rend le serveur presque indépendant du disque dur physique. L'option `apc.stat` gère ce paramètre, mise à false, elle évite tout appel disque. Ceci évite l'appel disque, ainsi que l'appel système nécessaire. Le gain en performances devient important pour les applications possédant un nombre de fichiers sources significatif (Frameworks).

Au contraire, certaines applications nécessitent des mises à jour fréquentes des sources. Cela se fait, en général, au travers d'outils tels que `rsync`, `cvs` ou `svn`. Certains de ces outils altèrent la date de modification des fichiers, vers une date passée. Si c'est le cas, `apc.stat_time` doit être mis à true pour forcer APC à se baser sur les dates de création des fichiers, et non plus sur les dates de modification.

2.3.3. TTL

APC propose deux configurations du TTL (Time To Live) : une configuration utilisateur, et une configuration par défaut dite de secours. Le TTL limite le temps pendant lequel une valeur demeure en cache, ce mécanisme permet de s'assurer qu'aucune donnée périmée ne sera chargée du cache alors qu'une donnée plus fraîche existe.

`apc.gc_ttl` contrôle le TTL du garbage collector. Comme de multiples processus peuvent accéder en même temps à une donnée, celle-ci ne doit être détruite que lorsque tous les processus l'ont relâchée. Si un processus meurt sans relâcher sa référence à la donnée, alors APC ne pourra jamais déterminer si celle-ci est périmée ou valide. `gc_ttl` est utilisé pour contrôler combien de temps APC doit attendre avant d'effacer une donnée, même si celle-ci demeure accédée par un ou plusieurs processus. Typiquement, on spécifie une valeur haute de manière à être sûr de ne pas effacer une donnée du cache toujours valide.

2.3.4. Remplissage du cache au démarrage

Le cache APC est vidé à chaque (re)démarrage du serveur, ainsi les performances seront impactées par son remplissage lors des premières requêtes. APC ne donne des résultats pertinents que lorsqu'il est bien rempli.

Remplir le cache au démarrage est une technique qui contourne ce problème en insérant les données (c'est-à-dire les variables, et les fichiers PHP) dans le cache avant qu'il ne traite la première requête HTTP via le serveur.

La commande `iptables` sous Linux est utile dans ce but. Il faut bien se rendre compte que le remplissage doit être fait via un processus du serveur Web, et non un client PHP quelconque. Ainsi, par exemple, un script PHP utilisant `compile_file()` et `apc_store()` peut être placé dans un espace restreint sur le serveur. Avant le démarrage du serveur, `iptables` peut être utilisé pour détourner toutes les IP sauf 127.0.0.1 (par exemple) du serveur web.

Le serveur est ensuite démarré et une commande telle que `curl` peut alors lui être envoyée vers la page spéciale qui se chargera de compiler et de mettre en cache toutes les données nécessaires. Enfin, `iptables` lève le blocage du serveur, qui est alors prêt à encaisser pleinement sa première vague de requête, avec un cache déjà plein.

2.3.5. Filtrage

Dans certains cas, il peut y avoir des fichiers qui ne doivent pas être mis en cache. Ceci peut venir du fait de leur grosse taille, ou de leurs fréquentes modifications. Aussi, on peut vouloir ne plus mettre en cache certains fichiers qui s'avèrent buggés avec le cache (ceci peut arriver des inclusions conditionnelles par exemple).

1. `apc.max_file_size` : Restriction sur la taille maximum d'un fichier à cacher. Ceci peut créer des problèmes de performance lors de la demande en cache de ces fichiers-là, mais si la mémoire est limitée, l'option demeure intéressante
2. `apc.filters` : Expression régulière utilisée pour exclure tout fichier dont le nom a une correspondance.
3. `apc.cache_by_default` : Inverse la tendance de `apc.filters` : APC cachera uniquement les fichiers qui ont une correspondance sur le filtre, au lieu de les ignorer.

2.3.6. RFC1867, progression de l'upload de fichier

PHP 5.2.0 a apporté le support de la RFC1867, la progression de l'upload de fichier. Cette option permet à APC ou à une autre extension, d'être tenu au courant d'informations concernant l'upload d'un fichier. Javascript peut alors demander au serveur des détails sur l'upload en cours, et celui-ci lui renverra ces informations.

APC supporte ce mécanisme en insérant et en gardant à jour des variables utilisateur avec des détails concernant l'upload.

1. `apc.rfc1867` : option booléenne d'activation/désactivation du support de la RFC1867
2. `apc.rfc1867_freq` : Fréquence de mise à jour en octets ou pourcentage du fichier. Les variables APC doivent être mises à jour fréquemment pour fournir des infos pertinentes, mais pas trop souvent au risque d'impacter les performances.
3. `apc.rfc1867_name` : Le nom du champ caché HTML utilisé pour générer le nom de la variable APC. Cette valeur doit être unique et peut être utilisée dans les requêtes suivantes pour obtenir des détails concernant l'avancement de l'upload.
4. `apc.rfc1867_prefix` : Préfixe utilisé pour le nom de la variable. Ceci permet d'assurer l'unicité des noms de variables et ainsi pouvoir les reconnaître.

Un formulaire compatible avec les uploads rfc1867 ressemble à ceci :

```
<form action="/upload.php" method="post"
enctype="multipart/form-data" />
<input type="file" name="myfile" />
<input type="hidden" name="APC_UPLOAD_PROGRESS"
value="A86DCF0C">
<input type="submit" value="upload" />
</form>
```

La variable APC contenant l'information serait alors A86DCF0C préfixée par la valeur de `apc.rfc1867_prefix` (dont la valeur par défaut est `'upload_'`).

3. API de APC et cache utilisateur

En plus de permettre la mise en cache de l'OPCode issu des sources des fichiers PHP, APC offre aussi un cache dit utilisateur. L'extension APC rajoute en effet des fonctions au langage PHP, qui vont permettre de stocker dans le cache utilisateur des variables PHP.

En général ces variables contiennent des informations lourdes à calculer, comme par exemple des résultats de base de données, ou d'autres traitement coûteux, qui pourront alors être chargés depuis le cache directement. Sauf pour les objets, il n'est pas nécessaire

de sérialiser ces données avant de les stocker dans APC. Les ressources, elles, ne sont pas stockables en cache en raison de leur intimité et unicité vis-à-vis d'une requête HTTP.

En prenant des précautions comme avec les sessions sur plusieurs serveurs (session clustering), le cache utilisateur est très utile pour stocker tout ce qui touche tous les utilisateurs en même temps.

Nous allons passer quelques fonctions APC en revue, pour le reste, la documentation officielle ([Lien13](#)) est présente.

infos

1. `array apc_cache_info` ([string \$cache_type [, bool \$limited]])
2. `array apc_sma_info` ([bool \$limited])

Insertion

1. `bool apc_store` (string \$key , mixed \$var [, int \$ttl])
2. `bool apc_add` (string \$key , mixed \$var [, int \$ttl])
3. `bool apc_compile_file` (string \$filename)

Récupération

1. `mixed apc_fetch` (string \$key)

Effacement

1. `bool apc_delete` (string \$key)
2. `bool apc_clear_cache` ([string \$cache_type])

Le Zend Framework ([lien14](#)) propose une classe permettant de gérer le cache utilisateur dans `Zend_Cache`

Grâce à des fonctions comme `apc_compile_file()`, il est possible de gérer facilement la mise à jour d'un site possédant plusieurs serveurs. Plutôt que de migrer le code sous forme de révision ou de fichier, on peut demander conditionnellement aux différents serveurs de compiler la nouvelle révision. Si la fonctionnalité ajoutée doit pouvoir être annulée facilement et rapidement, c'est encore mieux.

Il faut en effet faire attention aux sites ayant plusieurs serveurs, que les caches de chacun d'entre eux soient bien mis à jour en même temps, sinon des utilisateurs risquent de se retrouver aléatoirement avec des portions anciennes du site.

4. Conclusions

Les caches d'OPCode sont vraiment géniaux pour PHP, et doivent être installés sur tous les serveurs nécessitant des performances et une haute scalabilité ([Lien15](#)).

Cet article d'introduction peut être complété par le manuel de PHP et de l'extension APC ([Lien16](#)).

En plus de moi-même (Brian), APC est maintenu par une quantité de contributeurs, notamment George Schlossnagle, Daniel Cowgill, Rasmus Lerdorf, Gopal Vijayaraghavan, Edin Kadribasic, Ilia Alshanetsky, Marcus Börger, Sara Golemon. Je (Brian) suis très heureux du travail qu'ils accomplissent, les patches et corrections de bugs, que ce soit pour APC, PHP ou d'autres projets.

Voici une liste (non exhaustive) des autres possibilités de cache d'OPCodes en PHP :

1. IonCube: http://www.ioncube.com/sa_encoder.php ([Lien17](#))
2. Zend Platform: <http://www.zend.com/en/products/platform/> ([Lien18](#))
3. Xcache: <http://xcache.lighttpd.net/> ([Lien19](#))
4. Eaccelerator: <http://www.eaccelerator.net/> ([Lien20](#))

Cet article est inspiré de la conférence "APC @ FaceBook" ([Lien21](#)).

Retrouvez l'article de Brian Shire traduit par Julien Pauli en ligne : [Lien22](#)

Créer des vignettes redimensionnables

Ce tutoriel aborde le contrôle de la taille des vignettes qui figurent sur votre site web. Il nous arrive parfois de ne pas avoir suffisamment d'espace pour intégrer des vignettes de grande taille et pourtant on souhaite éviter les petites images à peine reconnaissables. En utilisant cette astuce on peut modifier les dimensions de base de la vignette, pour l'afficher en grand au survol de la souris.

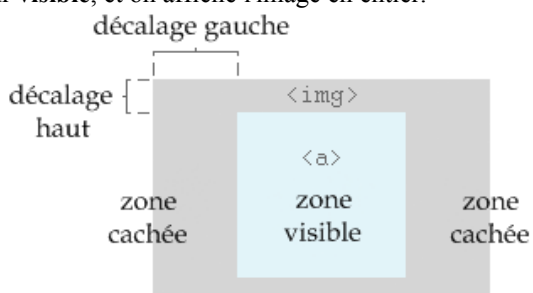
1. Vue d'ensemble

On n'effectue pas véritablement un redimensionnement de l'image en tant que tel. Il s'agit d'un redimensionnement spécifique à la zone de la vignette survolée par la souris. Comment faisons-nous cela ? En utilisant la propriété CSS **overflow** !

La propriété **overflow** détermine l'apparence du contenu lors du survol de la zone contenant. Si le conteneur a une taille fixe, pour une raison ou une autre, on utilise alors **overflow** pour définir ce qu'il se produit. Les valeurs possibles pour la propriété **overflow** sont : **visible**, **hidden**, **scroll** et **auto**. C'est la combinaison de ces valeurs que nous utiliserons ici afin que notre astuce puisse marcher. Fondamentalement, nous cachons une partie de la vignette dans son état de base, et nous la montrons entièrement lors du survol de la souris.

2. Le Principe

L'idée qui se cache derrière tout ceci est de placer une image dans un certain conteneur. Étant donné que l'on aborde les vignettes, ce conteneur sera une balise `<a>`. On définit la taille (largeur et hauteur) du conteneur avec les valeurs souhaitées et on met la propriété CSS **position** du conteneur à **relative**. L'image incluse a une position absolue. On utilise des valeurs négatives à haut et à gauche pour déplacer l'image. Le conteneur voit sa propriété **overflow** mise à **hidden**. Ainsi il n'y aura qu'une seule partie visible de l'image dans sa zone. Le reste sera caché, donc **hidden**. Pour **a:hover**, on met à la propriété **overflow** du conteneur la valeur **visible**, et on affiche l'image en entier.



Disposition des éléments sur la page

3. Le Code

Cette astuce peut être utilisée pour les listes de vignettes ou une seule, comme indiqué sur la page de demo ([Lien23](#)). Pour le code, on utilise des balises standards :

```
<a href="#"></a>
```

L'état initial des vignettes pourrait se présenter comme suit :

```
ul#thumbs a
{
    display      :   block;
    float        :   left;
    width        :  100px;
    height       :  100px;
    line-height  :  100px;
    overflow     :   hidden;
    position     :  relative;
    z-index     :   1;
}

ul#thumbs a img
{
    float        :   left;
    position     :  absolute;
    top          :  -20px;
    left         :  -50px;
}
```

La balise `<a>` a une certaine largeur et hauteur pour les éléments qui entrent en compte dans la conception de votre site. De même, **overflow** est mis à **hidden**. On manipule par la suite les valeurs négatives en haut et à gauche pour "rognier" l'image et la mettre à la taille souhaitée. Si vous voulez aller plus loin, vous pouvez définir une zone de rognage pour chaque image de votre liste et cibler précisément la zone que vous voulez montrer.

```
ul#thumbs a img
{
    float        :   left;
    position     :  absolute;
    top          :  -20px;
    left         :  -50px;
}

ul#thumbs li#image1 a img
{
    top          :  -28px;
    left         :  -55px;
}

ul#thumbs li#image2 a img
{
    top          :  -18px;
    left         :  -48px;
}

ul#thumbs li#image3 a img
{
    top          :  -21px;
    left         :  -30px;
}
```

Dorénavant, lorsque l'utilisateur survole l'image avec la souris on met **overflow** à **visible** :

```
ul#thumbs a:hover
{
    overflow      : visible;
    z-index      : 1000;
    border       : none;
```

Notez l'utilisation de z-index pour le conteneur par défaut et celui survolé. Cela est extrêmement important étant donné que nous voulons placer l'image survolée au-dessus des autres. Autrement elle se retrouverait dessous et notre astuce serait incomplète.

Retrouvez l'article de CSS Globe en ligne : [Lien24](#)

Introduction au CSS3 - Partie 1 : Qu'est-ce que c'est ?

Cet article marque le début d'une série qui fournit une introduction au nouveau standard CSS3, qui est conçu pour remplacer le CSS2. Nous commencerons par le tout début, comme si vous n'aviez jamais entendu parler du CSS3, pour vous amener à vous sentir prêt à le mettre en œuvre.

1. Qu'est-ce que c'est ?

Le CSS3 offre une immense variété de nouvelles façons de modifier le design de vos sites web, sans pour autant impliquer de grandes modifications. Ce premier tutoriel vous donnera une introduction très basique aux nouvelles possibilités créées par le standard.

2. Les modules

Le développement de CSS3 va être divisé en 'modules'. La vieille spécification était simplement trop longue et complexe pour être mise à jour d'un seul tenant, alors elle a été séparée en morceaux plus petits - avec quelques nouveaux ajouts. Certains de ces modules incluent :

- Le modèle de boîtes ;
- Le module de listes ;
- La présentation des hyperliens ;
- Le module vocal ;
- Les arrières plan et les Bords ;
- Les effets de texte ;
- La mise en page multi-colonnes.

3. Le développement

Beaucoup de nouveaux modules sont maintenant terminés, dont le SVG (Scalable Vector Graphics : pour le dessin vectoriel), Media Queries (recherche dans les médias) et les Namespaces (espaces de nommage). Les autres sont toujours en écriture.

Il est difficile de donner une date à partir de laquelle les

navigateurs web adopteront les nouvelles fonctionnalités du CSS3 - quelques nouvelles versions de Safari ont déjà commencé à le faire.

Les nouvelles fonctionnalités seront implémentées progressivement dans les différents navigateurs et il peut se passer un à deux ans avant que tous les modules soient entièrement adoptés.

4. Comment le CSS3 m'affecte-t-il?

Heureusement, d'une façon globalement positive, le CSS3 bénéficiera d'une rétro-compatibilité complète, si bien qu'il ne sera pas nécessaire de changer les designs existants pour s'assurer qu'ils marchent - les navigateurs web continueront toujours à supporter le CSS2.

Le principal impact sera la possibilité d'utiliser les nouveaux sélecteurs et propriétés qui sont disponibles. Ceux-ci vous permettront à la fois d'utiliser les nouvelles fonctions de design (animations ou dégradés par exemple), et d'utiliser les fonctions actuelles de design de manière bien plus aisée (par exemple en utilisant les colonnes).

Les futurs articles de cette série se focaliseront chacun sur un module différent de la spécification CSS3, et les nouvelles fonctionnalités qu'ils apporteront. Le prochain parle des bordures CSS3.

Retrouvez l'article de Design Shack traduit par Mickael Ruau en ligne : [Lien25](#)

Introduction au CSS3 - Partie 2 : Les bordures

Pour la deuxième partie de cette série d'articles sur le CSS3, nous allons nous pencher sur les bordures. Toute personne ayant utilisé le CSS connaît les propriétés de bordure. C'est un excellent moyen de structurer le contenu, créer des effets autour des images et d'améliorer la mise en page.

1. Introduction

Les bordures, avec le CSS3, atteignent un niveau supérieur offrant la possibilité d'utiliser des dégradés, des coins arrondis, des ombres et des bordures d'images. Nous allons chercher à en apprendre plus sur le sujet en utilisant des exemples dans la mesure du possible.

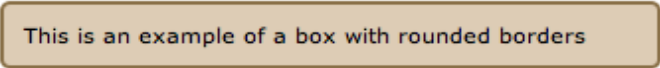
Tous les exemples montrés ci-dessous peuvent être vus sur notre page d'exemples. Cependant, beaucoup d'entre eux ne sont pas compatibles avec les dernières versions de certains navigateurs.

Voir la page d'exemples ([Lien26](#))

2. Les bordures arrondies

Réaliser des bordures arrondies en utilisant l'actuel CSS n'est pas chose facile. Toutefois, il existe de nombreuses méthodes disponibles, mais aucune n'est vraiment simple. De plus, la création d'images pour chaque bordure est souvent nécessaire. En utilisant le CSS3, la création d'une bordure arrondie devient incroyablement facile. Elle peut être appliquée à tous les coins ou individuellement et sa largeur/couleur est facilement modifiable. Le code CSS est :

```
.border_rounded
{
    background-color: #ddccb5;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    border: 2px solid #897048;
    padding: 10px;
    width: 310px;
}
```




Exemple de bordures arrondies avec le CSS3

3. Les dégradés

Les bordures dégradées peuvent s'avérer efficaces si elles sont correctement utilisées. Ce code est un peu plus complexe, vous obligeant à définir chaque couleur du dégradé. Le code CSS est :

```
.border_gradient
{
    border: 8px solid #000;
    -moz-border-bottom-colors: #897048 #917953 #a18a66
    #b6a488 #c5b59b #d4c5ae #e2d6c4 #eae1d2;
    -moz-border-top-colors: #897048 #917953 #a18a66
    #b6a488 #c5b59b #d4c5ae #e2d6c4 #eae1d2;
    -moz-border-left-colors: #897048 #917953 #a18a66
    #b6a488 #c5b59b #d4c5ae #e2d6c4 #eae1d2;
    -moz-border-right-colors: #897048 #917953 #a18a66
    #b6a488 #c5b59b #d4c5ae #e2d6c4 #eae1d2;
    padding: 5px 5px 5px 15px;
    width: 300px;
}
```

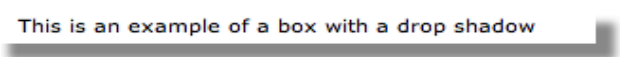


Exemple de bordures dégradées avec le CSS3

4. Les boîtes à ombre portée

A l'heure actuelle, il est difficile d'ajouter une ombre à un élément. C'est un bon moyen pour mettre une certaine zone en évidence, mais comme n'importe quel effet il doit être utilisé avec parcimonie. Le code CSS est :

```
.border_shadow
{
    -webkit-box-shadow: 10px 10px 5px #888;
    padding: 5px 5px 5px 15px;
    width: 300px;
}
```

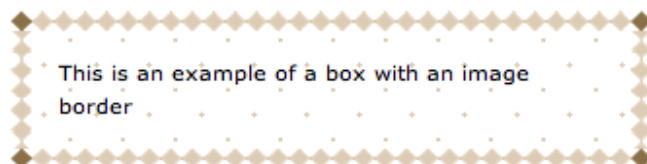


Exemple de boîtes avec effet d'ombre avec le CSS3

5. Les bordures en images

La bordure en images est un des ajouts les plus utiles. Je suis vraiment impatient de découvrir comment les gens choisiront de les utiliser. Le CSS a la capacité de répéter, ou étirer une bordure d'image de votre choix. Le code CSS est similaire à ce qui suit (il varie selon les différents navigateurs) :

```
.border_image
{
    -webkit-border-image: url(border.png) 27 27 27 27
    round round;
}
```



Exemple de bordures avec image avec le CSS3

6. En conclusion

Les bordures sont révolutionnées ! Ces ajouts dans le CSS3 ont été faits pour vous faire gagner énormément de temps en tant que designer. C'est un grand pas vers la simplification des mises en pages et ils vous permettront de créer des boîtes attrayantes sans même utiliser Photoshop.

Le prochain article de cette série parlera d'un nouveau domaine dans le CSS3 : les effets de texte.

Retrouvez l'article de Design Shack traduit par Alban Lelasseux en ligne : [Lien27](#)

Les derniers tutoriels et articles

Le Data Binding en Silverlight 2

Cet article vous présentera le Data Binding en Silverlight 2.

1. Data Binding ?

Le Data Binding est simplement le moyen utilisé pour faire le lien entre votre interface et votre source de données.

Ce concept marque encore plus la séparation entre les différentes couches de votre application (ici entre la couche présentation et les autres).

On appelle l'interface *la cible* et le fournisseur de données, *la source*.

2. Le Data Binding en Silverlight 2

2.1. Comment faire

Pour illustrer cet article, nous allons créer un petite application de gestion de tutoriels.

Nous allons d'abord commencer par créer notre classe **Tutorial** :

```
public class Tutorial
{
    public string Title { get; set; }
    public string Author { get; set; }
    public string Url { get; set; }
    public int Rate { get; set; }
}
```

Un tutoriel a un titre, un auteur, une url et une note. Rien de bien compliqué.

Maintenant que nous avons notre objet métier, nous allons créer notre interface (avec simplement des **TextBlock**) :

```
<Grid x:Name="LayoutRoot" Background="White">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="75" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="25" />
        <RowDefinition Height="25" />
        <RowDefinition Height="25" />
        <RowDefinition Height="25" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <TextBlock Grid.Row="0" Text="Titre : " />
    <TextBlock Grid.Row="1" Text="Auteur : " />
    <TextBlock Grid.Row="2" Text="URL : " />
    <TextBlock Grid.Row="3" Text="Note : " />
    <TextBlock x:Name="TxtTitle" Grid.Row="0"
        Grid.Column="1" Text="{Binding Title}" />
    <TextBlock x:Name="TxtAuthor" Grid.Row="1"
        Grid.Column="1" Text="{Binding Author}" />
    <TextBlock x:Name="TxtUrl" Grid.Row="2"
        Grid.Column="1" Text="{Binding Url}" />
    <TextBlock x:Name="TxtRate" Grid.Row="3"
        Grid.Column="1" Text="{Binding Rate}" />
</Grid>
```

Et voilà. Vous apercevez par la même occasion la syntaxe à utiliser pour Binder une propriété de notre source, à une propriété de notre cible (ici des **TextBlock**) :

```
Property="{Binding PropertyName}"
```

Maintenant il faut dire à nos **TextBlock** qui est notre source. Nous allons d'abord créer un tutorial.

Dans la méthode **InitializeComponent** de notre *Page.xaml*.

```
Tutorial templates = new Tutorial()
{
    Author = "Benjamin Roux",
    Title = "Templates avancés en Silverlight",
    Url = "http://broux.developpez.com/articles/csharp/
templates-silverlight/",
    Rate = 10
};
```

Maintenant nous allons spécifier pour chacun de nos **TextBlock**, notre objet :

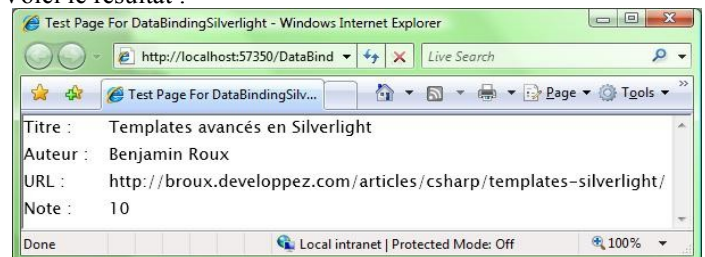
```
TxtAuthor.DataContext = templates;
TxtRate.DataContext = templates;
TxtTitle.DataContext = templates;
TxtUrl.DataContext = templates;
```

Rébarbatif non ?

On peut factoriser tout ça, en spécifiant le *DataContext* à l'élément qui englobe tous nos **TextBlock**, c'est-à-dire notre **Grid** qui se nomme *LayoutRoot* :

```
LayoutRoot.DataContext = templates;
```

Voici le résultat :



A noter qu'il existe une autre manière de spécifier le Binding, cette fois-ci par du code C#. Au lieu de faire par exemple

```
Text="{Binding Title}"
```

Il est possible de faire dans le code behind :

```
TxtTitle.SetBinding(TextBlock.TextProperty, new
System.Windows.Data.Binding("Title"));
```

Voilà pour la version basique. Vous avez sûrement remarqué que pour afficher l'URL nous avons aussi utilisé un **TextBlock** et vous vous demandez peut-être pourquoi est-ce que nous n'avons pas utilisé un **HyperLinkButton**. Nous allons voir ça dans la prochaine partie.

2.2. Les Converters

Alors pourquoi est-ce que nous n'avons pas utilisé un **HyperLinkButton** pour afficher mon URL ? Tout simplement car dans notre classe **Tutorial**, l'URL est stockée dans un string. Or la propriété *NavigateUri* veut un objet de type **Uri** : du coup cela n'aurait pas fonctionné.

Une solution aurait été de rajouter une propriété de type **Uri** à ma classe **Tutorial** et de binder cette propriété sur le *NavigateUri* de mon **HyperLinkButton**. Mais cela nous fait rajouter une propriété et, de plus, ce n'est pas super propre.

La solution : utiliser un **Converter** !

Les **Converters** sont simplement des classes qui permettent de convertir un objet d'un type en un autre.

C'est extrêmement simple à faire : il suffit d'implémenter l'interface **IValueConverter**.

Exemple avec un Converter, pour convertir un string en Uri :

```
public class String2UriConverter : IValueConverter
{
    #region IValueConverter Members

    public object Convert(object value, Type
targetType, object parameter,
System.Globalization.CultureInfo culture)
    {
        if (value is string) return new
Uri((string)value);
        else return value;
    }

    public object ConvertBack(object value, Type
targetType, object parameter,
System.Globalization.CultureInfo culture)
    {
        throw new NotImplementedException();
    }

    #endregion
}
```

Nous allons maintenant remplacer notre **TextBlock** par un **HyperLinkButton** :

```
<HyperlinkButton Grid.Row="2" Grid.Column="1"
Content="{Binding Url}"
NavigateUri="{Binding Url,
Converter={StaticResource String2UriConverter}}" />
```

Voici la syntaxe pour utiliser un **Converter** en XAML :

```
Property="{Binding PropertyName,
Converter={StaticResource String2UriConverter}}"
```

En C# :

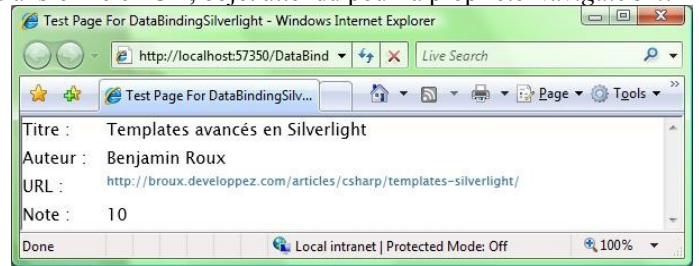
```
System.Windows.Data.Binding binding = new
System.Windows.Data.Binding("PropertyName");
binding.Converter = new
Converters.String2UriConverter();
```

On n'oublie pas de rajouter notre **Converter** en ressource :

```
<UserControl x:Class="DataBindingSilverlight.Page"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml
/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:conv="clr-
namespace:DataBindingSilverlight.Converters"
Width="600" Height="300">
```

```
<UserControl.Resources>
    <conv:String2UriConverter
x:Key="String2UriConverter" />
</UserControl.Resources>
[...]
```

Et voilà le tour est joué. Au moment du Binding, notre **string** est transformé en **Uri**, objet attendu pour la propriété *NavigateUri*.



Nous allons maintenant voir les différents mode de Data Binding.

2.3. Les différents modes

Il faut également savoir qu'il existe plusieurs mode de Binding (enum C#). Ces derniers sont répertoriés sur la page de la MSDN ([Lien28](#)).

Pour rappel, on appelle l'interface **la cible** et le fournisseur de données **la source**.

Type	Description
OneTime	Met à jour la propriété de la cible .
OneWay	Met à jour la propriété de la cible . Une modification sur la source est répercutée sur la cible (valeur par défaut).
TwoWay	Met à jour la propriété de la cible. Une modification sur la source est répercutée sur la cible et inversement.

Pour choisir le mode, il suffit de le spécifier lors du Binding :

```
Property="{Binding PropertyName,
Mode=OneWay/TwoWay/OneTime}"
```

En C# :

```
System.Windows.Data.Binding binding = new
System.Windows.Data.Binding("PropertyName");
binding.Mode = BindingMode.OneTime;
```

Pour les modes **OneWay** et **TwoWay**, les changements ne seront pas effectués tout seuls, il faut que la classe utilisée pour le Binding implémente l'interface **INotifyPropertyChanged** ([Lien29](#)).

Prenons l'exemple avec notre gestionnaire de tutoriels.

Nous allons rajouter un bouton qui modifie la note de notre **tutoriel**, pour voir le résultat :

```
<Button Grid.Row="4" Grid.ColumnSpan="2"
Content="Changer note" Height="50" Width="100"
Click="Button_Click" />
```

```
private void Button_Click(object sender,
RoutedEventArgs e)
{
    templates.Rate = 15;
}
```

Un clic sur le bouton modifie effectivement la note de notre **tutoriel**, mais aucun changement visuel n'apparaît. Pour le faire on

pourrait éventuellement définir notre *DataContext* à *null* avant de le redéfinir sur notre objet, mais cela reste un bidouillage.

La solution est donc, comme expliqué plus haut, d'implémenter l'interface **INotifyPropertyChanged**, dans notre classe **Tutorial** :

```
public class Tutorial : INotifyPropertyChanged
{
    private string mTitle;
    private string mAuthor;
    private string mUrl;
    private int mRate;

    public string Title
    {
        get { return mTitle; }
        set
        {
            mTitle = value;
            NotifyPropertyChanged("Title");
        }
    }

    public string Author
    {
        get { return mAuthor; }
        set
        {
            mAuthor = value;
            NotifyPropertyChanged("Author");
        }
    }

    public string Url
    {
        get { return mUrl; }
        set
        {
            mUrl = value;
            NotifyPropertyChanged("Url");
        }
    }

    public int Rate
    {
        get { return mRate; }
        set
        {
            mRate = value;
            NotifyPropertyChanged("Rate");
        }
    }

    #region INotifyPropertyChanged Members

    public event PropertyChangedEventHandler
    PropertyChanged;

    public void NotifyPropertyChanged(string
    propertyName)
    {
        if (PropertyChanged != null)
        {
            PropertyChanged(this, new
            PropertyChangedEventArgs(propertyName));
        }
    }

    #endregion
}
```

Chaque fois qu'une propriété est modifiée, nous appelons la

méthode **NotifyPropertyChanged**, qui se charge de lancer l'évènement **PropertyChanged**.

Désormais, lorsque notre source de données sera modifiée, toutes les propriétés de nos objets visuels bindées sur des propriétés de notre objet seront mises à jour.

Nous venons d'illustrer le mode **OneWay** : la source de donnée a été modifiée.

Pour illustrer le mode **TwoWay**, nous allons remplacer le **TextBlock** de la note par une **TextBox**. On n'oublie pas de modifier le mode de Binding également :

```
<TextBox x:Name="TxtRate" Grid.Row="3" Grid.Column="1"
Text="{Binding Rate, Mode=TwoWay}" />
```

Désormais, lorsque nous modifierons la valeur de la **TextBox** et que cette dernière perdra le focus, la source de donnée sera mise à jour !

Voici pour les différents mode de Binding, c'est à vous de choisir le mode qui vous convient selon ce que vous voulez faire :

- **OneTime** pour des données qui ne changeront pas
- **OneWay** pour des données qui sont susceptibles d'être modifiées ailleurs
- **TwoWay** pour des données que l'utilisateur peut modifier et qui peuvent aussi être modifiées ailleurs...

2.4. Afficher une liste d'objets

Nous avons vu comment afficher un seul objet, nous allons maintenant voir comment afficher une liste d'objets.

Tout d'abord, nous allons créer une liste de tutoriaux :

```
List<Tutorial> tutoriels = new List<Tutorial>();
```

```
Tutorial templates = new Tutorial()
{
    Author = "Benjamin Roux",
    Title = "Templates avancés en Silverlight",
    Url = "http://broux.developpez.com/articles/csharp/
templates-silverlight/",
    Rate = 10
};

Tutorial animations = new Tutorial()
{
    Author = "Benjamin Roux",
    Title = "Les animations en Silverlight",
    Url = "http://broux.developpez.com/articles/csharp/
animations-silverlight/",
    Rate = 10
};
[...]
tutoriels.Add(templates);
tutoriels.Add(animations);
```

Nous voici avec une liste de **Tutoriel**.

Pour les afficher, nous avons plusieurs possibilités. Tout d'abord une **ListBox**, ou encore un **ItemsControl** (équivalent du **Repeater** en ASP.NET). C'est ce dernier que nous allons utiliser.

```
<ItemsControl x:Name="Tutoriels">
    <ItemsControl.ItemTemplate>
        <DataTemplate>
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="75" />
                    <ColumnDefinition Width="*" />
```

```

</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
    <RowDefinition Height="25" />
    <RowDefinition Height="25" />
    <RowDefinition Height="25" />
    <RowDefinition Height="25" />
    <RowDefinition Height="*" />
</Grid.RowDefinitions>
<TextBlock Grid.Row="0"
Text="Titre :" />
/>
<TextBlock Grid.Row="1" Text="Auteur :"
/>
<TextBlock Grid.Row="2" Text="URL :" />
<TextBlock Grid.Row="3" Text="Note :" /
>
<TextBlock x:Name="TxtTitle"
Grid.Row="0" Grid.Column="1" Text="{Binding Title}" />
<TextBlock x:Name="TxtAuthor"
Grid.Row="1" Grid.Column="1" Text="{Binding Author}" />
<HyperlinkButton Grid.Row="2"
Grid.Column="1" Content="{Binding Url}"
NavigateUri="{Binding
Url, Converter={StaticResource String2UriConverter}}" /
>
<TextBox x:Name="TxtRate" Grid.Row="3"
Grid.Column="1" Text="{Binding Rate}" />
</Grid>
</DataTemplate>
</ItemsControl.ItemTemplate>
</ItemsControl>

```

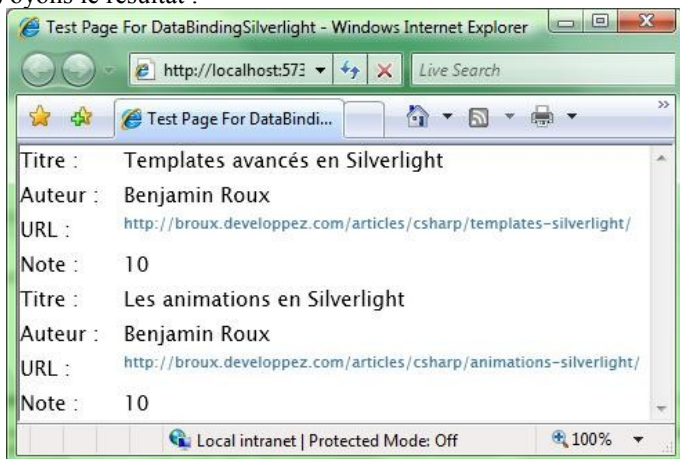
On reprend notre UI d'avant et on l'encadre par un contrôle de type **ItemsControl**.

On n'oublie pas de spécifier notre source de données à ce contrôle :

```
Tutoriels.ItemsSource = tutoriels;
```

Cette fois on utilise la propriété *ItemsSource* et non *DataContext*.

Voyons le résultat :



Présentation du logiciel ReSharper pour Visual Studio .Net

Test et présentation de l'add-in pour Visual Studio: ReSharper 4.0

1. Présentation

ReSharper est un add-in pour Visual Studio qui se présente lui-même comme l'add-in le plus "intelligent". Il y a trois ans, Thomas Lebrun présentait la version 1,5 de ReSharper ([Lien33](#)) et ses fonctionnalités, et ReSharper était alors déjà reconnu comme le meilleur add-in pour Visual Studio.

Il y a peu, sortait la version 4.0 de ReSharper, toujours plus

Nous avons tous nos éléments les uns à la suite des autres. Vraiment pratique !

On peut également modifier les éléments comme précédemment : **TextBox** pour la note, ou modification de la note via un bouton. Tous les changements sont effectués visuellement.

En revanche, il y a un problème : l'ajout ou la suppression d'éléments dans notre liste.

Si l'on rajoute un bouton et que lors du clic sur le bouton on tente de rajouter un élément à notre **List**, comme ceci.

```

Tutorial databinding = new Tutorial()
{
    Author = "Benjamin Roux",
    Title = "Le Data Binding en Silverlight",
    Url = "http://broux.developpez.com/articles/csharp/databinding-silverlight/",
    Rate = 10
};
tutoriels.Add(databinding);

```

L'élément n'apparaît pas ! On pourrait comme précédemment mettre la propriété *ItemsSource* à *null* pour la remettre sur notre **List** par la suite mais c'est encore et toujours du bidouillage.

La solution est d'utiliser une collection qui implémente *INotifyCollectionChanged* ([Lien30](#)).

Ca tombe bien : Silverlight 2 met à notre disposition une classe qui l'implémente : *ObservableCollection<T>* ([Lien31](#)).

Cette classe fait tout comme la classe **List<T>**, sauf qu'elle implémente **INotifyCollectionChanged**.

On remplace dans notre code :

```

ObservableCollection<Tutorial> tutoriels = new
ObservableCollection<Tutorial>();

```

Et désormais, lorsqu'un élément est ajouté ou supprimé de notre collection, un événement est lancé. Il dit à notre **ItemsControl** que la source a changé et qu'il doit donc faire les modifications nécessaires !

2.5. Conclusion

Voici la fin de notre article sur le Data Binding en Silverlight 2 qui, j'espère, vous sera très utile. Il est à noter que des nouveautés y seront apportées lors de la release de Silverlight 2.

Retrouvez l'article de Benjamin Roux en ligne : [Lien32](#)

télécharger très simplement une version d'essai de 30 jours à l'adresse suivante <http://www.jetbrains.com/downloads.html> ([Lien34](#)) et l'installer sur votre ordinateur. ReSharper est capable de détecter les versions de Visual Studio installées et de se configurer automatiquement (Il est compatible avec VS 2005 et VS 2008).

Une fois installé, vous n'avez qu'à lancer Visual Studio et accepter le contrat utilisateur de la version d'essai et saisir votre licence si vous en avez une.

3. Les fonctionnalités

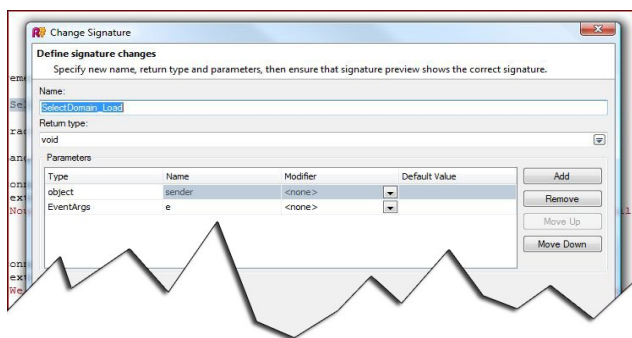
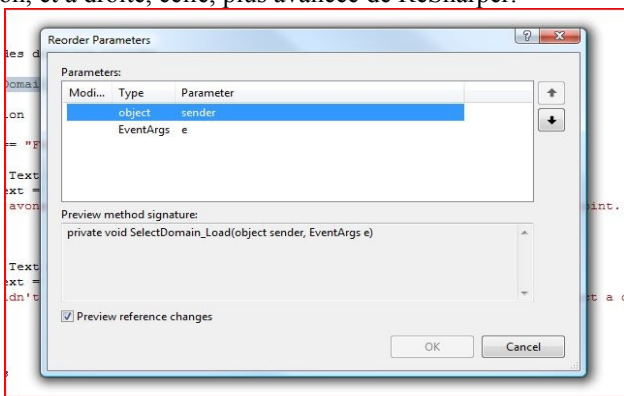
3.1. Refactoring amélioré

Pour rappel, le refactoring est le fait de modifier le code pour le rendre plus propre et/ou plus optimisé sans changer son comportement (pas de réécriture de nouvelle méthode, classe ou autre).

Si ma mémoire est bonne, le premier vrai refactoring est apparu avec Visual Studio 2005. Pourtant, il était déjà possible grâce à ReSharper d'avoir des fonctionnalités équivalentes. Ce qui aujourd'hui vous paraît normal comme l'encapsulation d'un membre pour en faire une propriété, l'extraction d'une méthode ou tout simplement le renommage récursif d'un objet, tout ceci n'était alors pas possible sans ReSharper. Maintenant c'est possible par défaut mais ReSharper fait tout en mieux et il le fait pour presque tout ce qui est utilisable dans Visual Studio : C#, VB.NET, ASP.NET, XML, XAML et build scripts.

Prenons pour exemple des refactorings simples mais utiles. Nous avons tout d'abord l'optimisation des using qui consiste à supprimer les using inutiles.

Vous pouvez également changer la signature d'une méthode en ajoutant, éditant ou supprimant des paramètres. La capture suivante montre à gauche la fenêtre de Visual Studio pour cette action, et à droite, celle, plus avancée de ReSharper.



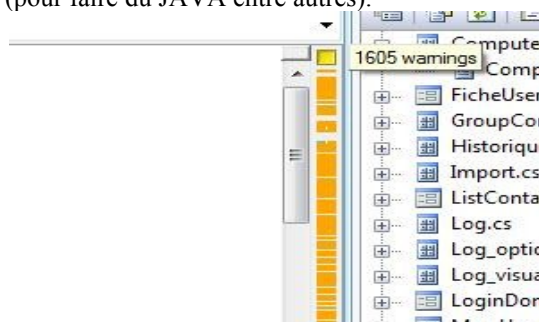
Vous avez également des fonctionnalités toutes simples comme le safe delete, qui, avant de supprimer un objet, vérifie que celui-ci n'est pas appelé ailleurs. Il n'est alors plus nécessaire de régénérer

la solution pour voir si l'on pouvait le supprimer ou non.

Pour plus d'infos sur les différents refactorings possibles, cliquez ici ([Lien35](#))

3.2. Le code inspection

L'une de mes fonctionnalités préférées est le code inspection. Il s'agit d'une analyse en arrière plan du code de vos classes ou simplement de vos fichiers XML/XAML. L'inspection regarde votre code pour lister dans la partie droite de l'éditeur, les erreurs, les warnings et les optimisations possibles un peu comme vous pouvez déjà le trouver dans d'autres éditeurs connus comme Eclipse (pour faire du JAVA entre autres).

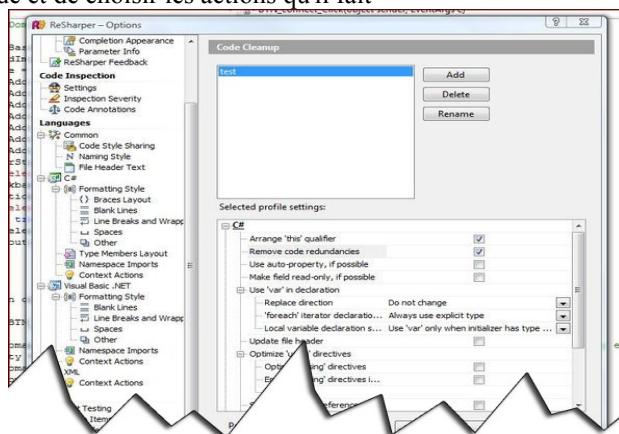


A partir de cette barre de droite, vous pouvez cliquer sur le signet, ce qui vous amènera directement à la ligne incriminée et de là, en cliquant sur la petite icône apparaissant en début de ligne, vous pourrez choisir l'action à entreprendre.

3.3. Code cleanup

Le code cleanup est une action, via un seul bouton qui permet d'un seul coup d'automatiser plusieurs tâches basiques, de formatage de code et de faire du refactoring. Il existe deux modes par défaut, le mode complet et le mode format qui ne fait que du formatage.

Il est bien entendu possible, via les options de créer votre propre mode et de choisir les actions qu'il fait



3.4. Amélioration de la productivité

Ce sont souvent des toutes petites choses, une seconde grappillée par là, une autre par ici, qui font que l'on gagne un temps non négligeable dans le développement d'applications conséquentes. Ainsi, plus vous aurez un gros projet, plus ces améliorations auront une réelle plus-value.

Avec ReSharper, la productivité est améliorée par les fonctionnalités de code inspection et de refactoring qui évitent (ou plutôt réduisent) le process de test et de vérification du code, mais c'est aussi et surtout des petites choses qui vous simplifient les tâches les plus courantes.

Ainsi, vous avez des raccourcis qui, lorsque vous commencez à

saisir la signature d'une méthode, vous permettent en tapant ctrl+Maj+Entrée, de fermer la parenthèse, de créer les accolades et de place correctement votre curseur. Vous avez également le CamelHumps, qui vous permet de saisir uniquement les majuscules du nom d'une classe pour que l'auto-complétion sache trouver cette classe. Dans l'exemple suivant, plutôt que de taper ProcessPriorityClass, il me suffit de saisir " PPC ", de faire ctrl+espace et de choisir les classes qui correspondent à ce pattern.

Enfin, vous avez les snippets qui, à partir d'un mot clé, comme " foreach " vous permet d'un seul raccourci clavier d'écrire la structure de la boucle foreach et de saisir uniquement les propriétés manquantes (compteur, limite, incrémentation).

3.5. Support de LINQ et de C# 3.0

ReSharper 4.0, est maintenant capable de prendre en charge la dernière mouture de C# mais également des petites nouveautés (si l'on peut dire) comme LINQ. Cette prise en charge peut se voir au niveau de toutes les fonctionnalités de ReSharper.

- Refactoring adapté au C# 3.0 comme les nouveaux initialiseurs d'objets. L'utilisation du mot-clé " var ", la conversion en expression lambda, les auto-propriétés, etc

```
var monping = new Process();
monping.StartInfo.FileName = "ping.exe";
monping.StartInfo.UseShellExecute = false;
monping.StartInfo.CreateNoWindow = true;
monping.StartInfo.RedirectStandardOutput = true;
```

```
var monping = new Process
{
    StartInfo =
    {
        FileName = "ping.exe",
        UseShellExecute = false,
        CreateNoWindow = true,
        RedirectStandardOutput = true
    }
};
```

- Nouveau code-inspection sur ces spécificités du langage C# 3.0
- Auto-complétion capable de prendre en charge les méthodes d'extension ou la possibilité de générer des expressions lambdas dans certains cas de figure
- L'ajout de snippets spécifiques à l'écriture de requêtes LINQ comme le join et le from.

```
203
204
205
206
207
208
209
210
211
212
213
214
```

```
var domainTest = new DirectoryEntry("LDAP:" +
string test = domainTest.SchemaClassName;
```

```
= new TreeNode(domain.Name);
= domain;
listDomains.Items.Add(childNode.Text);
```

4. Autres fonctionnalités

4.1. La personnalisation des fonctionnalités

Une chose qu'il est important de préciser avec ReSharper est qu'il n'est pas intrusif. Par défaut, en raison de son grand nombre de fonctionnalités, il est nécessaire de créer un certain nombre de menus et de comportement dans Visual Studio mais il est possible de choisir le niveau d'intégration dans l'IDE.

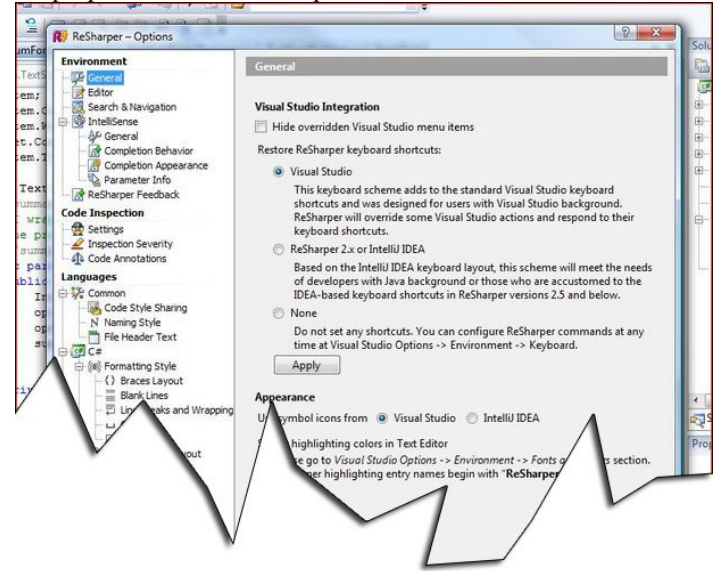
Cela peut se faire de deux manières :

- Via les options
- Via l'explorateur de templates

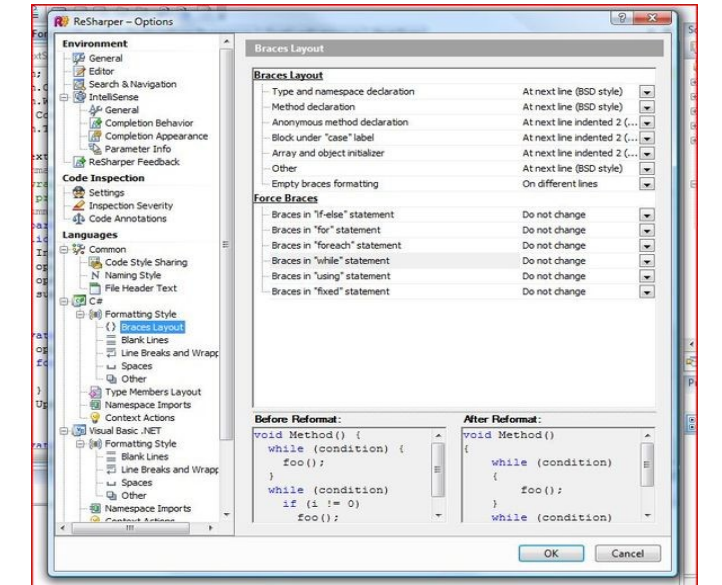
1- Les options

Ce qui est considérable avec ReSharper, c'est son nombre d'options qui permettent de le personnaliser très très finement.

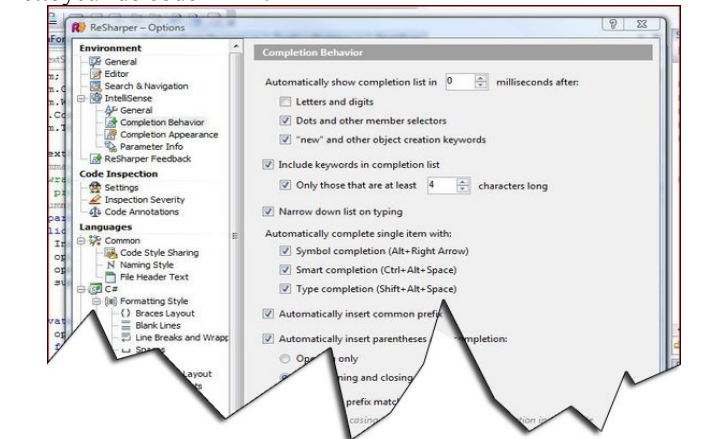
Nous pouvons tout d'abord choisir le niveau d'intégration de l'addin avec ses menus, la façon dont ils apparaissent et aussi les fonctionnalités qu'ils proposent. ReSharper propose par exemple, son propre menu d'auto-complétion.



Il existe aussi une grande partie d'options concernant le formatage du code, tout cela en plus du formatage possible par Visual Studio.



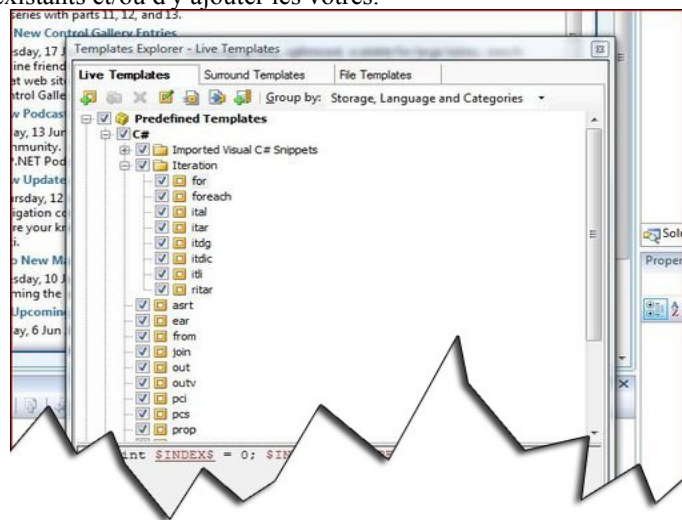
Enfin, et plus important, vous pouvez personnaliser l'inspection du code (vue dans le chapitre précédent) pour définir quelles types d'erreurs ou d'optimisations ReSharper vous proposera. Il est au même endroit possible de contrôler le comportement de ReSharper avec le code comme définir les expressions régulières qui aident à faire fonctionner l'explorateur de TODO ou encore le nettoyeur de code XML.



2- Les snippets

Les snippets existent depuis un certain moment mais ReSharper propose lui aussi les siens. Cela peut aller des snippets courants comme les boucles for ou foreach, aux snippets tous simples qui permettent en tapant trois lettres de déclarer une propriété (getsetter) et ce, de façon plus rapide que le bouton droit > Refactor > Encapsulate field.

Bien entendu, il est possible de personnaliser les snippets existants et/ou d'y ajouter les vôtres.



4.2. L'ajout de plugins

Qu'est ce qu'un add-in sinon un plugin. Et bien, malgré que ReSharper soit un plugin pour Visual Studio, il est possible d'ajouter des plugins à ReSharper. ReSharper compte déjà neuf plugins officiels qui ont pour la plupart, comme objectif, d'affiner

une fonctionnalité spécifique de ReSharper. Ainsi, nous trouverons des plugins comme Agent Smith qui, un peu comme le fait FxCop, permet de vérifier des conventions de nommage pour avoir un code propre.

Mais vous trouverez également des plugins facilitant l'utilisation de NHibernate (pour faire de l'ORM) ou encore log4net (pour le logging) ou encore des plugins, dédiés au Unit Testing et permettant d'utiliser NUnit, xUnit, MbUnit ou encore MSTest pour améliorer l'étape de test de vos applications.

Pour avoir des infos plus détaillées sur les plugins existants, vous pouvez vous rendre sur la page suivante ([Lien36](#)).

5. Conclusion

Je suis quelqu'un de très porté sur l'ajout de fonctionnalités à des outils déjà existants et il y a peu encore, je m'intéressais même à la création d'add-ins pour Office ou pour Visual Studio. Pourtant depuis Visual Studio 6 (la première version que j'ai jamais utilisé), je n'ai jamais trouvé d'add-in valant le coup d'être installé ou gardé sur Visual Studio, outil on ne peut plus complet.

Pourtant, j'ai découvert ReSharper du temps de Visual 2003 et le refactoring était déjà alors LA fonctionnalité qui justifiait son achat. Avec la venue de Visual Studio 2005, ReSharper devenait moins intéressant et a valu sa mise de côté, de la part de beaucoup de développeurs de mon entourage. Aujourd'hui, avec sa version 4.0, ReSharper revient sur le devant de la scène et peut se targuer sans conteste, d'être l'add-in le plus complet pour Visual Studio et ajoutant un grand nombre de fonctionnalités améliorant la fiabilité des problèmes et la vitesse de développement.

Retrouvez l'article de Louis-Guillaume Morand en ligne : [Lien37](#)

Les derniers tutoriels et articles

Pointeurs intelligents

Ce document présente un outil indispensable à l'écriture de code correct en C++ : Les pointeurs intelligents. Après une présentation du problème que ces pointeurs aident à résoudre, il décortique comment il est possible de créer un tel pointeur, et enfin présente les pointeurs intelligents les plus courants, et comment les utiliser.

1. Introduction

1.1. À qui est destiné ce document

Ce document s'adresse à un public ayant déjà des notions de base en C++, en particulier les notions de pointeurs, d'exceptions et l'utilisation de templates sont supposées connues. Mais les notions présentées ne sont pas avancées, et les connaître est utile à toute personne désirant écrire un programme en C++ robuste.

Ce document ne reprend pas toutes les fonctions des pointeurs intelligents, mais seulement celles qui me semblaient les plus courantes. Une bonne référence sur le sujet peut donc être un complément intéressant à cet article.

Il y a d'autres ressources sur ce sujet sur le site, par exemple Boost.SmartPtr : les pointeurs intelligents de Boost de Matthieu Brucher ([Lien38](#)) ou Gérer ses ressources de manière robuste en C++ par Aurélien Regat-Barrel ([Lien39](#))

Les sections « Historique » et « Notre propre pointeur intelligent : ValuePtr » ne sont pas indispensables à la compréhension du reste de l'article.

1.2. Historique

La notion de pointeur intelligent existe depuis un certain temps dans le langage. C++98 (le standard actuellement en vigueur, 98 signifiant qu'il a été publié en 1998) définit même dans sa bibliothèque un pointeur intelligent, nommé `auto_ptr`. Cette première tentative n'est pas sans problèmes, et il est généralement déconseillé d'utiliser cette classe.

Après la sortie du standard, un certain nombre de personnes intéressées par l'évolution du langage se sont rassemblées pour fonder ce qui allait devenir boost, une collections de bibliothèques d'intérêt général pour les développeurs C++, ayant pour vocation de servir de terrain d'expérimentation en grandeur nature pour le prochain standard. La bibliothèque qui a probablement le plus servi à assurer le succès de cette initiative est la bibliothèque `smart_ptr`, qui contient un certain nombre de pointeurs intelligents, parmi lesquels `boost::shared_ptr` et `boost::weak_ptr`, mais aussi `boost::scoped_ptr`.

Par la suite, le comité de normalisation a publié, sous le nom TR1, une liste de bibliothèques qui ferait partie du prochain standard C++. On peut voir le TR1 comme une version 1.1 du standard. A l'heure où cet article est écrit, la plupart des compilateurs commencent à proposer une implémentation de TR1. Ce dernier contient deux pointeurs intelligents, `std::tr1::shared_ptr` et `std::tr1::weak_ptr`, qui reprennent les versions issues de boost.

Enfin, le prochain standard, nommé temporairement C++0x, contiendra aussi des pointeurs intelligents. Même si la situation reste en théorie susceptible d'évoluer, cette partie du standard est assez stabilisée, et il est prévu qu'il contienne `std::shared_ptr`, `std::weak_ptr` et `std::unique_ptr` (qui remplace `std::auto_ptr`, conservé uniquement pour préserver le code existant, et `boost::scoped_ptr`).

On peut avoir un peu peur de se perdre entre tous ces pointeurs intelligents, d'autant que chaque version a des différences subtiles avec les autres. Il n'empêche que ces versions partagent aussi beaucoup de choses en commun, et qu'une sorte de compatibilité ascendante existe. Si un choix est à réaliser, ce document se placera dans l'optique C++0x, mais à part ce qui concerne `unique_ptr`, il devrait être directement utilisable si vous utilisez les pointeurs intelligents de boost ou de TR1.

2. Les pointeurs intelligents, pourquoi, comment ?

2.1. Problèmes des pointeurs nus

Avant de voir les différents types de pointeurs intelligents, il est utile de revenir sur les pointeurs classiques du C++, que je nommerai pointeurs nus par opposition aux pointeurs intelligents. Ces pointeurs présentent un certain nombre de problèmes qui en rendent l'usage difficile et risqué, les différents pointeurs intelligents ayant pour objectif de corriger ces problèmes.

Il y a globalement trois risques d'erreur, et un problème de clarté, avec les pointeurs nus.

2.1.1. Ne pas libérer de la mémoire allouée dynamiquement

A partir du moment où on alloue dynamiquement de la mémoire, il faut la désallouer quand on a fini de s'en servir. Sinon, la mémoire reste réservée, et au fur et à mesure que le temps passe, notre programme va "manger" la mémoire disponible sur la machine, conduisant à des baisses de performance et à des explosions, c'est ce qu'on nomme une fuite mémoire.

Cette contrainte pose deux problèmes : Elle oblige déjà à une grande discipline de codage, afin de ne pas oublier un `delete` quelque part, et en plus, elle est très difficile à mettre en œuvre correctement.

```
int g();  
  
int f()  
{  
    int *i = new int(42);  
    g();  
}
```

```

delete i;
}

class A
{
public:
    A() : i(new int(314)), j(new int(42)) {}
    ~A()
    {
        delete j;
        delete i;
    }
private:
    // Fonctions déclarées privées pour empêcher le
    compilateur de générer
    // un constructeur de recopie et un opérateur
    d'affectation par défaut.
    A(A const &);
    A& operator= (A const &);

    int *i;
    int *j;
};

```

Dans ces deux cas, on peut penser avoir correctement fait le travail de désallocation. Mais ce n'est pas le cas. Si par exemple la fonction g lance une exception, le delete i ne sera jamais appelé.

Dans le second cas, même problème, si le new int(42) échoue, par manque de mémoire, une exception est lancée. Cette exception fait que la classe A sera considérée comme n'ayant jamais été construite, et on ne passera donc pas dans son destructeur. La mémoire alloué pour i ne sera donc jamais libérée.

Le plus difficile, c'est que tant qu'une exception n'est pas lancée à un moment bien précis, le code va tourner sans problèmes. Et même si l'exception est lancée, une fuite mémoire n'est pas forcément très visible et peut donc passer inaperçue pendant longtemps. Ce genre d'erreur risque donc fortement de ne pas être détectée pendant la phase de tests, jusqu'au jour où elle deviendra très gênante, quand le code sera déployé...

La solution classique à ce problème en C++ est le RAII ([Lien40](#)). Dans les deux exemples cités, le RAII classique résoudrait le problème. Par contre, le RAII lie la durée de vie des objets avec la portée de la variable locale. C'est souvent un peu simpliste, puisque justement, si on a utilisé de l'allocation dynamique, c'est que l'on souhaite décorréliser la durée de vie de l'objet avec la portée où il est déclaré. Les pointeurs intelligents peuvent être vus comme une extension du RAII ayant une gestion plus fine de la durée de vie de l'objet pointée. Parmi ceux-ci, boost::scoped_ptr représente ni plus ni moins que la mise en place de RAII pour gérer de la mémoire.

2.1.2. Libérer plusieurs fois de la mémoire allouée dynamiquement

Le code suivant possède des erreurs :

```

int *i = new int(14);
delete i;
delete i; // Erreur : Pas deux désallocations
int *j;
delete j; // Erreur : Désallocation d'une zone non
allouée
int *k = NULL;
delete k; // Ok : On peut faire delete NULL sans
problèmes

```

La règle est qu'on ne peut désallouer qu'une zone qui est allouée

(ou NULL). Ce cas se présente moins souvent que le cas précédent.

2.1.3. Accéder à la valeur pointée par un pointeur invalide

Le dernier type de problème pouvant arriver, c'est de tenter de déréférencer un pointeur invalide. Ça arrive le plus souvent quand plusieurs pointeurs pointent sur une même zone de mémoire, que l'on fait delete sur l'un des pointeurs, mais que les autres pointeurs ne sont pas au courant et continuent de tenter d'y accéder.

```

int *i = new int(42);
int *j = i;
delete i;
*j = 2;

```

2.1.4. Un pointeur nu est peu explicite

Prenons par exemple une fonction :

```

MaStructure const *getInformation();

```

Il est évident que cette fonction va retourner un pointeur sur une zone de mémoire contenant les informations que l'on désire. Mais qu'est-on sensé faire de ce pointeur quand on a fini d'utiliser ces informations ? Rien ? L'effacer avec delete ? L'effacer avec free ? Autre chose ? On n'en sait rien a priori, toutes ces possibilités ayant des cas d'application concrets. Certes, la documentation associée à la fonction getInformation, si elle est écrite correctement, doit nous fournir la réponse, mais il serait plus agréable de ne pas avoir à se reposer sur une documentation pour ce genre de choses.

2.2. Qu'est-ce qu'un pointeur intelligent

Si les pointeurs nus présentent tous ces problèmes, ne serait-il pas possible de les remplacer par autre chose qui ne les ait pas ? C'est de cette idée que sont nés les pointeurs intelligents. Tout comme il existe différents scénarios types d'utilisation des pointeurs, il existe différents pointeurs intelligents, chacun répondant à un besoin particulier.

Un pointeur intelligent est une classe qui encapsule la notion de pointeur, tout en offrant une sémantique de plus haut niveau. On a vu que les principaux problèmes liés aux pointeurs nus étaient liés à la durée de vie des objets pointés, et au lien entre celle-ci et la durée de vie des pointeurs eux-mêmes. C'est donc principalement autour des opérations liées à la durée de vie des pointeurs (création, copie, destruction...) qu'on ajoutera de l'intelligence, le reste de la classe étant juste là pour donner un accès agréable au pointeur sous-jacent.

Si l'on voulait donner une définition d'un pointeur intelligent, on pourrait donc dire qu'il s'agit d'une classe que l'on utilise presque comme un pointeur, mais qui possède un mécanisme gérant la durée de vie des objets pointés.

Dans du code C++ moderne, utilisant ce genre de techniques, on se retrouve donc à avoir très peu de delete, la mémoire étant presque systématiquement gérée automatiquement.

Parmi les usages classiques des pointeurs, il y en a deux qui ne sont généralement pas couverts par les pointeurs intelligents :

- Les pointeurs sur des tableaux de caractères, ayant pour but de représenter des chaînes de caractères, pour lesquels les classes std::string et std::wstring apportent les mêmes avantages, tout en offrant en plus un certain nombre d'opérations spécifiques à des chaînes de caractères
- Les pointeurs ayant pour but de gérer des tableaux de

taille dynamique, rôle géré directement par les conteneurs de la bibliothèque standard, et en particulier `std::vector`.

Il y a aussi certains styles de programmation pour lesquels des pointeurs intelligents sont moins indispensables, en particulier quand chaque objet appartient de manière unique et non équivoque à un gestionnaire s'occupant de sa durée de vie, et que par design, on sait qu'on n'aura pas de destruction prématurée.

2.3. Notre propre pointeur intelligent : ValuePtr

Afin de rendre plus concrète la notion de pointeur intelligent, et d'étudier les mécanismes du C++ qui rendent leur écriture possible, nous allons maintenant définir notre propre classe de pointeurs intelligents, correspondant à un scénario non prévu dans le standard. Ceux qui sont uniquement intéressés par l'utilisation des pointeurs intelligents sans désirer savoir ce qui se passe sous le capot peuvent passer cette section. D'autant plus que c'est dans cette section que seront utilisés les aspects les plus avancés du C++ de l'article.

La classe que nous allons écrire ne sera pas de qualité industrielle (par exemple, pas de gestion de la thread safety, peu de gestion de l'exception safety...), elle a juste pour but d'illustrer les principaux mécanismes mis en jeu.

2.3.1. Problématique

On veut manipuler des formes mathématiques, des cercles, des rectangles... Dans le monde idéal, on écrirait du code comme ça :

```
vector<Shape> v;
v.push_back(Rectangle(0, 0, 10, 20));
v.push_back(Circle(0, 0, 10));
for(vector<Shape>::iterator it = v.begin();
    it != v.end();
    ++it)
{
    it->draw();
}

vector<Shape> v2 = v;
for(vector<Shape>::iterator it = v2.begin();
    it != v2.end();
    ++it)
{
    it->move(2, 2);
}
```

Sauf que comme on le sait, ça ne marche pas ainsi. Il y a ce qu'on appelle du slicing. Si on essaye par exemple de mettre un Circle dans notre vecteur, on va se retrouver quelque part à l'intérieur du vecteur à faire un code équivalent à `Shape s = aCircle`. Or, un objet a en C++ un type bien déterminé, ici Shape, avec un arrangement mémoire connu à la compilation. On ne peut pas faire entrer un Circle dans de la mémoire prévue pour un Shape, il n'y a pas assez de place. Cette opération va donc convertir le Circle en Shape, ce qui correspond à supprimer toutes les variables membres ajoutées au niveau de la classe dérivée, à couper (slice) l'objet pour n'en garder que la partie définie dans la classe de base.

La façon d'utiliser du polymorphisme en C++ passe par des pointeurs (ou des références). On a une zone mémoire allouée pour un Circle, mais on va y accéder par l'intermédiaire d'un pointeur sur un Shape, et c'est ce pointeur que l'on va mettre dans notre conteneur :

```
vector<Shape*> v;
v.push_back(new Rectangle(0, 0, 10, 20));
v.push_back(new Circle(0, 0, 10));
```

```
for(vector<Shape*>::iterator it = v.begin();
    it != v.end();
    ++it)
{
    (*it)->draw();
}

vector<Shape*> v2 = v;
for(vector<Shape*>::iterator it = v2.begin();
    it != v2.end();
    ++it)
{
    (*it)->move(2, 2);
}
```

Les problèmes sont multiples : On doit désormais s'occuper de désallouer correctement ce qui a été alloué (ce que je n'ai pas fait ici), de plus, on a sans forcément le vouloir une sémantique d'entité, alors qu'initialement on avait une sémantique de valeur. Ainsi, la deuxième partie du code modifie aussi le contenu de `v`, ce qui n'était pas le cas dans la version précédente.

Nous sommes dans le cas où l'on utilise des pointeurs alors que l'on souhaite manipuler des données ayant une sémantique de valeur, uniquement parce que l'on veut du polymorphisme. L'idiome classique pour gérer ce genre de cas est l'idiome lettre enveloppe. On va ici voir une autre approche, un peu plus générique dans sa mise en œuvre : On va demander à l'utilisateur de nos classes de manipuler des pointeurs intelligents vers les objets, mais ces pointeurs auront la particularité de copier les objets pointés lorsqu'on copie le pointeur, permettant de restaurer une sémantique de valeur.

2.3.2. Structure de base

Le premier point, c'est de choisir comment nommer notre classe. On l'appellera ValuePtr, pour indiquer qu'il s'agit d'un pointeur avec une sémantique de valeur.

On veut pouvoir créer des ValuePtr qui pointent sur divers types de données, des Shape, des Circle, des Rectangle, mais aussi d'autres types que l'on n'a pas encore envisagés. On va donc faire de ValuePtr un template, qui aura comme argument le type sur lequel le pointeur intelligent va pointer.

Enfin, notre classe va encapsuler un pointeur, et aura donc une donnée membre de ce type. Voici donc la structure de base de notre pointeur intelligent.

```
template<class T>
class ValuePtr
{
private:
    T* myPtr;
};
```

2.3.3. Ressembler à un pointeur

On veut que notre pointeur intelligent ressemble en terme d'utilisation à un pointeur. Les deux opérations les plus fondamentales que l'on peut effectuer sur un pointeur `p`, c'est accéder à l'objet pointé, par l'écriture `*p`, ou accéder au contenu de l'objet pointé, par l'écriture `p->`.

L'écriture `*p` n'est en pratique pas utilisée si souvent que ça, mais il n'est pas compliqué de la fournir. On va définir l'opérateur `*` unaire pour qu'il effectue cette tâche. Comme on veut qu'à partir de cet opérateur, l'utilisateur accède directement à l'objet pointé et non à une copie, cet opérateur va retourner une référence vers le contenu du pointeur.


```

template<class T>
class ValuePtr
{
public:
    T& operator*() const {return *myPtr;}
private:
    T* myPtr;
};

```

De son côté, l'écriture `p->` est à la base de l'utilisation du pointeur. On va cette fois la reproduire en surchargeant l'opérateur `->` sur notre classe :

```

template<class T>
class ValuePtr
{
public:
    T* operator->() const {return myPtr;}
private:
    T* myPtr;
};

```

L'opérateur `->` est un peu spécial. Alors qu'en général, `a @ b` se traduit par `operator@(a, b)` ou `a.operator@(b)`, dans le cas de l'opérateur `->`, `a->b` se traduit par `(a.operator->())->b`. Donc, notre opérateur `->` retourne le pointeur nu interne, et le programme va transmettre au pointeur nu ce que l'utilisateur a voulu faire à notre `ValuePtr`. Contrairement au cas précédent, le pointeur nu n'est utilisé que comme une valeur temporaire, auquel l'utilisateur n'a pas un accès direct, et il ne peut donc pas faire de bêtises avec.

2.3.4. Implémenter notre sémantique

Nous voulons que quand on crée un `ValuePtr`, il prenne la responsabilité d'un pointeur que l'on passe en argument du constructeur. Ainsi, quand on détruit le `ValuePtr`, il va automatiquement détruire la donnée pointée par ptr.

```

template<class T>
class ValuePtr
{
public:
    ValuePtr(T* ptr) : myPtr(ptr) {}
    ~ValuePtr() {delete myPtr;}
    T* operator->() {return myPtr;}
private:
    T* myPtr;
};

```

Que doit-il se passer quand l'utilisateur copie le pointeur ? On veut tout simplement que la valeur pointée soit copiée elle aussi. Une première tentative pourrait être de passer par le constructeur de copie de la valeur pointée :

```

template<class T>
class ValuePtr
{
public:
    ValuePtr(ValuePtr const &other) :
        myPtr(new T (*(other.myPtr)))
    {}
    ValuePtr &operator=(ValuePtr const &other)
    {
        // Je sais que ce code ne gère pas l'auto-
        // affectation
        // mais le but est juste de présenter le
        concept
        delete myPtr;
        myPtr = new T(*(other.myPtr));
        return *this;
    }
    // ...
};

```

Le problème est que notre principal cas d'utilisation est pour des classes faisant partie d'une hiérarchie polymorphe. On peut donc avoir un `ValuePtr<Shape>` qui en fait pointe sur un `Circle`, mais dans la copie, on va créer un nouvel objet du type `Shape`. Il nous faut utiliser l'idiome du constructeur de copie virtuel ([Lien41](#)) : Nous allons imposer à notre type `T` d'avoir une fonction (virtuelle en général) `clone` qui produit un double de l'objet existant. On va tant qu'à faire en profiter pour corriger les problèmes d'auto-affectation et d'exception safety de l'opérateur d'affectation.

```

template<class T>
class ValuePtr
{
public:
    ValuePtr(ValuePtr const &other) :
        myPtr(other->clone())
    {}
    void swap(ValuePtr &other)
    {
        std::swap(myPtr, other.myPtr);
    }
    ValuePtr &operator=(ValuePtr const &other)
    {
        ValuePtr<T> temp(other);
        swap(temp);
    }
    // ...
};

```

Nous avons désormais un `ValuePtr` fonctionnel, mais très minimaliste. Voyons comment on peut en rendre l'usage plus simple pour l'utilisateur.

2.3.5. Flexibilité pour le type pointé

Actuellement, on impose au type pointé d'implémenter l'idiome du constructeur virtuel par une fonction nommée `clone()`. Pas `Clone()`, ni `Copy()`, ni `copie()`, ni... C'est très restrictif. Trop.

La solution future, quand le C++0x sera sorti, sera simplement de définir dans le concept associé à notre template que l'on a besoin de faire `p.clone()` sur le paramètre template, et de laisser l'utilisateur utiliser les concept_map pour faire correspondre sa fonction de copie avec ce que veut le concept. Voyons étape par étape comment y parvenir :

On déclare dans un concept qu'un objet `Clonable` est un objet qui possède une fonction `clone` retournant un pointeur du même type. Ce concept est auto afin que toutes les classes ayant cette fonction soient immédiatement considérées comme répondant au concept (sinon, il faudrait définir un concept map dans tous les cas).

```

auto concept Clonable<class T>
{
    T* T::clone();
}

```

Ensuite, on utilise ce concept pour définir notre pointeur intelligent. Ça a deux effets :

- Tout d'abord, si jamais dans l'implémentation de notre pointeur intelligent, on essaye d'utiliser sur un objet de type `T` une fonctionnalité autre que celle définie dans `Clonable`, le code échouera.
- Ensuite, si on essaye d'instancier un pointeur intelligent sur un type non `Clonable`, l'instanciation échouera (avec un message d'erreur qui devrait avoir du sens pour l'utilisateur).


```

template <Clonable T>
class ValuePtr
{
public:
    ValuePtr(ValuePtr const &other) : myPtr(other-
>clone())
    {
    }
    // ...
};

```

Pour l'instant on n'a rien résolu, notre classe A par exemple n'étant **a priori pas Clonable** :

```

class A
{
public:
    A* copy() {return new A(*this);}
};

```

Mais finalement, la personne désirant créer un ValuePtr<A> peut dire que si, A est en fait Clonable, et que quand le pointeur intelligent désire appeler la fonctionnalité clone promise dans le concept, il peut en fait appeler la fonction copy sur l'objet en cours. Il suffit pour ça d'écrire un concept map :

```

concept_map Clonable<A>
{
    A* A::clone() {this->copy();}
};

```

Et voilà, le tour est joué.

La solution dans les frontières du langage actuel consiste à passer par l'intermédiaire de ce qu'on appelle une classe de traits. Comme d'habitude, on résout le problème en ajoutant une indirection supplémentaire :

```

template<class T>
class CloneTraits
{
    T* clone(T* t) {return t->clone();}
}

template<class T>
class ValuePtr
{
public:
    ValuePtr(ValuePtr const &other) :
        myPtr(CloneTraits::clone(other.myPtr))
    {}
    // ...
};

```

A priori, on n'a pas gagné grand-chose, juste la complexité apportée par une classe supplémentaire. Par contre, la personne qui souhaite utiliser notre pointeur intelligent avec un type ayant une autre écriture pour clone peut toujours spécialiser la classe CloneTraits :

```

class A
{
public:
    A* copy();
};

template<>
class CloneTraits<A>
{
    A* clone(A* a) {return a->copy();}
};

```

Si vous voulez plus de détails à propos du fonctionnement de cette

technique, vous pouvez aller voir Présentation des classes de Traits et de Politiques en C++ par Alp Mestan ([Lien42](#))

2.3.6. Ressembler à un pointeur : La suite

Il y a bien des opérations que l'on peut faire sur un pointeur qui ne sont pas prises en compte par notre type. Par exemple, on peut convertir un pointeur sur une classe dérivée vers un pointeur sur une classe de base, ou plus généralement, si U est convertible en T, U* est convertible en T* :

```

template<class T>
class ValuePtr
{
public:
    template<class U> ValuePtr(ValuePtr<U> const &uPtr)
    :
        myPtr(uPtr.get()->clone())
    {}
    T* get() {return myPtr;}
    // ...
};

```

Deux remarques : Déjà, la déclaration de ce constructeur ne dispense pas du constructeur de copie, même si on pourrait croire qu'il s'agit d'un cas plus général de ce dernier. En effet, un constructeur template n'est jamais pris en compte quand il s'agit de trouver un constructeur de copie. Ensuite, ValuePtr<U> étant un type distinct de ValuePtr<T>, on ne peut pas accéder à la donnée privée uPtr.myPtr, d'où la présence de la fonction get, qui peut de toute façon être utile en elle même.

Parmi les opérations supportées par les pointeurs, il y a les opérations arithmétiques, mais celles-ci servent à gérer des tableaux, et sont donc peu utiles pour notre cas.

Il y a aussi la gestion de pointeur nul (par exemple, un ValuePtr créé avec le constructeur par défaut), et le fait qu'un tel pointeur doive se convertir en false dans un contexte où un booléen est attendu. Le plus basique pour y parvenir est (le explicit devant operator bool n'est possible qu'en C++0x. Pour obtenir du code valide en C++98, il faut simplement ne pas le mettre ; on s'expose alors à quelques inconvénients, lire safe_bool ([Lien43](#)) pour plus d'informations) :

```

template<class T>
class ValuePtr
{
public:
    explicit operator bool() const
    {
        return myPtr;
    }
};

```

Enfin, on pourrait vouloir permettre la conversion depuis notre pointeur intelligent vers un pointeur nu. La méthode pour le faire consisterait à surcharger operator T*() mais ce n'est pas forcément une bonne idée. En effet, cette conversion implicite donnerait un accès direct au pointeur nu, et permettrait de fausses manipulations. Bien qu'un tel accès soit probablement nécessaire dans certains cas, il semble plus judicieux de ne le fournir que par une fonction, ce qui oblige un appel explicite, et permet de s'assurer qu'il s'agit bien d'un choix conscient de l'utilisateur.

2.3.7. Conclusion

On peut voir que si définir un pointeur intelligent n'est pas très compliqué en principe, en définir un bon commence à devenir plus difficile. Et encore, je suis passé à côté de plein d'aspects (multithreading, par exemple). D'où l'intérêt d'avoir à notre

disposition des pointeurs mitonnés aux petits oignons au fil des ans par des experts du domaine.

3. Les pointeurs intelligents standards

3.1. shared_ptr

Le modèle derrière ce pointeur est celui où plusieurs pointeurs permettent l'accès à une même donnée, sans que l'un de ceux-ci soit investi du rôle particulier consistant à être le responsable de la durée de vie de l'objet. Cette responsabilité est partagée (d'où le nom shared) entre tous les pointeurs pointant à un moment donné sur l'objet.

Comment dans ce cas la mémoire sera-elle libérée ? En fait, quand un pointeur est détruit, il vérifie s'il n'était pas le dernier à pointer sur son objet. Si c'est le cas, il détruit l'objet, puisqu'il est sur le point de devenir inaccessible. Afin de connaître cette information, un compteur de référence est associé à l'objet, à chaque fois qu'un nouveau pointeur pointe sur l'objet, le compteur de référence est incrémenté. A chaque fois que le pointeur arrête de pointer sur cet objet (parce qu'il est détruit, ou s'apprête à pointer sur un autre), le compteur est décrémenté. Si le compteur atteint 0, il est temps de détruire l'objet.

Ce fonctionnement assez simple cache certaines subtilités ou usage avancés, que nous allons voir dans la suite de ce chapitre.

3.2.1. Utilisation de base

Comme il est usuel, un shared_ptr est un template ayant comme paramètre le type d'élément sur lequel il doit pointer. Prenons un petit code d'exemple :

```
void f()
{
    shared_ptr<int> a(new int (42));
    cout << a.use_count() << endl;
    shared_ptr<int> b = a;
    cout << a.use_count() << endl;
    {
        shared_ptr<int> c;
        shared_ptr<int> d (new int (314));
        cout << a.use_count() << endl;
        c = a;
        cout << a.use_count() << endl;
        d = c;
        cout << a.use_count() << endl;
    }
    cout << a.use_count() << endl;
    b.reset();
    cout << a.use_count() << endl;
}
```

La fonction use_count affiche la valeur du comptage de référence d'un shared_ptr. Elle ne sert qu'à des fins de débogage, ou d'explication comme ici. Ce programme va commencer par afficher 1, c'est à dire qu'au début, il n'y a qu'un seul pointeur pointant sur l'objet (ici, un entier valant 42). On peut noter que c'est une bonne habitude de stocker dès que possible un pointeur nu dans un pointeur intelligent, afin de s'assurer qu'il n'aura pas l'occasion de fuir. A tel point que le standard définit une fonction make_shared qui évite toute présence d'un pointeur nu dans le code utilisateur, et que l'on aurait pu utiliser ainsi :

```
shared_ptr<int> a = make_shared<int>(42);
```

Cette fonction a aussi l'avantage d'être plus sûre (voir plus loin) et d'offrir un gain de performances par rapport au code précédent. Seul problème : Elle demande des techniques spécifiques à C++0x

pour pouvoir être implémentée, et il faut donc apprendre encore quelque temps à s'en passer.

Revenons à notre premier exemple. La deuxième ligne affiche 2 : On a construit un autre shared_ptr qui pointe sur la même zone. La troisième ligne affiche toujours 2 : On a construit un pointeur c sur un entier, mais sans l'initialiser. Un tel pointeur ne pointe sur rien. C'est un peu l'équivalent d'un pointeur nul, pour des shared_ptr. De même, le pointeur d pointe sur un autre entier. Il n'y a donc aucune raison de modifier le comptage de référence vers notre entier 42.

Par contre, à la ligne suivante, ce comptage passe à 3 : On a fait pointer c vers le même objet, le comptage augmente donc. À la ligne suivante, le compteur est encore incrémenté, puisque d pointe désormais aussi sur notre objet. À cette occasion, d s'est arrêté de pointer sur l'entier valant 314, et le comptage de référence de cet objet a donc diminué. Comme il a atteint 0 (il n'y avait que d pour pointer dessus), cet objet a donc été détruit.

Quand on sort du scope, les variables c et d sont détruites, notre comptage de référence redescend donc à 2. Puis, on appelle b.reset(), qui a le même effet que si on avait écrit b = shared_ptr<int>(); b ne pointe plus sur rien après cet appel. La dernière ligne que l'on affiche est donc 1. A la fin de la fonction, a sera détruit, faisant passer le comptage de référence à 0, et l'entier valant 42 sera détruit.

3.1.2. Compatibilité entre shared_ptr

Il existe un certain nombre de conversions possibles entre des pointeurs nus. Les conversions équivalentes existent entre les shared_ptrs. En voici quelques exemples :

```
class A { /*...*/ };
class B : public A { /*...*/ };

B* b1(new B);
A* a(b1);
B* b2 = dynamic_cast<B*>(a);

shared_ptr<B> sb1(new B);
shared_ptr<A> sa(sb1);
shared_ptr<B> sb2 = dynamic_pointer_cast<B>(sa);
```

On peut noter en particulier que le mot clef dynamic_cast est remplacé pour les shared_ptr par une fonction dynamic_pointer_cast qui s'utilise de manière semblable (sauf que le paramètre template est B, et non pas shared_ptr), et qui retourne un pointeur ne pointant sur rien si la conversion n'a pu avoir lieu. Il existe de même une fonction static_pointer_cast et mimant l'effet d'un static_cast pour un pointeur nu.

3.1.3. Risque de fuite mémoire : Initialisation

Il y a deux cas où malgré l'utilisation de shared_ptrs, on a toujours des fuites mémoires. Le premier est lié non pas aux shared_ptrs, mais au fait que, temporairement, la mémoire n'a pas été gérée par des shared_ptrs. Soit le code suivant, a priori bien intentionné :

```
int f(shared_ptr<int> i, int j);
int g();

f(shared_ptr<int> (new int (42)), g());
```

Ce code respecte bien le conseil d'enrober au plus vite un pointeur nu dans un pointeur, mais il présente un risque de fuite mémoire. En effet, le compilateur doit effectuer 4 actions quand il voit cette ligne :

1. Créer un entier valant 42 dans une nouvelle zone

- mémoire allouée pour l'occasion par new
- 2. Créer un `shared_ptr` avec le pointeur obtenu à l'étape 1
- 3. Appeler la fonction `g`
- 4. Appeler la fonction `f`

Il y a quelques contraintes d'ordre entre ces opérations, par exemple 2 ne peut avoir lieu qu'après 1, et 4 qu'à la fin. Par contre rien n'empêche le compilateur de choisir l'ordre : 1 3 2 4.

Dans ce cas, si l'appel de `g` lance une exception, comme le `shared_ptr` n'a pas encore eu le temps de prendre possession de la mémoire fraîchement allouée, il n'aura pas la possibilité de la libérer. A ce problème, deux solutions :

Soit ne jamais créer de `shared_ptr` temporaires, en remplaçant le code précédent par :

```
int f(shared_ptr<int> i, int j);
int g();

shared_ptr<int> si (new int (42));
f(si, g());
```

Soit ne pas allouer de pointeur nu du tout, ce qui évite qu'ils puissent fuir (il faut le C++0x pour pouvoir faire ça) :

```
int f(shared_ptr<int> i, int j);
int g();
f(make_shared<int>(42), g());
```

3.1.4. Risque de fuite mémoire : Cycle

Le plus gros problème à l'utilisation d'un `shared_ptr` est probablement qu'il ne gère pas correctement les cycles. Considérons le code suivant (dans du vrai code, les cycles seraient évidemment moins faciles à détecter) :

```
class A
{
    // ...
    shared_ptr<B> myB;
};

class B
{
    // ...
    shared_ptr<A> myA;
};

shared_ptr<A> a = new A;
shared_ptr<B> b = new B;
cout << a.use_count() << ", " << b.use_count() << endl;
a->myB = b;
cout << a.use_count() << ", " << b.use_count() << endl;
b->myA = a;
cout << a.use_count() << ", " << b.use_count() << endl;
a.reset();
b.reset();
```

Si l'on s'intéresse aux compteurs de référence de nos objets, on constate qu'à la première ligne, ils valent tous deux 1, rien de surprenant jusque là. À la seconde ligne d'affichage, ils valent respectivement 1 et 2. En effet, deux pointeurs pointent sur l'objet de type B : `b`, et `a->myB`. À la troisième ligne d'affichage, les valeurs sont toutes deux de 2.

Après les resets, les objets A et B sont devenus inaccessibles depuis notre code, mais `myB` dans l'objet de type A fait survivre l'objet de type B en maintenant son comptage de référence à 1, et `myA` dans ce dernier fait survivre l'objet de type A. Nos deux objets se font donc survivre mutuellement. Nous avons une fuite mémoire. C'est le principal défaut des `shared_ptr`s par rapport à un système comme un glaneur de cellules. La classe `weak_ptr`, qui

fait l'objet du chapitre suivant, permet de s'en sortir dans ce genre de situations.

3.1.5. Obtenir un shared_ptr à partir de this

Il y a deux cas où l'on veut obtenir un `shared_ptr` à partir du pointeur nu `this`. Le premier, c'est dans une fonction membre d'une classe, si on a besoin d'appeler sur nous-même une fonction externe à la classe qui prend en paramètre un `shared_ptr` (pour que ça marche, il faut bien entendu que l'on soit certain que l'objet courant est déjà géré par `shared_ptr`).

Il n'y a pas de solutions non intrusives à ce problème. Il faut modifier la classe afin qu'elle sache d'elle-même comment retrouver le `shared_ptr` lui correspondant. Pour ça, on fait dériver la classe d'une classe nommée `enable_shared_from_this` :

```
class A;
void registerA(shared_ptr<A> ptr);

class A : public enable_shared_from_this<A>
{
    void f()
    {
        registerA(shared_from_this());
    }
};
```

On remarque que `enable_shared_from_this` est un template, qui prend en paramètre template la classe même dans laquelle on veut obtenir cette fonctionnalité.

Le second cas est un peu particulier, c'est quand on veut obtenir la même chose, mais dans le constructeur de la classe. Le problème, c'est que lors de sa construction, l'objet n'est pas encore construit, et donc ne peut pas encore avoir été associé à un `shared_ptr`.

Reprenons l'exemple précédent :

```
class A;
void registerA(shared_ptr<A> ptr);

class A
{
public:
    A()
    {
        registerA(???);
    }
};
```

La solution à ce problème est de modifier légèrement l'architecture du code, afin de passer par une factory au lieu de vouloir faire le travail dans un constructeur.

```
class A;
void registerA(shared_ptr<A> ptr);

class A
{
private:
    A() {}
public:
    static shared_ptr<A> create()
    {
        shared_ptr<A> result(new A);
        registerA(result);
        return result;
    }
};
```

3.1.6. Spécifier une fonction de désallocation

Il y a des cas où l'on n'a pas créé un objet à l'aide de `new`, mais à l'aide d'une autre fonction. Par exemple, supposons que l'on utilise une bibliothèque qui nous permette de manipuler des `Toto*`, que l'on obtient et détruit à l'aide des fonctions suivantes :

```
Toto* createToto();  
void deleteToto(Toto *t);
```

Si l'on écrit un simple `shared_ptr<Toto>`, quand il va vouloir détruire l'objet, il va appeler `delete` dessus, et non pas la fonction spécifique `deleteToto`. Est-ce à dire que l'on doit se passer de la sûreté et du confort des `shared_ptr` dans ce cas ? Heureusement non, les `shared_ptr` ont un second argument dans leur constructeur qui permet de spécifier comment l'objet géré doit être détruit :

```
shared_ptr<Toto> createSafeToto()  
{  
    return shared_ptr<Toto> (createToto(),  
    &deleteToto);  
}
```

3.1.7. Aliasing

Il peut arriver qu'un utilisateur souhaite manipuler une sous partie d'un objet par `shared_ptr`. Supposons une classe :

```
struct Voiture  
{  
    vector<Roue> mesRoues;  
};
```

Supposons que nos voitures soient gérées par l'intermédiaire des `shared_ptr`, et intéressons nous au code qui travaille avec les roues. On a envie que ce code influe sur la durée de vie d'une voiture, et que tant que ce code possède une roue, la voiture à laquelle la roue est attachée ne soit pas détruite.

On peut envisager plusieurs façons d'y parvenir. Par exemple, la roue peut avoir un `shared_ptr` sur sa voiture (attention aux boucles !). Une autre solution serait que le code gérant les roues prenne en paramètre des `shared_ptr<Voiture>`, mais ce n'est pas très propre car ça introduit un couplage inutile entre ce code et le code de voiture (que faire par exemple si je veux utiliser le même code sur des roues en vrac non encore montées ?).

Finalement, les `shared_ptr` présentent une autre option, plus claire, à ce problème. L'idée est de spécifier un `shared_ptr` sur une sous-partie de l'objet mais qui partage son comptage de référence avec l'objet dans sa totalité.

```
struct Voiture  
{  
    vector<Roue> mesRoues;  
};  
  
shared_ptr<Roue> getRoue(shared_ptr<Voiture> const &v,  
int id)  
{  
    // Premier argument : Où est mon comptage de  
référence  
    // Second argument : Où est-ce que je pointe  
    return shared_ptr<Roue>(v, &v->mesRoues[id]);  
}
```

3.1.8. Inconvénients des shared_ptr

Le premier inconvénient des `shared_ptr`, c'est qu'ils ne gèrent pas les cycles, on en a parlé.

Un second inconvénient peut être les performances. Sur deux points, ils sont moins performants que des pointeurs nus (mais ils apportent plus de choses, donc il ne faut pas conclure trop vite) :

- Il y a besoin d'espace supplémentaire pour gérer le comptage de référence. De plus, cet espace peut demander une allocation mémoire supplémentaire lors de la création, sauf si la fonction `make_shared` (C++0x) est utilisée.
- Chaque opération de copie du pointeur doit incrémenter le compteur de référence, ce qui peut être assez coûteux dans un contexte multithread. Comparé à un glaneur de cellule, le coût est probablement comparable, selon l'usage qui en est fait, mais cette opération a lieu pendant l'opération elle-même, et non pendant un moment où le système attend, ce qui est plus perceptible. La `move semantic` offerte par C++0x (et qui commence à être disponible dans certains compilateurs (gcc par exemple) peut nettement réduire le nombre de ces coûteuses opérations de copie. En attendant cette fonctionnalité, certaines implémentations ont spécialisé certaines classes (comme `vector<shared_ptr<T>>` pour Visual C++/TR1) afin de minimiser ces copies dans des cas d'utilisation très courants.

3.2. weak_ptr

`weak_ptr` a été conçu spécifiquement pour travailler en collaboration avec `shared_ptr`, pour casser les cycles. L'idée de base est de dire qu'en fait, parmi les pointeurs sur un objet, certains (les `shared_ptr`) se partagent la responsabilité de faire vivre ou mourir ce dernier, le possèdent, et d'autres (les `weak_ptr`) y ont un simple accès, mais sans aucune responsabilité associée.

Par exemple, imaginons une base de données géographiques. On pourrait avoir une classe `Région` et une classe `Département`. Une région contiendrait une liste de départements, et un département aurait besoin de savoir dans quelle région il est situé. Si l'on n'y prend pas garde, on a donc un lien cyclique entre régions et départements. Dans ce cas, on pourra décider que la région connaît ses départements sous forme de `shared_ptr`, et que le département ne possède qu'un `weak_ptr` sur sa région.

Un `weak_ptr` n'impacte donc pas le comptage de référence de l'objet sur lequel il pointe. Mais quel avantage peut-il bien apporter par rapport à un simple pointeur nu qui pointerait sur le même objet qu'un `shared_ptr` ? La sécurité. En effet, au moment où le compteur de référence de l'objet passe à 0, ce dernier est détruit. Si un autre pointeur essaye alors d'accéder à cet objet, c'est un comportement indéfini. Avec un `weak_ptr`, l'utilisateur est averti que l'objet pointé est désormais mort, et qu'il doit se passer de lui. Nous allons voir comment.

3.2.1. Utilisation de base

Il est possible de créer un `weak_ptr` soit à partir d'un `shared_ptr`, soit à partir d'un autre `weak_ptr`. Contrairement à `shared_ptr`, il n'est pas possible d'en créer un à partir d'un pointeur nu, un `weak_ptr` n'étant là que pour travailler en collaboration avec un `shared_ptr`.

La première grosse différence que l'on voit entre les deux types de pointeurs à l'utilisation, c'est que `weak_ptr` n'est pas vraiment un pointeur intelligent, il lui manque en effet un surcharge de l'opérateur `->`.

Le mode d'emploi pour accéder à l'objet pointé est de créer un `shared_ptr` à partir du `weak_ptr`, et de travailler à partir de ce nouveau pointeur. Pour ça, deux méthodes sont possibles, le constructeur de `shared_ptr` prenant un `weak_ptr` en paramètre et la fonction `lock` (qui est souvent plus explicite):


```

void f(weak_ptr<T> weakData)
{
    shared_ptr<T> sharedData = weakData.lock();
    // ou : shared_ptr<T> sharedData(weak_data);
    if (sharedData)
    {
        sharedData->onPeutTravaillerAvec();
    }
    else
    {
        // l'objet pointé a été détruit
    }
}

```

Pour quelle raison cette obligation ? Imaginons l'espace d'un instant qu'on puisse travailler directement avec un `weak_ptr` :

```

void f(weak_ptr<T> weakData)
{
    if (weakData.expired())
    {
        // l'objet pointé a été détruit, c'est le
        // rôle de la fonction expired de nous en
        avertir
    }
    else
    {
        weakData->onPeutTravaillerAvec(); // ?
    }
}

```

Blogs C++

Thinking in C++ est désormais disponible en français !

Bonjour,
Après un travail de titan effectué par des membres de la rédaction ainsi que des contributeurs du site Developpez.com, je peux enfin vous annoncer la publication de la traduction de **Thinking in C++** volume 1.

Pour ceux qui ne connaissent pas, il s'agit d'un livre disponible en ligne qui permet d'apprendre le C++ de manière très correcte. Beaucoup de livres sont basés sur une ancienne approche qui est banie par les experts C++ de nos jours, c'est à dire présentation du C, apprentissage du C, présentation du C++, apprentissage du C++, ou similaire. Celui-ci enseigne le C++ comme un langage tout à fait différent du C (ce qui est réellement le cas !) et ne part pas sur

Blogs Qt

Qtopia devient Qt Extended pour la version 4.4

Bonjour,
Si vous suivez un peu l'actualité du côté de Qt, vous avez sûrement remarqué le nouveau site ([Lien46](#))... Mais avez-vous suivi l'actualité de la plateforme de création d'applications graphiques pour appareils mobiles nommée Qtopia ?



Qtopia, la plateforme pour le développement sur appareils mobiles, a été renommée en **Qt Extended** et a sorti sa version 4.4 !

Les principales fonctionnalités offertes dans cette version sont :

Dans un environnement multithread, ce code qui semble pourtant bien intentionné peut avoir un problème si l'objet est détruit entre le moment où on teste `expired` et le moment où l'on appelle la fonction.

Qu'à cela ne tienne, diront certains, il suffit que l'opérateur `->` teste si l'objet existe encore, et lance une exception si ce n'est pas le cas, en mettant les verrous qui vont bien pour que l'objet survive le temps que l'on appelle l'opérateur `->`. Mais même là, ça ne marche pas : Au moment où la fonction `onPeutTravaillerAvec()` est appelée, le code de l'opérateur `->` a totalement fini de s'exécuter, et rien n'empêche donc l'objet d'être détruit alors qu'on travaille dessus.

Le passage par un `shared_ptr` permet donc, si le comptage de référence est sur le point de passer à 0 dans un autre thread, de prolonger la durée de vie de l'objet pointé le temps qu'on en ait fini avec lui.

En pratique, l'utilisation de `expired`, tout comme de `use_count` est très rare, et, sauf pour du code de test ou de débogage indique probablement une mauvaise compréhension du fonctionnement du code en multithread.

Retrouvez la suite de l'article de Loïc Joly en ligne : [Lien44](#)

le C. Il présente énormément de choses et ce de manière très très correcte, voir même très bonne.

L'équipe C++ de Developpez.com vous recommande vivement de lire la traduction que nous avons réalisée, avec comme pour la plupart de nos cours et articles une version PDF (plus de 400 pages) ainsi qu'une version HTML hors-ligne.

Nous sommes donc heureux de pouvoir vous annoncer enfin l'existence d'un cours sur le C++ en **français** sur lequel nous sommes à l'unanimité d'accord pour dire qu'il a une très bonne approche et qu'il est **celui à conseiller** à tout **débutant** en C++.

La liste des personnes à remercier est très longue mais n'oublions pas le travail qu'ils ont accompli et ce pour fournir enfin cet excellent cours.

Je vous souhaite à tous une bonne lecture.

Retrouvez ce billet sur le blog d'Alp Mestan : [Lien45](#)

- une architecture modulaire afin de pouvoir sélectionner les fonctionnalités dont on a besoin pour notre application
- une interface utilisateur nettement améliorée
- un système de communication par IP basé sur Telepathy
- une unique boîte de réception pour tous les types de messages reçus : SMS, MMS, e-mail, ...
- un nouvel outil pour tester le système concerné par votre application

Voilà donc de quoi intéresser les développeurs pour appareils mobiles.

Pour en savoir plus, rendez-vous sur le site de Trolltech/Nokia : ici ([Lien47](#))

Retrouvez ce billet sur le blog d'Alp Mestan : [Lien48](#)

Les derniers tutoriels et articles

Personnalisation du ruban: Les fonctions d'appel Callbacks

Ce tutoriel présente les fonctions d'appel Callbacks sous Excel2007.

1. Introduction

Le tutoriel précédent (La personnalisation du ruban ([Lien49](#))), présentait les attributs et les contrôles utilisables dans vos projets. Ce nouvel article décrit les fonctions d'appel VBA (Callbacks).

Une partie des attributs contenus dans le fichier xml de personnalisation peuvent être associés à des fonctions d'appel VBA. Les paramètres définis dans votre code xml sont alors liés à des procédures placées dans un module standard du classeur. Vous pouvez déclencher une macro depuis le ruban (à l'aide des événements `onAction`, `onChange` ...), mais également gérer dynamiquement, paramétrer et modifier les attributs des contrôles par VBA, en leur ajoutant le préfixe **get**.

La caractéristique d'un attribut **getNomAttribut** est de pouvoir mettre à jour dynamiquement sa propriété `NomAttribut`.

Par exemple, au lieu d'indiquer le paramètre `itemHeight="LaValeur"` explicitement en dur dans le fichier xml de personnalisation, vous écrirez `getItemHeight="NomMacroDefinitionHauteurItem"`.

Ensuite, il vous restera à placer la fonction d'appel dans un module standard du classeur:

```
Sub NomMacroDefinitionHauteurItem(control As IRibbonControl, ByRef returnedVal)
```

Les chapitres suivants décrivent les fonctions disponibles et proposent quelques exemples d'utilisation.

Remarques :

Bien entendu les Callbacks ne fonctionnent pas si vous désactivez les macros à l'ouverture du classeur.

Attention à bien respecter la casse lorsque vous rédigez vos codes dans le fichier xml de personnalisation.

2. Description des arguments contenus dans les fonctions d'appel

Les fonctions sont constituées d'arguments qui récupèrent des informations dans le ruban ou y renvoient des paramètres.

Voici une description des éléments présents dans les fonctions Callbacks:

L'argument **control**.

Exemple: `Sub ___OnAction(control As IRibbonControl)`

Cet argument identifie le contrôle manipulé par VBA. Il possède 3 propriétés en lecture seule:

* **id** est l'identifiant unique qui permet d'identifier chaque contrôle. Si vous attribuez la même procédure à plusieurs

contrôles du fichier xml de personnalisation, l'id vous permet de repérer lequel est en cours d'utilisation.

* **tag** est l'information personnelle complémentaire du contrôle, que vous avez défini dans le fichier xml.

* **context** (Je n'ai pas réussi à le faire fonctionner).

Le mot clé **Byref**.

Exemple : `Sub ___GetEnabled(control As IRibbonControl, ByRef enabled)`

Lorsqu'un argument est précédé du mot clé `ByRef`, cela signifie que la donnée spécifiée dans votre fonction va être utilisée pour mettre à jour l'attribut du contrôle.

Quand la procédure est appelée par l'application, un objet `IRibbonControl` représentant le contrôle est défini. Le code vérifie la propriété 'id' de l'objet et en fonction de sa valeur, il assigne une donnée à la variable. Par exemple : `enabled = True`.

L'argument **pressed**.

Exemple : `Sub ___OnAction(control As IRibbonControl, pressed As Boolean)`

Cet argument renvoie la valeur `Vrai` si le contrôle est coché (pour les `checkBox`) ou si la touche est enfoncée (pour les `toggleButton`). La valeur `Faux` est renvoyée dans le cas contraire.

L'argument **index**.

Exemple : `Sub ___GetItemLabel(control As IRibbonControl, index As Integer, ByRef label)`

Cet argument définit le numéro de l'élément dans les contrôles de type menu déroulant (`comboBox`, `dropDown`, `Gallery`).

0= le premier élément dans la liste

1= le deuxième élément

... etc ...

L'argument **Id**.

Exemple : `Sub NomProcedure(control As IRibbonControl, Id As String, index As Integer)`

Cet argument renvoie l'identificateur unique de l'item sélectionné (pour les contrôles `dropDown` et `Gallery`).

L'argument **text**.

Exemple : `Sub ___OnChange(control As IRibbonControl, text As String)`

Renvoie la donnée contenue dans les contrôles de saisie (`editBox`, `comboBox`).

3. Les fonctions d'appel

3.1. onLoad

La fonction **onLoad** s'applique au contrôle **customUI**.

Elle définit la procédure VBA qui doit être déclenchée lors du

chargement du ruban.

Il est possible d'appliquer des actualisations à n'importe quel moment par macro, grâce à la fonction de rappel **onLoad**.

Ce rappel est déclenché une seule fois, après l'ouverture du classeur. Celui-ci va charger une variable objet qui déclarée avec un type **IRibbonUI** et sera placée dans un module standard.

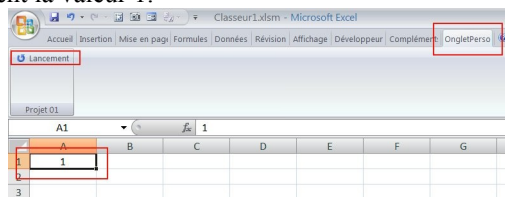
Par exemple : `Public objRuban As IRibbonUI`.

L'objet **IRibbonUI** possède deux méthodes :

Invalidate() qui actualise en une seule fois tous les contrôles personnalisés du classeur.

InvalidateControl(ControlID As String) qui actualise un contrôle particulier (ControlID correspond à l'identificateur unique du contrôle).

Cet exemple active un bouton de manière conditionnelle. Le contrôle est utilisable uniquement si le contenu de la cellule A1 contient la valeur 1.



Placez ce code dans le fichier xml de personnalisation :

Xml

```
<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/cust
omui" onLoad="RubanCharge">
<!-- onLoad="RubanCharge" est déclenché lors du
chargement du ruban personnalisé. -->
<ribbon startFromScratch="false">
<tabs>
<tab id="OngletPerso" label="OngletPerso"
visible="true">
<group id="Projet01" label="Projet 01">
<!-- onAction="ProcLancement" définit la macro
déclenchée lorsque vos cliquez sur le bouton. -->
<!-- getEnabled="Bouton1_Enabled" gère la
condition d'activation ou de désactivation. -->
<button id="Bouton1" label="Lancement"
onAction="ProcLancement" size="normal"
imageMso="Repeat"
getEnabled="Bouton1_Enabled"/>
</group>
</tab>
</tabs>
</ribbon>
</customUI>
```

Ouvrez le classeur Excel. Validez l'éventuel message d'erreur "Impossible d'exécuter la macro 'RubanCharge'...". Ce message est normal car le classeur ne contient pas encore les fonctions de rappel.

Placez ce code dans un module standard :

Vba

Option Explicit

```
Public objRuban As IRibbonUI
Public boolResult As Boolean
```

```
'Callback for customUI.onLoad
```

```
'Est déclenché lors du chargement du ruban
personnalisé.
```

```
Sub RubanCharge(ribbon As IRibbonUI)
boolResult = False
Set objRuban = ribbon
End Sub
```

```
'Callback for Bouton1 onAction
```

```
'La procédure déclenchée lorsque vous cliquez sur le
bouton.
```

```
Sub ProcLancement(control As IRibbonControl)
MsgBox "ma procédure."
End Sub
```

```
'Callback for Bouton1 getEnabled
```

```
'Active ou désactive le bouton en fonction de la
variable boolResult
```

```
Sub Bouton1_Enabled(control As IRibbonControl, ByRef
returnedVal)
returnedVal = boolResult
End Sub
```

Et placez ce code dans le module objet Worksheet ([Lien50](#)) de la feuille (où vous allez modifier le contenu de la cellule A1):

Vba

Option Explicit

```
'Evenement Change dans la feuille de calcul.
```

```
Private Sub Worksheet_Change(ByVal Target As Range)
'Vérifie si la cellule A1 est modifiée et si la
cellule contient la valeur 1.
```

```
If Target.Address = "$A$1" And Target = 1 Then
boolResult = True
```

```
Else
boolResult = False
```

```
End If
```

```
'Rafraichit le bouton personnalisé
```

```
If Not objRuban Is Nothing Then
objRuban.InvalidControl "Bouton1"
End Sub
```

Refermez puis ré-ouvrez le classeur (les macros doivent impérativement être activées).

La variable "objRuban" va être chargée depuis la procédure "RubanCharge" : `Set objRuban = ribbon`

Si les conditions sont ensuite remplies (en fonction des modifications dans la cellule A1), le contrôle "Bouton1" va être activé ou désactivé.

Rappel:

Pour les contrôles de type comboBox, gallery et dynamicMenu, vous pouvez insérer l'attribut **invalidateContentOnDrop** dans le fichier xml de personnalisation afin d'actualiser automatiquement le contenu du contrôle lorsque celui-ci est sélectionné.

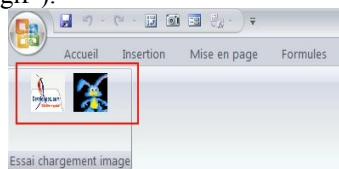
3.2. loadImage

La fonction **loadImage** s'applique au contrôle **customUI**.

Elle définit la procédure VBA qui va charger des images. De cette manière, toutes les images sont récupérées en une seule fois, au moment où vous activez un élément personnalisé de votre ruban.

Cet exemple suppose qu'il existe des images nommées "logo16.gif" et "lapin4.gif" dans le répertoire "C:\dossier\". Les

noms de fichiers sont spécifiés dans l'attribut image (image="logo16.gif").



Placez ce code dans le fichier xml de personnalisation :

Xml

```
<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/cust
omui"

    loadImage="MacroChargementImage">

<ribbon>
<tabs>
<tab id="Tab01" label="Test" >
<group id="Groupe01" label="Essai chargement
image">

<button id="button01"
size="large"
image="logo16.gif"/>

<button id="button02"
size="large"
image="lapin4.gif"/>

</group>
</tab>
</tabs>
</ribbon>

</customUI>
```

Et la fonction de chargement d'image, à placer dans un module standard :

Vba

```
'Callback for customUI.loadImage
Sub MacroChargementImage(imageID As String, ByRef
returnedVal)
Set returnedVal = LoadPicture("C:\dossier\" &
imageID)
End Sub
```

3.3. getEnabled

La fonction **getEnabled** s'applique aux contrôles **command**, **button**, **checkBox**, **comboBox**, **dropDown**, **dynamicMenu**, **editBox**, **gallery**, **labelControl**, **menu**, **splitButton**, **toggleButton**.

Elle spécifie une valeur booléenne qui indique si le contrôle doit être activé (true ou 1) ou désactivé (false ou 0).

Cet exemple active un bouton personnalisé uniquement si la cellule A1 contient la valeur 1.

Dans le fichier xml de personnalisation :

Xml

```
<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/cust
omui" onLoad="RubanCharge">
<!-- onLoad="RubanCharge" est déclenché lors du
chargement du ruban personnalisé. -->
<ribbon startFromScratch="false">
<tabs>

<tab id="OngletPerso" label="OngletPerso"
visible="true">
<group id="Projet01" label="Projet 01">
```

```
<!-- onAction="ProcLancement" définit la macro
déclenchée lorsque vos cliquez sur le bouton. -->
<!-- getEnabled="Bouton1_Enabled" gère la
condition d'activation ou de désactivation. -->
<button id="Bouton1" label="Lancement"
onAction="ProcLancement" size="normal"
imageMso="Repeat"
getEnabled="Bouton1_Enabled"/>

</group>
</tab>

</tabs>
</ribbon>
</customUI>
```

Dans un module standard :

Vba

Option Explicit

```
Public MonRuban As IRibbonUI
Public boolResult As Boolean
```

```
'Callback for customUI.onLoad
'Est déclenché lors du chargement du ruban
personnalisé.
Sub RubanCharge(ribbon As IRibbonUI)
boolResult = False
Set MonRuban = ribbon
End Sub
```

```
'Callback for Bouton1 onAction
'La procédure déclenchée lorsque vous cliquez sur le
bouton.
Sub ProcLancement(control As IRibbonControl)
MsgBox "ma procédure."
End Sub
```

```
'Callback for Bouton1 getEnabled
'Active ou désactive le bouton en fonction de la
variable boolResult
Sub Bouton1_Enabled(control As IRibbonControl, ByRef
returnedVal)
returnedVal = boolResult
End Sub
```

Dans le module objet Worksheet de la feuille (où vous allez modifier le contenu de la cellule A1):

Vba

```
'Evenement Change dans la feuille de calcul.
Private Sub Worksheet_Change(ByVal Target As Range)

'Vérifie si la cellule A1 est modifiée et si la
cellule contient la valeur 1.
If Target.Address = "$A$1" Then _
boolResult = Target = 1

'Rafraichit le bouton personnalisé
If Not MonRuban Is Nothing Then
MonRuban.InvalidateControl "Bouton1"
End Sub
```

Les fonctions d'appel peuvent également être appliquées aux commandes prédéfinies.

Un exemple qui désactive la commande "Copier" si le classeur est ouvert en lecture seule :

Xml

```
<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/cust
omui">
  <commands>

    <command idMso="Copy"
      getEnabled="macroGetEnabled"/>

  </commands>
</customUI>
```

Dans un module standard du classeur :

Vba

```
'Callback for Copy getEnabled
Sub macroGetEnabled(control As IRibbonControl, ByRef
returnedVal)
  'Désactive la commande si le classeur est ouvert en
lecture seule.
  returnedVal = Not ThisWorkbook.ReadOnly

  'Nota:
  'Le raccourci clavier (Ctrl+C) n'est pas désactivé.
End Sub
```

3.4. onAction

La fonction **onAction** s'applique à tous les contrôles.

Elle est déclenchée lorsque vous cliquez sur le contrôle afin de l'utiliser.

Vous trouverez de nombreux exemples dans les différents chapitres de ce tutoriel.

Une deuxième utilisation de la fonction **onAction**, consiste à désactiver temporairement et réattribuer des procédures personnelles aux commandes prédéfinies du ruban (**onAction Repurposed**).

Dans ce cas, la procédure contient un deuxième argument : 'cancelDefault'.

Sub NomProcédure(control As IRibbonControl, ByRef cancelDefault).

Pour désactiver la commande prédéfinie, spécifiez **cancelDefault = True**.

Pour réinitialiser la commande, indiquez **cancelDefault = False**.

Cet exemple ajoute un onglet 'test' qui contient un bouton bascule (toggleButton). Si vous cliquez dessus, la commande prédéfinie 'Recherche' (onglet Accueil/groupe Edition) est désactivée. Réutilisez le bouton pour réinitialiser la commande 'Recherche'.

Placez ce code dans le fichier xml de personnalisation du classeur :

Xml

```
<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/cust
omui"
  onLoad="RubanCharge" >
  <commands>
    <command idMso="FindDialogExcel"
onAction="MaRecherchePerso" />
  </commands>
  <ribbon startFromScratch="false">
    <tabs>
      <tab id="Tab01" label="Test" >
        <group id="Groupe01" label="essai de
réattribution d'un contrôle prédéfini">
          <toggleButton id="ToggleButton01"
```

```
imageMso="SheetProtect"
size="large"
getLabel="MacroGetLabel"
onAction="ModifStatutBoutonRecherche" />
    </group>
  </tab>
</tabs>
</ribbon>
</customUI>
```

Dans le module objet "ThisWorkbook" du classeur :

Vba

Option Explicit

```
Private Sub Workbook_Open()
  boolResult = False
End Sub
```

Dans un module standard du classeur :

Vba

Option Explicit

```
Public boolResult As Boolean
Public objRuban As IRibbonUI
```

```
'Callback for customUI.onLoad
  'Est déclenché lors du chargement du ruban
personnalis .
Sub RubanCharge(ribbon As IRibbonUI)
  Set objRuban = ribbon
End Sub
```

```
'Callback for ToggleButton01 onAction
  'R cup re la valeur du bouton bascule puis
d sactive ou r initialise la commande
  'pr d finie 'Recherche'
Sub ModifStatutBoutonRecherche(control As
IRibbonControl, pressed As Boolean)
  boolResult = pressed
  objRuban.Invalidate
End Sub
```

```
'Callback for FindDialogExcel onAction
  'R attribue temporairement la commande 'Recherche'
ou la r initialise
  'en fonction de la valeur de la variable boolResult
(utilisation du toggleButton)
Sub MaRecherchePerso(control As IRibbonControl, ByRef
cancelDefault)
  If boolResult Then
    MsgBox "Le bouton 'Recherche' est
temporairement d sactiv "
    'Vous pouvez  ventuellement ajouter vos
proc dures de
    'recherche personnelles.
  '
  cancelDefault = True
Else
  cancelDefault = False
End If
End Sub
```

```
'Callback for ToggleButton01 getLabel
  'Attribue une  tiquette au bouton bascule en
fonction de sa position
Sub MacroGetLabel(control As IRibbonControl, ByRef
returnedVal)
  If boolResult Then
```



```

        returnedVal = "Bouton 'Recherche' Désactivié"
    Else
        returnedVal = "Bouton 'Recherche' opérationnel"
    End If
End Sub

```

3.5. getVisible

La fonction **getVisible** s'applique aux contrôles **tabSet**, **tab**, **group**, **box**, **button**, **buttonGroup**, **checkBox**, **comboBox**, **dropDown**, **dynamicMenu**, **editBox**, **gallery**, **labelControl**, **menu**, **separator**, **splitButton**, **toggleButton**.

Elle renvoie une valeur booléenne qui indique si le contrôle est visible (true) ou masqué (false).

Dans cet exemple, le bouton "Copier" qui placé dans un onglet personnel sera visible uniquement si le nom de l'utilisateur de la session est "mimi".

Placez ce code dans le fichier xml de personnalisation :

Xml

```

<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/cust
omui">
    <ribbon startFromScratch="false">
        <tabs>
            <tab id="TB01" label="Test">
                <group id="GR01" label="Essais
de bouton">
                    <button idMso="Copy"
size="normal" getVisible="SiUtilisateur" />
                </group>
            </tab>
        </tabs>
    </ribbon>
</customUI>

```

Dans un module standard du classeur:

Vba

```

'Callback for Copy getVisible
Sub SiUtilisateur(control As IRibbonControl, ByRef
returnedVal)
    returnedVal = Environ("username") = "mimi"
End Sub

```

Remarque :

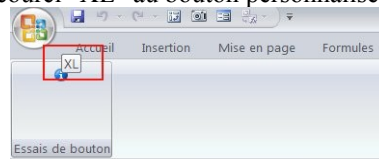
La fonction d'appel **getVisible** ne marche pas sur les onglets prédéfinis si ceux-ci ont préalablement été masqués par l'attribut **startFromScratch="true"**.

3.6. getKeytip

La fonction **getKeytip** s'applique aux contrôles **tab**, **group**, **button**, **checkBox**, **comboBox**, **dropDown**, **dynamicMenu**, **editBox**, **gallery**, **menu**, **splitButton**, **toggleButton**.

Elle définit la procédure VBA qui associera un raccourci clavier au contrôle. Les raccourcis clavier sont composés de 1 à 3 caractères maximum et sont accessibles après l'utilisation de la touche ALT.

Exemple à placer dans le fichier de personnalisation, pour associer le raccourci "XL" au bouton personnalisé :



Xml

```

<customUI
xmlns="http://schemas.microsoft.com/office/2006/01/cust
omui">
    <ribbon startFromScratch="false">
        <tabs>
            <tab id="TB01" label="Test">
                <group id="GR01" label="Essais
de bouton">
                    <button id="bouton01"
imageMso="Refre
shStatus"
onAction="Macro
Bouton"
size="normal"
getKeytip="Defi
nitRaccourci" />
                </group>
            </tab>
        </tabs>
    </ribbon>
</customUI>

```

Dans un module standard :

Vba

```

Option Explicit

'Callback for bouton01 getKeytip
'Définit le raccourci clavier qui va être associé au
bouton.
Sub DefinitRaccourci(control As IRibbonControl, ByRef
returnedVal)
    returnedVal = "XL"
End Sub

'Callback for bouton01 onAction
Sub MacroBouton(control As IRibbonControl)
    MsgBox "Vous avez cliqué sur le bouton '" &
control.id & "'"
End Sub

```

Retrouvez la suite de l'article de SilkyRoad en ligne : [Lien51](#)

Concevez un gestionnaire de Post-It pour vos applications Access

Ce document a pour but de vous montrer comment concevoir un mini gestionnaire de Post-It pour vos applications Access. Au final, vous-même ou un utilisateur êtes informés d'une remarque que vous ou un tiers avez enregistrée par l'apparition d'un message à chaque lancement de l'application ou chargement d'un formulaire en particulier. Vous devez être relativement à l'aise avec Microsoft Access et connaître la conception de formulaires mais également avoir des notions du langage Visual Basic for Application afin mettre en pratique cet exemple.

1. Avant propos

Ce document a pour but de vous montrer comment concevoir un mini gestionnaire de Post-It pour vos applications Access.

Au final, vous-même ou un utilisateur êtes informés d'une remarque que vous ou un tiers avez enregistrée par l'apparition d'un message à chaque lancement de l'application ou chargement d'un formulaire en particulier.

Ce petit plus qui enrichira votre application permettra de laisser un ou plusieurs messages (*en quelque sorte privés*) à un utilisateur de votre choix ou bien un message d'information visant à l'avertir de ne pas oublier d'effectuer telle ou telle opération comme la mise à jour d'une ou plusieurs fiches par exemple...

Pour réaliser ce tutoriel, je me suis intéressé de près au tutoriel de Cafeine : Un formulaire auto-extensible pour Access ([Lien52](#)).

1.1. Niveau

Ce tutoriel s'adresse plus particulièrement aux développeurs intermédiaires avec un bon niveau d'approche de la base de données Access.

La notion de savoir faire en matière de programmation et de conception de formulaires graphiques sont particulièrement requis.

Je vous recommande de lire le tutoriel sur les conventions typographiques ([Lien53](#)) ; cela vous permettra de mieux comprendre la nomenclature que j'adopte systématiquement pour nommer variables et contrôles...

1.2. Versions

Ce tutoriel est applicable pour vos applications qui ont été développées à partir de la version 97 de Microsoft Access (*avec quelques adaptations pour la version 97*).

2. Présentation du projet

Lorsque j'ai créé l'application de "Gestion des traitements PeopleSoft" pour une entreprise dans laquelle j'ai travaillé, j'avais implémenté un *gestionnaire de Post-It* pour que les utilisatrices puissent se souvenir que telle tâche était à effectuer ou bien d'échanger avec des collègues des messages qui ne concernaient que l'application elle-même.

Il faut bien entendu se mettre dans le contexte pour comprendre la philosophie de l'ERP PeopleSoft et ce que les traitements nécessitent comme enrichissement d'informations et éviter d'oublier d'ajouter ou modifier telle ou telle tâche pour un Job donné...

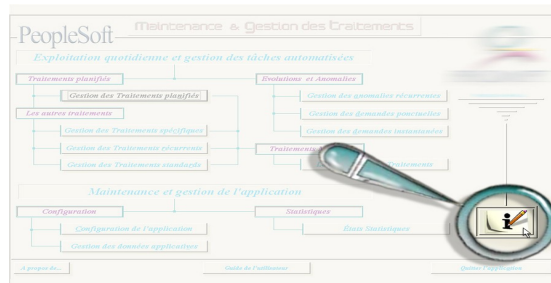
L'idée de mettre ce *Mini Gestionnaire de Post-It* permettait la mise en mémoire de messages pour soi-même ou pour un collègue sans passer par la messagerie interne et ainsi, éviter d'encombrer sa boîte à mails, qui plus est, avec le risque de ne pas les lire.

En effet, le gestionnaire de Post-It proposé ici est obligatoirement affiché pour l'utilisateur ciblé dès le chargement de l'application et ne peut donc pas l'ignorer, si peu que vous ayez tout mis en place pour.

C'est ce que je vais tenter de vous montrer dans ce tutoriel.

2.1. Idée de bouton pour générer un message

Dans l'illustration ci-dessous qui représente un menu principal (*volontairement éclairci et floué*), j'ai mis en évidence le bouton d'appel du générateur de messages. Un clic sur ce bouton exécute l'ouverture du formulaire de rédaction d'un nouveau message...



3. Gestion des messages

Pour mettre en oeuvre ce tutoriel, il est nécessaire de concevoir deux formulaires :

- Un pour la **rédaction des messages** ;
- Un pour **l'affichage des Post-It en mode Popup** à partir du formulaire de votre choix ;

Le formulaire dédié à la rédaction des messages est composé d'une liste déroulante qui contient les utilisateurs potentiels de votre application associée à une case à cocher qui permet de libérer cette liste pour choisir un autre utilisateur que vous-même. Cette option est définie par défaut à l'ouverture du formulaire.

Il est effectivement sous-entendu dans ce projet, que l'affichage des Post-It est dédié plutôt à **vous-même** comme un **pense-bête**. À partir du moment où vous affectez un message à un utilisateur en particulier, c'est cet utilisateur qui verra son application ouverte avec un message au premier plan avant qu'il puisse accéder à l'application.

En revanche, il n'est pas prévu dans ce projet de pouvoir choisir plusieurs destinataires d'un même message mais rien ne vous empêche, et ce sans grosses modifications, de mettre cette option en place auquel cas il faudra adapter le code d'une part et ajouter un contrôle de *Zone de liste* qui se substituera à la *Zone de liste déroulante*.

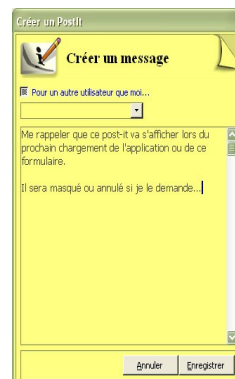
Vous pourrez alors sélectionner un ou plusieurs utilisateurs, rédiger votre message et enfin l'enregistrer.

Le formulaire dédié à l'affichage des messages est un formulaire doté de propriétés particulières en ce qui concerne l'alimentation de la zone de texte puisque celle-ci adapte sa hauteur dynamiquement en fonction du contenu à afficher. Pour ce faire, une procédure qui calcule la hauteur nécessaire en fonction de la police de caractères est appelée systématiquement avant d'ouvrir le formulaire.

Sur ce formulaire, j'ai également ajouté la possibilité de le faire glisser à l'aide de la souris du fait que celui-ci est dépourvu de barre de titre.

3.1. Création du formulaire de rédaction des messages

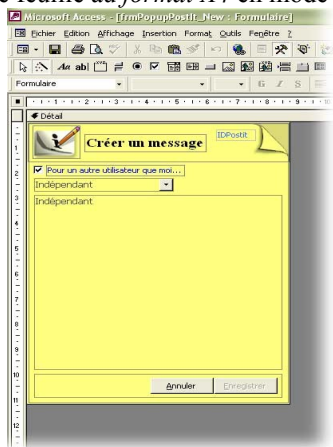
Le formulaire permettant de rédiger des messages est composé de 11 contrôles...



Tant que le message est en cours de rédaction, vous avez la possibilité de choisir un autre utilisateur ou d'annuler la rédaction de celui-ci.

3.1.1. Mode création

Pour réaliser ce formulaire, j'ai choisi une disposition verticale pour rappeler une feuille au format A4 en mode portrait.



En haut du formulaire et en arrière-plan, se trouvent :

- Une *image* représentant une feuille cornée,
- Sur celle-ci, au premier plan vous disposerez une autre *image* représentant l'illustration du bouton du menu principal vu ci-avant,
- Une *étiquette* dans lequel vous saisirez l'intitulé "**Créer un message**"
- Et enfin, un champ *texte* caché dont la *source de contrôle** est l'identifiant du *PostIt*, à savoir **IDPostit**.

* *Source de contrôle* : propriété facultative

Juste en dessous de cette image à quelques pixels, vous poserez :

- Une case à cocher dont l'intitulé est "**Pour un autre utilisateur que moi...**"
- La *liste déroulante*, quant à elle est disposée sur le même bord gauche que la *case à cocher* et contient la liste des utilisateurs avec leur identifiant.

Exemple de requête pour alimenter la liste déroulante :

```
SELECT IDUtilisateur, Prenom & " " & NomFamille AS
NomCompleet
FROM TBLUtilisateurs
WHERE IDUtilisateur <> 123456789;
ORDER BY NomFamille
```

Dans mon application, l'ID qui représente ma personne n'est pas, bien entendu 123456789, celui-ci étant stipulé comme tel pour la forme.

Il vous appartiendra d'adapter la **condition WHERE** en conséquence surtout si l'ID des utilisateurs de votre application est sous la forme

Alphabétique ou *Alphanumérique*.

Cette requête liste tous les utilisateurs potentiellement enregistrés dans la base de données en excluant intentionnellement l'utilisateur **123456789** qui est censé être vous-même à partir du moment où la case est cochée.

Dans la première colonne, c'est-à-dire la **Colonne 0**, l'IDUtilisateur est inscrit et la colonne est **masquée**.

En effet, la propriété **Largeurs colonnes** de la liste déroulante est égale à **0cm;4cm**

C'est ce champ qui identifie l'utilisateur par une valeur unique en faisant office de **Clé primaire**.

Dans la seconde colonne, c'est-à-dire la **Colonne 1**, le champ **NomCompleet** représente la concaténation des champs *Prenom* et *Nom*.

On suppose bien entendu que vous avez fait en sorte d'enregistrer ces informations avec la casse adéquate.

Si ce n'était pas le cas, rien ne vous empêche de forcer la mise en forme des caractères dans la requête elle-même avec respectivement les fonctions *StrConv()* et *UCase()*.

Pour plus d'informations concernant ces deux fonctions, je vous invite à consulter l'aide de Microsoft Access.

Vous dessinerez ensuite la *Zone de texte* devant recevoir le message juste en dessous de cette liste déroulante et encore au-dessous de celle-ci, un *Rectangle* dans lequel vous poserez :

- Un *bouton de commande* intitulé **Annuler**
- Un autre *bouton de commande* intitulé **Enregistrer**

Vous dessinerez enfin un autre **Rectangle** qui lui, détoure à la fois la *zone de texte* et la *zone des boutons* et ferez en sorte de *faire mourir* la **bordure Haut** de cette zone de texte sur la **bordure Bas** de l'image représentant la feuille cornée.



Détail de la disposition des contrôles

Vous disposerez ce rectangle de manière à ce qu'il soit positionné en **arrière-plan** par rapport à tous les autres contrôles comme le montre la figurine ci-dessus...

Retrouvez la suite de l'article de Jean-Philippe Ambrosino en ligne : [Lien54](#)

Est-il possible de récupérer des informations dans un classeur placé sur le Web ?

Les formules de liaison fonctionnent pour lire le contenu des cellules d'un classeur placé sur le Web.

Ici, la fonction renvoie le contenu de la cellule A1

```
='http://monSite/Dossier/[leClasseur.xls]Feuill1'!$A1
```

Comment changer l'icône d'un classeur ?

Cet exemple remplace l'icône Excel (image qui s'affiche dans l'angle supérieur gauche de la fenêtre) par un icône personnalisé, dès l'activation du classeur.

La procédure doit être placée dans le module ThisWorkbook du classeur.

Vba

Option Explicit

```
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" _
    (ByVal lpClassName As String, ByVal lpWindowName As String) As Long
```

```
Private Declare Function SendMessageA Lib "user32" _
    (ByVal hwnd As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
```

```
Private Declare Function ExtractIconA Lib "shell32.dll" _
    (ByVal hInst As Long, ByVal lpszExeFileName As String, _
    ByVal nIconIndex As Long) As Long
```

```
Private Declare Function DestroyIcon Lib "user32.dll" _
    (ByVal hIcon As Long) As Long
```

```
Private Sub Workbook_Activate()
```

```
    Dim Fichier As String
    Dim x As Long
```

```
    'Chemin et nom du fichier icône à afficher
    Fichier = "C:\DELL\DELLSUPPORT.ICO"
    'Vérifie si le fichier existe
    If Dir(Fichier) = "" Then Exit Sub
```

```
    x = ExtractIconA(0, Fichier, 0)
    SendMessageA FindWindow(vbNullString, Application.Caption), _
        &H80, False, x
```

```
    DestroyIcon SendMessageA(FindWindow(vbNullString, Application.Caption), _
        &H80, False, x)
```

```
End Sub
```

```
Private Sub Workbook_Deactivate()
```

```
    Dim Fichier As String
    Dim x As Long
```

```
    Fichier = Application.Path & "\excel.exe"
```

```
x = ExtractIconA(0, Fichier, 0)
```

```
SendMessageA FindWindow(vbNullString, Application.Caption), _
    &H80, False, x
```

```
DestroyIcon SendMessageA(FindWindow(vbNullString, Application.Caption), _
    &H80, False, x)
```

```
End Sub
```

Comment faire référence à un contrôle dans une feuille de calcul ?

Lorsque vous écrivez :

Vba

```
'Modifie le texte sur le bouton "Btn1" de la Feuill1
ThisWorkbook.Worksheets("feuill1").Btn1.Caption = "X"
```

Vous faites référence au **codeName** de la feuille de calcul. c'est l'équivalent de :

Vba

```
Feuill1.Btn1.Caption = "X"
```

Lorsque vous écrivez :

Vba

```
Dim wsht As Worksheet
```

```
Set wsht = ThisWorkbook.Worksheets("feuill1")
wsht.Btn1.Caption = "X"
```

Votre déclaration de variable fait référence à la feuille de calcul (**Dim wsht As Worksheet**) et cela provoque une erreur.

Pour faire fonctionner la procédure dans le deuxième exemple, utilisez :

Vba

```
Dim wsht As Object
```

```
Set wsht = ThisWorkbook.Worksheets("feuill1")
wsht.Btn1.Caption = "X"
```

Pourquoi certaines de mes cellules nommées ne fonctionnent plus sous Excel2007 ?

Excel 2007 permet d'utiliser 16 384 colonnes dans chaque feuille de calcul (pour 256 colonnes dans les anciennes versions). Les entêtes de colonnes vont donc désormais jusqu'à XFD.

Si par exemple, vous utilisiez une cellule nommée **TVA2** sous Excel2003, celle-ci correspond à une référence de cellule sous Excel2007. L'application sait gérer ce cas de figure en faisant précéder automatiquement le nom par un symbole underscore. La cellule nommée deviendra donc **_TVA2** sous Excel2007 et les formules faisant référence à ce nom seront automatiquement mises à jour.

Par contre, il vous restera à modifier vos procédures VBA et toutes les fonctions INDIRECT faisant référence à ce nom.

Retrouvez ces questions et de nombreuses autres sur la FAQ Excel : [Lien55](#)

Comment ouvrir la fenêtre pour ajouter les contacts à un message par VBA ?

Ajouter le code ci-dessous dans un nouveau module, il vous suffit d'exécuter celui-ci depuis un nouveau message. La fenêtre de sélection de contact comme destinataire du message s'ouvrira

```
Sub OpenFenContacts()  
    Dim CBp As Variant  
    Set CBp = ActiveInspector.CommandBars.FindControl(,  
353) 'carnet d'adresse  
    CBp.Execute  
End Sub
```

Est-il possible de personnaliser le menu contextuel "Pièce jointe" d'un message sous Outlook 2007 ?

Oui, il est possible de personnaliser le menu contextuel d'une pièce jointe dans un message, pour cela nous utiliserons une nouveauté d'Outlook 2007 : l'événement **AttachmentContextMenuDisplay** de l'objet **Application** dans le module de classe **ThisOutlookSession**. Cet événement se déclenche avant l'ouverture du menu contextuel correspondant au jeu de collection de pièces jointes.

Pour créer un nouveau contrôle dans ce menu contextuel utiliser le code suivant :

Ce code ne fonctionne que sous Outlook 2007

```
'Déclare l'objet comme jeu d'objets correspondant aux  
pièces jointes sélectionnées  
Dim objAttachments As AttachmentSelection  
  
Private Sub Application_AttachmentContextMenuDisplay( _  
    ByVal CommandBar As Office.CommandBar, _  
    ByVal Attachments As AttachmentSelection)  
    '-----  
    ' Procédure : Application_AttachmentContextMenuDisplay  
    ' Auteur : Dolphy35 -  
    ' http://dolphy35.developpez.com/  
    ' Date : 24/04/2008  
    ' Détail : Création d'un contrôle dans le menu  
    ' contextuel des pièces jointes et  
    ' affecte une macro  
    '-----  
    ' Déclaration de l'objet en tant que bouton de commande  
    Dim objButton As CommandBarButton  
    'Instancie à l'objet la sélection des pièces  
    jointes  
    Set objAttachments = Attachments  
    'Instancie l'objet en tant que nous nouveau  
    contrôle : Bouton  
    Set objButton = CommandBar.Controls.Add( _  
        msoControlButton, , , True)  
    'Paramétrage du nouveau bouton  
    With objButton
```

```
.Style = msoButtonIconAndCaption 'Icône +  
commentaire  
.Caption = "Test" 'Commentaire  
.FaceId = 355 'ID de l'icône du bouton  
.OnAction = "Projet  
1.ThisOutlookSession.InfosPJ" 'Affectation de la  
macro au bouton lors du clic  
End With  
End Sub
```

Lors du clic sur le nouveau contrôle du menu contextuel, vous exécuterez la macro InfosPJ.

```
Sub InfosPJ()  
    '-----  
    ' Procédure : msgtest  
    ' Auteur : Dolphy35 -  
    ' http://dolphy35.developpez.com/  
    ' Date : 24/04/2008  
    ' Détail : Affiche dans une boîte de dialogue le nom  
    ' des pièces jointes  
    ' sélectionnées ainsi que la taille en  
    ' octets  
    '-----  
    ' Déclaration des objets et variables  
    Dim objAttachment As Attachment  
    Dim strTemp As String  
    'initialisation de la variable  
    strTemp = ""  
    'boucle permettant de sortir les pièces jointes du  
    jeu de sélection des pièces jointes  
    For Each objAttachment In objAttachments  
        'Objet pièce jointe  
        With objAttachment  
            strTemp = "Nom : " & .FileName 'nom de  
            la pièce jointe  
            strTemp = strTemp & vbCrLf & "Taille : " &  
            .Size & " octets" 'taille de la pièce jointe  
        End With  
        'affichage des infos de la pièce jointe  
        contenus dans la variable.  
        MsgBox strTemp  
    Next  
End Sub
```

Comment utiliser un fichier HTML comme signature dans Outlook 2007 ?

Contrairement à Outlook 2003, la version 2007 n'intègre pas la possibilité d'ajouter un fichier HTML en tant que signature. Une parade subsiste, il vous suffit de placer ce fichier directement dans le dossier contenant les signatures d'Outlook :

Pour XP ==> C:\Documents and Settings\\"UTI"\Application Data\Microsoft\Signatures\
Pour Vista ==>

C:\Users\\"UTI"\AppData\Roaming\Microsoft\Signatures\
'-----

Retrouvez ces questions et de nombreuses autres sur la FAQ Outlook : [Lien56](#)

Les derniers tutoriels et articles

Indexer avec SQL-Server... oui mais quoi ?

Qu'est ce qu'un index ? Comment en poser ? Quoi indexer ? Quelles colonnes ? Que faut-il indexer ? Où les poser ? Autant de questions qui appellent des réponses que traite cet article...

1. Qu'est-ce qu'un index ?

Dans la vie courante nous sommes entourés d'index : un code postal, le n° d'un immeuble dans une rue, les numéros de téléphone sont des index. Autrefois, lorsque le téléphone fit son apparition (Clémenceau aurait dit du téléphone "Quoi ? On vous sonne comme un laquais ???") il n'y avait pas de numéro. On agita une manivelle qui avait pour effet de réveiller l'opératrice du standard auquel on était physiquement relié. Puis on demandait à cette personne de nous connecter avec Monsieur le Marquis de Carabas à Toledo. Il s'ensuivait un échange de bons procédés entre les opératrices des différents relais afin d'établir une liaison physique entre ces deux clients. Puis vint une première numérotation... Elle était faite ville par ville. On vit alors Fernand Raynaud dans son célèbre sketch demander désespérément le 22 à Asnières et n'obtenir que New York ! Dans les années 50, l'automatique fit son apparition et l'on pouvait appeler en interurbain une personne en composant directement son numéro. Ainsi le commissaire Maigret fut-il joignable à PELLEPORT 38 52. Puis l'automatique fut étendu à toutes les régions de France. Depuis quelques années seulement on peut joindre n'importe qui sur terre avec un simple numéro. Mais les numéros se sont allongés. Le mien, complet, est 00 33 6 11 86 40 66.

Remarquez la structure de ce numéro. Il commence par un double zéro qui indique que l'on va s'intéresser à l'international. Puis le second groupe de chiffres indique le pays. Enfin le suivant indique la nature du réseau (ici le réseaux de téléphones cellulaires). Chacun des groupes de chiffres précise un peu plus les informations.

Une chose importante des index est leur structuration. On convient à l'évidence que si les numéros des immeubles étaient distribués au hasard sur chaque bâtiment[1], il serait difficile (mais pas impossible) de trouver le 68 avenue des Champs Élysés. En fait, il suffirait - au pire - de parcourir toute l'avenue pour trouver le bon bâtiment.

On comprend donc que l'organisation des données dans un index est primordiale pour accélérer les recherches. En fait, les données d'un index sont systématiquement triées et les données stockées dans une structure particulière favorisant les recherches : liste ordonnée ou arbre, le plus souvent.

2. Index et norme SQL

Comme nous le savons tous, le langage SQL est fortement normalisé. La norme la plus actuelle, en cours de finalisation, portera le nom de SQL:2008 (ISO/IEC 9075).

Aussi curieux que cela paraisse, la norme passe sous silence la notion d'index[2]. En effet, ces derniers ne sont que des artifices

destinés à résoudre une problématique physique. Or, SQL ne s'intéresse qu'à des concepts logiques...

Pour autant, la plupart des éditeurs se sont accordés pour définir un ordre pseudo-SQL, CREATE INDEX[3], afin de proposer des méthodes de manipulation des index.

3. Structure logique d'un index

Nous avons dit que les données d'un index sont triées. Si un index est multicolonne, le tri de la clef d'index fait que l'information y est vectorisée. En effet, chaque colonne supplémentaire précise la colonne précédente. De ce fait, la recherche dans un index multicolonne n'est accélérée que si les données cherchées sont un sous-ensemble ordonné du vecteur. Voyons cela en termes pratiques à l'aide d'un exemple....

Exemple pratique :

CLI_NOM	CLI_PRENOM	CLI_DATE_NAISSANCE
DUPONT	Alain	21/01/1930
DUPONT	Marcel	21/01/1930
DUPONT	Marcel	01/06/1971
DUPONT	Paul	21/01/1930
MARTIN	Alain	11/02/1987
MARTIN	Marcel	21/01/1930
...		

Dans cet index composé d'un nom d'un prénom et d'une date de naissance, la recherche sera efficace pour les informations suivantes :

- CLI_NOM
- CLI_NOM + CLI_PRENOM
- CLI_NOM + CLI_PRENOM + CLI_DATE_NAISSANCE

En revanche, la recherche sur une seule colonne CLI_PRENOM ou CLI_DATE_NAISSANCE et a fortiori sur ces deux colonnes n'aura aucune efficacité du fait du tri relatif des colonnes entre elles (vectorisation)... En effet chercher "Paul" revient à parcourir tout l'index (balayage ou *scan*) alors que chercher "MARTIN", revient à se placer très rapidement par dichotomie au bon endroit (recherche ou *seek*).

4. Des index créés automatiquement

La plupart des SGBDR créent automatiquement des index lors de la pose des contraintes PRIMARY KEY et UNIQUE. En effet, le travail de vérification de l'unicité d'une clef primaire ou candidate s'avérerait extrêmement long sans un index.

En revanche la plupart des SGBD relationnels ne prévoient pas de créer des index sous les clefs étrangères (FOREIGN KEY), ni sous les colonnes auto incrémentées. Il y a à cela différentes raisons,

que nous allons expliquer plus en détail un peu plus loin dans cet article...

5. Création manuelle d'un index

Une des syntaxe basique de création d'un index est la suivante :

```
CREATE INDEX #nom_index
ON #nom_schem.#nom_table
( #colonne1 [ { ASC | DESC } ] [, #colonne2
[, ... #colonneN ] ] )
```

Les colonnes spécifiées dans la parenthèse constituent la clef d'index, c'est-à-dire les informations recherchées.

La plupart des SGBDR acceptent des paramètres tels que :

- la clusterisation des données (CLUSTERED, NONCLUSTERED) [4], c'est-à-dire l'imbrication de l'index au sein de la table
- le facteur de remplissage des pages d'index (FILL_FACTOR ou PCT_FREE)
- la nature de la structure de données (TREE, HASH, BITMAP)
- la spécification d'un emplacement physique de stockage

Par exemple, pour un index créer sous SQL Server, on peut utiliser l'ordre suivant :

```
CREATE UNIQUE CLUSTERED INDEX X_CLI_PRENOMDTN
ON S_COM.T_CLIENT (CLI_NOM ASC,
                  CLI_PRENOM,
                  CLI_DATE_NAISSANCE DESC)
INCLUDE (CLI_TEL)
WITH (FILLFACTOR = 80,
      SORT_IN_TEMPDB = ON,
      ALLOW_ROW_LOCKS = ON,
      MAXDOP = 3)
ON STORAGE_IDX
```

Dans cet exemple, un index unique de type CLUSTER, de nom X_CLI_PRENOMDTN est créé sur la table T_CLIENT du schéma S_COM et comporte les colonnes CLI_NOM (ordre ascendant explicite), CLI_PRENOM (ordre ascendant implicite) et CLI_DATE_NAISSANCE (ordre descendant explicite). En sus, les informations de la colonne CLI_TEL sont ajoutées de façon surnuméraire, mais elles ne sont pas indexées, donc pas "triées".

Techniquement cet index est doté d'un facteur de remplissage de 80%, le tri pour le construire devra s'effectuer dans la base tempdb qui sert pour SQL Server à tous les objets temporaires. Enfin cet index est autorisé à faire du verrouillage de ligne et ne peut pas utiliser plus de 3 threads en parallèle pour une même recherche. Pour terminer les données de cet index figureront dans l'espace de stockage STORAGE_IDX.

6. Quels index créer ?

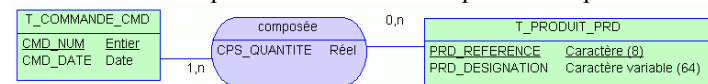
On devrait créer des index derrière toutes les clefs étrangères, afin d'accélérer les jointures ainsi que pour les principales colonnes fréquemment recherchées. Cependant ceci appelle plusieurs remarques...

6.1. Indexation des clefs étrangères

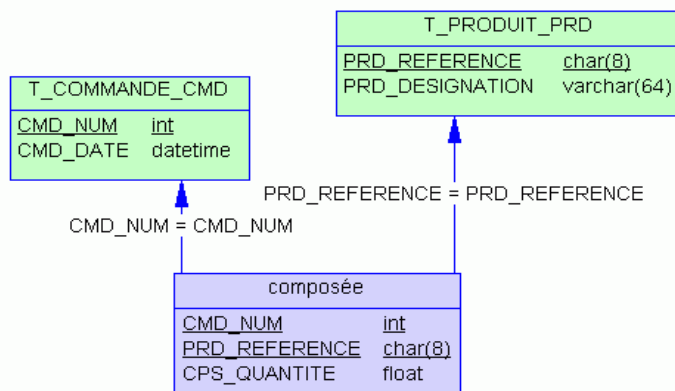
Si l'intention d'indexer les clefs étrangères est bonne, elle n'est pourtant pas nécessaire pour toutes les clefs étrangères.

En effet, pour les clefs étrangères qui naissent d'une association de type 1:1 (un à un) ou de type 1:n (un à plusieurs) ceci est parfait. Il n'en va pas de même pour les tables de jointure qui découlent de la transformation du modèle conceptuel en modèle physique.

Étudions un cas pratique... Soit deux entités représentant des commandes et des produits en association plusieurs à plusieurs :



Le modèle physique qui en découle est le suivant :



Ce modèle conduit à la base de données suivante :

```
create table T_COMMANDE_CMD (
    CMD_NUM int not null
    PRIMARY KEY,
    CMD_DATE datetime not null)
create table T_PRODUIT_PRD (
    PRD_REFERENC char(8) not null
    PRIMARY KEY,
    PRD_DESIGNATION varchar(64) not null)
create table T_J_COMPOSEE_CPS (
    CMD_NUM int not null
    foreign key (CMD_NUM) references T_COMMANDE_CMD (CMD_NUM),
    PRD_REFERENC char(8) not null
    foreign key (PRD_REFERENC) references T_PRODUIT_PRD (PRD_REFERENC),
    CPS_QUANTITE float not null,
    constraint PK_T_J_COMPOSEE_CPS primary key (CMD_NUM, PRD_REFERENC))
```

Dans cette base de données, on a de fait les index suivants à cause des PRIMARY KEYS

- T_COMMANDE_CMD (CMD_NUM)
- T_PRODUIT_PRD (PRD_REFERENC)
- T_J_COMPOSEE_CPS (CMD_NUM, PRD_REFERENC)

Si l'on devait obéir à la règle qui veut que l'on crée systématiquement un index sous toutes les clefs étrangères, il faudrait donc rajouter à cette base, les objets suivants :

```
CREATE INDEX X_CPS_CMD ON T_J_COMPOSEE_CPS (PRD_REFERENC)
CREATE INDEX X_CPS_CMD ON T_J_COMPOSEE_CPS (CMD_NUM)
```

Or, l'un de ces index est totalement inutile parce que redondant. En effet, l'index sur la seule colonne CMD_NUM est déjà inclus dans l'index de clef primaire de cette même table. Il n'y a donc pas lieu de le créer.

En revanche, le second index est parfaitement utile car la vectorisation de l'information contenue dans les clefs d'index rend inefficace la recherche sur la partie PRD_REFERENC dans l'index de clef primaire.

6.2. Indexation des colonnes les plus recherchées

Là aussi, il est inutile de se précipiter sur la création d'index sur chacune des colonnes figurant dans une clause WHERE. En effet,

certaines expressions de recherche pourront activer la recherche dans l'index, d'autres pas.

Les anglophones ont inventé un terme afin de définir qu'un prédicat est "recherchable" ou pas. C'est le terme sargable (Search ARGument).

Voyons cela à l'aide d'un exemple.

Soit la requête :

```
SELECT *
FROM T_CLIENT_CLI
WHERE CLI_NOM LIKE '%ONT'
AND SUBSTRING(CLI_PRENOM, 2, 2) = 'au'
AND MONTH(CLI_DATE_NAISSANCE) = 1
AND YEAR(CLI_DATE_NAISSANCE) = 1930
```

La première expression du filtre WHERE recherche un client dont le nom se termine par ONT. A l'évidence un index sur le nom du client ne sert à rien car les index ordonnent les données dans le sens de lecture des lettres c'est à dire de gauche à droite. Si vous voulez que cette recherche particulière soit "sargable" alors vous pouvez indexer le nom recomposé à l'envers soit à l'aide d'une colonne calculée persistante qui reprend le nom et le reverse, soit utiliser une fonction dans la création de l'index.

Exemple :

```
CREATE INDEX X_CLI_NOM_REVERSE ON S_COM.T_CLIENT_CLI
(REVERSE(CLI_NOM))
```

La seconde expression du filtre WHERE recherche à l'intérieur d'un prénom le motif 'au'. Aucun index conventionnel ne permet de rendre sargable une telle expression. Seul l'implantation de structures de données comme des index rotatifs permet d'être efficace sur ce sujet, mais cela est un autre débat !

La troisième et la quatrième expressions de ce filtre WHERE recherchent les données du mois de janvier de l'année 1930. Dès qu'une fonction est appliquée sur une colonne elle en déforme le contenu et aucun index n'est activable. Or une telle expression peut parfaitement être "sargable" à condition d'exposer la colonne sans aucune déformation de l'information.

Convenons donc que cette expression :

```
AND MONTH(CLI_DATE_NAISSANCE) = 1
AND YEAR(CLI_DATE_NAISSANCE) = 1930
```

Peut être réécrite ainsi :

```
AND CLI_DATE_NAISSANCE BETWEEN '1930-01-01'
AND '1930-01-31'
```

Dès lors, un index sur la colonne CLI_DATE_NAISSANCE pourra être utilisé.

Il y aurait beaucoup de choses encore à dire sur ce qui est indexable et ce qui n'a pas d'intérêt de l'être, ou encore comment transformer des expressions non "sargables" en expressions "cherchables"...

Disons simplement que :

Un index sur de nombreuses colonnes, comme un index dont la clé est démesurée, ou encore un index pour une expression non "cherchable" n'offrent en général aucun intérêt !

Comme tout index coûte en temps de traitement, un index inutile est contre-performant.

7. Index multicolonne et longueur des clés d'index

Comme nous l'avons déjà vu, un index multicolonne ne sera efficace que pour des recherches dans le sens du vecteur constitué

par les colonnes dans l'ordre positionnel. Il est donc inefficace de créer un index multicolonne pour des requêtes qui filtrent alternativement sur l'une ou l'autre colonne exclusivement. Néanmoins lorsque l'on est en mesure de tirer bénéfice d'un index multicolonne, il est intéressant de bien choisir l'ordre des colonnes dans le vecteur.

En effet, on obtiendra une efficacité plus grande en posant en premier les colonnes ayant la plus forte dispersion.

Par exemple s'il faut créer un index sur le sexe, le prénom et le nom, il y a fort à parier que la meilleure combinaison sera nom + prénom + sexe (dans cet ordre précis).

En fait, la dispersion des données est plus forte pour un nom qu'un prénom (il y a moins de prénoms différents que de noms de famille différents) et bien plus encore par rapport au sexe qui ne présente généralement que deux valeurs.

Quant à la longueur de la clé d'index, c'est-à-dire au nombre d'octets qui compose l'information vectorisée, mieux vaut qu'elle soit la plus petite possible. En effet un index dont la clé est petite occupe peu de place et donc permet une recherche bien plus rapide qu'un index de quelques centaines d'octets. A mon sens, on doit s'intéresser à ne jamais faire en sorte que la clé d'index dépasse quelques dizaines d'octets.

Il est à noter que certains développeurs pensent naïvement qu'en créant un index sur une colonne de type "texte" (BLOB) on peut activer une recherche mot par mot. Compte tenu de la structure que nous avons indiquée, il ne sera jamais possible de rechercher un mot précis à l'intérieur des informations d'une colonne par le biais d'un index ordinaire. Cela est en revanche possible si votre SGBDR accepte la création d'index de type "plein texte" (Full text) et donc le prédicat de recherche normatif qui va avec (CONTAINS).

8. Index couvrant

La notion d'index couvrant est intéressante. Arrêtons-nous sur un index classique et regardons comment sont traitées les données.

Soit la requête :

```
SELECT CLI_ID, CLI_NOM, CLI_DATE_NAISSANCE
FROM T_CLIENT_CLI
WHERE CLI_NOM = 'DUPONT'
```

A l'évidence, un index sur la seule colonne CLI_NOM va rendre efficace la recherche. Cependant, cet index ne contient pas les données de CLI_ID [5] ni de CLI_DATE_NAISSANCE. Pour fournir toutes les informations, le moteur SQL est donc dans l'obligation, une fois qu'il a trouvé les clients de nom DUPONT dans l'index, de se reporter vers la table afin de compléter les données avec le CLI_ID et la date de naissance. Cela oblige bien évidemment à lire des données supplémentaires.

Il est possible d'éviter cette double lecture (index + table) en utilisant la notion d'index couvrant.

Un index est dit couvrant, si la seule lecture de l'index suffit à récupérer toutes les informations nécessaires au traitement de la requête.

Ainsi l'index suivant :

```
CREATE INDEX X_CLI_NOMIDN ON T_CLIENT_CLI (CLI_NOM,
CLI_ID, CLI_DATE_NAISSANCE)
```

N'oblige plus à cette double lecture. On économise ainsi du temps et l'optimisation qui en ressort est meilleure.

Certains SGBDR, comme SQL Server, permettent de créer des index dont certaines colonnes sont incluses en redondance bien que ne faisant pas partie de la clef d'index.

Voici un tel index sous SQL Server :

```
CREATE INDEX X_CLI_NOMIDDN ON T_CLIENT_CLI (CLI_NOM)
INCLUDE (CLI_ID, CLI_DATE_NAISSANCE)
```

Retrouvez la suite de l'article de Frédéric Brouard en ligne : [Lien57](#)

La gestion des erreurs en SQL procédural avec MySQL

Depuis la version 5, MySQL supporte la programmation intégrée (ou SQL procédural), qui permet de le rendre très autonome. Nous verrons dans cet article quelles sont les possibilités qui s'offrent à nous pour reporter la programmation sur le SGBD. Je ne prétends pas aborder toutes les possibilités du SQL procédural, mais seulement proposer une des très nombreuses utilisations que l'on peut en faire. Ce tutorial peut constituer une introduction au SQL procédural pour les débutants, et une solution efficace de gestion d'erreurs pour les plus confirmés.

1. Petit rappel du rôle d'un SGBDR

Le SGBD (ou Système de Gestion de Base de Données) est un programme permettant de stocker les informations.

Dans le passé, sans SGBD, nous étions obligés de faire nous-mêmes de la lecture / écriture de fichiers et gérer l'organisation physiques des données.

C'était une tâche très lourde, car avec des données nombreuses, à l'organisation complexe, la création et l'optimisation des algorithmes d'accès aux données étaient difficiles à réaliser. Avec le temps, des programmes ont proposé divers moyens pour optimiser ces tâches, et surtout pour éviter de réinventer la roue à chaque fois.

Le but d'un SGBD est bien entendu de proposer une interface entre les données physiques et une application quelconque. Mais le rôle des SGBDR (Système de Gestion de Base de Données Relationnelles) est bien plus ambitieux : **le contrôle de l'intégrité**. Le contrôle de l'intégrité est une tâche permettant d'assurer la **cohérence des données** au travers de contraintes.

Il nous suffit alors de préciser des règles (les contraintes) qui définissent les valeurs possibles que peuvent prendre telle ou telle colonne. Ces contraintes permettent d'alléger énormément les contrôles à faire du côté applicatif. Il est conseillé d'avoir lu "Limiter la complexité du code applicatif grâce au SGBD" ([Lien58](#)) qui parle de ce type de pratique.

Même si les contraintes permettent une grande liberté d'action, elles sont parfois insuffisantes pour gérer parfaitement l'intégrité de notre domaine de gestion. En effet, l'aspect métier possède très souvent une sémantique propre qui dépasse largement la simple gestion de l'intégrité référentielle. Puisque le rôle du SGBDR est de gérer parfaitement la cohérence des données, il est alors nécessaire de pouvoir y implémenter son propre code et ses propres fonctions afin de contrôler l'information en fonction de notre domaine de gestion.

Ainsi, le SQL intégré (ou SQL procédural) permet de programmer le SGBD pour réaliser telle ou telle opération, qu'elle permette de gérer l'intégrité ou simplement de faciliter l'exploitation des données.

Le PL/SQL (Oracle), ou le T-SQL (SQL-Server) sont du SQL procédural. Ici, nous allons utiliser le langage de MySQL (qui ne porte pour le moment aucun nom).

Le but est de rendre la base de données la plus autonome possible, et qu'il ne dépende d'aucune application pour garantir une intégrité parfaite.

2. Un exemple qui pose problème

Sans plus attendre, nous allons voir un cas simple de problème qui peut se poser.

Nous souhaitons gérer un système de participation à des courses.

Création de la structure des données

```
CREATE TABLE `Sportif`
(
    `numSportif` INTEGER AUTO_INCREMENT,
    `nomSportif` VARCHAR(20) NOT NULL,
    `prenomSportif` VARCHAR(20) NOT NULL,
    CONSTRAINT `PK_SPORTIF` PRIMARY KEY
    (`numSportif`)
) ENGINE = `innnoDB`;
```

```
CREATE TABLE `Course`
(
    `numCourse` INTEGER AUTO_INCREMENT,
    `libelleCourse` varchar(50) NOT NULL,
    `dateCourse` DATETIME NOT NULL,
    `nbMaxParticipant` INTEGER NOT NULL,
    CONSTRAINT `PK_SPORTIF` PRIMARY KEY
    (`numCourse`)
) ENGINE = `innnoDB`;
```

```
CREATE TABLE `Participer`
(
    `numSportif` INTEGER NOT NULL,
    `numCourse` INTEGER NOT NULL,
    `tempsParticipation` FLOAT NULL,
    CONSTRAINT `PK_PARTICIPER` PRIMARY
    KEY(`numSportif`, `numCourse`)
) ENGINE = `innnoDB`;
```

```
ALTER TABLE `Participer`
ADD CONSTRAINT `FK_PARTICIPER_SPORTIF` FOREIGN KEY
(`numSportif`) REFERENCES `Sportif`(`numSportif`),
ADD CONSTRAINT `FK_PARTICIPER_COURSE` FOREIGN KEY
(`numCourse`) REFERENCES `Course`(`numCourse`);
```

Insertion d'un jeux d'essai

```
INSERT INTO `Sportif` (`numSportif`, `nomSportif`,
`prenomSportif`)
VALUES
(1, 'MARTIN', 'Pierre'),
(2, 'BERNARD', 'Jean'),
(3, 'THOMAS', 'Jacques'),
(4, 'PETIT', 'François'),
(5, 'DURAND', 'Charles'),
(6, 'RICHARD', 'Louis'),
(7, 'DUBOIS', 'Jean-Baptiste'),
(8, 'ROBERT', 'Joseph'),
(9, 'LAURENT', 'Nicolas'),
(10, 'SIMON', 'Antoine'),
(11, 'MICHEL', 'Marie'),
(12, 'LEROY', 'Marguerite');
```

```
INSERT INTO `Course` (`numCourse`, `libelleCourse`,
`dateCourse`, `nbMaxParticipants`)
VALUES
(1, 'Pique du Geek', '2009-03-14 08:00:00', 5),
```

```

(2, 'Course de noel', '2009-12-25 14:30:00',
30),
(3, 'je veux pas courir !', '2009-06-29
12:00:00', 11);

INSERT INTO `Participer` (`numSportif`, `numCourse`)
VALUES
(1, 1),
(4, 1),
(5, 1),
(8, 1),
(9, 1);

```

La course Pique du Geek est complète (5/5), alors que se passe-t-il si nous réalisons une insertion supplémentaire ? Et bien rien, puisque le SGBD est incapable de comprendre le sens de la colonne "nbMaxParticipants". La solution la plus évidente est d'effectuer un contrôle au niveau applicatif, c'est-à-dire après avoir fait un SELECT, contrôler ceci puis effectuer ou non l'insertion.

Nous pourrions effectuer le contrôle côté applicatif; en quoi pourquoi est-ce une mauvaise pratique ?

Le rôle du SGBD étant en grande partie de contrôler l'intégrité, il n'assume pas ici entièrement son rôle. Il est gênant de ne pas pouvoir faire confiance à son SGBD en contrôlant nous-mêmes les valeurs qu'on lui envoie ; c'est en partie son rôle de traiter la validité de ces informations. Nous allons donc faire en sorte d'automatiser ce contrôle côté SGBD.

3. La solution : le trigger

3.1. Qu'est-ce qu'un trigger ?

Un trigger est un déclencheur. Autrement dit, c'est la possibilité d'associer un certain code à un événement.

Cela fait beaucoup penser à la programmation événementielle qui associe elle aussi du code à un événement (onclick, onfocus...). C'est, en quelque sorte, la même chose, mis à part que les événements ne sont pas directement liés aux actions de l'utilisateur, mais aux requêtes effectuées auprès du SGBD.

Liste des événements possibles :

- INSERT
- UPDATE
- DELETE

Puisque ces événements provoquent une modification dans le contenu de la base de données, il est possible d'agir avant (BEFORE) ou après (AFTER) ces modifications. Nous agissons souvent après pour ajouter des tuples qui seront liés aux nouvelles données, alors que nous utiliserons généralement BEFORE dans un souci de contrôle de l'intégrité (annuler une "mauvaise donnée").

3.2. La syntaxe d'un trigger

Prenons l'exemple d'un trigger qui se déclenchera avant un insert sur la table *Participer* :

Syntaxe d'un trigger

```

CREATE TRIGGER `nomTrigger`
BEFORE INSERT
ON `Participer`
FOR EACH ROW
BEGIN
    -- Code à exécuter;
END;

```

La clause FOR EACH ROW permet de préciser le type de trigger.

En effet il existe plusieurs triggers, les triggers sur table et les triggers sur tuple. FOR EACH ROW précise qu'il s'agit d'un trigger sur tuple, c'est-à-dire que le code sera exécuté pour chaque tuple concerné par l'action.

Cela sous-entend que les préfixes OLD et NEW que nous utiliserons plus tard deviennent utilisables.

Pour le moment MySQL gère uniquement les triggers sur tuple.

A propos du délimiteur

Le délimiteur est source de beaucoup d'erreurs. Il dépend du client, certains en ont besoin, d'autres pas. De plus ceux qui en ont besoin n'utilisent pas nécessairement la même syntaxe. Si la plupart des clients graphiques (Toad For MySQL, MySQL Query Browser) gèrent eux-mêmes le délimiteur, il n'en est pas de même pour PhpMyAdmin, et le client texte mysql (client en ligne de commande). Le client texte MySQL accepte la commande DELIMITER qui permet de modifier le délimiteur signalant la fin de la procédure, fonction, trigger, ou simple requête (par défaut, le délimiteur est ;).

Par exemple voici un exemple de trigger avec l'utilisation d'un délimiteur // :

Utilisation d'un délimiteur avec le client texte MySQL

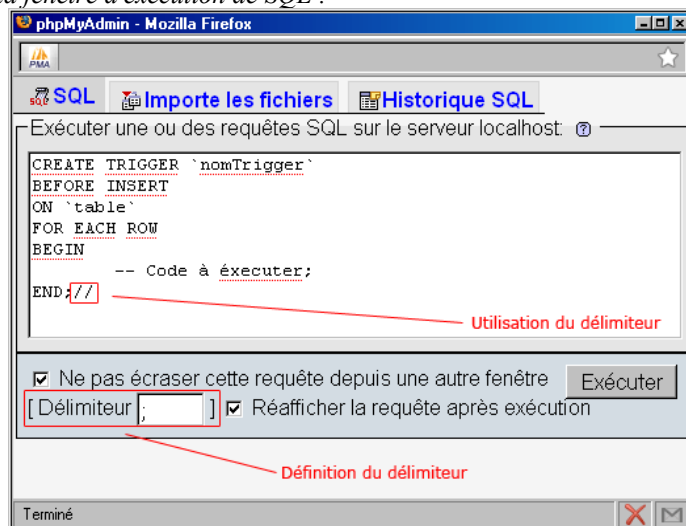
```

DELIMITER //

CREATE TRIGGER `nomTrigger`
BEFORE INSERT
ON `table`
FOR EACH ROW
BEGIN
    -- Code à exécuter;
END; //

```

PhpMyAdmin, quant à lui, offre un paramètre à remplir en bas de la fenêtre d'exécution de SQL :



3.3. Une solution au problème

Nous allons maintenant créer un trigger qui permettra de contrôler l'insertion. En reprenant le problème précédemment posé, il nous serait utile de n'effectuer les insertions dans *Participer* que s'il reste de la place dans la course concernée.

Contrôle d'insertion de participation

```

CREATE TRIGGER `TR_INSERT_PARTICIPANT`
BEFORE INSERT
ON `Participer`
FOR EACH ROW
BEGIN
    DECLARE CURRENT_NB INTEGER;
    DECLARE MAX_NB INTEGER;

```

```

SELECT COUNT(*) INTO CURRENT_NB FROM
`Participer` WHERE numCourse = NEW.numCourse;
SELECT `nbMaxParticipants` INTO MAX_NB FROM
`Course` WHERE numCourse = NEW.numCourse;

IF (CURRENT_NB >= MAX_NB) THEN
    SET NEW.numCourse = NULL;
    SET NEW.numSportif = NULL;
END IF;
END;
//

```

Le fonctionnement de ce trigger est simple : avant chaque insertion, on vérifie s'il reste de la place et, si ce n'est pas le cas, on modifiera les informations saisies, provoquant ainsi un viol des contraintes d'intégrité, et un refus de l'insertion de la part du SGBD. Cela n'est pas la manière la plus élégante de faire, puisqu'il est impossible de savoir si l'échec d'insertion provient d'une mauvaise saisie, ou bien d'un manque de place. Nous allons peu à peu faire évoluer le système pour arriver à quelque chose de correct.

La gestion de ce type d'erreur spécifique au domaine de gestion est normalement assurée par l'utilisation d'exceptions personnalisées. Les exceptions sont disponibles sur MySQL, mais seul le système peut les propager, c'est-à-dire que le développeur ne peut déclencher de lui-même une exception.

Il n'est donc pas possible d'aborder le problème comme dans d'autres SGBD plus aboutis comme Oracle ou SQL-Server. Nous verrons plus tard ce que MySQL prévoit de mettre en oeuvre dans ses prochaines versions concernant cette gestion d'exceptions personnalisées.

3.4. Traiter tous les cas de figure

Si le SGBD est très performant dans sa gestion de l'intégrité, le développeur est soumis au risque de commettre des erreurs et d'oublier des cas possibles (c'est d'ailleurs ce que nous cherchons à éviter en reportant la gestion des erreurs sur le SGBD). Ici, nous avons prévu que l'on puisse ajouter des participations, mais rien n'empêche de modifier le nombre maximum de participations. Il faut donc prévoir et traiter ce cas.

Nous allons empêcher une diminution trop importante du nombre maximal de participations à une course.

Contrôle de modification des courses

```

CREATE TRIGGER `TR_UPDATE_COURSE`
BEFORE UPDATE
ON `Course`
FOR EACH ROW
BEGIN
    DECLARE CURRENT_NB INTEGER;
    DECLARE MAX_NB INTEGER;

    IF (NEW.nbMaxParticipants <
    OLD.nbMaxParticipants) THEN
        SELECT COUNT(*) INTO CURRENT_NB FROM
        `Participer` WHERE numCourse = OLD.numCourse;
        SET MAX_NB = NEW.nbMaxParticipants;

        IF (CURRENT_NB > MAX_NB) THEN
            SET NEW.nbMaxParticipants =
NULL;

```

```

END IF;
END IF;
END;
//

```

Nous sommes maintenant certains que le nombre de participations réelles dans *Participer* sera en accord avec le nombre maximal de participations prévu par course.

4. Comment générer un code erreur pour le récupérer côté applicatif par la suite ?

La solution précédemment décrite est incorrecte dans le sens où elle provoque volontairement une erreur qui n'a aucun rapport avec sa cause fonctionnelle. Nous allons voir comment faire pour utiliser la programmation stockée et les mêmes contrôles, mais de manière plus élégante.

4.1. Utilisation de fonctions utilisateurs

Une fonction utilisateur, comme dans la plupart des langages procéduraux, est un processus recevant des paramètres et renvoyant une valeur. MySQL permet aussi les procédures stockées (qui sont comme leur nom l'indique, de simples procédures) mais elles ne nous seront pas utiles dans ce tutoriel. Le but est de créer et utiliser une interface entre le code applicatif et les données. Des fonctions feront l'intermédiaire et filtreront ce qui doit effectivement être contrôlé, et renverront au besoin un code d'erreur.

Reprenons le premier problème, l'ajout de participation :

Fonction d'ajout de participation

```

CREATE FUNCTION `NEW_PARTICIPATION` (P_numSportif INT,
P_numCourse INT) RETURNS INT
BEGIN
    DECLARE CURRENT_NB INTEGER;
    DECLARE MAX_NB INTEGER;
    DECLARE C_ERROR INTEGER;

    SET C_ERROR = 0;
    SELECT COUNT(*) INTO CURRENT_NB FROM
    `Participer` WHERE numCourse = P_numCourse;
    SELECT `nbMaxParticipants` INTO MAX_NB FROM
    `Course` WHERE numCourse = P_numCourse;

    IF (CURRENT_NB < MAX_NB) THEN
        INSERT INTO `Participer` (`numSportif`,
        `numCourse`) VALUES (P_numSportif, P_numCourse);
    ELSE
        SET C_ERROR = 1001;
    END IF;

    RETURN C_ERROR;
END;
//

```

Il suffit d'appeler cette fonction pour toutes les insertions dans la table *Participer*, et de se servir de son code de retour pour évaluer le succès de l'insertion :

Utilisation de la fonction

```

SELECT NEW_PARTICIPATION(<numSportif>, <numCourse>) AS
ERROR;

```

Retrouvez la suite de l'article d'Alain Defrance en ligne : [Lien59](#)

Intégrez un modèle de calcul à vos requêtes SQL : la clause MODEL d'Oracle

La clause MODEL, spécifique à Oracle et introduite par la version 10g, permet de définir un modèle de calcul matriciel à l'intérieur d'une simple requête SQL. Pour ceux qui ne connaissent que la modélisation relationnelle normalisée, disons que MODEL permet de réaliser des calculs complexes, au besoin itératifs, sans programmation PL/SQL et dans un simple SELECT. Pour ceux qui ont déjà tâté des concepts décisionnels (*Business Intelligence*), disons que MODEL permet de définir dimensions et indicateurs par un mapping proche du

ROLAP, à l'intérieur d'un simple SELECT. Ce tutoriel présente la clause MODEL en vous initiant à son utilisation.

L'auteur : Rob van Wijk est consultant senior sur Oracle pour Ciber Nederland, et tient un blog nommé About Oracle ([Lien60](#)).

La source : ce tutoriel est traduit du blog de Rob, et adapté pour Développez.com. Cette présentation et quelques notes en italiques ont été ajoutées par le traducteur. Les versions originales sont disponibles ici : *SQL Model Clause Tutorial, part one* ([Lien61](#)), *part two* ([Lien62](#)).

1. Quelques exemples pour débiter

La clause MODEL du SQL d'Oracle vous permet de construire un modèle composé d'une ou plusieurs matrices, avec un nombre variable de dimensions. Le modèle utilise une partie des colonnes disponibles dans la clause FROM. Il doit contenir au moins une dimension et un indicateur (*measure*), et éventuellement une ou plusieurs partitions. Le modèle peut être représenté comme un classeur de tableur, contenant des feuilles de calcul différentes pour chaque valeur calculée (indicateur). Une feuille de calcul présente un axe horizontal et un axe vertical (deux dimensions) ; vous pouvez séparer votre feuille en plusieurs zones identiques, chacune dédiée à un pays ou un département différent (partition).

La figure ci-dessous présente un modèle tiré de la classique table EMP, avec *deptno* comme partition, *empno* comme dimension, et les deux indicateurs *sal* et *comm*.

	indicateur sal	indicateur comm
Partition deptno 10	sal[7782] = 2450	comm[7782] = null
	sal[7839] = 5000	comm[7839] = null
	sal[7782] = 1300	comm[7782] = null
Partition deptno 20	sal[7369] = 800	comm[7369] = null
	sal[7566] = 2975	comm[7566] = null
	sal[7788] = 3000	comm[7788] = null
	sal[7876] = 1100	comm[7876] = null
	sal[7902] = 3000	comm[7902] = null
Partition deptno 30	sal[7499] = 1600	comm[7499] = 300
	sal[7521] = 1250	comm[7521] = 500
	sal[7654] = 1250	comm[7654] = 1400

Une fois le modèle mis en place, vous définissez les règles qui modifient la valeur des indicateurs. Ce sont ces règles qui sont au coeur de la clause MODEL. Avec quelques règles, vous pouvez faire des calculs complexes sur vos données, et même créer de nouvelles lignes. Dans le modèle, les colonnes d'indicateurs deviennent des tableaux indexés (*hash tables*) dont les clés sont les colonnes de dimension. Oracle y applique les règles de calcul à toutes les partitions. Une fois ces calculs effectués, le modèle est re-converti en lignes de données traditionnelles.

Selon mon expérience, le diagramme syntaxique de la clause MODEL présenté par la documentation Oracle est passablement complexe et tend à effrayer tout le monde. Pour éviter d'en arriver là, je vais tenter une autre approche, en utilisant un série de petits exemples sur la table EMP, en commençant par les plus simples, et en introduisant progressivement de nouveaux éléments. A la fin de cet article, vous trouverez un script que vous pouvez télécharger et exécuter sur votre propre base de données.

7782	CLARK	2450
7839	KING	5000
7934	MILLER	1300

3 lignes sélectionnées

Voici le contenu (classique) de la table EMP pour le département 10. L'équivalent de cette requête SQL avec une clause MODEL (qui ne fait rien) est :

```
SQL> select empno
2      , ename
3      , sal
4  from emp
5  where deptno = 10
6  model
7      dimension by (empno)
8      measures (ename, sal)
9      rules
10     ()
11 /
```

EMPNO	ENAME	SAL
7782	CLARK	2450
7839	KING	5000
7934	MILLER	1300

3 lignes sélectionnées

Nous avons ici deux indicateurs, *ename* et *sal*, et une dimension, *empno*. La combinaison des colonnes de partitions et de dimensions doit permettre d'identifier chaque cellule de manière unique. Cette contrainte est vérifiée à l'exécution, et sa violation produira une erreur ORA-32638.

Autrement dit, cette combinaison des dimensions et partitions doit pouvoir servir de clé unique pour les lignes qui vont être traitées par la clause MODEL. Cette limite est la plus importante (elle peut imposer une agrégation préalable), mais aussi la plus révélatrice du fonctionnement du modèle. Une matrice de calcul est créée, où dimensions et partitions vont servir d'axes de coordonnées.

Les habitués d'Hyperion Essbase peuvent considérer les partitions comme des dimensions sparses, tandis que les dimensions de la clause MODEL seraient les dimensions denses d'Essbase.

Comme il y a deux indicateurs, Oracle crée deux tableaux à une dimension, indicés par *empno*. Sur la ligne 9, vous voyez le mot-clé RULES, sans contenu pour le moment. RULES est optionnel, mais par souci de clarté je l'utiliserai systématiquement.

```
SQL> select empno
2      , ename
3      , sal
4  from emp
5  where deptno = 10
6  model
7      dimension by (empno)
8      measures (sal)
9      rules
10     ()
11 /
```

```
SQL> select empno
2      , ename
3      , sal
4  from emp
5  where deptno = 10
6  /
```

EMPNO	ENAME	SAL
7782	CLARK	2450
7839	KING	5000
7934	MILLER	1300

Une fois le modèle de calcul appliqué, toutes les partitions, dimensions et indicateurs sont re-convertis en colonnes à l'intérieur de lignes classiques, ce qui veut dire que vous ne pouvez avoir dans le SELECT que des colonnes citées dans le modèle. Si, par exemple, je n'avais pas inclus la colonne *ename* comme indicateur, j'aurais eu ce message d'erreur :

```
ERREUR à la ligne 2 :
ORA-32614: expression MODEL SELECT interdite
```

Avec l'exemple suivant, je crée une nouvelle ligne :

```
SQL> select empno
 2      , ename
 3      , sal
 4 from emp
 5 where deptno = 10
 6 model
 7   dimension by (empno)
 8   measures (ename,sal)
 9   rules
10   ( ename[7777] = 'VAN WIJK'
11     )
12 /
```

EMPNO	ENAME	SAL
7782	CLARK	2450
7839	KING	5000
7934	MILLER	1300
7777	VAN WIJK	

4 lignes sélectionnées

La règle en ligne 10 indique que l'indicateur *ename* est croisé avec la dimension *empno* et prend la valeur « VAN WIJK ». Si la table EMP avait déjà un *empno* 7777, la règle aurait écrasé l'*ename* correspondant. Mais comme 7777 n'existe pas dans EMP, une nouvelle cellule a été créée, qui se traduit par une nouvelle ligne dans le résultat du SELECT. Notez bien que cette ligne n'est pas insérée dans une table, mais seulement affichée dans le résultat.

La clause MODEL ne s'utilise que dans une requête SELECT. Cela inclut le INSERT... SELECT et le MERGE INTO... USING (SELECT...).

Avec une seconde règle, vous pourriez également renseigner la colonne *sal* :

```
SQL> select empno
 2      , ename
 3      , sal
 4 from emp
 5 where deptno = 10
 6 model
 7   dimension by (empno)
 8   measures (ename,sal)
 9   rules
10   ( ename[7777] = 'VAN WIJK'
11     , sal[7777] = 2500
12     )
13 /
```

EMPNO	ENAME	SAL
7782	CLARK	2450
7839	KING	5000
7934	MILLER	1300
7777	VAN WIJK	2500

4 lignes sélectionnées

La requête renvoie à la fois les lignes existantes et les nouvelles lignes. L'expression-clé RETURN UPDATED ROWS vous permet de ne renvoyer que les lignes modifiées ou créées :

```
SQL> select empno
 2      , ename
 3      , sal
 4 from emp
 5 where deptno = 10
 6 model
 7   return updated rows
 8   dimension by (empno)
 9   measures (ename,sal)
10   rules
11   ( ename[7777] = 'VAN WIJK'
12     , sal[7777] = 2500
13     )
14 /
```

EMPNO	ENAME	SAL
7777	VAN WIJK	2500

1 ligne sélectionnée

Tous les calculs sont exécutés sur chaque partition. Pour le constater, enlevons le filtre « deptno = 10 » et affichons le *deptno*. Comme partition ou comme indicateur ? Pour commencer, testons ce qui se passe si nous définissons *deptno* comme indicateur :

```
SQL> select empno
 2      , ename
 3      , sal
 4      , deptno
 5 from emp
 6 model
 7   return updated rows
 8   dimension by (empno)
 9   measures (ename,sal,deptno)
10   rules
11   ( ename[7777] = 'VAN WIJK'
12     )
13 /
```

EMPNO	ENAME	SAL	DEPTNO
7777	VAN WIJK		

1 ligne sélectionnée

Une ligne est créée, comme attendu, avec un *deptno* NULL. Testons maintenant avec *deptno* comme partition :

```
SQL> select empno
 2      , ename
 3      , sal
 4      , deptno
 5 from emp
 6 model
 7   return updated rows
 8   partition by (deptno)
 9   dimension by (empno)
10   measures (ename,sal)
11   rules
12   ( ename[7777] = 'VAN WIJK'
13     )
14 /
```

EMPNO	ENAME	SAL	DEPTNO
7777	VAN WIJK		30
7777	VAN WIJK		20
7777	VAN WIJK		10

Une nette différence ! Dans la table EMP, il n'y a que des *deptno* 10, 20 et 30, ce qui fait donc trois partitions dans le modèle. La règle est appliquée aux trois, ce qui donne donc trois nouvelles lignes.

Pourrait-on inclure deptno comme dimension ? Oui, mais cela forcerait à indiquer dans la règle le numéro de département comme caractéristique de l'indicateur : par exemple,

```
ename[7777, 10] = 'VAN WIJK',
ename[7777, 20] = 'DINIMANT',
ename[7777, 30] = 'SCHNEIDER'.
```

Plus largement, les dimensions permettent de définir les axes qui interviendront dans le calcul des indicateurs. Au contraire, les calculs sont exécutés à l'identique d'une partition à l'autre ; on peut donc dire que les partitions sont précisément les éléments qui ne jouent pas dans le calcul.

Pour l'instant, ces exemples ne vous ont sans doute pas convaincu d'utiliser la clause MODEL. Nous avons ajouté des tas de lignes de code SQL pour un résultat très simple : une utilisation créative de l'opérateur ensembliste UNION ALL et de la table DUAL nous aurait permis de faire la même chose. L'objectif de cette première partie était simplement de vous montrer les bases de l'utilisation de la clause MODEL. Dans la seconde partie, je vous montrerai les références multi-cellules, les modèles de références et les itérations. Et c'est là que les choses deviendront autrement plus intéressantes.

2. Statistiques et simulation

2.1. Références multi-cellules

Il est possible de traiter plusieurs cellules avec une seule règle, en utilisant une *référence multi-cellules*. Pour illustrer cette notion, je vais introduire une nouvelle table de démonstration avec une clé primaire composée, afin de pouvoir travailler sur des dimensions multiples. Cette table contient les ventes mensuelles de deux livres en 2005 et 2006. Voici la requête de création :

```
SQL> create table sales
2 as
3 select 'The Da Vinci Code' book, date '2005-03-01'
month, 'Netherlands' country, 5 amount from dual union
all
4 select 'The Da Vinci Code', date '2005-04-01',
'Netherlands', 8 from dual union all
5 select 'The Da Vinci Code', date '2005-05-01',
'Netherlands', 3 from dual union all
6 select 'The Da Vinci Code', date '2005-07-01',
'Netherlands', 2 from dual union all
7 select 'The Da Vinci Code', date '2005-10-01',
'Netherlands', 1 from dual union all
8 select 'The Da Vinci Code', date '2005-02-01',
'United Kingdom', 15 from dual union all
9 select 'The Da Vinci Code', date '2005-03-01',
'United Kingdom', 33 from dual union all
10 select 'The Da Vinci Code', date '2005-04-01',
'United Kingdom', 47 from dual union all
```

```
11 select 'The Da Vinci Code', date '2005-05-01',
'United Kingdom', 44 from dual union all
12 select 'The Da Vinci Code', date '2005-06-01',
'United Kingdom', 11 from dual union all
13 select 'The Da Vinci Code', date '2005-08-01',
'United Kingdom', 2 from dual union all
14 select 'The Da Vinci Code', date '2005-05-01',
'France', 2 from dual union all
15 select 'The Da Vinci Code', date '2005-08-01',
'France', 3 from dual union all
16 select 'The Da Vinci Code', date '2006-01-01',
'France', 4 from dual union all
17 select 'Bosatlas', date '2005-01-01',
'Netherlands', 102 from dual union all
18 select 'Bosatlas', date '2005-02-01',
'Netherlands', 55 from dual union all
19 select 'Bosatlas', date '2005-03-01',
'Netherlands', 68 from dual union all
20 select 'Bosatlas', date '2005-04-01',
'Netherlands', 42 from dual union all
21 select 'Bosatlas', date '2005-05-01',
'Netherlands', 87 from dual union all
22 select 'Bosatlas', date '2005-06-01',
'Netherlands', 40 from dual union all
23 select 'Bosatlas', date '2005-07-01',
'Netherlands', 31 from dual union all
24 select 'Bosatlas', date '2005-08-01',
'Netherlands', 26 from dual union all
25 select 'Bosatlas', date '2005-09-01',
'Netherlands', 22 from dual union all
26 select 'Bosatlas', date '2005-10-01',
'Netherlands', 23 from dual union all
27 select 'Bosatlas', date '2005-11-01',
'Netherlands', 88 from dual union all
28 select 'Bosatlas', date '2005-12-01',
'Netherlands', 143 from dual union all
29 select 'Bosatlas', date '2006-01-01',
'Netherlands', 31 from dual union all
30 select 'Bosatlas', date '2006-02-01',
'Netherlands', 18 from dual union all
31 select 'Bosatlas', date '2006-03-01',
'Netherlands', 15 from dual union all
32 select 'Bosatlas', date '2006-04-01',
'Netherlands', 11 from dual union all
33 select 'Bosatlas', date '2006-05-01',
'Netherlands', 17 from dual union all
34 select 'Bosatlas', date '2006-06-01',
'Netherlands', 9 from dual union all
35 select 'Bosatlas', date '2006-07-01',
'Netherlands', 12 from dual union all
36 select 'Bosatlas', date '2006-08-01',
'Netherlands', 20 from dual union all
37 select 'Bosatlas', date '2006-09-01',
'Netherlands', 4 from dual union all
38 select 'Bosatlas', date '2006-10-01',
'Netherlands', 5 from dual union all
39 select 'Bosatlas', date '2006-11-01',
'Netherlands', 1 from dual union all
40 select 'Bosatlas', date '2006-12-01',
'Netherlands', 1 from dual
41 /
```

Table créée

Retrouvez la suite de l'article de Rob van Wijk traduit par Antoine Dinimant en ligne : [Lien63](#)

Les derniers tutoriels et articles

Démarrage du noyau Linux

Cet article a pour but de vous présenter le processus de démarrage de Linux.

1. Introduction

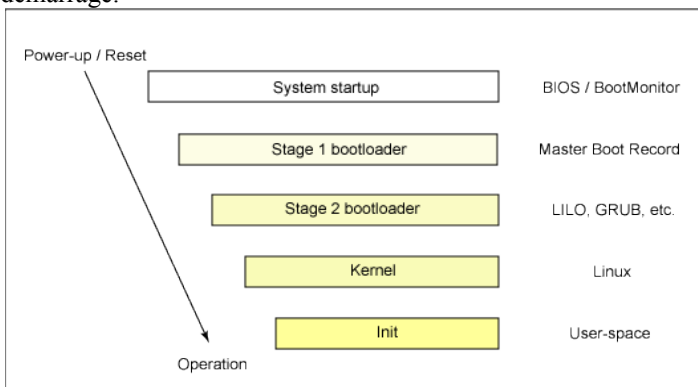
Le processus de mise en route d'un système d'exploitation consiste en l'exécution d'un certain nombre d'étapes. Mais, bien que vous soyez en train de démarrer un ordinateur personnel basé sur une architecture x86 ou un système embarqué avec un PowerPc, la plupart des opérations sont étonnamment semblables. Cet article explore le processus de démarrage de linux depuis le bootstrap initial jusqu'au lancement de la première application dans l'espace utilisateur. Tout au long, vous allez apprendre des choses à propos des nombreuses autres étapes liées au démarrage, tel que le chargeur d'amorçage, la décompression du noyau, l'initial RAM disk,...

Dans les premiers temps de l'informatique, démarrer un ordinateur signifiait fournir une bande perforée contenant les instructions du programme de démarrage ou encore charger manuellement un programme en utilisant un panneau frontal. Aujourd'hui, les ordinateurs sont équipés avec un certain nombre de services pour simplifier le démarrage, mais ceci ne le rend pas pour autant simple.

Commençons avec une vue de haut niveau du démarrage d'un système linux pour que vous puissiez voir le processus dans sa globalité. Ensuite, nous nous attarderons individuellement sur chaque phase du processus. Les références vous aideront à naviguer à l'intérieur du noyau et à approfondir.

2. Vue générale

La figure 1 vous donne une vision très globale du processus de démarrage.



Vue très globale du processus de boot

Quand un système démarre, ou est redémarré, le processeur exécute du code à une adresse fixe. Dans un PC, cette adresse fixe est celle du BIOS (*Basic Input/Output System*), qui est stocké dans une ROM sur les cartes mères. Dans un environnement embarqué, le processeur exécute le début du segment de code pour démarrer un programme à une adresse connue dans une mémoire flash/ROM. Dans tous les cas, le résultat est le même. Mais à cause de la flexibilité offerte par le PC, le BIOS doit aussi

déterminer quels périphériques sont candidats pour démarrer dessus. Nous y reviendrons plus tard.

Quand un périphérique sur lequel on peut démarrer est trouvé, le premier programme du processus de démarrage est chargé en RAM puis exécuté. Ce chargeur de démarrage fait au plus 512 bits (un secteur) et son rôle est de charger le deuxième programme.

Quand le deuxième programme (le chargeur d'amorçage) est chargé en RAM et exécuté, un *splash-screen* est souvent affiché et le noyau linux ainsi que l'optionnel *initrd* (initial RAM disk: le système de fichier principal temporaire) sont chargés en mémoire. Après que les images aient été chargées, le deuxième programme passe la main à l'image du noyau et le noyau est alors décompressé. À ce stade, le deuxième programme vérifie aussi le matériel, énumère les périphériques attachés, monte le périphérique principal et charge les modules du noyau nécessaires. Quand tout cela est fini, le premier programme de l'espace utilisateur (*init*) démarre, et l'initialisation à haut niveau du système a lieu.

C'est ainsi que, de façon très synthétique, se déroule le démarrage d'un système linux. Maintenant creusons un peu plus loin et regardons dans les détails ce qui se passe.

3. Démarrage du système

Le démarrage du système dépend du matériel sur lequel linux est démarré. Sur des plateformes embarquées, un bootstrap est utilisé quand le système est mis sous tension, ou re-démarré. On peut par exemple citer u-boot, RedBoot ou encore MicroMonitor de Lucent. Les plateformes embarquées disposent habituellement d'un programme principal de démarrage. Ce programme réside dans une zone spéciale de la mémoire flash sur la machine cible et fournit un moyen pour télécharger une image du noyau Linux dans la mémoire flash puis pour l'exécuter. En plus de pouvoir stocker et démarrer une image du noyau linux, ce programme principal de démarrage assure aussi plusieurs niveaux de test et d'initialisation du matériel. Dans une plateforme embarquée, il regroupe souvent le premier et le deuxième programme.

Dans un ordinateur, le démarrage de Linux commence dans le BIOS à l'adresse 0xFFFF0. La première chose que réalise le BIOS est le *power-on self test* (POST). Le rôle du POST est de vérifier le matériel. La seconde chose que fait le BIOS est d'énumérer puis d'initialiser les périphériques locaux.

Étant donné les différents usages des fonctions du BIOS, ce dernier est conçu en deux parties : le POST et les fonctions utilitaires. Après que le POST soit fini, il est nettoyé de la mémoire mais les fonctions utilitaire du BIOS restent et sont toujours disponibles pour le système que l'on va démarrer.

Pour démarrer un système d'exploitation, la routine du BIOS cherche un périphérique qui est à la fois actif et démarrable selon l'ordre de préférence défini par les réglages du CMOS (*metal oxide semiconductor*). Un périphérique démarrable peut être une disquette, un CD-ROM, une partition d'un disque, un périphérique situé sur le réseau ou encore une clé USB.

Le plus souvent, Linux est démarré depuis un disque dur, où le *master boot record* (MBR) contient le premier chargeur d'amorçage. Le MBR est un bloc de 512 octets, localisé dans le premier secteur du disque (secteur 1 du cylindre 0, tête 0). Après que le MBR ai été chargé en RAM, le BIOS lui donne le contrôle.

Extraire le MBR

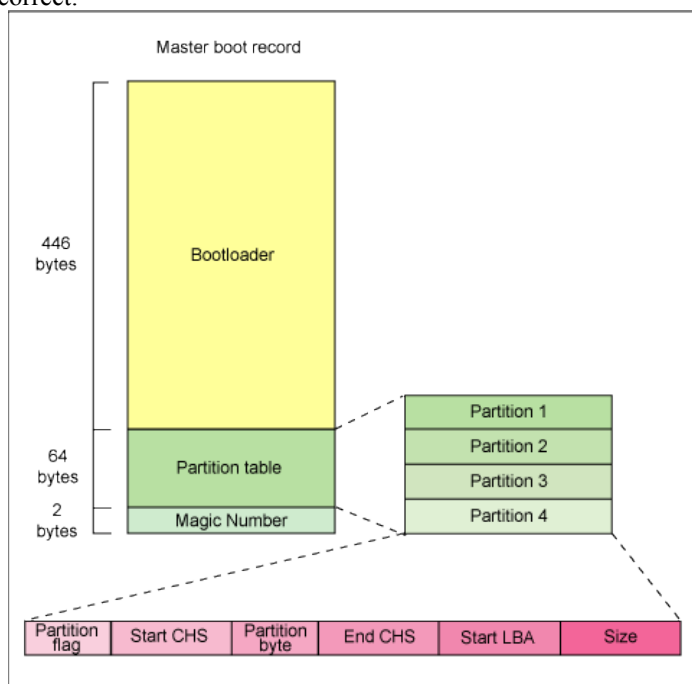
Pour voir le contenu de votre MBR, utilisez les commandes suivantes:

```
# dd if=/dev/hda of=mbr.bin bs=512 count=1
# od -xn mbr.bin
```

La commande `dd`, qui doit être exécutée en tant que `root`, lit les 512 premiers octets depuis `/dev/hda` (le premier disque de type IDE) et écrit le tout dans le fichier `mbr.bin`. Puis la commande `od` affiche le contenu du fichier dans un format hexadécimal, ASCII si possible.

4. Phase 1: le chargeur d'amorçage

Le premier chargeur d'amorçage qui réside dans un unique secteur de 512 octets contient à la fois un programme de chargement et une petite table de partition (regardez la figure 2). Les 446 premiers octets sont précisément le premier chargeur d'amorçage, qui contient en même temps du code exécutable et les messages d'erreurs. Les 64 prochains octets forment la table des partitions, qui contient un enregistrement sur 16 bits de chacune des partitions primaires ou étendues présente sur le disque. Le MBR se termine avec avec 2 bits étant définis comme étant un nombre magique (0xAA55), nombre qui sert à vérifier que le MBR est correct.



Shéma détaillé du MBR

Le travail du premier chargeur d'amorçage est de trouver puis de charger le deuxième chargeur d'amorçage. Il réalise ceci en regardant à travers la table des partitions celles qui sont marquées

comme actives. Quand il trouve une partition active, il examine quand même les autres partitions pour vérifier qu'elles sont bien inactives. Quand cette vérification est terminée, l'enregistrement actif de la partition est chargé en RAM puis exécuté.

5. Phase 2: un autre chargeur d'amorçage

Le deuxième chargeur d'amorçage pourrait être à juste titre appelé le *kernel loader* (ou encore chargeur de noyau). En effet, le devoir de ce deuxième programme est de charger le noyau Linux (une obligation) et l'*initial RAM disk* (qui est optionnel).

La combinaison du premier et du deuxième programme est appelée soit *Linux Loader* (LILO) ou *GRand Unified Bootloader* (GRUB) dans un environnement de type x86. À cause d'un certain nombre de problèmes de LILO qui ont été corrigés dans GRUB, nous regarderons seulement GRUB (Regardez les ressources additionnelles sur GRUB, LILO et les sujets apparentés dans la section ressource.)

Une bonne chose de GRUB est sa connaissance du système de fichier utilisé par Linux. En effet, au lieu d'utiliser des secteurs bruts comme LILO le fait, GRUB est capable de charger un noyau linux depuis système de fichier de type ext2 ou ext3. Il y arrive en utilisant 2 niveaux de chargement parmi les 3 possibles. En effet, le premier niveau (le MBR) démarre un niveau intermédiaire (noté 1.5) qui peut utiliser un système de fichier particulier, ce système étant celui sur lequel le noyau se situe. Par exemple, le niveau 1 va démarrer le `reiserfs_stage1_5` pour charger le noyau linux si celui-ci se trouve sur une partition de type reiserfs. Il chargera `e2fs_stage1_5` si le noyau se trouve sur une partition formatée en ext2 ou 3. Quand le niveau intermédiaire est chargé et a démarré, le deuxième programme peut enfin être chargé.

Avec le deuxième programme, GRUB peut, sur demande, afficher une liste des noyaux disponibles (cette liste étant définie dans `/etc/grub.conf` avec un lien symbolique depuis `/etc/grub/menu.lst` et `/etc/grub.conf`). Vous pouvez sélectionner un noyau et même le modifier avec des paramètres additionnels. Vous pouvez même utiliser la ligne de commande fournie par GRUB pour avoir un plus grand contrôle sur le processus de démarrage.

Avec le deuxième programme en mémoire, le système de fichiers est consulté, et l'image du noyau par défaut ainsi que `inirttd` sont chargés en mémoire. Quand les images sont prêtes, le deuxième programme invoque l'image du noyau.

Le répertoire `/boot/grub` contient le `stage1`, `stage1_5` et `stage2` ainsi qu'un bon nombre d'autres chargeurs d'amorçage (par exemple, les CR-Rom utilisent `iso9660_stage1_5`)

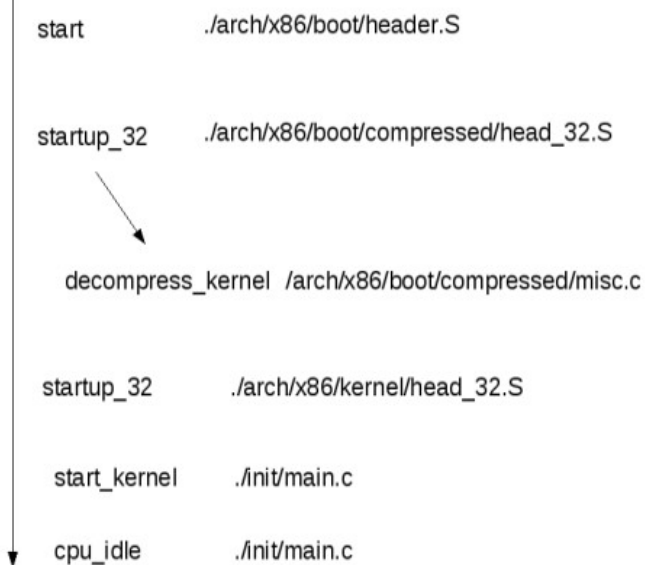
6. Le noyau

Avec l'image du noyau en mémoire et l'abandon du contrôle par le deuxième programme, la phase du noyau peut commencer. L'image du noyau n'est pas plus qu'un noyau exécutable, mais compressé dans une image. Typiquement, c'est soit une `zImage` (une image compressée faisant moins de 512 koctets) ou une `bzImage` (une grosse image compressée, faisant plus de 512 koctets), qui a été précédemment compressée avec `zlib`. Au début du noyau se trouve une routine qui dresse une liste minimale de la configuration du matériel puis qui décompresse le noyau contenu dans l'image pour ensuite le placer en "haute" mémoire. Si `initrd` est présent, la routine déplace le noyau dans la RAM et note cette action pour plus tard. Ensuite la routine appelle le noyau et le démarrage du noyau peut commencer.

Quand la `bzImage` (pour une image destinée au i386) est appelée,

vous commencez à `./arch/x86/boot/header.S` dans la routine assembleur nommée `start` (regardez la figure pour les grandes lignes). La routine effectue plusieurs initialisations basiques sur le matériel puis appelle la routine `startup_32` dans `./arch/x86/boot/compressed/head_32.S`. Cette routine met en place un environnement basique (pile,...) et nettoie le *Bloc Started by Symbol* (BSS). Le noyau est ensuite décompressé à travers l'appel d'une fonction C nommée `decompress_kernel` localisée dans `./arch/x86/boot/compressed/misc.c`. Quand le noyau est décompressé en mémoire, il est appelé. A ce moment, il y a appel d'une autre fonction `startup_32` mais cette fois dans `./arch/x86/kernel/header_32.S`

Dans la nouvelle fonction `startup_32` (aussi appelée *swapper* ou encore processus 0), les tables de pages sont initialisées et la pagination de la mémoire est activée. le type de CPU ainsi que le coprocesseur arithmétique (FPU) sont détectés durant cet appel, puis mis de coté pour être réutilisé plus tard. La fonction `start_kernel` est ensuite appelée (`init/main.c`) pour utiliser la partie du noyau non dépendante de l'architecture. Cette dernière fonction peut être considérée comme la fonction principale du noyau.



Chaîne d'appel des principales fonctions dans le noyau

Avec l'appel à `start_kernel`, une liste d'appel a des fonctions d'initialisations à lieu pour configurer les interruptions, procéder à la configuration de la mémoire et charger `initrd`. À la fin, un appel à `kernel_thread` est réalisé (dans `./arch/x86/kernel/process.c`) pour démarrer la fonction `init` (après l'appel à `cpu_idle`). Avec les interruptions activées, l'ordonnanceur pré-emptif prend périodiquement le contrôle pour fournir le multi-tâche.

Durant le démarrage du noyau, `initrd` a été chargé en mémoire par le deuxième programme puis copié en RAM et monté. `initrd` sert en tant que système de fichier principal temporaire en RAM et permet au noyau de démarrer sans avoir monté un seul disque physique. Depuis que les modules nécessaires pour interfacer le noyau avec les périphériques peuvent être une part de `initrd`, le noyau peut être très petit mais pour autant supporter un grand nombre de diverses configurations. Après que le noyau ait démarré, le système de fichiers est changé via un appel à `pivot_root`, ce qui provoque le démontage de `initrd` et le montage du vrai système de fichier principal.

Le service offert par `initrd` permet de créer de tout petits noyaux linux avec des drivers compilés en tant que modules chargeables sur demande. Ces modules donnent au noyau les moyens d'accéder aux disques et aux systèmes de fichiers présents sur ces disques, aussi bien qu'au reste du matériel. Étant donné que la

racine du système de fichier (/) pointe obligatoirement sur une partition du disque dur, le programme `initrd` fournit de quoi accéder au disque physique et monter la vraie racine du système de fichier. Dans une plateforme embarquée qui ne possède pas de disque dur, `initrd` peut être le système de fichier principal final, ou ce dernier peut aussi être monté via le réseau et la technologie NFS (Network File System)

Sortie de la fonction `decompress_kernel`

La fonction `decompress_kernel` correspond au fameux message de décompression du noyau : *"Uncompressing Linux... Ok, booting the kernel."*

7. Init

Après que le noyau ait démarré et qu'il se soit initialisé, il démarre le premier programme de l'espace utilisateur. C'est le premier programme appelé à être écrit avec la bibliothèque C standard. En effet, avant ce programme, aucun de ceux qui ont été appelés n'avaient été écrits en C standard.

Dans un ordinateur personnel, la première application utilisateur est souvent `/sbin/init`, même si ce n'est pas une obligation. Les systèmes embarqués requièrent rarement une initialisation aussi intensive que ce que fait `init` (à travers sa configuration dans `/etc/inittab`). Dans beaucoup de cas, on peut appeler un simple script `schell` qui va démarrer les applications nécessaires.

8. Résumé

Plus que linux lui même, le processus de démarrage de linux est hautement flexible, supportant un grand nombre de processeurs et d'architectures. Dans les débuts, les chargeurs d'amorçages fournissaient un moyen de démarrer le noyau sans aucune fioriture supplémentaire. LILO a étendu les capacités de démarrage mais manquait du support des systèmes de fichiers. La dernière génération de chargeurs d'amorçages, tel GRUB, permet à Linux de démarrer depuis un grand nombre de systèmes de fichier (de Minix à ReiserFs).

9. Ressource

Apprendre

- [Boot Records Revealed \(Lien64\)](#) est une source importante d'information sur les MBR et sur les divers boot-loader. Le site ne fait pas qu'étudier des MBR mais discute aussi de LILO, GRUB ainsi que des divers boot-loader de Windows.
- Regarder la page sur la géométrie des disques ([Lien65](#)) pour comprendre leur architecture. Vous trouverez un résumé des attributs des disques
- Un live-cd ([Lien66](#)) est un système d'exploitation qui démarre depuis un CD ou un DVD sans l'utilisation d'un disque dur.
- vous Boot loader showdown: Getting to know LILO and GRUB (developerWorks, August 2005) ([Lien67](#)) vous permettra d'examiner en détail LILO et GRUB
- LILO ([Lien68](#)) a été le précurseur de GRUB, mais vous pouvez toujours le trouver dans les distributions.
- la commande `mkinitrd` ([Lien69](#)) sert à créer *l'initial RAM disk*. Cette commande est très utile pour créer un premier système de fichier principal pour configurer le démarrage de façon à permettre l'accès aux blocs et accéder au vrai système de fichier principal
- Sur le site du projet Debian Linux Kernel ([Lien70](#)), vous pourrez trouver plus d'informations sur le noyau linux, le démarrage et le développement sur des architectures

embarquées.

Obtenir les logiciels

- GNU GRUB ([Lien71](#)) est un shell pour démarrer rempli d'option et très flexible.
- coreboot ([Lien72](#)) est un BIOS de remplacement. Il ne permet pas seulement de démarrer linux, mais coreboot est lui même un noyau linux
- OpenBios ([Lien73](#)) est un autre BIOS portable qui peut s'installer sur un grand nombre d'architectures comme x86, ou encore AMD64
- Sur kernel.org ([Lien74](#)), obtenez le dernier noyau stable.

A propos de l'auteur.

M. Tim Jones est un ingénieur architecte pour les systèmes embarqués et l'auteur de *GNU/Linux Application Programming, AI Application Programming, and BSD Sockets Programming from a Multilanguage Perspective*. Sa formation d'ingénieur va du développement de noyau pour la géo-synchronisation de système spatiaux à l'architecture des systèmes embarqués en passant par le développement réseau. Tim est employé en tant qu'ingénieur consultant chez Emulex Corp. à Longmont, Colorado, USA.

Retrouvez l'article de Tim Jones traduit par David Côme en ligne : [Lien75](#)



Les derniers tutoriels et articles

Apple Event 14 Octobre 2008 : Découvrez les nouveaux portables d'Apple

Résumé des annonces faites durant l'Apple Event du 14 Octobre 2008, concernant la nouvelle génération de portable, un "nouveau" procédé de fabrication, les nouveaux écrans 24", ... Et tout cela accompagnés des commentaires de la rédaction Mac.

1. Invitation

Le carton d'invitation de cet Apple Event était clair : la star de la soirée, c'était la gamme des portables Apple.

Et ce carton d'invitation a alimenté la rumeur : Certains y voyaient la photo d'un MacBook 13" en Aluminium.

Avaient-ils torts, avaient-ils raisons ?

2. Le nouveau MacBook Pro

2.1. Le nouveau MacBook Pro 15" : de grands changements



Voilà l'image de ce nouveau MacBook Pro.

2.1.1. Les changements visuels

Tout d'abord, on remarquera que l'écran du MacBook Pro est entouré d'un bandeau noir, lui donnant un air de famille avec les écrans des iMac.

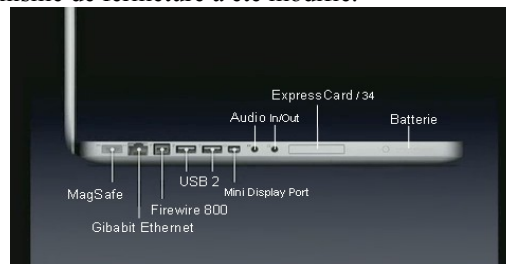
A noter que l'écran 15" n'est plus disponible qu'en Glossy.



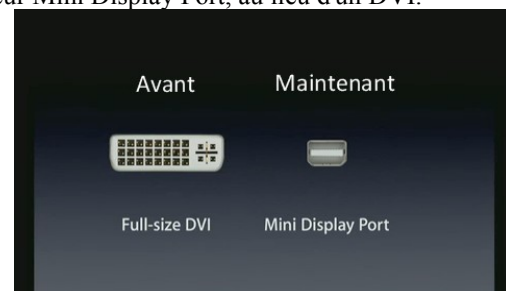
Le clavier du MacBook Pro est semblable à celui du MacBook Air.

Le Touchpad est plus grand. On notera également l'absence de boutons.

Le mécanisme de fermeture a été modifié.



Toute la connectivité a été ramenée du côté gauche. On notera la présence d'un seul port Firewire 800, et d'un connecteur Mini Display Port, au lieu d'un DVI.

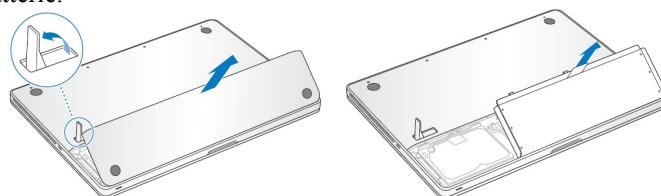


Cette image (montée) montre ô combien ce connecteur est bien plus simple.

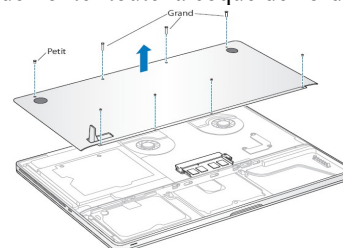
Grâce au fait que la coque est maintenant faite en UNIBODY, le MacBook Pro a encore gagné en épaisseur, puisqu'il ne fait plus que 24mm de hauteur (ou 0.95 pouces).

2.1.2. Les autres grands changements

L'accès à la batterie et au disque dur a été entièrement repensé. Ainsi, il est très facile d'ouvrir la partie inférieure de la coque de l'ordinateur qui donne accès au disque dur (format SATA) et à la batterie.

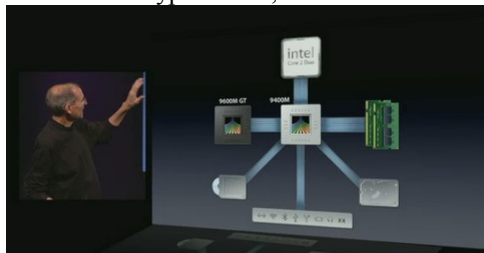


Par contre l'accès à la mémoire est plus difficile qu'auparavant, puisqu'il faudra démonter toute la coque de l'ordinateur.



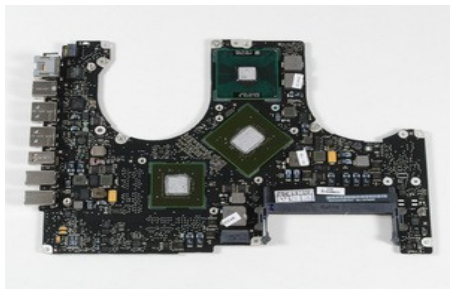
Le MacBook Pro intègre maintenant 2 cartes graphiques : La GeForce 9400M qui est une carte intégrée au chipset et la GeForce 9600M GT qui est une carte dédiée.

La mémoire vive est de type DDR3, à 1066Mhz.



A noter que du point de vue des options, on est également gâté

2.1.3. D'autres infos



Deux ventilateurs sont nécessaires pour refroidir la carte mère de ce MacBook Pro.

Le lecteur optique a maintenant une connexion SATA.

La batterie a maintenant une capacité de 50Watts/heure, au lieu de 60Watts/heure précédemment, ce qui fait que son autonomie passe de 5 heures à 4 heures lorsqu'on sollicite la carte graphique la plus puissante.

A noter qu'Apple indique qu'en utilisant la carte graphique intégrée 9400M, l'autonomie repasse à 5 heures, comme précédemment.

Au niveau Audio, les nouveaux MacBook Pro (tout comme les nouveaux MacBook Alu) continuent d'offrir des entrées numériques SP/DIF, du son 2.1 (deux hauts parleurs directionnels de chaque côté, plus un subwoofer pour les basses), mais, et cela en a étonné plus d'un, supporte maintenant en entrée les prises Jack à 4 conducteurs. Ce qui permet d'y mettre les écouteurs de l'iPhone qui contient également un micro. Et ce micro sera reconnu par Mac OS X.

Plus besoin d'acheter un micro USB comme auparavant. Enfin, on a envie de dire.

L'article d'AppleInsider ayant découvert l'amélioration au niveau audio ([Lien76](#))

2.1.4. Les Différents Modèles

Le Mac Book Pro d'entrée de gamme est repris à 1999\$, 1799€ en France et Belgique, 2549 CHF en Suisse, 2149\$ au Canada.

Comme auparavant, l'écran bénéficie d'un rétro-éclairage LED de 15,4".

Par contre, il aura 2 cartes graphiques : La GeForce 9400M qui est une carte intégrée, et qui prend sa mémoire sur celle de la mémoire vive. La GeForce 9600M GT avec 256Mo de mémoire dédiée.

Le disque Dur aura une taille de 250Go.

La mémoire vive, de type DDR3, fréquentée à 1066Mhz, sera de 2Go.

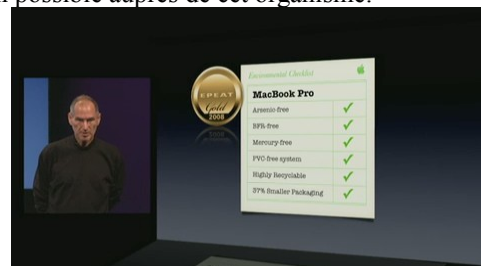


Le MacBook Pro de haut de gamme est repris à 2499\$, 2249€ en France, 3199 CHF en Suisse, 2699\$ au Canada.

Les différences par rapport au MacBook Pro 15" d'entrée de gamme sont :

- Le processeur de 2,53Ghz et 6Mo de Cache, au lieu de 2,4Ghz et 3Mo
- 4Go de RAM
- 512Mo de RAM sur la carte graphique
- Disque dur de 320Go au lieu de 250

GreenPeace sera très content d'Apple vu qu'elle a obtenu pour la première fois la médaille d'or EPEAT 2008, qui est la plus haute distinction possible auprès de cet organisme.



2.2. Et le MacBook Pro 17" ?

Et bien, le MacBook Pro 17" ne voit comme changement que son disque dur passer de 250Go à 320Go. Et son écran est maintenant disponible uniquement en 1920*1200.

Il se verra mettre à jour plus tard.

3. Le nouveau MacBook

Le MacBook a également droit à des changements, pour certains, en profondeur.

3.1. Le MacBook d'entrée de gamme

Le MacBook d'entrée de gamme est ENFIN, fournit avec le SuperDrive, un Lecteur/Graveur de DVD. Il était temps.

Et son prix passe à 949€, 999\$ US.

C'est le seul changement au niveau du MacBook d'entrée de gamme que l'on a noté.

3.2. Les MacBook Aluminium

Adieu le MacBook Noir, et le MacBook Blanc ++ Ils font place aux MacBook Aluminium.



Le MacBook Aluminium est conçu également selon le nouveau procédé de fabrication UNIBODY. Tout comme le MacBook Air et le MacBook Pro 15".

Il utilise le chipset 9400M de Nvidia, tout comme le MacBook Air et le MacBook Pro 15". Et il a également toute la connectivité sur le coté gauche. (Sur le MacBook Air, toute la connectivité est sur le coté droit).



Avoir un MacBook en Alu, c'est vraiment super. Mais avoir un MacBook sans port Firewire. Ca, c'est dur à digérer.

Ca fait mal. Très mal.

Adieu le Mode Target. Ce petit plus qui faisait qu'on aimait nos Mac.

Adieu la connexion de disque dur Firewire qui donnent de bien meilleurs débits soutenus que les disques durs USB2.

Adieu la connexion de caméra équipée de port Firewire.

Le MacBook Alu, tout comme le MacBook Pro et le MacBook Air, recoit une médaille d'or EPEAT. Pas mal.



Les deux modèles du MacBook Alu remplace le MacBook Blanc haut de gamme et le MacBook Noir.



4. Conclusion

Ce qu'on aime beaucoup

- L'uniformité visuelle de la gamme des portables, puisqu'on retrouve la même coque, le même TrackPad Multi-Touch, le même clavier, des écrans à rétro-éclairage LED, ...
- La MacBook d'entrée de gamme qui continue à être vendue est ENFIN avec un graveur de DVD.
- Le châssis en aluminium UNIBODY qui renforce encore la solidité des MacBook Pro, et qui équipe également les nouveaux MacBook
- Le rétro-éclairage LED qui équipe les MacBook Alu et tous les MacBook Pro
- Le Trackpad Multi-Touch
- Le clavier rétro-éclairé sur le MacBook de haut de gamme.
- Une carte graphique digne de ce nom dans le MacBook
- La lecture des vidéos qui profite de la puissance du GPU, soulageant le processeur
- Un portable Mac en Alu à la portée de la bourse
- Le connecteur mini DisplayPort
- Accès au disque dur grandement simplifié
- Le support des micros non amplifiés en entrée

Ce qu'on n'aime pas du tout

- Pas de port Firewire sur le MacBook
- Pas d'adaptateurs graphiques inclus avec les MacBook/MacBook Pro
- Pas de grand gain de performance au niveau processeur et mémoire
- Accès à la mémoire moins aisé qu'auparavant
- Dalle d'écran uniquement disponible en brillant, même sur le MacBook Pro

On ne recommande pas du tout l'achat du MacBook Pro 17" qui doit encore connaître sa métamorphose, ni du MacBook Blanc, qui est clairement un produit has been.

Comme le disait M. Ive dans la vidéo ([Lien77](#)), ainsi que Steve durant la keynote ([Lien78](#)), le MacBook est maintenant aussi beau à l'intérieur qu'à l'extérieur.

Retrouvez la suite de l'article de la rédaction Mac en ligne : [Lien79](#)

Régler les problèmes de permission avec AFP sous Leopard Server

Pour la rentrée, je vous propose un article plus long que la moyenne, et qui aborde les problématiques de gestion des permissions sous AFP. Car sous Mac OS X Server 10.5, j'ai constaté que de nombreux administrateurs (moi y compris, avant de me pencher vraiment sur le sujet il y a quelques mois de cela) ont eu du mal avec le fonctionnement des permissions AFP.

1. Cas classique

Pour la rentrée, je vous propose un article plus long que la moyenne, et qui aborde les problématiques de gestion des permissions sous AFP. Car sous Mac OS X Server 10.5, j'ai constaté que de nombreux administrateurs (moi y compris, avant de me pencher vraiment sur le sujet il y a quelques mois de cela) ont eu du mal avec le fonctionnement des permissions. Cas classique :

- Un utilisateur A appartenant à un groupe (appelons-le *Travail*) crée un fichier Toto.doc.
- Un administrateur a créé un point de partage "*Boulot*" sur un serveur AFP sous 10.5. Il lui a attribué les

autorisations POSIX suivantes : Possesseur : lui-même, en lecture/écriture ; Groupe : *Travail*, en lecture et écriture ; Autres : lecture seule

- L'utilisateur A copie son fichier toto.doc sur le point de partage, via AFP ;
- L'utilisateur B, appartenant aussi au groupe *Travail*, essaie de modifier le fichier toto.doc.

Et là ... impossible ! Le système lui indique qu'il n'a pas les autorisations suffisantes pour modifier le fichier. Pourtant, en toute logique, le fichier devrait avoir des permissions en lecture/écriture pour les membres du groupe travail... sauf que ce n'est pas le cas, le groupe attribué au fichier ne disposant que de la

lecture seule.

Comment est-ce possible ?

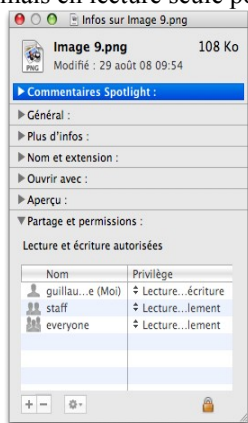
Pourquoi les autorisations sont attribuées de façon incorrecte au fichier ?

On pourrait croire qu'il s'agit d'un bug. Pourtant, le comportement est *exactement* celui attendu. Le problème est plus insidieux, et la faute en incombe aux... ACLs. Mais l'explication est particulièrement tordue, et, pour la comprendre, il faut remonter de quelques années en arrière...

2. Avant Mac OS X Server 10.2.4

Avant l'arrivée de Jaguar Server dans sa version 10.2.4, le comportement du partage de fichier respectait les autorisations Posix. Cela signifie que les autorisations d'un fichier copié sur un serveur étaient les mêmes que celles du fichier source, c'est à dire que le masque appliqué est identique entre le fichier source et la copie sur le serveur. En pratique, un fichier dont le groupe est en lecture seule sur la machine cliente *restera en lecture seule s'il est copié sur le serveur*.

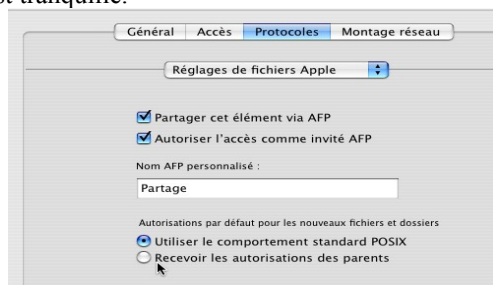
Mais cela posait de nombreux problèmes pour les utilisateurs et les administrateurs informatiques, car cela empêchait le transfert des informations entre utilisateurs. Pourquoi ? Parce que le masque appliqué par défaut à tous les fichiers créés sur Mac OS X est 022, ce qui fait qu'un fichier est toujours en lecture/écriture pour son possesseur, mais en lecture seule pour son groupe.



Quand on copie un fichier sur le serveur, c'est ce masque qui est conservé. Si chaque fichier généré sous Mac OS X se voyait attribué un masque en lecture/écriture pour le groupe (002), il n'y aurait aucun souci, le masque étant transféré sur le serveur, et les autorisations seraient alors correctes.

3. De Mac OS X 10.2.4 à 10.4 : l'héritage des permissions

Comme modifier le comportement de Mac OS X impliquait des soucis de sécurité un peu plus larges, Apple a donc modifié le client et le service AFP en 10.2.4 pour permettre l'héritage des permissions, comme expliqué dans l'article 107623 de la Knowledge Base ([Lien80](#)). Ainsi, un fichier récupère les permissions d'accès du groupe associé au dossier où il est placé. Il suffit de cocher la case *Recevoir les autorisations des parents*, et hop, on est tranquille.



C'est ce que l'on attend en fait la plupart du temps, et c'était d'ailleurs le comportement de ce bon vieil AppleShare IP ou du partage de fichiers personnel sous Mac OS 9. Et finalement, ça marchait très bien. Sauf qu'une des grosses nouveautés de Mac OS X Server 10.4 allait imposer un nouveau changement, très attendu mais assez pernicieux.

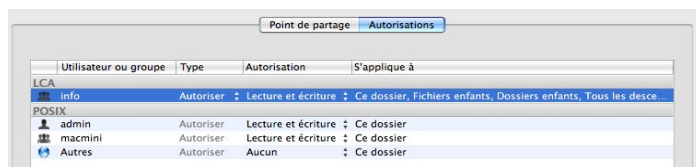
4. Mac OS X Server 10.4 : bienvenue aux ACL !

Les ACL (*Access Control Lists*, appelées aussi *listes de contrôle d'accès* ou *LCA*) ont apporté une grande souplesse d'utilisation dans la gestion des permissions. Cependant, sous Mac OS X Server 10.4, il faut explicitement activer les ACL pour chaque volume.

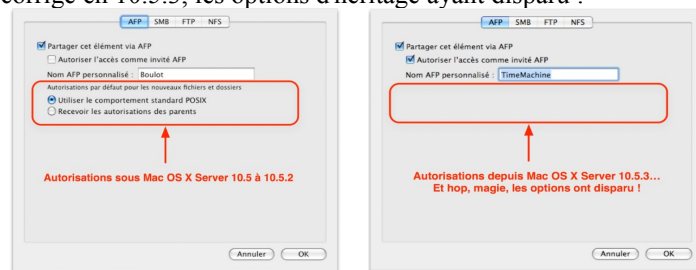
Si les ACL ne sont pas actives, les permissions d'un partage peuvent toujours être attribuées façon POSIX ou via l'héritage des permissions. Mais si elles sont actives, l'héritage ne fonctionne plus, d'ailleurs la case ad hoc est désactivée. Et devinez quoi ? Dans ce cas, c'est l'attribution des autorisations façon POSIX qui prend le relais par défaut ! Il faut alors éventuellement rajouter des autorisations via des ACL pour que les différents utilisateurs puissent modifier le fichier.

5. Mac OS X Server 10.5 : ACL obligatoires

Après toute cette description, vous comprendrez peut-être ce qui se passe sous Mac OS X Server 10.5 : les ACL sont systématiquement actives *par défaut* et ne peuvent pas être désactivées (il n'y a aucune case qui le permette dans Server Admin ou ailleurs). Ce qui implique que, par défaut, c'est toujours le comportement POSIX qui prime. Et du coup, il est relativement simple de corriger le souci et permettre à nos deux utilisateurs de s'échanger leurs fichiers : il suffit de rajouter le groupe en lecture/écriture... sous forme d'entrée dans les ACL. C'est à dire, dans Server Admin, de glisser le groupe *Travail* dans la ligne LCA, et non pas la ligne Groupe en POSIX, et lui attribuer les autorisations en lecture/écriture.



Mais pourquoi tant de confusion ? D'abord, l'interface de Server Admin a souffert durant quelques versions d'un oubli gênant : jusqu'à Mac OS X Server 10.5.2 inclus, la case d'héritage des permissions du parent était toujours visible dans l'interface graphique, quand bien même elle n'avait aucun effet. Cela a été corrigé en 10.5.3, les options d'héritage ayant disparu :



La faute aussi à la documentation de Mac OS X Server, qui manque probablement de clarté quand à la différence entre la gestion POSIX et les ACL, et leur influence sur un modèle de partage de fichiers assez courant. Une grande partie des problèmes vient également du masque par défaut appliqué aux fichiers sous Mac OS X : si chaque groupe avait par défaut l'accès en lecture et écriture sur tout nouveau fichier généré, on aurait

beaucoup moins de soucis...

Enfin, une partie de ce comportement vient de la difficulté à concevoir un modèle où la sécurité, les origines UNIX de Mac OS X et les besoins des utilisateurs d'un système graphique puissent coexister paisiblement. Et finalement, quand on comprend le fonctionnement de Mac OS X, on comprend aussi que l'héritage des permissions était un patch qu'il était nécessaire de supprimer, les ACL étant gérées au niveau du noyau de Mac OS X et non pas du seul client AFP.

Vous savez donc désormais comment agir : ne vous occupez plus spécialement des autorisations POSIX, et concentrez-vous sur les ACL. Vous pourrez ainsi gérer vos autorisations du partage de fichiers plus efficacement et de façon bien plus souple. Au

passage, lisez la documentation ([Lien81](#)) de Mac OS X Server 10.5, en particulier celle-ci ([Lien82](#)) (PDF, 1,4 Mo).

6. Remarques

N.B. : ceux qui ont installé leur serveur 10.5 en mode standard (et non pas avancé) n'ont même pas souffert de ce problème. Pourquoi ? Parce que dans l'interface des Préférences Serveur, la notion de POSIX n'est pas représentée : en réalité, tout est géré via les ACL...

N.B.2 : une autre solution aurait pu consister à modifier le masque global appliqué aux fichiers, mais il faut le faire aussi sur chaque poste client... Voir ce document ([Lien83](#)) pour la méthode.

Retrouvez l'article de Guillaume Gete en ligne : [Lien84](#)

Découvrez comment automatiser votre travail

Avec Tiger (Mac OS X.4) puis Léopard (Mac OS X.5), Apple a introduit un nouvel outil pour vous aider à construire des traitements automatiques sur vos machines.

Alors qu'avant il fallait passer par le langage de programmation AppleScript pour réaliser des flots de traitements automatiques, Apple vous propose avec Automator un outil plus visuel, plus ouvert, pour fabriquer ces mêmes tâches.

1. Présentation préliminaire

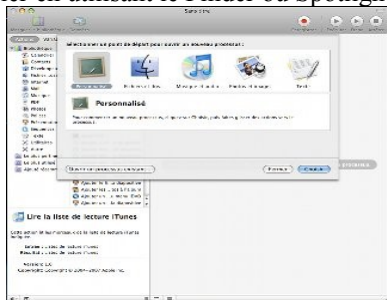
1.1. Que vous permet de faire Automator ?

Concrètement le logiciel Automator vous permet de construire des enchaînements d'actions que l'on appelle *processus*.

Une fois construit, un processus peut être réutilisé à volonté et permet donc d'automatiser des tâches répétitives.

1.2. Où trouver Automator ?

Automator est une application. À ce titre il se trouve dans le dossier "Applications" de votre Mac. Vous pouvez naviguer jusqu'à ce dossier en utilisant le Finder ou Spotlight.



L'application Automator vous permet donc de construire un processus. Mais une fois le processus mis au point vous pouvez le sauvegarder pour le réutiliser plus tard avec d'autres documents en entrée.

Vous avez le choix entre :

- Construire une application indépendante ;
- Sauver votre processus comme un module d'une autre application.

Dans Mac OS X.5, Automator peut générer des modules pour :

- le Finder ;
- les actions de dossier ;
- une alarme iCal ;
- l'application Transfert d'Images ;
- le système d'impression ;
- le menu des scripts.

1.3. Qu'est-ce qu'une action ?

L'action est la brique de base pour construire un processus. Une

action se définit par :

- les paramètres d'entrée : les données qu'elle utilise ;
- le traitement qu'elle réalise sur ses paramètres ou à partir de ses paramètres ;
- les résultats qu'elle rend : les données de sortie.

1.4. Qu'est-ce qu'un processus ?

Un processus est un enchaînement d'actions qui utilise un ensemble d'objets pour leur appliquer un traitement spécifique.

Sous cette définition relativement générique on peut imaginer une action qui s'applique sur un ensemble de documents pour les publier sur un site internet.

En règle générale, une action accepte en entrée un ensemble de fichiers ou de dossiers et rend un résultat sous la forme du même ensemble de fichiers ou dossiers avec le traitement appliqué. Selon l'action, l'objet d'origine peut avoir été modifié ou non par le traitement.

2. À la découverte de l'éditeur de processus

2.1. Lancez Automator.

Vous allez avoir une fenêtre qui sera partiellement recouverte d'une boîte de dialogue où l'application vous demandera quel type de processus créer.



Ces modèles de processus donnent un point de départ pour :

- manipuler des fichiers à partir du Finder ;
- manipuler des fichiers audio ;
- manipuler photos et images ;
- manipuler du texte.

Le modèle *Personnalisé* permet de construire un processus sans aucun point de départ.

Le bas de la boîte de dialogue est dédié aux actions :

- À gauche, un bouton permet d'ouvrir un processus existant ;
- Le bouton *Fermer* annule la création du processus et ferme la fenêtre de l'application ;
- *Choisir* est utilisé pour valider le choix du point de départ et pour construire un nouveau processus.

2.2. Création d'un processus

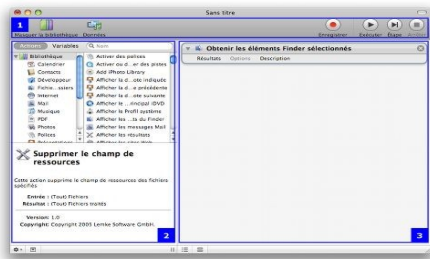
Choisissez un processus "Fichiers et dossiers".



Remarquez qu'une série d'options vous est proposée pour affiner le modèle de processus.

Vous pouvez conserver les options par défaut. Cliquez sur le bouton "Choisir".

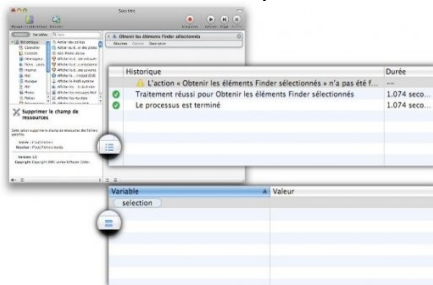
La fenêtre de l'éditeur de processus est maintenant affichée et présente par défaut trois zones que nous allons détailler ci-dessous.



1. La barre d'outils, personnalisable, qui présente les principales actions disponibles.
2. La bibliothèque affiche l'ensemble des actions et variables disponibles. Cette partie peut être masquée et affichée à volonté.
3. L'éditeur est la partie principale. C'est ici que l'on peut voir les actions s'enchaîner.

Deux éléments complémentaires sont également disponibles lors de la mise au point du processus :

- Le journal d'exécution.
- La liste des variables du script.



3. Passez à l'action

3.1. Les différentes étapes

Vous allez maintenant mettre en oeuvre votre premier processus. Il va consister à produire une planche contact pour une sélection de photos.

Nous avons besoin d'une sélection de quelques photos à partir de l'application iPhoto. Cette sélection va ensuite être copiée dans un

dossier placé sur le bureau. Enfin, l'ensemble de ses images va être utilisé pour produire la planche contact.

Les étapes du processus seront donc :

1. Sélectionner des images dans iPhoto.
2. Copier ces photos dans un dossier pour les manipuler sans modifier les originaux.
3. Appliquer un cadre autour des photos.
4. Produire la planche contact.

3.2. La sélection des images

1. Commencez par ouvrir Automator et sélectionnez un processus "Photos et images".
2. Dans les options, choisissez les options "Obtenir le contenu de : ma bibliothèque iPhoto", puis "Demandez des photos et albums lors de l'exécution de mon processus".
3. Validez avec le bouton "Choisir".

Vous êtes maintenant dans l'éditeur de processus. Une action "Demander des photos" a été ajoutée comme point de départ de notre nouveau processus.



Vous pouvez saisir dans le champ "Invitez :" un message à destination de l'utilisateur.

- Tapez un message comme "Sélectionnez vos photos pour la planche contact."
- Pour pouvoir manipuler plusieurs photos conservez l'option "Autoriser des sélections multiples".

3.3. La copie dans un dossier

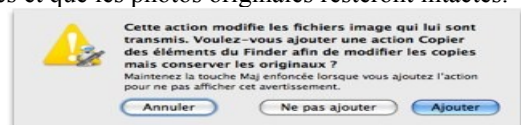
Nous allons copier les photos dans un dossier "Planche contact" sur le bureau pour leur appliquer une bordure.

Affichez la bibliothèque d'actions si elle est masquée. Dans le champ de recherche tapez simplement "bord" pour rechercher les actions relatives à ce terme de bordure.



Faites glisser l'action trouvée dans l'éditeur situé à droite.

Une feuille de dialogue vous alerte que les images seront modifiées et qu'il est préférable de les copier dans un nouvel emplacement. Cela vous garantit que seules les copies seront modifiées et que les photos originales resteront intactes.



Cliquez sur le bouton "Ajouter". Votre processus contient maintenant trois actions :

1. Demander des photos
2. Copier des éléments du Finder

3. Ajouter une bordure aux images.



Vous pouvez remarquer une flèche qui descend de l'action précédente pour s'incruster dans un arrondi situé sur le haut de l'action suivante. Ce lien entre les actions indique que le résultat de l'action du dessus est repris par l'action suivante pour être à nouveau manipulé.

Les photos sélectionnées dans la première action sont ainsi transmises à l'action de copie pour être dupliquées dans un autre emplacement. Les copies résultantes sont à leur tour transmises à l'action qui va ajouter la bordure.

Il faut définir le dossier cible pour stocker les duplicatas. Dans l'action de copie des photos vous allez choisir dans la liste "Vers :" la valeur "Autre..." . Un dialogue de fichier s'ouvre et il vous faut créer le dossier "Planche contact" sur le bureau puis valider.

Cochez également l'option "Remplacer les fichiers existants". Celui-ci vous évitera un échec du processus si des fichiers de même nom existent. Ils seront alors simplement remplacés.

3.4. Traitement des photos

Pour le traitement des photos, l'option est déjà présente et, dans un premier temps, il suffit de définir les dimensions des photos cibles. Cochez simplement l'option "Dimensionner l'image avant le montage" pour s'assurer que les photos tiendront dans les dimensions indiquées.

3.5. Produire la planche contact

Nous pouvons maintenant utiliser les photos encadrées de noir pour fabriquer une planche contact. Comme précédemment il suffit de chercher une action en tapant "contact".

Dans la liste d'actions trouvées il suffit de faire glisser "Nouvelle planche contact PDF" à la fin du processus.

Saisissez un nom de fichier pour la planche contact et c'est fini.

Blogs Mac

Créer une image ISO d'un CD/DVD de PC sous Mac

Dans un précédent post ([Lien86](#)), j'avais détaillé une méthode permettant de faire facilement des images ISO d'un CD/DVD.

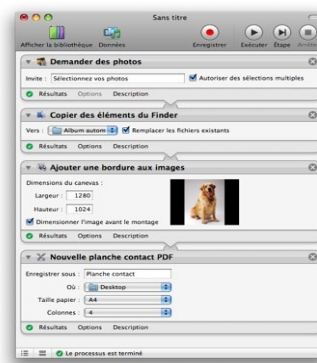
Je me suis malheureusement rendu compte que si cette méthode fonctionnait bien pour les CD/DVD de Mac (au format HFS) cela ne fonctionnait pas du tout pour les CD/DVD formatés pour Windows/Linux (ISO/JOLIET, UDF).

Ce post vous fournit donc la méthode à utiliser pour ces disques:

Insérez le CD/DVD dont vous souhaitez faire l'image. Il va apparaître sur le bureau. Notez vous son nom. Dans notre exemple il s'appellera "MONCD1".

Ouvrez un terminal

Entrez y la commande suivante:



3.6. Lancer votre processus

Sélectionnez quelques photos et après quelques secondes vous devriez voir apparaître un nouveau document PDF qui présente votre sélection de photos.

Sauvez ce processus si vous souhaitez le réutiliser ultérieurement. Je vous conseille de les stocker dans un dossier "Processus" dans votre dossier maison.

4. Conclusion

Vous venez d'avoir un aperçu rapide d'Automator.

Cependant cette visite guidée d'introduction est bien loin d'avoir exploré toutes les capacités de cette application.

Il est en effet possible d'aller bien plus loin. Par exemple :

- On peut utiliser comme action des scripts écrits dans d'autres langages comme AppleScript et des scripts de shell Unix.
- Il est possible d'utiliser des variables pour stocker les résultats d'une action et les réutiliser plus loin.
- Un processus peut en appeler un autre.
- Un processus peut utiliser un service web.
- Automator peut enregistrer les actions de l'utilisateur et les transformer en processus.

Les développeurs peuvent également proposer des actions soit pour intégrer leurs applications avec Automator, soit pour proposer des actions inédites.

D'autres articles vous permettront de découvrir des exemples multiples de processus et pousser plus en avant l'exploration de cette technologie.

Retrouvez l'article de Sylvain Gamel en ligne : [Lien85](#)

```
hdiutil makehybrid -o moncd1.iso -udf -iso -joliet /Volumes/MONCD1
```

Laissez ensuite l'outil faire son travail. Vous pouvez tester le fichier iso créée en double cliquant dessus depuis une fenêtre du finder. L'image sera alors montée.

Dans la commande ci-dessus *moncd1.iso* sera le fichier image créée.

/Volumes/ est une constante que vous devrez saisir quelque soit le nom du CD/DVD dont vous souhaitez faire une image. *MONCD1* est le nom du disque tel qu'il apparaît sur votre bureau.

J'ai lu sur certains forums que *hdiutil* ne serait pas capable de faire des images de certains DVD. Il semble que sa limite de taille d'image soit à 4,7Go. Si quelqu'un pouvait me confirmer/infirmier cela, ce serait très gentil.

Retrouvez ce billet sur le blog de Neilos : [Lien87](#)

Mac OS X Leopard. Le livre des secrets

Ce livre est une référence pour tous les utilisateurs de Mac OS X, débutants ou avertis. Il montre comment tirer parti de toutes les ressources de ce système d'exploitation, pour maîtriser ses principales applications, optimiser son fonctionnement, activer de nouvelles fonctions, résoudre les problèmes les plus courants et découvrir les secrets que masque l'interface graphique du système.

En effet, Mac OS X ne se contente pas d'allier ergonomie et simplicité. Il est aussi l'héritier novateur du système le plus fiable au monde, Unix. Ce livre amène progressivement à découvrir ces deux aspects. Ainsi, vous apprendrez comment vous réparer dans Mac OS X 10.5, comment profiter au mieux des applications livrées avec le système et tirer parti de leurs **ressources cachées**, quels sont les **remèdes aux pannes**, aux erreurs et aux blocages les plus embarrassants, comment exploiter la puissance d'Unix et **écrire des scripts personnalisés**, quelles sont les recettes pour configurer la messagerie intégrée et le serveur web Apache, pour créer un site FTP et améliorer le firewall intégré, etc.

Ce livre met l'accent sur les **nouveautés de Mac OS X 10.5 Leopard** en s'appuyant notamment sur les dernières versions du Finder, de Mail, de Safari, d'Aperçu, de TextEdit, d'iTunes et d'autres applications. Il fait le point sur les nouvelles commandes du Terminal, sur les évolutions de Postfix et d'IPFW et s'attarde sur les fonctions de personnalisation et de sécurisation du système.

Critique du livre par Vincent Brabant

Bien que le dos du livre indique qu'il s'adresse aussi bien aux débutants qu'aux avertis, je ne le conseillerais pas à quelqu'un qui débute vraiment sous Mac OS X et qui espère pouvoir être pris par la main pour découvrir son nouvel environnement. D'autres livres comme celui de David Pogue ou celui de chez Agnosys me semblent plus indiqués.

Par contre, je conseille très fortement ce livre à ceux qui désirent en savoir plus sur les rouages internes à Mac OS X Leopard. Et qui veulent aller plus loin, comme installer et configurer un autre pare-feu que celui livré avec Mac OS X ou installer et configurer un serveur de messagerie, ...

Mon Mac & moi : iWork '08

Oubliez les suites bureautiques lourdes et compliquées : avec iWork '08, vous disposez de trois applications pour réaliser des documents à l'aspect professionnel, mais toujours avec la simplicité d'utilisation reconnue des logiciels Apple.

Créez avec Pages des rapports ou des invitations de toute beauté. Exploitez la puissance de Numbers pour réaliser des tableaux attractifs agrémentés de superbes graphiques en quelques secondes. Et devenez le roi des présentations spectaculaires pour vos confrères, amis ou partenaires professionnels grâce à Keynote.

Avec ses multiples astuces et conseils exclusifs, cet opus de la collection *Mon Mac et Moi* rendra votre utilisation de la suite iWork '08 encore plus efficace et agréable.

Mon Mac & Moi : Mieux comprendre et Mieux utiliser

Une collection d'ouvrages ludiques, accessibles et tout en couleurs, permettant d'être rapidement efficace sur les

fonctionnalités multiples et diverses de votre Mac. Tous les titres sont développés par des formateurs professionnels que vous pouvez mieux connaître en vous connectant sur le site officiel de la collection : www.monmacetmoi.com. Vous y trouverez également les informations sur les prochaines publications ainsi que sur notre réseau de revendeurs.

Critique du livre par Vincent Brabant

C'est le deuxième livre de chez Agnosys que je lis, et je suis, encore une fois, conquis par la qualité d'ensemble du livre.

Les auteurs du livre font vraiment un excellent usage des couleurs pour illustrer leur propos, conseils, et nous guider dans la découverte de cette suite bureautique.

En en peu plus de 150 pages, ils nous expliquent comment tirer profit de Pages, de Numbers et de Keynote, réservant une place de choix pour Page, ce qui est tout à fait justifié vu non seulement la richesse de ses fonctionnalités, mais aussi parce qu'ils ne reviennent plus pour Numbers et Keynote sur les fonctions communes avec Pages.

Ce livre m'a vraiment conquis. Car très bien illustré, les avis/commentaires sont vraiment à propos, et de plus, cela devrait me permettre de faire maintenant de meilleures présentations que dans le passé.

La seule chose que je reprocherais à ce livre, est qu'il n'aille pas un peu plus en profondeur pour certaines choses, comme par exemple comment rendre ses présentations Keynote disponibles sur l'iTunes Store.

Mais en 150 pages, on ne peut pas tout couvrir non plus. Peut-être l'occasion de faire un livre "Aller plus loin avec iWork '08" ?

Mac OS X Leopard efficace. Déploiement, administration et réparation

Tous les rouages du Mac pour une administration efficace.

- Comprenez l'architecture de Mac OS X et mesurez l'impact de la migration vers Intel.
- Maîtrisez le système de fichiers et la gestion des droits.
- Organisez vos sessions utilisateurs et vos mots de passe
- Installez et supprimez proprement applications et polices.
- Intégrez Leopard dans un réseau hétérogène (Windows, Linux...)
- Surmontez les problèmes de compatibilité et de configuration des imprimantes.
- Contrôlez et administrez des Mac à distance avec Apple Remote Desktop et le Terminal.
- Partitionnez vos disques et déployez Mac OS X sur un parc de machines.
- Etablissez une stratégie de sauvegarde et de chiffrement.
- Automatisez les tâches quotidiennes d'administration avec Automator, AppleScript et les scripts shell.
- Choisissez la bonne solution de virtualisation pour profiter des autres systèmes d'exploitation

A qui s'adresse cet ouvrage ?

- Aux utilisateurs experts qui souhaitent s'approprier les secrets de Leopard
- À tous ceux qui utilisent professionnellement un Mac

- Aux administrateurs, connaisseurs ou non du monde Apple

Mac OS X Leopard pour les pros

Mac OS X n'a jamais été aussi séduisant. Tirant parti des possibilités des processeurs Intel, le Mac met un pied dans le monde PC, sans perdre une once de son identité, de son ergonomie, et de la compatibilité avec les PowerPC. Leopard propose quantité de nouveautés pour satisfaire tant les utilisateurs aficionados que les administrateurs ne sacrifiant rien à la sécurité ni à l'efficacité de leurs réseaux. Autant de fonctions parfois cachées derrière la belle interface du système, dont la maîtrise s'impose désormais dans le monde professionnel.

Consultant Macintosh sous le prestigieux label *Apple Distinguished Professional*, Guillaume Gete est fondateur de Gete.Net Consulting. Auteur de l'un des sites de référence du monde Mac, www.gete.net ([Lien88](#)), il propose ses services de formateur Apple et collabore régulièrement aux principaux magazines spécialisés sur Mac (SVM Mac, Univers Mac, Mac & Co, ...).

Critique du livre par Vincent Brabant

Je conseille fortement ce livre à tous ceux qui veulent connaître leur machine, mais aussi les rouages internes, la face cachée par le GUI de Mac OS X Leopard.

Je trouve personnellement que l'auteur de ce livre a une façon d'expliquer les choses qui fait que même des choses complexes comme Kerberos deviennent claires.

Aussi, c'est la première fois qu'on m'explique aussi clairement le fonctionnement de FileVault, pour ne citer que cet autre exemple.

Ce ne sont que 2 exemples que je cite ici, mais c'est comme cela tout au long du livre.

De plus, les dessins humoristiques, et le style d'écriture de l'auteur font que c'est un livre très agréable à lire, même lorsqu'il aborde des sujets fort techniques comme Kerberos (oui, je sais, je l'ai déjà cité, l'aisance et la facilité avec laquelle l'auteur l'a expliqué m'a vraiment frappé).

Et la mise en page du livre, et la quasi absence de coquilles ne font que rehausser ce sentiment d'avoir un livre de qualité dans les mains. Aussi bien du point de vue présentation, mais surtout de son contenu.

A posséder absolument !!!

Big book of Apple hacks

Tips & Tools for unlocking the power of your apple devices

- Optimize your operating system, whether you have Mac OS X Leopard or Tiger
- Customize the applications that come with Mac OS X, including Mail, Safari, Dashboard, and the iLife suite
- Tweak system and device settings in minutes with Quick Hacks
- Perfect your peripherals with hacks that hone your hardware, from apple TV to the new iPod Touch
- Protect your data with backups and keep your secrets by tightening security
- Open up the iPhone and iPod to all kind of possibilities
- Run Windows and other operating systems on your Mac

Big Book of Apple Hacks

Bigger in size, longer in length and broader in scope, this new collection of tips tricks and hacks lets you get the most out of Mac OS X, your iPhone, the new line of iPods, and Apple TV. Want to tweak system preferences ? Alter or add keyboard shortcuts ? Connect drives and devices ? These hacks take you under the hood with complete step-by-step instructions so you can tinker in ways that Apple doesn't expect. finally, you can control the systems you own !

Critique du livre par Vincent Brabant

Ce livre est quelque peu particulier dans le sens où il ne propose pas que des recettes (Hacks) pour nos Macs, mais propose également des recettes pour d'autres accessoires comme les iPods, iPhones, ...

Les recettes sont variées, et concerne aussi bien la maintenance, que la sécurité, les copies de sauvegarde, personnaliser les applications, ...

On a même droit à des recettes dédiées toutes spécialement à Leopard.

Vous découvrirez également comment changer le disque dur d'un Mac Book, comment faire tourner un Mac Mini dans sa voiture, continuer à surfer sur internet à l'aide de son portable et de son routeur adsl, alors qu'on a une coupure de courant dans la maison, comment utiliser son Mac pour la domotique, ...

J'ai vraiment beaucoup apprécié ce livre, qui m'a même fait découvrir les écrans de maintenant de mon iPod Nano 3G.

Attention que les recettes concernant les iPod et les iPhone ne concerne pas les iPods sortis en septembre, ni les iPhone 3G ou ayant le firmware 2.x.

Ce qui est normal vu que le livre est sorti bien avant. Mais je tenais à attirer votre attention sur ce point.

Autre petite remarque : ce livre n'est disponible qu'en Anglais.

Retrouvez ces critiques de livre sur la page livres Mac : [Lien89](#)

Blogs 2D/3D/Jeux

Que s'est-il passé à la GC de Leipzig?



Nous allons revenir aujourd'hui sur le dernier grand événement en date du jeu-vidéo, à savoir la **Games Convention de Leipzig**. Le plus grand salon européen des loisirs vidéo ludiques s'est déroulé du 21 au 24 Aout 2008 avec une journée spéciale le 20 Aout réservé aux visiteurs professionnels. Cette année, le salon a accueilli un nombre record de visiteurs à savoir près de 203 000 personnes. Les 547 exposants, répartis sur 115 000 m², représentaient internationalement 234 sociétés.



De grands noms ont apportés leur touche de nouveauté ou d'actualité. **Les Sims 3** ont enfin annoncés leur date de sortie à savoir le 20 février 2009. **Grand Theft Auto IV**, quand à lui, sortira dans une version PC en novembre. Annuellement, nous participons à la guerre des jeux de foot. Ainsi **FIFA 2009** et **Pro Evolution Soccer 2009** sortiront en cette fin d'année. Le second pourra même être accompagné d'un pad spécial optimisé reconnaissant beaucoup plus de mouvements sur Xbox360. Les grands titres attendus se sont montrés un peu plus en détails comme **Street Fighter 4**, **Operation Flashpoint 2 : Dragon Rising**, **Starcraft 2** et **Diablo 3**. Les classiques **Resident Evil 5**, **Need For Speed : Undercover** et **Prince of Persia** étaient également de la fête avec des orientations complètement différentes des versions précédentes. Enfin, le jeu d'un studio français a fait grandes impressions, **FUEL** d'Asobo Studio. Ce jeu de course off-road s'est montré en quelques images et semble très prometteur.



Lors du salon, Sony fut le seul constructeur à organiser une conférence pour annoncer quelques nouveautés. Parmi elles, on notera principalement **un nouveau pack pour la PlayStation 3** avec un disque dur de 160Go au tarif de 449€ prévu pour fin octobre. **Un clavier sans-fil pour la manette** de la PS3 a également été présenté avec la possibilité de raccorder une souris. La nouvelle version de la console portable de Sony a également montré le bout de son nez. **Le modèle 3000 de la Playstation Portable** présente quelques améliorations et arrivera au mois d'octobre au prix de 199€ en pack avec un jeu.



Lors du salon, les organisateurs ont attribué certaines récompenses dans différentes catégories :

- Meilleur jeu sur PC : **Spore** - Electronic Arts
- Meilleur jeu sur Xbox 360 : **Mirror's Edge** - Electronic Arts
- Meilleur jeu sur PSP : **Resistance Retribution** - Sony
- Meilleur jeu sur PS3 : **LittleBigPlanet** - Sony
- Meilleur jeu sur Wii : **Skate It** - Electronic Arts
- Meilleur jeu sur DS : **Sonic Chronicles : La Confrerie des Tenebres** - SEGA
- Meilleur jeu sur mobile : **Pro Evolution Soccer 2009** - Konami
- Meilleur jeu online : **Warhammer Online : Age of Reckoning** - GOA
- Meilleur hardware : **PlayTV** - Sony Computer Entertainment

Pour finir, quelques réactions de grands noms du jeu-vidéo à propos du salon :

Jörg Trouvain, Vice Président Senior de European Publishing Activision Blizzard

"La GC 2008 à Leipzig a réussi à se surpasser. Chez Activision Blizzard, nous sommes très heureux de la manière dont elle s'est déroulée. Les exposants et les organisateurs ont accompli quelque chose d'incroyable et nos clients, partenaires, invités et visiteurs ont vraiment été ravis. De nos point de vue, la GC est le meilleur mondialement dans son domaine."

Dr. Olaf Coenen, Directeur de Electronic Arts Allemagne

"De notre point de vue, la Games Convention a été un succès complet. Nous avons eu un formidable public varié dans les allées qui a répondu très enthousiaste à un large choix de jeux. L'intérêt des médias a été plus grand que les années précédentes et tout ceux qui en ont pris part ont accompli ensemble le positionnement des jeux vidéos au centre de la société."

Ralf Wirsing, Manageur Général d'Ubisoft Allemagne

"Comparativement à l'année précédente, la Games Convention s'est encore plus internationalisé. Elle a montré à tous la présence de la presse venant du monde entier sur notre industrie. Comme les années précédentes, nos équipes de ventes ont été capable, à la GC, d'établir les fondations pour un marché de la période de Noël réussi. Dans le processus, nous avons clairement dépassé les cibles que nous nous étions fixées."



Le dernier jour de l'évènement, Wolfgang Marzin, CEO du Leipziger Messe GmbH a annoncé que *"la Games Convention sera de retour à Leipzig en 2009"*. Il annonça également qu'il se déroulera du 19 au 23 Aout 2009.

A l'année prochaine avec peut-être de grandes surprises !!

Retrouvez ce billet sur le blog de la rubrique 2D/3D/Jeux : [Lien90](#)

Conception d'un moteur physique

Ce tutoriel a pour but de présenter un projet qui pourrait constituer un début de moteur physique. Reposant sur des formes géométriques de base, nous avons souhaité implémenter les lois de la dynamique.

1. Mon travail

Dans le cadre d'un projet d'étude mené lors de ma dernière année de cycle ingénieur, j'ai développé avec l'aide d'un ami, **Julien Peyre**, un début de moteur physique. Nous avons géré les collisions pour des primitives de base.

- Ce projet a été développé sous Visual C++ Express.
- Pour faire le rendu de nos calculs, nous avons utilisé OpenGL
- Quelques vidéos d'exemples sont disponibles ici ([Lien91](#))
- Les sources sont là ([Lien92](#))
- Le plus intéressant à mon sens est le rapport écrit ([Lien93](#)) dans lequel on trouve les principes physiques et mathématiques permettant une implémentation.

2. Introduction

Avant de devenir une science à part entière, la mécanique a longtemps été une section des mathématiques. Elle en était un domaine applicatif totalement naturel.

La mécanique statique a été le premier domaine étudié avec des notions fondamentales comme l'équilibre ou les forces. Ensuite, ces notions ont laissé place à la dynamique, basée sur les phénomènes qui régissent les mouvements des solides. Deux scientifiques ont apporté une immense contribution à ce domaine : Galilée, avec sa théorie sur la chute des corps, et Newton, avec ses célèbres lois du mouvement.

La puissance des processeurs actuels permet la réalisation de ces calculs en temps réels. Ainsi il devient possible de modéliser et de visualiser le comportement des solides. Dès lors on s'aperçoit que les lois, établies depuis le XVIème siècle, fournissent des résultats très satisfaisants. Elles sont d'ailleurs communément utilisées dans les jeux vidéo.

Quel est le lien entre les jeux vidéos et Newton nous direz-vous ? Suite à l'essor des jeux vidéo en tant que loisirs de masse, les développeurs sont à la recherche d'univers toujours plus immersifs. Pour cela, ils souhaitent modéliser le plus fidèlement possible les phénomènes naturels. Des efforts importants ont été faits ces dernières années dans deux domaines de recherche : la qualité graphique et la dynamique.

Un nouveau type d'application est alors apparu : les moteurs physiques. Un moteur physique est une bibliothèque intégrable à un programme qui permet la résolution des équations physiques. Typiquement, il permet de détecter les collisions entre plusieurs solides et de calculer leurs réponses.

Ce rapport a pour but de présenter un projet qui pourrait constituer un début de moteur physique. Reposant sur des formes géométriques de base, nous avons souhaité implémenter les lois de la dynamique.

2.1. Algorithme général

Les objets de la scène sont donc décrits dans un fichier texte qui va être lu par le programme. Ce fichier va notamment initialiser les positions des objets. Leur position initiale est ainsi fixée dans ce fichier. Ensuite, à chaque frame, le moteur va réaliser ce même

processus :

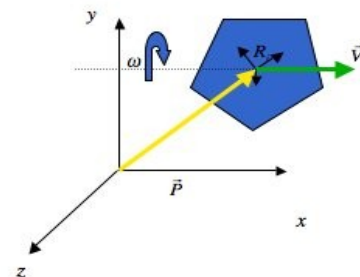
- Calculs des nouvelles positions grâce à une intégration dans le temps des équations de la dynamique sans tenir compte des collisions.
- Détection des collisions grâce à des algorithmes spécifiques à chaque type de collisions.
- Gestion éventuelle de ces collisions pour relancer le système dans un état valide : calcul des vitesses réfléchies.
- Mise à jour de l'état des solides.



3. Leçon 1 : Calculs des nouvelles positions

Pour simuler le mouvement des objets nous avons basé notre implémentation sur la mécanique des solides indéformables. Pour décrire le mouvement d'un objet cinq paramètres indispensables doivent être définis :

- Sa position, P .
- Sa masse, M .
- Son repère locale, R_i .
- Sa vitesse linéaire (vitesse du centre de masse), \vec{V} .
- Sa vitesse angulaire, ω .



Représentation d'un solide

Cependant il est à noter que dans l'implémentation de notre moteur nous n'avons pas tenu compte de la vitesse angulaire des sphères. En effet, nous avons décidé de mettre de côté cet aspect puisque si l'on associe une texture uniforme à une sphère, sa rotation n'est pas perceptible. En outre, ceci peut être considéré comme amélioration future.

3.1. Équations de la dynamique

Nous utilisons dans ce projet les deux équations fondamentales suivantes :

$$m\vec{a} = \sum \vec{F}_i$$
$$I \frac{d\omega}{dt} = \sum M_i$$

m : Masse de l'objet. \vec{F}_i : Force i appliquée sur l'objet.
 \vec{a} : Accélération de l'objet.
 I : Matrice d'inertie de l'objet M_i : Moment i appliqué sur l'objet

A chaque frame, nous réalisons un bilan des forces exercées sur l'objet pour calculer la nouvelle accélération ainsi que le bilan des moments pour en déduire la nouvelle vitesse angulaire.

La matrice d'inertie permet de traduire la répartition de la masse du solide et ainsi de favoriser certains couples. Elle joue grossièrement le même rôle pour la vitesse angulaire que la masse pour la vitesse linéaire. La matrice d'inertie d'un solide a la forme

suivante :

$$I = \iiint \rho(x,y,z) \begin{bmatrix} (y^2 + z^2) & xy & xz \\ xy & (x^2 + z^2) & yz \\ xz & yz & (x^2 + y^2) \end{bmatrix} dx dy dz$$

où $\rho(x,y,z)$ est la densité volumique du solide.

Matrice d'inertie des solides utilisés :

$$\begin{bmatrix} 2mR^2/5 & 0 & 0 \\ 0 & 2mR^2/5 & 0 \\ 0 & 0 & 2mR^2/5 \end{bmatrix} \quad \begin{bmatrix} m \frac{b^2 + c^2}{12} & 0 & 0 \\ 0 & m \frac{a^2 + c^2}{12} & 0 \\ 0 & 0 & m \frac{a^2 + b^2}{12} \end{bmatrix}$$

Pour une sphère / Pour un cube

3.2. Équation de la cinématique

La cinématique du solide peut être représentée mathématiquement comme l'intégration sur le temps des équations différentielles liant position, vitesse et accélération.

$$\vec{V} = \frac{d\vec{P}}{dt} \quad \vec{a} = \frac{d\vec{V}}{dt}$$

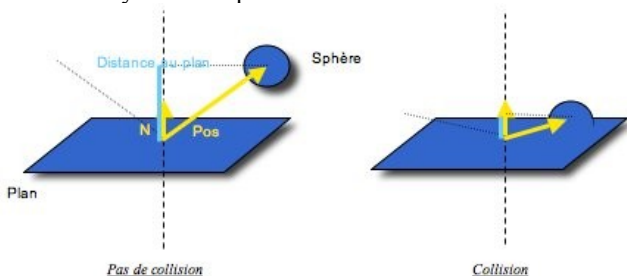
Il est également possible de définir la vitesse d'un point quelconque, P_1 , du solide : $\vec{V}_{P_1} = \vec{V}_G + \vec{GP}_1 \wedge \vec{\omega}$ où \vec{GP}_1 est le vecteur liant P_1 au centre de masse.

4. Leçon 2 : Gestion d'une sphère

4.1. Collision Sphère / Plan

4.1.1. Détection de la collision

Une fois le calcul des nouvelles positions des objets effectué, il est nécessaire de détecter si ils rentrent en collision avec un autre objet. Dans le cas de la collision Sphère / Plan, cette détection se fait aisément en comparant la distance du centre de la sphère au plan avec le rayon de la sphère :



La condition de collision est $|\vec{P}\vec{N}| \leq R$, avec R rayon de la sphère.

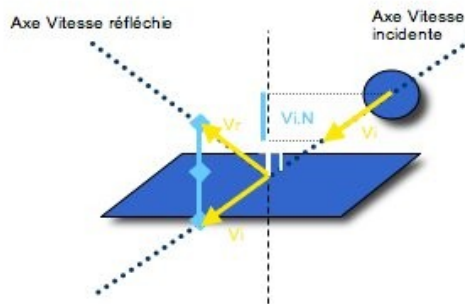
4.1.2. Gestion de la collision

Une première idée pour gérer la collision entre deux corps rigides est d'appliquer une force de réaction aux objets. Cependant cette force ne va pas arrêter les objets immédiatement et ne va donc pas empêcher leur interpénétration car elle ne pourra pas changer instantanément le sens des vitesses.

Une force va mettre trop de temps à inverser les vitesses or les corps sont déjà en contact, il faut donc agir immédiatement sur leur vitesse. C'est pourquoi nous allons calculer la vitesse réfléchiée par rapport à la normale de la collision (la normale du plan dans ce cas). Celle-ci remplacera ensuite l'ancienne vitesse.

La vitesse réfléchiée peut être obtenue grâce à la relation suivante:

$$\vec{V}_r = \vec{V}_i - 2 \times (\vec{V}_i \cdot \vec{N}) \times \vec{N}$$

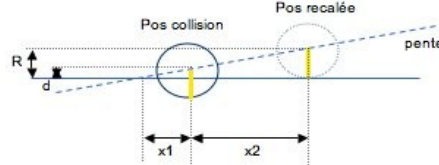


Construction du vecteur vitesse réfléchi

4.1.3. Recalage au point de contact exact

Du fait que le calcul des positions ne peut se réaliser de manière continue (il s'effectue à chaque frame donc tous les Dt secondes), les objets seront déjà interpénétrés au moment où l'on détecte la collision. Il est alors nécessaire de replacer les objets au point de contact exact pour être certain qu'à la frame suivante, les deux objets se seront bien séparés.

En effet si l'on ne réalise pas ce recalage, il est possible que les objets n'aient pas eu le temps de compenser l'interpénétration à la frame suivante (notamment si le Dt suivant est faible). Dès lors, le moteur va détecter de nouveau une collision, etc.



Recalage de la sphère sur le plan

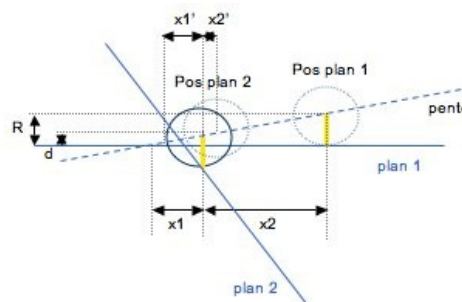
On recale l'objet dans la direction de sa vitesse au moment où l'on détecte la collision. On cherche en fait à « remonter dans le temps » afin de trouver le point de collision exact. On calcule dans un premier temps la pente de la vitesse (représentée par des pointillés bleus sur le schéma ci-dessus).

On a alors $x_1 = d / Pente$ or $x_1 + x_2 = \frac{d}{R}$ d'où $\frac{d}{Pente} + x_2 = \frac{d}{R}$ et donc finalement :

$$x_2 = \frac{R - d}{Pente}$$

Connaissant x_2 on peut alors aisément replacer la sphère à la bonne position.

Dans le cas où la sphère entre en collision avec plusieurs objets (par exemple avec deux plans), on calcule pour chacune des collisions la distance x_2 nécessaire au recalage de la sphère. Finalement, on garde le plus grand x_2 pour recalculer physiquement la sphère et ainsi éviter toute interpénétration avec les autres objets :



Recalage de la sphère lors d'une collision avec 2 plans

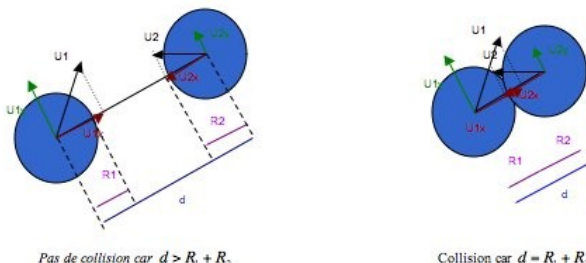
4.2. Collision Sphère / Sphère

4.2.1. Détection de la collision

La détection de la collision Sphère / Sphère est relativement simple. Soient deux sphères :

- S_1 de centre C_1 et de rayon R_1
- S_2 de centre C_2 et de rayon R_2

On forme le vecteur $\overline{C_1C_2}$ et l'on pose $d = |\overline{C_1C_2}|$. Ensuite, il suffit de vérifier que $d \leq R_1 + R_2$ pour s'assurer de la collision.

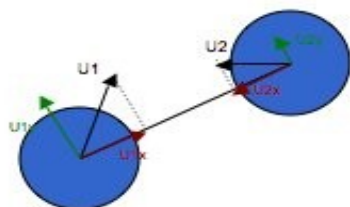


Pas de collision car $d > R_1 + R_2$

Collision car $d = R_1 + R_2$

4.2.2. Gestion de la collision

Notre gestion de la collision va prendre en compte l'inertie des deux objets. En effet, une boule de bowling lancée à grande vitesse ne sera pas influencée par le contact avec une balle de ping-pong.



Gestion de la collision Sphère / Sphère

Tout d'abord, on se place dans le repère local défini par l'axe de direction $\overline{C_1C_2}$ et l'axe perpendiculaire. Seule la composante des vitesses suivant l'axe $\overline{C_1C_2}$ sera modifiée. On donnera ici la solution sans démonstration. On pose donc :

$$\begin{cases} V_{1x} = \frac{(U_{1x} \times M_1) + (U_{2x} \times M_2) - (U_{1x} - U_{2x}) \times M_2}{M_1 + M_2} \\ V_{1y} = U_{1y} \end{cases} \text{ et l'on a } \overline{V_1} = \overline{V_{1x}} + \overline{V_{1y}}$$

De même :

$$\begin{cases} V_{2x} = \frac{(U_{1x} \times M_1) + (U_{2x} \times M_2) - (U_{2x} - U_{1x}) \times M_1}{M_1 + M_2} \\ V_{2y} = U_{2y} \end{cases} \text{ et l'on a } \overline{V_2} = \overline{V_{2x}} + \overline{V_{2y}}$$

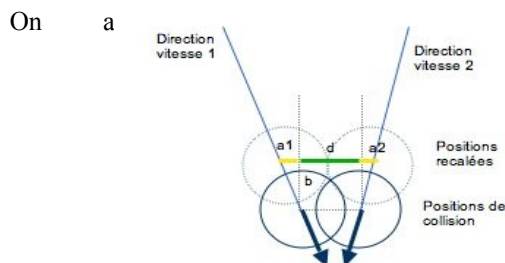
On remarquera dans ces formules que si les sphères sont de masse

égale et si les vitesses au point de rencontre sont les mêmes alors on a $\overline{V_{1x}} = -2\overline{U_{1x}}$. On retrouve alors le calcul classique d'un vecteur réfléchi par rapport à une normale (suivant $\overline{C_2C_1}$) soit $\overline{V_r} = \overline{V_i} - 2 \times (\overline{V_i} \cdot \overline{N}) \times \overline{N}$.

4.2.3. Recalage au point de contact

Pour les mêmes raisons que précédemment (voir recalage Sphère / Plan), on va chercher à recalcer les deux sphères.

Dans le cas de cette collision, on va reculer les deux sphères suivant la direction de leur vitesse respective jusqu'au moment où elles ne possèdent plus qu'un seul point d'intersection : leur point de contact.



Calcul du recalage Sphère / Sphère

$\begin{cases} b/a_1 = p_1 \\ b/a_2 = p_2 \end{cases}$ avec p_1 (resp. p_2) pente de la vitesse de l'objet 1 (resp. 2). De plus il faut que $a_1 + a_2 + d = R_1 + R_2$ pour être au point exact de contact. Donc :

$$\begin{cases} a_1 \times p_1 = a_2 \times p_2 \\ a_1 + a_2 + d = R_1 + R_2 \end{cases}$$

On a donc un système à deux équations et deux inconnus que l'on peut résoudre aisément par les méthodes classiques. On arrive alors à :

$$\begin{aligned} a_1 &= \frac{p_2 \times (R_1 + R_2 - d)}{p_1 + p_2} \\ a_2 &= \frac{p_1 \times (R_1 + R_2 - d)}{p_1 + p_2} \\ b &= \frac{p_1 p_2 \times (R_1 + R_2 - d)}{p_1 + p_2} \end{aligned}$$

Il faudra donc reculer la sphère 1 de $\sqrt{a_1^2 + b^2}$ et la sphère 2 de $\sqrt{a_2^2 + b^2}$.

Suite à la présentation orale du projet, une question nous a permis de voir une limite dans cette méthode. En effet, la variable « b », posée ainsi, suppose que les vitesses sont coplanaires ; ce qui n'est pas forcément vrai.

Il faudrait donc trouver la démonstration dans le cas général, en distinguant les variables b_1 et b_2 .

Retrouvez la suite de l'article de Gregory Corgie en ligne : [Lien94](#)

Architecture multi-cœurs dans les moteurs 3D

Ceci est une traduction d'un article en espagnol de Pablo Zurita intitulé "Arquitectura Multihilos Para Motores 3D", disponible à cette adresse : <http://www.pablo-zurita.com.ar/spanish/2007/07/26/arquitectura-multihilos-para-motores-3d/> ([Lien95](#)).

1. Abstract

Les processeurs de plusieurs cœurs sont maintenant très répandus et ne sont plus l'apanage exclusif des serveurs et des super-calculateurs. La plupart des ordinateurs personnels et consoles de jeux vidéo utilisent maintenant des processeurs multi-cœurs qui

permettent d'exécuter de manière parallèle plusieurs flots de données. D'autre part, la vitesse de chaque cœur s'est stabilisée et il est difficile d'améliorer la performance d'une application qui s'exécute sur un seul cœur simplement en multipliant le nombre de ces cœurs. Créer une architecture pour des applications 3D qui utilisent un tel hardware est un nouveau défi car il nécessite

l'interaction entre plusieurs sous-systèmes. Dans cet article, on expliquera l'architecture du Chromaticity Engine (NB : le moteur 3D de l'auteur original de cet article - Pablo Zurita -), un moteur 3D qui fut pensé pour les applications interactives. L'objectif final de cet article est de montrer les différents inconvénients de créer un moteur 3D optimisé pour les applications interactives, et comment nous avons résolu ces problèmes dans le Chromaticity Engine.

2. Introduction

La complexité des moteurs 3D et des applications les utilisant rend très difficile le portage d'un simple cœur à plusieurs cœurs. Au niveau de l'architecture, les différents sous-systèmes d'un moteur sont souvent dépendants entre eux, et il y a donc deux manières de voir les choses. La première est d'utiliser la même architecture que celle utilisée pour un cœur, mais de créer plusieurs threads pour certaines fonctions. L'avantage de cette méthode est que la majorité du code n'est pas modifiée, et que des APIs comme OpenMP permettent de simplifier grandement ce processus. Le problème de cette solution réside dans le fait que deux modèles fondamentalement différents sont mélangés : le modèle utilisé pour les processeurs simple cœur, et le modèle utilisé pour les processeurs multi-cœurs. Il est donc quasiment impossible de tirer parti au maximum de ces nouveaux processeurs. L'autre modèle consiste à laisser chaque sous-système se mettre à jour de manière indépendante, tout en maintenant une certaine cohérence entre ces sous-systèmes. Ce modèle permet d'utiliser de manière plus efficace les ressources de ces nouveaux processeurs multi-cœurs grâce à une architecture adéquate. C'est ce modèle qui est utilisé dans le Chromaticity Engine et qui sera expliqué.

Il faut noter que les architectures multi-cœurs varient grandement d'une plate-forme à l'autre, et qu'il est donc nécessaire que cette architecture soit suffisamment flexible pour s'adapter à ces différentes plates-formes. Nous allons observer trois types de processeurs principaux :

- Le Cell Broadband Engine de Sony, Toshiba et IBM, utilisé notamment dans la console PlayStation 3.
- Le Xenon d'IBM, utilisé dans la console Xbox 360 de Microsoft.
- Le Core 2 d'Intel, utilisé dans les PC ainsi que dans la dernière série de Macs d'Apple.

Compte tenu de ces différences, nous allons décrire plus en détail l'architecture de ces processeurs.

3. Processeurs

Les différences entre les processeurs que nous allons évaluer sont majeures. La première différence se note dans la quantité de cœurs de chaque processeur et la capacité de calcul de chacun de ces cœurs.

Dans le cas du Core 2 d'Intel, nous avons un processeur de deux ou quatre cœurs, un cache L1 de 64 KB pour chaque cœur, un cache L2 de 4 MB partagé, est out of order ([Lien96](#)) et un thread hardware pour chaque cœur. Ce processeur est très facile à programmer puisqu'il partage de nombreuses caractéristiques avec les processeurs des générations précédentes. Le plus important est que le cache L2 est très grand. Une première idée consisterait à assigner un cœur à un ou plusieurs sous-systèmes du moteur jusqu'à utiliser tous les cœurs disponibles.

Le Xenon d'IBM est un processeur à trois cœurs, avec un cache L1 de 64KB pour chaque cœur, un cache L2 de 1 MB partagé entre les trois cœurs et le GPU, une exécution in order ([Lien96](#)), et deux threads symétriques en hardware pour chaque cœur. Il est

important ici de noter que bien qu'il paraît très différent du Core 2, nous allons voir que les différences sont bien moindres par rapport au Cell. Dans tous les cas, certaines différences vont devoir être prises en compte dans l'architecture du moteur. Tout d'abord, l'exécution dans l'ordre des instructions va entraîner une perte de performance si nous n'organisons pas bien les instructions à exécuter, notamment si un thread effectue une instruction complexe. De plus, le cache L2 est bien plus petit que celui d'un Core 2, mais il reste suffisamment grand pour ne pas trop modifier notre architecture générale. Nous pouvons donc faire les mêmes remarques que pour le Core 2 : une première idée consisterait à assigner chaque cœur à un ou plusieurs sous-systèmes même si, dans ce cas, il faut faire davantage attention à la façon de gérer les informations et l'ordre d'exécution.

Enfin, le Cell Broadband Engine de Sony, Toshiba et IBM est un processeur avec un "Power Processor Element" (PPE), chargé de contrôler six ou huit "Synergistic Processing Elements" (SPE). Le PPE a un cache L1 de 64KB, un cache L2 de 512KB et deux threads gérés en hardware. Ce cœur a comme fonction principale de contrôler les SPEs et de réaliser des opérations qui ne se subdivisent pas bien dans les SPE. Chaque SPE a 256KB de mémoire sans cache, il a un modèle d'exécution in order ([Lien96](#)) et un seul thread en hardware. C'est le processeur le plus compliqué à programmer. Les SPE ont un cache très petit ce qui implique une séparation de chaque travaux afin de maintenir chaque SPE occupé. Il est donc impossible d'assigner un SPE à un sous-système car la combinaison d'un cache très petit et d'une exécution dans l'ordre créeraient des goulots d'étranglement. Un soin tout particulier devra donc être apporté au planificateur de tâches du moteur afin que chaque SPE soit occupé, tout en ne bloquant pas sur une tâche en particulier.

4. Parallélisme

Maintenant que nous en savons un peu plus sur l'architecture de ces processeurs multi-cœurs, nous allons définir un modèle de parallélisme afin que chaque plate-forme soit supportée, en évitant de créer une architecture trop compliquée à maintenir. On ignorera donc la possibilité de paralléliser chaque fonction puisque le gain de performance est relativement minime. D'autre part, nous devons prendre en compte l'interaction entre chaque sous-système d'un moteur 3D (Figure 1). Il faut donc trouver une solution qui permette l'interaction entre chacun de ces sous-systèmes, sans que cela implique une complexité excessive qui ferait perdre tout intérêt à une parallélisation du code.

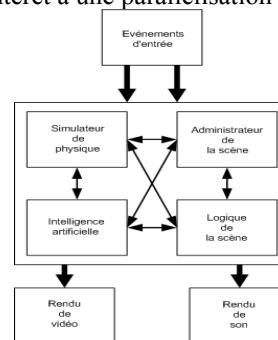


Figure 1 : Exemple d'interaction entre différents sous-systèmes d'un moteur 3D. Comme nous pouvons le voir, il y a des dépendances entre chaque sous-système jusqu'au moment de dessiner la scène et d'exécuter les sons.

Un modèle de parallélisation consisterait à maintenir une boucle principale similaire à celle utilisée dans une architecture simple cœur, et de, tout simplement, paralléliser les sous-systèmes qui n'interagissent pas avec les autres. Par exemple, si nous avons un sous-système "simulateur de particules", nous pouvons le rendre parallèle au système d'intelligence artificielle et d'exécuter de

manière simultanée dans différents cœurs ces deux tâches (Figure 2). Hélas, ce modèle est peu utile dans le cadre d'un moteur 3D car la quantité de sous-systèmes pouvant être parallélisés sont limités. Pour ce modèle, des processeurs comme le Cell attendraient patiemment des instructions pour être exécutées sur les différents SPEs mais, de manière générale, 80% des SPEs resteraient libres. Ce modèle permet donc peu de liberté sur la manière d'utiliser les différents cœurs.

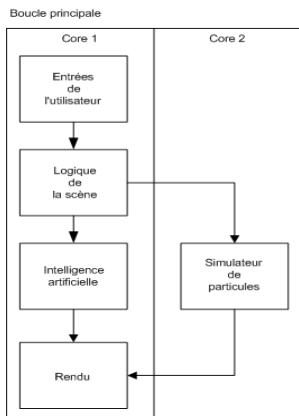


Figure 2 : Exemple de parallélisation de sous-systèmes indépendants entre eux. Dans ce cas, seuls l'intelligence artificielle et le simulateur de particules sont indépendants entre eux et peuvent donc être exécutés de manière parallèle. Mais le reste de la boucle reste dans un seul cœur, qui prend donc la majorité du temps d'exécution.

Le Chromaticity Engine utilise un modèle hybride de parallélisation puisque chaque sous-système se met à jour de manière totalement parallèle afin de travailler constamment avec la dernière information disponible pour chacun des autres sous-systèmes. De plus, chaque sous-système dispose d'un niveau de parallélisation supplémentaire pour les objets indépendants entre eux dans ce sous-système.

Le premier niveau de parallélisation consiste à placer dans chaque cœur un sous-système spécifique (Figure 3). La dépendance entre chacun de ces sous-systèmes continue d'exister, mais dès qu'un sous-système a terminé une tâche, tous les autres sous-systèmes sont mis à jour avec la dernière information de ce premier sous-système. Le grand avantage de cette méthode est qu'il est très scalable sur le nombre de processeur, et que chaque système peut être exécuté sur son propre cœur.

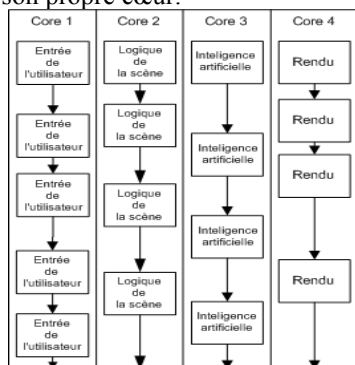


Figure 3 : Exemple du premier niveau de parallélisation dans le Chromaticity Engine. Dans ce cas, nous pouvons voir que chaque sous-système réside dans son propre cœur et travaille de manière indépendante. Au moment d'actualiser chaque sous-système, la dernière information de chaque sous-système est d'abord récupérée puis utilisée.

Le deuxième niveau de parallélisation se déroule au niveau de chaque sous-système. Dans un sous-système, certaines opérations sont indépendantes et il est donc possible de réaliser ces opérations en parallèle (Figure 4). Par exemple, si nous avons un

sous-système "animation" qui va modifier la géométrie des objets dynamiques de la scène, il est possible d'animer deux objets de manière parallèle puisqu'un objet ne dépend pas de l'autre. Ce modèle de parallélisation n'est pas utile dans une architecture comme le Core 2 d'Intel puisque la quantité de cœurs est plus ou moins égale au nombre de sous-systèmes d'un moteur, dans la majorité des cas. Toutefois, dans le cas d'un processeur comme le Cell, ce modèle est très important puisque les SPEs, très nombreux, peuvent contenir peu d'information.

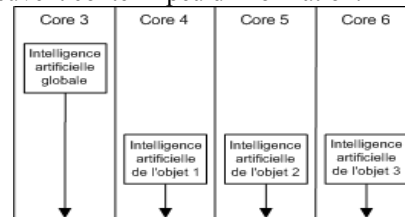


Figure 4 : Exemple du second niveau de parallélisation dans le Chromaticity Engine. Dans ce cas, l'intelligence artificielle globale et l'intelligence artificielle de trois objets (par exemple 3 personnages dans la scène) peuvent être mise à jour de manière parallèle. Les objets ne seront pas forcément mis à jour dans le même cœur.

Une caractéristique que chaque plateforme partage est le fait qu'il soit nécessaire de créer des "points de synchronisations" lorsque l'exécution d'un ou plusieurs sous-systèmes est plus rapide ou plus lente que les autres sous-systèmes. Par exemple, si la mise à jour des sons audio est trop lente par rapport au reste, il y aura au final un décalage entre les sons et l'image. Il est donc nécessaire de créer certains points de synchronisations afin que tous les systèmes restent synchronisés.

5. Scheduler ou planificateur de tâches

Une grosse erreur de design consisterait à laisser chaque sous-système décider lui-même quel cœur utiliser. La raison est que, soit le sous-système ne dispose pas d'information suffisante pour prendre une bonne décision sur quel cœur utiliser, ou alors la logique sur quel thread utiliser dans chaque sous-système, pouvant créer de gros soucis de maintenance et de performance. Pour cette raison, le Chromaticity Engine dispose d'un planificateur de tâches se chargeant de gérer toutes les opérations à exécuter. Ce scheduler va être différent pour chaque type de processeur. Dans le cas du Core 2, on peut obtenir de bonnes performances en laissant un ou plusieurs sous-systèmes dans chaque thread, tandis que dans le cas du Cell, le travail sera différent et plus complexe puisqu'il devra assigner différents SPEs à chaque processus de chaque sous-système. Il faut aussi noter que le scheduler ne peut pas construire et détruire des threads constamment car cela entraînerait une baisse importante de performance. De plus, un thread ne doit jamais être tué mais commettre un "suicide" car tuer un thread est une des opérations les plus coûteuses sur toutes les plates-formes.

Pour le Core 2 d'Intel, le scheduler devra garder les sous-systèmes équilibrés sur les différents threads. Pour ceci, il est nécessaire de récupérer des données sur les différents sous-systèmes afin de modifier les sous-systèmes présents sur chaque thread. Un des problèmes est qu'il n'y a pas de moyen de spécifier de manière définitive sur quel cœur va s'exécuter un thread. Sous Windows, on peut suggérer un cœur à utiliser via la fonction SetThreadAffinityMask, mais elle ne nous assure pas qu'un thread s'exécutera effectivement sur le cœur désiré. Il est donc plus judicieux de laisser le scheduler de Windows décider à notre place sur quel cœur tel thread prendra place.

La politique de planification du Xenon est relativement similaire à celle du Core 2. Toutefois, il est nécessaire de spécifier quel

thread utiliser, sinon tous les threads que nous créons vont être exécutés sur le même thread que le thread que nous venons de créer. Pour ceci on doit utiliser la fonction `XsetThreadProcessor` avec l'argument 0 ou 1 pour le coeur 1, 2 ou 3 pour le coeur 2, et 4 ou 5 pour le coeur 3. A part ça, le scheduler effectuera les mêmes opérations que celui du Core 2.

Enfin, le scheduler du Cell est quant à lui très différent et bien plus complexe. Chaque travail envoyé à chaque SPE devra être plus petits. Le scheduler va donc résider dans le PPE et va s'assurer que le moteur utilise tous les SPEs. Pour ceci, le scheduler devra récupérer en temps réel le temps d'exécution du thread de chaque SPE. Le modèle que nous allons décrire est plus ou moins similaire à [14], où le scheduler possède une file FIFO (First In First Out) de différentes tâches qui seront exécutées sur les SPEs. Il est très important que chaque tâche soit la plus petite possible. En effet, comme nous l'avons vu plus haut, le cache de chaque SPE est très petit et, de plus, les instructions s'exécutant de manière ordonnées, une tâche trop grosse provoquera rapidement un goulot d'étranglement et donc une baisse de performance. Il faut donc garder des tâches les plus petites possibles de manière à ce que le SPE soit de nouveau libre le plus rapidement possible.

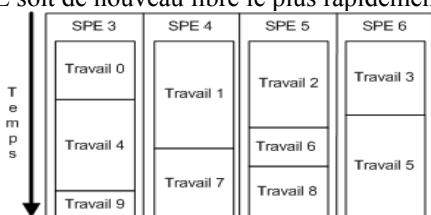


Figure 5 : Exemple du scheduling du Chromaticity Engine. Le PPE se charge de récupérer chaque tâche puis de les envoyer aux différents SPEs.

6. Portabilité et maintenance

Maintenir un moteur 3D pour plusieurs plates-formes est une tâche complexe. D'autant plus complexe si l'on souhaite en plus la portabilité. Les différentes APIs disponibles pour chaque architecture de processeurs sont différentes entre elles. Si nous souhaitons quelque chose de facilement maintenable, il est donc nécessaire de séparer au maximum du reste du moteur le code spécifique à chaque plate-forme. Par exemple, utiliser un système de `#define` pour créer un nouveau thread ne serait pas une bonne idée. Il est nécessaire d'abstraire toutes les opérations spécifiques de telle manière que le scheduler puisse créer un thread de manière transparente sans qu'il sache comment créer ce thread sur la plate-forme utilisée. Quand tout le code est ainsi "caché" de manière modulable, il est bien plus facile de maintenir une seule version du moteur, de le porter à d'autres plates-formes et même pour faire des tests. Il est donc nécessaire de prendre en compte les différentes APIs pour chaque plate-forme. Par exemple, la création de threads se base sur Boost Threads. A la fin, nous n'obtenons donc pas une classe qui supporte tout ce qu'une bibliothèque contiendrait, mais plutôt différentes opérations minimales afin d'obtenir les résultats souhaités sur toutes les plates-formes. Le reste du moteur utilisera donc ces fonctions plutôt que des fonctions spécifiques à une plate-forme.

7. Résultats

Le Chromaticity Engine a été testé sous plusieurs plates-formes. Pour tester ses performances, nous avons créé une scène avec 524 288 objets dynamiques, chacun de 18 triangles. A aucun moment nous n'avons ni dessiné la scène ni lancé des sons afin d'éviter de fausser les résultats à cause du GPU. Le moteur devait, ici, maintenir un graphe de scène avec toute la géométrie de la scène et, en plus, réaliser un octree (subdivision de l'espace de la scène).

Pour la version PC, le moteur fut testé sous Windows Vista avec un Core 2 Duo E6700, Core 2 Quad QX6600 et un AMD Athlon X2 6000+. Le gagnant fut évidemment le Core 2 Quad QX6600 grâce à l'utilisation de ses 4 coeurs (Figure 6). Comme spécifié précédemment, il n'est pas possible de spécifier de manière certaine sur quel coeur sera exécuté tel ou tel thread.

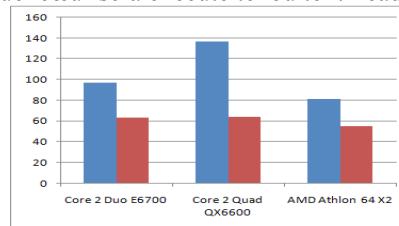


Figure 6 : Performance des différents processeurs en Hertz. En bleu, la performance du moteur tournant sur plusieurs threads, et en rouge le moteur tournant sur un seul thread.

Les gains de performances du moteur comparés au moteur tournant sur un seul coeur avec un seul thread sont donc notables.

Pour la version du Cell, le moteur a été exécuté sur une PlayStation 3 sur Yellow Dog Linux 5.0. Ici, la différence entre utiliser un coeur sur le PPE et le SPE, contre utiliser deux coeurs sur le PPE et les SPEs est très notable (Figure 7).

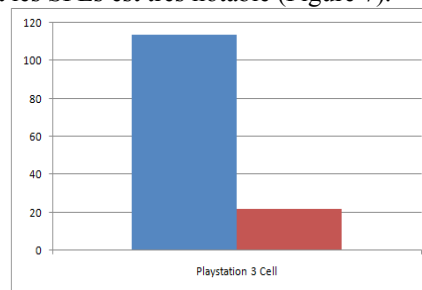


Figure 7 : Performance en Hertz. En bleu la performance du moteur tournant sur le PPE et les SPEs, et en rouge le moteur tournant sur un seul thread sur le PPE et utilisant un seul SPE.

Nous n'avons pu effectuer de mesures pour le Xenon car il nous fallait le SDK de Microsoft, seulement disponible pour les studios de jeux vidéo et les créateurs de middleware.

8. Conclusion

Le Chromaticity Engine est un moteur 3D qui a été écrit en réponse aux changements récents des processeurs. Nous sommes passés d'un seul thread en hardware à, au minimum, un thread en hardware par coeur (au minimum, 2). De plus, les différents coeurs n'augmentent plus de fréquence aussi rapidement qu'auparavant, ce qui implique que nous n'aurons dorénavant moins de gains de performances si nous n'utilisons plus qu'un seul coeur [11]. De ce fait, il est aujourd'hui nécessaire de créer une architecture multithreads qui soit facile à maintenir et à porter sur différentes plates-formes. Grâce aux modèles de parallélisation, il est possible de créer des moteurs 3D compatibles avec les architectures de différents processeurs, et donc d'éviter l'écriture d'un code trop compliqué et difficile à maintenir.

Il sera nécessaire, dans le futur, de valider au mieux l'architecture décrite dans cet article, bien que nous avons déjà pu l'essayer sur une quantité limitée de plates-formes avec succès. Il serait notamment intéressant d'analyser les performances de cette architecture dans le domaine des visualisations scientifiques dans lesquelles les architectures des processeurs sont très différentes.

Retrouvez l'article de Pablo Zurita traduit par Michaël Gallego en ligne : [Lien97](#) et participez au débat sur le forum ([Lien98](#)).

Liens

- Lien1 : <http://ant.apache.org/>
Lien2 : <http://java.developpez.com/articles/ant/partie1/>
Lien3 : <http://java.developpez.com/articles/ant/partie2/>
Lien4 : http://www.jmdoudoux.fr/java/dej/chap059.htm#chap_59
Lien5 : <ftp://ftp-developpez.com/skebir/tutoriels/java/ant/fichiers/sources.zip>
Lien6 : <http://skebir.developpez.com/tutoriels/java/ant/>
Lien7 : <http://java.developpez.com/livres/>
Lien8 : <http://pecl.php.net/>
Lien9 : <http://pecl.php.net/packages/APC>
Lien10 : <http://pecl4win.php.net/>
Lien11 : <http://julien-pauli.developpez.com/tutoriels/php/apc/images/apc-monitor.png>
Lien12 : <http://pecl.php.net/packages/VLD/>
Lien13 : <http://www.php.net/apc>
Lien14 : <http://zend-framework.developpez.com/>
Lien15 : <http://dico.developpez.com/html/3001-Systemes-scalable.php>
Lien16 : <http://www.php.net/apc>
Lien17 : http://www.ioncube.com/sa_encoder.php
Lien18 : <http://www.zend.com/en/products/platform/>
Lien19 : <http://xcache.lighttpd.net/>
Lien20 : <http://www.eaccelerator.net/>
Lien21 : <http://tekrat.com/talks/>
Lien22 : <http://julien-pauli.developpez.com/tutoriels/php/apc/>
Lien23 : <http://cssglobe.developpez.com/tutoriels/css/creer-vignettes-redimensionnables/fichiers/index.html>
Lien24 : <http://cssglobe.developpez.com/tutoriels/css/creer-vignettes-redimensionnables/>
Lien25 : <http://designshack.developpez.com/tutoriels/css/introduction-css3/>
Lien26 : <http://designshack.developpez.com/tutoriels/css/introduction-css3-bordures/fichiers/exemples.htm>
Lien27 : <http://designshack.developpez.com/tutoriels/css/introduction-css3-bordures/>
Lien28 : [http://msdn.microsoft.com/en-us/library/system.windows.data.bindingmode\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/system.windows.data.bindingmode(VS.95).aspx)
Lien29 : [http://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged(VS.95).aspx)
Lien30 : [http://msdn.microsoft.com/en-us/library/system.collections.specialized.inotifycollectionchanged\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/system.collections.specialized.inotifycollectionchanged(VS.95).aspx)
Lien31 : [http://msdn.microsoft.com/en-us/library/ms668604\(VS.95\).aspx](http://msdn.microsoft.com/en-us/library/ms668604(VS.95).aspx)
Lien32 : <http://broux.developpez.com/articles/csharp/databinding-silverlight/>
Lien33 : <http://morpheus.developpez.com/resharper/>
Lien34 : <http://www.jetbrains.com/downloads.html>
Lien35 : http://www.jetbrains.com/resharper/features/code_refactoring.html
Lien36 : <http://www.jetbrains.com/resharper/plugins/index.html>
Lien37 : <http://lgmorand.developpez.com/dotnet/resharper/>
Lien38 : <http://matthieu-brucher.developpez.com/tutoriels/cpp/boost/smartptrs/>
Lien39 : http://arb.developpez.com/c++/raii/shared_ptr/
Lien40 : http://cpp.developpez.com/faq/cpp/index.php?page=pointeurs#POINTEURS_raii
Lien41 : http://cpp.developpez.com/faq/cpp/?page=classes#CLASS_clone
Lien42 : <http://alp.developpez.com/tutoriels/traitspolicies/>
Lien43 : <http://www.artima.com/cppsource/safebool.html>
Lien44 : <http://loic-joly.developpez.com/tutoriels/cpp/smart-pointers/>
Lien45 : http://blog.developpez.com/alp/?title=thinking_in_c_est_desormais_disponible_e
Lien46 : http://blog.developpez.com/alp/?title=qt_se_refait_une_beaute
Lien47 : [http://adserver.adtech.de/?adlink\[3.0\]224\[14296281\]\[16\]AdId%3D1974983%3BBnId%3D3%3Blink%3Dhttps%3A%2F%2Ftrolltech.com%2Fabout%2Fnews%2Fqt-extended-4.4-released](http://adserver.adtech.de/?adlink[3.0]224[14296281][16]AdId%3D1974983%3BBnId%3D3%3Blink%3Dhttps%3A%2F%2Ftrolltech.com%2Fabout%2Fnews%2Fqt-extended-4.4-released)
Lien48 : http://blog.developpez.com/alp/?title=qtopia_devient_qt_extended_pour_la_versi_4
Lien49 : <http://silkyroad.developpez.com/excel/ruban/>
Lien50 : <http://silkyroad.developpez.com/VBA/EvenementsFeuille/>
Lien51 : <http://silkyroad.developpez.com/excel/callbacks/>
Lien52 : <http://cafeine.developpez.com/access/tutoriel/autoextensible/>
Lien53 : <http://argyronet.developpez.com/office/vba/convention/>
Lien54 : <http://argyronet.developpez.com/office/access/postit/>
Lien55 : <http://excel.developpez.com/faq/?page=sommaire>
Lien56 : <http://outlook.developpez.com/faq/?page=sommaire>
Lien57 : <http://sqlpro.developpez.com/cours/quoi-indexer/>
Lien58 : <http://alain-defrance.developpez.com/articles/SGBD/contraintes/>
Lien59 : <http://alain-defrance.developpez.com/articles/SGBD/MySQL/SQLintegre-error/>
Lien60 : <http://rwijk.blogspot.com/>
Lien61 : <http://rwijk.blogspot.com/2007/10/sql-model-clause-tutorial-part-one.html>
Lien62 : <http://rwijk.blogspot.com/2007/10/sql-model-clause-tutorial-part-two.html>
Lien63 : <http://antoun.developpez.com/oracle/model/>
Lien64 : http://mirror.href.com/thestarman/asm/mbr/MBR_in_detail.htm
Lien65 : <http://www.rwc.uc.edu/koehler/comath/42.html>
Lien66 : <http://en.wikipedia.org/wiki/LiveCD>
Lien67 : <http://www.ibm.com/developerworks/linux/library/l-bootload.html>
Lien68 : <http://www.freshmeat.net/projects/lilo/>
Lien69 : <http://www.netadmintools.com/html/mkinitrd.man.html>
Lien70 : <http://debianlinux.net/linux.html>
Lien71 : <http://www.gnu.org/software/grub/>
Lien72 : http://www.coreboot.org/Welcome_to_coreboot
Lien73 : <http://www.openbios.org/>
Lien74 : <http://www.kernel.org/>
Lien75 : <http://come-david.developpez.com/tutoriels/boot-linux/>

Lien76 : http://www.appleinsider.com/articles/08/10/18/inside_the_new_macbooks_audio_and_video.html
Lien77 : <http://www.apple.com/macbook/the-new-macbook/watch.html#large>
Lien78 : <http://www.apple.com/quicktime/qtv/specialevent1008/>
Lien79 : <http://mac.developpez.com/evenements/apple/portable/MacBook2008/>
Lien80 : <http://docs.info.apple.com/article.html?artnum=107623>
Lien81 : <http://www.apple.com/server/macosx/resources/>
Lien82 : http://images.apple.com/server/macosx/docs/File_Services_Admin_v10.5.pdf
Lien83 : <http://support.apple.com/kb/HT2202>
Lien84 : <http://gete-net.developpez.com/tutoriels/mac/server/AFP/permissions/>
Lien85 : <http://sylvain-gamel.developpez.com/tutoriel/mac/automator/decouverte/>
Lien86 : http://blog.developpez.com/index.php?blog=138&title=creer_une_image_iso_d_un_disque_sous_mac
Lien87 : http://blog.developpez.com/index.php?blog=138&title=creer_une_image_iso_d_un_cd_dvd_de_pc_so
Lien88 : <http://www.gete.net/>
Lien89 : <http://mac.developpez.com/livres/>
Lien90 : http://blog.developpez.com/jeux?title=que_s_est_il_passe_a_la_gc_de_leipzig
Lien91 : <http://gregory.corgie.free.fr/dotclear/index.php?2007/02/18/21--physic-engine-la-section-video>
Lien92 : ftp://ftp-developpez.com/gregorycorgie/tutoriels/physic/fichiers/Sources_PhysicEngine.rar
Lien93 : ftp://ftp-developpez.com/gregorycorgie/tutoriels/physic/fichiers/Rapport_PhysicEngine.pdf
Lien94 : <http://gregorycorgie.developpez.com/tutoriels/physic/>
Lien95 : <http://www.pablo-zurita.com.ar/spanish/2007/07/26/arquitectura-multihilos-para-motores-3d/>
Lien96 : http://en.wikipedia.org/wiki/Out-of-order_execution
Lien97 : <http://jeux.developpez.com/tutoriels/multithread/>
Lien98 : <http://www.developpez.net/forums/d609006/technologies-divers/developpement-2d-3d-jeux/apporter-multithread-developpement-jeux/>