Supplément **Spécial C++** **24 pages**

PROGRAMMEZ!

Le magazine du développement

PROgrammez!

www.programmez.com

mensuel n°143 - juillet-août 2011

© ALDEBARAN DR

Programmer votre robot !

La saga de Nao, le premier robot humanoïde !

Votre première application Nao : jouer au Trivial Pursuit

Jeux

Initiez-vous à la 3D avec Direct3D 11.0

© Microsoft DR

CODING 4 FUN

Hacker votre climatisateur

GEEK APPROVED

Développer pour **Kinect**

© inPulse DR

inPulse La montre programmable

F#
Découvrir la programmation fonctionnelle

Multicore
Concurrency runtime et parallélisme avec Visual C++

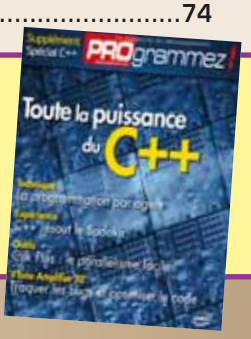
Java
Maîtriser Spring 3

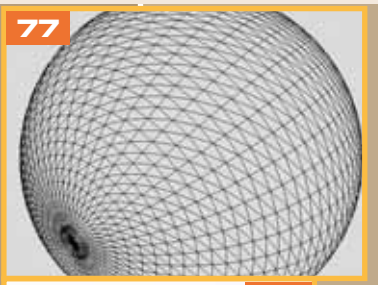
M 04319 - 143 - F: 5,95 € - RD

Printed in France - Imprimé en France - BELGIQUE 6,45 €
SUISSE 12 FS - LUXEMBOURG 6,45 € - DOM Surf 6,90 €
Canada 8,95 $ CAN - TOM 940 XPF - MAROC 50 DH

PROgrammez! Le magazine du développement
numéro 143 - juillet/août 2011

sommaire\\





Smile

OPEN SOURCE SOLUTIONS

Smile recrute 200 personnes en 2011

Développeurs JEE/PHP, chefs de projets, ingénieurs système... et passionné(e)s des CMS, des portails, de l'e-commerce, de la GED, de l'ERP, du Système...

N'hésitez plus, déposez votre candidature sur **www.smile.fr/carrieres**

. Paris . Lyon . Nantes . Bordeaux . Montpellier . Aix . Lille . Barcelone . Kiev . Genève . Casablanca . Amsterdam

éditO


©V.T.

Amusez vos neurones !

Enfin l'été et les vacances. Comme chaque année, nous nous posons la même question : comment amuser vos neurones ? Après quelques nuits blanches de réflexion, nous vous avons concocté deux dossiers spéciaux. Tout d'abord, nous revenons sur l'épopée d'un constructeur français, Aldebaran. Il a inventé le robot Nao qui connaît un beau succès. Et, avec le programme Développeur – bon, ok il est à 3 600 euros, mais pour un passionné, le sacrifice en vaut la chandelle – vous bénéficiez de Nao et de l'ensemble de ses outils de développement. Et, dans ce numéro, laissez-vous guider pour créer votre première application Nao : jouer au Trivial Pursuit… Ce robot possède de prodigieuses capacités !

Notre second dossier est du pur coding4fun ! Imaginez : hacker un climatiseur, programmer une montre en C, utiliser votre Kinect ! Mais nous n'oublions pas la recherche actuelle et de demain, et là, vous allez être surpris par l'imagination des chercheurs et ingénieurs : exosquelette, bioinformatique, intelligence ambiante ! De quoi faire réfléchir sur la place et le rôle de l'informatique dans la société actuelle.

L'été sera chaud sur plusieurs domaines. De nouvelles tablettes arrivent en France : RIM, HP notamment. Avec toujours une même idée : se faire une place sur ce marché en pleine explosion et essayer de briller près d'Apple et de l'incontournable iPad. Les premiers Chromebooks sont disponibles depuis la mi-juin et nous suivrons de près cette nouvelle approche proposée par Google. Courant septembre, nous devrions découvrir de nombreux détails sur le prochain Windows, Windows 8. Mais le mois de juillet ne sera pas en reste avec MacOS X Lion en attendant en automne iOS 5.0 et les services iCloud… De quoi nous occuper…

Plusieurs plaintes en justice seront à suivre de très près. Si Apple et Nokia ont finalement conclu un accord amiable, Apple n'en a pas fini avec Samsung autour de brevets sur les technologies mobiles. Et si nous n'avons pas beaucoup de nouvelles de l'attaque d'Oracle contre Android - Google, l'affaire se poursuit. Oracle demande tout simplement plusieurs milliards, environ 4-5, de dommages et intérêts, et des royalties sur chaque terminal Android, tandis que Google exige l'abandon de la plainte et le règlement, par son adversaire, des frais de justice… Derrière cette affaire se cache une violente guerre des brevets et une course effrénée pour les acheter.

Au moment où vous lirez ces lignes, nous saurons sans doute qui pourra mettre la main sur les 7000 brevets concernant les réseaux et mobiles de Nortel. Le montant de la transaction dépassera sans aucun doute le milliard de dollars. Qui de Google, Intel, Apple ou Ericsson gagnera ? Il y a quelques mois, un lot de 882 brevets détenus par Novell et vendus suite au rachat de l'éditeur, a provoqué de multiples rebondissements : rachat, blocage, rachat par lots, etc. Finalement, Microsoft / Oracle / EMC / Apple pourront acquérir la plupart des brevets mais sous conditions, pour éviter toute action juridique future et éviter de bloquer toute innovation qui pourrait utiliser des brevets rachetés. C'est une véritable guerre des nerfs où chaque géant tente de prendre des positions, d'intimider l'autre. Tandis que l'Open Source regarde souvent en spectateur, en cherchant à éviter que cette guerre puisse paralyser le monde ouvert…

François **Tonic**
Rédacteur en chef

Le magazine du développement
PROgrammez !
www.programmez.com

Editeur : Go-02 sarl, 21 rue de Fécamp 75012 Paris - diff@programmez.com.
Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky.
Rédacteur en Chef : François Tonic - ftonic@programmez.com. **Ont collaboré à ce numéro :** F. Mazué, D. Catuhe. **Experts :** F. Esnault, S. Cordonnier, S. Reverdy, P. Catro-Brouillet, M. Rolland, C. Williams, G. Renard, P. Cauchois, S. Warin, M. Giraud, M. Salson, C. Consel, B. Bertran, A. Crepet, V. Pernin, C. Pichaud.
Illustrations couverture : © Aldebaran, Microsoft, inPulse
Publicité : Régie publicitaire, K-Now sarl. Pour la publicité uniquement : Tél. : 01 41 77 16 03 - diff@programmez.com.
Dépôt légal : à parution - Commission paritaire : 0712K78366 ISSN : 1627-0908. Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles Belgique. **Directeur de la publication :** J-C Vaudecrane

Abonnement : Programmez 22, rue René Boulanger, 75472 Paris Cedex 10
Tél. : 01 55 56 70 55
abonnements.programmez @groupe-gli.com
Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30. **Tarifs** abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € - CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € - Canada : 68,36 € - Tom : 83,65 € Dom : 66,82 € - Autres pays : nous consulter. **PDF :** 30 € (Monde Entier) souscription exclusivement sur www.programmez.com

■ Pour faciliter la transition Android vers Windows Phone 7, les développeurs disposent d'un mapping API entre les deux systèmes : **Windows Phone 7 Interoperability**. Le site officiel fournit les documentations, les correspondances d'API, des vidéos. Windows Phone 7 a besoin de développeurs, plus que jamais : http://windowsphone.interoperabilitybridges.com/

■ **SkySQL** est un jeune projet. Pour l'aider à se développer et à concevoir les nouvelles briques, l'équipe a reçu une subvention de 250 000 euros ainsi qu'un prêt pour soutenir la recherche et développement.

■ L'IDE **Real Studio** voit ses performances améliorées grâce à une refonte du compilateur avec l'arrivée de LLVM. Il s'agit d'un ensemble d'outils et de composants de compilateur et de toolchain. Cette implémentation doit aider le compilateur Real à optimiser le code, à mieux détecter le code mort et à assurer un meilleur support des processeurs ARM.

■ La fondation **Mozilla** travaille sur un nouveau projet : réduire la mémoire utilisée par Firefox. Pour ce faire, le projet MemShrink a été

lancé. Il doit permettre trois bénéfices : rapidité, stabilité et réduction mémoire. C'est encore plus vital dans un navigateur mobile.

■ **Oracle** dévoile une nouvelle version de JDeveloper, la 11g Rel 1. Cette version inclut JSF 2, une mise à niveau du framework maison ADF, et des améliorations sur RESTful, Maven, Hudson, etc. L'outil est toujours gratuit. Site : http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html

NetBeans : la preuve par 7

Disponible depuis fin avril dernier, NetBeans 7 confirme le regain de forme de l'IDE Java d'Oracle. Plus discret qu' Eclipse, il n'en demeure pas moins un excellent outil. La v7 propose de nombreuses nouveautés et améliorations : support des bases de données Oracle et de JDK 7, modification du langage (projet Coin), améliorations diverses sur l'interface, les performances, Git, les éditeurs, les langages. Une mise à jour est disponible depuis fin mai ainsi qu'un plug-in JavaFX 2.0.

Les équipes travaillent d'ores et déjà sur les prochaines évolutions. Deux sont annoncées : 7.0.1 et la 7.1. La 7.0.1 apparaîtra en juillet. Elle bénéficiera d'une complète certification JDK 7. Glassfish 3.1.1 sera intégré. Les autres améliorations concernent les performances, le support des langages et la mise à jour des packages et composants. La 7.1 est la prochaine version majeure, attendue pour fin novembre 2011. Cette version inclura toutes les nouveautés de JavaFX 2 (et surtout sa version finale). Sinon, on peut noter de nouvelles améliorations de performances, sur les éditeurs graphiques.

Site : http://wiki.netbeans.org/NetBeans_701 et http://wiki.netbeans.org/NetBeans_71

Conférence What's Next Paris : du cloud, du cloud et du Java !

L'événement What's Next, organisé par Zenika, s'est tenu les 26 et 27 mai à Paris. Il avait pour ambition de faire un point précis sur le monde Java, Eclipse, tout en apportant une expertise internationale aux développeurs, architectes présents. On pouvait ainsi voir sur scène : Neil Gafter, qui a beaucoup contribué à Java et est aujourd'hui chez Microsoft, Adrian Colyer (co-fondateur de SpringSource), Howard Lewis Ship (créateur de Tapestry). Le cloud computing fut très présent avec cloudfoundry, SQLFabric (base NoSQL), le projet Orion (un IDE dans le navigateur), Java sur Windows Azure, ElasticSearch (moteur de recherche distribué).

L'intervention de Neil Gafter, la seconde journée, fut très suivie. Il revint longuement sur l'évolution de Java et les nouveautés de Java 7, tout en parlant des orientations de Java 8. Puis, Windows Azure entra en scène : comment déployer un projet Java sur Azure ! Nous avons pu rencontrer Neil, quelques minutes après son intervention. Et il s'exprime sans détour : il est

D.R.

ravi qu'Oracle ait pris le contrôle de Java. Pour lui, Sun n'avait pas de vision du langage alors qu'Oracle a le mérite d'établir un calendrier, une vision technique et technologique de Java, avec une vraie gouvernance. Notre expert rappelle aussi qu'il ne s'agit pas juste d'avoir un agenda, une implémentation, il faut une expertise globale, une vision d'ensemble, notamment sur le cloud. D'autre part, Neil évoque les efforts de Microsoft sur l'interopérabilité des langages et le dialogue entre les communautés (PHP, Java, Apache, Drupal…) et l'éditeur. Et finalement, si aujourd'hui, Neil travaille sur C# et la partie compilateur, il rappelle que Java et C# se ressemblent beaucoup et qu'ils évoluent beaucoup…

La conférence fut un beau succès : plus de 650 participants. Une des ambitions était de réunir les groupes utilisateurs Java, même si What's Next se veut avant tout européen et pas uniquement franco-français.

Quelques sites :

http://www.spaceinvade.rs/dev/java/whats-next-zenika-paris-mai-2011/
http://blog.courtine.org/2011/05/26/premiere-journee-a-la-whats-next/
http://www.whatsnextparis.com/

1&1 DUAL HOSTING

LA NOUVELLE RÉFÉRENCE DE L'HÉBERGEMENT

Un savoir-faire, des compétences et une qualité de service qu'aucun autre hébergeur ne vous propose.

- ✓ **Sécurité optimale :** Site hébergé simultanément dans 2 centres de données ultrasécurisés !
- ✓ **Rapidité exceptionnelle :** Connectivité de 210 Gbits/s
- ✓ **Hébergement vert :** Approvisionnement en énergie renouvelable
- ✓ **Innovation permanente :** 1000 développeurs en interne

OFFRE SPÉCIALE ÉTÉ : 1&1 DUAL AVANCÉ

1 AN GRATUIT*

- 3 noms de domaine **INCLUS**
- 250 Go d'espace disque
- Trafic **ILLIMITÉ**
- 500 comptes email
- Statistiques de votre site
- 100 bases de données MySQL de 1 Go
- Applications Click & Build **ILLIMITÉES** (au choix parmi 65 applications à installer en 1 clic)
- PHP5, PHP Dev, Zend Framework, Ruby,SSI, Accès SSH, gestionnaire de version Git
- **NOUVEAU : Redondance géographique !** Disponibilité maximale de votre site grâce à un hébergement simultané à deux endroits différents

… et bien plus encore !

Découvrez tous nos packs d'hébergement à prix réduit ainsi que nos autres offres sur www.1and1.fr

1&1

MEMBRE DE united internet

Appelez le **0970 808 911** (appel non surtaxé) ou consultez notre site Web

www.1and1.fr

* Le pack 1&1 Dual Avancé est gratuit pendant 1 an sous réserve d'un engagement de 24 mois. A l'issue de la première année, ce pack est au prix habituel de 9,99 € HT/mois (11,95 € TTC/mois). Frais de mise en service de 11,95 € TTC. Conditions détaillées sur 1and1.fr. Offre sans engagement de durée minimum également disponible.

Les Rapports DevExpress

PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

XtraReports Suite

Cross-Platform .NET Support

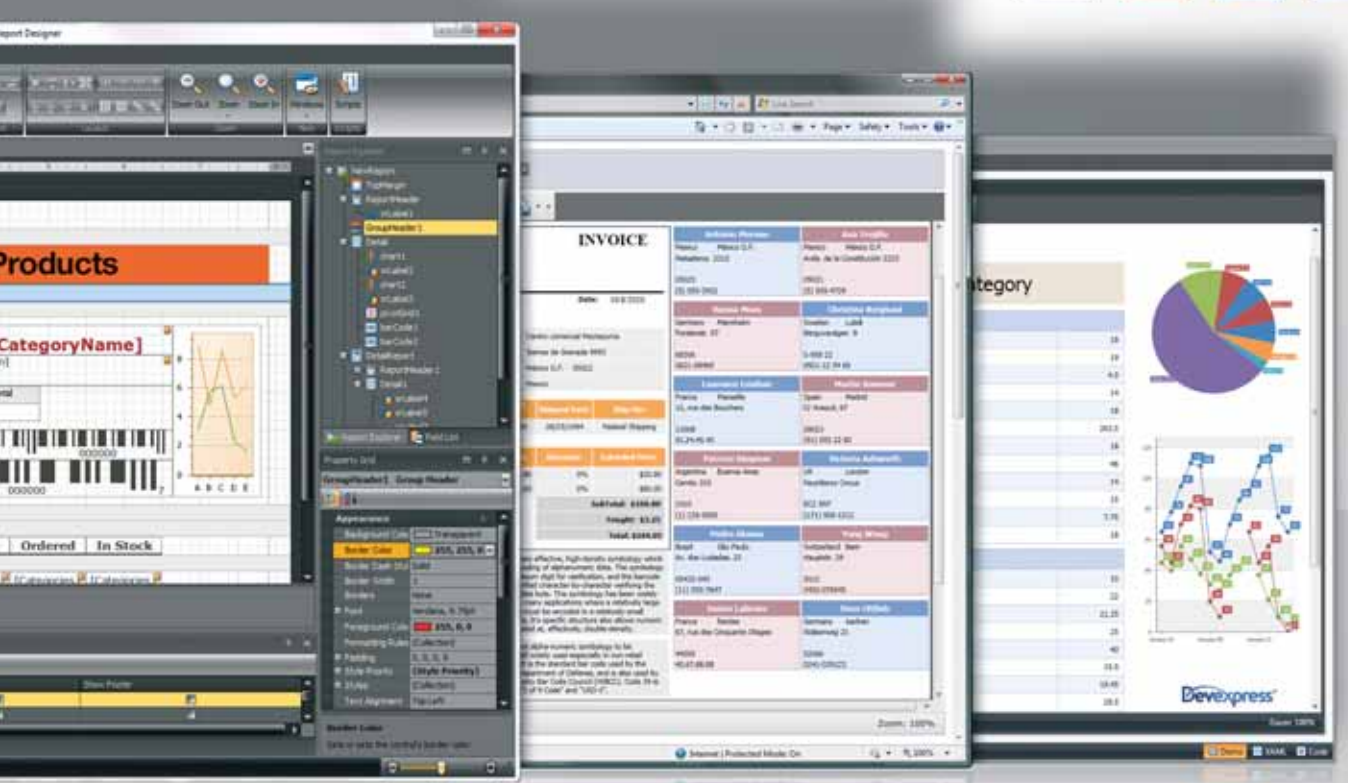
DevExpress crée des contrôles de présentation aux fonctionnalités complètes, des systèmes de rapports, des outils de productivité pour IDE et des frameworks d'applications Business pour Visual Studio .Net. Nos technologies vous aident à construire ce qu'il ya de meilleur, à avoir une vision plus claire des logiciels complexes, à améliorer votre productivité et à créer, dans le temps le plus court, des applications étonnantes pour Windows et pour le Web.

XtraReports Suite, de DevExpress, est une plateforme de Reporting de nouvelle génération pour Visual Studio. Elle vous permet de créer très rapidement des rapports professionnels, au pixel près, ciblant toutes les plateformes majeures Windows, incluant **WinForms**, **ASP.Net**, **Silverlight**, **WPF**. Au-delà de son ensemble de fonctionnalités de niveau professionnel, XtraReports Suite est livré avec un Designer de Rapports ergonomique pour l'utilisateur final, vous permettant de répondre avec la plus grande flexibilité aux exigences de Reporting.

Téléchargez aujourd'hui votre version gratuite d'évaluation et faites l'expérience de la Différence DevExpress.

www.DevExpress.com/Reporting

WIN ASP SL WPF



DevExpress™

WWW.DEVEXPRESS.COM

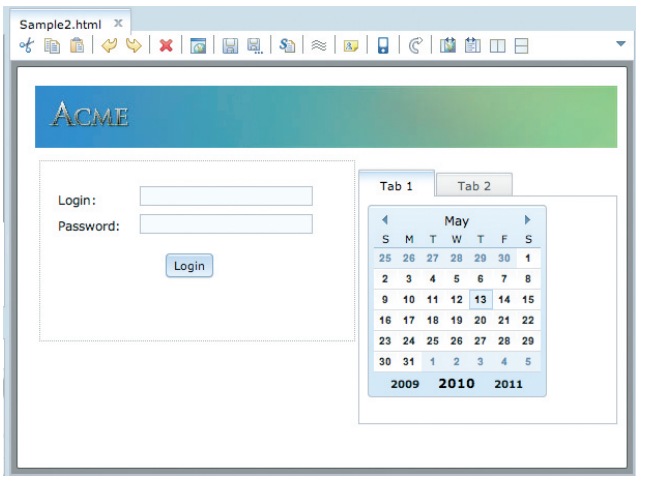
■ **Suivi des procès :** Oracle réclame plusieurs milliards de dollars à Google sur la supposée violation de la licence Java dans Android. Apple met un terme à son procès avec Nokia grâce à un accord amiable (on parle de 400 millions de $ par an).

■ Un nouveau framework Java fait son apparition, **webmotion**. Il repose sur une architecture REST et Java EE 6. Il propose deux fonctions essentielles : assurer la couche présentation par le biais de pages, exposer des services REST (pour les appels XHR, par exemple). A l'instar des frameworks web existants, il permet de sérialiser le résultat de services en JSON ou XML.
Site : http://webmotion.debux.org/

■ **PCC 1.0** est disponible depuis le printemps dernier. PCC signifie portable C compiler. Il se base sur C99. Il doit offrir un compilateur C léger et extrêmement portable, indépendant de tout projet, dont GCC et il peut s'utiliser avec l'ensemble des systèmes BSD.

■ **Oracle** a décidé d'adoucir ses positions envers les communautés après plusieurs mois de fortes tensions. Ainsi, l'éditeur a offert en mai dernier l'outil Hudson à la fondation Eclipse. D'autre part, l'éditeur se déleste de OpenOffice.org, fragilisé depuis la scission et la création de LibreOffice. La suite bureautique sera désormais sous la houlette de la fondation Apache. Comme on pouvait s'en douter, The Document Foundation (développant LibreOffice) n'est pas tout à fait d'accord car elle aurait bien voulu mettre la main sur OpenOffice.org... mais pourquoi Oracle l'aurait-il fait ?

HTML 5 va-t-il bouleverser la situation des outils de développement web ?



On n'entend plus que cela : HTML5 va tout écraser. L'engouement actuel autour de HTML5 est exagéré pour plusieurs raisons. Tout d'abord, les éditeurs, développeurs... travaillent sur des spécifications incomplètes du langage et des balises car le W3C n'a pas encore publié la version finale de HTML5, ni les batteries de tests qui suivront la publication des spécifications. Elles devraient apparaître fin 2011. La v1 de HTML5 ne devrait pas être officielle avant 2012. Et attention, il faudra que les logiciels implémentent la v1 correctement, ce qui prendra de nombreux mois. Quid des implémentations actuelles ? Aucune réponse claire pour le moment. Cela n'empêche pas les éditeurs de multiplier les annonces.

La société française 4D, plus connue pour son SGBD, a annoncé Wakanda, une nouvelle plateforme de développement JavaScript et de déploiement. Il repose sur HTML5, CSS3, etc. Wakanda ambitionne de révolutionner le développement et le déploiement des applications à partir du Cloud sur tous types d'interfaces Desktop ou Touch (tablettes et smartphones). « *Choisir JavaScript comme langage unique pour développer côté client et côté serveur, depuis la manipulation des données jusqu'à l'interface, était il y a trois ans un pari audacieux* », poursuit le PDG, Laurent Ribardière. La plate-forme reprend l'approche 4D et propose de nombreuses interfaces graphiques. Elle comprend ainsi les trois éléments suivants :

- Wakanda Studio est un IDE multi-projets qui gère graphiquement la création des interfaces, l'édition du code source (JavaScript, HTML, etc.), la gestion du modèle objet de l'application, et qui inclut également un remote debugger.
- Wakanda Server intègre un datastore NoSQL, un serveur HTTP multi-thread, et le moteur JavaScript du projet WebKit, l'un des plus rapides du marché.
- Wakanda Framework est automatiquement déployé sur tous les navigateurs. Il comprend un data provider communiquant avec Wakanda Server, des widgets intelligents basés sur Java Script, CSS3 et HTML5.

IBM a lancé le projet Maqetta (http://maqetta.org). Il s'agit d'un projet open source pour créer un éditeur visuel de création de sites et applications HTML5. Il doit supporter le développement multi-cible (mobile, site, desktop), proposer un bon support de Javascript, des plug-ins, divers designers, une architecture asynchrone, le support de webdev, une approche collaborative, etc. Adobe avait l'idée d'un outil purement HTML5, le projet Roma. Mais quelques semaines après son apparition en préversion, Adobe décida de l'arrêter. Cependant, il a conservé un autre projet : Wallaby. Il s'agit d'un convertisseur Flash vers HTML5. Pour Adobe, même si HTML5 n'est pas un concurrent frontal à Flash (pas pour le moment dirons-nous), l'éditeur sait que la menace existe. La question est de savoir comment s'approprier la technologie sans casser le modèle économique, car l'écosystème Flash rapporte de l'argent... Et Adobe doit bouger, car la concurrence, bouge (Microsoft, Apple, Google).
■

Google : les API de la colère

Depuis les annonces de la conférence Google I/O, Google a été obligé de revoir partiellement les plans de suppressions d'API. L'éditeur voulait éteindre 18 API (certaines avec des dates précises, d'autres non) : Blog search, Translate et Transliterate, Video search, Virtual Keyboard, Code Search, etc. Les développeurs et utilisateurs ont tout de même crié au scandale par rapport aux API Translate, particulièrement utilisées pour le service de traduction. Google a rapidement rectifié le tir mais ce revirement aura un inconvénient pour le développeur : il lui faudra payer ! Google sortira une nouvelle version des API mais en version payante, sans indication de date. Dans le même temps, Google a annoncé 7 nouvelles API : Discovery Service, Tasks, Books, Pagespeed Online, Places, Prediction, Fusion Tables. Autre nouvelle qui a suscité une réaction d'étonnement : la disparition des portails de recherche pour Linux, Mac, Microsoft, BSD. Pour Google, il s'agit de se concentrer sur un unique portail.
Pour en savoir plus : http://googlecode.blogspot.com/2011/05/spring-cleaning-for-some-of-our-apis.html#comments

Windows 8 : fin 2012 ?

Microsoft semble vouloir respecter son cycle de développement pour Windows. Les premières démonstrations de Windows 8 ont été faites il y a quelques semaines. L'interface reprendra les concepts introduits par Windows Phone 7. Beaucoup d'interrogations se font jour sur le modèle de développement. Windows 8 donnera-t-il la priorité au modèle Web (HTML5, Javascript, etc.) au détriment du modèle .Net, C++ ? Il est impensable de casser le support .Net et C++ dans Windows, toutes les applications s'appuient dessus. Mi-juin, David Burela, sur son blog, précise que Windows 8 supportera les applications C#, XAML, VB, C++, HTML5. Le nouveau framework Jupiter acceptera l'ensemble des langages même si un focus sera mis sur HTML5, XAML. On devrait commencer à en savoir plus à la rentrée.
Site : http://davidburela.wordpress.com/2011/06/14/premature-cries-of-silverlight-wpf-skill-loss-windows-8-supports-all-programming-models/

Haro sur le mapping objet-relationnel !

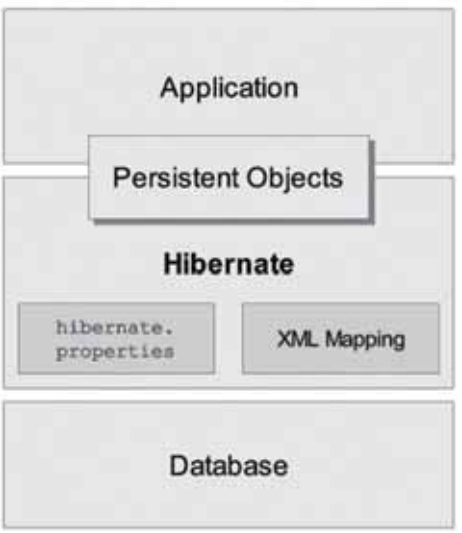
L'accès aux bases de données relationnelles depuis Java a été pendant des années une grande source de discussions pour les architectes, et d'inspiration pour divers outils plus ou moins aboutis. Un beau jour, Hibernate a émergé et calmé cette effervescence. Depuis, c'est LA solution phare pour la persistance. Bien sûr, depuis un moment, il y a aussi JPA (EJB 3) mais ce n'est globalement que la standardisation JEE de ce qu'on trouve dans Hibernate. Et l'on pourrait même se dire : enfin, voici un sujet sur lequel tout le monde est tombé d'accord !

Pourtant, la réalité est plus subtile et nous constatons sur les projets que si le principe de l'usage d'un ORM est moins discuté aujourd'hui, son application ne se fait pas sans douleur. Le problème essentiel, c'est que cette pratique donne bien l'illusion d'avoir une base de données orientée objet alors qu'il n'en est rien. Et la problématique de manipulation de données telle qu'elle a été couverte par les bases de données relationnelles est tellement spécifique qu'à un moment ou à un autre, cette fameuse « illusion » nous revient à la figure…

Un outil à ne pas mettre entre toutes les mains

En effet, qu'il s'agisse de la complexité du mapping d'une base existante ou à l'inverse, de l'inadaptation des schémas générés depuis un modèle objet aux exigences du relationnel, l'ORM c'est un peu le tapis sous lequel on cache la poussière : on croise les doigts pour n'avoir jamais à le soulever… D'après mon expérience, la grande majorité des problèmes de performance des applications de gestion en Java sont justement situés à ce niveau.

Beaucoup de développeurs Java n'ont pas une culture suffisante des bases de données relationnelles et de la manière dont leur outil de persistance va interagir avec elles. Ils ont tendance à croire que ce n'est pas nécessaire puisque les ORM sont justement là pour éviter de s'en préoccuper. Et c'est là que l'erreur se situe souvent. Combien d'applications explosent-elles au moment de la recette, quand ce n'est pas directement à la mise en prod, parce que les développeurs ont consciencieusement décrit leur modèle objet avec toutes les relations possibles et imaginables, ce qui conduit au chargement de la moitié des données à la moindre requête ?



Alors vous me répondrez sans doute que ces problèmes sont souvent liés à une mauvaise utilisation de l'ORM. C'est vrai. Quand on sait faire les bons mappings, quand on écrit des requêtes de façon judicieuse, quand on sait éviter les pièges, ça se passe beaucoup mieux. Mais arriver à un résultat efficace nécessite une bonne connaissance de ces aspects et justement… une bonne maîtrise du fonctionnement des bases sous-jacentes. La question se pose alors : à quoi bon devoir utiliser un ORM s'il faut quand même bien connaître le SGBDR et en plus maîtriser l'ORM sur le bout des doigts ?

De fausses bonnes idées

Pour avoir connu l'époque où il fallait se contenter de JDBC, je ne peux que soutenir un outil qui permet d'éviter tout ce code rébarbatif pour copier mes données de Result Setvers, mes objets, ou pour écrire des requêtes de mise à jour en fonction des changements opérés sur un objet. Par contre, le besoin d'éviter à tout prix le SQL, de vouloir monter les relations de façon « automatique » ne m'est jamais apparu évident. J'ai l'impression qu'il est né d'une position dogmatique de développeurs Java qui ne voulaient pas voir le code pollué par un paradigme étranger. Cela part d'un bon sentiment mais force est de constater que quand on utilise un SGBDR, on ne peut pas faire comme s'il n'existait pas !

L'ORM apporte en outre des mécanismes délicats à maîtriser. Rien que l'option du « lazy loading » me semble une fausse bonne idée : cela présente un confort au départ mais l'exécution de requêtes masquées à des moments mal maîtrisés, alors que la session peut se trouver dans un état inapproprié, se révèle gênante. Elle est de plus incompatible avec les architectures SOA et notamment peu utilisable avec des web services.

Plus généralement, l'ORM considère le modèle objet au détriment de l'approche orientée services qui est pourtant la plus courante pour les applications n-tiers.

Conclusion

Bref, sans vouloir rejeter en bloc les ORM, je pense utile d'insister sur les précautions à prendre quand on les utilise. J'aurais tendance à préconiser un pragmatisme fort visant à exploiter l'ORM simplement comme un outil permettant d'alléger le code tout en évitant les fonctionnalités incompatibles avec les choix d'architecture.

Que les puristes m'excusent mais quitte à être puriste, autant faire de l'objet de bout en bout avec une base de données orientée objet. Et si la base de données relationnelle s'avère incontournable, faisons avec.

■ Frédéric **Esnault,**
Directeur Technique
Open Wide Technologies

Osez la photo 3D avec FinePix W3

Avec le FinePix W3, Fujifilm veut mettre la photo 3D à la portée de tous. L'appareil est relativement compact et possède deux objectifs, pour réaliser deux photos simultanées et créer l'effet 3D. L'écran de visualisation est lui-même 3D (3D sans lunettes), ce qui permet de voir le résultat. L'enregistrement des photos est parfois un peu lent et pénalise la prise de vue rapide. Par contre, pour profiter de vos photos 3D, il vous faudra une télévision 3D ou un cadre photo numérique spécifique. La prise de vue est parfois délicate : profondeur de champ, objet se détachant du reste (un mur n'aura aucun effet). Le tout dans une luminosité suffisante (problème de grains). Attention aussi aux vitres, l'appareil pourrait voir double… Et la vitesse de mise au point et de l'enregistrement pénalisent l'utilisation. Prix de l'appareil : environ 330 euros.

Intel parle de son futur

Lors du grand salon Computex, Intel a dévoilé quelques idées et projets sur le futur de la mobilité avec l'ultrabook. Selon les responsables, il doit associer performances, réactivité, sécurité, design fin et élégant. En réalité, la conférence aura permis de dévoiler les nouvelles gammes de processeurs. Intel Core évoluera avec la génération Ivy Bridge (2e semestre 2012). Un meilleur rendement énergétique est attendu avec des performances graphiques en hausse. Elle devrait accueillir le nouveau procédé de gravure 22 nanomètres. Puis en 2013, suivront les puces haswell, autre étape vers l'ultrabook, car l'ultra est avant tout une nouvelle catégorie de machines. Mais comme il faut aussi s'occuper de l'existant, les puces Atom vont elles aussi évoluer avec Cedar Trail. Son design doit permettre des machines sans ventilateurs…

■ **WD** propose une boîtier spécial de transport pour les disques durs My Passport. Le boîtier durci WD Nomad est constitué d'une coque extérieure en polycarbonate et d'un revêtement absorbeur de chocs en élastomère, à l'intérieur. Il répond formellement au standard militaire MIL-STD-810G pour les équipements durcis. Son loquet à haute résistance et son liner en silicone assurent l'étanchéité nécessaire à la protection contre la salissure, la poussière, et l'humidité. Prix : 23 €.

agenda \\

JUILLET

• Du 04 au 07 juillet, Sophia Antipolis, cycle de conférences gratuit **SophiaConf2011**, au cours duquel vous pourrez apprendre avec les meilleurs experts, et échanger avec vos confrères et consœurs de Sophia Antipolis. www.sophiaconf2011.fr

ETRANGER

• Du 02 au 05 août 2011, USA, Hôtel Hilton San Diego Bayfront, **FileMaker Developer Conference 2011**, 16e conférence annuelle des développeurs FileMaker. http://www.filemaker.fr/pr/devcon2011

• Du1er au 05 octobre, USA, Los Angeles Convention Center, **Adobe Max 2011**. Inscriptions ouvertes sur : http://max.adobe.com/attend/pricing/

■ **Buffalo** lance une nouvelle gamme de NAS professionnelle basée sur Atom Dual Core : TeraStation Pro. Elle peut embarquer jusqu'à 8 baies. Cette gamme propose 4 modèles et un modèle 1U. TeraStation Pro stocke de 2 à 16 To et supporte tous les Raid et le Raid 6. En cas de panne, la fonction 'Fail Over' autorise le basculement automatique d'une unité défaillante vers une unité de secours. De même, la fonction 'Hot swap' permet de remplacer à chaud un disque défectueux sans perte des données ni interruption de service. Enfin, il est possible de répliquer les données sur une seconde unité, locale ou distante. A partir de 476 € HT.

■ **Kingston** dévoile son dernier disque SSD : HyperX SSD utilisant le contrôleur SandForce. SandForce est réputé pour sa fiabilité. Ce nouvel ajout sur le disque SSD doit donc fiabiliser encore plus l'usage d'un disque en production. HyperX SSD intègre de nouveaux mécanismes de performances via le user configurable over provisioning, le support de Trim et SMART. Disponible à partir du 11 juillet.

■ **Oxygen Audio** propose son nouvel autoradio, le MP504. Le MP 504 possède un amplificateur intégré de 4 x 55 Watts pour alimenter les haut-parleurs de l'habitacle. Pour obtenir le meilleur du son en voiture, il est possible d'ajuster le niveau de la sortie subwoofer, calibrer les graves et aigus ainsi que régler la balance et le Fader. Vous profitez de toutes vos musiques et vidéos quelle que soit la source, puisque le MP 504 prend en charge tous les formats audio/vidéo : CD, DVD, carte SD, clé USB, etc. Il dispose bien entendu d'un tuner AM/FM avec support RDS pour écouter la radio tout au long de votre trajet. 30 stations de radio peuvent être mémorisées.

■ **HP** et **RIM** dévoilent leurs tablettes ! Playbook de Rim est disponible depuis quelques jours en France. Elle utilise un système QNX, la vidéo HD, un GPS, un écran 7". Environ 500 euros pour la version de base. HP ne veut pas laisser la concurrence sur ce marché prometteur. Le constructeur lance TouchPad (début juillet). Elle utilise WebOS, fonction NFC. Des concurrents sérieux à l'iPad ?

■ **Belkin** annonce la disponibilité de sa coque translucide *Snap Shield* pour iPad 2. Si les amateurs de tablettes pensent systématiquement à protéger l'écran de leur iPad 2 des rayures et autres chocs, le côté pile de l'appareil est bien souvent négligé. Prix : 19,99 euros.

SharePoint 2010
Rendez vos sites « sexy »

Que ce soit pour mettre en œuvre des espaces collaboratifs, des portails intranet ou des sites internet, SharePoint est aujourd'hui une solution connue du monde entier et adoptée quotidiennement par de plus en plus d'entreprises.

Mais pour tous ces types de sites, un critère important pour que les utilisateurs adhèrent à la mise en place d'un nouvel outil dans l'entreprise concerne l'aspect ergonomique et esthétique de celui-ci, car c'est la première chose à laquelle les utilisateurs seront confrontés lorsqu'ils utiliseront cette plateforme.

Une mauvaise réputation à effacer

De très nombreuses entreprises, lorsqu'elles implémentent SharePoint au sein de leur système d'information et notamment sur les espaces collaboratifs, n'attachent que très peu d'attention à l'aspect esthétique des sites. C'est évidemment une erreur, car même si Microsoft fournit différents thèmes graphiques permettant aux administrateurs des sites de pouvoir changer en quelques clics de souris les couleurs de leurs espaces, cela reste limité et ne change pas fondamentalement l'ergonomie de départ. Le fait que les entreprises n'accordent que peu d'intérêt à cet aspect esthétique, vaut à SharePoint une mauvaise réputation, notamment celles faisant de lui un produit rigide ou difficile à rendre attractif et beau. Cette mauvaise réputation n'est évidemment pas justifiée et peut être gommée de manière assez simple.

Personnaliser un site grâce aux thèmes

Dans sa version 2010, SharePoint a simplifié la conception des thèmes qui sont désormais basés sur des fichiers de type Office Thème (extension THMX), pouvant être conçus depuis PowerPoint. Une fois le thème conçu, celui-ci doit être téléchargé dans la galerie de thèmes de site pour pouvoir être utilisé par l'administrateur du site. Si jamais ce dernier préfère choisir les couleurs directement depuis son navigateur, c'est également possible. Plus besoin de faire appel à des développeurs ou d'avoir besoin de connaissances en création de feuilles de styles CSS pour simplement changer les couleurs d'un site [Fig.1]. Bien qu'intéressant, ce système de thèmes permet uniquement de changer les couleurs de l'interface et les polices. Si l'utilisateur a besoin de mettre une image de fond à son site ou de modifier l'apparence des puces ou pictogrammes, alors les thèmes ne suffisent plus.

Création d'une charte graphique

Dans la plupart des cas, notamment pour les projets de création de sites intranet ou internet, une partie importante du projet concernera l'image de l'entreprise qui sera projetée aux utilisateurs ou aux internautes qui accéderont aux différents sites. Dans ces deux cas, mais qui peuvent également s'appliquer aux espaces collaboratifs, le point de départ pour commencer à travailler sera la charte graphique, créée généralement pour l'occasion par une web-agency qui aura été mandatée spécialement pour cela par l'entreprise [Fig.2]. Comme nous pouvons le constater sur cette création graphique et pour peu que vous soyez un tant soit peu familiarisé avec SharePoint, nous sommes ici très loin du design et de l'ergonomie proposée en standard par Microsoft quand on installe une plateforme SharePoint.

Création d'une page maître et des feuilles de styles

Pour pouvoir appliquer une telle charte graphique à un site SharePoint, la première étape consistera à créer une page maître, qui sera en charge d'appliquer le design général et commun à toutes les pages du site. Cela concernera principalement l'en-tête, la navigation et le pied de page. Plutôt que de partir d'une feuille blanche en ne sachant pas quoi mettre à l'intérieur d'une page maître, le plus simple, et ce principe s'applique d'ailleurs à d'autres aspects dans SharePoint (modèles de sites, feuilles de styles, features ...), partons de la page maître fournie avec la plateforme. Cette page maître se trouve par défaut dans le répertoire *C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\GLOBAL* et se nomme V4.master. Vous constaterez que d'autres pages maîtres sont disponibles dans le même répertoire. Celles-ci sont utilisées par certains modèles de sites spécialisés de SharePoint (ex : sites de réunion). Pour créer notre propre page maître, il suffit de créer une copie de ce fichier sur notre disque dur (où bon vous semble) et de commencer à faire les modifications à l'intérieur. Si nous nous attardons quelques instants sur l'anatomie de cette page maître et si nous la comparons avec la version 2007 de SharePoint, la première chose qui sautera aux yeux des développeurs concernera la construction du code HTML.

```
<div id="s4-ribbonrow" class="s4-pr s4-ribbonrowhidetitle">
  <div id="s4-ribboncont">
      ...
  </div>
  <div id="notificationArea" class="s4-noti">
```

```
  </div>
  <div id="WebPartAdderUpdatePanelContainer">
      ...
</div>
</div>
```

Dans l'extrait ci-dessus, nous pouvons voir que désormais, SharePoint 2010 repose essentiellement sur les principes du XHTML et CSS dans la construction des pages. L'époque où toute la mise en page était basée sur des tableaux imbriqués les uns dans les autres est désormais révolue.

Qu'est-ce que cela signifie pour les développeurs vous demanderez-vous ?

Là où auparavant, le rendu n'était pas garanti sur les navigateurs autres qu'Internet Explorer, désormais les navigateurs tels que Chrome, Firefox ou encore Safari sont désormais pleinement supportés, ce qui fera la joie des intégrateurs HTML maîtrisant parfaitement XHTML et CSS. En effet, ces derniers pourront enfin travailler sur SharePoint de la même manière qu'ils pouvaient le faire sur d'autres plateformes telles que Joomla, ezPublish, Typo3… Cependant, il faudra toujours faire attention aux spécificités de chaque navigateur (même si les choses ont tendance à se réduire de plus en plus), plus besoin d'être formé spécifiquement à SharePoint pour intégrer une charte graphique. Comme nous l'avons vu, la structure HTML repose désormais sur XHTML et sur les feuilles de styles CSS. Si nous reprenons l'exemple précédent, nous voyons que les balises DIV possèdent des identifiants qui permettront ensuite de surcharger la mise en forme de celles-ci.

Les styles par défaut de SharePoint définis, entre autres, dans le fichier nommé CoreV4.css, dont vous pouvez trouver un aperçu ci-dessous, et dans lequel sont définis les niveaux de visibilité ou l'image d'arrière-plan du ruban de SharePoint.

```
body #s4-ribbonrow { display:none; }
body #s4-titlerow { display:block !important; }
body #s4-workspace
{
  overflow:visible !important;
  width:auto !important;
  height:auto !important;
}
body #s4-ribboncont
{
  padding:0px;
  background:url("/_layouts/images/bgximg.png") repeat-x -0px -565px;
}
```

La quasi-totalité de la bonne intégration d'un design, pensé par un graphiste ne connaissant pas nécessairement les contraintes imposées par SharePoint, repose donc sur l'intégrateur HTML et sa bonne connaissance des normes XHTML et CSS. Il est toutefois important que la web-agency qui a créé la charte graphique, ait une connaissance avancée de SharePoint pour éviter de proposer des éléments visuels, qui seront difficilement intégrables en HTML ou nécessitant de lourds développements pour pouvoir être intégrés.

Déploiement et test de la page maître et des feuilles de styles

Maintenant que la page maître et les styles sont réalisés, il est temps de tester que tout fonctionne correctement dans notre site SharePoint. Le déploiement peut être réalisé de 2 manières différentes dans SharePoint :

- Automatiquement via l'utilisation de solutions (fichiers WSP)
- Manuellement via l'interface web

Utilisons cette deuxième possibilité si nous considérons que nous sommes encore en phase de développement et que la création d'une solution est encore prématurée, même si cette solution sera à privilégier pour une mise en production et une industrialisation ultérieure. Pour effectuer un déploiement manuel de notre page maître et des feuilles de styles, il suffit de se rendre dans l'interface d'administration de notre site et d'aller dans la galerie de pages maîtres [Fig.3]. Téléchargez simplement le fichier dans la galerie puis dans l'écran d'administration du site, choisissez d'appliquer cette

Fig.1

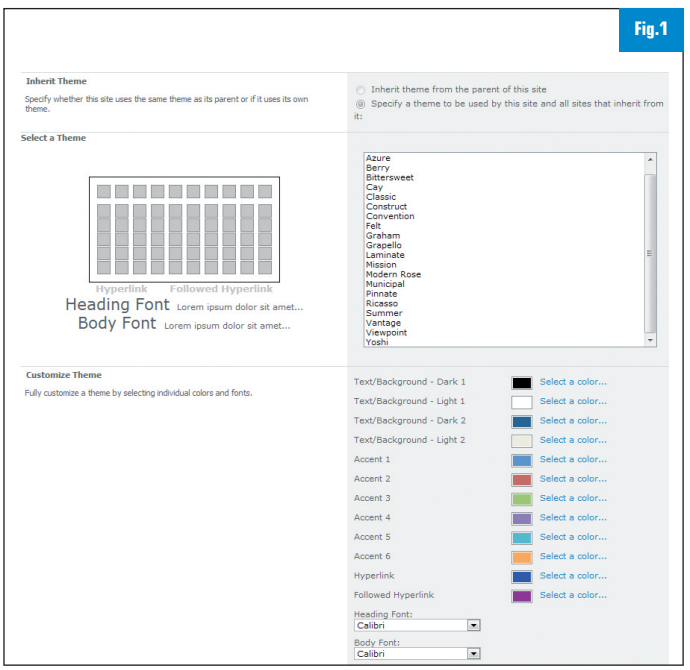


Fig.2

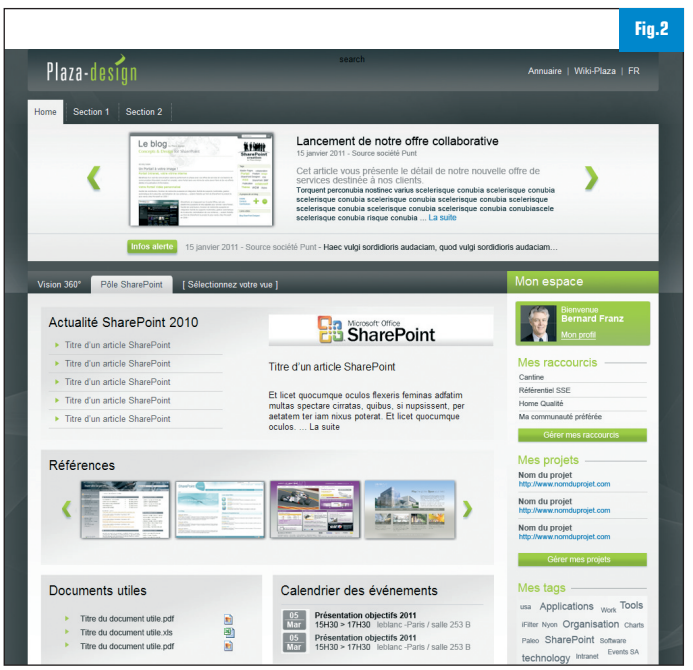
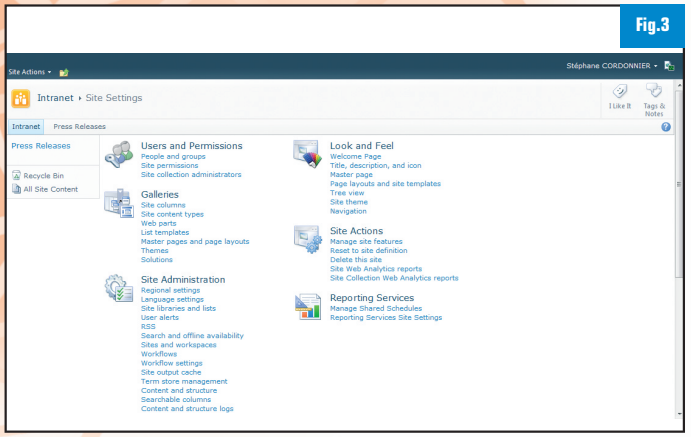


Fig.3



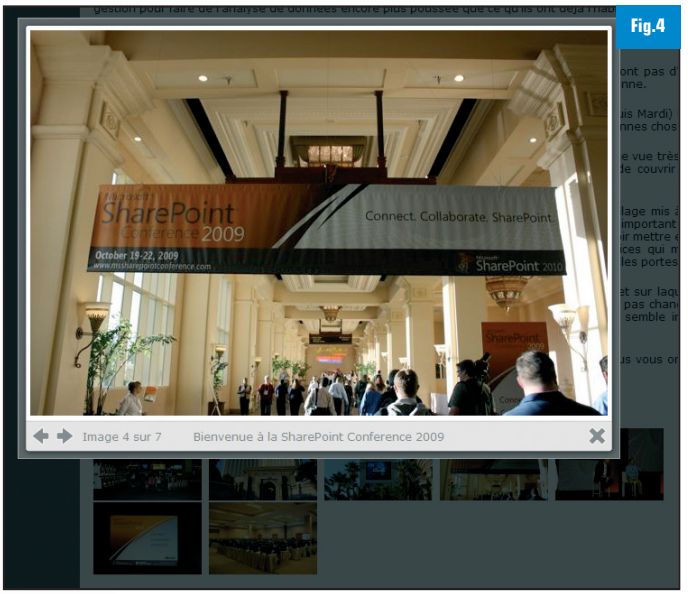
page maître dans la section « Look and Feel ». Les styles quant à eux peuvent être téléchargés dans la bibliothèque de documents nommée « Style Library » et disponible à la racine de votre site SharePoint (veillez à ce que les références dans la page maître pointent vers le bon répertoire).

Aller plus loin avec des composants riches

Appliquer une charte graphique et des styles est certes crucial, mais cela ne rend pas nécessairement un site attractif. En effet, les standards ergonomiques actuels préconisent d'utiliser des composants interactifs (carrousel, diaporama, vidéo…) afin de rendre les sites plus vivants. De très nombreux éditeurs de logiciels ou de sociétés de services possèdent ou proposent de tels composants dans leurs offres. Il ne faut donc pas hésiter à les utiliser. Ils sont généralement basés sur des frameworks tels que jQuery, pour ajouter une touche d'interactivité dans vos sites.

Vous trouverez ci-contre un exemple de diaporama qui s'alimente sur une bibliothèque d'images SharePoint, et tirant parti de jQuery, afin de proposer une mise en forme un peu plus évoluée qu'une simple liste d'images dans une page [Fig.4].

Fig.4



Conclusion

Rendre un site SharePoint sexy est donc quelque chose d'assez simple, pour peu que les critères suivants soient assemblés correctement : une charte graphique tenant compte des contraintes de SharePoint, un intégrateur HTML maîtrisant parfaitement XHTML et CSS et des composants visuels ajoutant une touche d'interactivité dans votre site.

■ Stéphane **Cordonnier**
Directeur Technique – MCNEXT
www.mcnext.com

Les outils pour SharePoint

Un des gros reproches que l'on faisait à SharePoint 2003 et 2007 était le manque d'outils de développement. Il en résultait une certaine frustration et une insuffisance certaine de productivité. Ce défaut, couplé à l'architecture relativement complexe et la lenteur par rapport à asp.net, a fini par faire de SharePoint le « cauchemar » du développeur.

Microsoft a bien sorti Visual Studio Extensions pour WSS, mais cette extension n'a jamais eu beaucoup de succès malgré de nettes améliorations au cours des versions. C'est donc vers des outils tiers que nombre de développeurs se sont tournés : WSPBuilder (wspbuilder.codeplex.com) en est un des meilleurs exemples. Pour SharePoint 2010, Microsoft n'a pas reproduit la même erreur : Visual Studio 2010 intègre un modèle de projet solide pour SharePoint ainsi qu'un certain nombre d'éléments de base : Composant Visual Web Part, Composant Web Part, Flux de travail, Module, Type de contenu…

Visual Studio 2010 : la tour de contrôle

VS2010 c'est aussi l'explorateur de serveurs afin de parcourir la structure et le contenu d'un site SharePoint, la possibilité de paramétrer les options de déploiement. On peut encore aller plus loin en utilisant TFS pour générer ses builds… VS2010 propose 14 éléments. Que faire si ces éléments ne suffisent pas ? Là encore Microsoft a pensé à nous en permettant aux développeurs de réaliser simplement des extensions à Visual Studio. La communauté SharePoint en a ainsi réalisé plusieurs, de qualité.

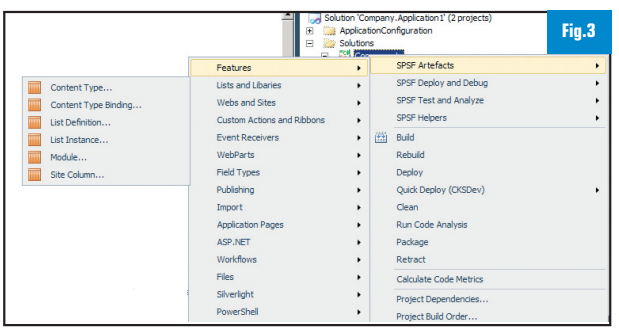
CKS: Development Tools Edition (cksdev.codeplex.com) étend les fonctionnalités SharePoint de VS2010 en ajoutant 14 éléments : Starter Master Page, Branding, Site Definition, Delegate Control… Ces éléments peuvent être rajoutés dans des projets SharePoint VS2010. En prime,

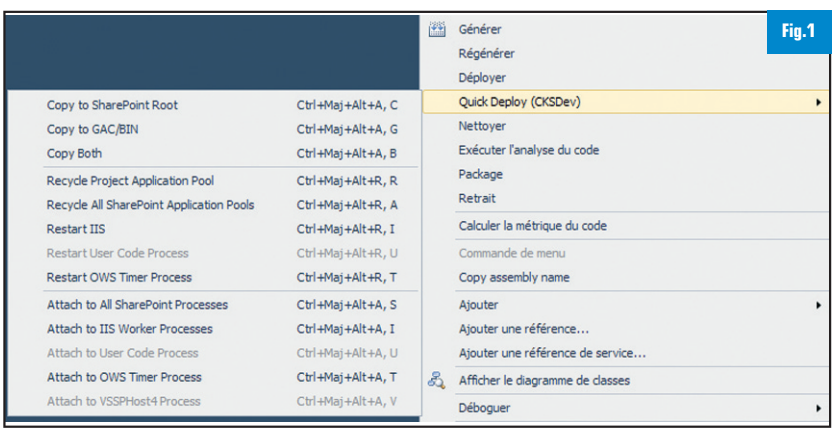
CKSDEV fournit des actions rapides pour déployer les éléments du dossier SharePoint, les DLL mais aussi pour redémarrer ou s'accrocher à certains processus : [Fig.1].
CKSDEV rajoute également des éléments dans l'explorateur de serveur : on peut ainsi explorer depuis Visual Studio 2010 les master pages, les web parts, les styles… On se rapproche ainsi de certaines fonctionnalités de SharePoint Designer. CKSDEV comporte 2 modèles de projet : « SharePoint Console Application » et « Import WSPBuilder Project ». Ce dernier pourra être fort utile lors de migration SP2007 vers SP2010.
SharePoint Software Factory (spsf.codeplex.com) va plus loin que CKSDEV puisque cette extension de Visual Studio (2008 et 2010) est constituée de son propre modèle de solution : un wizard permet de choisir entre 2 modèles : le modèle « VS 2010 » et le modèle « 14 Hive » (classiquement utilisé avec WSPBuilder en SP2007) ainsi que différentes options : utilisation de FxCop, StyleCop. Il faut aussi noter la compatibilité avec SP 2007 [Fig.2].
SPSF a l'avantage du nombre de types d'éléments SharePoint proposés : on compte ainsi plus de 90 templates et exemples d'éléments. SPSF inclut aussi un projet de Build, à base de msbuild. Finalement, comme CKSDEV, SPSF fournit des helpers pour déployer, configurer, débuguer… SPSF est ainsi un des outils les plus aboutis dans cette version de SharePoint [Fig.3].
Tout comme CKSDEV, SPSF est 100% compatible avec Visual Studio sans extension : ainsi, pour ouvrir la solution et compiler/déployer, nul besoin d'extension.
Visual Studio 2010 SharePoint Power Tools (http://visualstudiogallery.msdn.microsoft.com) est une extension officielle Microsoft qui fournit une « Sandboxed Visual WebPart », ainsi qu'un support avancé à la compilation en mode bac à sable. Donc une petite extension bien pratique quand on travaille en « bac à sable ». caml.net.intellisense (camldotnet.codeplex.com) aide à concevoir les éléments en CAML en rajoutant une IntelliSense complète et à jour de tous les nœuds et attributs CAML. Qui ne s'est jamais demandé que mettre à tel ou tel attribut ? Et qui n'a jamais passé 1h sur bing pour trouver la réponse ? Une extension à retenir… [Fig.4].
Un projet en version bêta, mais prometteur : LINQ to SharePoint DSL Extension for Visual Studio 2010 (http://visualstudiogallery.msdn.microsoft.com). Cette extension propose de générer des classes SharePoint à partir d'un schéma de listes SharePoint, un peu comme Entity Framework. L'accès aux données est grandement simplifié !
Fatigué de développer en modèle objet SharePoint JavaScript sans IntelliSense ? Mavention a développé pour vous des snipets qui simplifient l'ajout d'intelliSense dans Visual Studio pour le modèle objet SharePoint JavaScript. D'ailleurs, un grand nombre d'autres outils Mavention sont disponibles sur le blog Waldek Mastykarz, MVP SharePoint (blog.mastykarz.nl/tools). Visual Studio 2010 et ses extensions permettent donc de développer des composants pour SharePoint, de les déployer et de les débugger. Un des inconvénients de Visual Studio 2010 est le manque d'interactivité avec SharePoint : Quel outil utiliser quand on veut modifier le plus rapidement possible le design ? Ou quand on veut tester en direct des mises en pages ?
La réponse est SharePoint Designer 2010 (SPD). Cet outil permet de travailler connecté à SharePoint. C'est à la fois un outil pour les administrateurs, pour les designers, mais aussi pour les développeurs : à l'inverse de VS2010 où l'on travaille sur des éléments que l'on déploie par la suite, avec SPD 2010, on interagit directement avec SP2010. Par exemple, avec VS2010, pour créer et améliorer une Page Maître, il faut créer une feature qui instancie la page maître, et la déployer à l'aide d'une solution SharePoint. A chaque modification de cette page, il faudra redéployer le package. Avec SPD 2010, en quelques clics on peut créer une Page Maître, et la modifier en direct. De plus, SharePoint Designer nous offre un aperçu que ne propose pas Visual Studio.
Autre avantage : contrairement à VS 2010, SPD 2010 n'a pas besoin d'être déployé sur le serveur pour fonctionner : il convient donc aussi très bien à des administrateurs de sites. Tout comme Visual Studio, cet outil, a subi de nombreuses évolutions depuis 2007 : une meilleure fiabilité, de nouvelles actions telles que la création de Workflow réutilisables, de nouvelles interaction avec Visual Studio 2010… En général, les développeurs s'en servent pour :

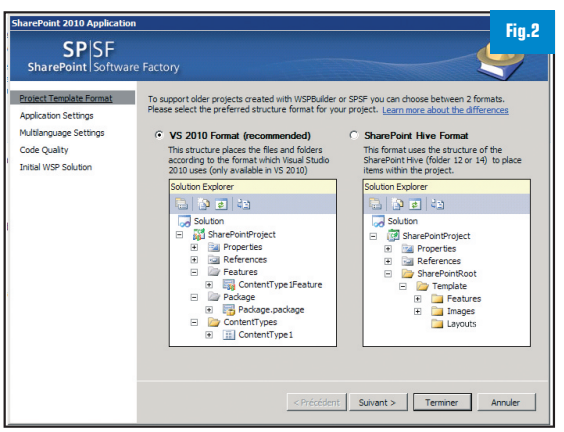
- Créer, tester et améliorer le design ;
- Créer et tester différents types d'éléments (Workflow, formulaires, page maître, css…) ;
- Exporter ces éléments, puis les importer dans Visual Studio.

D'autres outils « connectés » existent

SharePoint Manager 2010 (spm.codeplex.com) est un outil qui permet d'explorer les « entrailles » de SharePoint à l'aide d'une







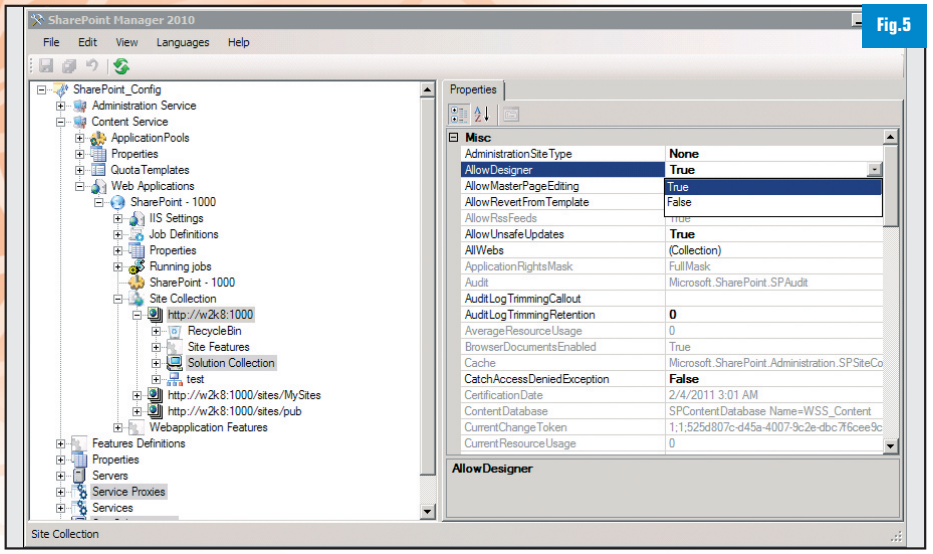


Fig.5

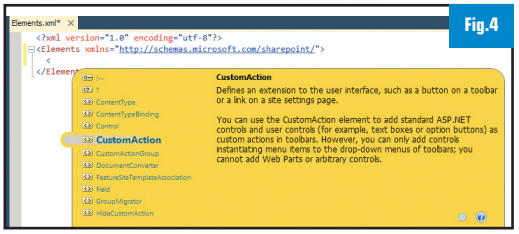


Fig.4

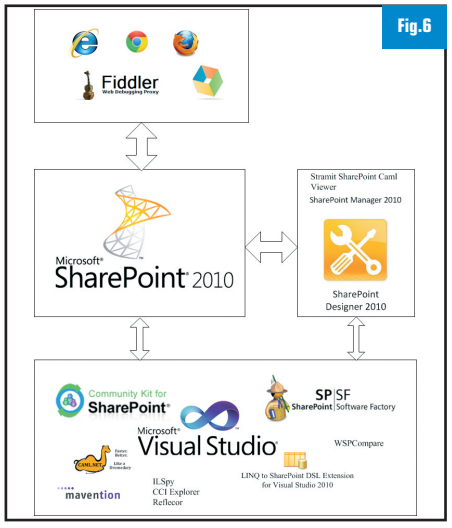


Fig.6

application Windows : on peut ainsi voir et modifier toutes les propriétés des objets SharePoint : que ce soit un site, un élément, un type de contenu… C'est en quelque sorte un Explorateur de serveur puissance 10 [Fig.5]. Stramit SharePoint Caml Viewer (spcamlviewer.codeplex.com) permet de visualiser les requêtes CAML générées par SharePoint pour les différentes vues. Un super outil pour SharePoint 2007 et 2010. Toujours à propos de CAML, U2U CAML Query Builder Feature (www.u2u.net/Tools/SharePointCamlQueryBuilder.aspx) est un composant qui s'installe sur SharePoint et qui permet de créer des requêtes CAML en mode WYSIWYG. Tout développeur .net a dans sa boîte à outils un décompileur : les programmeurs SharePoint en ont régulièrement besoin afin de percer certains mystères cachés dans les DLL SharePoint… Reflector (reflector.red-gate.com) est devenu payant. De nouveaux outils open source sont sortis : ILSpy (wiki.sharpdevelop.net/ilspy.ashx), CCI Explorer (cciexplorer.codeplex .com) …

SharePoint : avant tout un portail web

Fiddler (fiddler2.com) ou HttpWatch (http-watch) sont des outils qui permettent de tracer ce qu'il se passe au niveau requêtes http et ainsi de connaître le nombre de requêtes, leurs tailles, le contenu exact du flux HTML… Quand il s'agit de sites exposés sur Internet et d'optimisations, ces outils sont indispensables. Problème de CSS, js ou HTML ? L'extension Firebug (getfirebug.com) résoudra bien des soucis. Internet Explorer et Chrome fournissent des outils en standard tout aussi pratiques.

Pour finir, quelques outils que j'utilise aussi régulièrement : WSPCompare (wspcompare.codeplex.com) pour comparer 2 versions de packages WSP et L'explorateur Windows. Pourquoi ce dernier outil ? La ruche 14 (dossier d'installation SharePoint) est une mine d'informations : toutes les fonctionnalités SharePoint sont construites de façon standard, avec des features, des fichiers en CAML… Libre au développeur de chercher comment Microsoft a fait pour telle ou telle fonctionnalité !

En conclusion, Microsoft nous fournit des outils de base solides, mais aussi des possibilités d'extensions. C'est donc également vers la communauté SharePoint / .net qu'il faut constamment guetter afin d'avoir les derniers outils pour gagner en productivité [Fig.6].

■ Sylvain **Reverdy**
Consultant / Formateur SharePoint Winwise (www.winwise.com) à Tahiti
http://www.sharepointofview.fr/sylvain

Construire son développement

Construire un environnement de développement SharePoint opérationnel est souvent critiqué pour sa complexité. Heureusement, différentes méthodes s'offrent à nous pour répondre en fonction des moyens disponibles. Nous n'aborderons pas l'installation des différents prérequis mais nous verrons d'abord comment choisir sa plateforme, ensuite définir les comptes de sécurité et enfin comment automatiser l'installation d'une ferme de développement.

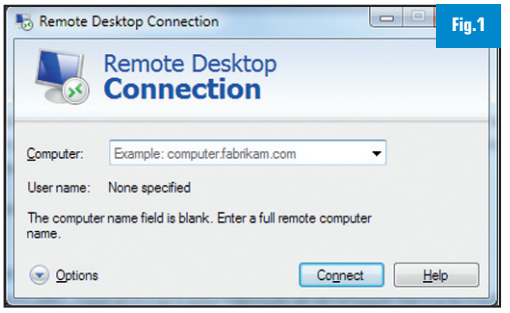


Fig.1

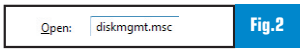


Fig.2

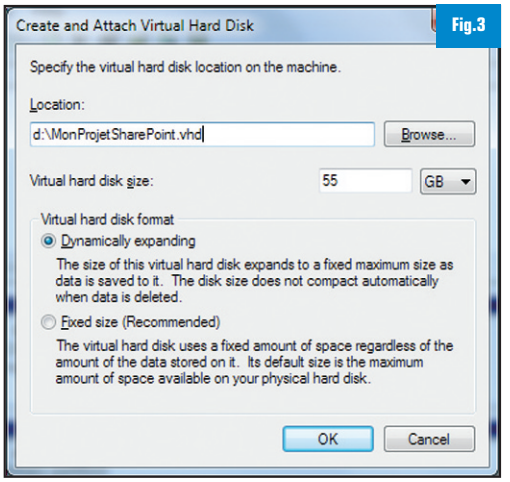


Fig.3

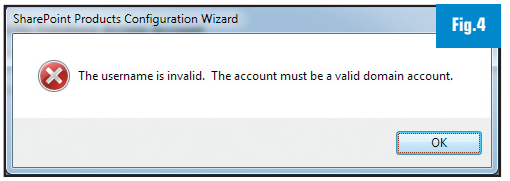


Fig.4

environnement de SharePoint

Terminal Server

La méthode la plus simple pour avoir une plateforme de développement est de posséder une machine (virtuelle ou physique) jointe au domaine d'entreprise sur laquelle sont installés les outils de développement ainsi qu'une ferme SharePoint. On accède ensuite à cette machine par la Connexion Bureau à Distance ou d'autres outils compatibles Terminal Server comme RDCMan de Microsoft (http://www.microsoft.com/downloads/en/details.aspx?FamilyID=4603c621-6de7-4ccb-9f51-d53dc7e48047) [Fig.1].

Machine virtuelle locale

Depuis MOSS 2007, la majorité des développeurs SharePoint préfèrent avoir leurs environnements dans des machines virtuelles locales avec leur propre domaine Active Directory. L'outil privilégié, Virtual PC 2007, ne peut maintenant plus être utilisé car il n'est pas capable de virtualiser un OS 64 bits, ce qui est un pré requis de SharePoint 2010. Son successeur, Windows Virtual PC, n'est pas exempt de cette limitation. Nous avons maintenant le choix entre Hyper-V, VMware Workstation/Player, Oracle Virtual Box, chacun ayant ses avantages et inconvénients (voir tableau ci-dessous).

Afin d'économiser le temps d'installation, Microsoft fournit une VM de démonstration, Information Worker, avec toute la gamme des produits 2010 et utilisable en développement : http://www.microsoft.com/downloads/en/details.aspx?FamilyID=751FA0D1-356C-4002-9C60-D539896C66CE

Les machines virtuelles offrent une parfaite autonomie aux développeurs mais nécessitent de gros moyens matériels. Une configuration minimale pouvant être un Core i7, 6 GO de RAM et des disques durs de 7200 tours minute.

Boot sur VHD

Si on dispose de machines dont la configuration n'est pas assez puissante pour faire tourner des machines virtuelles, on peut profiter d'une nouvelle fonction de Windows 7 et 2008 R2 : le boot sur VHD. Concrètement, on peut démarrer sur un disque virtuel qui va utiliser toutes les ressources de la machine comme un partitionnement classique. Pour créer un disque virtuel, on peut utiliser la Gestion des disques de Windows : [Fig.2 et 3].

Ceci fait, il ne reste qu'à indiquer au BCD, la base de registre régissant les systèmes démarrables, qu'on souhaite rattacher une nouvelle entrée. La méthode la plus simple est de faire une copie de l'entrée courante et de spécifier que le périphérique de stockage est un VHD. Toutes les opérations sont à effectuer avec cmd.exe en Administrateur et l'outil bcdedit.exe.

Installation sur Windows 7

SI aucune des options précédentes n'est viable, on peut également installer SharePoint 2010 sur un Windows Vista x64 SP1 ou supérieur. Lancer l'installeur depuis un Vista/7 affichera un message d'erreur indiquant que ce type d'OS n'est pas supporté, c'est normal : il faut modifier votre fichier Files\Setup\config.xml dans votre support d'installation (ou bien en créer un nouveau). Il faut rajouter un nœud Setting dont l'Id est AllowWindowsClientInstall et la valeur est True. Il faut ensuite choisir le mode d'installation de SharePoint : Farm ou Standalone. Ce dernier installe l'intégralité des services ainsi qu'un SQL Server Express. Il faut savoir que le choix conditionne ensuite le type de comptes de services qu'on pourra mettre ultérieurement **uniquement par les interfaces graphiques** :

- Une installation Farm impose des comptes de domaine.
- Une installation Standalone impose des comptes locaux.

[Fig.8].

Par PowerShell, il est en revanche possible de spécifier tout type de compte. Enfin, le service de Profils Utilisateurs pourra démarrer, mais le service de synchronisation ne fonctionnera qu'avec un compte de domaine.

Automatisation des installations de plateformes

SharePoint 2010 Easy Setup Script :

Easy Setup Script est un package conçu par Microsoft pour faciliter l'installation non seulement de SharePoint mais aussi de ses applications périphériques telles que Visual Studio 2010, Office ou encore Expression.

L'installation peut être :

- locale à un Windows Vista SP1 ou supérieur.
- dans un nouveau VHD créé à partir d'un modèle existant et qui sera ensuite attaché au démarrage.

On peut choisir finement ce que l'on souhaite installer grâce au fichier de paramétrage XML. On peut soit spécifier dans ce fichier les clés d'enregistrement et les emplacements des binaires des différentes applications soit les laisser tels quels pour télécharger et installer des versions d'évaluations.

http://www.microsoft.com/downloads/en/details.aspx?FamilyID=54dc2eef-e9ea-4c7b-9470-ec5cb58414de

AutoSPInstaller

Ce package de scripts permet d'installer et configurer de bout en bout une ferme SharePoint 2010. Il repose sur deux fichiers de configuration, le premier décrivant où sont les fichiers d'installation de SharePoint et le second décrivant finement les composants de la ferme (applications Web, services applicatifs…). Site : http://autospinstaller.codeplex.com

Comparatif des solutions de virtualisation

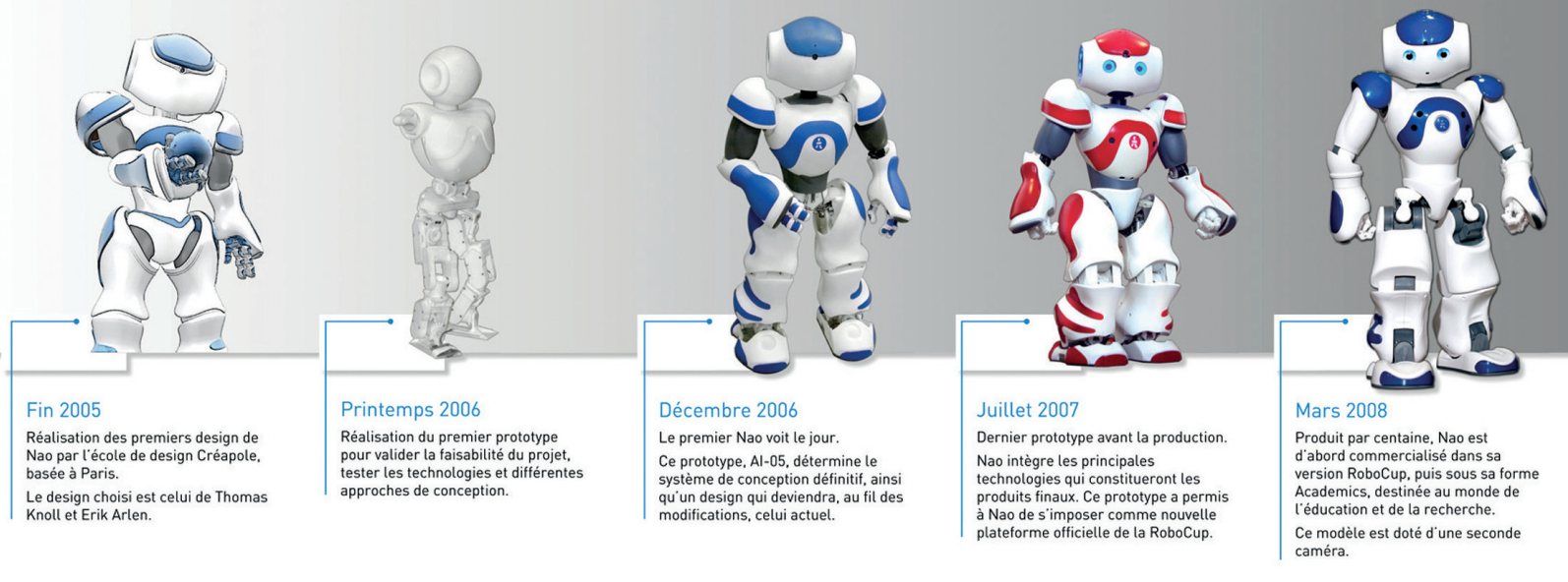
	VMware	Hyper-V	Virtual Box
Support VHD	Détourné	Natif	Natif
Partage de fichiers	Possible hors réseau	Nécessite un accès réseau	Possible hors réseau
Copier-Coller	Natif avec les additions	Nécessite une connexion Remote Desktop	Natif avec les additions
Allocation mémoire	Utilisation du fichier d'échange lors de dépassement	Dynamique avec le SP1	Dynamique par configuration en ligne de commande depuis la v 3.2.0.
Prix	Gratuit avec VMware Player.	Intégré à Windows Server ou gratuit en standalone.	Gratuit
Snapshots	Uniquement dans VMware Workstation	Natif	Natif

■ Pierrick **Catro-Brouillet**
Consultant/Formateur SharePoint,
http://blogs.developpeur.org/spbrouillet,
Winwise

La saga de Nao

Une plateforme ouverte aux développeurs

Une aventure extraordinaire s'ouvre désormais aux développeurs : Aldebaran, le concepteur de Nao, le seul robot humanoïde, a décidé de faire appel à la communauté des développeurs pour élargir le champ de ses applications.



Fin 2005

Réalisation des premiers design de Nao par l'école de design Créapole, basée à Paris.

Le design choisi est celui de Thomas Knoll et Erik Arlen.

Printemps 2006

Réalisation du premier prototype pour valider la faisabilité du projet, tester les technologies et différentes approches de conception.

Décembre 2006

Le premier Nao voit le jour.

Ce prototype, AI-05, détermine le système de conception définitif, ainsi qu'un design qui deviendra, au fil des modifications, celui actuel.

Juillet 2007

Dernier prototype avant la production.

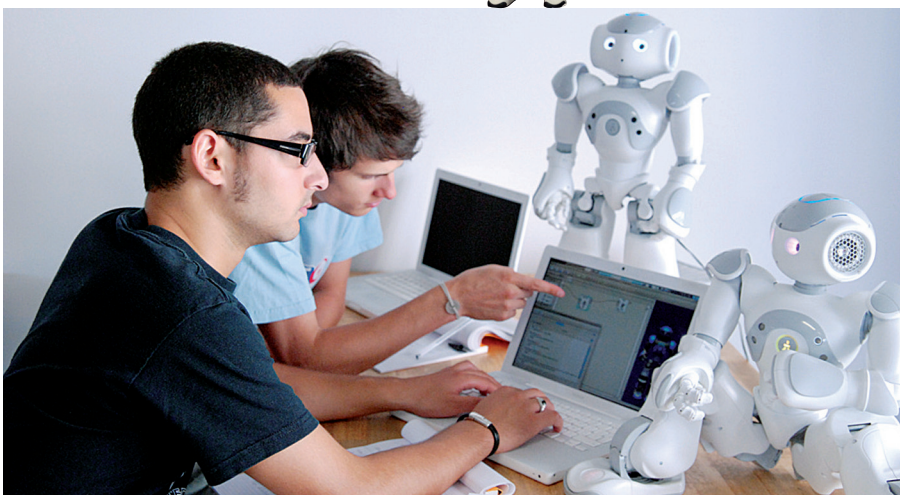
Nao intègre les principales technologies qui constitueront les produits finaux. Ce prototype a permis à Nao de s'imposer comme nouvelle plateforme officielle de la RoboCup.

Mars 2008

Produit par centaine, Nao est d'abord commercialisé dans sa version RoboCup, puis sous sa forme Academics, destinée au monde de l'éducation et de la recherche.

Ce modèle est doté d'une seconde caméra.

A 18H, dans les locaux d'un immeuble post-industriel parisien, tout le monde s'affaire. L'ambiance de ruche d'une start-up. Nous sommes dans l'antre d'Aldebaran, l'entreprise qui a conçu et qui fabrique Nao. Ce robot made in France mesure 58 cms de haut, et pèse 5,2 kgs ; il marche, parle une vingtaine de langues, se géolocalise, reconnaît les visages et la parole.



Une plateforme ouverte

Jérôme Monceaux est le 7e employé d'Aldebaran, autant dire qu'il y travaille presque depuis l'origine. Concepteur de Chorégraphe, l'outil de programmation de Nao, il a créé l'architecture logicielle du robot. Le programme principal de NAO est écrit majoritairement en C++, mais *« avec un esprit de cross-langage, afin que ce middleware s'adapte à tout »*, nous explique-t-il. Le système est un Linux embarqué basé sur la distribution OpenEmbedded. A l'époque, Microsoft proposait MSRS, un langage robotique, mais pas compatible avec l'embarqué et avec les robots humanoïdes.

> *« Il est très simple de créer son propre plug-in »*

« Notre grande force, et qui fait même que nous sommes en avance, c'est que nous avons tout intégré ! » nous explique Jérôme Monceaux, en charge du développement des applications de Nao. Le robot en effet comprend sa plate-forme logicielle : programmes embarqués et librairie de plug-ins : ALMotion, pour contrôler les déplacements du robot, ALTextToSpeech, moteur de synthèse (le robot parle 20 langues). Ces modules sont nombreux : détection et reconnaissance vocale, détection et reconnaissance de visages, reconnaissance d'image, localisation des sources sonores… Mais le middleware est aussi un environnement ouvert : *« il est très simple de créer son propre plug-in, et de l'exploiter dans les langages de programmation compatibles »*. Il existe plus de 1500 robots dans le monde, et une centaine de développeurs particuliers développent déjà pour Nao, dont le quart en France. En effet, en 2009, Aldebaran avait fait un bêta-programme pour se constituer un écosystème de développeurs. En 2011, la société a décidé de mettre en place un programme ambitieux, volontariste, afin d'inciter les développeurs particuliers à prendre en main la plateforme et à participer à la création de la « vie » de NAO.

Aldebaran lance le Developer Program, qui *« permet de mettre à disposition le robot à des particuliers qui ont des capacités de développeur, dans le but de participer à cette grande aventure de la robotique humanoïde qui commence aujourd'hui. On met à la disposition du développeur de nombreux outils pour créer ses applications, pour réinventer la robotique humanoïde, et la partager avec les autres clients Aldebaran, que ce soit du partage de lignes de code (Python, C++ ou Chorégraphe) ou au travers d'un Store qui sera bientôt disponible pour les possesseurs de Nao.»*

Dans le cadre de ce programme, les développeurs bénéficient d'un prix spécial pour

Depuis Juin 2009

La version actuelle de Nao, dotée de mains préhensibles et d'un capteur d'electricité statique au sommet du crâne.

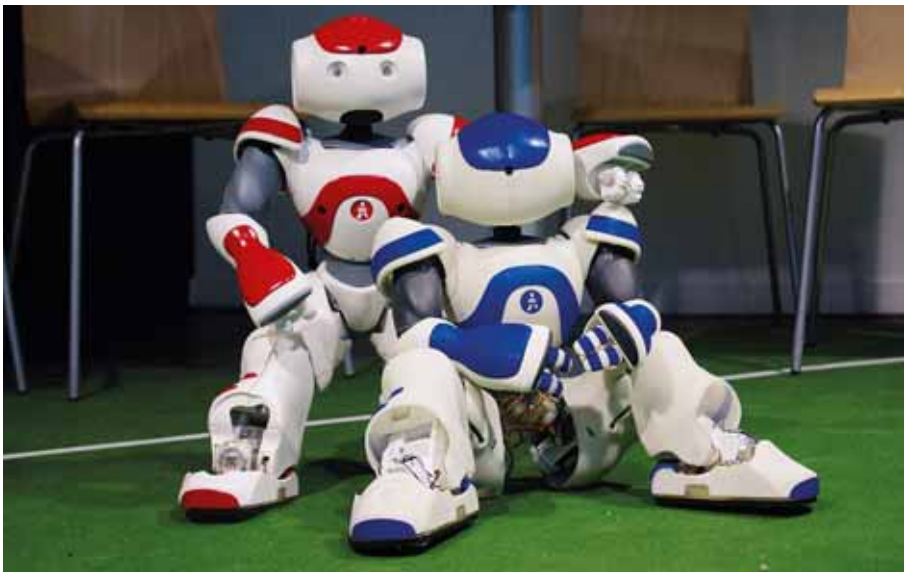
ALDEBARAN Robotics

acquérir Nao, ses plug-in et les outils de développement : 3 600 euros, au lieu de 12 000 euros. Le panel des applications que les développeurs pourront créer est infini : professionnelles, ludiques, ou de service. On peut imaginer différents services pour *« la famille, les enfants, ou encore de jeunes autistes ou des personnes atteintes de maladie d'Alzheimer ; ou tout autre service qui rend la vie plus simple »*, explique Jérôme Monceaux.

« Nao est connecté en permanence : je peux y accéder par Internet pour le contrôler et visualiser les données. Je peux contrôler son volume sonore par exemple », nous explique Jérôme, qui coupe ainsi le son de Nao, pour les commodités de la démo. Bien entendu, on accède aussi via le browser à sa configuration actuelle, aux applications qu'il lance, au langage qu'il parle.

Un robot qui marche

En décembre 2005, **Bruno Maisonnier** décide de créer un robot humanoïde grand public et programmable. La production commence en 2007. Il y a aujourd'hui 1500 Naos en service dans le monde, et 150 personnes chez Aldebaran, dont une cinquantaine à la R&D. A l'époque, Bruno Maisonnier estime que la technologie mondiale est prête à offrir les éléments qui pourraient répondre aux besoins d'un robot humanoïde de taille et prix raisonnable. Il rassemble donc autour de lui, quelques « intrépides » ingénieurs, et ensemble, ils se lancent dans ce projet fou de créer un robot humanoïde de toute pièce. L'équipe originale, composée d'à peine une dizaine d'ingénieurs et quelques stagiaires qui sont mécatroniciens, informaticiens, électroniciens… Pour lancer un tel développement, les compétences devaient être très larges et couvrir une grande partie des besoins. Linux, mathématique des mouvements de chaînes articulés, traitement du signal, synthèse vocale, analyse d'image, cross compilation, serveurs qui n'arrêtent pas d'être trop petits, problème de priorité entre les threads, interférences électromagnétiques et parasites en tous genres, solution de backup server, papiers dans l'imprimante : La liste est longue des sujets à aborder simultanément pour que le corps du projet avance : créer la première plateforme humanoïde européenne.



L'équipe en 2005

Jérôme Monceaux, nous confie, que 5 minutes avant la première représentation télévisée de NAO sur Direct 8 en direct, NAO ne voulait pas démarrer. *« C'était la première fois que NAO se trouvait sur ses propres batteries que nous avions logées dans un sac à dos adapté. Nous souhaitions absolument que NAO soit montré en robot autonome en énergie. En plein stress, nous avons appelé l'électronicien de la bande qui, pris d'un éclair de génie, s'écria « Coupe le fil bleu!!!! » et le NAO démarra enfin quelques secondes avant le lancement de l'émission… »*

L'équipe s'est donc lancée dans le défi de faire marcher cette machine. Cette contrainte distingue le robot des objets habituels : il peut tomber, et doit donc disposer d'une résistance, d'une fiabilité particulière ! Contrôler un robot qui marche nécessite une infrastructure logicielle dédiée. A peine deux prototypes sont-ils créés, en 2006, que la prestigieuse RoboCup sélectionne Nao comme robot officiel de la compétition, succédant à l'Aibo de Sony ! Les 17 employés que compte l'entreprise alors ont un défi à relever : ils ont quelques mois pour stabiliser les logiciels,

Anatomie de Nao

- **25** moteurs
- **350** processeurs, tous programmables
- **OS** : distribution Linux customisée basée sur OpenEmbedded

Le modèle standard, H 25, dispose de 25 moteurs (2 dans les mains, 2 dans l'épaule, 2 dans le cou…)

la plateforme et livrer 35 robots à Atlanta ! *« Nous avons pris des cheveux blancs»,* se souvient Jérôme. Mais ce challenge les a amenés à améliorer la qualité du robot à une vitesse phénoménale.

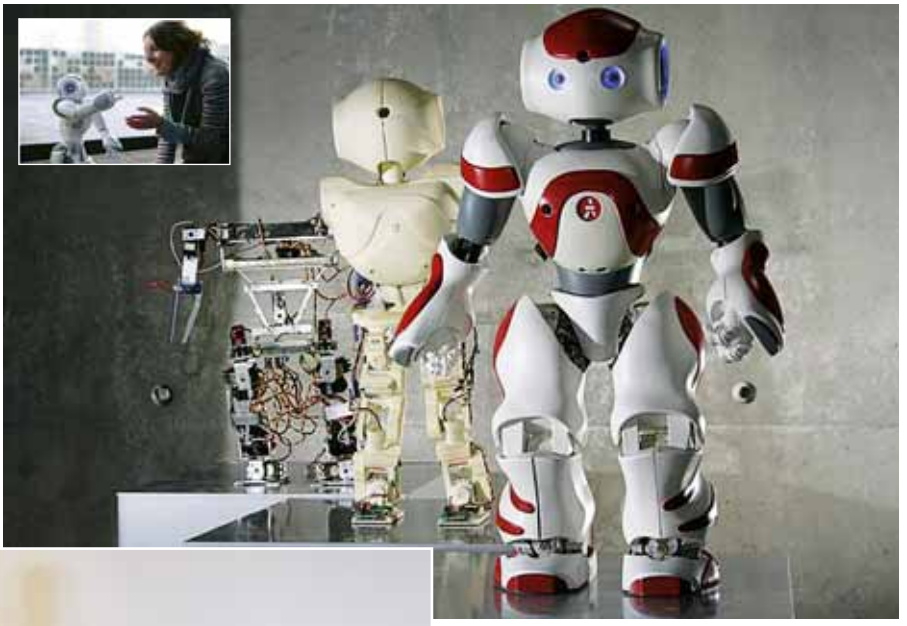
> *« La mécatronique sera au point avant l'intelligence artificielle »*

Chez l'homme, le centre de l'équilibre réside dans l'oreille interne. Nao dispose d'un capteur spécifique, une centrale inertielle, qui contrôle l'orientation du torse. Comment faire marcher le robot, sans qu'il chute ? Et s'il chute, comment le protéger ? (Comme nous, Nao a le « réflexe » de mettre sa main vers le sol, en cas de chute…)

David Gouaillier, « Motion Creator » chez Aldebaran, met au point *« les outils pour faire marcher Nao sans danger »*. David est le deuxième employé d'Aldebaran. Il était étudiant à l'Université de Versailles, où il préparait une thèse sur le mouvement des robots humanoïdes. Il rencontre Bruno Maisonnier en février 2005, et vient épauler Brice Marnier, qui mettait au point l'architecture électronique (qui comprend actuellement 80 cartes). Le tandem réalise le premier prototype. David terminera sa thèse quelques années plus tard, en parallèle à son emploi.

Nao est équipé du « meilleur moteur au monde » : celui qui anime le robot envoyé par la NASA sur Mars. Mais, nous explique David Gouaillier, la technologie actuelle des moteurs, ou « actionneurs », ne permet pas encore la même fluidité et souplesse que celle des êtres vivants, la « compliance » disent les spécialistes. Comment reproduire cette « intelligence des muscles » ? *« Résoudre ces difficultés passionne les chercheurs ; ces progrès en mécatronique seront mis au point avant l'intelligence artificielle »*, prédit David. D'ailleurs, une échéance est déjà fixée par les organisateurs de la RoboCup, qui ont planifié pour 2050 un match de football entre humains et robots !

■ Jean **Kaminsky**



Jérôme Monceaux et NAO

Le robot de demain : ROMEO

Roméo est un ambitieux projet. Il sera deux fois plus haut que Nao : 1,30 m. Il est dédié au service des hommes et se doit donc d'être plus proche de leur taille.

Les pouvoirs publics ont lancé des appels à projets pour contribuer à la prise en charge des personnes âgées et dépendantes.

Demain, elles pourront rester à domicile, avec l'aide et sous la surveillance de Roméo…

La révolution robotique et Nao

Après la saga de Nao et d'Aldebaran Robotics et avant de passer à l'utilisation réelle du robot, arrêtons-nous quelques instants sur le marché, le programme développeur et le robot en lui-même.

Le marché robotique est difficile à cerner car cela concerne aussi bien les robots de type aspirateurs-robots que les modèles de type Nao et les kits de montage. En 2009, ce marché représentait environ 3,3 milliards de dollars, en 2020, il en pèsera peut être plus de 100 ! C'est une véritable révolution qui s'annonce pour l'aide à la personne, le risque industriel, le divertissement, le domaine médical, etc. Le robot prendra des formes très différentes : humanoïde pour l'aide à la personne, les thérapies, « forme mobile » avec roues pour un site industriel, etc. L'interaction entre la machine, l'homme et son environnement se fera aussi de manière très différente : autonome avec une intelligence d'action selon la situation, piloté à distance, programme pré-configuré. On peut dire qu'une robotique utilitaire et de service va voir le jour et existe déjà. Mais il est certain que de nouveaux usages apparaîtront durant les prochaines années.

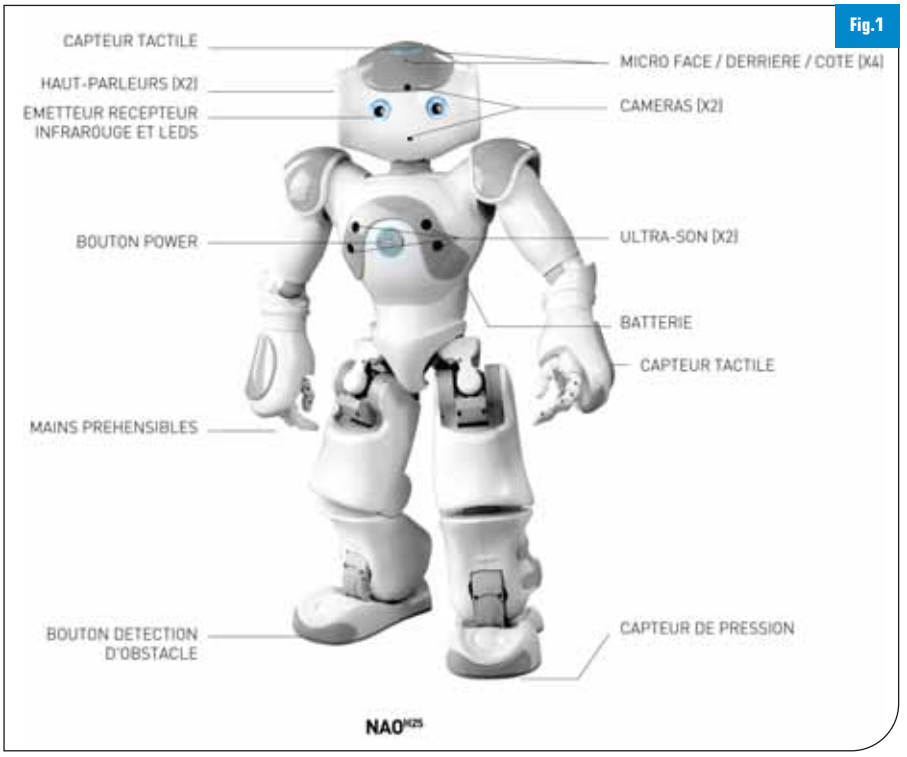


Fig.1

La France, terre robotique

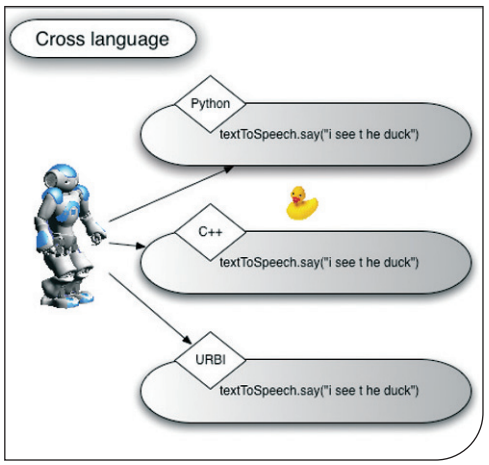
Notre pays a une réelle capacité de développement, de recherche autour des technologies robotiques. Plusieurs sociétés sont actuellement à la pointe de la technologie et des usages. La plus connue est Aldebaran avec Nao. Il existe quelques dizaines de constructeurs et d'éditeurs spécialisés. La communauté « Cap Robotique » en regroupe une douzaine. Son objectif est de faire de la France un des leaders mondiaux et faire émerger durablement le marché, les usages, les nouveaux objets communicants. Romeo, robot humanoïde, fait partie des projets de Cap Robotique. Lyon a accueilli en mars dernier un des plus importants salons robotiques européen : Innorobo 2011. Il a rassemblé 80 exposants, 10 000 entrées, 100 robots présentés, des cycles de conférences.

Aujourd'hui, le marché demeure émergent et la recherche est la clé de voûte des sociétés. La partie matérielle n'est qu'un aspect d'un robot. Il faut en concevoir le design, l'usage, son intelligence. Chaque société possède son propre laboratoire interne avec une ou plusieurs spécialités. Par exemple, l'INRIA travaille autour du projet Acroban, un robot humanoïde capable de prendre la main. Le projet se définit ainsi *« Acroban est un robot léger et souple capable de se déplacer dynamiquement (marche semi-passive), d'effectuer des mouvements mimant le vivant, et d'offrir de nouvelles possibilités d'interactions physiques et ludiques entre un homme et un robot. Nous avons développé cette plateforme pour explorer le rôle de la morphologie dans l'acquisition de savoir-faire moteurs complexes, ainsi que pour explorer de nouveaux types d'interactions homme-robots ».* Mais cette industrie demeure fragile, elle nécessite des soutiens, une politique active et volontariste. Car pour que ces sociétés vivent et continuent à travailler en France, il faut un marché. Notre pays a un réel atout à condition d'être offensif. Aux politiques et responsables de faire l'effort. Le financement reste crucial surtout avec un marché interne faible, et du fait qu'il faut s'imposer au niveau mondial. Dernièrement, Aldebaran a bénéficié d'un investissement de Intel Capital, à hauteur de 13 millions de dollars. Bruno Bonnell (Robopolis) croit fermement aux chances de la France mais il

Robopolis : quel robot tu veux ?

Le monde robotique est vaste et parfois obscur pour ceux qui n'y sont pas habitués. Robopolis est un membre fondateur de Syrobo et on y trouve Bruno Bonnell (co-fondateur d'Infogrammes, gérant de Robopolis. Cette boutique distribue en exclusivité plusieurs marques iRobot, Genibo, Robotis… La diversité prime, allant du robot domestique à la plate-forme robotique. Et les prix varient énormément de -300 euros à plus de 10 000 !

Site web : http://www.shop-robopolis.com



ne faut pas laisser les autres pays prendre trop d'avance. Pour cela, il faut rassembler les compétences (pourquoi pas une valley robotique), créer une école spécialisée, des partenariats.

Il existe même un syndicat robotique : Syrobo. Un des arguments est que *« SYROBO veut rassembler autant les acteurs industriels que les universités et centres de recherches intéressés par le développement de la robotique de services. Les sociétés d'informatique, d'électronique, de mécanique mais également de matériaux, comme les plastiques ou textiles ou encore de logiciels applicatifs dans les domaines du langage, de la vision, de l'intelligence artificielle…sont invitées à participer aux études, évènements et présentations organisés par Syrobo. »*. Car la robotique est l'agrégation de compétences multiples : designer, technicien, développeur, sociologue, philosophe, chercheurs, geek, etc.

Nao : quel marché ?

Aujourd'hui, le robot Nao n'est pas destiné au grand public, même si le programme développeur est proposé à un prix agressif : 3 600 €. Actuellement, ce programme se destine avant tout aux chercheurs, universitaires, geeks, hobbyistes. Son objectif est de concevoir de nouveaux usages, développer des briques logicielles, augmenter l'intelligence artificielle, mieux concevoir l'interaction homme – machine, peaufiner la technologie vocale, le comportement. Les chantiers sont nombreux. L'un des objectifs est de pouvoir contribuer activement au futur NaoStore. L'utilisateur pourra télécharger des comportements, programmes pour son robot directement depuis la boutique en ligne. Le programme développeur permet d'accéder à l'ensemble de la documentation technique. Les participants profiteront des dernières versions de Nao, de tous les outils de développement, auront un accès privilégié au code source du robot et une porte grande ouverte à la communauté. Pour aider les développeurs, chercheurs, la société propose un outillage très complet (voir article technique) accompagné d'API et de frameworks.

Nao, c'est quoi ?

Nao est un robot de 58 cm de hauteur, pour environ 5,2kg. Il possède une autonomie d'environ 90 minutes. La connexion réseau se fait par Ethernet et wifi. C'est surtout un concentré de technologies : cerveaux-moteurs, capteurs tactiles, 32 capteurs différents, caméras embarquées, microphone, haut-parleurs, logiciels embarqués pour l'intelligence et les fonctions (système Linux temps réel adapté, carte mère située dans la tête), plus de 10 moteurs intégrés, gyromètre, accéléromètre, infrarouge, etc.

Il n'y a qu'à regarder le schéma pour se rendre compte de la complexité du robot [Fig.1]. Surtout, Nao évolue régulièrement, le laboratoire interne travaille sur de nombreuses améliorations et de nouvelles fonctions, par exemple, comment améliorer la dissipation de la chaleur (la tête chauffe car la carte mère, et donc le processeur, est située là), améliorer le comportement sur l'équilibre particulièrement. Il s'agit de développer des « accessoires » comme des têtes personnalisables ou encore un dock de recharge pour que Nao se branche et recharge sa batterie tout seul.

Pour aider Nao à se développer et faire croître sa communauté, Aldebaran a récemment libéré le code embarqué du robot. Cette annonce concerne NAOqi, véritable centre névralgique, car c'est lui qui gère et envoie les ordres à l'ensemble des composants et des membres. Il se programme en C++, Python, C# ou Urbi.

Etes-vous prêt à faire une nouvelle expérience ?

■ François **Tonic**

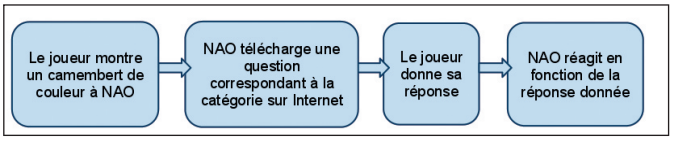
Génération Robots mise sur Nao

Cette année, Génération Robots a lancé deux activités supplémentaires autour du robot. La première consiste à créer ses propres capteurs intelligents pour robots. La seconde activité, consiste en la réalisation de comportements évolués pour les robots NAO. Une équipe de développement composée d'ingénieurs et de docteurs en robotique travaille quotidiennement avec NAO pour lui apprendre des comportements de jeu faisant une large place à l'interaction sociale homme-robot. Génération Robots anime des formations à la programmation de NAO en partenariat avec Aldebaran Robotics. http://www.generationrobots.com

Programmez Nao... c'est trivial

Cela fait maintenant trente ans que petits et grands jouent au Trivial Pursuit. Du Tonton Albert qui connaît la réponse à toutes les questions à la petite cousine Sophie qui sèche toujours sur les questions de géographie, tout le monde y trouve son compte. Mais NAO, le plus joueur des robots humanoïdes, était pour ainsi dire exclu de la partie. Il était temps d'y remédier !

Lançons-nous ! Créons ensemble le premier Trivial Pursuit robotique ! Utilisation des API fournies, création de mouvements fluides, analyse d'image en C++, accès à un webservice, nous allons faire ensemble un tour des capacités de NAO.



Installation du SDK

Le SDK fourni avec NAO est dense. Il contient :

- la suite Choregraphe pour Windows, Linux et MAC OS X, qui fournit un environnement graphique de développement et d'animation sur NAO. Il propose également un robot simulé, qui permet de tester des comportements sans avoir de robot.
- la documentation complète de NAO : ses caractéristiques techniques, son fonctionnement interne, les techniques de programmation et la description exhaustive de ses API

Tous les outils sont disponibles en ligne sur le site du NAO Developer Program, developer.aldebaran-robotics.com. En ouvrant gratuitement un compte, on peut télécharger le SDK et utiliser la suite Choregraphe pour une période d'essai de 30 jours.

Architecture du framework de Développement

Le framework de développement, appelé NAOqi (« qi » signifie esprit en chinois) essaie de répondre à tous les besoins spécifiques de la robotique : traitements parallèles, gestion des ressources, synchronisation, modularité. Quelques-unes de ces caractéristiques :

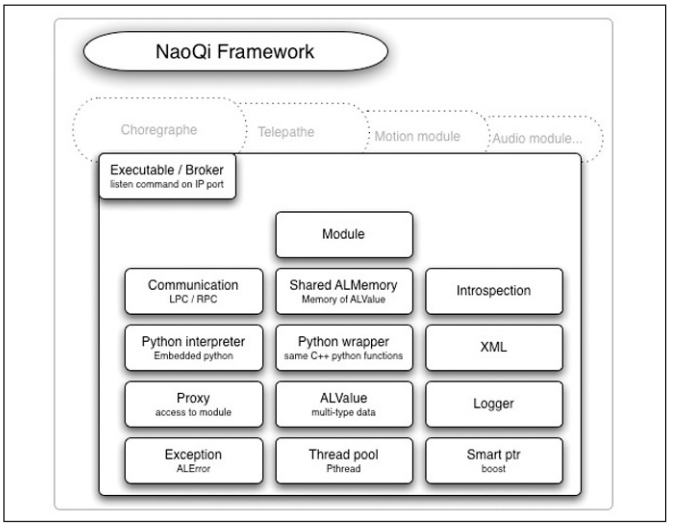
- *Cross-language* : il permet de contrôler NAO en C++, en Python, ou en Urbi.

Adapté à la programmation distribuée : il permet de trouver quels programmes offrent quelles méthodes, y compris sur le réseau.

- *Modulaire* : on peut inscrire des nouvelles méthodes en chargeant un « module », ce qui peut être fait soit en « local », soit en « remote ». En local, le module est dans une bibliothèque partagée, et partage l'espace d'adressage de NAOqi, pour une performance optimale. En remote – très utile pour du calcul distribué – le module est dans un exécutable séparé, pour plus de sécurité. Pour passer d'un mode à l'autre, on n'a pas besoin de changer une seule ligne de code !

On peut compter une vingtaine de modules fournis avec le robot : citons par exemple : ALMotion, ALTextToSpeech, ALFaceDetection (le préfixe AL signifiant Aldebaran Library).

Un de ces modules est particulièrement important : il s'agit d'ALMemory. Il permet de partager des données de manière optimale entre tous les modules (local ou remote) connectés au NAO. Son API permet d'insérer des données de manière atomique, de les mettre à jour, ou encore d'inscrire des callbacks à appeler lors de la mise à jour d'une valeur.



Appeler les méthodes d'un module se fait de la même façon, que ce soit en local ou en remote, grâce au design pattern proxy. Il faut tout d'abord créer un objet « proxy » vers un module, et ensuite appeler sur cet objet la méthode désirée.

Cela donnera en C++ :

```
ALTextToSpeechProxy* tts = getParentBroker()->getProxy(« AL
TextToSpeech ») ;
tts->say(« Hello, robots! ») ;
```

Et en Python :

```
tts = ALProxy(« ALTextToSpeech »)
tts.say(« Hello, robots! » ) ;
```

Nous créons ainsi un proxy vers le module de synthèse vocale, et demandons à synthétiser la phrase « Hello, robots! ».

Extracteur de camembert

Dans le monde de NAO, les outils permettant de sélectionner des valeurs ayant du sens à partir de données brutes sont appelées des extracteurs. Un extracteur envoie des valeurs dans ALMemory, afin que d'autres modules ou programmes puissent accéder facilement à ces valeurs. Les extracteurs peuvent être de types :

- proprioceptifs : les données proviennent de NAO lui-même
- perceptifs : elles proviennent de l'observation par NAO de son environnement

Génération Robots

Le spécialiste du robot personnel programmable

Robot + Android = ∞

Donnez vie à votre code

Vous avez envie de coder autre chose que des pages web et des applications en ligne de commande ?
Passez au concret en vous attaquant à un vrai challenge.
Combinez programmation sur smartphone et sur robot mobile dans le langage de votre choix et découvrez un nouveau monde d'applications passionnantes.

Frais d'expédition offerts
Code promo: **PROGRAMMEZ11**

Expédition gratuite sans montant minimum de commande
Offre valable jusqu'au 15/09/2011

www.generationrobots.com

- extraceptifs : elles proviennent d'Internet, d'un device distant…

L'extracteur que nous allons utiliser fait partie des extracteurs perceptifs : il utilise uniquement la caméra de NAO.

L'analyse d'image se fait généralement en C++, pour des raisons de performance. Regardons un peu ce qui est déjà disponible en matière de vision sur NAO :

- ALVideoDevice : ce module permet de lancer le « framegrabber », c'est-à-dire la récupération d'image. La méthode getImageLocal() permet quant à elle de récupérer un tableau contenant la largeur, la hauteur, le nombre de couches, l'espace de coloration, le timestamp, et enfin les pixels qui composent l'image. Ce module permet aussi de paramétrer la prise de vue : résolution, frame rate, saturation, contraste – une trentaine de paramètres en tout.
- ALVisionToolbox : ce module offre des méthodes telles que takePicture, pour enregistrer une photo, startVideoRecord, pour enregistrer un film, ou encore halfPress, qui permet de faire un autofocus.
- ALFaceDetection : comme son nom l'indique, ce module est spécialisé dans la détection de visages, indiquant le nombre de visages vus et leur position dans l'image.
- ALVisionRecognition : ce module permet de reconnaître de manière générique n'importe quel objet enregistré dans une base sur le robot.

Nous aurions pu tout programmer en utilisant des bibliothèques telles qu'OpenCV pour analyser, trier les milliers de pixels provenant directement du module ALVideoDevice… Heureusement, d'autres se sont posé ces questions avant nous !

Les camemberts colorés que nous voulons faire reconnaître sont des objets, donc regardons le module Vision Recognition de plus près. Il s'agit d'un extracteur qui se sert d'une base d'objets à reconnaître située sur le robot, au format « vrd ». L'extracteur est chargé de détecter ces objets sur les images provenant de la caméra. Sa grande force est de fonctionner même si l'objet est retourné, de côté, ou plus ou moins loin de la caméra. Lorsqu'un objet est détecté, sa référence et son positionnement sont stockés dans ALMemory, de manière à ce que tous les autres modules, et tous les comportements, puissent bénéficier de cette information.

Fabriquer la base de données

Pour enregistrer les objets à reconnaître dans la base, on utilisera le widget « Video Monitor » de Choregraphe. Celui-ci nous évitera une saisie manuelle fastidieuse des caractéristiques physiques des objets, puisqu'il permet d'enregistrer directement les nouveaux objets en trois clics. Voici la marche à suivre :

Lancer la capture d'image du Video Monitor, puis cliquer sur « Learn ». Il apparaît alors un décompte, un peu comme dans les photomatons. Nao enregistre l'image à apprendre au bout du décompte [Fig.A].

Il s'agit maintenant de détourer la zone significative de l'objet à reconnaître : [Fig.B].

A la fin de ce détourage grossier, NAO va estimer si l'image possède suffisamment de points caractéristiques pour pouvoir être reconnue. Si c'est le cas, l'image est ajoutée à la base de données, et on peut alors lui associer un nom : [Fig.C].

Utiliser le module VisionRecognition

Lorsqu'un objet est reconnu, le module Vision Recognition place dans ALMemory une valeur formatée comme suit :

[[TimeStampField] [Picture_info_0 , Picture _info_1, . . . , Picture_info_N-1]] avec autant de Picture_Info que d'objets reconnus dans l'image. Chaque Picture_info = [[labels_list], matched_keypoints, ratio, [boundary_points]], on ne rentrera pas plus en détail ici car seule l'information des labels nous intéresse.

En Python, voici comment extraire cette valeur de label :

```
def __extract_label(self, data) :
   for info in data[1] :

  if data[0][1] in [« orange », « blue », « green », « yellow »,
  « pink », « brown »] :
    return data[0][1]
    return None
```

Récupérer et poser des questions

NAO peut facilement se connecter à Internet via WiFi ou Ethernet. Nous allons utiliser cette capacité pour récupérer directement des questions de quiz depuis le Web. CarlC, un des membres les plus actifs du NAO Developer Program, a créé une base de données de questions, accessibles sur 'http://trivia.alt-view.co.uk'.

Ce serveur fournit l'API suivante :

- http://trivia.alt-view.co.uk/session.qu?command=getsessionid

Cette page nous renvoie un cookie de session. Grâce à celui-ci, nous allons pouvoir éviter de poser les mêmes questions plusieurs fois au cours d'une même séance.

- http://trivia.alt-view.co.uk/question.qu?language=fr&sessionid={sessionid}

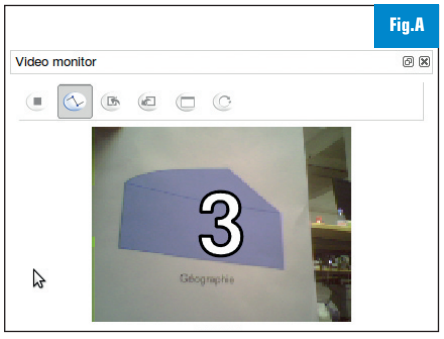
Cette page retourne une question dans la langue souhaitée au format suivant :

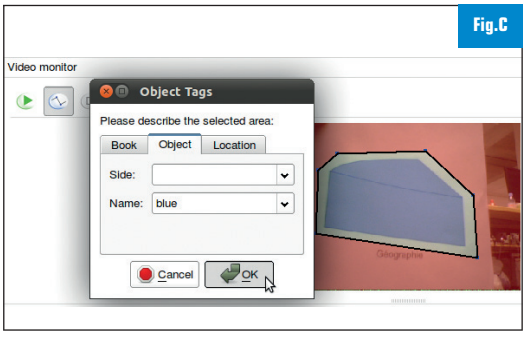
{intitulé de la question}|{bonne réponse}|{mauvaise_réponse1}|{mauvaise_réponse2}|{langue}|{difficulté}|{catégorie}

On aura par exemple :

Dans quelle ville est mort George Harrison?Los Angeles|Londres|Paris|French|Medium|Divertissement

Nous allons récupérer les questions en Python, pour cela il faut





créer une nouvelle boîte Choregraphe de type script, que nous allons appeler «Create Question ». Les scripts des boîtes Choregraphe possèdent toutes le même squelette, que nous allons faire apparaître dans la suite en grisé. Il s'agit ensuite généralement d'implémenter les fonctions qui y sont déjà présentes.
Nous allons utiliser dans notre script deux modules Python:
- urllib, qui permet de récupérer du contenu web
- random, qui permet entre autres de générer un nombre aléatoire

```
import urllib
import random
```

Ensuite, initialisons les variables dont nous aurons besoin dans notre code : ici, l'adresse du serveur et notre cookie de session.

```
class MyClass(GeneratedClass):
  def __init__(self):
    GeneratedClass.__init__(self)
    self.server = 'http://trivia.alt-view.co.uk'
    self.session_id = None
```

Ajoutons maintenant le corps de la fonction onLoad, qui sera appelée lors du chargement du projet.

```
def onLoad(self):
  # Récupérons tout d'abord le cookie de session
  session = urllib.urlopen(self.server + '/session.qu ? command
=getsessionid')
  # on insère cet identifiant dans la variable d'instance session_id
  self.session_id = session.read()
```

La boîte possède une entrée appelée onStart, de type string, de manière à ce que l'on puisse passer la catégorie de la question en argument. Lorsque cette entrée sera stimulée, la méthode onInput_onStart sera appelée et nous devrons alors télécharger, analyser, puis transmettre la question du Trivial Pursuit à la boîte suivante.

```
def onInput_onStart(self, category) :
#Téléchargement de la question :

  url = self.server + '/question.qu?language=en&category=' + category
  if self.session_id:
    url = url + '&sessionid=' + self.session_id
  response = urllib.urlopen(url)
  output = response.read()
```

Séparation du retour : l'énoncé de la question d'un côté, et les trois réponses de l'autre.

```
  question, rest = output.split('|', 1)
  answers = ["", "", ""]
  answers[0], answers[1], answers[2], rest = rest.split('|', 3)
```

Construction de l'énoncé de la question : NAO devra dire *« Quelle est la couleur du cheval blanc d'Henri IV ? Est-ce 1. Bleu 2. Blanc 3. Noir ?»*

```
  r = random.randint(1, 3) #randomisation de l'ordre des réponses
  question = question + ' Est-ce  1. ' + answers[(r-1) + ' 2. '
+ answers[r %2] + ' 3. ' + answers[(r+1) %2]
  correct_answer = r
```

Maintenant que nous avons construit la question à poser, il s'agit de se rappeler de la bonne réponse, pour pouvoir vérifier plus tard la réponse du joueur. Nous allons ici enregistrer la bonne réponse dans la variable ALMemory « TrivialPursuit/CorrectAnswer », de manière à pouvoir partager cette information avec l'ensemble des boîtes de notre comportement :

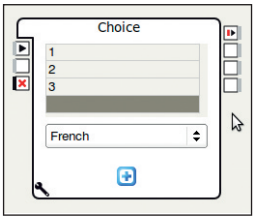
```
  memory = ALProxy('ALMemory')
  memory.insertData('TrivialPursuit/CorrectAnswer', correct_answer)
```

Enfin, nous activons la sortie de la boîte en transmettant l'énoncé de la question :

```
  self.onStopped(question)
```

Faire un bon programme interactif n'est pas si facile. En effet, les interfaces homme-machine sont un sujet très délicat, où les goûts et couleurs du développeur doivent s'associer avec des règles de design et de facilité d'utilisation.
Réaliser une interface sur un robot humanoïde est encore plus difficile. Il faut savoir synchroniser paroles et mouvements, démarrer et arrêter la reconnaissance vocale au bon moment, utiliser des codes d'interaction compréhensibles, tout cela en s'assurant que le robot garde des mouvements fluides.
De plus, il n'y a pas de réel standard d'interaction homme – robot comme il peut y en avoir pour les interfaces homme- ordinateur. Star Wars a posé les bases d'un standard avec C3PO, mais nous devons les enrichir ! Les développeurs d'Aldebaran Robotics se posent ces questions depuis quelques années et ont réalisé un certain nombre de briques d'interactions primaires, sous forme de boîtes Choregraphe. Dans le cas de NAO Trivial Pursuit, nous avons besoin de poser une question, et d'attendre que le joueur donne une réponse. Une boîte d'interaction primaire répond à ce besoin : c'est la boîte choix.
Cette boîte a beaucoup de paramètres qui permettent de régler finement l'interaction. Citons parmi les plus importants :
- l'activation ou non des animations gestuelles
- le nombre de fois où NAO répète la question s'il n'a pas eu de réponse
- l'activation des capteurs tactiles

La boîte Choix prend en entrée un argument de type string, qui est l'énoncé de la question, et active la sortie qui correspond à la réponse choisie par l'utilisateur, ou la sortie par défaut s'il n'y a pas eu de réponse.



Nous allons donc pouvoir relier la sortie de notre boîte «Create Question » à l'entrée de la boîte choix. Il faut ensuite valider la réponse de l'utilisateur. Nous allons créer une nouvelle boîte de type script, « Check Answer », qui contiendra le script suivant :

```
def onInput_onStart(self, answer) :
  memory = ALProxy('ALMemory')
  correct_answer = memory.getData('TrivialPursuit/CorrectAnswer')
  if correct_answer == p:
    self.onRight()
  else:
    self.onWrong()
```

Nous comparons donc la réponse choisie par l'utilisateur à la bonne réponse enregistrée dans ALMemory. Si l'utilisateur a juste, nous stimulerons la sortie onRight ; dans le cas contraire ce sera onWrong.

Animer NAO

La force de NAO, c'est son expressivité. C'est le premier robot humanoïde qui sait à la fois faire des mouvements de TaiChi et raconter des blagues. Pour l'exposition universelle de Shanghaï 2010, nous avions réalisé une première mondiale en faisant danser 20 NAO de manière synchronisée sur le boléro de Ravel. NAO est utilisé entre autres dans les projets de recherche européens GVLex (Gestes et Voix pour une Lecture expressive) et Feelix Growing (pour Feel, Interact, Express). Par ailleurs, des chercheurs se servent de NAO pour élaborer de nouvelles manières d'interagir avec les enfants autistes.

Cette expressivité repose sur différents modules :

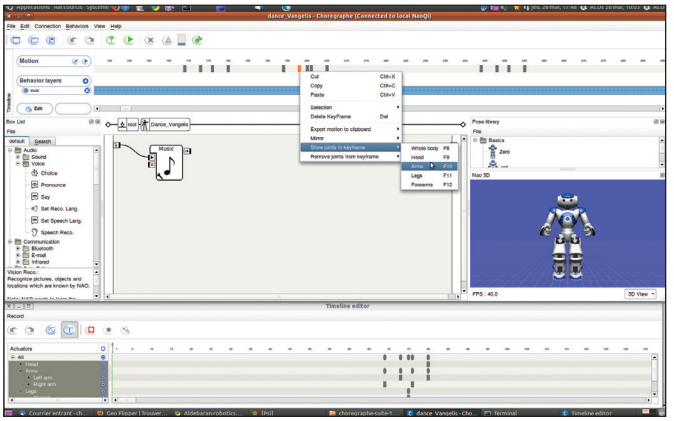
- ALMotion, qui permet de contrôler les positions des servo-moteurs. On peut retrouver dans ce module des méthodes simples à comprendre, comme « WalkTo » ou « openHand », mais aussi des fonctionnalités plus avancées comme « angleInterpolation » ou encore « getStiffnesses ».
- ALTextToSpeech, qui permet de faire dire un texte à NAO. Ce module gère actuellement une vingtaine de langues différentes, la langue par défaut étant l'anglais. Les principales méthodes du module sont « say » et « setLanguage ».
- ALAudioPlayer, qui permet de jouer des sons, au format wav ou mp3, grâce à la méthode « playFile », Ce module propose une API très riche, comme par exemple la méthode « setPanorama », qui permet de définir la variation d'intensité suivant la direction.
- ALLeds est le module de contrôle des LED de NAO, à savoir : 10 LED situées sur la calotte de la tête, 10 leds sur chaque oreille, et 20 segments situés dans les yeux. Les LED sont un élément important d'expressivité : mettre les LED des yeux de couleur vive donnera la sensation que NAO est joyeux, les mettre en rouge le fera paraître en colère. Et éteindre les yeux pendant 0.2 secondes toutes les 5 à 10 secondes donnera l'impression très réaliste que NAO cligne des yeux !

Yeux neutres *yeux enjoués* *yeux en colère :*

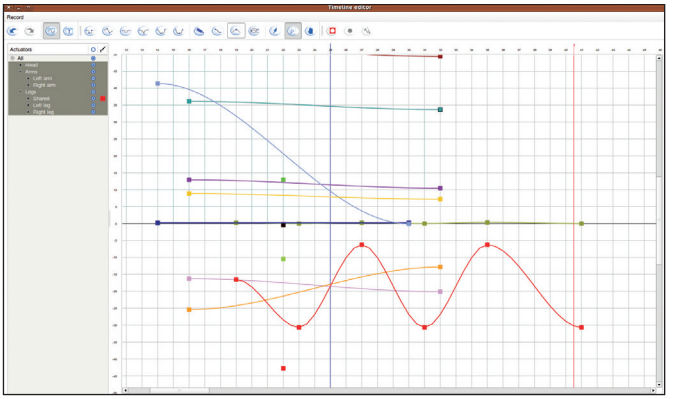
Ces modules sont utilisables en Python et en C++, grâce aux ALProxy. Mais NAO possède 25 moteurs et donc définir un mouvement directement dans le code, en définissant le passage d'une position à une autre de ces moteurs, est très laborieux.

Heureusement, Choregraphe permet de créer graphiquement des animations, au sein de boîtes particulières, appelées timelines. Le principe est simple : on enregistre sur une Motion timeline des « keyframes », c'est-à-dire des positions clefs de NAO. L'interpolation du mouvement d'une keyframe à une autre se fera alors automatiquement. Cette technique s'inspire des logiciels classiques d'animation (comme Flash par exemple).

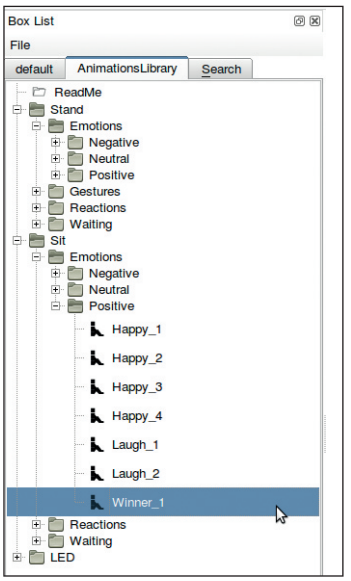
On peut régler plus finement les détails de chaque mouvement



grâce à un mode avancé de Choregraphe : le Motion Editor. Chaque moteur de NAO est alors représenté sous forme de courbe, et on peut changer la méthode d'interpolation entre deux points (méthode quadratique, Béziers, ...), ou encore voir les limites en vitesse et en angle de chaque moteur.



Malgré ces outils, créer une bonne animation demande tout de même un certain talent artistique. C'est pourquoi on trouve dans les équipes d'Aldebaran Robotics des personnes ayant, par exemple, une formation aux outils d'animation de personnages 3D. Ces personnes créent des mouvements particulièrement réussis pour NAO, qu'ils partagent ensuite au sein de « bibliothèques de boîtes » pour Chorégraphe. Dans le cadre de notre application, nous allons nous servir de boîtes provenant de la bibliothèque « AnimationsLibrary ». Dans cette bibliothèque, nous trouverons deux animations bien adaptées à notre jeu : Winner_1 et Sad_1. La première fait lever les bras de NAO en signe de victoire. La seconde le replie sur lui-même dans une position triste.

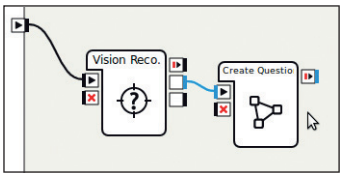


Intégration, packaging, et diffusion

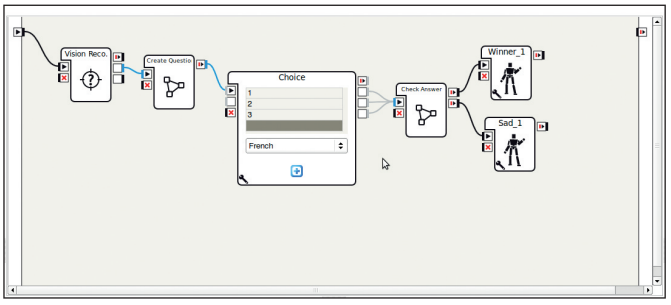
Nous allons maintenant pouvoir intégrer tout ce que nous avons vu jusqu'ici dans notre

application « Nao Trivial Pursuit ». Là encore, Chorégraphe est l'outil idéal pour finaliser une application. Dans notre cas, nous voulons un modèle évènementiel (« lorsqu'un camembert est détecté, lancer la question ». Pour faire cela, nous allons utiliser des boîtes de type « Flow Diagram ». Celles-ci ont la particularité de contenir elles-mêmes d'autres boîtes, et de faire des liens entre ces sous-boîtes. Nous allons ici relier la sortie de la boîte «Vision Reco.», qui est de type « string », à l'entrée de la boîte «Create Question », qui est elle aussi de type string.

Quand un camembert est détecté, la couleur de celui-ci est ainsi transmise d'une boîte à l'autre. Nous relions maintenant la boîte Create Question à la boîte « Choice », qui va énoncer la question, et reconnaître la réponse du joueur.



Nous allons relier la sortie de la boîte « Choice » à notre « Check Answer » que nous avons fait précédemment, et enfin relier les deux sorties aux animations correspondantes (Winner_1 si la réponse est bonne, Sad_1 si la réponse est mauvaise).



Voyons maintenant comment diffuser notre application. Tout d'abord, il faut savoir que les comportements faits avec Chorégraphe peuvent être sauvegardés de deux façons :

- en mode dossier : ce mode convient particulièrement pour versionner le code et/ou travailler à plusieurs sur un même projet
- au format « crg » : le crg est un package regroupant la boîte de plus haut-niveau du comportement et toutes ses sous-boîtes, mais aussi les sons, et toutes les autres données externes. C'est un format compressé qui convient bien à l'échange sur le réseau.

Nous proposons, en bêta pour l'instant, deux manières de diffuser ces comportements crg :

- la première consiste à les proposer en téléchargement sur un NAOstore, au mode de fonctionnement similaire à celui de l'AppStore ou de l'Android Market
- la seconde, plus innovante et plus adaptée au robot, est de diffuser son comportement au sein d'un « channel ». Un channel est un ensemble de comportements ayant une thématique commune. Les comportements sont ajoutés, enlevés, ou mis à jour en permanence sur les channels. Chaque channel peut avoir plusieurs contributeurs. Ce système permet à chacun d'ajouter des nouvelles fonctionnalités à son robot : souscrire aux channels « fun », « blagues » et « jeux de société », par exemple, donnera un NAO très joueur. « Nao Trivial Pursuit » pourrait tout à fait trouver sa place au sein de différents channels, tels que « Edutainment », ou encore « jeux de société ».

Enjoy !

■ Manuel **Rolland** - *Aldebaran Robotics*
■ Charlotte **Williams** - *Aldebaran Robotics*

L'été sera

Hackez votre Kinect, votre mon
bioinformatique, exosquelette :

L'humanité a connu de multiples révolutions économiques, technologiques et industrielles : l'écriture, la vapeur, l'électricité, le charbon, le pétrole et bien entendu, le silicium. Depuis deux mille ans, les Hommes cherchent à simplifier le travail, à automatiser des tâches. Le plus ancien ancêtre connu de l'ordinateur, ou tout le moins de l'automatisme, est la fabuleuse machine d'Anticythère, découverte en pleine mer il y a un siècle sur laquelle j'avais écrit un article pour une revue d'archéologie. Cette machine demeure partiellement une énigme pour les chercheurs, notamment quant à son fonctionnement exact. Il s'agit d'une sorte de calculatrice permettant de déterminer les positions du soleil et de la lune, avec un calendrier astronomique utilisant deux cycles astronomiques : le cycle de Méton et le cycle de Saros (pouvant déterminer les dates des éclipses). Un mode d'emploi était inscrit sur une plaque de bronze de la machine. Et elle possédait plusieurs cadrans, sur deux côtés. Sa construction remonterait aux alentours de 90 – 85 avant notre ère. Et proviendrait peut- être de Rhodes. Ce ne serait pas l'unique machine de ce type, plusieurs autres calculatrices auraient existé à Rome. La finesse et le perfectionnement du mécanisme laissent rêveur.

Cette digression historique nous amène au milieu du 20e siècle avec la machine allemande Enigma considérée comme l'un des premiers ordinateurs, puis ce furent les énormes machines à cartes perforées. Un des sauts technologiques sera franchi avec le programme lunaire Apollo. Et depuis, la course du silicium n'a jamais cessé, surtout à partir du premier véritable microprocesseur (1971). Finalement, en 40 ans, la base de l'informatique n'a pas changé : le silicium. Ce qui change, c'est la miniaturisation, l'apparition de nouveaux matériaux, la multiplication des transistors, des CPU, des GPU, etc. Mais à un moment, nous arriverons à la limite des possibilités physiques du silicium et de l'intégration toujours plus dense et plus fine. Même si nous avons contourné les limites de la loi de Moore sur le doublement de la puissance processeur, la physique rattrapera immanquablement les ingénieurs. Et ceux-ci vont tenter de biaiser la limite comme nous l'avons vu il y a peu sur les transistors dits tridimensionnels.

Vers une autre informatique

Peut-on réellement dépasser le silicium et trouver une autre voie ? La réponse est oui. La physique elle-même a fourni une piste hasardeuse mais possible : l'informatique quantique reprenant les fondamentaux de la physique quantique. Nous

geek !

tre inPulse... votre climatiseur ! en direct des labos.

avions fait un reportage sur ce sujet il y a presque 10 ans dans Programmez!. La science du vivant est sans doute une des pistes les plus prometteuses. C'est ce que l'on appelle la bioinformatique ou biotechnologie (au sens large). Mais ce terme recouvre diverses réalités : s'inspirer de la nature, intégrer des éléments du vivant dans l'informatique, s'inspirer de l'Homme et de ses neurones, informatique ADN. Les possibilités sont aussi vastes que la nature elle-même. Est-ce utopique ? De la science fiction ? Pas du tout, la bioinformatique existe. Au moins dans la tête des chercheurs et dans les laboratoires. ADN, neurones, nanotechnologies, nous avons finalement l'embarras du choix. Mais, il y a fossé entre la théorique du laboratoire et son utilisation par le grand public. Et ensuite c'est toute la manière de penser l'informatique, de la programmer qui doit être refondée. Nos langages et outils ne sont plus adaptés. D'ailleurs, ils trouvent d'ores et déjà leurs limites dans le massivement parallèle ou encore l'informatique prouvée et prédictible, sans oublier, notre incapacité à concevoir une programmation accessible à tout le monde. Bref la programmation naturelle que l'on nous promet depuis longtemps.

Dans ce grand dossier d'été, nous allons aborder plusieurs pistes disponibles ou cantonnées aux laboratoires : robotique, système enfoui et embarqué, intelligence ambiante ou encore bioinformatique. Le thème est immense et passionnant. Et surtout, rien n'est immuable. Par exemple, les théories de Einstein sur l'univers et son expansion sont aujourd'hui largement remises en cause par les récentes découvertes, notamment sur l'absence (ou du moins sa non-découverte) de la matière noire...

Bonne découverte !

■ François Tonic

Exosquelette : la fin de l'accessibilité ?

Nous sommes habitués aux titres provocateurs. Mais le sujet de l'exosquelette mérite que l'on s'y attarde et que l'on tente d'en comprendre la portée, l'intérêt et même son avenir. La science fiction s'est emparée du sujet depuis fort longtemps. Mais nous oublions vite qu'il existe des exosquelettes pour l'Homme et qu'ils fonctionnent ! Nous pourrions résumer cette technologique particulière à cette formulation : *« lève-toi et marche ! »*.

Qu'est-ce qu'un Exosquelette ?

Un exosquelette ou squelette externe, par opposition à endosquelette (être humain), est une caractéristique anatomique externe qui supporte et protège un animal (fourmi, crabe…).

Beaucoup d'invertébrés, comme les insectes, les crustacés et les mollusques, possèdent un exosquelette. La partie abdominale d'un exosquelette est communément appelée « carapace ». Des recherches technico-scientifiques développent actuellement des exosquelettes biomécaniques ou motorisés pour des besoins militaires, mais aussi médicaux ou industriels (support du corps, décupler la force humaine, …). Ce sont des versions modernes et techniques des armures des chevaliers du Moyen Âge, lorsqu'ils enveloppent des êtres humains; ce sont aussi les peaux des robots humanoïdes.

Source : http://fr.wikipedia.org/wiki/Exosquelette

Quel est le stade d'évolution actuel des exosquelettes ?

Comme mentionné ci-dessus, les exosquelettes se développent massivement depuis quelques années mais le public ne s'en rend pas forcément compte. Ils ciblent de très nombreux domaines en y apportant qualité de support, renforcement des capacités d'un individu, augmentation de la force humaine, etc. Vous pouvez imaginer les plus grands « délires ». Regardez Iron Man et vous comprendrez parfaitement ce que peut faire, ou pourrait faire, un exosquelette. Certes, nous sommes encore loin du concept d'Iron Man mais d'ores et déjà, la technologie actuelle de l'exosquelette apporte des réponses concrètes et technologiques à des problèmes réels.

Dans le monde, il existe des sociétés fabriquant des exosquelettes, l'une des plus connues est la firme japonaise Cyberdyne (une venture du Prof. Sankai et de son laboratoire à l'Université de Tsukuba). La société a créé le Hybrid Assistive Limb ou HAL. Le premier modèle remonte à 2008. Et l'ambition est grande : produire 500 unités de HAL par an !

Les capacités et l'ergonomie du HAL sont assez étonnantes et répondent à différents besoins. Nous vous invitons à regarder les différentes vidéos proposées en ressource. Au-delà de HAL complet comprenant l'ensemble du corps (jambes, hanches, bras, torse), il existe une version légère « Robot Suit HAL for Well-being ». Ce modèle doit permettre d'aider les personnes à mobilité réduite (ou ayant du mal à marcher) à pouvoir se déplacer. L'exosquelette soutient les jambes (et donc supporte le poids du corps) et aide la personne à se mouvoir. Terminons notre propos avec le Berkeley Bionics et le eLegs. A l'instar du Robot Suit HAL for Well-being, eLegs permet de faciliter la marche à des personnes handicapées par exemple après un accident. Cette société a développé plusieurs modèles et travaille activement à proposer des modèles performants. Une autre solution de la resocialisation ?

Quel(s) avenir(s) pour les exosquelettes ?

Alors, convaincu, ou vous faut-il d'autres pistes d'avenir pour ces squelettes technologiques ? Laissons votre imagination et votre esprit comprendre le bouleversement que ces exosquelettes peuvent impliquer sur nos sociétés et certains métiers. Ils ont clairement pris place dans notre ère technologique et vont maintenant évoluer

étape par étape vers une mondialisation à moyen ou long terme. Bien entendu, un modèle « standard » revient entre 4 et 12 000 euros. Le HAL est proposé à la location au Japon à plus de 1 000 euros par mois.

« tentez l'exercice financier en comparant les coûts d'infrastructure dédiés à l'accessibilité (rien que dans les lieux publics) et la couverture sociale d'un tel outil pour une personne en fauteuil roulant ... sans même prendre en compte la notion de confort pour une personne à mobilité réduite de se retrouver à même hauteur que le reste de la population ... ». Source : http://www.dailymail.co.uk/sciencetech/article-1071511/The-robot-suit-rent-help-walk-1-260-month.html

Parlons des autres applications : une infirmière devant porter des patients potentiellement 2x plus lourds qu'elle, les métiers lourds du domaine de l'industrie (actuellement nous parlons des métiers pénibles, en France, face à la notion de retraite) ...Alors, les exosquelettes ? une piste réelle ? Et que dire des applications militaires ? Je vous laisse imaginer dans quelques années lorsque ces exosquelettes de renfort seront encore plus fluides dans leurs mouvements (support de charges considérables, avantage dans les combats à mains nues pour les militaires équipés de tels outils, résistance humaine, ...). Ne tombons pas dans l'excès Iron Man. Cependant, comme vous l'aurez constaté, nous n'en sommes pas si loin. La meilleure manière de prédire le futur ne serait-elle pas de l'inventer tout simplement !!! Merci à ces ingénieurs et entrepreneurs de génie pour cette invention singulière !

■ Grégory **Renard**
CIO Wygwam - Research & Innovation Manager, Architecte d'Innovation xBrainLab
Wygwam, xBrainLab - marques du Groupe Usilink
Microsoft Regional Director - Microsoft MVP (Most Valuable Professional) Windows Azure

L'exosquelette va-t-il aider les handicapés ?

Ressources

Présentation de HAL par son concepteur le Prof. Sankai
- http://www.youtube.com/watch?v=ynL8BCXih8U

Applications Métiers lourds :
- http://www.youtube.com/watch?v=oLM5MNnrpJs&feature=fvw

Applications Accessibilité – Walking Assist ou ReWalk
- http://www.youtube.com/watch?v=WcM0ruq28dc
- http://www.youtube.com/watch?v=pp4XUvgqkbU&feature=fvw
- http://www.youtube.com/watch?v=gQRQs-N-ZIM
- http://www.youtube.com/watch?v=424UCSN3Fjg

Applications Militaires :
- http://www.youtube.com/watch?v=0hkCcoenLW4
- http://www.youtube.com/watch?v=KZ_qR8zCLDc
- http://www.youtube.com/gregoryrenard#p/c/DC37034CF395C6BF/7/fRkg6H0ZP8A

Applications Grand Public :
- http://www.youtube.com/watch?v=B_k30yeCk4c&feature=related

La montre programmable inPulse

L'heure, information si utile… que l'homme depuis la nuit des temps continue d'innover pour trouver de nouvelles manières de la présenter… Dans cet article nous allons vous faire découvrir une montre atypique, la montre inPulse (http://getinpulse.com). Si cette montre est capable de donner l'heure, elle ne va pas le faire toute seule : il faut lui dire comment faire, car elle est programmable. Partons à la découverte de cet objet qui risque de booster sérieusement votre geek-factor…

Le cœur de la montre inPulse est un petit micro-contrôleur ARM7 cadencé à 52MHz, disposant de 32KB de mémoire pour le code, et de 8KB de RAM. L'écran de 1,3" a une résolution de 96x128 pixels. La montre dispose d'un vibreur, d'un bouton et d'une petite batterie Lithium-Ion de 150mAh… petite batterie qui se charge avec un câble micro-usb et qui permettrait de tenir 4 jours d'affilée, selon votre utilisation… en tous cas 24h sans problème. Dernier point, et pas des moindres, la montre dispose d'un chipset bluetooth compatible L2CAP qui permettra donc d'échanger de l'information avec des équipements comme des PC ou des smartphones. C'est également par cette connexion Bluetooth que se fera la programmation de la montre.

LE KIT DE DÉVELOPPEMENT

Maintenant qu'on connaît les capacités matérielles, attaquons-nous à la programmation. Il faut pour cela télécharger le kit de développement sur le site http://www.getinpulse.com/guide/. Il existe des versions de ce kit pour Mac, Linux, et PC, et on peut même utiliser un simulateur, pour préparer son code en attendant de recevoir sa montre par exemple ! La documentation du SDK est disponible sur http://www.getinpulse.com/pulsesdk et nous explique les API qui existent, pour ceux qui n'aiment pas creuser directement dans les header files du SDK :-p. Nous allons passer la prochaine partie de cet article à étudier les plus importantes.

Les API

Les API de programmation de la montre sont d'assez bas niveau. On peut manipuler les pixels, des polices de caractères et des images. Il n'y a pas de notion de contrôles pour l'interface graphique. Nous allons à travers quelques exercices simples essayer de comprendre comment fonctionnent ces API.

Récupérer l'heure et gérer le temps

Oui, il peut être utile de commencer par cela ! l'API est simple et nous permet de remplir une structure de données avec l'heure courante :

```
struct pulse_time_tm now;
pulse_get_time_date(&now);
```

Au passage, cela me permet de préciser que la montre est mise à l'heure lorsqu'elle est flashée, ou lorsqu'on lui envoie un message avec un petit utilitaire du SDK. Assurez-vous donc que votre PC est à l'heure que vous voulez voir apparaître sur la montre ! Il est également possible de piloter des timers, et des alarmes. La différence étant que le timer est interne à l'application alors que l'alarme est persistante même après un reset de la montre. Voici un exemple de timer qui permet de détecter un appui long sur le bouton de la montre :

```
int32_t button_timer;

void pulse_button_long_down(int arg)
{
  // action pour un appui long
}

void main_app_handle_button_down()
{
  button_timer = pulse_register_timer(1000, &pulse_button_long
_down, 0);
  // action pour appui court
}

void main_app_handle_button_up()
{
  if(button_timer)
    pulse_cancel_timer(&button_timer);
}
```

Dessiner un rectangle

Pour dessiner un rectangle, il faut commencer par initialiser la surface de dessin à l'équivalent de la surface du rectangle. Ensuite, une simple *boucle for* nous permettra de parcourir les pixels de cette surface initialisée, et de leur assigner une couleur :

```
color24_t color = { 0x44, 0x99, 0x11, 0x00 };

pulse_set_draw_window(0, 10, 20, 30);

for (int i = 0; i < 20 * 20; i++) {
```

```
  pulse_draw_point24(color);
}
```

Afficher un message

Le moyen le plus simple mais aussi le moins élégant est d'utiliser l'instruction *printf* pour afficher un message. Ceci étant dit, cela ne fera qu'utiliser la police de caractères du système, sans formatage. En revanche, il est aussi possible d'utiliser une API qui permet de définir une zone de texte dynamique, et d'y associer une police de caractères. On a donc un contrôle beaucoup plus fin sur l'endroit, la manière et le texte affiché. Cette opération se passe en plusieurs parties : d'abord, on charge la police de caractères, et on configure la zone dans laquelle on va écrire le texte, ensuite, on initialise le widget de texte, et enfin on affiche le texte. Voici un exemple qui permet d'afficher l'heure avec la police de caractères Clockopia :

```
struct PWTextBox clock_text_box;
struct PWidgetTextDynamic clock_text_widget;
enum PWTextStyle clock_text_style = (PWTS_TRUNCATE | PWTS_
CENTER);
char clock_text_buffer[16];

clock_text_box.top = 0;
clock_text_box.bottom = pulse_get_font_height(FONT_CLOCKOPIA_30);
clock_text_box.left = 0;
clock_text_box.right = SCREEN_WIDTH - 5;

struct pulse_time_tm now;
pulse_get_time_date(&now);

if(!pulse_font_valid(FONT_CLOCKOPIA_30))
printf(«invalid font!\n»);

pulse_init_dynamic_text_widget(&clock_text_widget, clock_text
_buffer, FONT_CLOCKOPIA_30, COLOR_WHITE24, clock_text_style);

sprintf(clock_text_buffer, «%d:%02d», now.tm_hour, now.tm_min);
pulse_render_text(&clock_text_box, &clock_text_widget);
```

Afficher une image

L'affichage d'une image peut aussi permettre d'enrichir l'interface : par exemple en utilisant des images et un système de sprites codé à la main, un hacker a réussi à faire une horloge mondiale qui représente la terre, en 3D, avec les zones éclairées par le soleil et l'heure de la zone correspondante en dessous ! L'API permet aussi de gérer la transparence, si besoin.

```
if(pulse_image_valid(IMAGE_MSGLIST_EMAIL_GLOW))
{
  // use pulse_draw_image_with_transparency() if necessary
  pulse_draw_image(IMAGE_MSGLIST_EMAIL_GLOW, 0,0);
}
else
{
  printf(«bad image\n»);
}
```

La compilation et le téléchargement du code sur la montre

Le plus simple pour compiler un programme sur la montre est d'utiliser le service de cloud-compilation d'Allerta, qui est utilisable à partir d'un petit script python. Il est également possible de cross-compiler le code directement sur son PC mais c'est tout de suite plus compliqué à installer…

Les ressources (images, polices de caractères) se compilent et se chargent indépendamment du code. Il ne faut donc pas oublier de s'occuper de celles-ci avant de s'occuper d'uploader le code !

Pour formater les ressources, il suffit de placer tous les fichiers de polices de caractères truetype et les images (au format bitmap) dans le dossier resource à la racine du SDK et d'appeler le script resource_packer.py.

```
pierre@vaio:~/inpulse/pulse_simulator$ ls resources/
fonts.txt  msglist_email_glow.bmp  readme.txt
pierre@vaio:~/inpulse/pulse_simulator$ python tools/resource
_packer/resource_packer.py
Put all BMP, PNG and TTF files in resources/
Be sure to update fonts.txt with your specific font selection
File added: resources/fonts.txt
```

```
File added: resources/readme.txt
File added: resources/msglist_email_glow.bmp

msglist_email_glow.bmp converted successfully

resouces packed successfully

Created resource pack:  /home/pierre/inpulse/pulse_simulator
/resources/inpulse_resources.bin
Created resource header:  /home/pierre/inpulse/pulse_simulator
/include/app_resources.h
```

Ensuite, avant de déployer les ressources et le code sur la montre, il faut en connaître l'adresse MAC. Pour cela, un petit scan des périphériques bluetooth environnants devrait vous l'indiquer :

```
pierre@vaio:~/inpulse/pulse_simulator$ hcitool scan
Scanning ...
  00:50:C2:xx:xx:xx  inPulse5656
```

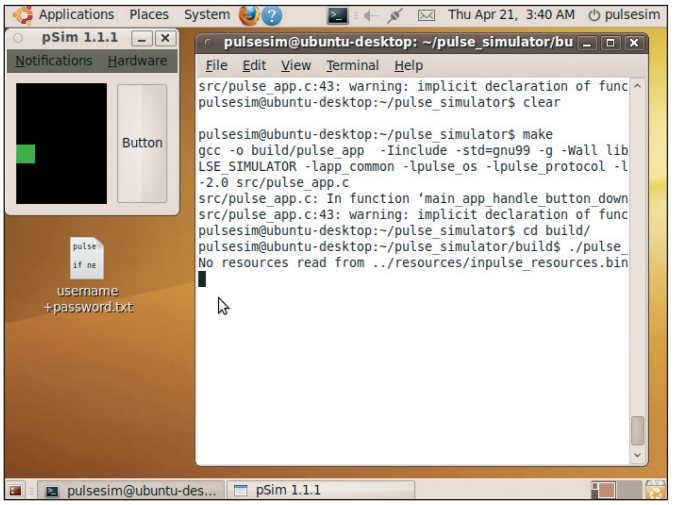
Enfin, avec le compilateur « cloud » une simple ligne de commande suffit pour charger les ressources ou le code :

```
pierre@vaio:~/inpulse/pulse_sdk$ python compileandload.py -
d 00:50:C2:xx:xx:xx
```

Le flag –d permet de compiler et charger le code dans la montre. Ajoutez –r à cette ligne pour y charger les ressources.
Le chargement prend un peu de temps.... Et il n'est pas possible de debugger directement le code sur la montre. Dans ces conditions, utiliser le simulateur peut être un confort.

Le simulateur

Le simulateur ne reproduit pas exactement les fonctionnalités de la montre, et toutes les API n'y sont pas implémentées (les plus importantes le sont). Prenez donc le temps d'étudier dans les headers ou la documentation d'API si les fonctions que vous utilisez sont supportées !
Le simulateur s'installe uniquement sous Linux... Si vous n'avez pas de machine sous Linux à disposition, un guide est fourni pour installer une machine virtuelle sous Windows sur le site http://www.getinpulse.com/simulator/.



Sous Linux le simulateur s'installe tout simplement en le dézippant, après avoir installé les dépendances (dont vous trouverez la liste sur la page de téléchargement). Ensuite, la structure des répertoires est la même que dans le SDK : les ressources sont dans le dossier resources, le code dans le dossier src, les headers dans le dossier include.
En revanche, pour la compilation c'est un peu différent : le SDK dispose d'un Makefile à sa racine. Un simple appel à la commande make compilera le code pour vous, mais attention vous devrez toujours charger les ressources séparément !
Ensuite, pour lancer le code il suffit de passer dans le répertoire build et de lancer l'application pulse_app en ligne de commande.

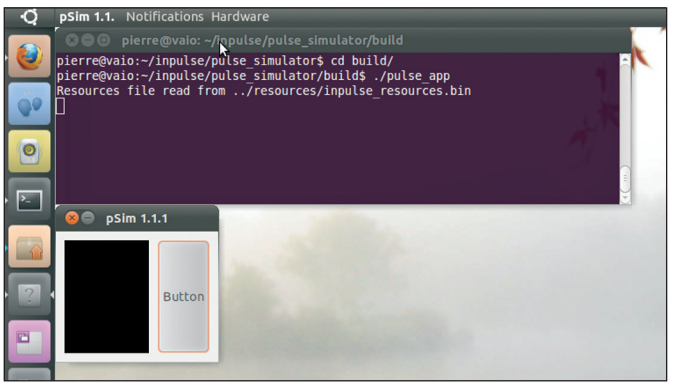
Les possibilités offertes par la connexion Bluetooth

Le framework OpenWatch permet de faire communiquer un smartphone avec une montre bluetooth, et la montre inPulse peut justement être compatible avec ce framework : il suffit d'y installer le firmware standard fourni sur le site http://getinpulse.com. Une fois ce firmware installé et avec la bonne application, par exemple OpenWatch sur l'Android market, vous pourrez envoyer des notifications à votre montre comme par exemple le numéro de l'appelant quand le téléphone sonne. Il est également possible de piloter la montre avec la librairie L2CAP, ce qui a permis par exemple à un développeur de transformer la montre en télécommande pour powerpoint.

CONCLUSION

La montre inPulse est aujourd'hui une des montres hackables les plus évoluées du marché, si ce n'est la plus avancée. L'équipe d'Allerta, l'entreprise qui conçoit la montre, est déjà en train d'explorer des nouvelles fonctionnalités, comme par exemple la possibilité d'y intégrer un accéléromètre. On pourrait aussi imaginer des versions avec plusieurs boutons, voire un écran tactile...
Dans tous les cas, le marché de la montre intelligente est voué à se développer : Fossil, un fabricant de montres bien connu, est déjà dans la course, Texas Instrument propose un kit de montre basique à quelques dizaines de dollars, et des fabricants d'équipement comme Sony-Ericsson, Blackberry ou LG y ont déjà commercialisé des produits pour accompagner leurs smartphones !
Quant à Microsoft, c'est dans le domaine de la montre intelligente (avec les montres SPOT) qu'est né le .NET MicroFramework !

■ Pierre **Cauchois**



Supplément
Spécial C++

Le magazine du développement
PROgrammez !
www.programmez.com

Toute la puissance du C++

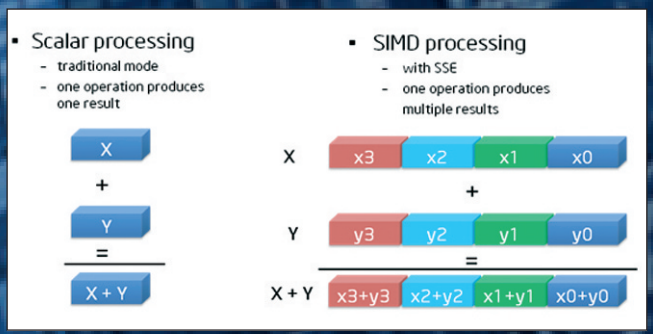
Technique
La programmation par agent

Expérience
C++ résout le Sudoku

Outils
Cilk Plus : le parallélisme facile

VTune Amplifier XE
Traquer les bugs et optimiser le code

SOMMAIRE



Experts : A. D. Robison, E. Vernié, S. Blair-Chappell, R. Geva, L. Akyll

Edito

C++ L'ultra performance !

Insubmersible, voilà comment nous pourrions définir C++ et même le C. Depuis 20 ans, combien de fois n'avons-nous pas entendu annoncer la fin du C++, son peu d'intérêt, etc. Il aurait tous les défauts, aucun avantage, comparé à des langages plus modernes comme Java, C#, VB, etc. La complexité exponentielle des systèmes, la multiplication des logiciels embarqués, les exigences de performance et de fiabilité font que C++ retrouve une nouvelle jeunesse. C++ est bel et bien de retour !

Soyons clairs, dans les systèmes embarqués, enfouis, dans les approches HPC, massivement parallèles et distribuées de type HPC, dans les performances, seul le langage C++, avec son approche native, peut répondre à ces différentes problématiques. Oui, les langages managés tels que Java et C# apportent une réponse de plus haut niveau grâce à l'utilisation de librairies, API, frameworks afin de faciliter la parallélisation du code. L'appel de code C, C++ directement de ces langages facilite l'optimisation des zones critiques pour la fiabilité, les performances.

C++ constitue la base de la programmation moderne et pour réellement maîtriser les concepts objets que l'on retrouve dans de nombreux langages, vous devez maîtriser C++. Pour les hautes performances, le multicore, la criticité des systèmes, l'embarqué, C++ demeure et demeurera la meilleure réponse, même s'il n'est pas toujours simple de trouver de bons experts Cpp, ni de se former. Mais aujourd'hui, pour aider le développeur C++, des outils existent. Intel complète ses gammes année après année : compilateur, profiling / tuning, librairies optimisées. La gamme Parallel Studio facilite la parallélisation du code. Cilk Plus va encore plus loin en apportant une réponse élégante et efficace au multicore.

Tout au long de ce cahier spécial, nous allons explorer plusieurs techniques de développement C++ et découvrir la mise en place d'outils tels que Cilk Plus ou encore VTune Amplifier XE. Et vous verrez, qu'avec le profiling de code, on peut aller très loin et traquer la non-performance jusqu'au cœur des processeurs et des registres !

Enjoy #include

François Tonic
Rédacteur en chef Programmez !

C++ au service des utilisateurs et de l'interface

L'index Tiobe Software sur l'utilisation des langages donne un baromètre très intéressant. En juin 2011, Tiobe classait C et C++ en 2e et 3e place avec respectivement 16,2 % et 9,8 % d'utilisation par les développeurs, soit plus de 25 % en les cumulant, dépassant largement Java qui atteint 18,5 %. Cette popularité n'est pas anodine et correspond à une vraie tendance et non à une simple mode. On pourrait même y ajouter Objective-C, un dérivé du C, en 7e place avec 4,4 %.

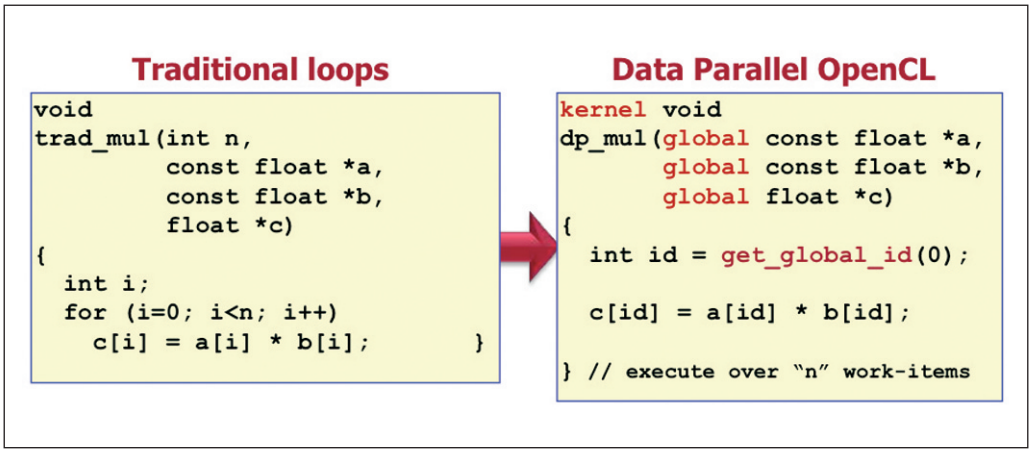
C et C++ sont deux langages répondant parfaitement aux exigences de l'expérience utilisateur, à l'interface graphique. Ces langages prennent leur sens dans des projets multiplateformes, ou tout simplement sur certaines plateformes mobiles. Ainsi, le système iOS se programme avec Objective-C, langage historique de MacOS X, dérivé du C. Bada, le système de Samsung, se code en C++. Et même si des designers graphiques facilitent la construction des applications, cela reste du C et C++. On peut même coder les interfaces et interactions tactiles en C et C++. C'est le cas par défaut sur iOS. Sous Windows, Windows Touch se programme aussi bien en C# qu'en C++. Bien entendu, tout le monde ne dispose pas d'interfaces C++. Android et Windows Phone 7 ne proposent pas de modèle de développement C, C++.

Qt = C++ = portabilité optimale

Une des références actuelles dans l'interface graphique en C++ est la librairie Qt. Disponible sur desktop et mobile, Qt permet de concevoir rapidement de puissantes interfaces graphiques grâce à des designers / IDE. La force de Qt est dans la richesse du framework et sa disponibilité sur de nombreuses plateformes.
Nokia l'utilise pour Symbian et le système Meego repose aussi sur Qt / C++ pour l'interface. Il existe d'autres frameworks d'interface : wxWidgets, FLTK.
N'oubliez pas non plus que les moteurs 3D s'utilisent le plus souvent en C++, question de performances et de nativité du code.

OpenCL : sous le capot, un tigre !

L'utilisateur ne voit pas ce qu'il se passe réellement sous son interface utilisateur. Aujourd'hui, avec la multiplication des cœurs, des GPU, les architectures se complexifient et il faut sans cesse optimiser le code.
Pour concilier, sans tout réécrire, CPU – GPU et utiliser au mieux les ressources matérielles, nous disposons d'OpenCL.
Il s'agit d'un modèle de développement ouvert pour la programmation parallèle en environnement hétérogène.



Exemple d'une implémentation.

Cette approche est idéale pour les jeux, le HPC. Avec OpenCL, le développeur écrit un code et OpenCL s'occupe d'utiliser les ressources disponibles : CPU, GPU, DSP, etc. Cette approche évite de coder, d'optimiser pour chaque ressource.
Le développeur doit en comprendre la logique et intégrer le framework mais les gains de performances méritent les efforts, même si cette technologie n'est pas généralisée partout. Intel propose son kit de développement : Intel OpenCL SDK pour Windows 7 et Linux.

Tout n'a pas forcément besoin d'être codé en pur natif, vous pouvez parfaitement faire le « core code » en langage managé et des portions très précises, des fonctions en natif. D'autre part, maîtriser le C++, c'est mieux maîtriser la programmation et ses subtilités...

Question : avec quel langage sont codés les systèmes d'exploitation ?
En C. Alors oui, C++ a presque 30 ans d'existence mais il n'a jamais été aussi dynamique et jeune ! Et n'oubliez pas que ce langage possède une normalisation ISO (C++ ISO) sur le langage (Core Language) et la librairie standard (ou C++ standard Library).
Et C++ continue d'évoluer. Comme nous allons le voir dans ce cahier spécial, le développeur dispose aujourd'hui de puissantes extensions au langage, notamment pour le parallélisme.

Et, une future norme est actuellement en préparation : C++0x (ou plutôt C++1x).
En avril dernier, le document final des spécifications a été approuvé, ouvrant la voie à la nouvelle norme C++ ! Le langage C aura droit lui aussi à sa nouvelle version.

François Tonic

C++ soyez le maître de la jungle

Dans la jungle des langages de programmation, C++ peut sembler une bête complexe. Pourquoi devriez-vous apprendre ce langage, outre le défi de l'apprendre ? C++ représente un défi, car il est puissant et pratique. Ses principes sont simples même si une partie de sa complexité vient de son évolution, très réussie, depuis un quart de siècle. Vous pouvez l'utiliser sans avoir besoin de connaître ses évolutions passées. Cet article n'est pas destiné à vous apprendre C++, mais plutôt à vous mettre en appétit avec quelques clés qui ont fait son succès.

D.R.

L'APPLICABILITÉ

Aucun langage ne peut tout proposer, même si C++ offre une utilisation très large. Voici quatre de ses points forts :

1- C++ s'utilise sur une très grande variété de matériels. Les environnements de développement existent sur des machines de toute taille, du plus petit système embarqué aux monstres de calculs !

2- C++ s'applique aux tâches, aux fonctions les plus diverses. Le langage permet de faire de la programmation système, des librairies, aussi bien que de l'interface graphique, des analyses numériques ! Il est particulièrement adapté aux problèmes de performances ou aux grandes capacités mémoires. Lorsque ces facteurs ne sont pas critiques, par exemple dans la programmation web, des langages plus dynamiques, comme Java, C#, Python..., peuvent être plus pratiques. Mais lorsque vous avez besoin d'exécuter du code sur des systèmes enfouis, de hautes performances, de manipuler d'importantes volumétries de données, ces langages ne peuvent rivaliser avec le C++ !

3- le C++ va d'un (très) bas niveau à un niveau élevé de programmation. La clé du succès en C++ est de fournir une abstraction avec un minimum de contraintes. Sur le bas niveau, les sous-ensembles C permettent un accès

efficace et fiable aux ressources matérielles. Par exemple, il fait une abstraction des adresses machines comme les pointeurs et les références. Sur le haut niveau, il fournit des mécanismes pour implémenter sa propre abstraction. Par exemple, les classes permettent de définir de nouveaux types. Ou encore, l'usage des templates.

Vous pouvez réutiliser des librairies hautement indépendantes tout en minimisant le temps d'exécution. Vous pouvez apprendre le C++ en commençant par les sous-ensembles C ou apprendre le langage en commençant par les conteneurs, les algorithmes du Standard Template Library (STL). Si vous êtes familier avec le C, vous pouvez apprendre comment les classes sont des extensions (directes) de la structure C et C++ et comment les templates C++ sont propres et une alternative puissante aux macros. Surtout, vous n'avez pas besoin d'apprendre tout le langage. Beaucoup de développeurs apprennent des éléments clés, des points très précis du langage, les éléments qui leur sont utiles !

4- une abondance de code réutilisable ! Cette riche base de données permet de vous focaliser sur les nouveaux défis au lieu de toujours réinventer la roue ! En outre, vous pouvez utiliser facilement des librairies C en C++. Car C++ est omniprésent et de nombreux langages interopèrent avec lui !

Divers modèles de programmation

Certains langages forcent le modèle de développement. Par exemple, Java force l'orienté objet que vous le vouliez ou non. C++ encourage différents modèles de programmation et l'orienté objet en est un parmi d'autres. En effet, la norme ISO C++98 fait 776 pages mais ne mentionne l'orienté objet qu'une seule fois ! Ainsi, vous pouvez utiliser le C++ comme un langage procédural ou comme un « meilleur C » sur la surcharge des opérateurs, l'encapsulation via les déclarations private dans un struct. Une classe C++ n'est qu'un struct avec un accès au private. Ou vous pouvez aussi continuer à utiliser des variables globales et des goto en dépit de la déclaration de certains fanatiques de la programmation ! Et vous n'avez pas à abandonner vos codes C ! Vous pouvez convertir le code C en C++. Il est aussi facile d'appeler des librairies C en C++.

Vous pouvez utiliser C++ comme un langage orienté objet. Quand l'objet a débuté, il avait une réputation de faible performance, et de difficulté à compiler avec des langages comme Simula et Smalltalk. C++ a changé la donne. Il supporte à la fois l'héritage unique et multiple. Et l'usage d'une fonction virtuelle est à peine plus coûteuse qu'un simple appel de fonction.

Un autre modèle de développement est la programmation générique. En effet, le faible impact de l'abstraction a permis à la programmation générique de réellement décoller ! Dans ce modèle, vous pouvez implémenter des algorithmes de la manière la plus générique possible. Par exemple, écrire une routine pour le tri d'un tableau de chaînes dans l'ordre croissant. C'est assez spécifique car limité à des tableaux, et le tri se fait dans une seule direction. Vous pouvez utiliser le C++ STL header « algorithm ». C'est générique et efficace !

Un autre paradigme de développement C++ est la métaprogrammation par patron (template metaprogramming), le code s'exécute lors de la compilation. Des bibliothèques complexes utilisent cette approche pour optimiser les « décisions ». Par exemple, un nombre entier N connu au moment de la compilation, un métaprogramme peut calculer la factorielle de N. Autre approche de cette métaprogrammation par template, l'expression templates : une librairie pour faire une optimisation de haut niveau lors de la compilation. Autre exemple, la librairie « fusion » dans Boost. Elle utilise une combinaison metaprogrammation en compilation et une approche runtime pour être plus efficace en conteneurs hétérogènes. Pour en savoir plus : http://www.boost.org/doc/libs/1_46_0/libs/fusion/doc/html/fusion/

Outre les performances pures, la métaprogrammation permet aux bibliothèques de (bien faire) avancer les vérifications lors de la compilation, voir : http://www.boost.org/doc/libs/1_46_0/libs/mpl/doc/tutorial/dimensional-analysis.html.

Toutes ces méthodes résument bien la philosophie intrinsèque de C++ : faites confiance aux développeurs. Certes, des fonctions, subtilités peuvent être mal comprises, mal utilisées mais si vous vous en servez judicieusement, quelle puissance sous le capot ! Et vous êtes libre de les utiliser, ou non, à votre gré.

Longévité

La longévité est un autre avantage (énorme) de C++ ! Après plus de 25 ans d'usage, le langage continue à croître et à s'adapter. Ainsi, la prochaine génération de la norme C++, alias C++0x, fournit un support pour l'écriture multithread, si cruciale aujourd'hui en contexte multicore. Elle inclut aussi les expressions lambda, simplifiant les routines génériques. Et bien que non officielle, cette évolution est présente (partiellement au moins) dans les dernières versions des compilateurs GNU, Intel et Microsoft.

Avec les différentes évolutions, il y a des éléments qui peuvent paraître étranges ou redondants, mais ils sont vitaux pour l'exécution et la compatibilité avec du code vieux de 10, 15, 20 ans. Par exemple, C++ hérite des déclarations « static » du C. Un moyen pour limiter la visibilité d'un symbole dans un seul fichier source. C++ introduit d'autres solutions pour limiter cette visibilité. Cependant, il existe des bizarreries incontournables. Dans les expressions, un tableau se désintègre quand un pointeur va vers le premier élément. La syntaxe déclarative abstraite peut être confuse pour les nouveaux développeurs mais comme pour la conjugaison, elle devient intuitive avec l'expérience.

Pour mieux connaître l'histoire du C++ : Bjarne Stroustrup, « Design and evolution of C++ ».

PERFORMANCE

C++ est fait pour les performances sur un socle matériel commun. Nous avons évoqué plus haut, l'abstraction et son impact limité. Un élément clé de cela est le principe du "pay as you go". Vous ne payez que pour ce que vous utilisez. Dans d'autres langages, vous risquez de payer plus cher l'abstraction. Ainsi, les fonctions virtuelles sont souvent utilisées pour ajouter des "extra pointer" à un objet pointant vers une table. Or, un objet peut avoir quelques bytes, mais plusieurs millions d'objets, bref, vous mesurez l'importance des pointeurs...

C++ permet de choisir des compromis entre la taille du code et la rapidité, spécialement dans le polymorphisme runtime et le polymorphisme compilateur. Pour résumer, si la taille du code est pour vous une priorité, écrivez les algorithmes pour fonctionner sur des objets abstraits dont le comportement des dépendances de type est encapsulé dans des fonctions virtuelles. Ainsi l'algorithme sera compilé d'un seul bloc de code appelant les fonctions virtuelles. Par contre si la performance est votre souci, la problématique sera toute autre. Vous pourrez utiliser un template pour écrire vos algorithmes+.

Remerciements

Pablo Halpern, Anton Potapov, Andrey Marochko. Traduit et adapté de l'anglais par François Tonic (Programmez !). Merci à Ralph de Wargny (Intel).

Arch D. Robison
Utilisateur de C++ depuis 1988, architecte d'Intel Threading Building Block. Développe en C++ des jeux.

microsigma

Boostez les performances de votre code C/C++ série ou parallèle !

Partenaire Elite Intel® Software et spécialiste des outils de développement depuis 1984,
Micro Sigma vous accompagne dans la mise en œuvre des outils Intel® pour la performance et l'optimisation des applications C++ sur multicore.

Conseil et choix des outils Intel pour augmenter les performances :

Les compilateurs et bibliothèques sont disponibles à l'unité ou en kit complet à un prix réduit

	SUITES		COMPILATEURS			ANALYSE	
	Intel® Parallel Studio XE 2011	Intel® C++ Studio XE 2011	Intel® Composer XE 2011	Intel® C++ Composer XE 2011	Intel® Fortran Composer XE 2011	Intel® VTune™ Amplifier XE 2011	Intel® Inspector XE 2011
Compilateurs Intel® XE	✓		✓				
> Intel® C++ Composer XE	✓	✓	✓	✓			
> Intel® Fortran Composer XE	✓		✓		✓		
> Intel® Integrated Performance Primitives	✓	✓	✓	✓			
> Intel® Parallel Building Blocks	✓	✓	✓	✓			
> Intel® Math Kernel Library	✓	✓	✓	✓	✓		
Intel® VTune™ Amplifier XE	✓	✓				✓	
Intel® Inspector XE (Thread Checker)	✓	✓					✓

Formations Programmation Parallèle et Optimisation :

La solution pour une prise en main rapide des outils Intel

micro sigma a conçu avec Intel une gamme de formations exclusives pour les développeurs C/C++ ou Fortran qui souhaitent améliorer les performances et introduire de la parallélisation avec les compilateurs, bibliothèques analyseurs, outils de vérification et profileurs de performances Intel. Formation pratique alternant exposés théoriques et réalisations d'exemples sur plateformes multicore .

Formations inter-entreprises une session par trimestre sur Paris

Formations sur mesure intra-entreprise sur votre environnement de développement

Consulting , audit de l'existant conseil et mise en oeuvre des outils de développement Intel

Renseignez vous au 0 810 120 240 ou info@microsigma.fr

http://www.microsigma.fr/intel

microsigma

Plateau technique pour tester votre code avec les outils Intel :

Prenez rendez-vous avec un Ingénieur spécialiste des outils pour évaluer les gains de performance possibles sur les nouveaux processeurs Intel.

Après ce rendez-vous qui peut aussi être proposé dans vos locaux vous pourrez évaluer les différentes options pour adapter votre code et bénéficier des dernières évolutions technologiques.

Licences avec support et maintenance 1 an inclus:

Livraison électronique des clés de licences par email (ESD)

Les licences sont perpétuelles (sans limite de temps) , tarif Commercial ou Academic pour l'éducation
À l'issue du contrat initial d'un an, le client peut le renouveler pour continuer à bénéficier du support technique et des mises à jour en ligne sur le site Intel® Premier Support.

Licence mono-poste pour un utilisateur nommé.

Licences flottantes pour 2 ou 5 utilisateurs en réseau.

Licences spécifique pour chaque environnement Windows*, Linux ou Mac OSX

Acquisition en volume avec Volume License Program:

Pour vos achats en quantité, à partir de 10 licences, vous bénéficiez d'un meilleur prix et de services sur mesure.

Micro Sigma vous assiste de l'initialisation du contrat Intel VLP au suivi des acquisitions, gestion des utilisateurs et du parc de licence.

Tarif dégressif en fonction du nombre de licences.

Simplification du suivi des dates d'échéances des maintenances.

Portail de gestion pour le suivi des licences et la gestion des utilisateurs.

intel Software Elite Reseller

Renseignez vous au 0 810 120 240 ou info@microsigma.fr

http://www.microsigma.fr/intel

La programmation par Agent

Avec l'avènement des multi-cœurs sur nos PC de bureau, portables et maintenant mobiles, il est difficile de passer à côté de toutes les innovations en termes de librairies et outils pour développer sur ce type d'architecture. Intel, avec le couple Threading Building Blocks (TBB)/Intel Parallel Studio et Microsoft, avec le couple Parallel Pattern Library (PPL)/Visual Studio 2010, fournissent avec ces deux librairies (TBB et PPL), des services de haut niveau, qui permettent à un développeur C++, de s'affranchir de certaines contraintes inhérentes à la programmation Parallèle.

Nous y retrouvons des algorithmes pour mettre en place des boucles parallèles sans trop d'efforts (parallel_for, parallel_foreach), des primitives de synchronisations (reader_writer, section critique et autres) qui évitent de bloquer inutilement le processeur, et des conteneurs concurrents, (Concurrent_vector, concurrent_queue, etc..) qui prennent en charge, entre autres, les problèmes de "false sharing" et effet "ping-pong" qui pourraient survenir au niveau des caches processeurs.
Néanmoins, en programmation parallèle, un nouveau paradigme de développement émerge, la programmation par Agent qui facilite le développement asynchrone, et la mise en place du modèle dit Pipeline. Ce modèle de développement est pris en charge dans Visual Studio 2010, par la librairie Asynchronous Agent Library.

POURQUOI LES AGENTS ?

En programmation parallèle, l'un des problèmes récurrent et non déterministe qui survient, est le partage d'états entre différents threads. Ce partage d'états, s'il n'est pas bien maîtrisé, engendre systématiquement des problèmes de concurrences d'accès. C'est-à-dire qu'une variable peut, à tout moment, être dans un état incohérent, et donc générer soit des résultats erronés soit, dans le pire des cas, une instabilité de l'application.
Développer avec les agents, va permettre d'outrepasser ce problème. En effet, une donnée ou un état sera isolé d'un agent par rapport à l'autre, par l'intermédiaire d'un message (ou bloc asynchrone) qui sera véhiculé entre chaque agent. Un agent envoie (**send**) le message contenant les informations, l'agent suivant reçoit (**receive**) le message, le traite, et envoie un autre message de même nature à l'agent suivant dans le pipeline, et ainsi de suite [Fig.1]. La méthode **receive** bloque le thread entrant, tant que le message n'est pas reçu. Il n'y a donc pas besoin de mettre en place des mécanismes complexes de synchronisation type **Event** par exemple, pour que le message soit transmis correctement d'agent en agent. On peut alors se focaliser sur le fonctionnel de l'agent, plutôt que sur de la plomberie multithreads toujours sujette à erreur.
Les agents permettent de développer des applications parallèles dites à gros grain, et sont basés sur des templates. Pour implémenter un agent il faut procéder comme suit :

- Il faut inclure l'entête **agent.h** et utiliser l'espace de nom **Concurrency**

```
#include <agents.h>
using namespace ::Concurrency;
```

- Dériver sa classe de la classe **agent**

```
class MonAgent : public agent
```

- Ensuite dans le constructeur, il faut définir un point d'entrée pour envoyer le message avec la méthode **send()** et un point d'entrée pour recevoir le message avec la méthode **receive()**.

```
explicit MonAgent(ITarget<RawData>& messageEnvoye,ISource<Raw
Data>& messageRecu);
```

Puis il faut implémenter la méthode **run()**. Méthode qui sera appelée directement par la plate-forme agent lorsque l'agent démarrera

```
void MonAgent::run()
{
 do
 {
   //Le thread bloque ici en attente d'un message
   _data=receive(_messageRecu)
   //Télécharger les données
   TraitementDesDonnees(_data) ;
   //Envoi du message à l'agent suivant dans le pipeline
   send(_messageEnvoye,_data);

 }
 while(_data.Sentinel == FALSE)
done();
}
```

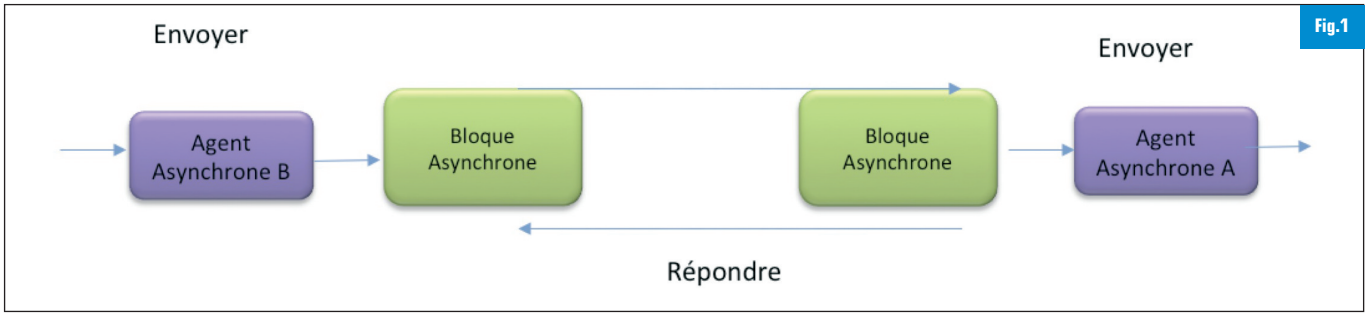


Fig.1

Dans notre exemple, l'agent ne sait pas combien de messages il va recevoir. Il entre donc dans une boucle, qui s'arrêtera lorsqu'une variable (ex **Sentinel**) est à **TRUE**, indiquant que l'agent précédent dans le pipeline a envoyé toutes les données et a fini son travail.
Une fois que toutes les données ont été reçues, l'agent met fin à son travail à l'aide de la méthode **done()**.
On notera ici l'implémentation simple et rapide du modèle **Producteur/Consommateur.**
Enfin pour démarrer l'agent et attendre qu'il ait fini :

```
unbounded_buffer <RawData> msgTargetResponse;
unbounded_buffer <RawData> msgDowloadBitmap;
MonAgent monagent(msgDowloadBitmap, msgTargetResponse)
monagent.start ();
agent::wait (&monagent);
```

unbounded_buffer<> est une classe qui représente une structure de message asynchrone capable de stocker un nombre illimité de messages. Cette classe conserve les messages dans une file d'attente en mode FIFO, garantissant que celui qui le recevra, les recevra tous et dans le bon ordre.
Vous noterez également dans notre exemple, que la classe **unbounded_buffer** prend comme paramètre le type **RawData**. C'est une structure que nous avons définie, et qui nous permettra de transmettre un état d'agent en agent. Chaque agent remplissant un champ de la structure qui lui est assigné.

```
struct RawData
{
HRESULT hr; //Contient un numéro de l'erreur sous forme de HRESULT
BYTE* BufferVignette; //Pointeur sur le tableau de bytes des vignettes
BYTE* BufferImageComplete; //Pointeur sur le tableau de bytes de l'image complète
struct ImageResult *ImageBingResult; //Pointeur sur le résultat de la recherche
ID2D1Bitmap *D2D1Thumbnail; //Pointeur Direct2D sur une vignette
ID2D1Bitmap *D2D1ImageComplete; //Pointeur Direct2D sur l'image complète
WCHAR Agent[25]; //Contient une chaine de caractère du nom de l'agent
WCHAR Message[1024]; //Message de debug
D2D1_RECT_F CurRect; //Contient la position de la vignette à l'écran
BOOL Sentinel;
//Variable qui détermine si toutes les images ont été transmises
BOOL EncoursArret;//Est-ce que l'arrêt est demandé
BOOL Sauvegarder; //Est-ce que la sauvegarde du bitmap sous forme de fichier a été faite
}user_rawdata;
```

Le champ **ImageBingResult**, sera rempli par l'agent Bing et utilisé par l'agent HTTP.
Les champs **BufferVignette** et **BufferImageComplete**, seront remplis par l'agent HTTP et utilisés par l'agent WIC
Les champs **D2D1Thumnail** et **D2D1ImageComplete**, seront remplis par l'agent WIC et utilisés par l'agent Display, et ainsi de suite.

L'AGENT MICROSOFT BING

L'agent Bing a pour rôle de retrouver les adresses électroniques des images et les fournir à l'agent suivant dans le pipeline. Avant d'aller plus loin, faisons un rappel sur les API de Bing.
Le moteur de recherche sur internet **Microsoft Bing**, vous permet de faire de la recherche sur le Web, les Images, la vidéo, les news, j'en passe et des meilleures. Comme illustré sur la figure suivante :

Avec les API Bing Version 2, vous pourrez ajouter ces fonctionnalités de recherches à votre application.
Plusieurs protocoles sont disponibles pour accéder au service, JSON, XML et SOAP, c'est ce dernier que nous utiliserons. Pour utiliser le protocole SOAP, dans notre application, il nous faut obtenir la structure du service en lui-même. Cette structure contient la définition même du service, les noms et signature de méthodes, les types de retour, etc.
La définition du service est spécifiée dans un fichier au format WSDL (Web Service Description Language) et obtenue à l'adresse http://api.bing.net/search.wsdl?AppID=APPID
Le paramètre APPID est un **global unique identifier** que vous devez créer pour votre application à l'adresse suivante : http://www.bing.com/developers/createapp.aspx. Pour de plus amples détails sur la manière d'obtenir cet APPID voir l'article *Consommer les services Microsoft Translator via les nouvelles API WWSAPI de Windows 7* dans une application Win32
Pour manipuler les services Microsoft Bing, il faut dans un premier temps définir la source de la recherche, à l'aide de la structure **SearchRequest**.
Cette structure devra être définie, entre autres, avec :
- Le numéro d'identification de l'application, l'APPID
- Le texte de la requête
- Le ou les types de sources que l'on souhaite obtenir. Différents **SourceType** sont possibles, tels que, la vidéo, les images, le web, les news, la publicité, la traduction, et autres que je vous laisse découvrir à cette adresse : http://msdn.microsoft.com/en-us/library/dd251056.aspx
- Et d'autres options, telles que **Adult** qui définit la restriction de la recherche à du contenu visible ou non.

Une fois la requête exécutée, la réponse est lisible dans la structure **SearchResponse**. Cette structure possède comme membre d'autres structures :
- Qui contiendront les données en fonction du ou des **SourceType** choisis. Nous y retrouverons les structures **WebResponse**, **ImageResponse**, **VideoResponse**, **TranslationResponse**, etc.

Chaque structure ayant ses propres membres qui ne sont pas toujours les mêmes en fonction du SourceType.
- Une structure de type **Error**, qui contiendra les éventuelles erreurs survenues lors de l'appel.

Pour télécharger le SDK de Bing : http://www.bing.com/developers/appids.aspx

CRÉATION DU PROXY BING

La création du proxy Bing se fait en deux étapes. Tout d'abord, nous utilisons l'utilitaire **svcutil.exe** ; disponible avec le SDK de Windows 7 ; en lui passant l'adresse de la description du Service Microsoft Bing

```
svcutil /t:metadata http://api.bing.net/search.wsdl?AppID=<VOTRE APPID>
```

L'outil crée en local le fichier de description du service Microsoft Bing **schemas.microsoft.com.LiveSearch.2008.03.Search.wsdl**
Ce fichier au format XML, contient toute la description du service.

La seconde étape consiste à créer à partir du fichier WSDL le code source, ainsi que les fichiers d'entêtes du proxy client que nous utiliserons dans la section suivante. Pour cela on utilise l'outil **WSUTIL** de la manière suivante :

```
wsutil.exe *.wsdl
```

wsutil, est en charge de transformer la description au format wsdl en un format natif langage C. Les deux fichiers suivants sont alors créés.
schemas.microsoft.com.LiveSearch.2008.03.Search.wsdl.h
schemas.microsoft.com.LiveSearch.2008.03.Search.wsdl.c

Remarque :
*Vous trouverez les outils **svcutil** et **wsutil** dans le répertoire C:\Program Files\Microsoft SDKs\Windows\v7.1\Bin.*

Le fichier *.H* contient toutes les définitions des méthodes, structures et énumérations que nous utiliserons.

UTILISATION DE WWSAPI

Je ne vais pas rentrer ici dans le détail des API WWSAPI. Néanmoins, voici les étapes à suivre :

- Il faut inclure l'entête de WWSAPI

```
#include <WebServices.h>
```

- Se lier à la librairie **WebServices.lib**
- Ensuite dans le code, il faut initialiser et ouvrir un proxy, à l'aide des méthodes **LiveSearchPortBinding_CreateServiceProxy** et **WsOpenServiceProxy**
- Initialiser la structure **SearchRequest**
- S'il y a initialisation de ressources, il faut également libérer les ressources, avec les API WWSAPI associées
- Ensuite, il faut effectuer la recherche en utilisant la méthode **LiveSearchPortBinding_Search**

IMPLÉMENTATION DE L'AGENT BING

L'agent bing est le premier du pipeline, il prendra comme paramètre dans son constructeur :
Un point d'entrée pour envoyer un message :

```
ITarget<RawData>& messageEnvoye
```

Un point d'entrée pour recevoir un éventuel message d'arrêt :

```
ISource<BOOL>& stop
```

Une structure : **WebServiceConfig**,

```
struct WebServiceConfiguration
{
  WCHAR Requete[1024];
  int NombreVignettes;
  int NumeroPage;
  int TotalPages;
  int TotalVignettes;
  BOOL NouvelleRequete;
  BOOL Sauvegarder;

};
```

qui contiendra, entre autres, le texte de la requête, le nombre de vignettes à télécharger et le numéro de page à utiliser.

```
class DPEBingAgent_API DPEBingAgent : public DPEBaseAgent
{
public :
explicit DPEBingAgent(ITarget<RawData>& messageEnvoye,
       ISource<BOOL>& stop,
       WebServiceConfiguration config);
protected:
 void run();

//Code omis pour plus de clarté
}
```

Le constructeur s'implémente de la manière suivante :

```
DPEBingAgent::DPEBingAgent(ITarget<RawData> &messageEnvoye,
         ISource<BOOL>& stop,
         WebServiceConfiguration config)
 :_messageEnvoye(messageEnvoye),
 _stop(stop),
 _config(config),
 _wsError(nullptr),
 _wsHeap(nullptr),
 _wsServiceProxy(nullptr)
{

 InitialiserProxyWeb();
 InitialiserServiceWeb();
}
```

Vous noterez que notre agent Bing dérive de **DPEBaseAgent** qui est en fait une classe qui dérive elle-même de la classe **agent**, et qui propose des services supplémentaires que nous réutiliserons dans tous nos agents. Je ne rentrerai pas dans le détail de ces services, mais vous trouverez avec cet article, le code complet de l'application.
Enfin on implémente la méthode **run()** de la classe agent

- La méthode **run()**, appelle la méthode **RechercherImages()**, qui retourne, si tout va bien, le nombre d'images dans une structure de type **SearchResponse**
  Le nombre de réponses nous servira comme nombre d'itérations de notre boucle. A chaque itération, nous copions (**CopierReponse()**) une réponse dans la structure **RawData**, que nous envoyons **asend()** au prochain agent dans le pipeline.
  Note *: La méthode asend() diffère de la méthode send(), dans le sens où elle envoie le message en mode asynchrone, c'est-à-dire qu'elle n'attend pas l'éventuelle réponse de l'agent suivant dans le pipeline.*

La méthode **CopierReponse()**, récupère les adresses électroniques des images à partir de la structure **SearchResponse**, et les copie dans les structures **ImageResult**, et **Thumbnail**

```
typedef struct ImageResult
{
    WCHAR* Title;
    WCHAR* MediaUrl;
    WCHAR* Url;
    WCHAR* DisplayUrl;
    unsigned int Width;
    unsigned int Height;
    unsigned int FileSize;
    WCHAR* ContentType;
```

```
    struct Thumbnail* Thumbnail;
} ImageResult;

typedef struct Thumbnail
{
    WCHAR* Url;
    WCHAR* ContentType;
    unsigned int Width;
    unsigned int Height;
    unsigned int FileSize;
    unsigned int RunTime;
} Thumbnail;
```

Comme vous pouvez le constater, ces deux structures contiendront les informations nécessaires, Url, Taille, largeur et hauteur des images etc., qui seront utilisées par les agents suivants dans le pipeline.
A la dernière itération, nous informons (**_data.Sentinel=TRUE**) l'agent suivant, que le nombre de réponses total a été atteint. Enfin on signale que l'agent a terminé son travail avec la méthode **done()**.

Note : *Le commentaire* //Code omis pour plus de clarté, *correspond au code de gestion d'erreur, qui permet si une erreur survient, de diffuser l'erreur d'agent en agent, et surtout d'arrêter l'exécution du pipeline.*

L'AGENT HTTP

L'agent HTTP, a pour rôle, de télécharger en fonction d'une adresse électronique fournie par l'agent BING, les données brutes d'une image, et les envoyer à l'agent suivant dans le pipeline.
Pour pouvoir télécharger des images provenant du résultat du service Bing, nous avons à notre disposition dans Windows 7, deux API, l'API WinINet http ://msdn.microsoft.com/en-us/library/aa385483(v=VS.85).aspx et WinHTTP http ://msdn.microsoft.com/en-us/library/aa384273(v=VS.85).aspx
Dans notre exemple, nous utiliserons l'API WinINET, car notre application n'est pas destinée à tourner sur un serveur. Par contre, il sera préférable d'utiliser l'API WinHTTP, si vous souhaitez développer un service, ou faire tourner du code sur un serveur.

UTILISATION DE WININET

▪ Tout d'abord il faut inclure l'entête **WinInet.h**

```
#include <WinInet.h>
```

▪ Se lier à la librairie **Wininet.lib**
▪ Ensuite il faut ouvrir une session internet avec l'API **InternetOpen()**

```
HINTERNET _hSession
_hSession=InternetOpen(L"WinInet agent V 0.5 (EV)",
INTERNET_OPEN_TYPE_PRECONFIG,NULL,NULL,0);
```

▪ Avec le handle **_hSession** obtenu, on ouvre une adresse électronique à l'aide de la méthode **InternetOpenUrl()**.
Ou le paramètre **Url.c_str()** contient l'adresse électronique de l'image à télécharger. Souvenez-vous, cette adresse est fournie par l'agent Bing.
▪ Ensuite, il faut lire les données brutes à l'aide de la méthode **InterneteadFile()**
hOpenUrl, est un handle qui provient de l'ouverture de l'Url à l'étape 3
pBuffer, est un tableau d'octets qui a été initialisé à la taille de l'image Bing, et que nous sauvegarderons dans notre structure RawData et que nous fournirons à l'agent suivant dans le pipeline
dwNumberOfByteToRead, correspond au nombre d'octets à lire.
lpdwNumberOfBytesRead, correspond au nombre d'octets effectivement lus
▪ Enfin il faut libérer les deux handles **_hSession** et **hOpenUrl** que nous avons ouverts.

```
InternetCloseHandle(hOpenUrl);
InternetCloseHandle(_hSession);
```

IMPLÉMENTATION DE L'AGENT HTTP

L'agent HTTP est le second agent du pipeline, il recevra un message de l'agent Bing contenant les adresses électroniques à télécharger et enverra un message à l'agent suivant dans le pipeline contenant les données brutes de l'image. Son constructeur prendra alors comme paramètres :
Un point d'entrée pour envoyer un message :

```
ITarget<RawData>& messageEnvoye
```

Un point d'entrée pour recevoir le message de l'agent Bing :

```
ISource<RawData>& messageRecu ;
```

Un point d'entrée pour recevoir un éventuel message d'arrêt :

```
ISource<BOOL>& stop
```

Une variable booléene **telechargerImageComplete** indiquant s'il faut télécharger la vignette ou l'image complète (voir le code source complet)
Le constructeur s'implémente de la manière suivante :

```
DPEHttpAgent::DPEHttpAgent(ITarget<RawData>& messageEnvoye,
        ISource<RawData>& messageRecu,
        ISource<BOOL>& stop,
         BOOL telechargerImageComplete)
  :_messageEnvoye(messageEnvoye),
  _messageRecu(messageRecu),
  _stop(stop),
  _telechargerImageComplete(telechargerImageComplete),
  _hSession(0)

{

  _hSession=InternetOpen(L"WinInet agent V 0.5 (EV)",
INTERNET_OPEN_TYPE_PRECONFIG,NULL,NULL,0);
}
```

Puis on implémente la méthode **run()**, **voir code source complet**. Vous noterez que la méthode run() rentre dans une boucle qui se finira lorsque la variable **Sentinel** de notre message sera à **TRUE**. C'est l'agent précédent dans le pipeline, en l'occurrence l'agent BING, qui en avertira notre agent HTTP. A l'initialisation de la boucle, l'agent HTTP, attend de recevoir (**_data=receive()**) un message. Tant que le message n'est pas envoyé par l'agent BING, la méthode **receive()** bloque l'appel. Dès qu'un message est reçu, l'agent HTTP, va télécharger en fonction d'une adresse électronique fournie par l'agent BING, les données brutes de l'image. Données brutes qu'il sauvegarde dans la structure **RawData** et l'envoie immédiatement à l'agent suivant dans le pipeline. Enfin l'agent signale qu'il a fini son travail, à l'aide de la méthode **done()**.
Retrouver la suite de cet article et le code complet sur http://msdn.microsoft.com/fr-fr/visualc/msdn.bing.windows7

Eric Vernié - *Microsoft France*

Le Sudoku puissance 39

Lars Peters Endresen et Håvard Graff, deux ingénieurs de talents d'Oslo, partagent avec nous, la création d'une première mondiale : un Sudoku avec 39 indices… Explications.

LA NATURE DU CHALLENGE

Il existe plus de $6x10^{21}$ grilles valides de Sudoku en format 9 x 9 cases. Utiliser la force brute pour essayer toutes les combinaisons n'est pas un exercice de programmation trop difficile, la vraie difficulté consistait à concevoir un programme qui terminerait ses calculs avant la mort du programmeur ! Le défi relevé par nos ingénieurs était de produire un puzzle avec 39 indications, sachant que d'autres avaient déjà conçu une grille avec 38.

Etape 1 : changer l'algorithme

Plutôt que d'utiliser la force brute afin de créer toutes les solutions possibles, nous avons décidé de prendre un puzzle existant, d'en retirer une ou deux indications, puis d'utiliser un solveur récursif pour produire un nouveau puzzle. Pour trouver un nouveau puzzle à 17 indications, nous démarrons avec un puzzle à 18, retirons deux indications, et recherchons une solution valide.

Ainsi, par exemple, dans le puzzle de la [Fig.1], les indications 3 et 9 sont retirées en premier de la colonne 1. Le « solveur » remplit alors chacune des cellules non trouvées avec une liste de possibilités valides. Le solveur élimine récursivement les possibilités pour trouver un puzzle valide, en faisant attention à ne pas avoir d'indications redondantes. Nous appelons cette méthode l'algorithme « -2+1 »

Nous utiliserons la même technique pour trouver le "trente-neuvième". En prenant un « trente-huitième », nous retirons une indication puis en ajoutons deux. Nous l'appelons l'algorithme « -1+2 ».

Etape 2 : optimisation du code

Les CPU modernes ont des instructions qui peuvent travailler sur plusieurs données simultanément, c'est-à-dire, en utilisant le « single Instruction Multiple Data » ou sIMD.

Remplacer les instructions traditionnelles par des instructions sIMD produit du code beaucoup plus rapide.

Les exemples incluent MMX et les différentes extensions de streaming sIMD (SSE, SSE2,...).

Fig.1

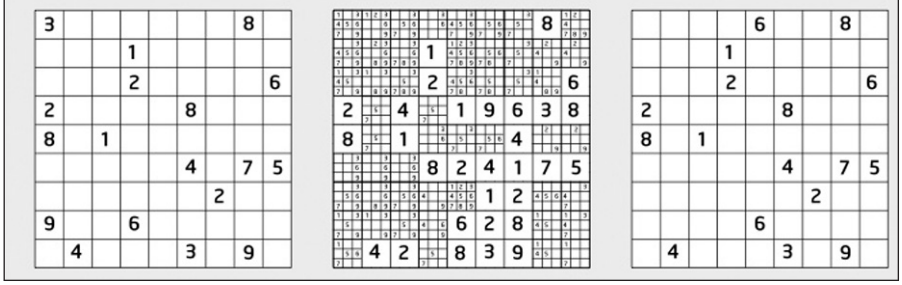


Fig.2

```
for(int num=0; num < 9; num++)
{
    __m128i xmm0 = _mm_and_si128(BinSmallNum, BinNum[num]);
    for(int i=0; i < 9; i++)
    {
     __m128i BoxSum    =   _mm_and_si128(BinBox[i],    xmm0);
     __m128i RowSum    =   _mm_and_si128(BinRow[i],    xmm0);
     __m128i ColumnSum =   _mm_and_si128(BinColumn[i], xmm0);
     if (ExactlyOneBit(BoxSum))
     {
        int cell=BitToNum(BoxSum);
        FoundNumber(cell, num);
        return true;
        }
     }
}
```

xmm0 becomes a bitmask for all occurrences of a particular number

i is used to traverse each row, column and box

Test to see if exactly one number is used

Ajouter des SSE « intrinsèques »

Les SSE « intrinsèques » (SSE Intrinsics) sont des fonctions assembleurs, générées par le compilateur, qui peuvent être appelées par du code C, C++ et offrent un accès bas niveau aux fonctionnalités sIMD sans avoir besoin d'utiliser l'assembleur intégré. Si l'on compare à l'utilisation de l'assembleur intégré, les fonctions intrinsèques offrent un accès aux instructions qui ne peuvent pas être générées en utilisant les constructeurs standard des langages C et C++.

Le compilateur Intel supporte une large gamme d'extensions architecturales, des anciennes instructions MMX à la dernière génération SSE 4.2.

Le code de la [Fig.2] montre comment les registres 128 bits SSE2 sont utilisés dans le code Sudoku. La première génération du générateur Sudoku n'utilisait pas les instructions SSE ou intrinsèques. Retravailler la première version du code pour utiliser les registres SSE2 a pris du temps. L'ajout des SSE intrinsèques a amélioré les performances par un facteur de plusieurs centaines.

Utiliser les intrinsèques SSE a ses inconvénients. Vous pouvez verrouiller votre implémentation pour une génération particulière d'architecture. Les noms longs des fonctions SSE peuvent rendre votre code C++ presque incompréhensible, et la courbe d'apprentissage pour le programmeur est particulièrement ardue. Dans le cas du générateur Sudoku, l'amélioration des performances a largement compensé les efforts supplémentaires nécessaires.

Développement du code

Après avoir écrit le code initial non optimisé, nous avons travaillé sur l'amélioration des performances en trois étapes:

- **étape 1** : changement d'algorithme pour désactiver l'approche par force brute
- **étape 2** : optimisation du code pour tirer avantage des instructions SSE
- **étape 3** : ajout du parallélisme

Le code a été développé en deux ans, principalement en dehors des heures de bureau, représentant 2000 à 3000 heures de programmation.

Etape 3 : ajout du parallélisme

Nous avons utilisé les tasks OpenMP – qui sont définies dans OpenMP 3.0. OpenMP 3.0 est supporté par le compilateur Intel C/C++ dès la version 11. Ajouter le parallélisme a représenté environ deux semaines de travail – ce qui nous a semblé un gros effort à l'époque, mais en fait peu de chose comparé à la durée du projet. La [Fig.3] donne un exemple de l'utilisation des tâches OpenMP.

L'un des aspects les plus complexes de l'ajout du code OpenMP fut de comprendre comment les variables étaient traitées. Les données en OpenMP peuvent être partagées ou privées. L'ajustement de notre code pour arriver au bon niveau pour nos variables a nécessité plusieurs itérations. La majorité du temps pris par l'ajout du parallélisme fut de retravailler le code pour minimiser le besoin de partage de données entre les différentes tâches, et de s'assurer de l'absence de dépendance entre les différentes boucles parallélisées.

UTILISER INTEL CILK PLUS PLUTÔT QUE OPENMP

Intel Parallel Studio supporte différentes méthodes pour paralléliser un programme. Intel Parallel Building Blocks, qui offre plusieurs supports pour la programmation parallèle.

Comme indiqué précédemment, le « trente-neuvième » Sudoku fut parallélisé en utilisant OpenMP, ce qui est plus facile que l'utilisation des threads natifs.

Si le projet démarrait aujourd'hui, utiliser Intel Cilk Plus aurait été un choix attractif. Cilk est l'une des méthodes les plus simples pour paralléliser un programme.

Ajouter Cilk au code C existant est vraiment très simple. Cilk utilise trois mots-clés : cilk_spawn, cilk_sync, et cilk_for. Une fois le fichier en-tête cilk.h inclus dans le programme, les mots-clés sont prêts à l'emploi. Voir [Fig.4].

En Cilk, le programmeur ne contrôle pas le parallélisme d'un programme, mais exprime une intention. En introduisant les mots-clés Cilk dans un programme, le programmeur donne la permission au code de s'exécuter en parallèle. La décision d'exécuter le code en parallèle ou non repose sur le gestionnaire Cilk à l'exécution.

Le runtime Intel Cilk se charge automatiquement de l'équilibrage de charge.

La [Fig.5] montre comment le code Sudoku est parallélisé en utilisant le mot-clé Cilk cilk_for. Le code est inséré à la même place que la tâche OpenMP originale (voir [Fig.3]).

INTEL CILK PLUS EST LA MANIÈRE LA PLUS SIMPLE DE PARALLÉLISER UN PROGRAMME EXISTANT

En parallélisant du code, le programmeur doit faire attention à ne pas introduire de « data race ». Si des variables globales existent, le code doit être retravaillé pour restreindre la portée des variables - en utilisant par exemple des variables locales ou automatiques à la place des globales.

S'il est impossible d'éliminer toutes les variables globales, l'accès à ces variables doit être protégé pour qu'un seul thread à la fois puisse y accéder.

En Cilk, la méthode la plus simple pour gérer les variables globales et partagées est de les déclarer en tant que « réducteur » (reducer).

Quand un travailleur accède à un réducteur, il en obtient une vue privée dans laquelle il peut le manipuler en toute sécurité.

Les vues sont ensuite fusionnées dans le code non parallélisé par un appel à get_value(). Le code ([Fig.6]) montre comment le global gnumCilkPuzzlessolved est déclaré comme reducer_opadd. L'appel à get_value() combine toutes les valeurs du réducteur, on en obtient donc la valeur correcte [Fig.5].

Fig.3

```
#pragma omp parallel                          ← A pool of threads is created here
{
    #pragma omp single nowait                 ← One thread gets to execute the for loop
    {
        for( int i=0; i< NUM_NODES -1; i++)
        {
            NODE Node1 = pPuzzle ->Nodes [i];
            if (Node1.number > 0)
        {
            //create copy of the top level node;
            memcpy (&gPuzzles[i];pPuzzle, sizeof (SUDOKU));
            #pragma omp taskprivate (i)
            GenDoWork (&gPuzzles[i],i;
        }                                     ← The single "for loop" thread creates a task
    }                                           for each instance of GenDoWork ()
}
}
```

Fig.4

```
#include <cilk/cilk.h>
void work(int num}
{
    // add code here
}

void func1()
{
        cilk_spawn work(1);
        work(2);
        cilk_sync;
}

void func2()
{
    cilk_for(int i=0; i<9; i++)
    {
        work(3);
    }

}
```

Fig.5

```
#include <cilk/cilk.h>
.
.
.
cilk_for(int i = 0 ; i < NUM_NODES -1; i++ )

    {
        NODE Node1 = pPuzzle->Nodes[i];
        if(Node1.number > 0)
        {
            // create a copy of the top level node;
            memcpy(&gPuzzles[i],pPuzzle,sizeof(SUDOKU));
            GenDoWork(&gPuzzles[i],i);
        }
     }
```

Fig.6

```
int gNumCilkPuzzlesSolved;      // global variable
.
.
gNumCilkPuzzlesSolved++;         // somewhere in parallel code
.
.
int Tmp = gNumCilkPuzzlesSolved; // somewhere in serial code

(a) The global variable gNumCilkPuzzlesSolved is unsafe to use in parallel code.

#include <cilk/reducer_oppad.h>
cilk::reducer_opadd<int> gNumCilkPuzzlesSolved;
.
.
gNumCilkPuzzlesSolved++;          // in parallel code
.
.
int Tmp = gNumCilkPuzzlesSolved.get_value();// in serial code


(b) The  global variable is declared to be a reducer, making it safe to access.
```

CILK_SPAWN ET CILK_SYNC

Les lignes entre cilk_spawn et cilk_sync sont appelées la « continuation ». Cilk_spawn donne la permission au runtime d'exécuter le work(1) en parallèle avec le code de continuation. S'il y a un worker disponible, l'ordonnanceur « vole » le code de continuation au premier worker et l'assigne à un second worker -alors même que le premier continue à exécuter work (1). Après cilk_sync, le code retourne à une exécution « normale » (non parallélisée).

CILK_FOR

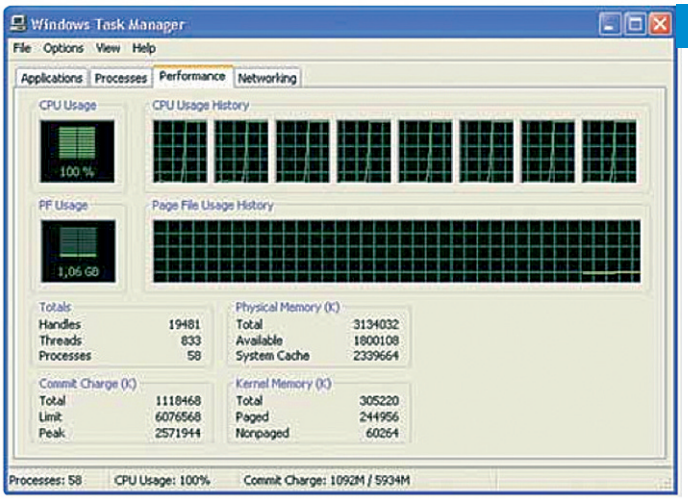
Il remplace la boucle standard C, C++ « for ». Les boucles sont partagées entre les worker disponibles. Aucun ordre d'exécution n'est garanti. Une fois toutes les boucles terminées, le programme continue. Si les boucles sont de durées différentes, l'équilibrage de charge sera géré par l'algorithme du scheduler.

LES RÉSULTATS

Dans notre première solution OpenMP, une fois le code parallèle ajouté au projet, nous avons été récompensé de voir que sur un sMt 4 cœurs - qui peut supporter huit threads matériels - les huit threads matériels étaient occupés. Voir [Fig.7]. Des expériences ultérieures utilisant Cilk ont montré que les solutions Cilk étaient aussi performantes qu'OpenMP, tout en étant plus simples à implémenter. La Figure 8 montre les trois nouveaux « trente-neuvièmes » que nous avons trouvés avec notre générateur de Sudoku.

Stephen **Blair-Chappell** - Intel compiler Labs

Fig.7



Intel Cilk Plus

Intel Cilk Plus contient ces fonctionnalités majeures :

- un ensemble de mots-clés pour l'expression du parallélisme de tâches
- des réducteurs, ce qui élimine les contentions pour les variables partagées entre les tâches, en créant automatiquement des vues pour chaque tâche et en les réduisant à une valeur partagée à la fin de l'exécution des tâches.
- des notations matricielles qui fournissent le parallélisme des données pour des sections de C, C++ pour manipuler des tableaux.
- des fonctions élémentaires, qui autorisent le parallélisme des données de fonctions entières ou opérations qui peuvent être ensuite appliquées à tout ou partie de tableaux ou scalaires.
- Le pragma simd, qui vous permet d'exprimer le parallélisme vectoriel pour utiliser le simd matériel tout en écrivant du code C, C++ standard avec un compilateur Intel.

Fig.8

				3				
		3	6		7			1
6		4		9	1	3		7
5					3		2	4
7	4			6	2	5		3
		2			5	7	1	
2		5	7	1	6	4		
4		6		2	9	1	7	5

				3				
		3	6		7			1
6		4		9	1	3		7
5					3		2	4
1	4			6	2	5		3
		2			5	1	7	
2		5	1	7	6	4		
4		6		2	9	7	1	5

		1		3	7	6	4	
							8	
6		4		9	1	3		7
		2			6			3
1	4			2	3	5		6
3				1	5	4	2	
2		5			9	7		4
4	1			7	2			
	7			5		2	6	

Cilk Plus : une extension du langage C pour la programmation parallèle

Depuis que les processeurs Intel sont majoritairement multicœurs, il est devenu obligatoire pour Intel de développer ses technologies de programmation parallèle. Il offre des solutions standard pour la programmation parallèle depuis plusieurs années, comme OpenMP et MPI. Tout en continuant ses investissements dans ces standards, Intel propose de nouveaux outils parallèles, basés sur le modèle de programmation appelé Parallel Building Blocks (PBB).

PBB comprend actuellement les trois composants suivants :

1. Threading Building Blocks: Il est actuellement disponible à la fois comme un produit supporté et en mode open source. TBB est implémenté comme une bibliothèque C++ indépendante du compilateur. Il offre à la fois des primitives de bas niveau et des algorithmes parallèles de haut niveau.
2. Cilk Plus: extension de langage, implémentée par le compilateur, qui offre des constructeurs parallèles pour les tâches et les données.
3. Array Building Blocks (ArBB): ArBB offre une solution programmation parallèle des vecteurs qui libère les développeurs des dépendances aux mécanismes parallèles de bas niveau spécifiques ou des architectures matérielles.

Nous allons nous concentrer sur Cilk Plus. Cet article décrit les points principaux et les bénéfices du langage, mais n'est pas une documentation complète.

Pour le programmeur

Le modèle de programmation Intel prend en considération à la fois la perspective matérielle, avec de multiples niveaux de ressources parallèles, et le point de vue du programmeur.

L'architecture offre différentes ressources parallèles : multicœurs, vecteurs, hyper-threading et les caches. Les programmeurs ont différents niveaux d'intérêt dans la programmation de ces ressources. PBB (et en particulier Cilk Plus) permet deux approches : le programmeur peut choisir de décrire explicitement le parallèle au niveau le plus bas, en utilisant des tâches (tasks), et dans chaque tâche, écrire du parallélisme au niveau vectoriel. De la même façon, le programmeur peut bénéficier de toutes les ressources matérielles parallèles en indiquant ce qui doit être fait en parallèle et les données sur lesquelles il veut travailler, et permettre ainsi au compilateur et au système d'adapter le travail aux ressources matérielles.

Beaucoup d'applications sont faites de composants écrits indépendamment. Comme par exemple des bibliothèques que l'auteur de l'application achète à une tierce partie. Ce peut être aussi le résultat de la division du travail d'une équipe importante en plus petits groupes. Les petits groupes sont plus intéressés par la communication au niveau des interfaces, plutôt que de l'implémentation. Pour atteindre ce niveau de collaboration dans la programmation parallèle, le gestionnaire de tâches doit permettre d'assembler les modules pour créer l'application, en gardant des performances correctes. Le langage Cilk Plus, comme tous les autres composants de TBB, utilise un mode de gestion des tâches en vol de travail (en Anglais : work stealing). Le vol de travail est une technique de gestion de tâches qui se caractérise par un mécanisme dans lequel chaque unité de travail peut migrer entre le travailleur (ou worker) qui le génère vers un autre travailleur qui va l'exécuter, uniquement au cas ou le second travailleur n'a aucun travail à exécuter. Un travailleur inactif, vole le travail d'une victime qui a une file de travaux en attente. Au contraire, il n'y a aucun mécanisme par lequel un travailleur qui rencontrerait des opportunités de travail parallèle, puisse déléguer ce travail à d'autres travailleurs. Les détails précis de ce mécanisme de gestion dépassent le cadre de cet article.

La parallélisation des tâches avec Intel Cilk Plus

Le langage Cilk Plus offre deux constructeurs pour l'implémentation du parallélisme au niveau des tâches. Le premier offre deux mots-clés : _Cilk_spawn et _Cilk_sync. L'exemple qui suit compare une implémentation récursive du code Fibonacci en C à une implémentation parallèle, dérivée de l'implémentation série en ajoutant les nouveaux mots clés :

```
int fib (int n)
{
    if (n <= 2)
         return n;
    else {
         int x,y;
        x = fib(n-1);
        y = fib(n-2);
        return x+y;
   }
}

int fib (int n)
{
    if (n <= 2)
         return n;
    else {
         int x,y;
        x = _Cilk_spawn fib(n-1);
        y = fib(n-2);
        _Cilk_sync;
        return x+y;
   }
}
```

On remarque que l'ajout des mots-clés n'est pas intrusif. Changer le programme série (le programme non parallèle donc) en un programme parallè-

le demande uniquement l'insertion des mots clés appropriés, mais ne modifie pas le programme sous-jacent. L'insertion ne modifie ni la lisibilité ni la maintenabilité du programme. Le code (série) original est facile à trouver et la logique du programme est aussi évidente que dans l'original. Cette observation met en avant l'un des principes majeurs de conception du langage. Il a été conçu pour la parallélisation de code C, C++ existant. Le langage se focalise sur la parallélisation du code tout en minimisant les modifications, mais en apportant des garanties importantes pour l'équivalence du code série.

Le mot clé _Cilk_spawn s'applique à l'appel d'une fonction, comme dans l'exemple ci-dessus fib(n-1). Cela signifie que la fonction engendrée est exécutée en parallèle avec la fonction incluse. La fonction incluse est appelée une fonction parent et la fonction engendrée est appelée enfant. Le mot-clé _Cilk_sync signifie que l'exécution du parent doit attendre la fin de toutes les tâches qui ont été engendrées à partir de la fonction parente. Notez que ce mécanisme réutilise le concept de fonction, qui est bien maîtrisé dans la programmation C, C++, à la fois comme une région parallèle (du côté parent) et comme une unité qui peut être générée (du côté enfant). Ces deux utilisations permettent la conception de la parallélisation de programmes existants avec un effort minimal. La réutilisation d'une région parallèle diminue le besoin d'introduire un nouveau concept syntaxique qui pourrait servir de région parallèle. La réutilisation d'une fonction comme l'unité de ce qui peut être généré, peut être vue comme une limite, puisque qu'en parallélisant un programme existant le développeur pourrait vouloir indiquer une diminution de la portée lexicale, comme une instruction ou une boucle, qui peut s'exécuter en parallèle d'une autre instruction.

Cependant, en utilisant des fonctions comme unité de génération, le modèle de données est bien compris par le programmeur, et les extensions du langage n'ont pas besoin d'introduire de nouvelles règles qui demanderaient un apprentissage supplémentaire. Dans le modèle courant, il apparaît clairement aux programmeurs quelles variables sont dans le champ de la fonction générée, quelles variables appartiennent à la fonction enfant, et comment les arguments sont transmis. Bien sûr, la transmission des arguments est conforme à la méthode du langage sous-jacent. Les valeurs des arguments sont évaluées par la tâche parent avant de détacher l'enfant pour autoriser la parallélisation. Dans l'exemple de fib(n-1), la valeur de n-1 est calculée par le parent avant que l'enfant ne soit généré. Un exemple moins trivial est quand le calcul des arguments se révèle plus compliqué, comme dans l'exemple suivant:

```
val = _Cilk_spawn f(g(x),h(y));
```

Les arguments de la fonction f sont eux-mêmes des appels de fonctions. Dans l'opération appliquée à l'expression complète, alors g et h auraient dû être autorisées à s'exécuter simultanément l'une par rapport à l'autre, comme par rapport à la fonction parent. Dans le cas d'une data race entre le code de la fonction f et h il n'y a pas de place pour insérer une directive de synchronisation.

Bien sûr l'attribut le plus important dans le comportement de _Cilk_spawn est lié à la symétrie entre les deux parties du parent, au point _Cilk_spawn. Un point _Cilk_spawn ne garantit pas une exécution en parallèle, mais offre plutôt une opportunité d'exécution en parallèle. Le travailleur dont l'exécution atteint le mot-clé _Cilk_spawn, continuera à exécuter l'une des deux branches identifiées par le _Cilk_spawn, et détachera l'autre, pour la placer dans une file d'attente pour une exécution ultérieure. Comme on l'a vu précédemment, l'élément de travail dans la file peut être volé ou non pour une exécution par un autre travailleur sur un autre cœur. Alors que l'on pourrait penser que la fonction marquée pour être détachée est celle qui sera mise en file d'attente pour exécution ultérieure, et que le travailleur qui atteint le mot clé _Cilk_spawn continuera à exécuter la suite de la fonction parent, l'exécution se fait dans l'autre sens. Le travailleur qui atteint le mot clé _Cilk_spawn va détacher la suite de la fonction parente, la placer dans sa file de travail, et continuera l'exécution de la fonction spécifiée par le mot clé _Cilk_spawn.

Pourquoi cette asymétrie est-elle importante ? La raison en est que l'ordre d'évaluation correspond à l'ordre d'évaluation d'un appel de fonction dans un programme série C / C++. La fonction parente continuera à exécuter l'enfant, comme c'est le cas dans un appel de fonction, et si la suite n'est pas volée, il reviendra, rechargera la suite comme un élément de travail, et continuera à l'exécuter. Cet ordre d'évaluation est le même que celui d'un appel de fonction dans un programme série. Les mots clés de l'extension peuvent être redéfinis. Par exemple, le programmeur peut utiliser le préprocesseur C, C++ et les #define comme suit :

```
#define _Cilk_spawn
#define _Cilk_sync
```

Appliquer ces directives à un code source qui utilise l'extension générera un programme valide dans le langage C, C++. Aussi longtemps que la version du programme ne génère pas de data race, la sérialisation du programme sera sémantiquement équivalente à l'exécution parallèle du programme et produira les mêmes résultats.

Le mot-clé _Cilk_sync dans l'exemple de la fonction fib() était nécessaire pour garantir que les valeurs des variables x et y seraient mises à jour par l'exécution asynchrone de fib (n - 1) et fib (n - 2) avant d'être utilisées. En plus du mot-clé explicite disponible, il y a un _Cilk_sync implicite à la fin de chaque fonction parent. Le mot-clé implicite est en fait inséré par le compilateur. La conséquence du _Cilk_sync implicite est que la fonction parent s'exécute aussi longtemps que les tâches générées sont en cours d'exécution. L'un des bénéfices de cette propriété est qu'elle permet aux programmeurs de continuer à voir les fonctions comme des unités de travail, et quand une fonction se termine tout le travail attaché correspondant est fini. Le second constructeur de tâches fournit le mot-clé _Cilk_for. C'est un moyen de paralléliser les boucles for en C, C++. Dans une instruction comme :

```
_Cilk_for (int i = 0; i < N; ++i)  {
    body;
}
```

Le compilateur garantit quelques restrictions du constructeur for. Cela inclut que la variable index (i dans notre exemple) soit présente dans les trois expressions de la boucle, qu'elle soit initialisée dans la première, qu'elle soit comparée à une valeur qui ne change pas dans la boucle, et qu'elle soit incrémentée par une valeur qui ne change pas dans la boucle. Il ne peut pas changer dans le corps de la boucle. L'effet de cette restriction est que quand la boucle va s'exécuter, son nombre d'itérations soit connu du gestionnaire d'exécution. L'exécution de la boucle _Cilk_for utilise une approche « diviser pour conquérir ». Le travailleur qui atteint le constructeur _Cilk_for, calcule le nombre d'exécutions. Il prend la moitié supérieure des itérations et les place en attente dans sa propre file pour une évaluation ultérieure. Il continue alors à diviser pour conquérir la moitié inférieure, jusqu'à ce que le nombre d'itérations devienne suffisamment petit (déterminé par heuristique à l'exécution) puis les exécute séquentiellement. Une fois terminé, le travailleur récupère le dernier élément de travail de sa propre file. Cela sera le deuxième ensemble d'itérations de la moitié inférieure. Si un autre travailleur a une file de travail vide et aucun travail à faire, il peut voler le travail du cœur qui a créé la liste des éléments de tra-

vail correspondant aux itérations de la boucle. C'est ainsi que le travail concurrent est créé. Dans ce cas, il volera, à partir du haut de la file, un élément correspondant au nombre maximum d'itérations. Cette séquence de vol est celle qui interfère le moins avec la localisation du cache de la victime. Il fournira le plus de travail au voleur en utilisant le plus petit nombre de vols. Utiliser un nombre minimum de vols fournit du travail pour tous les travailleurs avec un surcoût minimal. Aussi longtemps que le travail n'est pas volé, l'ordre d'évaluation des itérations de la boucle exécuté par un seul travailleur est exactement le même que l'ordre lors d'une exécution en série.

Vecteurs en Cilk Plus

Cilk Plus offre plusieurs possibilités pour utiliser le parallélisme disponible dans les vecteurs matériels, à l'intérieur des cœurs. Une boucle série peut être modifiée en une boucle parallèle en changeant le mot-clé « for » par _Cilk_for. Cela indiquera au compilateur qu'il n'y a pas d'ordre dans les itérations de la boucle. Comme décrit ci-dessus, le compilateur facilite l'exécution en parallèle de groupes d'itérations. De plus, il essaiera de vectoriser le code à l'intérieur des groupes d'itérations successives.

Une nouvelle syntaxe fournie avec Cilk Plus permet des opérations simples sur les matrices. Les langages C, C++ ne fournissent aucun moyen de gérer les opérations sur les matrices ; pour cela, le programmeur doit écrire une boucle et exprimer l'opération en termes d'éléments de la matrice, ce qui peut créer un ordre série qui n'est quelquefois pas voulu. L'inconvénient de l'écriture de boucle avec un ordre série est que, pour convertir la boucle série en une boucle vectorielle, le compilateur doit prouver que la gestion du vecteur serait équivalente à la gestion scalaire impliquée par la boucle et demandée par le langage. Dans la majorité des cas, ces preuves vont sûrement échouer. Par exemple, quand le programme utilise des pointeurs pour référencer la matrice (une pratique de programmation raisonnable qui permet les zones d'opération sur toutes les matrices, plutôt que d'écrire du code pour gérer une matrice spécifique) le compilateur a peu de chance de prouver que des pointeurs pointent vers des zones mémoire sans intersection.

Ecrire a[:] = b[:] + c[:]; indique au compilateur que les éléments des matrices b et c doivent être additionnés, et qu'aucun ordre n'est requis pour les opérations d'addition. Cette sémantique permet au compilateur de toujours générer du code vectorisé, au lieu du code scalaire et d'essayer de prouver que le code vectorisé serait équivalent.

L'extension de langage Cilk Plus définit un opérateur de section matricielle dont la syntaxe est array_base[start:length:stride] dans laquelle array_base représente n'importe quelle expression matricielle autorisée en C/C++, dont les matrices, les pointeurs et les matrices de longueur variables. « Start » est le premier élément de la matrice sur lequel s'applique la section, « Length » fournit le nombre d'éléments de la matrice et « Stride » est l'incrément entre deux éléments. L'utilisation de « Stride » est optionnelle, et s'il n'est pas fourni, alors la valeur 1 est utilisée par défaut. L'opérateur de section peut être appliqué à des matrices multidimensionnelles et mélangé à des sous-ensembles matriciels, comme dans a[0:5][3][0:8:2]; dans cet exemple, a doit être une matrice à 3 dimensions ou un pointeur vers une matrice. Le rang de l'expression est le nombre de dimensions sur lequel l'opérateur de section est utilisé. Dans l'exemple ci-dessus, le rang est 2.

Quelques fonctions intrinsèques sont fournies pour des opérations communes comme la somme élémentaire d'une matrice. Une opération de produit peut être exprimée comme :

```
x = __sec_reduce_add(a[:] * b[:]);
```

Ecrire une expression matricielle peut être la manière naturelle d'écrire un programme quand le programmeur exprime l'algorithme et la conception au niveau des opérations matricielles. Pour un programme série existant, un changement dans les notations matricielles peut être intrusif. Une modification moins intrusive est disponible dans Cilk Plus, sous la forme #pragma. Le pragma peut être ajouté à une boucle pour indiquer que la boucle est écrite en syntaxe scalaire, l'ordre série implicite n'étant pas obligatoire, alors que l'intention du programmeur est de demander au compilateur de générer du code vectorisé pour implémenter la boucle. Comme constructeur du langage, le pragma permet au compilateur de générer du code vectorisé sans avoir à prouver que ce dernier serait équivalent au code scalaire, et que le code scalaire n'est pas une demande du programmeur.

Un troisième constructeur permet au programmeur d'écrire une fonction scalaire en C/C++ standard et de la déclarer comme une « fonction élémentaire ». L'intention du programmeur en utilisant une fonction élémentaire est de la déployer sur plusieurs éléments de matrices sans obliger un ordre d'opérations sur les éléments des matrices. L'exemple simple qui suit montre l'utilisation de fonctions élémentaires pour additionner des matrices par éléments.

```
__declspec(vector)
float v_add(float x1, float x2)
{
    return x1+x2;
}
caller:
for (int j = 0; j < N; ++j) {
   res[j] = v_add(a[j],b[j]);
}
```

__declspec(vector)indique au compilateur que la fonction v_add doit être utilisée comme fonction élémentaire. Le compilateur génèrera deux versions du code pour ce type de fonction. En plus de la version « normale », une version vectorisée sera générée et recevra un vecteur d'arguments pour x1 et un autre pour x2, puis exécutera les opérations de la fonction en utilisant les registres vectoriels hardware sur tous les arguments en entrée, retournant un vecteur de résultats au lieu d'un résultat unique. La fonction peut être appelée dans un contexte scalaire, et dans ce cas le compilateur traduira l'appel de la fonction en un appel standard, d'une fonction scalaire. Quand la fonction est appelée dans un contexte parallèle, comme dans l'exemple de la boucle de l'exemple précédent, le compilateur appellera la version vectorisée.

Si la cible de la compilation est un processeur intégrant le registre vecteur XMM, comme par exemple avec le jeu d'instruction étendu SSE2, alors le compilateur déterminera que 4 valeurs consécutives de la matrice d'entrée peuvent entrer dans le registre. Il génèrera une version de la fonction pouvant traiter 4 éléments consécutifs de la matrice, et la fonction sera appelée « N/4 time » (ou 1,2 ou 3 temps supplémentaires en version scalaire, si N n'est pas divisible par 4).

Au lieu d'appeler la fonction dans une boucle « for », le programmeur pourrait choisir « _Cilk_for » comme constructeur pour appeler la fonction. Dans ce cas, non seulement le compilateur appellera la version vectorisée, mais cela aidera les itérations de la boucle à utiliser plusieurs cœurs. Avec l'utilisation de ce constructeur, le programmeur aura les bénéfices à la fois du parallélisme vectoriel et du multi-cœur.

Robert Geva, *robert.geva@intel.com*

Comprendre VTune Amplifier XE

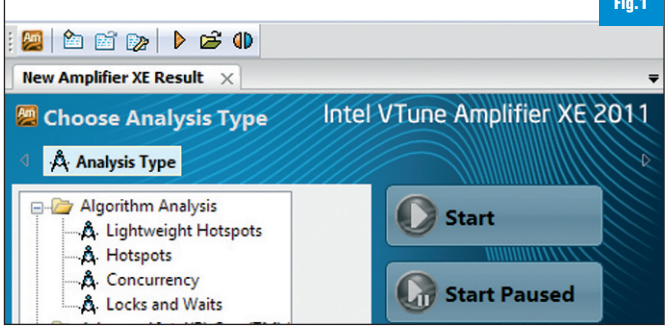
Intel VTune Amplifier XE est un outil puissant d'analyse de performances, qui peut aider les développeurs de logiciels à identifier les problèmes de performance et de micro architecture dans leurs applications.

Intel VTune Amplifier XE, avec son interface intuitive et améliorée, réalise des analyses de performances puissantes en quelques clics de souris. Il apporte des types d'analyse innovants et faciles à exécuter pour des analyses de performance algorithmiques et de micro architecture ; quelques types d'analyses prédéfinis pour les ajustements algorithmiques comprennent des analyses de points critiques légères, des analyses de points critiques, des analyses de simultanéité et des analyses de verrouillage et attente.

- Les analyses des hotspots aident le développeur à comprendre le flux applicatif (c'est-à-dire les informations de pile d'appel) et identifient les sections du code qui prennent trop de temps à s'exécuter (ou hotspot) en utilisant une technologie d'échantillonnage statistique (c'est-à-dire un échantillonnage de pile en mode utilisateur).
- Les analyses de la concurrence et des verrouillages (lock) et attentes (wait), comme les analyses hotspot, identifient les fonctions hotspot et les piles d'appel de ces fonctions, mais, en plus, mesurent comment une application utilise les processeurs disponibles sur un système donné, en tirant avantage de la technologie de profiling de tâche.
  - L'analyse de la concurrence identifie les sections à faible utilisation des processeurs, quand et comment les tâches s'exécutent, se synchronisent et attendent.
  - L'analyse de verrous et attentes aide à identifier la cause d'une mauvaise utilisation du processeur. Le problème le plus courant d'une faible utilisation est provoqué par une attente trop longue des tâches d'objets de synchronisation (verrous).
- Une analyse légère des hotspot est similaire à une analyse des hotspots mais utilise une technologie matérielle d'échantillonnage fondée sur les évènements (EBS en Anglais) pour localiser les hotspots d'une application [Fig.1].

Pour une analyse micro architecturale en profondeur, l'outil dispose de types d'analyses prédéfinis qui utilisent l'unité de monitoring des performances (PMU) pour échantillonner les évènements processeurs et ainsi identifier les problèmes de micro architecture comme les dysfonctionnements de cache, les cycles bloqués, etc. Les types d'analyse avancés sont définis pour les architectures de processeurs telles les Core 2, Core (Nehalem, Westmere) et les micro architectures de 2e génération de Core (SandyBridge). Quand ces types d'analyse sont utilisés, l'outil fournit des suggestions en mettant en évidence les fonctions problématiques.

Fig.1



Les analyses algorithmiques prédéfinies : hotspot léger, hotspot, concurrence et verrous et attente.

IDENTIFIER LES PROBLÈMES DE CALCUL

Le tuning de performances se concentre sur la réduction du temps que prend l'exécution d'une tâche convenablement définie. Les évènements de performances peuvent être utilisés pour mesurer le temps écoulé ; ainsi, réduire le temps écoulé pour terminer une tâche est équivalent à la réduction du nombre de cycles processeurs mesuré (tops d'horloge). L'analyse de hotspot « léger », qui est un type prédéfini d'analyse de VTune Amplifier XE rappelons-le, utilise les cycles de processeur et instructions «reculées(1) » pour analyser l'application. Le nombre de cycles, appelé également tops d'horloge, forme la base fondamentale pour mesurer la durée d'exécution d'un programme. La mesure totale du cycle est la vue du début à la fin du nombre total de cycles nécessaire pour terminer l'application analysée. Dans une situation typique de tuning de performances, la métrique du total des cycles peut être mesurée par les tops d'horloge.

L'un des objectifs de l'analyse et de l'optimisation micro architecturale est d'identifier les cycles dans lesquels aucune micro opération n'est envoyée en exécution. Les micro-opérations appelées aussi micro-ops ou _ops, sont de simples instructions du microprocesseur utilisées pour implémenter des instructions plus complexes. Les cycles dans lesquels aucune _ops n'est envoyée sont appelés cycles d'attentes et peuvent être dénombrés avec les évènements matériels PMU. Ces attentes peuvent créer un goulot d'étranglement important pour une unité d'un processeur. L'unité d'exécution par définition est toujours un goulot d'étranglement puisqu'elle définit le débit et une application sera aussi performante que son goulot d'étranglement. Aussi, il est extrêmement critique d'identifier les causes des cycles d'attente et de les retirer. En résumé, une unité d'exécution ne devrait pas rester inoccupée et attendre pour quelque raison que ce soit.

Il existe beaucoup de facteurs contributifs aux cycles d'attente et à une utilisation non optimale de l'unité d'exécution. Les accès à la mémoire, l'engorgement des pipelines, les problèmes de calcul (opérations à latence longue comme les divisions, changement de contrôle de mot FP, etc.) et les _ops ne se retirant pas du moteur sans ordre fixe (OOO) en sont quelques exemples. Les instructions à longues latences comme les divisions et les racines carrées peuvent introduire des attentes pendant l'exécution mais VTune Amplifier XE aide à les localiser et à déterminer si ces opérations contribuent aux cycles d'attente (voir tableau 1).

Exemple

Considérons pour cet exercice le problème des N corps. Celui-ci prédit les mouvements d'un groupe d'objets célestes qui interagissent gravitationnellement. L'application avance par étapes temporelles et à chacune d'elles, calcule la force nette sur chaque corps et met à jour sa position, son accélération et sa vitesse. Cette implémentation série nécessite O(N2) opérations à chaque itération.

(1) Instruction reculées: les générations récentes de processeurs Intel 64 et IA-32 intègrent des micro architectures utilisant un moteur d'exécution sans ordre. Elles sont aussi accompagnées par un « front end » ordonné et une logique de recul qui impose l'ordre du programme. Les instructions qui s'exécutent jusqu'à la fin sont appelées instructions reculées [4].

Intel Core 2 famille de processeurs (Intel Core 2 Duo/ Quad, etc.)	DIV	Compte le nombre de divisions exécutées. Cela comprend les divisions, divisions en virgule flottante et les racines carrées.
	CYCLES_DIV_BUSY	Compte le nombre de cycles où le processeur est occupé à diviser ou à extraire des racines carrées. La division peut être entière, X87 ou streaming SIMD (SSE). La racine carrée peut être X87 ou SSE.
Intel Core architecture (Intel Core i7, i5, i3; a.k.a Nehalem)	ARITH.DIV	Compte le nombre de cycles où le processeur est occupé à diviser ou à extraire des racines carrées. La division peut être entière, X87 ou streaming SIMD (SSE). La racine carrée peut être X87 ou SSE
	ARITH. CYCLES_DIV_BUSY	Compte le nombre de cycles où le processeur est occupé à diviser ou à extraire des racines carrées. La division peut être entière, X87 ou streaming SIMD (SSE). La racine carrée peut être X87 ou SSE
2e Generation Intel Core architecture (c.à.d. SandyBridge)	ARITH.FP_DIV	Compte le nombre de divisions exécutées.
	ARITH.FPU_DIV_ACTIVE	Compte le nombre de cycles où le processeur est occupé à diviser ou à extraire des racines carrées.

Tableau 1 : Evènements PMU utilisés pour compter des évènements spécifiques d'architecture.

```
...
// Lance la simulation sur un nombre fixe d'étapes temporelles
for (double s = 0.; s < STEPLIMIT; s += TIMESTEP)
{

  // Calcule l'accélération des corps
  for (i = 0; i < n - 1; ++i)
  {
    for (j = i + 1; j < n; ++j)
    {
      // Calcule la distance les séparant
      double dx = body[i].pos[0]-body[j].pos[0];
      double dy = body[i].pos[1]-body[j].pos[1];
      double dz = body[i].pos[2]-body[j].pos[2];

      double distsq = dx*dx + dy*dy + dz*dz;
      if (distsq < MINDIST) distsq = MINDIST;
      double dist = sqrt(distsq);

      // calcule le vecteur unitaire de j à I
      double ud[3];
      ud[0] = dx / dist;
      ud[1] = dy / dist;
      ud[2] = dz / dist;


     // F = G*mi*mj/distsq, but F = ma, so ai = G*mj/distsq
      double Gdivd = GFORCE/distsq;
      double ai = Gdivd*body[j].mass;
      double aj = Gdivd*body[i].mass;

      // Applique l'accélération en utilisant les vecteurs unitaires
      for (int k = 0; k < 3; ++k)
      {
        body[j].acc[k] += aj*ud[k];
        body[i].acc[k] -= ai*ud[k];
      }
    }
  }
  // Applique l'accélération et déplace les corps
  for (i = 0; i < n; ++i)
  {
    for (j = 0; j < 3; ++j)
    {
      body[i].vel[j] += body[i].acc[j] * TIMESTEP;
      body[i].pos[j] += body[i].vel[j] * TIMESTEP;
      body[i].acc[j] = 0.;
    }
   }
}

...
```

L'analyse de ce code sur une machine Intel Core i7 (x980), ayant les caractéristiques suivantes : 3.33GHz, 6 core + Hyper Threading autorisé, avec VTune Amplifier XE donne le résultat [Fig.2].
Une méthode d'optimisation est de remplacer la division par une multiplication réciproque comme ci-dessous :

// calcule le vecteur unitaire de j à i	// calcule le vecteur unitaire de j à i
double ud[3];	double ud[3];
ud[0] = dx / dist;	double dd=1.0 / dist;
ud[1] = dy / dist;	ud[0] = dx * dd;
ud[2] = dz / dist;	ud[1] = dy * dd;
	ud[2] = dz * dd;

Le code optimisé consomme 4.668 millions de moins de tops d'horloge et réduit le blocage des envois de 7.448 millions de cycles à 3.020 millions de cycles [Fig.3] ! De plus, l'application peut être parallélisée non seulement pour bénéficier des cœurs, mais aussi pour réduire l'impact des DIV. La parallélisation va permettre d'utiliser tous les ports des cœurs faisant les divisions.

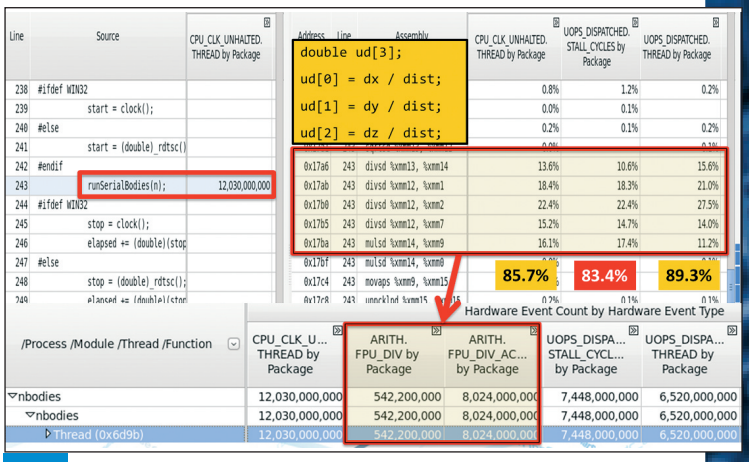


Fig.2 *La Figure 2 montre l'analyse du code, 85% des tops d'horloge, 89,3% du total des _ops envoyées et 83,4% des attentes d'envoi se produisent dans ce segment de code.*

ESTIMATION DES FLOPS

FLOPS (ou flops ou flop/s) signifie nombre d'opération en virgule flottante par seconde. Il s'agit d'une mesure utilisée fréquemment pour les calculs à hautes performances. Dans ce paragraphe, vous comprendrez comment le EBS peut aider les développeurs à estimer le nombre de flops. FLOPS fera référence aux opérations en virgule flottante en 32 et 64 bits et les opérations seront soit des additions soit des multiplications (calculées).

Comme démontré dans les figures 3 et 4, les opérations en virgule flottante peuvent être exécutées sur les vieux registres x87 ou les registres SSE. Cela dépend de comment le compilateur génère le code. Si les instructions de virgule flottante sont exécutées par les registres SSE, elles peuvent être des opérations scalaires ou vectorielles [Fig.4].

[Fig.5] : Intel AVX a introduit le support des larges registres à 256 bits (YMM0-YMM7 dans les modes opératoires de 32 bits ou moins, YMM0-YMM15 en mode 64 bits). Les 128 bits inférieurs des registres YMM sont équivalents à leur registre XMM 128 bits respectifs.

Le tableau 2 (page suivante) fournit les noms des évènements PMU qui sont utilisés pour la charge de calcul des opérations en virgule flottante exécutées par le matériel. Veuillez garder à l'esprit que toutes les instructions exécutées sont reculées à cause de la nature spéculative de l'architecture. Il est donc possible de sur-comptabiliser ces évènements (Tableau 2 et 3). VTune Amplifier XE peut utiliser n'importe quel évènement (ou tous), pour estimer les flops exécutés par le matériel. Afin de mesurer le temps écoulé, CPU_CLK_UNHALTED (les tops d'horloge) peut être utilisé. Si la fréquence du processeur est constante pendant la période de mesure, vous pouvez utiliser les tops d'horloge pour calculer le temps réellement écoulé. De la même façon, on peut utiliser CPU_CLK_UNHALTED.REF, qui compte le nombre de cycles de référence et qui n'est pas affecté par les changements de fréquence des tâches. La différence entre l'évènement de référence top d'horloge et l'évènement top d'horloge est que même si une tâche entre dans l'état d'arrêt (en exécutant l'instruction HLT), l'évènement de référence top d'horloge continuera à se décompter comme si la tâche continuait son exécution à la fréquence maximale.

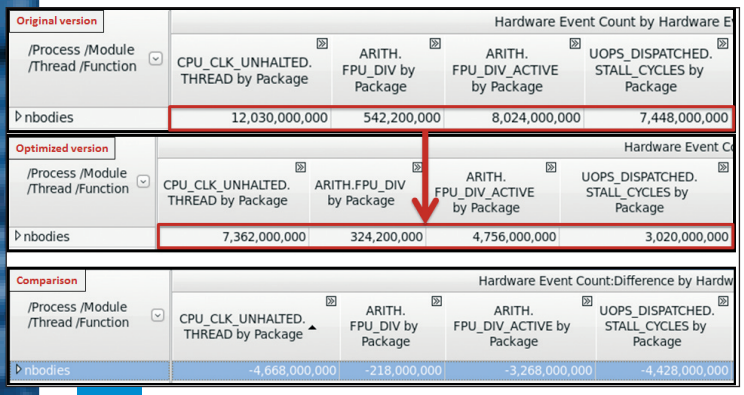


Fig.3 *Comparaisons des versions du code*

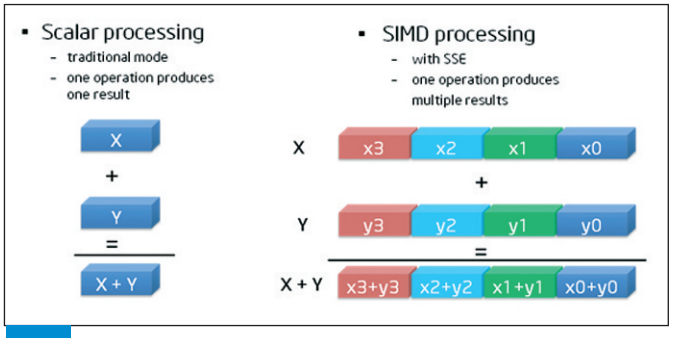


Fig.4 *Traitement scalaire vs. Traitement SIMD*

LA FORMULE DES FLOPS

La formule des FLOPS peut s'exprimer comme suit :

FLOPS = ((nombre d'ops FP / horloge) * nombre total d'ops) / Temps Ecoulé

Temps Ecoulé = CPU_CLK_UNHALTED / Fréquence processeur / Nombre de cœurs. Les cœurs ayant une valeur non nulle pour l'évènement CPU_CLK_UNHALTED doivent être pris en compte pour cette formule.

Pour démontrer comment la technologie EBS peut être utilisée pour estimer les FLOPS, on va utiliser une multiplication de matrice multithreadée. Cet exemple s'appuie sur le concept de pool de tâches et chaque tâche de ce pool exécute le code suivant :

```
double a[NUM][NUM];
double b[NUM][NUM];
double c[NUM][NUM];
…
  slice = (unsigned int) tid;
  from  = (slice * NUM) / NUM_THREADS;
  to    = ((slice + 1) * NUM) / NUM_THREADS;

  for(i = from; i < to; i++)
  {
    for(j = 0; j< NUM; j++)
    {
      for(k = 0; k < NUM; k++)
        // 2 fp ops / iteration: 1 ajout, 1 multiplie
        c[i][j] += a[i][k] * b[k][j];
    }
  }
…
```

L'application indique également les flops mesurés en divisant le total des opérations FP (2 / itération * NUM * NUM * NUM) par le temps écoulé. Le temps écoulé inclut seulement la partie de multiplication des matrices et non la surcharge de création et d'initialisation de la tâche. Afin de récupérer les échantillons pour la section de code considérée, on utilise les API

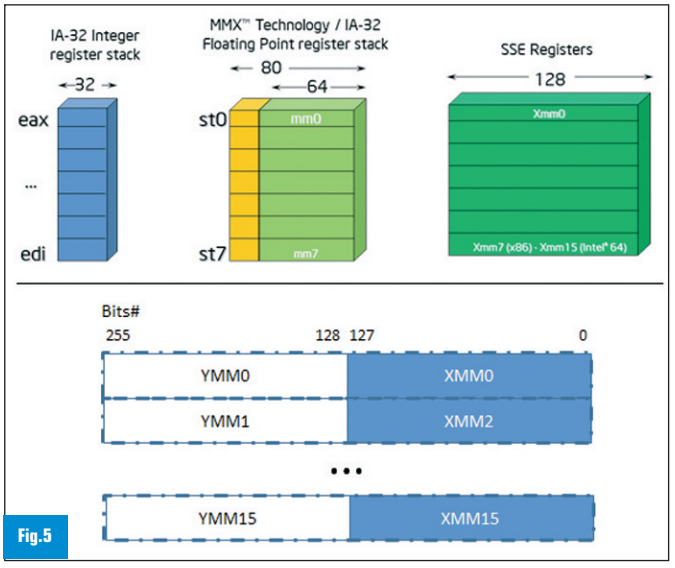


Fig.5 *Les entiers, virgule flottante, MMX, SSE (extension SIMD streaming) et AVX (extensions de vecteurs avancées) de l'architecture Intel.*

Windows Azure

Microsoft

.NET, PHP, Java...
Nous avons chacun notre langage, mais pensons tous Windows Azure.

.Net, PHP, Java... Grâce à la plateforme applicative Windows Azure, vous déployez et exécutez vos applications dans le Cloud quel que soit le langage ou l'outil de développement. Et disposez d'un environnement de développement et de production à la demande, en quelques minutes.

Avec Windows Azure, le code vous appartient, pour le reste comptez sur nous.
C'est ça la puissance du Cloud.

Pensez Windows Azure.
Essayez la plateforme Windows Azure gratuitement*
sur www.windowsazure.fr

*1 mois d'utilisation offert. Offre valable jusqu'au 31 octobre 2011 minuit.

Génération de processeur	Noms des événements processeur		
	FP operations using legacy x87		FP operations using SIMD
Intel Core 2 processor family (Intel Core 2 Duo/Quad, etc)	X87_OPS_RETIRED.ANY	Packed 64bit	SIMD_COMP_INST_RETIRED.PACKED_DOUBLE
		Packed 32bit	SIMD_COMP_INST_RETIRED.PACKED_SINGLE
		Scalar 64bit	SIMD_COMP_INST_RETIRED.SCALAR_DOUBLE
		Scalar 32bit	SIMD_COMP_INST_RETIRED.SCALAR_SINGLE
Intel Core architecture (Intel Core i7, i5, i3; a.k.a Nehalem)	FP_COMP_OPS_EXE.x87	Packed 64bit	FP_COMP_OPS_EXE.SSE_DOUBLE_PRECISION
		Packed 32bit	FP_COMP_OPS_EXE.SSE_SINGLE_PRECISION
		Scalar 64bit	FP_COMP_OPS_EXE.SSE_FP_SCALAR
		Scalar 32bit	FP_COMP_OPS_EXE.SSE_FP_SCALAR
2nd Generation Intel Core architecture (a.k.a SandyBridge)	FP_COMP_OPS_EXE.X87	Packed 64bit	FP_COMP_OPS_EXE.SSE_PACKED_DOUBLE
		Packed 32bit	FP_COMP_OPS_EXE.SSE_PACKED_SINGLE
		Scalar 64bit	FP_COMP_OPS_EXE.SSE_SCALAR_DOUBLE
		Scalar 32bit	FP_COMP_OPS_EXE.SSE_SCALAR_SINGLE

Tableau 2 : Les évènements PMU sont utilisés pour comptabiliser la charge des opérations en virgule flottante.

Génération de processeur	Noms des événements processeur	
	FP operations using AVX	
2e Generation Intel Core architecture (SandyBridge)	Packed 64bit	SIMD_FP_256. PACKED_DOUBLE
	Packed 32bit	SIMD_FP_256. PACKED_SINGLE
	Note : *AVX-256 groupés peuvent être comptés comme un et compteront comme SIMD FP 128.*	

Tableau 3 : Les évènements PMU sont utilisés pour comptabiliser la charge des opérations en virgule flottante utilisant AVX.

__itt_pause() (*interrompt la collecte*) et *__itt_resume()* (relance la collecte). Référez vous à la documentation de VTune Amplifier XE pour comprendre comment utiliser les API utilisateur. VTune Amplifier XE sur processeur Xeon est configurable comme montré dans la Figure 5 sur un système basé sur Core i7 (x980) : [Fig.6].

UTILISATION DES REGISTRES X87

Cet exemple est compilé en mode release (niveau d'optimisation fixé à 0x) sur Windows utilisant Visual Studio. L'application renvoie ce qui suit quand elle est analysée avec VTune Amplifier XE [Fig.7]. Les résultats de la figure 6 montrent comment le compilateur génère le code. Dans cette exécution, nous voyons clairement que nous n'avons récolté que des échantillons sur des opérations FP utilisant les registres x87. Si nous entrons les nombres dans la formule :

Formule MFLOPS = FP_COMP_OPS_EXE.FP / 1x10^6 / Temps Ecoulé

Temps Ecoulé = CPU_CLK_UNHALTED.THREAD / Fréquence Processeur / Nombre de coeurs. Soit :

Temps Ecoulé = 607,652,000,000.00 / 3.33 x 10^9 / 12 = 15.206 secs

MFLOP = 18,470,000,000.00 / 1x10^6/ 15.206 secs = 1214.652 Mflops

UTILISATION DES REGISTRES SSE

Maintenant, regardons la même application en utilisant les registres SSE. Si nous compilons l'application avec le compilateur Intel version 12.0, nous constatons les résultats suivants sous VTune Amplifier XE.

La première chose à noter dans les résultats affichés dans la figure 7, est la différence des noms de fonction là où se produit l'échantillonnage. Dans l'exemple précédent (Figure 6), nous obtenions les échantillons dans la fonction matrixMultiply, mais maintenant nous les voyons dans la fonction threadPool. Cela sera plus clair en entrant plus avant dans le détail de la fonction. Il est simple de constater comment le total des tops d'horloge consommés par chaque tâche est réduit d'environ 50 milliards de cycles (version utilisant les registres x87) à 5 milliards dans la version actuelle. Notez cependant que cette amélioration n'est pas uniquement due à l'utilisation de la vectorisation ou des registres SSE. D'autres optimisations

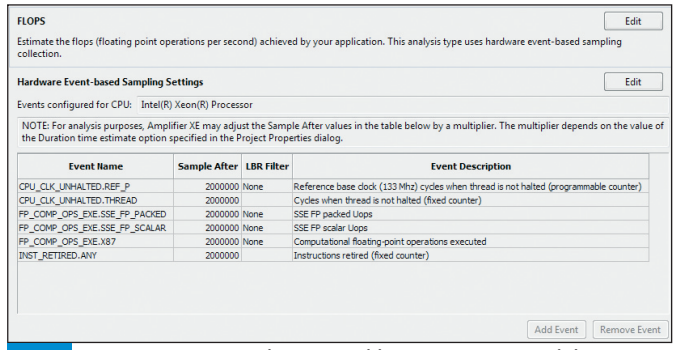


Fig.6 *Type d'analyse spécifique, créé pour mesurer les évènements PMU pertinents.*

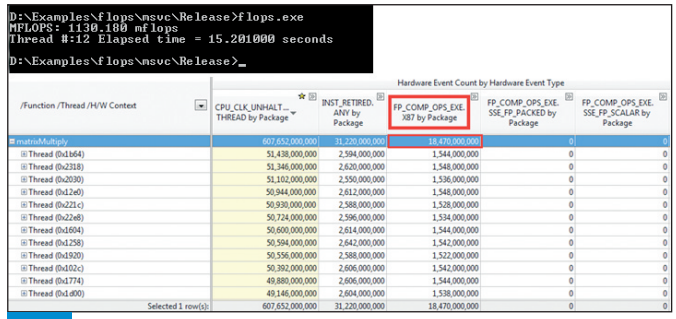


Fig.7 *Montre les résultats d'une analyse VTune Amplifier XE pour une application tirant parti des registres x87. La fonction matrikMultiply() est exécutée presque également par toutes les tâches.*

effectuées par le compilateur Intel ont contribué à cette amélioration.

L'évènement FP_COMP_OPS_EXE.SSE_DOUBLE_PRECISION est multiplié par 2 car deux opérations en virgule flottante double précision, groupées, peuvent être exécutées sur des registres 128 bits XMM. Pour des opérations en virgule flottante simple précision, le décompte total des opérations en virgule flottante simple précision doit être multiplié par 4.

Formule MFLOPS = 2 * FP_COMP_OPS_EXE.SSE_DOUBLE_PRECISION / Temps Ecoulé

Temps Ecoulé = CPU_CLK_UNHALTED.THREAD / Fréquence Processeur / Nombre de cœurs

Temps Ecoulé = (66,178,000,000 / 3.33 x10^9 / 12) = 1.656 secs

MFLOPS = 2 * FP_COMP_OPS_EXE.SSE_DOUBLE_PRECISION / 1 x 10^6 / 1.656 secs = 11053.140 mflops

Pour plus d'informations, connectez-vous au site :
http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe/ .

Levent **Akyil** - *Ingénieur logiciel senior Intel*

COMSOFT direct

Bechtle's Software Specialist

Un développement plus rapide Des applications plus rapides

INTEL PARALLEL STUDIO XE

La suite performance pour les développeurs C/C++ et Fortran

Intel® C++ Studio XE fournit une solution performante pour les développeurs C/C++ et fortran, Windows et Linux.
La suite tout-en-un contient les compilateurs C/C++ et Fortran, librairies, ainsi que des outils de vérification d'erreurs, sécurité et profiling.

Accroître la performance et la scalabilité

Composant de la suite Intel Parallel Studio XE, Intel® VTune™ Amplifier XE est un puissant outil de profiling threading et performance qui traite le comportement des applications séries et parallèles afin d'en améliorer la performance et la scalabilité. L'analyse Locks and Wait identifie l'une des causes les plus communes de ralentissement d'un programme parallèle : attendre trop longtemps un objet synchronisation (lock). Attendre est acceptable seulement si tous les cœurs travaillent. En revanche, la performance en souffre si l'attente se produit alors que des cœurs sont sous utilisés.

Retrouvez toutes les suites Intel chez Comsoft

Composants	Intel Parallel Studio XE	Intel C++ Studio XE	Intel C++ Composer XE	Intel Fortran Composer XE	Intel Composer XE Suite	Intel Cluster Studio
Intel C++ Compiler Professional Edition	✔	✔	✔		✔	✔
Intel Fortran Compiler Professional Edition	✔			✔	✔	✔
Intel Compiler Suite	✔				✔	✔
Intel Integrated Performance Primitives	✔	✔	✔		✔	✔
Intel Threading Building Blocks	✔	✔	✔		✔	✔
Intel Math Kernel Library	✔	✔	✔	✔	✔	✔
Intel MPI Library						✔
Intel Trace Analyzer and Collector						✔
Intel Cluster toolkit						✔
Intel Cluster Toolkit Compiler edition						✔
Intel VTune Performance Analyzer and Thread Profiler	✔	✔				
Intel Thread Checker	✔	✔				

Intel Volume Program

Gérez vos licences et réalisez des économies

Intel Volume Program est un système basé sur des points destinés aux utilisateurs qui souhaitent que leur revendeur gère tout ou une partie de leurs licences via un portail dédié.
Gérez aisément vos licences et dates d'expiration de maintenance grâce au portail Volume Licensing : générez des clés, visualisez les utilisateurs des licences... Et bien plus encore.

Retrouvez sur notre site de nombreuses ressources sur les produits Intel : Vidéos, tutoriaux...

http://www.comsoft-direct.fr/intel-xe

Contactez votre conseiller Comsoft 100 % Intel : Stéphane Baudry

Tél. 04 97 21 58 54 - stephane.baudry@comsoft.fr

13
OCTOBRE
2011

C++ DAY

Organisé par : intel + Microsoft

UN ÉVÈNEMENT C++
À NE PAS RATER !!

Une journée pour découvrir les dernières innovations matérielles et logicielles autour du C++.

keynote_speakers

Boris Jabes
Senior Program Manager
Visual C++ team
Microsoft Corporation

Jean-Pierre Duplessis
Architect, Microsoft Visual Studio,
Microsoft Corporation

James Reinders
Software Evangelist
Intel Corporation

Crytek
Development studios
for interactive entertainment

INFORMATIONS ET INSCRIPTION

WWW.MSDN.FR/CPPDAY

Télécharger l'application
pour votre téléphone
http:/ /gettag.mobi

Campus Microsoft - Centre de conférence
41 Quai du Président Roosevelt - Issy-les-Moulineaux

Construisez vous-mêmes l'internet des objets !

HACKING

Voilà des années qu'on nous promet l'internet des objets – sorte d'évolution du réseau dans lequel des objets communiqueraient entre eux de manière intelligente… On attendait la domotique, les robots, les voitures intelligentes… Et pourtant, le résultat fut maigre. Quelques cadres photos, et une série de lapins fort mignons… mais ma cafetière ne twitte pas, mon frigo ne fait pas les courses tout seul, et mon chauffage n'adapte pas encore sa puissance à la température extérieure (en tout cas chez moi !). Pourtant ce n'est pas si compliqué techniquement, il faut juste les bons outils. Démonstration par l'exemple dans cet article, en utilisant des technologies open source.

Nous allons hacker un climatiseur en y rajoutant une sonde de température, dont nous nous servirons pour l'allumer ou l'éteindre automatiquement, et accessoirement, poster ces informations sur un service de publication de flux d'objets connectés : OpenSense. Cela nous permettra par exemple de surveiller les courbes de température de la maison, les phases d'allumage et d'extinction du climatiseur, et pourquoi pas, le piloter à distance !

INTRODUCTION AUX TECHNOLOGIES UTILISÉES

Le .NET MicroFramework

Le .NET MicroFramework est un projet open source de Microsoft, hébergé par le groupe Visual Studio, qui vise à amener C# et la puissance du framework .NET aux microcontrôleurs, particulièrement ceux qui ne sont pas en mesure de faire tourner un système d'exploitation complet. C'est donc une version ultra-légère et bootable de la CLR qui offre au développeur .NET la possibilité d'écrire du code embarqué à partir d'un environnement qu'il connaît ! Le kit de développement du .NET MicroFramework est livré sous licence Apache 2.0, ainsi que le « porting kit » qui permet de l'adapter à de nouvelles architectures. L'outil de développement est Visual Studio : la version Express suffit.

On retrouve dans le MicroFramework les classes du framework .NET, même si elles ont été particulièrement « dégraissées » pour pouvoir tenir dans quelques Ko de RAM. Il y a également des classes permettant de piloter des entrées/sorties analogiques et numériques des microcontrôleurs, ainsi que les ports de communications et les interfaces classiques : SPI, I2C, UART, PWM, etc. Dans cet article nous verrons quelques-unes d'entre elles. Le kit de développement du MicroFramework se télécharge sur http://www.netmf.com. Il faudra en plus télécharger le kit de développement de la carte microcontrôleur qu'on utilisera, en l'occurrence la carte Netduino, sur http://www.netduino.com.

La carte Netduino

Pour ceux qui ne souhaitent pas s'attaquer à la création complète d'une carte électronique (c'est-à-dire vraisemblablement 90% d'entre nous) il existe des kits « tout faits » qui permettent de faire tourner le .NET MicroFramework. Pour cet article, j'ai choisi de parler du kit Netduino. C'est un hardware opensource lui aussi (c'est-à-dire que les données de conceptions, schémas et layouts, sont disponibles, copiables et modifiables), qui a le mérite d'être bon marché (34$ dans la version de base, 49$ dans la version plus qui dispose d'un port Ethernet) et qui est entièrement compatible avec la célébrissime carte Arduino, et notamment tous ces « shields » développés par la communauté qui permettent d'en étendre les capacités. La raison de préférer le Netduino à l'Arduino est pour moi toute simple : seul le premier permet d'utiliser le MicroFramework, et comme vous allez le voir dans la suite de l'article, ça permet d'aller très vite dans le développement, sans nécessairement être un expert en électronique ! [Fig.1].

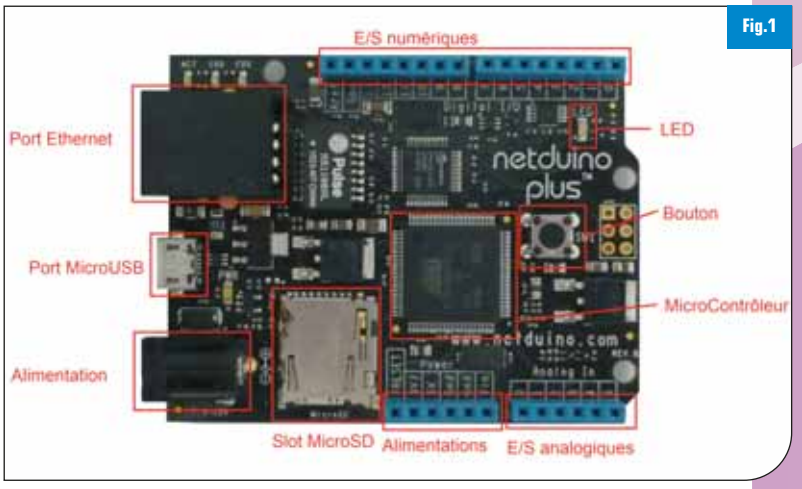


Fig.1

Le service Open.Sen.se

Open.Sen.se est un service « cloud » qui permet de connecter des sources de données (comme par exemple une sonde de température) a des applications qui vont la consommer (à travers une API, ou tout simplement un navigateur web ou une application mobile par exemple). C'est le créateur du lapin mentionné dans l'introduction de cet article qui est derrière ce nouveau service, pour l'instant en bêta. Vous pouvez demander un accès à la bêta d'OpenSense sur http://open.sen.se.

MON PREMIER OBJET CONNECTÉ : LE CLIMATISEUR

Nous voulons créer un climatiseur intelligent, c'est-à-dire qui s'allume et s'éteint en fonction de la température, qui publie son statut sur internet, et qui est pilotable à distance. D'un point de vue matériel, il nous faut donc ajouter sur la carte Netduino une sonde de température, et de quoi piloter l'allumage du climatiseur.

Le montage : la sonde de température

Pour ce montage, on utilise un capteur de température tout à fait basique, le LM35DZ. C'est en fait une résistance qui varie en fonction de la température. On l'alimente sur le 5V de la carte Netduino, et on branche la patte de sortie directement sur une entrée analogique. L'avantage de cette entrée est qu'elle dispose d'un convertisseur analogique-numérique intégré, c'est-à-dire qu'à partir d'une tension variable, on ressort une valeur entre 0 et 1023, ou tout autre intervalle qu'on aura pris soin de paramétrer. On peut lire cette valeur en 2 lignes de code avec le .NET MicroFramework :

```
AnalogInput tempSensor = new AnalogInput(Pins.GPIO_PIN_A0);
int temperature = CalculateTemperatureFromSensor(tempSensor.Read());
```

Ici la méthode CalculateTemperatureFromSensor est utilisée pour transformer la valeur entre 0 et 1024 en une valeur de température en °C. A vous de voir quelle formule fonctionne pour vous, en expérimentant ! C'est relativement simple car dans des conditions normales (température ambiante) la sortie du LM35DZ est linéaire. Chez moi c'est une simple division par 3.
Pour lire régulièrement cette température on va utiliser un timer qui se déclenchera toutes les 30 minutes. Si la température passe au-dessus d'un certain seuil, on allumera le climatiseur et on l'éteindra quand elle repasse en dessous.

```
Timer tempTimer = new Timer(PublishTempTask, null, 0, 1800000);
```

PublishTempTask étant la méthode à exécuter (qu'on étudiera plus tard), 1800000 étant l'intervalle en millisecondes.
Il faut maintenant brancher la carte Netduino au climatiseur, et on va essayer de se brancher directement sur le bouton d'allumage… Attention, selon l'équipement que l'on hacke, tous les boutons ne sont pas reliés de la même manière ! En effet certains sont déjà branchés sur des circuits numériques en 3.3V ou 5V, mais d'autres sont directement sur des circuits analogiques voire carrément le secteur ! Dans ce cas il faut utiliser un petit montage avec un relais ou un optotriac, et il faut faire attention à la puissance qu'on veut piloter : pour un bouton de radiateur par exemple, ça peut faire quelques milliers de watts, et tous les composants électroniques ne sont pas capables d'encaisser cela.
Dans le cas du climatiseur, une simple mesure au multimètre nous indique que l'interface de contrôle n'est pas sur le secteur mais bien sur un circuit numérique. Nous allons nous brancher en parallèle du bouton d'allumage de façon à ne pas altérer son fonctionnement normal. Le transistor est un composant qui permet d'agir comme un interrupteur, si on l'utilise dans un mode qu'on appelle « en commutation » ou « saturé bloqué ».

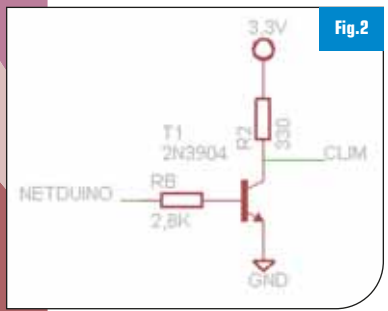
Petit rappel sur le transistor

Un transistor est un composant à 3 pattes, respectivement le collecteur, l'émetteur et la base. Injecter un courant dans la base polarise la jonction et permet le passage du courant du collecteur vers l'émetteur. Lorsqu'il est utilisé en commutation, on va utiliser cette propriété pour soit « bloquer » le transistor, c'est-à-dire qu'il n'y aura pas de courant circulant entre le collecteur et l'émetteur, soit en mode saturé, c'est-à-dire que la base sera suffisamment alimentée pour que le courant passe et la résistance collecteur-émetteur minimale. Dans ce mode, le transistor fonctionne comme un interrupteur ! L'avantage de ce mode de fonctionnement est que le courant dans la base, permettant de saturer le transistor est très faible (de l'ordre de 10mA) et peut largement être fourni par un microcontrôleur.
Ce montage, branché en parallèle du bouton de démarrage du climatiseur, nous permettra de démarrer ce dernier à partir d'une simple sortie numérique de la carte Netduino [Fig.2,3 et 4].
Le pilotage de cette sortie numérique est enfantin avec le .NET MicroFramework. On déclare sur quelle sortie de la carte Netduino on se branche, et on y écrit un booléen pour indiquer si on veut qu'elle soit à l'état haut ou bas.

```
OutputPort clim = new OutputPort(Pins.GPIO_PIN_D2, false);
clim.Write(true);
```

On dispose donc maintenant d'une sonde de température dont on peut lire la valeur, et d'une sortie permettant de piloter l'allumage du climatiseur. Notre partie « commande » est donc en place [Fig.5].

Fig.2



Transistor en commutation

Fig.3

Fils branchés en parallèle de l'interrupteur, reliant la carte Netduino au panneau de contrôle

Fig.4

Panneau de contrôle du climatiseur – le bouton de démarrage est celui de gauche

Il nous reste encore à poster ces informations sur open.sen.se, cela nous permettra d'avoir des courbes de températures et des informations d'utilisation du climatiseur (à vous de voir ce que vous en ferez après ;) On pourrait coupler cela à une application mobile permettant de piloter « manuellement » le climatiseur par exemple)

La connexion à Internet, et l'accès au service OpenSense

La carte Netduino Plus dispose d'une interface Ethernet que nous allons utiliser pour connecter le climatiseur au réseau. Le .NET MicroFramework nous renvoie la liste des interfaces réseau avec une simple méthode :

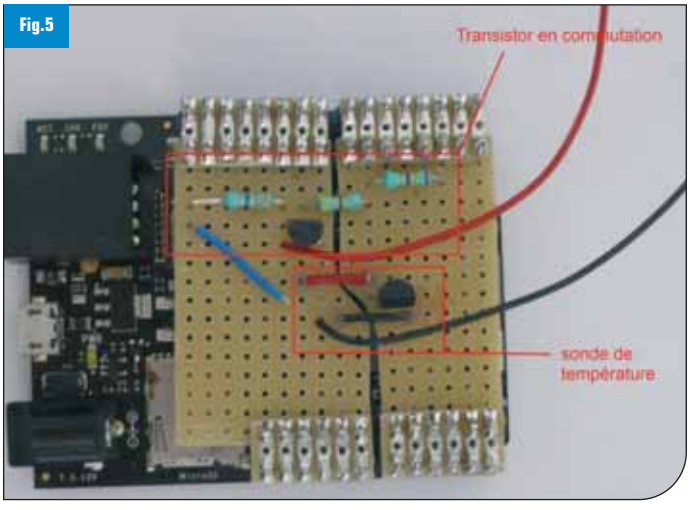
```
NetworkInterface[] ifaces = NetworkInterface.GetAllNetworkInter
faces();
```

Le .NET MicroFramework supporte évidemment le protocole DHCP, mais dans notre cas, s'agissant d'un équipement connecté en permanence, nous allons lui imposer une configuration statique :

```
NetworkInterface eth0;
if (ifaces.Length != 0)
{
    eth0 = ifaces[0];
   eth0.EnableStaticIP(«192.168.1.100», «255.255.255.0», «192
.168.1.1»);
   string[] DnsServers = new string[2] { «109.0.66.20», «109
.0.66.10» };
    eth0.EnableStaticDns(DnsServers);
}
```

Et nous voilà connecté à Internet ! Reste à poster les informations sur le service open.sen.se.
Vous pouvez découvrir le service OpenSense tout simplement sur http://open.sen.se. Vous pourrez y demander un accès à la bêta, et cet accès vous permettra d'accéder aux documentations de l'API, ainsi que de publier des flux (connecter des objets qui envoient de l'information) et créer des applications (ou bien utiliser les applica-

Fig.5



Montage de la sonde et du transistor en commutation sur la carte Netduino

Hacking hardware quelques conseils

Ne manipulez jamais un appareil branché ! Le conseil peut paraître évident, mais il faut toujours avoir en tête qu'on ne manipule jamais un appareil lorsqu'il est branché sur le secteur. Une bonne châtaigne au 220V peut, selon les personnes et les circonstances, avoir des conséquences oscillant entre une petite claque chatouilleuse, et la mort.

Lorsque vous utilisez des outils de type Dremel ou perceuse, portez une protection sur les yeux. Même si cela donne un air débile, on a l'air moins débile comme ça qu'avec un débris de lame ou de foret cassé à plusieurs milliers de tours-minute dans l'œil…

Dégagez votre espace de travail et éventuellement mettez-vous debout pour y travailler. Ne manipulez jamais sur vos genoux, ou sur un coin de table basse. Moins votre espace est dégagé et plus vous avez de chance que la prochaine maladresse que vous fassiez vous coûte cher. Qui est volontaire pour tester l'effet provoqué par la chute d'un fer à souder à 350° sur sa cuisse ?

Prévoyez la possibilité de « retirer » votre hack : 100% du temps, ouvrir un appareil et le bidouiller revient à en perdre la garantie. Et même si vous êtes en mesure de retirer le hack, ne croyez pas que le SAV se laissera berner, car vous laisserez forcément des traces. Mais vous aurez peut-être envie un jour de changer cet appareil par un autre plus puissant, de le jeter, ou de le céder à votre grand-mère, et dans la plupart de ces cas, vous serez content de pouvoir retirer votre hack et faire retrouver à votre appareil son fonctionnement initial, ou tout simplement réutiliser le matériel de ce hack sur un autre appareil. Prévoyez donc une stratégie de sortie ! Dans ces conditions essayez d'altérer au minimum l'aspect visuel de l'appareil que vous hackez. A condition bien sûr, que le but du hack ne soit pas justement d'en altérer l'aspect visuel !

Soignez la finition : bidouiller un appareil pour en comprendre et éventuellement en modifier le fonctionnement, c'est sûr que c'est la partie « marrante ». Mais laisser ce hack en fonctionnement après, c'est encore plus satisfaisant, et pour le laisser brancher, il faut s'assurer qu'il est sécurisé (essayez de ne pas mettre le feu chez vous), et qu'il n'est pas hideux (très important pour le W.A.F.) Prévoyez dès la phase de réflexion de votre hack comment vous allez le cacher, ou l'exposer, avec un boîtier par exemple.

Prenez votre temps, et soyez persévérant : ce dernier conseil est probablement le plus important. Le hacking hardware prend du temps. L'article que vous venez de lire à nécessité plusieurs heures de travail, de mesures, quelques schémas, quelques aller-retour au magasin de composants… Prendre son temps et revenir plusieurs fois sur les décisions prises est un processus normal, particulièrement quand on décide de potentiellement ouvrir et charcuter un équipement électroménager onéreux. Ne vous découragez pas, le résultat en vaut toujours la peine !

tions existantes, comme la publication de ce flux sur Facebook par exemple).
OpenSense implémente des webservices REST c'est-à-dire qu'on envoie des requêtes http, et que le service nous répond avec des données sérialisées au format JSON.
Le .NET MicroFramework étant une implémentation ultra-légère du framework complet, on retrouve les classes bien connues WebRequest et WebResponse. Il n'existe en revanche pas de classes pour la sérialisation et la désérialisation JSON, donc il faut la coder, mais c'est extrêmement simple. Nous allons coder une petite classe qui représentera une valeur numérique à poster sur le flux OpenSense et qui gèrera la sérialisation dans une méthode :

```
public class NumericFeedItem : IFeedItem
{
    public int Value { get; set; }

    public NumericFeedItem(int val)
    {
        Value = val;
    }

    public string ToJson(int FeedId)
    {
       return «{\r\n\t\»feed_id\»: « + FeedId.ToString() + «
,\r\n\t\»value\»: « + Value + «\r\n}»;
    }
}
```

L'argument FeedId représente l'identifiant du flux que vous aurez récupéré sur votre panneau de contrôle OpenSense.
Maintenant qu'on peut formater les éléments il faut les poster sur le flux :

```
HttpWebRequest req = (HttpWebRequest)WebRequest.Create(«http
://api.sen.se/events/?sense_key=» + ApiKey);
req.Method = «POST»;
req.ContentType = «application/json»;

byte[] postdata = System.Text.Encoding.UTF8.GetBytes(item.To
Json(this.Id));
req.ContentLength = postdata.Length;

using (Stream s = req.GetRequestStream())
{
    s.Write(postdata, 0, postdata.Length);
}

WebResponse resp = req.GetResponse();
```

Et voilà notre application toute prête à lire une température, allumer ou arrêter le climatiseur, et envoyer l'information sur OpenSense ! Comme quoi, ce n'est pas si compliqué l'internet des objets ! Nous n'avons pas encore couvert la partie pilotage du climatiseur, mais la solution est toute simple : il suffit de faire lire régulièrement, par exemple toutes les 30 secondes, au climatiseur un flux sur OpenSense, flux qui contiendrait un booléen, représentant l'état « allumé » ou « éteint ». Le climatiseur comparerait ce flux à son état, et agirait en conséquence. Ensuite il suffirait d'aller changer l'état de ce flux depuis par exemple un site web ou une application mobile… Cet exercice constituera vos devoirs de vacances de hackers hardware ! Retrouvez l'intégralité du code présenté dans cet article et même plus sur le blog de l'auteur.

■ Pierre **Cauchois**
Relations avec les développeurs - Microsoft France
@pierreca - http://blogs.msdn.com/pierreca

Programmez sur Kinect avec WPF OpenNI et NITE

1re partie

Sortie en novembre 2010 et anciennement connue sous le nom de code « Natal », la Kinect s'est vendue à 8 millions d'unités durant les deux premiers mois de sa commercialisation. En attendant le SDK officiel de Microsoft pour le développement Kinect sous Windows (prévu fin juin), nous allons parcourir dans cet article des possibilités de développement avec le framework OpenNI et NITE.

Développée à l'origine par une société israélienne PrimeSense, la Kinect est équipée d'un projecteur infrarouge inondant la pièce de points invisibles à l'œil nu. Grâce à une caméra infrarouge, elle est capable de calculer la profondeur de chaque point en fonction de l'intensité du rayonnement réfléchi, et d'obtenir ainsi un nuage de points de la profondeur de la scène. Elle renferme aussi une camera RVB classique qui permet à la Kinect d'être vue comme une caméra 3D, où en plus d'avoir l'information de la couleur sur chaque pixel dans une image, nous récupérerons l'information sur la profondeur de chacun d'entre eux dans la scène filmée [Fig.1 et 2].

En attendant la réponse officielle de Microsoft avec le SDK Kinect (disponible depuis fin Juin), nous allons nous intéresser au framework OpenNI développé par PrimeSense, Willow Garage, Side Kick et plus récemment Asus.

OPENNI

Distribué sous licence LGPL, OpenNI permet de concevoir des applications basées sur l'« interaction naturelle » ou Natural Interaction (OpenNI : Open Natural Interaction). Ce framework multi-langage et multiplateforme fournit toute la tuyauterie pour faire communiquer d'une part des capteurs et d'autre part des « percepteurs » responsables du traitement des données des différents capteurs [Fig.3]. Les capteurs pouvant être des caméras classiques (RGB), infrarouge, capteur 3D ou audio (microphones)… et bien sûr la Kinect ! Les percepteurs vont permettre de traiter les données fournies par les capteurs pour détecter les mouvements, suivre une main, reconstituer le squelette d'un corps, analyser la scène pour extraire le fond, détecter les personnes ou encore déterminer les coordonnées du sol.

OpenNI définit le concept de « Production Node », une unité logique responsable de fournir des données. Un nœud de production peut utiliser des nœuds enfants et être utilisé par des nœuds au-dessus de lui. On distingue deux types de nœuds de production : ceux de type « sensor » (le nœud produit les données acquises d'un capteur) et ceux de type « middle-ware » (le nœud produit des données issues d'un traitement). Par exemple, le suivi d'une main est réalisé par le nœud de production de type middle-ware «Hand Point Generator » qui lui-même utilise le nœud de production « Depth Generator » de type « sensor » implémenté au niveau d'un capteur comme la Kinect par exemple. On définit donc des « Production Chains » entre différents nœuds de production fournissant les données d'un capteur (plus bas niveau) ou les données d'un traitement (plus haut niveau). Cette architecture modulaire permet d'implémenter différents « middle-ware » ou devices respectant l'API définie par OpenNI. Car OpenNI, en soi, ne définit que les interfaces et non leurs implémentations. PrimeSense met à disposition les drivers et modules de la Kinect pour OpenNI en open-source sur le site https://github.com/avin2/SensorKinect permettant d'exploiter la Kinect comme un capteur dans l'architecture d'OpenNI.

NITE

Nite, quant à lui, est un jeu de modules « middle-ware » pour OpenNI et de contrôles proposés en licence gratuite. Ces modules implémentent toutes les interfaces OpenNI avec les « algorithmes maison NITE » et proposent aux développeurs différents contrôles de haut niveau. L'architecture est la suivante : [Fig.4].

On y trouve au niveau le plus bas l'implémentation des modules OpenNI par PrimeSense qui définissent des nœuds de production assurant la détection des mouvements, mains, personnes, scènes, etc. Ces modules sont exploités par le framework OpenNI. La couche « Control Management » permet de lier les flux de données des nœuds de production dans OpenNI avec les contrôles NITE au niveau le plus haut. Parmi les contrôles disponibles, nous trouverons le Push Detector, Steady Detector, Wave Detector, etc.

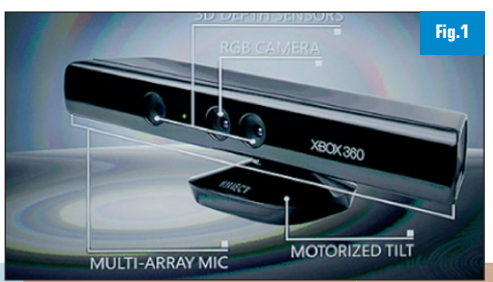


Fig.1


Fig.2

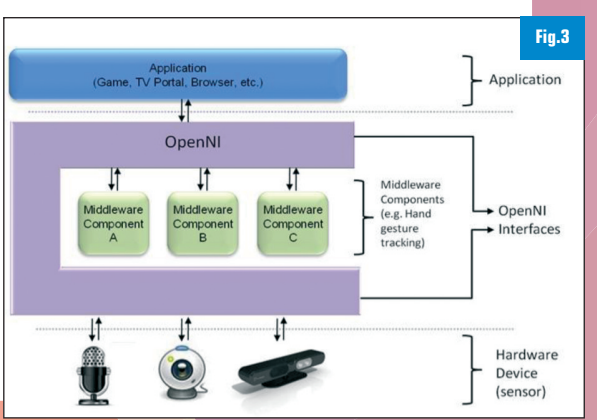


Fig.3

Installation

Pour pouvoir développer avec sa Kinect, il vous faudra l'adaptateur USB permettant de la connecter sur votre ordinateur. Cet adaptateur est inclus dans le package de la Kinect achetée toute seule (mais pas dans les packages incluant la Xbox slim). Vous pourrez sinon trouver des adaptateurs USB vendus seuls sur eBay pour une vingtaine d'euros. Avant de connecter la Kinect, il vous faudra installer les drivers.

1/ Installer OpenNI

Commencez tout d'abord à récupérer les dernières versions « unstable » sur le site officiel d'OpenNI (http://www.openni.org/downloadfiles/openni-binaries/20-latest-unstable) et téléchargez la version développeur 32bits (même sous Windows 64bits, utilisez la version 32bits pour ne pas avoir à recompiler le driver Kinect pour 64bits) [Fig.5].

2/ Installer le driver Kinect

Sur Github, téléchargez la dernière version de SensorKinect à l'adresse https://github.com/avin2/SensorKinect et lancez l'installation des drivers (fichier « SensorKinect-Win-OpenSource32-5.0.1.msi » du répertoire « Bin ») en acceptant bien sûr la confirmation de l'installation d'un driver non-signé [Fig.6].

3/ Connecter la Kinect

En connectant votre Kinect sur un port USB au travers de l'adaptateur, Windows devrait pouvoir installer la Kinect correctement [Fig.7].

4/ Installer NITE

Sur le site d'OpenNI, dans la catégorie « Middle-ware unstable », téléchargez et installez la version développeur de NITE. Lors de l'installation il vous faudra inscrire le numéro de série fourni gratuitement par PrimeSense **0KOIk2JeIBYClPWVnMoRKn5cdY4=** [Fig.8].

5/ Configuration

Avant de pouvoir tester, il faut ajouter le numéro de série de PrimeSense dans les fichiers de configuration d'OpenNI et NITE :

- C:\Program Files (x86)\OpenNI\Data\SampleConfig.xml
- C:\Program Files (x86)\PrimeSense\NITE\Data\ Sample-Scene.xml
- C:\Program Files (x86)\PrimeSense\NITE\Data\ Sample-Tracking.xml
- C:\Program Files (x86)\PrimeSense\NITE\Data\ Sample-User.xml

En ajoutant ou remplaçant la section suivante :

```
  <Licenses>
    <License vendor=»PrimeSense» key=»0KOIk2JeIBYClPWVnMoRK
n5cdY4=»/>
  </Licenses>
```

Tester !

Il ne vous reste plus qu'à tester ! Pour cela commençons par les exemples d'OpenNI dans le répertoire « C:\Program Files (x86)\OpenNI\Samples\Bin\Release » et lancez « NiSimpleViewer.exe » [Fig.9, 10 et 11]. Cet exemple exploite à la fois le « Depth Generator » (permettant de calculer la profondeur) et le « Image Generator » (permettant de récuperer l'image RVB de la caméra). Changez de mode en appuyant sur les touches 1, 2 et 3. Dans ce même dossier vous trouverez aussi des exemples en .NET exploitant le wrapper .NET « OpenNI.Net ». L'exemple « UserTracker.net.exe », développé en C# met en œuvre le « User Generator » capable de vous fournir les coordonnées dans l'espace du skeleton d'un utilisateur : [Fig.12]. Vous trouverez aussi des exemples exploitant les contrôles de NITE dans le dossier « C:\Program Files (x86)\PrimeSense \NITE\Samples\Bin\Release » ainsi que le wrapper .NET pour les contrôles NITE ici nommé « XnVNITE.net_1_3_1.dll ». Vous y retrouvez des contrôles « prêts à l'emploi » comme par exemple :

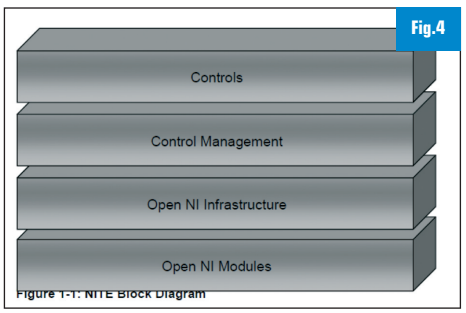
- Push Detector : permettant de détecter un « push » avec la main

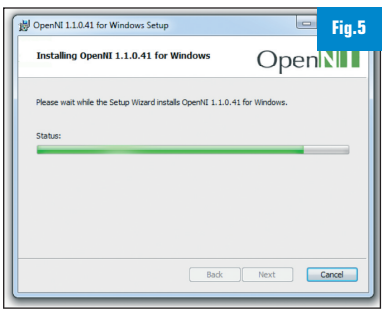
Ou encore les SelectableSliders 1D et 2D, le Circle Detector, etc [Fig.13 et 14], [Fig.15]

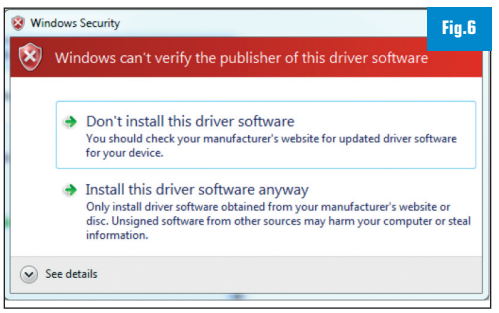
TRANSFORMEZ VOS APPLICATIONS WPF4 « TOUCH LESS » AVEC LA KINECT

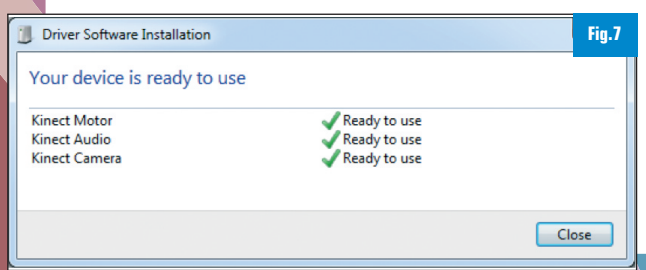
Arrivé avec le .NET 4.0, WPF4 intègre de manière native la gestion du multi touch pour le développement d'applications « Touch ». Fort de l'expérience Microsoft Surface qui intégrait alors son propre SDK pour le développement (rendant difficile le portage sur un autre support que la Surface), WPF4 propose une API générique ajoutant une couche d'abstraction entre l'application et le périphérique « Touch ». Il devient alors possible de développer sur un framework unique ses applications Touch en intégrant le toucher, les manipulations, les points de pivot, l'inertie, le panning, etc. Et cela sans se soucier du périphérique « Touch » en entrée (écran multi-touch, plusieurs souris, Surface v2 ou une Kinect par exemple !). Cela permet notamment le partage de code entre les applications Surface et Windows à l'image du projet « Microsoft Surface Toolkit » en profitant des contrôles comme le ScatterView dans ses applications WPF4 [Fig.16].

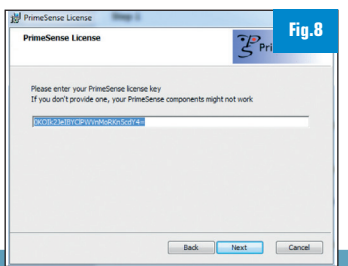
Avant de démarrer, éditez le fichier « Nite.ini » du répertoire « C:\Program Files (x86)\PrimeSense\NITE\Hands_1_3_1\Data »

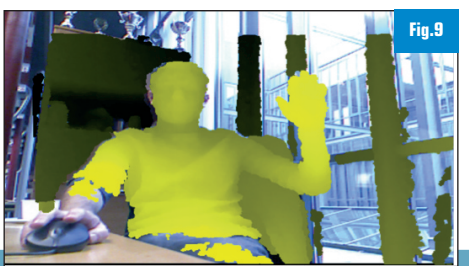












et dé-commentez les deux paramètres afin d'autoriser le tracking de plusieurs mains simultanément :

```
[HandTrackerManager]
AllowMultipleHands=1
TrackAdditionalHands=1
```

Mission 1 : Traquer la position des mains

Pour cela, définissons une classe « KinectHand » qui aura pour mission de lever des évènements lors de la « suppression» d'une main (sous-entendu lorsque l'on perd la main du tracking) et de sa mise à jour (comme par exemple son déplacement dans la scène). Nous n'aurons pas besoin de l'évènement de création en considérant que la 1re mise à jour fait suite à une création. Nous allons pour cela utiliser les contrôles fournis par NITE et plus particulièrement le WaveDetector permettant de détecter et de suivre une main en effectuant un « wave » à la Kinect (une sorte de « Coucou » avec la main !). Comme expliqué précédemment, il faut d'abord définir les « Production Nodes » à charger dans notre contexte OpenNI. Pour notre mission nous avons besoin du module « Depth » permettant de nous fournir l'information quant à la profondeur de chaque pixel de la scène, le module « Gesture » et « Hand » pour la détection et le tracking des mains. Le fichier de configuration définira cette chaîne de production de la façon suivante :

```
<ProductionNodes>
  <Node type=»Depth»>
    <Configuration>
      <MapOutputMode xRes=»640» yRes=»480» FPS=»30»/>
      <Mirror on=»true»/>
    </Configuration>
  </Node>
  <Node type=»Gesture» />
  <Node type=»Hands» />
</ProductionNodes>
```

Vous trouverez le fichier de configuration à l'emplacement « C:\Program Files (x86)\PrimeSense\NITE\Data\Sample-Tracking.xml » :

```
private readonly string SAMPLE_XML_FILE = @»C:\Program Files (x86)\PrimeSense\NITE\Data\Sample-Tracking.xml»;
```

Nous allons devoir définir les objets suivants :

```
// Contexte OpenNI
private Context context;
// Session NITE
private SessionManager sessionManager;
// Detecteur de «Wave»
private WaveDetector waveDetector;
// Filtre pour lisser les points
private PointDenoiser pointDenoiser;
// Thread de lecture
private Thread readerThread;
```

Et dans notre constructeur, les initialiser de la façon suivante :

```
// Création du contexte OpenNI
this.context = new Context(SAMPLE_XML_FILE);

// Création du SessionManager avec le «Wave» en focus et «Raise Hand» en quick focus (si déjà passé par le focus)
this.sessionManager = new SessionManager(this.context, «Wave», «RaiseHand»);

// Création du WaveDetector
waveDetector = new WaveDetector();

// Abonnements aux événements pour la mise à jour et destruction d'un point
waveDetector.PointUpdate += new EventHandler<HandEventArgs>(waveDetector_PointUpdate);
waveDetector.PointDestroy += new EventHandler<IdEventArgs>(waveDetector_PointDestroy);

// Création d'un filtre pour lisser les points
```


Fig.10


Fig.11


Fig.15

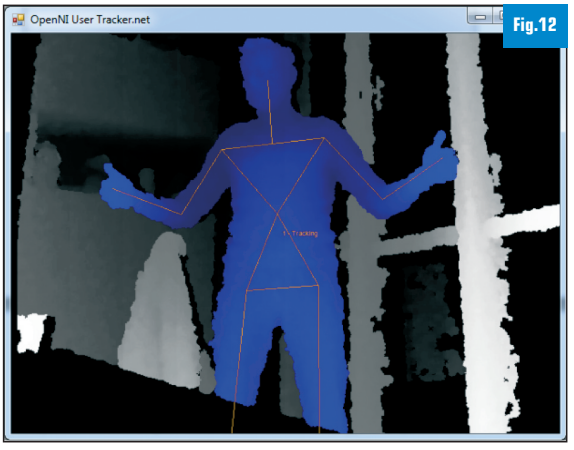


Fig.12

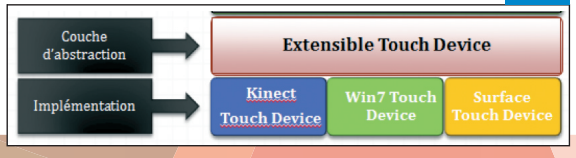


Fig.16

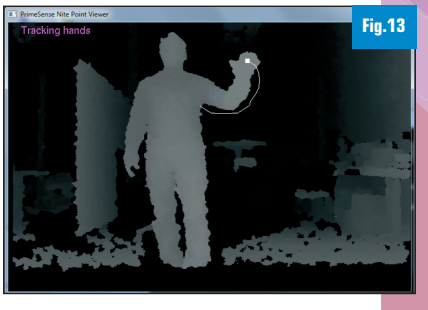


Fig.13

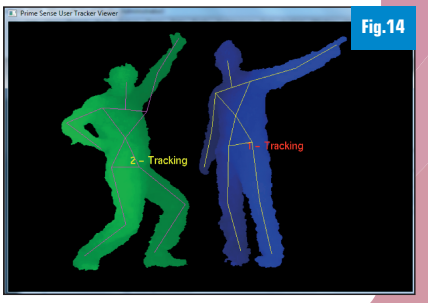


Fig.14

```
this.pointDenoiser = new PointDenoiser(20);

// Ajout du WaveDetector dans le PointDenoiser
this.pointDenoiser.AddListener(waveDetector);

// Définition du PointDenoiser comme listener de la session
this.sessionManager.AddListener(this.pointDenoiser);

// Lancement du thread de lecture
isRunning = true;
readerThread = new Thread(new ThreadStart(SpinInfinite));
readerThread.Start();
```

Notez que nous aurions pu aussi utiliser un FlowRouter pour agréger plusieurs détecteurs ou filtres (mais ici nous n'utilisons que le WaveDetector en passant par un filtre, l'utilisation du routeur n'est pas nécessaire). Définissons la méthode lancée par le thread de lecture :

```
private void SpinInfinite()
{
while (this.isRunning)
 {
// Mise à jour du contexte OpenNI
this.context.WaitAnyUpdateAll();
         this.sessionManager.Update(this.context);
}
}
```

Il ne reste plus qu'à réagir aux évènements du Wave Detector :

```
// Simple « Forward » l'événement

private void waveDetector_PointDestroy(object sender, IdEvent
Args e)
{
    if (HandPointDetroy != null)
    {
        this.HandPointDetroy(this, e);
    }
}

// « Forward » l'événement après correction des coordonnées
private void waveDetector_PointUpdate(object sender, HandEvent
Args e)
{
    if (HandPointUpdated != null)
    {
       Point3D newPoint = new Point3D(e.Hand.Position.X, e.
Hand.Position.Y, e.Hand.Position.Z);

        // Correction de l'origine du plan
        newPoint.X += this.Width / 2;
        newPoint.Y += this.Height / 2;

        // Limite max
        if (newPoint.X < 0)
        {
            newPoint.X = 0;
        }
        if (newPoint.Y < 0)
        {
            newPoint.Y = 0;
        }
        if (newPoint.X > this.Width)
        {
            newPoint.X = this.Width;
        }
        if (newPoint.Y > this.Height)
        {
            newPoint.Y = this.Height;
        }
        // Inversion de l'axe Y
        newPoint.Y = this.Height - newPoint.Y;

        this.HandPointUpdated(this, new HandPointEventArgs
(e.Hand.ID, e.Hand.UserID, newPoint));
    }
}
```

A noter que les propriétés Width et Height de cette classe sont définies aux valeurs de la « source », ici notre « Depth Generator » défini dans le fichier de configuration à 640 (width) sur 480 (height). Notez aussi que nous avons redéfini la classe d'argument ci-dessous afin de pouvoir modifier ses propriétés (ce qui n'est pas possible dans la classe originale « HandEventArgs »).

```
public class HandPointEventArgs : EventArgs
{
    public Point3D Position { get; set; }
    public int UserID { get; set; }
    public int ID { get; set; }

    public HandPointEventArgs(int id, int userId, Point3D point)
    {
        this.Position = point;
        this.UserID = userId;
        this.ID = id;
    }
}
```

Pour terminer, implémentez une méthode Dispose pour fermer votre thread de lecture :

```
public void Dispose()
{
this.isRunning = false;
this.readerThread.Join();
}
```

Le code propre à l'exploitation de la Kinect est déjà fini. Nous pouvons utiliser cette classe pour traquer les mains détectées par la Kinect en récupérant pour chacune d'elles l'ID de la main (unique), l'ID de l'utilisateur et la position dans l'espace (X, Y et Z). Reste à l'exploiter dans un TouchDevice ! Retrouvez le code source de cet article sur notre site.
La suite au prochain numéro.

■ Sébastien **Warin**
Technical Lead – xBrainLab - Microsoft MVP
http://sebastien.warin.fr - www.xbrainlab.com

Bioinformatique : les séquenceurs haut-débit

Depuis le milieu des années 2000, les séquenceurs haut-débit révolutionnent la biologie. Ils sont un formidable outil pour la recherche en biologie et serviront bientôt en médecine personnalisée. Traiter les données de ces séquenceurs demande des compétences pluridisciplinaires : connaissances biologiques, recherches théoriques en algorithmique de texte, et implémentations et technologies adaptées.

Le génome est une suite de bases A, C, G et T. Ce génome est codé dans l'ADN de chacune de nos cellules, contient des informations sur notre métabolisme, nos spécificités, et il est souvent la clé des maladies génétiques. En temps normal, toutes nos cellules ont exactement, ou presque, le même génome, il est à plus de 99% identique au sein de l'espèce humaine. Les génomes font de quelques milliers de bases pour les virus, et plusieurs millions pour les bactéries (4,6 Mb pour la bactérie Escherichia coli), à quelques milliards pour les organismes pluri-cellulaires (3,4 Gb pour l'homme). Le séquençage est la lecture de ce code. Les séquenceurs lisent des fragments de plusieurs bases appelées reads dont la longueur varie en fonction des technologies. Le génome s'obtient ensuite par assemblage des reads. Depuis 2005, les nouvelles technologies de séquençage ont révolutionné ce domaine, et posent des problèmes aux informaticiens.

Le séquençage d'ADN

Méthode de Sanger

La méthode de séquençage de Sanger a été mise au point en 1976. La séquence d'ADN est polymérisée en quatre ensembles de fragments de différentes tailles, chaque ensemble terminant sur une des 4 bases. Les fragments migrent ensuite dans un gel, ce qui permet de détecter leur longueur et de lire la séquence. Cette méthode a été optimisée et automatisée durant plus de 25 ans. C'est ainsi que, à partir de 1998, le premier génome humain a été séquencé. Un séquenceur produisait alors des reads de 500 à 600 bases dans 96 capillaires en environ 10 heures, c'est-à-dire environ 115 kb/jour [Fig.2].

Les séquenceurs haut-débit

Dans le début des années 2000, quelques études proposent des nouvelles technologies qui révolutionnent le séquençage. Il ne s'agit plus de mesurer la longueur de certains fragments, mais de lire directement la séquence d'ADN, base par base, par cycles successifs. Ces recherches ont été extrêmement vite transférées, et des séquenceurs haut-débit ont été proposés sur le marché à partir de 2004 (voir tableau ci-dessous). Toutes ces techniques traitent les bases une par une, en alternant des réactions chimiques couplées à une détection optique et des phases de nettoyage. Ces séquenceurs permettent de lire des reads dont la longueur varie en fonction des technologies.

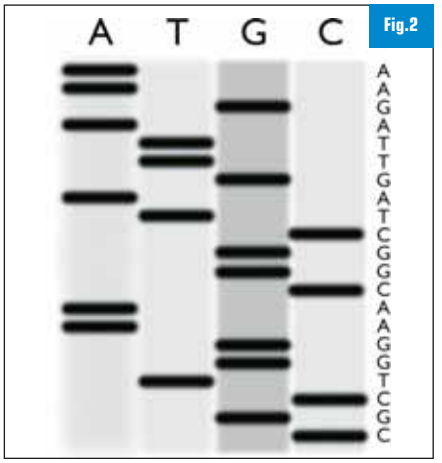


Fig.2

Séquençage par méthode de Sanger, produisant ici un read de 21 bases.

Coût et capacités

Par ces nouvelles techniques, les séquenceurs haut-débit ont permis une vraie rupture dans les quantités de données produites, qui font plus que décupler chaque année ! Ces séquenceurs coûtent aujourd'hui plusieurs centaines de milliers d'euros à l'achat, auxquels il faut ajouter le coût des réactifs. Une expérience (un run) nécessite quelques milliers d'euros de matériel, ainsi que plusieurs heures ou journées de préparation des échantillons [Fig.3]. Le plus grand centre de séquençage au monde, le Beijing Genome Institute (BGI), en Chine, possède actuellement 137 Illumina HiSeq~2000 et 27 AB Solid 4.0. En France, si le Genoscope, à Évry, reste le principal centre de séquençage, d'autres plateformes à Paris comme en région ont des capacités intéressantes, comme par exemple une dizaine de séquenceurs à Lille. De nouvelles technologies sont attendues dans les prochaines années pour améliorer le débit et la qualité des résultats, et baisser les coûts et le temps de préparation. Par exemple, la société Ion Torrent, rachetée par Roche en 2010, propose une technique de séquençage couplée à un microprocesseur sur silicium, et donc qui permet de se passer de l'optique.

Séquencer, pour quoi faire ?

Utilisation directe

Le séquençage du premier génome humain, de 1998 à 2003, n'a pas supprimé le besoin de séquençage, bien au contraire. Aujourd'hui de nombreuses études se focalisent sur le polymorphisme,

Société	Roche	Applied Biosystems	Illumina
Technique	Incorporation de bases, et lumière produite par oxydation de la luciférine.	Ligation de sondes contenant des paires de bases.	Incorporation de bases, étiquetées par des couleurs différentes.
Séquenceur	454 Titanium	Solid 4.0	HiSeq 2000
Longueur des reads	~ 400 bases	~ 75 bases	~ 100 bases
Erreurs fréquentes	insertions	substitutions	substitutions
Débit annoncé	~ 1 Gb/jour	~ 10 Gb/jour	~ 50 Gb/jour

c'est-à-dire sur les variations entre plusieurs individus. Le projet *1000 Genomes*, débuté en 2008, a pour objectif de séquencer le génome de 2500 personnes afin d'identifier les spécificités de chacun. Avant la fin de notre décennie, nous arriverons au génome à *1000 $*, et ce coût peut encore baisser. Cela rend possible la médecine personnalisée, le but étant de prescrire des traitements adaptés au métabolisme de chacun. Dans ces applications, on parle de *reséquençage*, car la séquence de chacun est très proche du génome de référence. Mis à part l'homme, environ un millier d'espèces seulement a été séquencé, ce qui est peu comparé aux millions d'espèces connues. De plus, une nouvelle discipline a fait son apparition dans les années 2000 : la *métagénomique* qui étudie des séquences d'ADN d'un écosystème dans son ensemble, sans séparation des espèces ni des individus, que ce soit dans des milieux naturels (océan, terre) ou bien chez l'homme (flore intestinale). Étudier des métagénomes permet aussi de découvrir des nouvelles séquences, provenant parfois d'espèces inconnues.

Utilisations indirectes

Un séquenceur peut aussi être utilisé indirectement. Lorsque l'on connait déjà un *génome de référence*, par exemple le génome humain, le séquençage sert à obtenir un ensemble de reads qui, une fois localisés, traduisent un phénomène biologique. C'est le passage du *séquençage* au *décryptage* du génome. Ainsi les études de *transcriptomique* examinent tous les ARN produits par une cellule à un instant donné. En fonction du nombre de reads localisés dans une zone donnée il est possible de comprendre quelles sont les parties du génome les plus actives en fonction des différents tissus (comme le rein, le cerveau ou la rétine) et de leur environnement. D'autres études concernent les sites de fixation des facteurs de transcription, ou bien la chromatine (la structure à grande échelle de l'ADN). Dans de nombreuses applications, le séquençage haut-débit est ainsi en train de remplacer d'autres techniques moins flexibles telles que les puces à ADN [Fig.4].

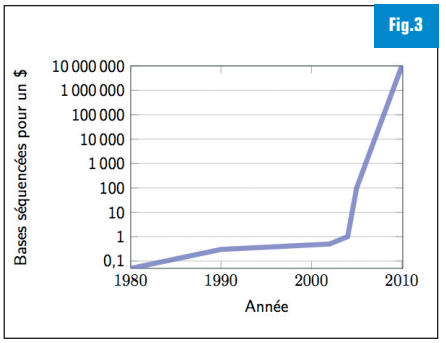
Défis algorithmiques

Algorithmique du texte

La bioinformatique des séquences illustre bien le transfert de recherche fondamentale en combinatoire vers une recherche appliquée. En effet, la recherche de mots dans un texte est un « vieux problème » d'*algorithmique du texte*, étudié dès les années 1970. Aujourd'hui, ces techniques sont utilisées dans les traitements de texte (pour rechercher un mot dans une page) ou dans les moteurs de recherche (pour rechercher les pages web ou apparaît un mot donné). Un read n'étant qu'un fragment d'un génome, il est généralement possible de retrouver sa localisation d'origine si on dispose du génome de référence de l'espèce. Dans toutes ces situations, il s'agit donc de trouver des *occurrences* d'une courte chaîne dans un très long texte, le texte pouvant être un ensemble de pages web ou un ou plusieurs génomes.

Mutations

On souhaite souvent faire des recherches approximatives, c'est-à-dire retrouver la courte chaîne même avec des erreurs, comme par exemple des erreurs orthographiques. Pour les séquences d'ADN, les différences entre chaînes similaires sont appelées les *mutations*. Lors de chaque division de cellule, ces mutations arrivent à une fréquence d'environ 10^{-5} par base mais n'ont généralement aucune conséquence sur la santé de l'individu. Les êtres humains ne possèdent en général que quelques dizaines de mutations par rapport au patrimoine génétique légué par leurs deux parents. Dans la population humaine, deux individus ont environ 0,5 % de différences dans leur génome. Ces différences concernent généralement la *substitution*, l'insertion ou la suppression d'une seule base, mais il peut aussi y avoir des fragments plus longs insérés ou supprimés. À plus grande échelle, ces mécanismes de mutations sont le support de l'*évolution*, et se retrouvent dans les séquences : nous avons plus de 95 % de notre génome en commun avec les grands singes, mais beaucoup moins avec les bactéries. Un read, puisqu'il provient du génome d'un individu, peut donc faire apparaître des différences avec le génome



Evolution du coût du séquençage.

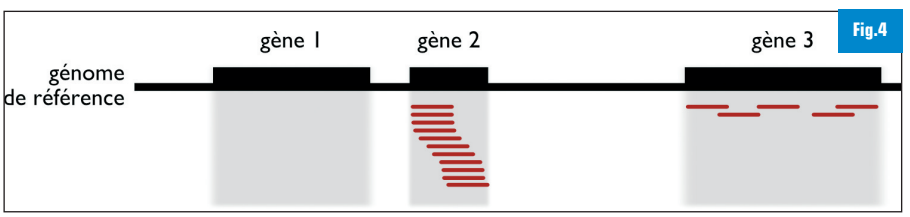
de référence. De plus, ces reads peuvent contenir des *erreurs de séquençage* en raison de la façon dont ils sont produits : les séquenceurs ne sont pas parfaits [Fig.5]. Paradoxalement, plus un read contient de mutations par rapport au génome de référence, plus il est difficile à localiser, mais plus il est potentiellement important pour les biologistes ou les médecins. Les mutations sont importantes car ce sont elles qui permettent de comprendre l'origine génétique de certaines maladies. Cela est d'autant plus vrai pour les cellules cancéreuses, qui connaissent souvent des phénomènes d'hyper-mutations entraînant de nombreuses différences avec le génome de référence, y compris la création de *chimères* par échanges génétiques entre deux parties du génome.

Solutions algorithmiques

Les problèmes bioinformatiques tels que la localisation des reads sur un génome de référence peuvent se résoudre par différentes techniques algorithmiques.

Indexation des k-mots

Pour rechercher rapidement des informations, une solution consiste à indexer le texte initial. Pour les séquences d'ADN, on peut par exemple indexer l'ensemble des suites de *k* lettres, suites appelées *k*-mots. Pour rechercher une chaîne quelconque, on extrait alors tous les *k*-mots de la chaîne à rechercher et on utilise l'index pour connaître la position de tous ces k-mots dans le texte. Ensuite, la comparaison est



Les techniques de RNAseq permettent de mesurer les niveaux d'expression de différents gènes, en localisant les reads sur le génome. Ici le gène 2 est fortement exprimé, et le gène 3 faiblement exprimé.

effectuée dans le voisinage de chacun des *k*-mots afin de vérifier que la totalité de la chaîne recherchée est bien présente dans le texte, éventuellement avec un certain nombre de mutations. Une telle approche est par exemple utilisée par BLAST (inventé en 1990), qui est certainement l'application bioinformatique la plus utilisée à ce jour. L'article de BLAST est d'ailleurs un des articles scientifiques le plus cité de tous les temps : des dizaines de milliers de scientifiques ont utilisé ou cité BLAST dans leurs travaux.

Arbre des suffixes

L'arbre des suffixes, proposé par Weiner en 1973, est une autre solution pour indexer des données. Il s'agit d'une structure dans laquelle tous les suffixes du texte sont enregistrés, c'est-à-dire toutes les chaînes du texte commençant à n'importe quel endroit du texte et se terminant à la fin. Un arbre des suffixes permet de rechercher n'importe quelle sous-chaîne du texte en temps proportionnel à la longueur de cette sous-chaîne, et donc en temps indépendant de la longueur du texte indexé. Autrement dit, en théorie, la recherche d'une sous-chaîne prendra autant de temps pour un texte d'une dizaine de pages que si l'ensemble des textes de la planète était indexé en utilisant un arbre des suffixes. Utiliser l'arbre des suffixes permet aussi de connaître la plus longue chaîne commune à deux textes ou encore de détecter des répétitions. Ce type de calcul est utilisé en bioinformatique afin de savoir quelles sont les parties qui ont été les plus conservées entre deux génomes d'espèces différentes, mais il a aussi des applications dans d'autres domaines comme la détection de plagiat.

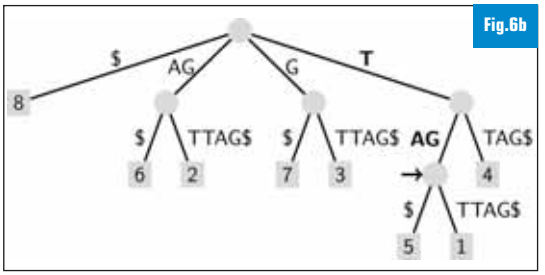
Nouvelles méthodes

De nombreuses recherches ont amélioré les index de k-mots et les arbres de suffixes, en proposant des structures plus pertinentes pour certains problèmes. Les méthodes que nous développons actuellement ne se contentent plus de rechercher un mot en autorisant quelques erreurs mais permettent aussi de trouver des informations qui sont éventuellement distantes sur le génome de référence, afin de découvrir des échanges génétiques. Au lieu de considérer le read comme un seul mot devant être trouvé de manière contigüe sur le génome, on peut alors le considérer comme un ensemble de segments plus courts pouvant se trouver à différents endroits du génome, segments pouvant contenir eux-mêmes des mutations.

Évolutions technologiques

Même les meilleurs algorithmes nécessitent de stocker et de traiter de grandes quantités de données. Le problème actuel de la bioinformatique est que l'explosion des données (multipliée par dix tous les ans) évolue à un rythme très supérieur à l'évolution technologique qui suit la loi de Moore (multiplication par deux tous les 18 mois). Un point important concerne l'adéquation entre les modèles théoriques de complexité de calcul et les implémentations. Par exemple, les différents niveaux de mémoire jouent un rôle important. Accéder au cache de premier niveau d'un processeur est 100 à 1000 fois plus rapide qu'accéder à la mémoire centrale d'un ordinateur, et 10^4 à 10^6 fois plus rapide que d'accéder à un disque dur. Cet élément est d'autant plus considérable que certaines structures d'indexation utilisent beaucoup de mémoire : un arbre des suffixes nécessite au moins dix octets par caractère du texte indexé. L'indexation d'un génome humain, qui est composé d'environ trois milliards de lettres, nécessite des dizaines de giga-octets avec l'arbre des suffixes. Il existe d'autres structures, compressées, qui apportent des compromis entre efficacité et occupation mémoire. Enfin, des projets actuels de séquençage tablent sur l'acquisition d'environ 1 péta-octet (mille téra-octets) de données. Les solutions apportées sont à la fois algorithmiques et technologiques, en utilisant du parallélisme, que cela soit à gros grain (grappes de calcul, cloud computing) ou à grain fin (SIMD, GPU). Cependant les algorithmes qui fonctionnent bien en séquentiel, sur un processeur à la fois, ne sont pas forcément ceux qui se parallélisent le mieux. C'est pour cela que d'importants efforts de recherche se consacrent à l'adaptation des algorithmes existants et au développement de nouveaux pour tirer parti des nouvelles architectures et de ces nouvelles données avec l'objectif ambitieux de faire évoluer l'algorithmique afin de résister à la divergence entre la croissance des capacités de calcul et de stockage, très lente, comparée à l'évolution explosive de volumes de données à traiter. Les enjeux qui se cachent derrière ces recherches sont considérables puisqu'elles conduiront la médecine à se doter d'outils extrêmement efficaces en termes de prévention des risques : identification de traitements de précaution, suivi ciblé de populations à risque, etc. Et, au delà de la médecine, il est probable que ce type de recherche aboutira à une dissémination de résultats en direction d'autres disciplines confrontées à des problématiques de fouille de données.

Recherche approximative de texte et localisation de reads dans un génome.



Arbre des suffixes sur le texte TAGTTAG$. Suivre un chemin de la racine de l'arbre (le cercle du haut) jusqu'à une feuille (les carrés) permet de lire un suffixe du texte. Par exemple, le chemin de la racine jusqu'à la feuille 4 permet de lire le suffixe TTAG$. Lire TAG à partir de la racine nous amène jusqu'au cercle marqué par une flèche. Regarder les feuilles se trouvant sous ce cercle nous donne le nombre de fois qu'apparaît TAG (deux feuilles = deux apparitions) et les positions auxquelles il apparaît dans le texte (5 et 1).

■ Mathieu **Giraud**
Titulaire d'un doctorat obtenu à l'Université de Rennes I, Mathieu est chercheur au CNRS, il travaille sur l'algorithmique pour la bioinformatique (recherches efficaces de similarités dans les séquences génomiques) ainsi que sur le calcul haute-performance (FPGA, GPU).

■ Mikaël **Salson**
Titulaire d'un doctorat obtenu à l'Université de Rouen, Mikaël est maître de conférences à l'Université Lille 1. Ses spécialités sont l'algorithmique du texte et l'analyse de reads de séquenceurs haut-débit.

L'Intelligence Ambiante à la carte

L'intelligence ambiante est généralement présentée comme un espace équipé de capteurs et d'actionneurs, couplés à un logiciel capable d'analyser, de réagir et d'anticiper les comportements de l'utilisateur et de répondre au mieux à ses besoins en matière de divertissement, de confort, de sécurité, etc.

Une vision

Il y a vingt ans Mark Weiser annonçait l'« ubiquitous computing » qu'il présentait comme la troisième vague de l'informatique après les mainframes et l'informatique individuelle. Cette vague a-t-elle commencé à déferler ? La maison intelligente est-elle en vue ? Des plates-formes prototypes préfigurent-elles les espaces intelligents de demain ? Des logiciels de recherche ont-ils été validés en vue de leur exploitation commerciale ? Des cas « d'usage incontournable » ont-ils été identifiés pour mettre à disposition l'intelligence ambiante auprès du grand public ?

La réalité

Aujourd'hui, ces questions sont encore largement du domaine de la recherche. Toutefois, l'évolution frénétique des technologies offre un nombre croissant d'opportunités de concrétiser cette vision. On assiste ainsi au déploiement massif de réseaux de communication (BlueTooth, WiFi, etc.), une montée en puissance des objets électroniques du quotidien (Smartphone, webcam, TV connectée, etc.), un taux d'équipement technologique bondissant en tous lieux (domicile, travail, lieux publics, etc.) et des coûts de production en réduction drastique. Outre les équipements matériels, il existe une offre pléthorique et renouvelée de services Web (courrier électronique, agenda, contacts de réseaux sociaux, etc.). L'intelligence ambiante dispose donc d'une large palette d'entités (équipements électroniques et services Web) pour constituer une infrastructure riche et évolutive [Fig.1].

L'INTELLIGENCE AMBIANTE À LA CARTE

L'infrastructure n'est toutefois qu'une étape vers l'avènement de l'intelligence ambiante, l'élément central en est l'utilisateur. Et parce qu'elle est au plus près de l'utilisateur, l'intelligence ambiante doit être définie « à la carte » pour s'adapter à ses besoins, ce dans les différents domaines d'usage de ses activités : confort, divertissement, sécurité, assistance, etc. Pour être « à la carte », l'intelligence ambiante doit proposer une infrastructure et des applications à « géométrie variable » [Fig.2].

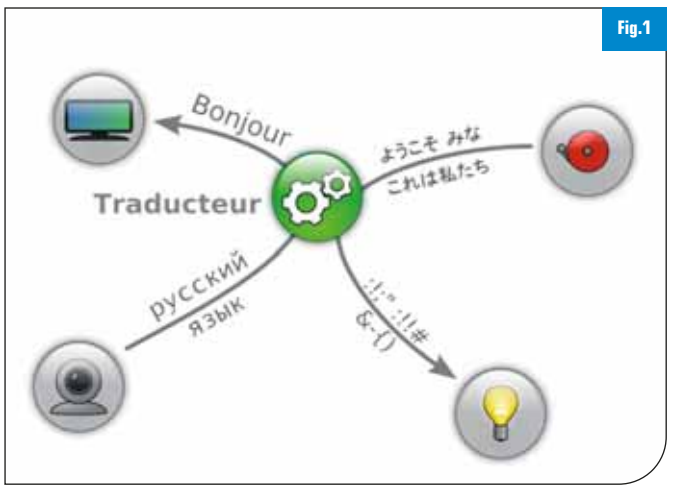
L'infrastructure à la carte

L'intelligence ambiante doit s'affranchir de l'hétérogénéité des équipements et des services Web pour permettre à une application de s'exécuter sur toute infrastructure possédant les fonctionnalités requises, quelle que soit leur mise en œuvre. Ainsi, par exemple, une fonctionnalité de détection de mouvement peut-elle être directement fournie par une caméra, ou par une couche logicielle analysant un flux vidéo. L'infrastructure doit pouvoir évoluer, en intégrant les nouveaux équipements et services Web, au gré des attentes de l'utilisateur et des innovations technologiques. Le défi est donc de définir une infrastructure « à la carte » [Fig.3].

Les applications à la carte

La versatilité des besoins des utilisateurs s'accommode mal du modèle traditionnel de l'édition logiciel dans lequel l'éditeur est l'unique vecteur pour répondre aux besoins des utilisateurs. Une brèche a été ouverte dans ce modèle par les plateformes logicielles ouvertes utilisées pour les Smartphones et les réseaux sociaux. L'intelligence ambiante, parce qu'elle s'entrelace graduellement dans les activités de l'utilisateur, doit pouvoir s'adapter et évoluer avec les besoins de celui-ci. Le défi est donc de proposer une plate-forme d'intelli-

Fig.1



L'hétérogénéité des équipements nécessite la mise en place d'une couche d'abstraction.

Fig.2

La maison, un lieu plein de ressources

gence ambiante avec des applications « à la carte » qui puisse concilier ouverture et sûreté.

DIASUITE

Les travaux de recherche effectués par l'équipe projet INRIA Phoenix qui compte treize chercheurs et ingénieurs ont permis la conception et le développement de DiaSuite : un atelier de génie logiciel et une plate-forme mettant en œuvre une approche « à la carte » de l'intelligence ambiante. Cette approche s'appuie sur des outils et des technologies standard comme Java, OSGI et Eclipse.

Diasuite comprend une phase de conception, en amont de la programmation, pour déclarer une application. Cette déclaration permet de guider et faciliter toutes les étapes critiques du cycle de vie d'une application d'intelligence ambiante : programmation, vérification, simulation, déploiement, exécution.

Un langage déclaratif, nommé DiaSpec, a été développé pour répondre aux spécificités de l'intelligence ambiante. Une déclaration DiaSpec comporte deux volets: (1) la définition d'un type d'infrastructure (par exemple, la maison) et (2) la description d'une application dans ses aspects fonctionnels (composants et interactions) et non-fonctionnels (sûreté de fonctionnement, qualité de services, sécurité).

La définition d'un type d'infrastructure

DiaSpec permet de définir l'ensemble des fonctionnalités que l'on peut déployer dans un type d'infrastructure, et ce, indépendamment de leur mise en œuvre, matérielle ou logicielle. Les équipements et services Web d'une infrastructure donnée sont choisis à la carte par l'utilisateur, en conformité avec la déclaration de son type, et peuvent évoluer au gré des besoins. Les instances spécifiques d'appareils et de services Web nécessitent, à l'instar d'un pilote de périphérique, une couche logicielle pour leur interfaçage. Enfin, la déclaration d'un espace intelligent peut être enrichie lorsque de nouvelles fonctionnalités sont proposées par des équipements ou des services Web.

La déclaration d'un type d'infrastructure est un élément fondamental à l'avènement de l'intelligence ambiante.

Elle permet de créer un référentiel que peut partager une communauté de développeurs dans un domaine donné (par exemple, le domicile). Dans DiaSuite, cette déclaration n'est pas contemplative : elle constitue l'une des entrées du générateur de framework de programmation qui produit le support permettant de découvrir et d'invoquer les entités d'une infrastructure donnée. Ce support permet à l'application de s'abstraire de l'infrastructure.

Un catalogue ouvert d'applications

Outre les fonctionnalités des entités d'une infrastructure, l'approche adoptée est également ouverte en termes d'applications. Une plate-forme logicielle reposant sur un catalogue ouvert d'applications a été créée, à l'instar des catalogues d'applications de Smartphones. Dans ce modèle, une application orchestre les fonctionnalités d'entités, sans pour autant connaître la nature de ces entités. Par exemple, une application permet de baisser le son du téléviseur lors d'un appel téléphonique entrant, et affiche le nom du correspondant. Pour ce faire, l'application dépend des fonctionnalités suivantes : le contrôle du téléviseur, le statut de la passerelle de téléphonie et un répertoire téléphonique. Ces fonctionnalités sont mises en œuvre différemment suivant les utilisateurs, ainsi le répertoire téléphonique peut prendre des formes multiples : contacts Google, amis Facebook, etc [Fig.4].

L'intelligence ambiante est « à la carte » puisque l'utilisateur installe ses applications en fonction de ses besoins, en les sélectionnant dans un catalogue en ligne ouvert. Si une application repose sur des fonctionnalités dont l'utilisateur ne dispose pas, il en est averti.

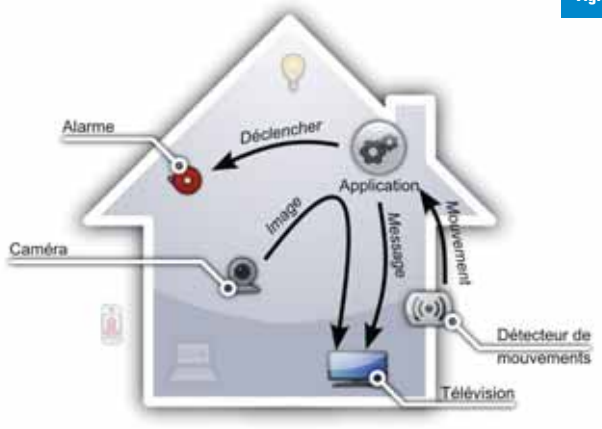
Par exemple, une application de surveillance du domicile peut utiliser un détecteur de mouvements dont l'utilisateur doit faire l'acquisition, s'il n'en dispose pas.

Un atelier de développement

Pour connaître un essor aussi remarquable que les Smartphones, l'intelligence ambiante doit faire émerger un flot d'applications innovantes, répondant aux demandes spécifiques des utilisateurs. Une telle stratégie suppose toutefois de relever deux défis : tout d'abord rendre le développement d'applications le plus aisé possible pour dynamiser et diversifier l'offre, ensuite concilier un catalogue ouvert d'applications tout en assurant les garanties de comportement requises par les utilisateurs.

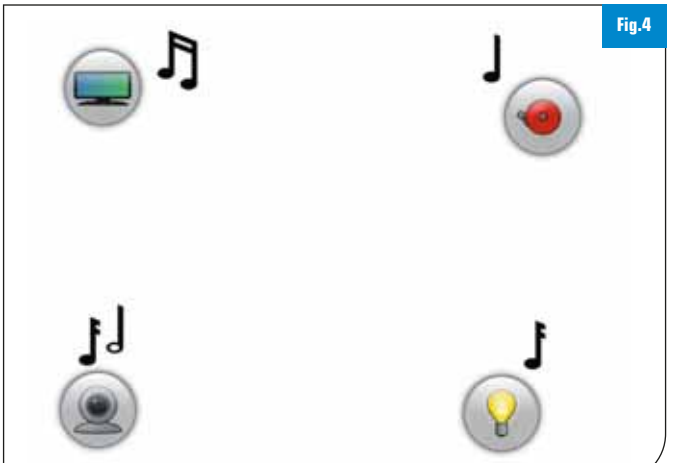
Ces deux problématiques constituent aujourd'hui des thématiques de recherche

Fig.3



La combinaison de ces ressources permet d'élaborer des applications

Fig.4



La première étape de la démarche consiste à définir l'infrastructure et inventorier l'ensemble des ressources disponibles.

particulièrement dynamiques, mais s'y ajoutent beaucoup d'autres, comme l'éthique [Fig.5].

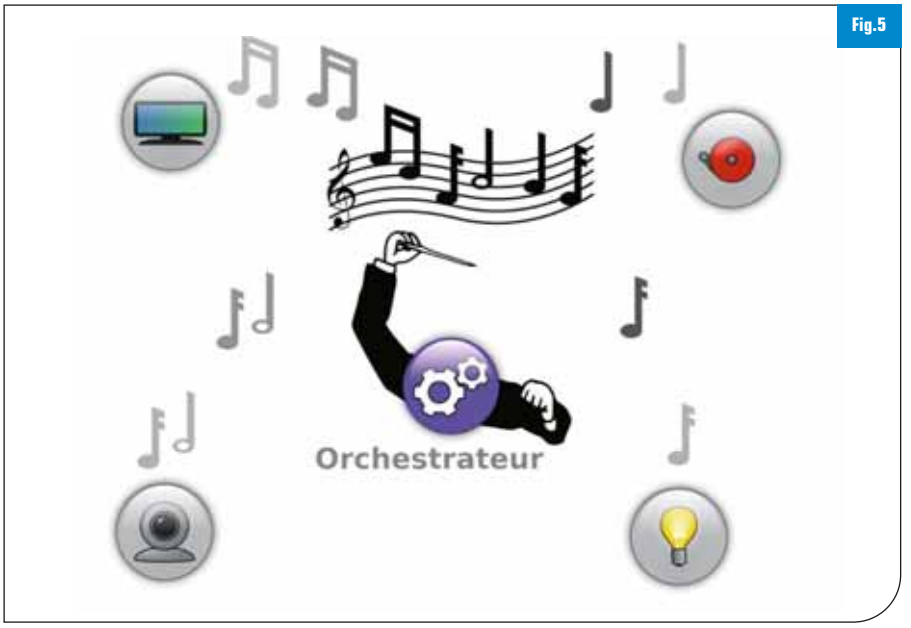


Fig.5

Les ressources peuvent alors être mobilisées par un orchestrateur sous la forme d'applications destinées à apporter un service à l'utilisateur.

Décrire une application

Pour répondre au premier défi, le développeur dispose de la définition de l'infrastructure qui permet à l'application développée de s'abstraire des variabilités des équipements et des services Web.

À partir de cette définition, le développeur élabore la description fonctionnelle et non-fonctionnelle de son application lors de la phase de conception.

La définition de l'infrastructure et la description de l'application forment la déclaration DiaSpec utilisée pour générer automatiquement un framework de programmation dédié en Java.

Ce framework encadre strictement la programmation de la logique applicative et fait levier sur les environnements de développement intégrés comme Eclipse pour une assistance dédiée à l'édition de code. Enfin, l'atelier logiciel pour l'informatique ambiante inclut un simulateur avec un rendu graphique 2-D permettant de tester visuellement une application et des scénarios d'usage grâce à des utilisateurs virtuels.

Vérifier une application a priori

Garantir le comportement d'une application (confidentialité, non-interférence, etc.) est essentiel au succès de l'intelligence ambiante. Son adoption serait compromise si des applications malicieuses étaient proposées, comme c'est parfois le cas pour les plateformes de Smartphones ou de réseaux sociaux. Dans notre approche, la garantie du comportement d'une application repose sur la description fournie par le développeur en prélude à la programmation. Non seulement, cette description aide au développement mais elle fournit également un cadre sémantique pour effectuer diverses vérifications sur l'application. Ainsi, les outils de Diaspec déterminent les ressources manipulées par l'application et la façon dont elles sont utilisées. Contrairement aux approches existantes (par exemple, un manifeste d'Android), une déclaration DiaSpec permet d'analyser finement le flot des informations d'une application, ce qui confère une meilleure pertinence aux alertes envoyées à l'utilisateur, comme la possibilité de fuite de données privées.

L'exploitation d'une déclaration DiaSpec informe donc précisément l'utilisateur sur les ressources utilisées par l'application, son comportement et ses interférences éventuelles avec des applications déjà déployées. Ces informations sont formulées simplement pour que l'utilisateur puisse être autonome dans l'administration de son espace intelligent.

Fig.6

La maison est enrichie par un flot d'applications.

LES PERSPECTIVES

L'émergence de l'intelligence ambiante nécessite de lever différents verrous. Il faut commencer par s'abstraire des nombreuses expertises sous-jacentes (réseaux, protocoles, systèmes embarqués), puis surmonter l'hétérogénéité des équipements, et enfin, dépasser le modèle traditionnel de l'édition logiciel pour couvrir un large spectre de besoins, au plus près de chaque utilisateur.

L'idée est qu'aujourd'hui, avec les objets existants dans l'environnement quotidien, on est déjà capable de faire énormément d'applications simples, malignes et utiles.

DiaSuite (http://diasuite.inria.fr) est le vecteur d'un nouveau modèle d'innovation pour l'intelligence ambiante : chaque développeur potentiel peut cibler des besoins particuliers, développer et tester une application indépendamment d'une infrastructure spécifique, publier cette application dans un catalogue et mesurer son utilité auprès d'utilisateurs [Fig.6].

■ Charles **Consel**, *Phoenix Research Group, INRIA Bordeaux - Sud-Ouest*
Professeur des Universités à l'Ecole Nationale Supérieure d'Electronique, d'Informatique et de Radiocommunications de Bordeaux (ENSEIRB) au sein de l'Institut Polytechnique de Bordeaux (IPB), où il a créé et dirigé le département de Télécommunications. Il a publié plus de soixante dix articles et a donné des exposés dans des laboratoires et universités de renommée internationale.

■ Benjamin **Bertran**, *Phoenix Research Group, INRIA Bordeaux - Sud-Ouest*
Ingénieur de recherche dans l'équipe Phoenix à l'INRIA Bordeaux Sud-Ouest. Il a participé aux différents travaux sur la convergence de la téléphonie sur IP et de la domotique. Il est maintenant en charge du développement de DiaSuite, la suite d'outils pour l'orchestration d'objets communicants.

Initiation à la 3D avec Direct3D 11

Pour faire de la 3D sur PC (ou ailleurs), il est nécessaire de bien comprendre les concepts de base afin de pouvoir par la suite construire nos applications sur des assises solides. Le but de cet article va donc être de mettre en place ces concepts puis de les implémenter simplement avec Direct3D 11. Un futur article se chargera d'aborder des notions plus complexes (telles que les ombres).

DU VERTEX AU PIXEL

Le premier élément que nous devons connaitre est le vertex (et son pluriel : les vertices). Un vertex est un point dans l'espace 3D. On peut donc le représenter dans sa version la plus simple par un vecteur à 3 valeurs : x, y et z. Tout ce que l'on peut voir dans une application 3D est donc intégralement constitué par un ensemble de vertices qui définissent l'ossature des objets à représenter [Fig.1].
Toutefois, ces vertices ne sont pas les seuls acteurs importants. En effet, pour qu'un objet soit totalement défini, on doit lui associer des faces. Les faces sont des triangles dont chaque sommet est un vertex. En effet, pour délimiter le volume d'un objet (que l'on appelle mesh en anglais), nous allons assembler des surfaces 2D. Or il se trouve que la plus simple forme géométrique définissant une surface 2D est le triangle. Les faces sont donc aussi des triplettes de valeurs i1, i2, i3 qui définissent le numéro dans la liste des vertices de chaque sommet de la face [Fig.2]. En termes de code, la définition d'un simple plan va donc être la suivante :

```
float[] vertices = new[]
  {
    -1.0f, -1.0f, 0f,
    1.0f, -1.0f, 0f,
    1.0f, 1.0f, 0f,
    -1.0f, 1.0f, 0f,

  };
```

```
short[] faces = new[]
  {
    (short)0, (short)1, (short)2,
    (short)0, (short)2, (short)3
  };
```

Un plan est donc constitué de 4 vertices distincts et de 2 faces reliant ces vertices (soit 6 indices). Le but d'une application 3D va donc être de passer de ces données 3D vers un seul objectif final : les pixels. En effet, elle devra transformer toutes ces informations en un unique tableau à deux dimensions dont chaque case sera une couleur. Ce tableau fera la taille de l'écran [Largeur x Hauteur] et chaque couleur servira alors à allumer un point sur l'écran.
Nous allons donc découper notre approche en deux temps. Tout d'abord nous allons voir comment avec une liste de vertices et une liste de faces nous allons pouvoir arriver à une liste de pixels (donc à des vecteurs 2D x,y). Puis nous verrons comment attribuer une couleur à chaque pixel.

FAIRE SON CINÉMA

Pour comprendre la transition de l'espace R3 (la 3D) à l'espace R2 (la 2D, et donc la dalle de l'écran), nous allons prendre l'analogie du cinéma. Nous allons nous glisser dans la peau d'un réalisateur d'un film publicitaire qui veut filmer une balle de tennis pour en faire la promotion.

Le monde global

Dans un premier temps, il faut considérer que la balle est rangée dans son étui. Pour la filmer, il faut donc l'amener sur le plateau de tournage. En 3D, nous appelons cela la transformation monde (World) c'est-à-dire le fait de prendre les coordonnées d'un objet et

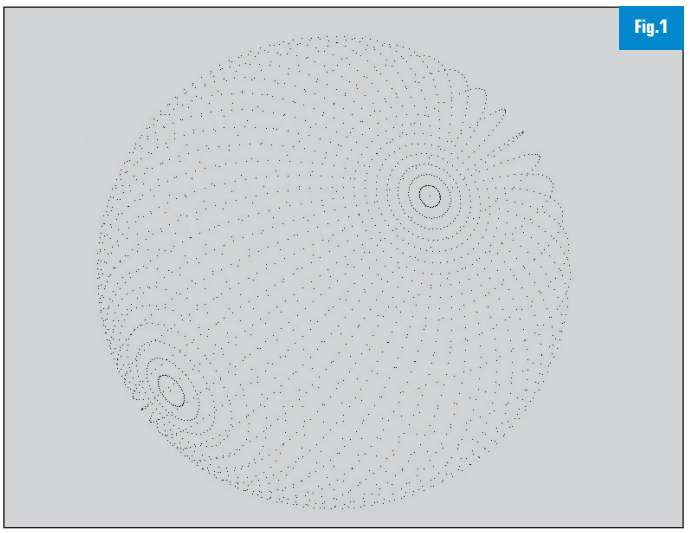


Fig.1

Les vertices d'une sphère

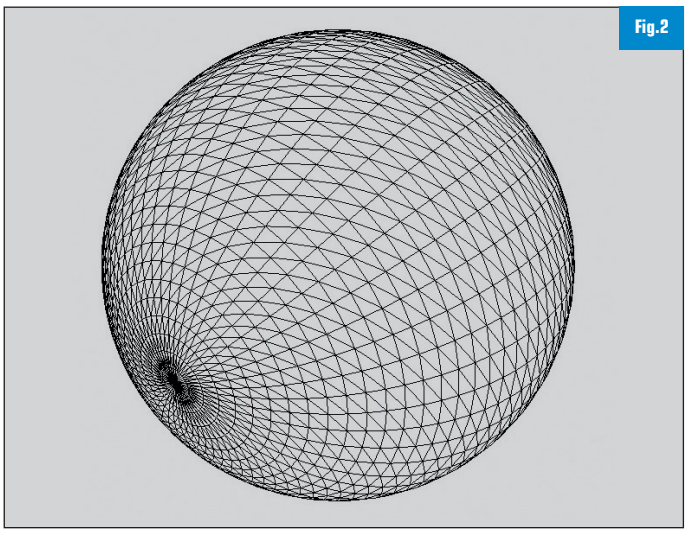


Fig.2

Les faces d'une sphère

de les déplacer vers le monde de la scène. En effet, pour construire une scène (ou filmer une publicité), nous allons faire appel à plusieurs objets qui à l'origine sont tous définis avec des coordonnées centrées sur le 0 du monde (soit [0, 0, 0]). Or si nous ne faisons rien, ils seront tous empilés au même endroit occupant tous le même espace (ce qui, il faut bien l'avouer est largement plus facile à faire en informatique qu'au cinéma). Il faut donc les déplacer (voire les retailler ou les tourner) pour les mettre là ou ils sont censés être.
Cette transformation monde (comme toutes les transformations en 3D) s'effectue par le biais d'une matrice. Une matrice est la représentation d'une transformation géométrique. Ainsi la multiplication d'un vecteur par une matrice donne un vecteur modifié par la transformation incluse dans la matrice. Les matrices sont multipliables entre elles (associatives) et le résultat de la multiplication de deux matrices donne une nouvelle matrice portant les deux transformations. Par exemple, imaginons deux matrices M1 et M2. Admettons que M1 soit une matrice de translation (c'est-à-dire qu'elle déplace les vecteurs qu'elle modifie) et que M2 soit une matrice de rotation (c'est-à-dire qu'elle fait tourner dans l'espace les vecteurs qu'elle modifie). Le résultat de M1 x M2 donne une matrice qui applique une translation PUIS une rotation.
Donc, en ce qui concerne notre matrice monde, nous pourrons mettre dedans les transformations nécessaires pour amener notre objet de son origine vers sa position finale dans la scène (avec donc la conjonction de translations, rotations et zooms nécessaires).

Le point de vue de la caméra

Une fois que les objets sont à leur position définitive dans la scène, nous allons appliquer une seconde matrice pour calculer leur position du point de vue de la caméra du réalisateur. En effet, nous effectuerons nos rendus depuis un point de vue donné que nous appellerons la « caméra » comme pour le cinéma car l'analogie est parfaite. Cette seconde matrice s'appelle la matrice de vue car finalement elle définit le point de vue. Elle se caractérise par essentiellement une position et une coordonnée cible, toutes deux sous la forme d'un vecteur R3.

La projection

Finalement, il reste une troisième matrice à définir que l'on appelle la matrice de projection. Son rôle est assez particulier dans le sens où elle doit transformer les vecteurs depuis l'espace tridimensionnel vers l'espace plat de l'écran. Ainsi elle assure le passage d'un vecteur 3 : (x1, y1, z1) à un vecteur 2 : (x2, y2) en tenant compte de la taille de la zone de rendu, de la focale de la caméra et du ratio largeur / hauteur.

Pipeline géométrique

Au final, chaque vertex va donc subir la transformation suivante :

$Matrice_{finale} = Matrice_{monde} * Matrice_{de\ vue} * Matrice_{de\ projection}$

La formule finale sera donc :

$Pixel = Vertex * Matrice_{finale}$

LES SHADERS OU COMMENT PROGRAMMER SON GPU ?

Maintenant que nous sommes au point sur la théorie nous allons voir comment demander à nos chers amis les GPU (Graphics Processing Unit, les cerveaux corvéables à souhait de nos cartes graphiques modernes) de faire le travail pour nous.

Au passage, il est intéressant de noter que si nous déléguons ce travail au GPU c'est pour la bonne raison que ces derniers sont faits pour les tâches répétitives (et c'est parfaitement le cas ici ou le même traitement doit être effectué sur chaque vertex).
Ce sont en effet des processeurs super-scalaires composés de très nombreuses unités de calcul en parallèle.
Pour programmer les GPU, nous allons utiliser un langage de haut niveau propre à Direct Graphics qui s'appelle le Microsoft HLSL (High Level Shader Language).
Ce langage proche du C/C++ permet de construire des shaders qui sont les unités d'exécution de base du GPU. Il existe plusieurs catégories de shaders en fonction de la tâche à réaliser.

Vertex shader

La première catégorie de shader qui va nous intéresser est constituée par les vertex shaders. Ce sont eux qui sont appelés les premiers et qui ont en charge le traitement des vertices pour en faire des pixels. De ce fait, ce sont eux qui vont faire la transformation géométrique avec notre MatriceFinale.

```
cbuffer globals
{
  matrix matriceFinale;
}

struct VS_IN
{
  float3 pos : POSITION;
};

struct PS_IN
{
  float4 pos : SV_POSITION;
};

// Vertex Shader
PS_IN VS( VS_IN input )
{
  PS_IN output = (PS_IN)0;

  output.pos = mul(float4(input.pos, 1), matriceFinale);

  return output;
}
```

Le vertex shader est donc en fait une fonction (qui peut, si besoin, en appeler d'autres) qui prend en paramètre un vertex (dont nous devons aussi définir le type) et retourne un pixel (dont on définit aussi le type). Pour le moment, les types que l'on manipule sont réduits à leur plus simple expression : un vecteur3 en entrée et un vecteur2 en sortie.
Bien évidemment par la suite, nous allons enrichir nos types pour transporter plus d'informations dans le pipeline de rendu. Le shader en tant que tel applique uniquement la multiplication entre le vertex d'entrée et la matrice de transformation que nous avons définie en tant que variable globale. Nous verrons dans un autre article que le travail effectué par les vertex shaders peut s'avérer beaucoup plus complexe.

Pixel shader

Une fois que l'on a notre liste de pixels, il faut bien leur attribuer une couleur. C'est le rôle des pixels shaders qui sont donc appelés une fois que les vertex shaders ont terminé leur travail. Il est important de noter qu'entre les vertex shaders et les pixels shaders il y a une étape supplémentaire, réalisée par les GPU : la rasterization.
Cette étape consiste d'une part à clipper les pixels (c'est-à-dire à ne garder que ceux qui sont dans l'écran) et d'autre part à faire l'interpolation nécessaire pour remplir les triangles. En effet, les vertex shaders ne travaillent que sur les sommets des faces (les vertices donc). Le but du rasterizer est donc d'interpoler les valeurs sur toute la surface de la face. Finalement, pour chaque pixel ainsi généré, le GPU appellera le pixel shader.

```
// Pixel Shader
float4 PS( PS_IN input ) : SV_Target
{
 return float4(1, 0, 0, 1);
}
```

Notre pixel shader prend donc en paramètre un pixel et retourne une couleur. Pour le moment, c'est la même couleur pour chaque pixel. Nous allons rajouter un peu de fun en allant lire dans une texture des informations un peu plus diversifiées. Pour cela, il faut modifier nos vertices pour qu'en plus des coordonnées de position ils transportent des coordonnées de texture (le pixel de la texture associé au vertex). Au niveau de notre vertex shader, nous devons juste faire un passage de cette information vers le pixel (puisque le vertex shader n'a pas d'usage de cette donnée). Au niveau du pixel shader, nous allons utiliser les coordonnées de texture pour aller lire pour chaque pixel la couleur à retourner dans la texture. Pour cela, il faut rajouter un sampler (qui finalement n'est qu'un outil pour exprimer la lecture dans une texture) et une variable de type texture.

```
cbuffer globals
{
 matrix matriceFinale;
}

Texture2D yodaTexture;
SamplerState currentSampler
{
 Filter = MIN_MAG_MIP_LINEAR;
 AddressU = Wrap;
 AddressV = Wrap;
};

struct VS_IN
{
 float3 pos : POSITION;
 float2 uv : TEXCOORD0;
};

struct PS_IN
{
 float4 pos : SV_POSITION;
 float2 uv : TEXCOORD0;
};
```

```
// Vertex Shader
PS_IN VS( VS_IN input )
{
 PS_IN output = (PS_IN)0;

 output.pos = mul(float4(input.pos, 1), matriceFinale);
 output.uv = input.uv;

 return output;
}

// Pixel Shader
float4 PS( PS_IN input ) : SV_Target
{
 return yodaTexture.Sample(currentSampler, input.uv);
}
```

Les fichiers .FX

Les fichiers .FX (ou fichiers d'effets) ont pour but de rassembler en un seul endroit à la fois le vertex et le pixel shader. Les déclarations des variables (constantes, types, textures, samplers) sont également embarquées.
De plus, la notion de technique est prise en compte avec la possibilité d'avoir au sein d'une technique plusieurs passes ou chaque passe contient sa configuration avec son vertex et son pixel shader.
Dans notre cas, il nous faudrait juste rajouter ceci à nos shaders :

```
// Technique
technique10 Render
{
 pass P0
 {
  SetGeometryShader( 0 );
  SetVertexShader( CompileShader( vs_4_0, VS() ) );
  SetPixelShader( CompileShader( ps_4_0, PS() ) );
 }
}
```

On peut donc constater que notre technique est assez simple avec une seule passe à prendre en compte. Ce fichier FX sera par la suite chargé par la classe Effect de Direct Graphics qui se chargera de gérer pour nous la tuyauterie (compilation, affectation des variables, affectation des shaders, etc.). Au passage, nous compilons vers les shaders 4.0 (compatible Direct3D 10 donc), car il y a encore assez peu de cartes compatibles Direct3D 11. Or, comme ce dernier peut fonctionner sur une carte graphique Direct3D 10, nous n'allons pas nous en priver.

MISE EN ŒUVRE AVEC DIRECT3D 11

Pour utiliser DirectX 11 (et sa partie 3D, Direct3D 11), nous allons utiliser le wrapper SlimDX (http://slimdx.org/download.php). Ce projet Open Source constitue un excellent point d'entrée vers DirectX pour les développeurs .NET. En effet, DirectX est une API COM et ne propose pas nativement de wrapper pour le monde managed. Microsoft fournit toutefois un wrapper nommé Windows API Code Pack (http://archive.msdn.microsoft.com/WindowsAPICodePack), même s'il s'avère moins pratique que SlimDX pour la partie 3D.

Initialisation

L'initialisation de Direct3D 11 va nécessiter la mise en place de 4 variables importantes :

- **Le device** qui va être notre intermédiaire avec le driver de la carte graphique
- **La swap chain** qui définit le mécanisme visant à ramener l'image produite sur la carte graphique vers la fenêtre Windows que nous utilisons comme support
- **Le back-buffer** (buffer de travail) qui est l'espace mémoire de la carte graphique ou sera produite l'image
- **La vue de rendu** qui est la vue sur le back-buffer et qui permet de le manipuler (en effet, en Direct3D 11, on ne manipule pas directement les buffers mais on passe par des vues, ce qui permet d'avoir un seul buffer et plusieurs vues dessus en fonction du besoin)

Pour créer tout ce petit monde, nous allons donc mettre le code suivant en place :

```
// Création de notre device (on accepte les cartes dx10 ou plus)
FeatureLevel[] levels = {
    FeatureLevel.Level_11_0,
    FeatureLevel.Level_10_1,
    FeatureLevel.Level_10_0
  };

// Définition de notre swap chain
SwapChainDescription desc = new SwapChainDescription();
desc.BufferCount = 1;
desc.Usage = Usage.BackBuffer | Usage.RenderTargetOutput;
desc.ModeDescription = new ModeDescription(0, 0, new Rational
(0, 0), Format.R8G8B8A8_UNorm);
desc.SampleDescription = new SampleDescription(1, 0);
desc.OutputHandle = Handle;
desc.IsWindowed = true;
desc.SwapEffect = SwapEffect.Discard;

Device.CreateWithSwapChain(DriverType.Hardware, DeviceCreation
Flags.None, levels, desc, out device11, out swapChain);

// Récupération du buffer de travail
backBuffer = Resource.FromSwapChain<Texture2D>(swapChain, 0);

// Definition de la vue de rendu
renderTargetView = new RenderTargetView(device11, backBuffer);
device11.ImmediateContext.OutputMerger.SetTargets(renderTarget
View);
device11.ImmediateContext.Rasterizer.SetViewports(new  View
port(0, 0, ClientSize.Width, ClientSize.Height, 0.0f, 1.0f));
```

La partie importante se situe au niveau de la description de la swap chain. Ainsi nous indiquons ici que nous voulons une swap chain sur le back-buffer, sans anti-aliasing (SampleDescription avec un seul sample et de qualité = 0) en mode fenêtré et sans conservation du buffer de travail précédent (SwapEffect.Discard).

Notons également au passage la déclaration des FeatureLevels qui permet d'indiquer que nous acceptons les cartes Direct3D 10, 10.1 et 11. Le but ici étant de faire un premier rendu, nous ne rentrerons pas en détail dans le reste des paramètres qui ont été initialisés à leur valeur par défaut (ou du moins leur valeur standard).

Mise en place de nos shaders

Pour utiliser nos shaders, nous allons donc utiliser une instance de la classe Effect :

```
using (ShaderBytecode byteCode = ShaderBytecode.CompileFromFile
(«Effet.fx», «bidon», «fx_5_0», ShaderFlags.OptimizationLevel3,
EffectFlags.None))
{
  effect = new Effect(device11, byteCode);
}

var technique = effect.GetTechniqueByIndex(0);
var pass = technique.GetPassByIndex(0);
layout = new InputLayout(device11, pass.Description.Signature,
new[] {
    new InputElement(«POSITION», 0, Format.R32G32B32_Float, 0, 0),
    new InputElement(«TEXCOORD», 0, Format.R32G32_Float, 12, 0)
     });
```

La compilation de l'effet se fait donc via le constructeur de Effect, qui prend le byte code issu de la méthode statique ShaderBytecode.CompileFromFile. Cette dernière prend notamment en paramètre le profil de compilation. Ici nous ciblons des shaders pour Direct3D 11, donc le profil « fx_5_0 » (pour Direct3D 10, nous aurions choisi « fx_4_0 »). Par la suite, afin de décrire au pipeline de Direct3D comment seront formés nos vertices, nous produisons un layout qui caractérise la structure d'un vertex.

Au passage, il est important de noter que nous passons à ce layout, une référence vers la première passe de notre technique afin de bien valider que notre effet est compatible avec la structure de nos vertices.

Préparation des données géométriques

Pour stocker et utiliser nos données géométriques, nous allons produire des buffers au sein de la carte graphique : un vertex buffer et un index buffer. Le premier contiendra donc les vertices (avec comme vu dans notre layout une position en 3D et une coordonnée de texture). Le second quant à lui portera la liste des indices de faces. Ce qui en termes de code, donne :

```
// Création du vertex buffer
var stream = new DataStream(4 * vertexSize, true, true);
stream.WriteRange(vertices);
stream.Position = 0;

var vertexBuffer = new Buffer(device11, stream, new BufferDescription
{
  BindFlags = BindFlags.VertexBuffer,
  CpuAccessFlags = CpuAccessFlags.None,
  OptionFlags = ResourceOptionFlags.None,
  SizeInBytes = (int)stream.Length,
  Usage = ResourceUsage.Default
});
stream.Dispose();

// Index buffer
stream = new DataStream(6 * 2, true, true);
stream.WriteRange(faces);
```

```
stream.Position = 0;

var indices = new Buffer(device11, stream, new BufferDescription
{
  BindFlags = BindFlags.IndexBuffer,
  CpuAccessFlags = CpuAccessFlags.None,
  OptionFlags = ResourceOptionFlags.None,
  SizeInBytes = (int)stream.Length,
  Usage = ResourceUsage.Default
});
stream.Dispose();
```

On peut constater que les deux buffers sont créés de manière à ce que le CPU n'y accède pas. Cela indique donc que c'est dans la mémoire de la carte graphique que l'on stockera ces données (Le plus performant pour le GPU). Le seul point qui les différencie est le type de binding (VerteBuffer ou IndexBuffer).
Par la suite, il suffit de donner les deux buffers au contexte de notre device :

```
// Transfert vers le device
device11.ImmediateContext.InputAssembler.InputLayout = layout;
device11.ImmediateContext.InputAssembler.PrimitiveTopology =
PrimitiveTopology.TriangleList;
device11.ImmediateContext.InputAssembler.SetVertexBuffers(0,
new VertexBufferBinding(vertexBuffer, vertexSize, 0));
device11.ImmediateContext.InputAssembler.SetIndexBuffer(indices,
Format.R16_UInt, 0);
```

En plus de transférer nos buffers, nous indiquons le layout des vertices et que les indices représentent une topologie de type Triangle.

Affectation des constantes des shaders

Pour fonctionner, en plus des buffers, nos shaders attendent aussi la définition de constantes et notamment la MatriceFinale et la texture à utiliser. Nous passons bien sûr par notre instance de la classe Effect pour transférer nos constantes :

```
// Texture
Texture2D texture2D = Texture2D.FromFile(device11, «yoda.jpg»);
ShaderResourceView view = new ShaderResourceView(device11, texture2D);

effect.GetVariableByName(«yodaTexture»).AsResource().SetResource(view);

RasterizerStateDescription rasterizerStateDescription = new Ras
terizerStateDescription {CullMode = CullMode.None, FillMode =
FillMode.Solid};

device11.ImmediateContext.Rasterizer.State = RasterizerState.
FromDescription(device11, rasterizerStateDescription);

// Matrices
Matrix worldMatrix = Matrix.RotationY(0);
Matrix viewMatrix = Matrix.Translation(0, 0, 5.0f);
const float fov = 0.8f;
Matrix projectionMatrix = Matrix.PerspectiveFovLH(fov, Client
Size.Width / (float)ClientSize.Height, 0.1f, 1000.0f);
```

```
effect.GetVariableByName(«matriceFinale»).AsMatrix().SetMatrix
(worldMatrix * viewMatrix * projectionMatrix);
```

Le calcul de notre $Matrice_{Finale}$ se fait donc comme indiqué plus haut en multipliant les trois matrices de base. Ces dernières sont construites grâce aux méthodes statiques de la classe Matrix.
Et c'est via la méthode GetVariableByName que l'on peut accéder aux constantes des shaders.

Rendu final

Nous avons donc deux buffers pour notre géométrie stockés dans la mémoire de la carte graphique, nos shaders qui sont compilés et prêts à l'emploi avec leurs constantes affectées.
Il ne nous reste donc plus qu'à donner l'ordre de rendu :

```
// Rendu
device11.ImmediateContext.ClearRenderTargetView(renderTarget
View, new Color4(1.0f, 0, 0, 1.0f));
effect.GetTechniqueByIndex(0).GetPassByIndex(0).Apply(device11.
ImmediateContext);
device11.ImmediateContext.DrawIndexed(6, 0, 0);
swapChain.Present(0, PresentFlags.None);
```

Le processus est le suivant :

- On efface le buffer de travail
- On récupère la technique qui nous intéresse, puis sa première passe à qui l'on demande via la méthode Apply d'affecter les shaders et de transmettre les constantes
- Puis le contexte du device lance l'ordre de rendu de 6 primitives. Le type des primitives ayant été défini lors de l'affectation du vertex buffer
- Finalement, la méthode swapChain.Present transfère l'image produite vers la fenêtre de notre application

Le superbe rendu final avec Direct3D 11

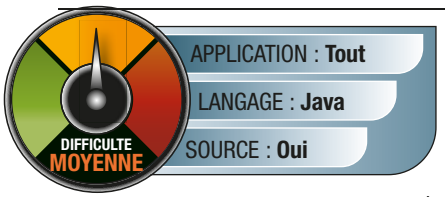
Conclusion

Nous avons donc désormais les bases nécessaires à la mise en place de rendus de plus haut niveau. Grâce aux shaders et à la gestion de notre fichier .fx, nous pouvons imaginer toutes sortes de comportements comme les célèbres bumps et autres joyeusetés. Ce que nous avons réalisé avec un simple plan fonctionne strictement de la même manière avec par exemple une ville entièrement modélisée. En effet, notre système prend juste une suite de vertices et de faces. Bien sûr il faudra alors travailler sur des techniques d'optimisation, mais le concept restera le même.
N'hésitez donc pas à jouer avec l'exemple de code fourni avec cet article afin de bien comprendre comment tout cela s'imbrique.

■ David **Catuhe**
Developer Evangelist, Microsoft France.

Les nouveautés de Spring 3, la suite !

Le mois dernier nous vous proposions un article sur les nouveautés des releases 3.x de Spring Framework concernant la configuration de vos applications. Nous allons, ce mois-ci, nous intéresser aux autres apports de Spring 3.x, notamment ceux relatifs à l'abstraction des caches, ou encore à l'intégration de certaines API de Java EE 6. Alors que WMWare sort sa solution PaaS, Cloud Foundry, l'écosystème Spring continue d'évoluer. Citons quelques exemples de projets très actifs et ambitieux comme Spring Data, Spring Batch ou Spring Social. Nous nous focaliserons néanmoins dans cet article uniquement sur les récents changements de Spring Framework.



QUELQUES CHANGEMENTS MINEURS POUR COMMENCER

Java 5 est désormais requis par Spring 3, la version 4 n'est plus supportée du fait de l'introduction des Generics et de l'utilisation des classes du package *java.util.concurrent*. Autre aspect intéressant : vous pouvez désormais, depuis la release 3.0 de Spring, créer vos propres annotations composites regroupant les stéréotypes Spring tels que @Service, @Transactional, @Scope. A noter qu'il n'est pas possible d'utiliser ce type d'annotation sur des classes de tests unitaires. Des discussions ont actuellement lieu sur le sujet sur la JIRA SPR-7827 actuellement affectée à la milestone 3.1.0-M2 de Spring. Pour finir, les fonctionnalités de mapping objet/XML (Spring OXM) autrefois dans *Spring WS* font désormais partie intégrante de *Spring Core*. Elles fournissent une abstraction API de sérialisation XML comme JAXB2, XStream, Castor, … et facilitent l'échange de messages XML via REST ou JMS.

SUPPORT REST, SPRING MVC ET LES API WEB

Dans le monde Java, il est possible de créer un service web RESTful de plusieurs façons. Pour ne citer que quelques exemples, vous pouvez utiliser la JSR 311 (JAX-RS) et son implémentation de référence Jersey, ou bien encore choisir le framework pionnier Restlet, ou encore développer vous-même votre solution. Spring 3.0 apporte une autre alternative avec son propre support REST. Bien que ce support REST ne soit pas une implémentation de la JSR 311, il a un plus large éventail de fonctionnalités que la spécification elle-même (notamment le support REST client). Il est surtout intégré dans la couche Spring MVC, ce qui facilite grandement son utilisation. Ainsi tout contrôleur Spring MVC peut devenir un contrôleur REST grâce à des annotations particulières, telles que @RequestMapping déjà existante et @PathVariable introduite dans la release 3.0 de Spring, permettant l'identification des ressources et les mappings d'URI.
Voici comment écrire la méthode d'un contrôleur permettant d'accéder à la liste des concerts d'un groupe donné :

```
@RequestMapping(«/gigs/{bandName}»)
public String searchTicketingsByName(@PathVariable String band
Name, Model model) {

List<Ticketing> ticketings = ticketOrderService.searchTicketings
ByBand(bandName.trim());
model.addAttribute(«ticketings», ticketings);
return «ticketings»;
}
```

Les services respectant l'approche RESTful utilisent les URI pour nommer les ressources. Les templates d'URI dans Spring MVC permettent de paramétrer les URI en déclarant directement des variables dans ces URI et en mappant ces variables avec des ressources données. Dans notre exemple, la variable *bandName* dans la déclaration de l'annotation *@RequestMapping* est indiquée entre accolade {} et elle est mappée directement avec le paramètre d'entrée *bandName* annoté avec *@PathVariable*. Lors de l'appel de la requête suivante http://monsite.com/gigs/Fugazi, nous demandons simplement d'accéder aux concerts du groupe Fugazi (/gigs/Fugazi).
Les paramètres de la méthode annotés par *@PathVariable* peuvent être de type simple (int, Date, etc.), Spring convertit automatiquement vers le type approprié (une exception *TypeMismatchException* est levée si le type n'est pas correct).
Une autre nouveauté de Spring 3.0 vis-à-vis du support REST concerne la négociation de contenu. Une architecture RESTful peut exposer de multiples représentations d'une ressource. Il existe deux stratégies pour qu'un client informe le serveur de la représentation qu'il supporte. La première stratégie, la plus simple, consiste à utiliser un URI distinct pour chaque ressource. L'URI http://monsite.com/gigs/Fugazi.pdf va permettre d'accéder à une liste PDF des concerts du groupe Fugazi tandis que l'URI http://monsite.com/gigs/Fugazi.xml permettra d'accéder à la même liste mais au format XML. La deuxième stratégie consiste pour le client à utiliser la négociation de contenu, supportée directement par le protocole HTTP. Pour une ressource donnée, un client HTTP peut indiquer dans l'en-tête de la requête HTTP, grâce au champ *Accept*, les formats de données qu'il comprend. Par exemple :
Accept: text/html,application/xhtml+xml,application/xml,application/pdf. Depuis sa version 3.0, Spring fournit l'API *ContentNegotiating ViewResolver* qui sélectionne une vue (view MVC) appropriée en fonction de la requête. Pour cela, il s'appuie sur plusieurs *ViewResolvers* et compare ainsi les types de médias indiqués dans l'en-tête de la requête avec les types de médias (Content-Type) supportés par

la vue associée. Côté client, Spring 3.0 introduit le RestTemplate. Très similaire au WebServiceTemplate, il permet d'éviter toute la verbosité technique pour l'écriture d'un client REST interrogeant un Web Service REST. Les conversions des objets java vers la requête HTTP et de la réponse HTTP en objets Java sont alors grandement facilitées grâce à des convertisseurs Spring (*HttpMessageConverter*) injectés au Rest Template. Le RestTemplate s'interface par exemple aisément avec un framework comme Jackson JSON processor pour le marsalling/unmarshalling Java/JSON.

Au-delà du support REST, Spring MVC a connu plusieurs grands changements depuis la version 3.0 de Spring. Et cela n'est pas prêt de s'arrêter : plusieurs évolutions sont annoncées pour la version 3.1, notamment la possibilité de déclarer des mappings abstraits à travers plusieurs contrôleurs ou encore une meilleure personnalisation des arguments des méthodes.

Parmi les API Java EE 6 plutôt dédiées au web, depuis sa version 3.0 Spring supporte JSF 2.0, apporte un support à la version 3.0 de l'API Servlet (déclaration des Servlets, Filters via des annotations) et à la version 2.0 de l'API Portlet.

Mais surtout une des grandes nouveautés est la gestion des conversations web, issue de Spring Web Flow, certes beaucoup plus limitée que celle du framework Seam.

Une conversation web peut ainsi être associée à une fenêtre ou un onglet de navigateur, ce qui vous permet de disposer d'espaces de travail distincts au niveau du client web.

D'AUTRES API JAVA EE SUPPORTÉES

Spring 3 supporte désormais JPA 2.0 et a également introduit le support de la JSR 303 Bean Validation. Hibernate Validator en est l'implémentation de référence. On peut donc de manière complètement intégrée à Spring valider des objets en s'appuyant sur leurs annotations (ex : @NotNull). Cela représente un réel intérêt dans le cadre de Spring MVC : l'ajout de l'annotation @Valid sur les paramètres à valider est ainsi suffisant pour déclencher une validation.

Les contraintes personnalisées peuvent également bénéficier de l'injection de dépendance.

SCHEDULING ET EXÉCUTION ASYNCHRONE

Spring a introduit le nouveau namespace *task* pour faciliter la configuration de tâches planifiées et d'exécutions asynchrones. Il suffit d'ajouter la déclaration suivante dans son contexte Spring XML

```
<task:annotation-driven />
```

Tâche planifiée

Le service ci-dessous déclare une tâche de rappel d'achat de billets se déclenchant toutes les secondes.

```
@Service
public class TicketOrderService {

  private AtomicInteger nbRappel = new AtomicInteger();

  @Scheduled(fixedDelay = 1000)
  public void rappel() {
    nbRappel.incrementAndGet();
    System.out.println(«tickets à réserver»);
  }
}
```

L'instanciation du service TicketOrderService et son arrêt dans un test au bout de 3 secondes produit le résultat suivant :

```
tickets à réserver
tickets à réserver
tickets à réserver
```

Spring a fait automatiquement appel à un *ScheduledThreadPoolExecutor* pour invoquer la méthode *rappel*.

Exécution asynchrone

Prenons maintenant l'exemple d'une tâche asynchrone :

```
@Service
public class PublicationTicketServiceImpl implements Publication
TicketService {

  @Override
  @Async
  public Future<String> publier() {
    return new AsyncResult<String>(envoiWebService());
  }

  private String envoiWebService() {
    try {
      Thread.sleep(2000); // Simulation d'un envoi long
    } catch (InterruptedException e) {
      Thread.currentThread().interrupt();
    }
    return «OK»;
  }
}
```

La méthode *publier* sera automatiquement exécutée dans un thread différent de l'appelant. On notera l'utilisation d'un *Future* (API Java 5 *java.util.concurrent*) de telle sorte que la récupération du résultat de l'exécution asynchrone puisse être annulée via un appel à cancel() et/ou bloquante via l'appel à get() avec éventuellement, un timeout.

```
Future<String> resultat = publicationTicket.publier();
System.out.println(«Resultat : « + resultat.get());
```

Cache abstraction

Vous avez aimé la configuration des transactions avec *<tx:annotation-driven/>*, vous allez adorer son équivalent en terme de cache.

Depuis la version 3.1, on a en effet la possibilité de configurer de manière transparente un cache simple et efficace en étant très peu intrusif dans le code. On pouvait, jusqu'à la version 3 utiliser le projet *Ehcache Annotations for Spring*. La fonctionnalité "Cache Abstraction" est bien plus puissante. Illustrons son fonctionnement sur une couche d'accès en base.

```
public interface TicketingRepository {

  @Cacheable(«ticket»)
```

```
  Ticketing findById(Long ticketingId);
}
```

Grâce à l'annotation *@Cacheable*, seul le premier appel à la méthode *findById* interrogera la base de données, les appels suivants seront retournés directement depuis le cache nommé *ticket*.
Les systèmes de cache étant basés sur le concept de clé/valeur, nous avons besoin d'une clé pour stocker les tickets. Par défaut, Spring utilise un hash de tous les paramètres de la méthode.
Lors de la mise à jour des tickets, on voudra bien sûr que le cache soit vidé. L'annotation *@CacheEvict* est là pour cela.

```
  @CacheEvict(value = «ticket», allEntries = true)
  void updateNbTickets(Ticketing ticketing);
```

Dans cet exemple, la mise à jour d'un ticket supprimera toutes les entrées du cache *ticket*. On peut bien sûr configurer la suppression plus finement en spécifiant une clé. Ainsi uniquement l'entrée de cache correspondant au ticket modifié sera supprimée.

```
  @CacheEvict(value = «ticket», key = «#ticketing.id»)
```

La syntaxe #… est basée sur l'expression langage de Spring, *SpEL*, disponible depuis Spring 3.0 et introduit dans l'article du mois dernier. On dispose donc de toute la puissance de ce langage pour affiner la configuration de l'utilisation du cache.
Il est ainsi possible de spécifier des mises en cache conditionnelles :

```
@Cacheable(value = «tickets», key = «#ticketingBand», condition =
«#ticketingBand.startsWith('A')»)
  List<Ticketing> findByBand(String ticketingBand, boolean audit);
```

Dans l'exemple précédent, on utilise uniquement le paramètre *ticketingBand* comme clé de cache et surtout, on ne met en cache que les tickets dont le nom du groupe commence par A.
La puissance de cette fonctionnalité de cache réside aussi dans l'abstraction de l'implémentation du cache sous-jacent. Les implémentations actuellement supportées sont un cache à base de *ConcurrentMap* et *Ehcache*. Une implémentation avec un backend GemFire (cache mémoire distribué et scalable), est prévue pour la prochaine release de Spring et déjà disponible sur GitHub.
Pour activer le cache, la configuration Spring doit se voir ajouter le namespace *cache:annotation-driven*. Avec l'implémentation Ehcache, on aurait ainsi :

```
  <cache:annotation-driven cache-manager=»ehCacheManager» />

  <bean id=»ehCacheManager» class=»org.springframework.cache.
ehcache.EhCacheCacheManager»
    p:cache-manager-ref=»ehCacheFactory» />

  <bean id=»ehCacheFactory»
class=»org.springframework.cache.ehcache.EhCacheManagerFactory
Bean»
p:config-location=»classpath:/ehcache.xml» />
```

Il faut en outre définir la configuration propre à Ehcache. Restons très simple et utilisons tous les paramètres par défaut :

```
<ehcache>
  <cache name=»ticket» timeToLiveSeconds=»100» />
  <cache name=»tickets» timeToLiveSeconds=»100» />
</ehcache>
```

On retrouve les régions Ehcache *ticket* et *tickets* référencées dans les annotations *@Cacheable* et *@CacheEvict*.
Pour l'instant la configuration est uniquement possible par annotations, il est prévu de pouvoir utiliser la configuration Java (voir article du mois dernier) pour la version 3.1.0.M2 et la configuration XML dans une prochaine milestone.

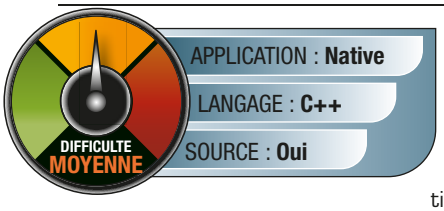
CONCLUSION

La version 3.1 de Spring n'est pas encore en release finale, mais les premières milestones présagent des nouveautés très intéressantes, notamment autour de Spring MVC et l'abstraction des caches. Cette dernière fonctionnalité sur les caches couvre une grande partie des cas d'usage de complexité simple et moyenne avec élégance. Pour les cas plus complexes, on préfèrera l'utilisation de framework dédié.
Quant à la version 3.2 de Spring, déjà en partie planifiée et prévue pour 2012, elle devrait, entre autres, apporter le support de Java SE 7 (JDBC 4.1, support du framework fork-join, etc.). On peut néanmoins s'interroger sur un éventuel rapprochement de Spring Core et des API JEE 6 et 7, rappelons que les pères de Spring se sont abstenus lors du vote du JCP sur CDI (Contexts and Dependency Injection for the Java EE Platform) 1.1, une des API qui s'apprête à être au cœur de JEE 7.

■ Agnès **Crepet**
Architecte JEE, JDuchess & Lyon JUG Leader Twitter : http://twitter.com/agnes_crepet

■ Vladislav **Pernin**
Architecte JEE chez Zenika
Twitter : http://twitter.com/vladislavpernin

Concurrency Runtime et la programmation parallèle avec Visual C++



Concurrency Runtime ou ConcRT est un framework pour la programmation parallèle en C++ qui permet de construire des applications robustes, scalables et avec un niveau d'abstraction qui évite au développeur de gérer la tuyauterie complexe inhérente aux applications multithreads. La promotion de cette librairie à pour but de faciliter le développement des applications multi-cores et ainsi, d'exploiter pleinement les capacités CPU des machines.

LE CONCURRENCY RUNTIME

ConcRT utilise un gestionnaire de tâches et des primitives de blocage coopératifs. L'architecture générale est décrite ici : [Fig.1].

- *Parallel Patterns Library*
  PPL fournit des classes collections (containers) et des algorithmes pour réaliser des opérations parallèles et des opérations distribuées sur des ensembles de données ou des collections. PPL permet aussi le parallélisme des tâches en fournissant des objets Task répartis sur les ressources matérielles.
- *Asynchronous Agents Library*
  Agents Library permet de faire communiquer différents éléments d'un programme au travers d'interfaces et met en œuvre un pipeline évolué.
- *Task Scheduler*
  Le Task Scheduler gère et coordonne les tâches à l'exécution. Le Task Scheduler est coopératif.
- *Resource Manager*
  Le Resource Manager a pour rôle de gérer les ressources mémoire et processeur.

Les fichiers d'entêtes suivants sont nécessaires pour tirer profit des différentes parties de CR :

Composant	Fichier d'entêtes
Parallel Patterns Library (PPL)	ppl.h
	concurrent_queue.h
	concurrent_vector.h
Asynchronous Agents Library	agents.h
Task Scheduler	concrt.h
Resource Manager	concrtrm.h

L'ensemble des classes et templates qui composent ConcRT sont incorporés dans le namespace Concurrency. Le code source est intégralement fourni dans le répertoire Visual Studio 2010 nommé CRT disponible sous C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\crt\src.

LES CLASSES DE COLLECTIONS THREAD-SAFE INSPIRÉES DE INTEL TBB

La mise au point d'applications multithreads se heurte souvent à l'utilisation de classes qui ne sont pas thread-safe ou partiellement. Ainsi, les templates de la librairie standard C++ STL peuvent être accédés par plusieurs threads en lecture mais les opérations d'écriture impliquent que les autres opérations de lecture/écriture doivent être protégées. PPL introduit les containers concurrent_queue et concurrent_vector qui permettent des opérations parallèles. Ces containers sont inspirés de l'implémentation Intel Threading Building Blocks alias TBB. De futures extensions introduisent les containers hash, set, map et list.

DÉMARRER AVEC PPL

La seule chose à faire est d'inclure le fichier d'entête ppl.h et éventuellement de faire un using namespace Concurrency pour éviter de renseigner Concurrency:: à chaque appel de primitives, classes ou templates. Exemple dans un programme console simple qui affiche l'identifiant du thread utilisé et fait une attente d'une seconde :

```
// Parallel Patterns Library (PPL) header
#include <ppl.h>
using namespace Concurrency;

void DoWhat(const std::string & command)
{
 Logger::LogDebug(«Enter ...» + command);
 wait(1000);
 Logger::LogDebug(«Leave ...» + command);
}

void Call_Parallel_Invoke_Simple()
{
 parallel_invoke(
   []() { DoWhat(«Go North»); },
   []() { DoWhat(«Go South»); }
   );
}

int _tmain(int argc, _TCHAR* argv[])
{
 Logger::LogDebug(«Enter main...»);
 Call_Parallel_Invoke_Simple();
 return 0;
}
```

La sortie sur la console indique que deux threads sont utilisés :

```
[DEBUG] [11:57:53.627] [TID:6488] Enter main...
[DEBUG] [11:57:53.628] [TID:6488] Enter ...Go South
[DEBUG] [11:57:53.628] [TID:5572] Enter ...Go North
[DEBUG] [11:57:54.628] [TID:6488] Leave ...Go South
[DEBUG] [11:57:54.628] [TID:5572] Leave ...Go North
```

L'avantage de cette encapsulation de la gestion des threads (qui sera un jour standardisée et donc multiplateforme) permet d'éviter une construction programmes en utilisant les API spécifiques pour la gestion des threads ou la mise au point d'une classe de gestion d'un pool de threads. De plus, nous allons voir plus loin, que la mon-

tée en charge de l'application est liée au nombre de processeurs (et cores) disponibles sur la machine qui exécute le code. Le ConcRT gère cela de manière native via les API Windows. On entrevoit donc ici une manière élégante de gérer les tâches avec un support objet et fortement typé.

LES BOUCLES PARALLÈLES

PPL introduit des boucles parallèles nommées parallel_for et parallel_for_each. La boucle parallel_for_each fonctionne comme son équivalent de la librairie standard C++ STL std::for_each. Attention, ces 2 boucles ne garantissent pas l'ordre de traitement. Il est aussi possible de spécifier une taille de pas pour permettre d'exécuter une boucle séquentielle dans une boucle parallel_for.

```
void DoParallel(size_t value)
{
 Logger::LogDebug(«Enter DoParallel...»);
 // Affiche la valeur du paramètre value
 ostrstream temp;
 temp << «Value is « << value << ends;
 Logger::LogDebug(temp.str());
 Logger::LogDebug(«Leave DoParallel...»);
}

void Call_Parallel_For()
{
 // Generation aléatoire de nombres dans le vector v
 unsigned int value = 0;
 vector<unsigned int> v;
 for(int i=0 ; i<10 ; i++)
 {
   rand_s(&value);
   v.push_back(value);
 }

 size_t total = v.size();

 Logger::LogDebug(«CALL parallel_for...»);
 // parallel_for -> l'indice du for est passé à la fonction DoParallel
 parallel_for(size_t(0), total, &DoParallel);

 Logger::LogDebug(«CALL parallel_for_each...»);
 // parallel_for_each -> chaque valeur du vecteur est passée à
la fonction
 parallel_for_each(v.begin(), v.end(), &DoParallel);
}
```

LES TÂCHES PARALLÈLES

La méthode parallel_invoke est la primitive la plus simple qui permet d'exécuter des tâches en parallèle et rend la main une fois qu'elles sont terminées. La fonction parallel_invoke accepte jusqu'à 10 fonctions en paramètre :

```
void Call_Parallel_Invoke()
{
 parallel_invoke(
   []() { DoWhat(«Left»); },
   []() { DoWhat(«Right»); },
   []() { DoWhat(«Up»); },
   []() { DoWhat(«Down»); },
   []() { DoWhat(«North»); },
   []() { DoWhat(«East»); },
   []() { DoWhat(«Sud»); },
   []() { DoWhat(«West»); }
   );
}
```

En interne, la fonction parallel_invoke() crée des tâches et attend la fin de leur exécution en utilisant la classe task_group et les méthodes run() et wait(). Ainsi, la fonction suivante est équivalente à la fonction Call_Parallel_Invoke_Simple :

```
void Call_Parallel_Invoke_Simple_Equivalent()
{
 // Comment traduire le code suivant avec la classe task_group ???
 //
 //parallel_invoke(
 //[]() { DoWhat(«Go North»); },
 //[]() { DoWhat(«Go South»); }
 //);

 task_group tg;
 tg.run([](){ DoWhat(«Go North»); });
 tg.run([](){ DoWhat(«Go South»); });
 tg.wait();
}
```

Avec PPL, les tâches (tasks) sont démarrées et gérées à travers l'utilisation de la classe task_group. La méthode run() permet de créer et de planifier une tâche. La méthode wait() permet d'attendre la fin d'exécution d'un groupe de tâches. A la différence des threads, les tâches ne démarrent pas immédiatement mais sont placées dans une queue qui est utilisée au fur et à mesure que les ressources processeur sont disponibles. La performance d'un programme varie en fonction du nombre de cores disponibles. La gestion des threads par le ConcRT est extrêmement efficace. L'interruption d'une tâche se fait via la méthode task_group ::cancel() . L'appel à cette méthode dans n'importe quel thread entraîne l'arrêt de toutes les tâches en cours, y compris celles qui sont planifiées.

```
void Call_Run_And_Cancel()
{
 task_group tg;
 tg.run([](){ DoWhat(«Go North»); });
 tg.run([](){ DoWhat(«Go South»); });
 // Attente pour permettre aux 2 tâches de démarrer...
 wait(500);
 tg.cancel();
 Logger::LogDebug(«Task Group cancel()...»);
 if( tg.is_canceling() == true )
   Logger::LogDebug(«Task Group en cours d'arret...»);
 // A partir d'ici, plus rien ne sera planifié !
 tg.run([](){ DoWhat(«Go East»); });
 tg.run([](){ DoWhat(«Go West»); });
 // Retourne le status code du task group
 task_group_status status = tg.wait();
}
```

L'appel à la méthode task_group ::cancel() fait passer le groupe de tâches dans un état (task_group_status::canceled) qui fait que l'appel à la méthode is_cancelling() retournera true.

LE BLOCAGE COOPÉRATIF

Avec le blocage coopératif, une tâche peut suspendre son exécution et redonner la main au Task Scheduler pour réutiliser des ressources et donc optimiser l'usage de ces ressources pour exécuter d'autres tâches. Contrairement aux primitives de bas niveau du système d'exploitation (Windows), les ressources processeur sont optimisées. Le blocage coopératif permet d'améliorer les performances des applications parallèles en faisant une utilisation pleine des ressources CPU. Les principaux objets de synchronisation sont de type verrou simple (classe critical_section), verrou de type SWMR alias Single Writer Multiple Readers (classe reader_writer_lock), event manuel (classe event).

Liste des opérations de blocage coopératif :

Opérations de blocage coopératif	Description
task_group::wait	attend la fin d'une tâche ou d'un groupe de tâches
critical_section::lock	acquisition d'un verrou de type section critique
critical_section::scope_lock	version exception safe
reader_writer_lock::lock	acquisition d'un verrou de type reader/writer en écriture
reader_writer_lock::scope_lock	version exception safe
reader_writer_lock::lock_read	acquisition d'un verrou de type reader/writer en lecture
reader_writer_lock::scope_lock_read	version exception safe
event::wait	equivalent à un manuel reset event
agent::wait	attend la fin de l'agent
agent::wait_for_all, wait_for_one	attend la fin de un ou plusieurs agents
Concurrency::wait	blocage pendant un intervalle
Context::Block	interne a CR
Context::Yield parallel_for, parallel_for_each,	donne la main à un autre contexte d'exécution
parallel_invoke	fonctions pour les algorithmes parallèles
send, asend	fonctions de transmissions de données par messages
receive	fonction de reception de message

LES TÂCHES ET LES THREADS

Lorsqu'une tâche doit s'exécuter, le Task Scheduler appelle la routine de cette tâche dans un thread de son choix. La tâche ne change pas de thread. C'est une garantie pour le code ayant une affinité avec un thread ou un usage par exemple du TLS (Thread Local Storage). La création d'une tâche est une opération moins coûteuse que la création explicite d'un thread. Il est possible pour une application de créer des centaines ou des milliers de tâches tout en tournant efficacement. Ceci n'est pas possible avec des threads explicites. La conception d'une application en tâches est moins compliquée que la conception d'une application à base de threads.

CRÉATION DES TÂCHES

La création des tâches peut se faire avec des fonctions lambda ou de simple pointers de fonctions. La classe task_group peut être utilisées de différentes manières. Pour une déclaration détaillée, la classe task_group utilise la méthode run() surchargée qui prend en paramètre un task_handle dans lequel on spécifie explicitement le nom de la fonction à utiliser. La syntaxe moderne du style C++ autorise l'utilisation du mot clé auto pour masquer le type retourné par la fonction make_task. La nouvelle norme C++ TR1 introduit des classes templates pour la gestion des pointeurs de fonctions et il est possible de voir toutes ces formes d'écriture dans une même fonction. Cependant, les fonctions à utiliser doivent avoir une signature de type void function(void).

```
void DoSimpleTask(void)
{
 Logger::LogDebug(«Enter DoSimpleTask...»);
 wait(1000);
 Logger::LogDebug(«Leave DoSimpleTask...»);
}


void Call_Task_Group_Run()
{
 // Déclaration du Task Group
 task_group g;
 // Avec une déclaration explicite de task_handle et de prototype
 task_handle<void (*)(void)> t1 = make_task(&DoSimpleTask);
 // Avec auto
 auto t2 = make_task(&DoSimpleTask);
 // Avec std::tr1::function
 std::tr1::function<void (void)> f3 = &DoSimpleTask;
 auto t3 = make_task(f3);
 // Création des tâches
 g.run(t1);
 g.run(t2);
 g.run(t3);
 g.run( &DoSimpleTask );
 g.wait();
}
```

LES NOUVEAUTÉS DU CONCURRENCY RUNTIME

Il existe un pack nommé ConcRT Sample Pack v0.4 qui contient des exemples de code et des extensions pour le framework Concurrency. Ce pack est disponible sur http://archive.msdn.microsoft.com/concrtextras. Il contient de nombreux fichiers d'entêtes :

Composant	Fichier d'entêtes
Extensions pour Agents Library	agents_extras.h
Event pour PPL	barrier.h
Queue particulière	bounded_queue.h
ConcRT helpers	concrt_extras.h
map thread safe	concurrent_unordered_map.h
set thread safe	concurrent_unordered_set.h
helpers	connect.h
template de classe hash thread safe	internal_concurrent_hash.h
template de classe list thread safe	internal_split_ordered_list.h
Extensions des algorithmes PPL	ppl_extras.h
Extensions PPL	ppltasks.h
classe semaphore	semaphore.h

Les éléments sont contenus dans le namespace Concurrency::samples. Ces nouvelles fonctionnailités tirent parti du standard C++ TR1. Le fichier d'entêtes pppltasks.h introduit de nouveaux templates et des nouvelles classes pour gérer les tâches. Ainsi, on y trouve le template de classe task<> qui prend en paramètre dans son constructeur un objet de type std::tr1::function. La fonction passée à la classe task peut aussi retourner un type donné et ainsi ressembler à un thread classique qui retourne, généralement 0 en cas de succès :

```
int DoSimpleTask_Random()
{
 Logger::LogDebug(«Enter DoSimpleTask_Random...»);
 wait(1000);
 // Génération d'un nombre aléatoire de 1..1000
 unsigned int r = 0;
 srand(1); rand_s(&r);
 r = (unsigned int) ((double)r / ((double) UINT_MAX + 1 ) * 1000.0) + 1;
 Logger::LogDebug(«Leave DoSimpleTask_Random...»);
 return r;
}

void Call_Task_Group_Run_Extras()
{
 std::tr1::function<int (void)> f1 = &DoSimpleTask_Random;
 task<int> t1(f1);
 t1.wait();
 int ret = t1.get();
 cout << «Tasks t1 returns « << ret << endl;

 std::tr1::function<void (void)> f2 = &DoSimpleTask;
 task<void> t2(f2);
 t2.wait();
}
```

Le ConcRT sample pack introduit des opérateurs qui rendent la syntaxe plus élégante. Pour créer deux tâches et attendre leur terminaison, il suffit d'utiliser l'opérateur &&.

```
void Call_Task_Group_Run_Extras_WithOperator()
{
 std::tr1::function<int (void)> f1 = &DoSimpleTask_Random;
 task<int> t1(f1);
 std::tr1::function<int (void)> f2 = &DoSimpleTask_Random;
 task<int> t2(f2);
 (t1 && t2).wait();
 cout << «Tasks t1 returns « << t1.get() << endl;
 cout << «Tasks t2 returns « << t2.get() << endl;
}
```

Pour enchaîner la création d'une tâche une fois que les deux premières tâches sont terminées, il suffit d'utiliser la méthode continue_with() :

```
void Call_Task_Group_Run_Extras_WithOperator2()
{
 std::tr1::function<void (void)> f1 = &DoSimpleTask;
 task<void> t1(f1);
 std::tr1::function<void (void)> f2 = &DoSimpleTask;
 task<void> t2(f2);
 std::tr1::function<void (void)> f3 = &DoSimpleTask;
 auto t3 = (t1 && t2).continue_with(f3);
 t3.wait();
}
```

LA GESTION DU TASK SCHEDULER

Il est possible de choisir la manière avec laquelle le ConcRT va gérer les tâches à travers l'utilisation de threads. La classe SchedulerPolicy permet de spécifier le nombre minimum et maximum de ressources CPU (processeurs virtuels) via les propriétés MinConcurrency et MaxConcurrency. La propriété SchedulerKing permet de choisir le type de threads : Win32 ou UMS. UMS ou User-mode scheduling est un mécanisme léger de gestion de threads et disponible dans les versions 64 bits de Windows 7 et Windows Server 2008 R2.

```
 SchedulerPolicy policy(2,
MinConcurrency, 1,
MaxConcurrency, 2);
   CurrentScheduler::Create(policy);
```

Il est aussi possible de contrôler finement la manière avec laquelle les tâches sont réparties en créant un groupe de scheduler via les classes Scheduler et ScheduleGroup. La méthode ScheduleTask() permet aussi de créer des tâches légères (light-weight task). La librairie Agents fait une utilisation intensive de cette méthode. ConcRT ne gère pas et ne capture pas les exceptions générées par les tâches légères. De plus, la terminaison d'une tâche légère n'est pas signalée par le runtime ConcRT.

OPTIMISATION DES ROUTINES D'ALLOCATION MÉMOIRE

Les allocations de mémoire sont des opérations qui peuvent être coûteuses dans les programmes multi-thread. Il existe plusieurs manières d'allouer de la mémoire, via les routines C classiques comme malloc()/free() ou en utilisant les mots clés C++ new/delete. ConcRT introduit deux fonctions nommées Alloc() et Free() qui allouent des blocs de mémoire en utilisant un cache (sub allocator) qui permet d'éviter l'utilisation de verrous ou de barrière mémoire. Les plages de mémoire allouées sont dédiées à chaque thread et donc ne peuvent pas être partagées.

LA LIBRAIRIE ASYNCHRONOUS AGENTS

Asynchronous Agents permet de créer facilement des classes qui se comportent comme des agents indépendants - en utilisant les tâches du Task Scheduler - et qui peuvent communiquer au travers d'un mode de messages, très sophistiqué en utilisant les fonctions send(), asend() et receive(). Cette partie fera l'objet d'un article dédié.

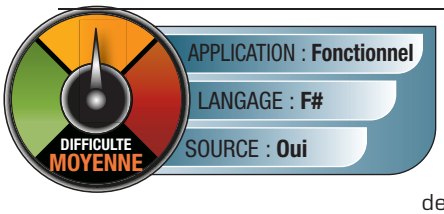
LES INTERFACES DU CONCURRENCY RUNTIME

Pour faciliter la tâche des développeurs, le ConcRT se base entièrement sur les API Windows et masque la complexité d'introspection de la machine sur laquelle tourne l'application. ConcRT détecte le nombre de processeurs physiques et virtuels présents, la version de l'OS pour vérifier le support spécifique comme UMS sur Windows 7 ou Windows Server 2008R2. De plus, ConcRT déclare un nombre important d'interfaces et de méthodes qu'il est possible de surcharger pour permettre la modification ou l'amélioration des routines associées au Task Scheduler, entre autres. Visual C++ 2010 fournit une nouvelle manière de réaliser des applications C++ multithreads, et utilisable au travers de la simple redistribution de la runtime C et donc facilement disponible en standard sur l'ensemble des machines et serveurs modernes d'entreprise. Seul C++ permet d'exploiter pleinement les ressources matérielles CPU disponibles. A vous de jouer !

■ Christophe **Pichaud** - *christophepichaud@hotmail.com*
MCSD, MCSD.NET - Consultant sur les technologies Microsoft

Découvrir la programmation fonctionnelle avec F# sur la plateforme .NET

Encore peu populaire, la programmation fonctionnelle présente pourtant de nombreux attraits. Bénéficiant de la puissance de la plateforme .NET, le langage F# est une excellente opportunité de se familiariser avec elle.



Au commencement (de l'informatique) étaient les langages Fortran et LISP. Le premier doté d'une syntaxe dictée par la constitution des cartes perforées, et par définition proche des machines, le second doté d'un haut niveau d'abstraction et par définition proche des mathématiques. Et dès le début de l'informatique, ce fut une scission entre les langages impératifs à la Fortran, et les langages fonctionnels à la LISP. Puis est apparue la programmation objet, fondamentalement impérative, mais avec un niveau d'abstraction supérieur à celle d'un Fortran, ou d'un langage C pour prendre un autre exemple. Nous voici donc avec trois paradigmes de programmation qui ont chacun leurs partisans, ceux de la programmation fonctionnelle étant hélas, les moins nombreux. Hélas, car c'est, de l'humble avis de votre serviteur, un problème essentiellement culturel, les langages fonctionnels étant en général enseignés après les autres. Hélas, car le paradigme fonctionnel a des qualités indéniables, au niveau de la concision, de l'expressivité et surtout de la sûreté du code. Seuls les langages fonctionnels permettent de prouver le code, c'est-à-dire de démontrer qu'il s'exécutera comme prévu. Le fossé entre les paradigmes est-il impossible à combler ? Sans doute pas. Une remarquable tentative a vu le jour dès 1996 avec le langage Objective Caml. Ce langage conçu en France par l'INRIA concilie les trois paradigmes. Malheureusement il n'a pas su devenir populaire. Peut-être à cause d'une syntaxe parfois un peu difficile, mais surtout par manque de promotion, en comparaison d'un C# ou d'un Java. Cependant, lorsque les projets sont ardus, comme pour l'écriture d'un compilateur, ou pour vérifier le code des systèmes embarqués de l'Airbus, c'est Objective Caml qui est utilisé…

1 F#, UN LANGAGE MULTIPARADIGME POUR .NET

Ce fossé entre les paradigmes, c'est peut-être F# qui va le combler. F# est un langage qui a été conçu par Microsoft pour sa plate-forme .NET. Au départ c'était le langage fonctionnel pur Haskell qui a servi de base de travail aux laboratoires de Microsoft. Puis ceux-ci se sont tournés vers Objective Caml. F# présente donc de nombreux point communs avec ce dernier, à tel point que des morceaux de code OCaml peuvent être directement réutilisés. Ou pour être exact, ceci était vrai avec les premières moutures de F#. C'est moins vrai maintenant avec la version définitive 2.0, mais les similitudes restent extrêmement fortes. Ainsi, il n'est pas exagéré de dire que F# est un Objective Caml pour .NET. F# est lui aussi multi-paradigme. Impératif, objet ou fonctionnel selon les besoins du programmeur, F# bénéficie en outre de toute la plate-forme .NET et des librairies de classes, et il supporte même la programmation générique. L'ensemble en fait un outil remarquablement souple et puissant, et, soutenu par un éditeur aux grands moyens, il devrait enfin rendre la programmation fonctionnelle plus populaire.

2 QU'EST-CE QU'UN LANGAGE FONCTIONNEL ?

Vaste sujet… Pour faire simple, disons que la programmation fonctionnelle traite le calcul comme l'évaluation de fonctions mathématiques et évite les données mutables. Un langage fonctionnel qui rejette totalement l'affectation et donc les données mutables est dit être un langage fonctionnel pur. C'est le cas de Haskell. Un langage fonctionnel qui accepte les données mutables est dit impur. C'est le cas du LISP moderne et aussi de F#. Dans tous les cas, les fonctions sont des valeurs du langage, puisqu'elles s'évaluent, et comme telles, peuvent être passées en argument à une fonction ou bien être une valeur de retour. Autrement dit, en programmation fonctionnelle, une fonction est une entité de première classe.

3 PREMIERS PAS

F# est intégré à Visual Studio 2010. Si vous possédez Visual Studio 2008, il suffit de télécharger un plugin à www.fsharp.net. Si l'on ne souhaite pas travailler avec Visual Studio, on pourra travailler avec des outils en ligne de commande que l'on obtiendra à la même adresse. Bien entendu, il est nécessaire que le framework .NET soit présent sur la machine de travail. Programmation fonctionnelle ou pas, la tradition est la tradition. Voici donc le fameux programme minimum 'Hello World!'

```
open System

printf «Programmez!\n»
```

Nous sommes sur la plateforme .NET et nous commençons par exprimer que nous voulons utiliser les fonctionnalités de base avec open System qui est l'équivalent d'un using System de C#. La ligne suivante n'appelle guère de commentaire et pour l'instant, que nous fassions de la programmation fonctionnelle ne saute pas aux yeux. Réécrivons un peu notre code:

```
open System

let print_msg msg =
    printf msg
```

```
    printf «Abonnez-vous ! :-)\n»

print_msg «Programmez!\n»
```

Dans ce code nous définissons une fonction: print_msg. La définition n'est pas une affectation en dépit de la présence du signe égal. L'instruction let crée une liaison (binding) entre le nom de la fonction et les arguments qu'elle reçoit, et le code qui sera exécuté lorsque la fonction sera invoquée. On remarque encore que le type de l'argument reçu par la fonction n'est pas donné dans la déclaration. Une des caractéristiques des langages fonctionnels est qu'ils sont dotés d'un système d'inférence de type. Un système d'inférence de type est très différent du typage dynamique. F# est un langage statiquement typé. Le système d'inférence de type dispense le programmeur des déclarations fastidieuses tout en veillant au grain. Ainsi si dans notre code nous écrivons :

```
print_msg 2
```

Le compilateur va se plaindre:

```
error FS0001: The type 'int' is not compatible with the type
 'Printf.TextWriterFormat<unit>'
```

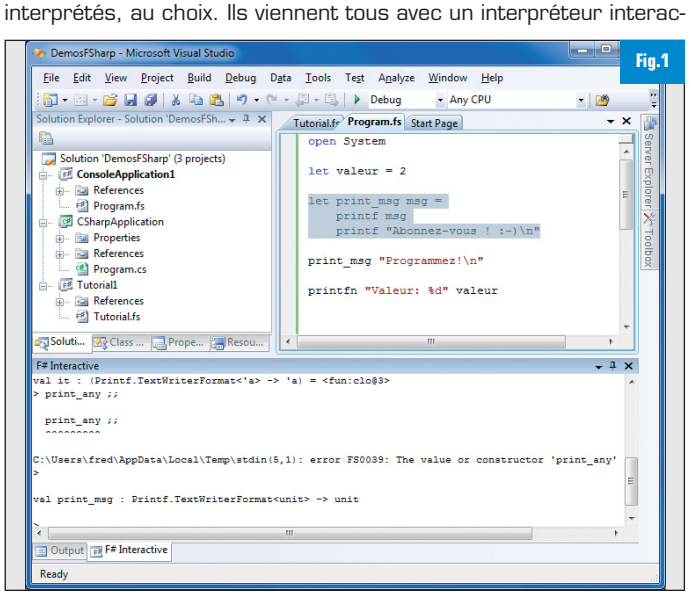
En clair, une valeur entière, un int, n'est pas ce que la fonction attend. Si le langage était dynamiquement type, à l'instar d'un PHP ou d'un Python, le code aura fonctionné sans problème. Revenons un instant sur l'instruction let. Nous aurions pu par exemple écrire:

```
let valeur = 2
printfn «Valeur: %d» valeur
```

Dans ce cas, valeur est … une valeur. Ce qu'il faut remarquer, c'est que la syntaxe est la même pour une simple valeur et pour une fonction. Logique, puisqu'une fonction est une valeur en F#.

4 L'INTERPRÉTEUR INTERACTIF

Les langages fonctionnels peuvent en général être compilés ou interprétés, au choix. Ils viennent tous avec un interpréteur interactif, ou REPL, pour Real-Eval-Print-Loop. Sous Visual Studio, c'est la combinaison de touches Ctrl-Alt-F qui le fait apparaître. Plus pratique encore, mettre une portion de code en surbrillance dans l'éditeur et appuyer sur la combinaison de touches Alt-Return, fait évaluer le code sélectionné dans l'interpréteur, comme le montre l'illustration. [Fig.1] L'interpréteur à évalué notre fonction print_msg. Comme aucun argument n'a été transmis pour l'évaluation, l'interpréteur nous dit ce qu'est notre fonction en nous donnant sa signature :



Evaluation d'un morceau de code dans l'interpréteur interactif.

```
val print_msg : Printf.TextWriterFormat<unit> -> unit
```

Il est également possible d'obtenir des informations sur le code en le survolant avec le curseur de la souris. [Fig.2] On remarque que cette fonction retourne 'unit' qui dans le code, s'écrit si besoin (). Unit est un type vide. Un peu l'équivalent du void de C. En effet, notre fonction écrit sur la console, mais elle ne calcule rien, ce qui est différent. Donc elle retourne une valeur vide. Car une fonction retourne nécessairement quelque chose. Ce quelque chose étant la dernière évaluation. Dans notre exemple c'est le résultat de printf qui, elle, retourne unit.

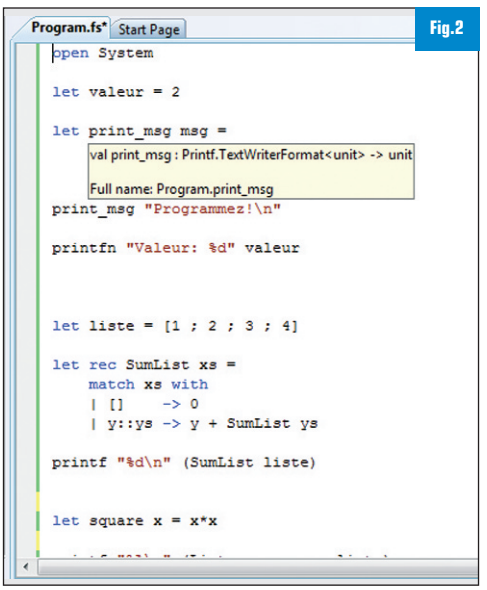
5 LISTES ET RÉCURSIVITÉ

Listes et récursivité sont omniprésentes en programmation fonctionnelle. La récursivité à mauvaise réputation en programmation impérative en ce qui concerne les performances et l'encombrement de la pile. Avec les langages fonctionnels, il en va autrement et employer la récurisivité est naturel, comme c'est le cas, si on prend le temps d'y réfléchir, dans la vraie vie. Voici un programme qui reçoit une liste d'entiers et qui calcule la somme des entiers de la liste :

```
let liste = [1; 2; 3; 4]

let rec SumList xs =
        match xs with
        | []    -> 0
        | y::ys -> y + SumList ys

printf «%d\n» (SumList liste)
```



Promenez votre souris sur un morceau de code F# et Visual studio l'analysera à la volée.

La première instruction *let* commence par définir la liste. Attention au ; qui en sépare les éléments. La seconde instruction définit une fonction récursive. Pour cela, la présence du mot-clé *rec* est requise, sinon le compilateur va éprouver un vide existentiel. Le corps de la fonction nous en apprend beaucoup. Nous avons ce qui s'appelle un filtrage de motif. Ainsi l'instruction match teste si l'argument reçu par la fonction est une liste vide ([]). Si oui, nous avons terminé notre travail et dans ce cas nous retournons une valeur nulle (ne pas confondre avec unit). Si la liste n'est pas vide, c'est que la liste est constituée d'une tête et d'une queue (cette dernière étant éventuellement la liste vide []). Dans ce cas, l'argument de la fonction est réexprimé en une tête **y** et une queue **ys** assemblées par l'opérateur cons ::. Alors la tête de la liste est ajoutée à la queue de la liste traitée par notre fonction. Dit autrement, chaque élément de la liste est ajouté à celui qui le suit, jusqu'à ce que la liste soit entièrement parcourue.

6 LES FONCTIONS SONT DES VALEURS

Comme nous l'avons dit au début de cet article, les fonctions sont des valeurs. Concrètement, nous pouvons passer une fonction en argument à une autre fonction, qui sera dite fonction de premier ordre. En programmation fonctionnelle, il est très courant d'utiliser une fonction de premier ordre pour appliquer une fonction à tous les éléments d'une liste. Ainsi, nous pourrions appliquer à la liste d'entiers du paragraphe précédent, une fonction qui calcule les carrés de chacun des éléments. Cela peut s'écrire ainsi :

```
let square x = x*x

printf «%A\n» (List.map square liste)
printf «%A\n» (List.map (fun x -> x*x) liste)
```

Nous commençons par définir une fonction qui calcule le carré d'un entier, puis dans la deuxième ligne de cet exemple nous appliquons cette fonction. La fonction de premier ordre utilisée pour cela s'appelle *map* et c'est une fonction du module List de F#, d'où l'écriture List.map. Cette fonction reçoit deux arguments. La fonction à appliquer, et la liste sur laquelle on applique, et c'est tout. La troisième ligne de notre exemple fait le même travail, mais avec une fonction anonyme.

7 LES TYPES COMPLEXES

Pour mieux découvrir les possibilités de F# dans le paradigme fonctionnel, je vous propose maintenant d'étudier un petit bout de code qui fait partie du «tutorial» de l'addon de Visual Studio. Cela n'a de tutorial que le nom, puisque rien n'est expliqué. Voici le code :

```
type Expr =
        | Num of int
        | Add of Expr * Expr
        | Mul of Expr * Expr
        | Var of string

let rec Evaluate (env:Map<string,int>) exp =
        match exp with
        | Num n -> n
        | Add (x,y) -> Evaluate env x + Evaluate env y
        | Mul (x,y) -> Evaluate env x * Evaluate env y
        | Var id    -> env.[id]

let envA = Map.ofList [ «a»,1 ;
                        «b»,2 ;
                        «c»,3 ]

let expT1 = Add(Var «a», Mul(Num 2, Var «b»))
let resT1 = Evaluate envA expT1

printf «%A\n» resT1
```

Ce code est difficile pour celui qui, arrivant sur F#, ne connaît pas OCaml ou plus généralement la programmation fonctionnelle. Ce code commence par déclarer un nouveau type, *Expr*, puis définit des constructeurs du type. Le sens du mot constructeur est ici très différent du sens de la programmation objet. Selon la déclaration, une Expr peut s'appeler *Num* et dans ce cas le type 'contient' un entier: *Num of int*. Il peut s'appeler *Add* et dans ce cas, il contient deux Expr: Add of Expr * Expr. Ici le signe * ne doit pas induire en erreur, il n'a aucun rapport avec la multiplication. Cette déclaration a quelque chose de fascinant et d'unique à la programmation fonctionnelle : elle définit un type en fonction de lui-même, autrement dit un type récursif. Au total Expr est un ensemble de deux conteneurs de valeurs, et de deux conteneurs d'Expr, qui comme leurs noms le suggèrent très fortement serviront pour faire des opérations sur les Expr. Expr est un type complexe, qui porte parfois le nom d'union discriminée. Passons directement à la fin du code. Une Map, au sens dictionnaire du terme, est déclarée, ainsi qu'une expression compliquée, expT1, construite à partir d'Expr. Le tout est passé à une fonction Evaluate, puis le résultat de l'évaluation est affiché sur la console. Examinons maintenant la fonction Evaluate. Il s'agit d'une fonction récursive. Elle reçoit d'abord un argument du nom de env dont le type, Map<string,int>, est ici explicitement précisé. Le système d'inférence de type a malgré tout ses limites. Elle reçoit ensuite une expression Expr. Il y a ensuite un filtrage de motif sur l'Expr. Dans chaque cas, le type est déconstruit, et son contenu est traité. Si on a affaire à une valeur (Num) on retourne simplement celle-ci. Si on a affaire à une clé de dictionnaire (Var) on retoune la valeur associée. Enfin si c'est une expression, on évalue son contenu en réinvoquant notre fonction. Au final, le code affiche le résultat: 2*2+1=5; En quelques lignes de code est ainsi écrit l'interpréteur d'un langage rudimentaire! Si ce code ne vous paraît encore pas très clair, c'est le moment de signaler que le débogueur de Visual Studio permet de tracer pas à pas le code F#, aussi bien qu'il le fait avec C# ou VB.NET. Le comportement du code vous apparaîtra alors plus clair.

8 AUPRÈS DE MON ARBRE

Une utilisation classique des types récursifs est le travail avec les structures arborescentes. Le code ci-dessous, tiré de MSDN, définit un arbre binaire dont les nœuds sont associés à une valeur entière :

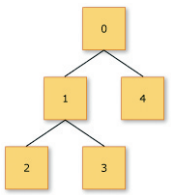
```
type Tree =
    | Tip
    | Node of int * Tree * Tree
```

```
let rec sumTree tree =
    match tree with
    | Tip -> 0
    | Node(value, left, right) ->
        value + sumTree(left) + sumTree(right)

let myTree = Node(0, Node(1, Node(2, Tip, Tip), Node(3, Tip, Tip)),
 Node(4, Tip, Tip))


printf «Valeur de tout l'arbre %d\n» (sumTree myTree)
```

Fig.3



Représentation graphique de notre arbre d'exemple.

Tip correspond à la fin d'une branche. La fonction récursive sumTree fait la somme de toutes les valeurs associées aux nœud en parcourant l'arbre récursivement. L'exemple définit un arbre, myTree, qui correspond à l'illustration. [Fig.3].

9 LA PROGRAMMATION OBJET

Il y a encore beaucoup à dire sur les atouts de la programmation fonctionnelle avec F#, mais nous nous intéressons maintenant à la programmation objet, car une des forces de F# est de pouvoir réutiliser les classes écrites avec d'autres langages .NET. En témoigne la capture d'un jeu de la vie écrit en F#. [Fig.4] Le code est disponible à www.fsharp.net. La réciproque est bien entendu vraie. Les autres langages de la plateforme peuvent invoquer du code écrit en F#. Cette opportunité permet d'unir le meilleur de deux mondes. Ainsi on peut imaginer une application telle qu'un compilateur. La partie qui traite les options de compilation pourrait être simplement écrite en C#. Et la partie qui compile le code, donc qui travaille avec des arbres syntaxique serait écrite en F#. Ne faisons plus languir les impatients qui se demandent comment ouvrir une fenêtre avec F#: [Fig.5].

```
open System.Windows.Forms

let myform = new Form()
myform.Text <- «Programmez!»
do Application.Run(myform)
```

Attention. Visual Studio ne permet que de créer des applications de type console pour F#. Pour que ce code compile et fonctionne, vous devrez ajouter manuellement deux assemblies .NET à votre projet: System.Windows.Form et System.Drawing. Nous savons que Text est une propriété d'une Form. F# voit les propriétés comme des valeurs mutables, c'est-à-dire que l'on peut modifier au fil du code. Bien remarquer que l'opérateur d'affectation d'une variable mutable est <- et non = qui est l'opérateur de binding comme nous l'avons dit plus haut. Une des forces de la programmation objet étant l'héritage, nous aurions pu souhaiter spécialiser notre fenêtre en la dérivant de Form. Aucune difficulté :

```
open System.Windows.Forms

type MyForm (title:string)  =
        inherit Form()
        do ( base.Text <- title)

let myform = new MyForm(«Programmez!»)
do Application.Run(myform)
```

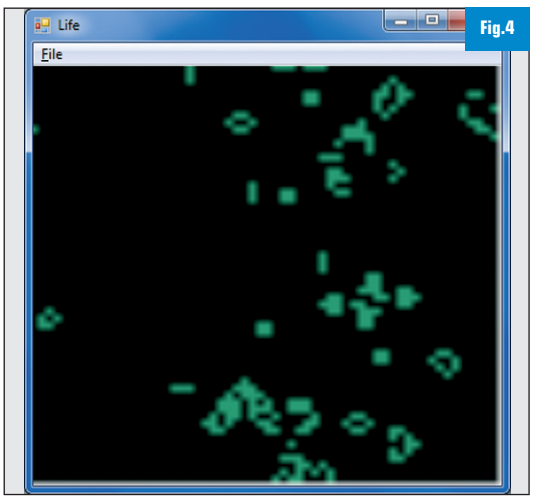
Notre classe dérivée MyForm hérite de Form(). L'écriture inherit Form() va provoquer l'invocation du constructeur de la classe de base. Le corps du constructeur de notre classe suit le mot-clé *do*. Dans ce constructeur nous accédons à une propriété de la classe de base. Tout l'arsenal de la programmation objet se retrouve avec F#. Nous n'allons pas plus loin, car tout l'apprentissage se résume à celui de la syntaxe. Par contre, signalons que F# permet d'invoquer des composants COM ou même d'invoquer une fonction native d'un DLL. Exemple :

```
[<DllImport(«user32.dll»)>]
extern int MessageBox(int32, string, string, uint32)
MessageBox(0, «Programmez!», «Abonnez-vous :-)», 0u) |> ignore
```

On remarquera ici l'opérateur de pipelining |> qui fait tomber la valeur de retour de MessageBox dans la trappe de *ignore*. Enfin, signalons que F# est capable de programmation asynchrone s'il est utilisé avec le framework Async présenté dans Programmez! 140.

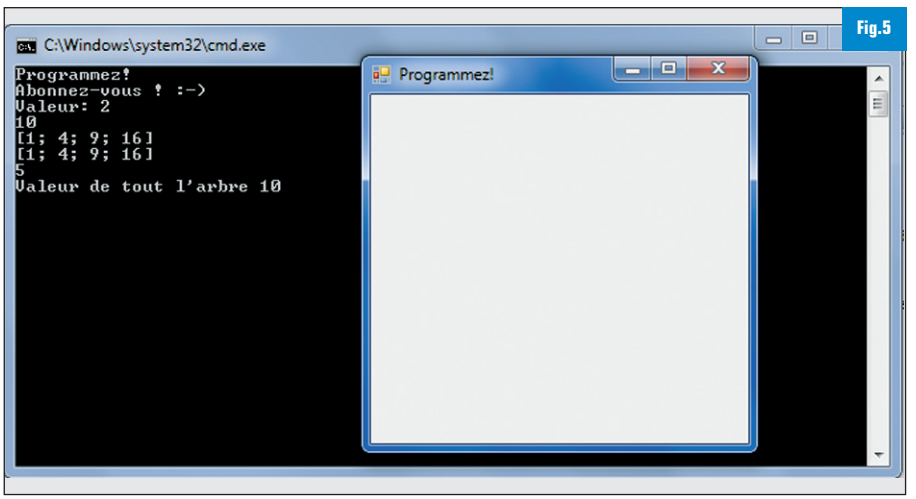
■ Frédéric **Mazué** - *fmazue@programmez.com*

Fig.4



Un jeu de la vie écrit en F#

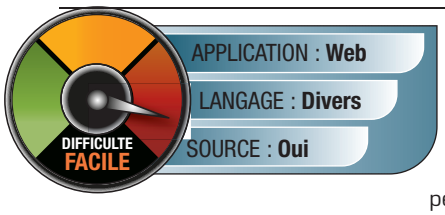
Fig.5



Une simple fenêtre, ouverte avec F#

Aborder le développement d'un site Web avec Webmatrix

Outil intégré et léger, Webmatrix permet de s'initier facilement au développement d'un site Web. Et même un peu plus ! Découverte.



Développer un site Web est une activité passionnante. Mais si on débute, les difficultés rencontrées peuvent s'avérer nombreuses et peut-être décourageantes. En plus du travail sur les pages du site proprement dit, il y a le problème de la configuration du serveur, qu'il s'agisse du serveur matériel (la machine) ou du serveur logiciel. Il a aussi le problème de la configuration du système de gestion des bases de données et la création de la base de données indispensable à tout site un petit peu élaboré et dont les pages sont dynamiques. Enfin, une fois le site fin prêt sur le poste du travail, il y a la question du déploiement, c'est-à-dire la question de son installation sur le serveur machine. C'est ici que se positionne Webmatrix, proposé gratuitement par Microsoft depuis quelque temps. Webmatrix est un environnement de développement intégré (EDI) léger et alternatif à Visual Studio, l'outil classique de Microsoft pour le développement. Webmatrix a pour but de rendre facile l'approche du développement Web au débutant. Il a aussi manifestement pour but, charité bien ordonnée commençant par soi-même, de valoriser certaines technologies Microsoft. Il est néanmoins relativement ouvert, car il permet aussi de travailler avec PHP pour le langage et avec MySQL pour le système de gestion des bases de données.

Léger mais très complet, Webmatrix est un outil réellement précieux pour le débutant. Toutefois son utilisation peut aller bien au-delà. Ainsi sa simplicité et la légèreté pourront être appréciées du professionnel pour la rapide mise en place de petits sites, ou pour la personnalisation de gestionnaires de contenu. On appréciera également les outils d'aide à l'optimisation du référencement. Découvrons maintenant Webmatrix par la pratique. La suite de cet article s'adresse plus particulièrement au débutant. Mais il est bon que celui-ci ait malgré tout quelques notions d'HTML et de C#.

1 INSTALLATION

Webmatrix s'installe très simplement, par exemple en se rendant à cette adresse/ http://www.microsoft.com/web/webmatrix/download/. On vous demandera alors d'autoriser l'exécution d'une application. Cette application est Web Platform Installer (Web PI). Elle se chargera d'installer les composants qui constituent Webmatrix. Hormis Webmatrix proprement dit, les composants sont ASP.NET pour le framework Web, IIS Developer Express pour le serveur logiciel qui sera utilisé lors du développement sur le poste de travail, et SQL Server Compact, une base de données légère, qui a la particularité de conserver toutes les données d'une base dans un seul fichier. A l'issue de l'installation, Web PI reste ouvert [Fig.1] et vous propose l'installation d'autres composants logiciels, ce que vous pouvez ignorer.

2 PREMIERS PAS AVEC PHP

Comme nous l'avons dit plus haut, Webmatrix fait la part belle aux technologies Microsoft. Voyons toutefois rapidement les particularités de travail avec PHP. Au démarrage, Webmatrix ouvre une fenêtre d'accueil vous demandant quel type de projet créer [Fig.2]. Un clic sur 'Mes sites' vous amènera aux sites que vous avez déjà créés. 'Site à partir de la galerie Web' vous propose un choix de sites sous CMS prêts à l'emploi. 'Site à partir du modèle' vous proposera de choisir parmi quelques squelettes de sites. Enfin 'Site à partir du dossier' vous permettra de reprendre d'anciens projet sous Webmatrix. Pour démarrer, choisissez un site WordPress sous 'Site à partir de la galerie Web'. WordPress requiert PHP et MySQL. Webmatrix va donc les installer, ainsi qu'un connecteur .NET pour MySQL puis il installera WordPress. Sachez que si vous utilisez déjà MySQL dans un autre contexte sur votre poste de travail, par exemple avec EasyPHP, celui que Webmatrix va installer entrera en conflit avec le précédent car les deux utilisent le même port. Vous devrez désactiver l'un pour

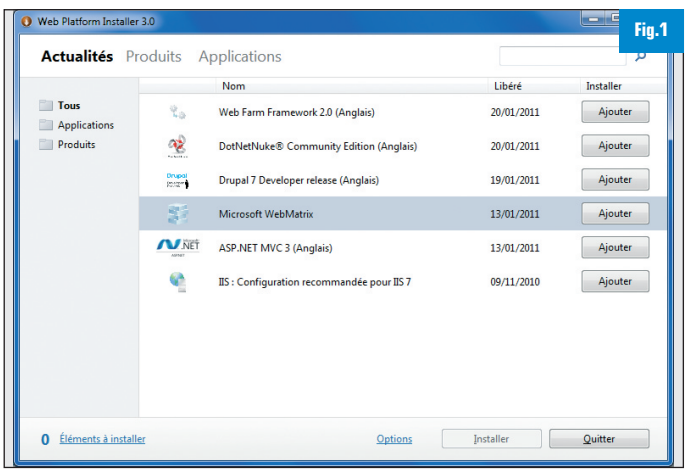


Fig.1

Une fois Webmatrix installé, Web PI vous propose l'installation d'autres logiciels

Fig.2

La page d'accueil de Webmatrix

utiliser l'autre, ou éventuellement reconfigurer l'un des deux, si vous envisagez une cohabitation prolongée :-). Du côté de Webmatrix, MySQL tourne comme un service. Lors de l'installation de WordPress, vous devrez choisir un nom de base de données ainsi qu'un nom d'administrateur accompagné d'un mot de passe. A la suite de quoi Webmatrix créera la base de données, puis vous arriverez dans l'interface de Webmatrix [Fig.3]. Tout est prêt et configuré. Il suffit de cliquer sur le bouton 'Démarrer' pour voir apparaître le site WordPress dans un navigateur. Quel navigateur ? Celui par défaut sur votre poste de travail, mais en faisant apparaître le menu déroulant sous le bouton 'Exécuter', vous pouvez sélectionner un autre navigateur. La colonne de gauche de l'interface utilisateur de WebMatrix mérite toute votre attention. L'onglet 'Site' vous permet d'affiner le comportement de votre site. Vous pouvez ainsi, et par exemple, choisir entre PHP 5.2 ou PHP 5.3, ou encore activer le SSL. L'onglet 'Fichiers' vous permettra d'accéder aux fichiers sources et de les éditer. L'onglet 'Base de données', vous permettra si besoin de créer de nouvelles tables. Enfin l'onglet 'Rapports' vous permettra de générer un rapport quant à la qualité du référencement de vos pages Web. Une fois un site développé, il faut le déployer sur le serveur. Pour cela on cliquera sur le bouton 'Publier' de la barre d'outils. On a alors le choix entre deux protocoles : Web Deploy ou FTP. Le premier est réservé à Windows Server. Dans ce cas tout est automatique, et il vous sera même proposé un choix d'hébergeurs compatibles avec Webmatrix. Si l'on dispose d'un serveur Apache sous Linux, cas extrêmement probable lorsqu'on travaille avec PHP, on devra se contenter du protocole FTP, ce qui n'est déjà pas si mal.

3 HELLOWORLD EN PHP

Voyons maintenant comment créer un projet à partir de zéro. Créez un nouveau site en choisissant un site vide comme modèle. Votre site sera *vraiment* vide, ne contenant qu'un fichier robots.txt, lui-même vide :-) Avant toute chose vérifiez que PHP est bien activé pour le projet, et sinon, activez-le. Pour cela allez sous l'onglet 'Site', plus cliquez sur 'Paramètres' comme illustré [Fig.4]. Il est maintenant temps de créer une première page. Allez d'abord sous l'onglet fichier. Cliquez ensuite avec le bouton droit de la souris sur votre projet puis sélectionnez 'Nouveau fichier' et choisissez un fichier PHP, par exemple baptisé index.php. Et vous obtenez… un fichier HTML :-) Modifiez-le comme ceci :

```
<!DOCTYPE html>
<html lang=»en»>
  <head>
    <meta charset=»utf-8» />
    <title></title>
  </head>
  <body>
    <?php echo «Programmez! Le magazine du développement.»; ?>
  </body>
</html>
```

Cliquez sur le bouton 'Exécuter'. Tout doit fonctionner normalement. Ne cherchez pas comment créer une base de données pour votre site PHP depuis Webmatrix, il ne sait pas le faire. Pour créer votre base de données, vous devrez lancer le ' MySQL Command Line Client' depuis le menu de Windows. Puis créer la base par exemple comme ceci :

```
create database mabase;
```

Ceci fait, revenez dans Webmatrix. Puis sous l'onglet 'Base de données', Cliquez avec le bouton droit sur 'Autres Connexions', et choisissez 'Nouvelle connexion'. Renseignez le dialogue qui va apparaître, par exemple comme illustré ci-dessous. Ceci fait, vous pourrez poursuivre votre travail entièrement depuis Webmatrix, y compris pour créer de nouvelles tables dans la base de données.

4 UN HELLOWORD DÉCOUPÉ AU RAZOR

Abordons maintenant ce qui fait la spécificité de Webmatrix, avec ASP.NET et Razor. On ne peut pas vraiment dire que Razor est un langage. C'est plutôt une syntaxe, qui permet d'insérer avec élégance du code C# ou VB.NET dans des pages Web. Commencez par créer un projet vide. Désactivez PHP, que nous n'utiliserons plus, depuis les paramètres du projet, et pendant que vous y êtes, insérez index.cshtml comme page par défaut possible, comme illustré [Fig.5]. La page par défaut sera celle affichée par le navigateur lorsque vous cliquerez sur le bouton 'Exécuter' et si aucune autre page n'est sélectionnée. Sinon c'est la page sélectionnée qui sera affichée, ce qui est une caractéristique qu'il faut garder à l'esprit si l'on souhaite éviter de s'agacer :-) Créez maintenant une page nommée index.cshtml. Dans le squelette de page, insérez par exemple ceci :

```
<body>
  <p>Programmez!</p>
</body>
```

Puis cliquez sur 'Exécuter'. La page doit s'afficher dans le navigateur. Mais nous sommes pour l'instant en pur HTML. Il est maintenant

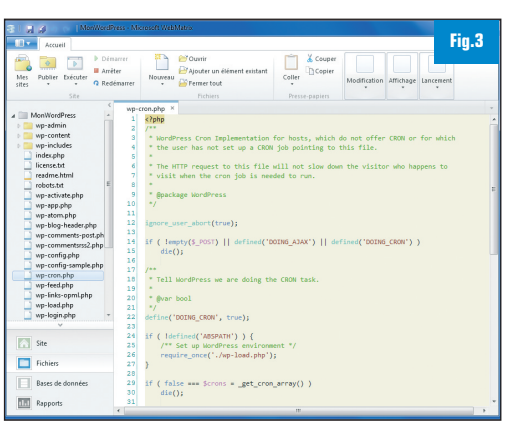


Fig.3

Vue d'ensemble de Webmatrix.

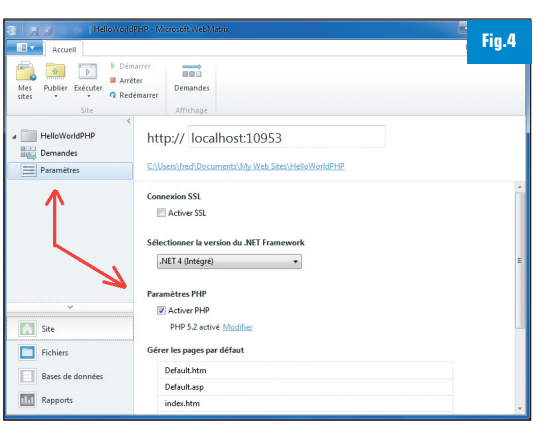


Fig.4

Pensez à activer PHP pour vos projets basés sur ce langage.

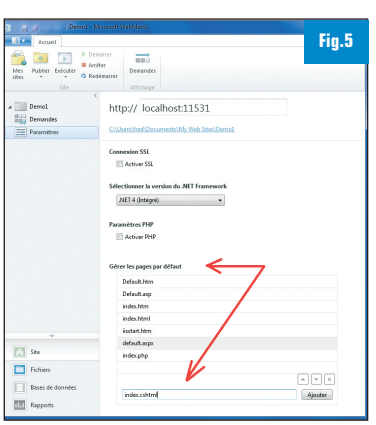


Fig.5

Configuration d'une connexion à une base de données MySQL.

temps d'écrire un peu de Razor. Modifiez le code de la page ainsi :

```
<!DOCTYPE html>
<html lang=»en»>
  <head>
    <meta charset=»utf-8» />
    <title></title>
  </head>
  <body>
    @{
      var msg = «Programmez!»;
      var hint = «Abonnez-vous :-)»;
      msg = msg + « – « + hint;
    }
    <p>@msg</p>
  </body>
</html>
```

Un code plutôt facile à lire, grâce à l'@ à tout faire, qui permet de définir des blocs de code ou de référencer une variable pour que le système insère sa représentation textuelle dans la page HTML. Le code est tout simplement du C# mais cela aurait pu être du VB.NET. Pour cela il suffit de délimiter le bloc un petit peu différemment. Par exemple :

```
@Code
  Dim msg = «Programmez!»
  @+ etc, etc… +@
End Code
```

5 TRAVAILLER AVEC DES OBJETS

Razor est donc aussi simple que cela. Mais une simplicité qui ne va pas sans puissance car il est possible d'utiliser toutes les classes du framework .NET. Ainsi nous pouvons enrichir le code de notre page :

```
@{
  var maintenant = DateTime.Now;
}
<p>La date d'aujourd'hui : @maintenant</p>
```

Ici nous utilisons simplement une propriété de la structure DateTime, mais nous aurions pu, tout aussi bien, dans notre bloc de code, instancier une classe quelconque avec l'opérateur new, puis invoquer ses méthodes. Dans notre exemple, le code qui «calcule» est totalement séparé de l'impression du résultat qui est placé entre les balises <p>. Mais il est très fréquent, dans ce genre de programmation de devoir écrire dans le corps d'une boucle, non seulement des résultats mais aussi des balises HTML, éventuellement non fermantes, pour mettre en forme les résultats. Dans un tel cas, on recourt à l'opérateur @: comme ceci :

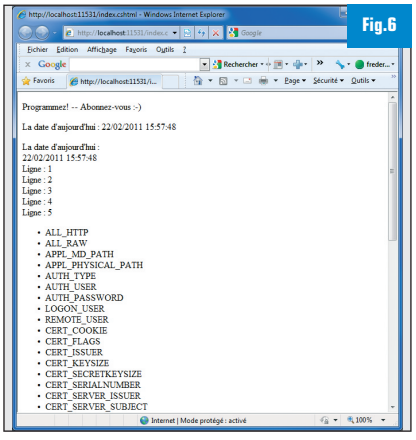


Fig.6

Notre premier site Webmatrix/Razor en action

```
@{
  var encore_maintenant = DateTime.Now;
  @: La date d'aujourd'hui : <br />
  @encore_maintenant
  @: <br />
}
```

On peut aussi préférer dans certains cas, comme du texte multiligne, insérer le texte dans des balises prévues à cet effet :

```
@{
  var encore_maintenant = DateTime.Now;
  <text> La date d'aujourd'hui : <br /></text>
  @encore_maintenant
}
```

6 STRUCTURES DE CONTRÔLES

Comme nous l'avons dit, nous travaillons avec du code C#, donc toutes les structures de contrôles nous sont accessibles pour gérer le flux d'exécution. Test if, boucles, etc. Ecrivons une boucle qui affiche des numéros de ligne. D'après ce que nous avons dit, ceci doit être correct :

```
@{
  for(var i = 1; i < 6; i++)
  {
    @: Ligne : @i <br />
  }
}
```

Et ça l'est effectivement, mais tout en étant un peu lourd. Razor propose alors un sucre syntaxique qui dispense de la déclaration du bloc de code :

```
@for(var i = 1; i < 6; i++)
{
  @: Ligne : @i <br />
}
```

Bien remarquer, le double usage de @. D'abord @: annonce une ligne de texte qui contiendra une balise à restituer brute. Puis le deuxième @ référence une variable pour insertion dans la ligne de texte. Maintenant, pour le plaisir, une petite boucle foreach, pour afficher toutes les variables internes du serveur : [Fig.6].

```
<ul>
@foreach (var item in Request.ServerVariables)
{
  <li>@item</li>
}
</ul>
```

7 HELPERS

Un des points forts de Webmatrix se situe dans les Helpers. Il s'agit de composants logiciels réutilisables. Un certain nombre est à notre

disposition, qu'il suffit d'installer, et bien entendu, il est possible de coder ses propres helpers. Beaucoup de ces Helpers sont vraiment sympathiques et facilitent le travail, par exemple avec Facebook, Paypal, ou Twitter. Les actualités du site de Programmez! étant relayées dans Twitter (êtes-vous abonné ? :), nous allons nous faire la main avec le helper Twitter. Pour cela, commencez par créer un site, comme au paragraphe 4. Créez votre page par défaut et lancez le site une première fois. Cliquez maintenant sur l'onglet 'Site', et dans la zone d'édition, vous verrez un lien 'Administration d'ASP.NEt Web Pages' [Fig.7]. Cliquer sur le lien vous amènera à l'interface d'administration de votre site. Mais vous devrez d'abord choisir un mot de passe puis le valider, selon une procédure qui vous sera expliquée. On vous demandera aussi de renommer un fichier que vous ne trouverez pas… Le truc consiste à cliquer sur l'onglet 'Fichiers', puis avec le bouton droit sur le répertoire de votre site, pour enfin sélectionner 'Actualiser'. Le fichier apparaît alors… :-) Quand tout sera en ordre, votre navigateur vous amènera sur une page qui vous propose l'installation de différents helpers. Choisissez celui de ASP.NET Web Helpers Library 1.0 et non celui de Twitter 1.0. qui apporte des Widgets Twitter [Fig.8]. Si vous avez l'esprit curieux, rafraîchissez à nouveau l'affichage du répertoire de votre site, pour voir ce qui a été installé et notez la librairie .dll dans le répertoire bin. Utilisons maintenant notre Helper. Le code est adapté du tutorial de Microsoft :

```
<h1>Twitter Feed de Programmez</h1>
<form action=»» method=»POST»>
<div>
 Donnez le nom du flux Twitter à afficher: 
 <input type=»text» name=»TwitterUser» value=»progmag»/> 
 <input type=»submit» value=»Submit» />
</div>
<div>
 @if (Request[«TwitterUser»].IsEmpty()) {
   @Twitter.Search(«progmag»)
 }
 else {
   @Twitter.Profile(Request[«TwitterUser»])7
 }
</div>
```

Le résultat est plutôt sympathique :-) [Fig.9].

8 ECRIRE SES PROPRES HELPERS

C'est un travail plutôt facile. Nous l'avons vu précédemment, un helper est une librairie déposée dans le répertoire bin du site. Il suffit de créer cette librairie. On le fera avec Visual Studio. Une version Express suffira amplement. Ou sinon, on pourra le faire avec Mono. Si vous travaillez avec Visual Studio, créez un projet de type Class Library, puis compilez par exemple ce code :

```
using System;

namespace HelperProgrammez
{
 public class Hint
 {
   public static String GetMsg()
   {
     return «Programmez! – Abonnez-vous :-)»;
   }
 }
}
```

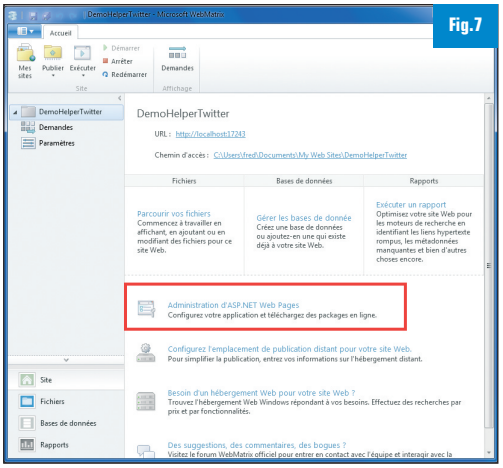
Récupérez ensuite le fichier HelperProgrammez.dll sous le répertoire bin\Debug de votre projet Visual Studio et déposez-le dans le répertoire bin de votre site. C'est tout. Vous pouvez utiliser le helper dans une page :

```
<p>@HelperProgrammez.Hint.GetMsg()</p>
```

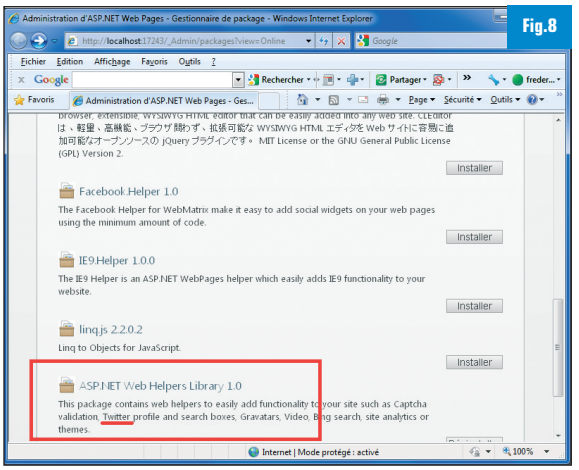
9 CONCLUSION

En dépit de petites lacunes, comme par exemple des assistants à l'écriture du code un peu trop rudimentaires, Webmatrix est un outil très sympathique, excellent pour se former au développement Web, ou pour mettre en place très rapidement de petits sites, ou encore pour customiser un site sous gestionnaire de contenu. Le lecteur qui voudra aller plus loin avec cet outil trouvera de nombreux tutoriels sur http://msdn.microsoft.com/fr-fr/asp.net, pour toutes les situations : travail avec les bases de données, avec les formulaires, avec le débogueur, etc.

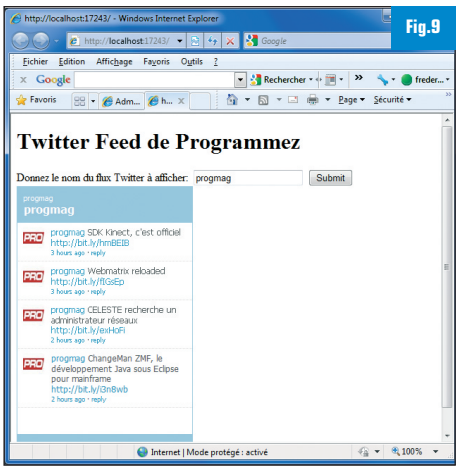
■ Frédéric **Mazué** - *fmazue@programmez.com*



Webmatrix vous permet d'accéder à l'interface d'administration de votre site.



Installation du Helper pour Twitter.



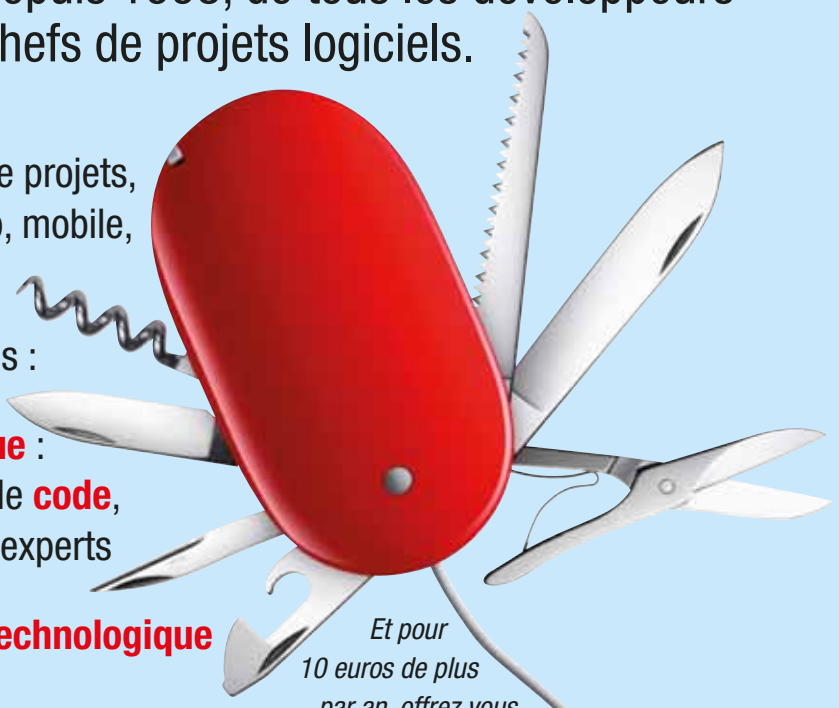
Le flux Twitter de Programmez! est désormais inclus dans votre nouveau site.

TOUT SAVOIR POUR TOUT FAIRE

Développez votre savoir-faire !

Programmez ! est le magazine de référence,
depuis 1998, de tous les développeurs
et chefs de projets logiciels.

Code, gestion de projets,
développement web, mobile,



Programmez ! est à la fois :

- votre **outil pratique** :
  articles de **code**,
  par les meilleurs experts
- votre **veille technologique**

Et pour 10 euros de plus par an, offrez vous l'accès illimité à toutes les archives et numéros en format électronique.

Abonnez-vous

à partir de 4€ seulement par mois

ABONNEZ-VOUS EN LIGNE

www.programmez.com

Toutes les offres sont en ligne

☐ OUI, je m'abonne

(à retourner, avec votre règlement à :
Groupe GLI, 22 rue rené Boulanger, 75472 Paris cedex 10)

☐ *Abonnement 1 an : 49€ 11 numéros par an au lieu de 65,45€, prix au numéro (*)*

☐ *Abonnement 2 ans au magazine : 79€ (*)*

☐ *Abonnement intégral :1 an au magazine + archives : 59€ (*)*

(Tarif France métropolitaine)*

☐ M. ☐ Mme ☐ Mlle Société .....

Titre : ..... Fonction : ..... Adresse mail .....

NOM ..... Prénom .....

N° ..... rue .....

Complément .....

Code postal : |_|_|_|_|_| Ville .....

☐ *Je joins mon réglement par chèque à l'ordre de PROGRAMMEZ* ☐ *Je souhaite régler à réception de facture* (écrire en lettres capitales)

GPU

Cuda par l'exemple

*Difficulté : *****
Editeur : Pearson
Auteur : collectif
Prix : 33 €

La technologie Cuda permet de développer en parallèle sur des GPU, les puces graphiques. Cette architecture facilite l'utilisation de cette source de puissance encore trop rarement utilisée. *Cuda par l'exemple* va vous expliquer comment utiliser, mettre en œuvre Cuda dans vos codes C. Il s'agit tout d'abord de découvrir sa structure, son fonctionnement et les adaptations du langage C. Puis les auteurs attaquent dans le dur : thread, données, textures, interopérabilité dans les traitements graphiques, atomicité, etc. L'explication est très complexe avec des exemples précis en C. Niveau intermédiaire et expert. Prend en compte Cuda 4.0.

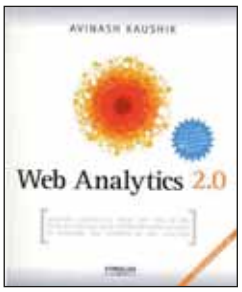
WEB

Ajax, jQuery et PHP

3e édition

*Difficulté : *****
Editeur : Eyrolles
Auteur : Jean-Marie Delfrance
Prix : 39,90 €

Trio infernal pour les uns, magique pour les autres, depuis plusieurs années, ils sont les fondations du web actuel et de tout site digne de ce nom. L'auteur a voulu miser sur trois mots : atelier, atelier et atelier. 42 ateliers sont disponibles dans ce livre où la pratique est cruciale pour comprendre, maîtriser. Tous les domaines sont abordés avec à chaque fois une explication, du code, la mise en pratique. Un must pour apprendre et maîtriser !



WEB

Web Analytics

*Difficulté : ****
Editeur : Eyrolles
Auteur : Avinash Kaushik
Prix : 38€

Aujourd'hui, un site web doit être suivi, analysé par les bons outils. Le web analytics 2.0 fait partie de la panoplie du super-webmaster. Quel est l'impact et la performance de mon site ? Comment optimiser l'activité du site et de mon réseau social ? Comment répondre aux attentes des visiteurs ? Qui vient et quand ? L'auteur décrit ici des stratégies bien particulières pour sublimer les outils de mesure de trafic Web en y incorporant des données qualitatives, des tests et de l'intelligence concurrentielle. Il y explique comment mesurer, analyser et agir en tenant compte de l'évolution rapide des technologies et des tendances du Web - y compris les réseaux sociaux, la vidéo, la mesure des mobiles et du comportement des internautes. Les possibilités sont nombreuses et il faut les bons outils. Découvrez dans cet ouvrage, toutes les techniques à connaître et à mettre en place.

LIVRE DU MOIS

Drupal 7

*Difficulté : *** - Editeur : éditions Eni*
Auteur : Christophe Aubry - Prix : 26,50 €

Vedette incontestable du monde des CMS, Drupal n'en finit plus de s'imposer et d'étendre son impressionnante capacité fonctionnelle. La v7 permet à l'outil de passer un nouveau cap. Cet ouvrage va vous apprendre à installer, configurer et utiliser l'interface d'administration de vos sites web.
Drupal permet une administration très précise des droits utilisateurs, qui sont amplement détaillés par l'auteur. Une des grandes forces de Drupal est la possibilité de personnaliser et de créer des types de contenu qui vont répondre à tous vos besoins en matière d'organisation et d'affichage de l'information. L'administration des commentaires est tout aussi personnalisable et efficace. De plus, Drupal permet l'ajout de fonctionnalités supplémentaires par l'intermédiaire de modules puissants et adaptables à vos besoins. Enfin, le design de vos sites web sera confié à des thèmes qui afficheront vos sites avec la structure, l'organisation et la navigation que vous aurez définies. Livre de référence.

CODE

Parallel Programming with MS Visual C++

*Difficulté : *****
Editeur : Microsoft
Auteur : collectif
Prix : 29,99 $

Voici sans doute l'ouvrage de référence pour développer en parallèle avec Visual C++. Les auteurs abordent les techniques essentielles : boucle, tâche, tâche dynamique. Les appendices sont encore plus intéressants, notamment sur les techniques de debug parallèle ou comment le task scheduler travaille. Incontournable pour le développeur C++ ! En Anglais.

MÉTHODE

CMMi 1.3

*Difficulté : ***
Editeur : Dunod
Auteur : Richard Basque
Prix : 39 €

Le CMMI, dont la version 1.3 a été publiée à l'automne 2010, est de plus en plus largement utilisé pour évaluer la maturité de processus, pour développer des plans d'amélioration ou pour mettre en œuvre des pratiques plus matures. Ce livre est un guide commenté du pourquoi et du comment de ce modèle de bonnes pratiques. Il a surtout pour vocation d'aider concrètement les organisations qui veulent réussir leurs projets, dans les délais, dans les budgets et à la satisfaction de leurs clients. Une excellente introduction pour ceux qui utilisent ou s'intéressent à CMMi.

PROCHAIN NUMÉRO

N°144 septembre 2011
parution 1er septembre

Créer de A à Z son site e-Commerce sa boutique en ligne

MySQL : faut-il encore l'utiliser ?

Netbeans 7.0 : le meilleur IDE pour Java

Les outils des Décideurs Informatiques

Vous avez besoin d'info sur des sujets d'administration, de sécurité, de progiciel, de projets ? Accédez directement à l'information ciblée.

Cas clients

Actu triée par secteur *Avis d'Experts*

Actus *Evénements* *Newsletter* *Vidéos*

www.solutions-logiciels.com

☐ **OUI**, *je m'abonne* (écrire en lettres capitales)

Envoyer par la poste à : Solutions Logiciels, service Diffusion, 22 rue rené Boulanger, 75472 PARIS - ou par fax : 01 55 56 70 20

1 an : 50€ *au lieu de 60€, prix au numéro* *(Tarif France métropolitaine)* **- Autres destinations :** *CEE et Suisse : 60€ - Algérie, Maroc, Tunisie : 65€ , Canada : 80€ - Dom : 75€ Tom : 100€*

10 numéros par an.

☐ M. ☐ Mme ☐ Mlle Société .....

Titre : ..... Fonction : ☐ Directeur informatique ☐ Responsable informatique ☐ Chef de projet ☐ Admin ☐ Autre .....

NOM ..... Prénom .....

N° ..... rue .....

Complément .....

Code postal : |_|_|_|_|_| Ville .....

Adresse mail .....

☐ *Je joins mon réglement par chèque à l'ordre de SOLUTIONS LOGICIELS* ☐ *Je souhaite régler à réception de facture*

Editeur : K-Now sarl au cap de 8000€ siège social : 21 rue de Fécamp, 75012 Paris

2 C'EST MIEUX QU' 1

JUSQU'AU (18 JUILLET)

commandez vite !

OPÉRATION POUR 1 EURO DE PLUS

Aucun abonnement à souscrire

RECEVEZ NON PAS 1 MAIS 2 SUPERBES MATÉRIELS

CHOISISSEZ

2 SUPERBES SMARTPHONES
Samsung GALAXY S II

ou

2 SUPERBES TABLETTES 10,1 pouces
Samsung GALAXY Tab 10.1

Vous pouvez mixer les matériels

ou

2 SUPERBES ORDINATEURS PORTABLES DELL INSPIRON DUO (PC / TABLETTE)

COMMANDEZ WINDEV®

*ou **WINDEV Mobile 16** ou **WEBDEV 16***

CHEZ PC SOFT

ET RECEVEZ NON PAS 1 MAIS 2 SUPERBES MATÉRIELS POUR « 1 EURO DE PLUS »

A propos de WINDEV 16

Environnement de développement professionnel totalement intégré (AGL, IDE, ALM), **WINDEV 16** est réputé pour sa **puissance** et sa **facilité** d'utilisation. **WINDEV 16** est livré complet et gère le **cycle complet de développement**. **WINDEV 16** permet d'utiliser toutes les bases de données.

Les applications créées fonctionnent avec toutes les versions de Windows: 2000, NT, 2003, XP, Vista, 7, sous TSE et Citrix, sur Netbook et Tablette, ainsi que sous Linux, Mac et sur Mobile et Internet...

Le code est **multiplateformes**: Windows, Linux, .Net, Internet, Java, PHP, J2EE, Android, Windows Mobile, Phone 7...

Fournisseur Officiel de la Préparation Olympique

www.pcsoft.fr

Tél province : 04.67.032.032 Tél Paris : 01.48.01.48.88

Détails de l'opération

Cette offre est valable, pendant la période de l'opération, sur les références : WD16EE, WB16EE, WM16EE.

Offre valable pour la France Métropolitaine uniquement. Cette offre est valable uniquement sur les tarifs «catalogue». Aucune remise ne sera appliquée sur ce tarif.

Attention: cette offre n'est pas valable sur les mises à jour, achats groupés, offres spéciales et échanges concurrentiels.

Cette offre est applicable pour toute commande reçue à PC SOFT entre le 8 Juin 2011 et le **18 Juillet 2011** inclus. La demande du matériel devra être parvenue à PC SOFT Montpellier avant le 15/9/2011.

Logiciel et matériel peuvent être acquis séparément.

Tous les détails sont sur **www.pcsoft.fr** ou appelez-nous !