par Michael Tschater
<tschater/at/web.de>

L´auteur:

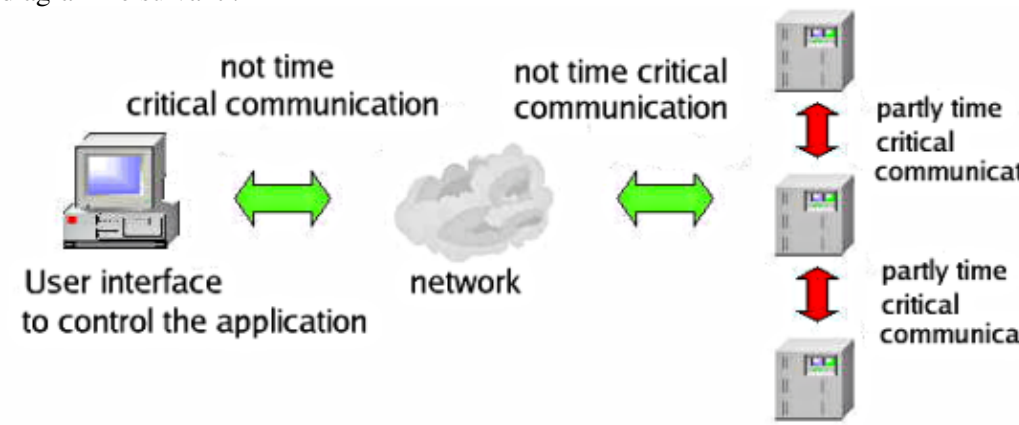
Michael est principalement occupé par des développements logiciels relatifs au matériel (firmware). Pour son projet actuel, une décision supplémentaire concernant l'environnement de développement – pour être utilisé pour la programmation de « l'interface » de son firmware – a dû être prise.

Traduit en Français par:
Florent Morel <fleuh–(at)–free.fr>

Logiciel de développement indépendant de la plate–forme

Résumé:

Pratiquement tous les équipements utilisés dans l'industrie peuvent être controllés à tra
un réseau. L'interface utilisateur tourne indépendemment du matériel et agit comme un
simple client, recevant et envoyant des données non critiques (i.e. paramètres
d'initialisation et résultats de mesures ou tests). Ceci est représenté en vert dans le
diagramme suivant :



Les projets logiciels requièrent souvent une réponse à la question : « Quel système d'exploitation doit être suppporté ? ». Tandis que les lecteurs de ce magazine préfèrent Linux, il est nécessaire de supporter d'autres systèmes (principalement Windows). En principe, la question du système ne doit pas représenter le problème dominant d'une application ; l'utilisateur doit être capable d'arriver à des résultats de manière intuitive. L'article suivant devrait démontrer que le choix d'une plate–forme spécifique n'est pas requis puisqu'il est possible d'écrire des programmes qui peuvent être compilés pour différents systèmes d'exploitation. Cet article sera limité aux PC avec Linux et Windov Il devrait aussi être possible d'utiliser les applications sur Mac et MacOSX mais ceci ne peut être démontré faute de matériel.

Avec les bibliothèques indépendantes des plate−formes, il est nécessaire de faire la différence entre deux approches pour ce qui est des boîtes de dialogues ou des menus :

1. Bibliothèques natives : pour afficher les éléments de l'application, les routines correspondantes du système d'exploitation sont utilisées. Cela assure que tous les contrôles apparaissent de la même manière que le reste des applications du système. Une bibliothèque native gère différemment les contrôles sous Linux et sous Windows 2000 ou XP.
2. La seconde possibilité est de programmer un comportement et un look particulier, ainsi tous les contrôles seront affichés de la même façon sous tous les systèmes d'exploitation.

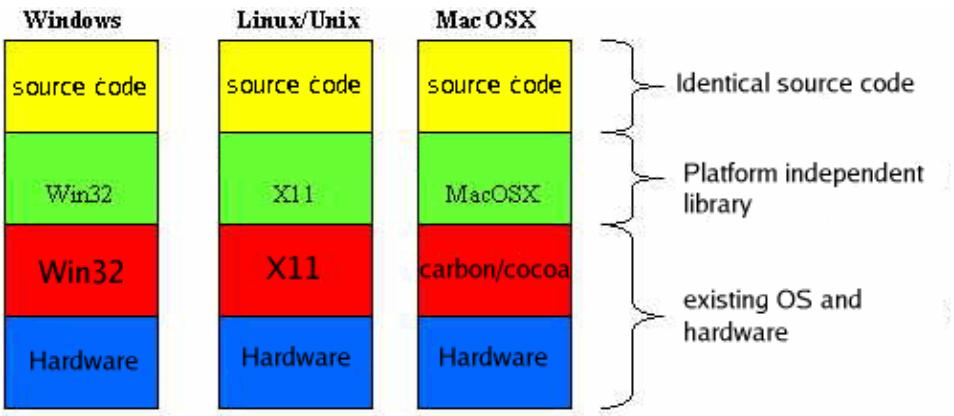
En plus des caractéristiques techniques des bibliothèques, d'autres facteurs opérationnels jouent un rôle qui doit être comparé :

- Environnement de développement : un environnement de développement intégré (ex : GUI builder, générateur de makefile) simplifie le travail du développeur.
- Documentation et support : aide immédiate en cas de problèmes.
- Coûts : alors que la plupart des bibliothèques sont disponibles gratuitement pour un usage privé, l'utilisation des bibliothèques dans les applications commerciales entraînera parfois un coût supplémentaire. Ces coûts devront être justifiés à sa hiérarchie lors de projets commerciaux.
- Coût réel pour effectuer le portage vers d'autres systèmes.

Dans certains cas, il est important de prendre encore une petite chose en compte, cependant cela ne s'applique pas à tous les projets :

- Le logiciel créé doit utiliser les contrôles natifs pour s'intégrer sans faille dans l'architecture existante. L'utilisateur ne doit pas être en mesure d'identifier les différences entre le nouveau logiciel et un existant sur son système.

En affichant les bibliothèque dans un modèle de couches, on obtient ceci :



Langages de Programmation

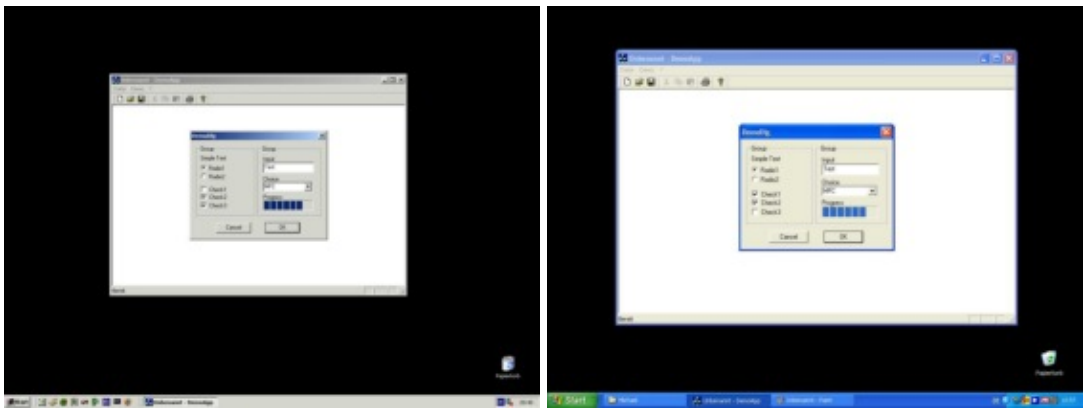
Le premier critère à choisir est le langage de programmation. Il y a plusieurs choix, qui vont être discutés plus bas :

1. bibliothèques C/C++
2. Java
3. Kylix
4. Smalltalk
5. Mozilla

Les alternatives à C et C++ seront expliquées plus en détail car elles sont moins prisées des développeurs.

Exemple d'application

Pour pouvoir comparer les différents paquets logiciels, un exemple d'application utilisant toutes les bibliothèques doit être généré. L'implémentation de l'application n'a aucune fonctionnalité mais elle montre les principaux contrôles. Pour les fenêtres, le programme créé sera du pur Windows (Visual C++ 6.0, bibliothèques MFC−Class), les autres paquets lui seront comparés. Nous comparerons le « look & feel » (design et comportement des contrôles). Les distributions Linux utilisées seront RedHat Fedora Core 2 et Debian 3.0.



Captures d'écran Windows 2000 et Windows XP (code source pour Visual C++ ici (win32_src.zip)).

Bibliothèques C/C++

Trolltech Qt

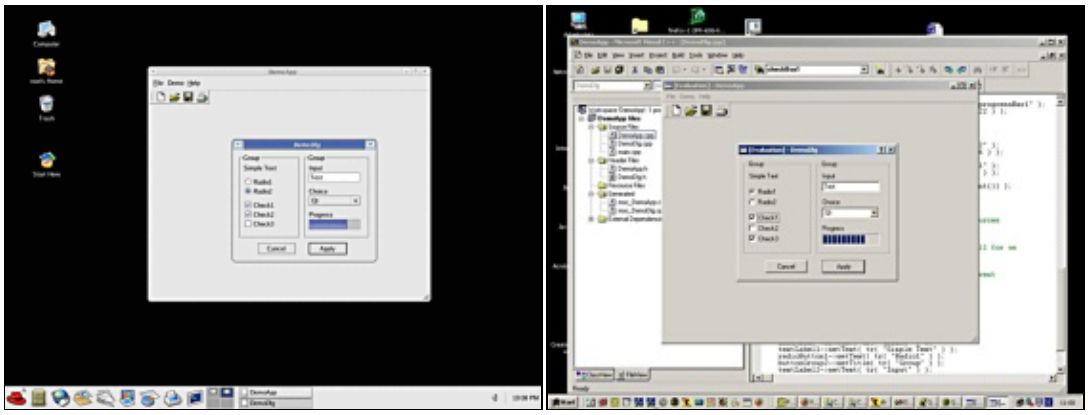
Qt est une bibliothèque de classes de la compagnie Norvégienne Trolltech pour créer des applications sur plate−forme indépendantes avec C++. KDE, un des gestionnaires de fenêtre de Linux, est basé sur le paquet Qt. À l'origine, Qt était sous une licence qui n'était pas tolérable de la part de beaucoup d'utilisateurs Linux. Pour cette raison, la bibiothèque GTK+ a été développée. C'est cette bibliothèque que l'on retrouve dans le gestionnaire de fenêtre Gnome. Pendant ce temps, la version Linux, tout comme la version MacOS, fut disponible sous licence GPL, incluant tout le code source. Qt pour Windows est toujours disponible dans le commerce. Une version d'évaluation doit pouvoir être téléchargée depuis leur page web − une distinction de licence est effectuée selon que l'utilisation est commerciale ou académique. C'est la version commerciale d'évaluation dont il sera question par la suite. Cette version requiert de s'enregistrer auprès de Trolltech.

En plus des versions pour Windows, Linux (Unix) et Mac, une version embarquée est disponible. Elle tourne sur différents Linux embarqués et offre une fenêtre d'administration plus épurée.

L'installation sous Linux se déroule comme prévu, sans accroc. Le générateur d'IHM (interface utilisateur) Qt Designer est inclus, tout comme la documentation détaillée, des exemples d'application, un guide de démarrage rapide (Quick Start Guide) et une revue des classes. Qt Designer génère une description XML de l'IHM. En utilisant l'outil QT−Tool qmake, vous pouvez générer un Makefile valide à partir de la description XML. Ce Makefile génère alors le code source C++ à partir de la description de l'IHM (Qt−Tool : uic) et appelle le compilateur Méta−Objet (Meta Object Compiler) (Qt−Tool : moc). Celui−ci traduit les extensions (en langage Qt) en code source C++. Ensuite, une procédure standard de make pour compiler l'exécutable peut être opérée.

La séquence suivante est nécessaire pour générer les fichiers souces manuellement (le fichier en entrée est MyDialog.ui) :

- uic MyDialog.ui > MyDialog.h
- uic −impl MyDialog.h MyDialog.ui > MyDialog.cpp
- moc −o moc_MyDialog.cpp MyDialog.h



Captures d'écran Linux et Windows 2000 (code source pour QtDesigner ici (qt_src.tar.gz)

Résumé sur Qt

Nom :	Trolltech Qt
Version :	3.3.2
Systèmes d'exploitation :	Linux, Win32, MacOS, Solaris, IRIX, AIX, HP−UX
Langage :	C++
Licence :	GPL ou licence propriétaire (commerciale)
Avantages :	• bibliothèque de base de KDE (Windows Manager sous Linux) • installation des paquets dans toutes les distributions standard (installation très simple)

	• contrôles génériques sous Windows • superbes environnements de développement • approuvée • support des migrations pour les applications Win32 MFC permettant une conversion incrémentale du code source MFC.
Inconvénients :	• possibles coûts de licence (montant élevé) • le logiciel d'évaluation produit des erreurs lors de l'installation sous windows
Environnements de développement :	e.g. QtDesigner, KDevelop
WWW :	http://www.trolltech.com
Documentation :	manuels, tutoriels, listes de diffusion e.g. http://doc.trolltech.com/3.3/index.html
Projets références :	• KDE Desktop (Defaut e.g. avec SuSE) • Navigateur Opera • Album Photoshop
Distribution :	très diffusé

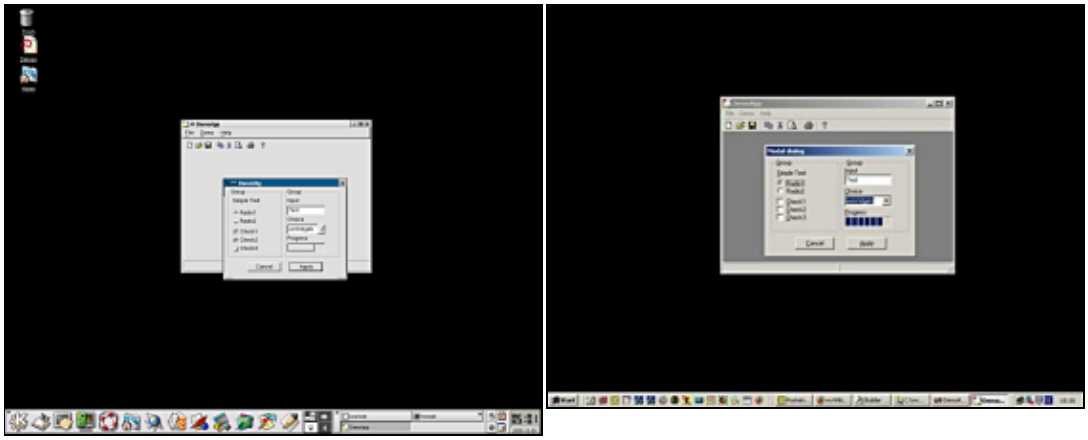
wxWidgets

L'outil wxWidgets est disponible depuis quelques 12 années, mais il n'obtint son nom actuel qu'il y a quelques mois. Le nom wxWindows, utilisé jusque là, fut abandonné après quelques « discussions » avec Microsoft. wxWidgets inclut une impressionante collection de classes pour tous les domaines d'application. La liste des applications références démontre la maturité du paquet logiciel.

Le programme est en C++, il est similaire à Visual C++ sous Windows.

Un inconvénient avec wxWindows2.4.2 sous RedHat Fedora Core 2 est que vous rencontrez des erreurs en compilant les programmes d'exemple. La cause est que des appels GTK+ sont déclarés private dans les versions GTK+ patchées par RedHat. L'appel à ces fonctions n'est donc pas permis. Cependant ce sont des problèmes mineurs. Tout fonctionne normalement quand la bibliothèque standard GTK+ est utilisée. Sous Debian, tout roule.

L'installation sous Windows fonctionne sans problème.



Captures d'écran Linux et Windows 2000 (code source pour wxWidgets ici (wx_src.zip)).

Overview of wxWidgets

Nom :	wxWidgets
Version :	2.4.2
Systèmes d'exploitation :	Linux, Win32, systèmes embarqués
Langage :	C++
Licence :	LGPL
Avantages :	• prise en main aisée (beaucoup d'exemples). • très bonne documentation.
Inconvénients :	• Problèmes avec la combinaison : Fedora Core 2 – wxWindows2.4.2
Environnements de développement :	
WWW :	http://www.wxwidgets.org
Documentation :	manuels, tutoriels, listes de diffusion, wiki e.g. http://wiki.wxwidgets.org
Projets références :	AOL Communicator
Distribution :	pas très diffusé

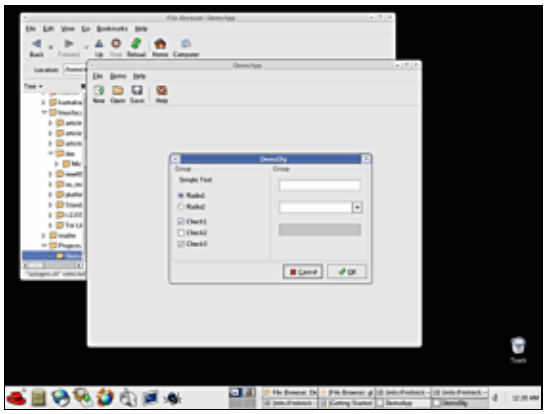
GTK+ (avec gtkmm)

L'acronyme signifie « The Gimp Toolkit » (la trousse à outils de The Gimp). Deux célèbres projets sont le Windows Manager Gnome – inclus dans beaucoup de distributions Linux – et l'application graphique The Gimp. Gnome est l'un des deux principaux environnements de bureau – avec KDE (voir Qt) – sous Linux.

C'est l'environnement par défaut dans nombre de distributions. Avec la sortie de GTK+ version 2, le « look & feel » a été sensiblement amélioré.

Une particularité de GTK+ est sa complète implémentation en C. En conséquence, le générateur d'IHM *glade2* produit du code C. En utilisant *gtkmm* (anciennement GTK−−), le code peut aussi être généré en C++.

Contrairement à l'apparence très « professionelle » de GTK+ pour Linux, 'GTK+ for Win32' n'est pas impressionant. Cliquer sur le lien de la page principale de GTK+− affiche immédiatement l'avertissement suivant : **« The program(s) might crash unexpectedly or behave otherwise strangely » (Le programme peut crasher de façon imprévue et agir de façon étrange).** (Mais bien sûr, beaucoup d'autres programmes commerciaux tournant sous Windows agissent de même). La stabilité semble dépendre fortement de la machine, des pilotes d'affichage, des autres programmes installés ou non présents (statut du 06 Septembre 2004). Les développeurs courageux cliqueront sur la page de téléchargement quand même. Pour un paquetage facile à prendre en main, veuillez repasser. Au lieu de cela, le développeur pourra lire des instructions sur comment installer nombres de composants logiciels, et comment revenir sur la page de téléchargement si certains composants spécifiques sont manquants. Cela correspond parfaitement au message de la page d'accueil de GTK+ pour Windows : « You are expected to be quite experienced to be able to use GTK+ in your own programs. This isn't Visual Basic. » (Il est nécessaire d'être expérimenté pour pouvoir utiliser GTK+ dans ses propres programmes. Ceci n'est pas Visual Basic). Après avoir installé les composants initiaux et une tentative ratée de commencer une des applications d'exemple, la plupart des développeurs peuvent avoir perdu leur envie d'aller voir plus loin avec cet outil. GTK+ pour Win32 pêche donc par une présentation et une prise en main pas très professionelle, ce qui ne le destine donc pas à ce type d'applications.



Capture d'écran de GTK+ pour Linux (code source pour glade2 ici (gtk_src.tar.gz)

Résumé sur GTK+

Nom :	GTK+ – The GIMP Toolkit
Systèmes d'exploitation :	Linux, Win32
Langages :	C (C++ avec gtkmm)
Licence :	LGPL
Avantages :	• bibliothèque de base de Gnome (gestionnaire de fenêtres sous Linux)

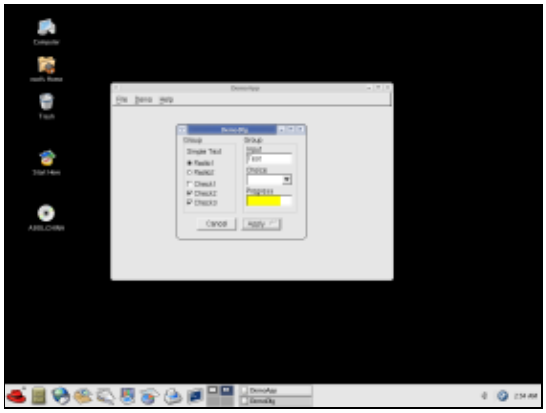
	• paquet d'installation inclus dans toutes les distributions standard (installation très simple) • contrôles génériques sous Windows • approuvé (sous Linux)
Inconvénients :	• L'implémentation Win32 est instable (statut du 09−2004)
Environnements de développement :	e.g. *2glade* (GUI Builder), Anjuta
WWW :	http://www.gtk.org
Documentation :	manuels, tutoriels, listes de diffusion e.g. http://developer.gnome.org/doc/API/2.0/gtk/index.html
Projets références :	• Gnome Desktop • The GIMP • Gnumeric
Distribution :	Linux: très bien diffusé, Windows: distribution marginale

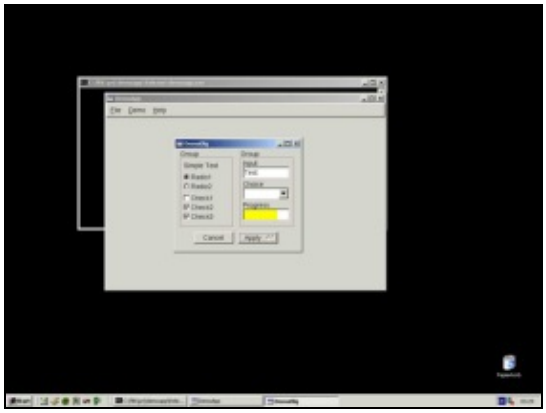
FLTK

Le Toolkit FLTK (pour Fast, Light Tool Kit) est un paquet trop méconnu. Il fut implémenté comme le successeur de XForms. Les sources complètes sont téléchargeables depuis le site du projet. La taile, 2.3Mo (Linux) ou 3Mo (Windows) prouve son nom. Installation sous Linux sans accroc : on dépacke, on lance '*make*', et c'est fini ! L'utilisateur a alors à sa disposition bibliothèques, applications d'exemples, le GUI builder « *fluid* » et un handbook de programmation. Évidemment, le nombre de classes mis à disposition est plus réduit que ceux de « poids−lourds » comme Qt ou wxWindows. Les classes incluses couvrent tout le domaine GUI, c'est−à−dire : fenêtres, menus, contrôles, OpenGL et affichage d'images. Les classes pour des communications réseau et autres ne sont pas incluses.

L'installation sous Windows était plus compliquée. Sous l'environnement de développement Visual C++, seul le projet principal nécessitait un portage. Cela génère cependant des problèmes avec les bibliothèques graphiques. Une solution simple est de supprimer les commentaires dans le fichier de configuration config.h. Une seconde spécificité de Windows est que la version DEBUG de la bibliothèque FLTK ouvre toujours une fenêtre DOS additionnelle. Cela assure, aux programmes démarrés depuis la ligne de commande, la possibilité d'écrire sur stderr et stdout.

D'un point de vue général, le Toolkit FLTK laisse l'impression d'être bien pensé. La documentation augmente la petite taille de l'exécutable (80ko pour un « hello world ») ainsi que des composants graphiques 2D et 3D (OpenGL) légers et rapides. De plus, la bonne portabilité doit être soulignée.





Captures d'écran de FLTK sous Linux et Windows 2000 (code source pour FLTK ici (fltk_src.tar.gz))

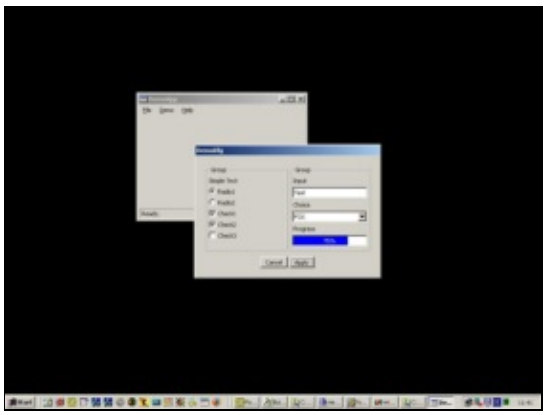
Résumé sur FLTK

Nom :	Fast Light Tool Kit
Version :	1.1.5rc2
Systèmes d'exploitation :	Linux, Win32, MacOS
Langage :	C++
Licence :	LGPL
Avantages :	• bibliothèque très légère • Code source incluant la documentation et l'environnement de développement « fluid ». • bon support OpenGL (non testé) • contrôles génériques sous Windows
Inconvénients :	• Installation sous Win32 (Visual C++) non sans problèmes • L'environnement de développement « fluid » n'est pas stable sous Windows.
Environnements de développement :	e.g. fluid (GUI Builder)
WWW :	http://www.fltk.org, Download : http://freshmeat.net/projects/fltk/
Documentation :	Manuels, Tutoriels, Listes de diffusion e.g. http://
Projets références :	• http://vtkfltk.sourceforge.net/
Distribution :	Petite distribution, très méconnu, même dans la communauté de développeurs.

Le Toolkit FOX

Le Toolkit FOX prétend être le plus rapide des toolkits disponibles. Il offre un large éventail d'éléments GUI et une interface OpenGL.

Les installations sous Linux et Windows se sont déroulées sans problème. Une documentation détaillée ainsi que des exemples de projets sont fournis. Un descriptif des classes n'est pas inclus dans la version présentée mais est disponible en ligne.



Capture d'écran pour FOX sous Windows 2000 (Code source ici (fox_src.zip)).

Résumé sur FOX

Nom :	FOX Toolkit
Version :	1.2.9
Systèmes d'exploitation : :	Linux, Win32
Langages :	C++
Licence :	LGPL
Avantages :	• Bonne documentation
Inconvénients :	
Environnements de développement :	
WWW :	http://www.fox-toolkit.org
Documentation :	Manuels, Tutoriels, Listes de diffusion
Projets références :	• X File Explorer (Xfe)
Distribution :	petite distribution

Autres possibilités

En plus des bibliothèques pré−citées, je souhaiterais aussi mentionner les projets suivants que je ne vais pourtant pas détailler.

- GNUstep [http://www.gnustep.org/]: utilisation limitée sous Windows
- Visual Component Framework [http://vcf.sourceforge.net/]: Pas de version Linux complète disponible

JAVA

En 1995, la compagnie Sun présentait un nouveau langage de programmation. En plus des ordinateurs de bureau, Java était prévu pour des applications professionelles (machines à café, grilles−pain, etc...). La principale percée vint tout d'abord des applications internet (« applets ») connectées directement aux navigateurs. Dans le même temps, Java fut utilisé pour des applications indépendantes (sur la machine locale), qui sont très reconnues pour la variété de leurs possibilités.

Ci−dessous, nous allons lister et expliciter brièvement les principales caractéristiques de Java.

Indépendant de la plate−forme

Java est indépendant de la plate−forme. Les applications Java consistent en du byte−code qui est interprété par une machine virtuelle. De ce fait, les applications peuvent tourner sur n'importe quel matériel pour lequel une machine virtuelle existe. L'interprétation par une machine virtuelle implique une exécution plus lente, comparée à du code compilé. Pour contrer cet inconvénient, des améliorations ont été développées, comme la compilation « juste−à−temps » (Just−in−Time −JIT), qui traduit les instructions du programme venant de la machine virtuelle en des instructions pour la machine physique. Le résultat dans ce cas est un programme stocké en mémoire, qui peut être exécuté rapidement sans interprétation. Des analyses plus poussées sur le comportement d'exécution avec la technologie HotSpot ont montré des améliorations supplémentaires.

Orientation Objet

Java est orienté objet. Les développeurs de la partie orientation objet se sont inspirés de Smalltalk. Probablement pour des raisons de performance, il existe toujours des types de données primitifs qui ne sont pas administrés en tant qu'objets.

Syntaxe du langage

La syntaxe de Java est similaire à celles du C ou C++, cependant, les bugs impliquant des contradictions ne furent pas adoptés. Un des principes de développement du langage était de combiner les meilleurs concepts des langages de programmation existant.

Quelques exemples :

- pas de pré−processeur. Le pré−processeur ainsi que les fichiers en−têtes (header files) ne sont plus nécessaires puisque toutes les informations sont lues directement depuis les fichiers de classes.
- pointeurs : Java n'utilise pas les pointeurs, le passage par référence est utilisé à la place. Une référence représentant un objet.
- corbeille : pour éviter les problèmes avec la création et la suppression des objets, l'administration des objets est réalisée par l'environnement d'exécution de Java (Java Runtime−Environment). En quittant la liste active, les objets sont automatiquement effacés. Les objets ou les listes en mémoire, qui ne sont pas autorisées, ainsi que les destructeurs défectueux sont évités par cette technique.
- exceptions : contrairement à la gestion des exceptions dans C++, les exceptions Java sont utilisées de façon plus intense, elles sont mêmes souvent obligatoires.

Bibliothèque de classes

Java inclut une bibliothèque de classes très étendue pour la génération de surfaces : JFC (Java Foundation Class). Le nom de code *Swing* a été adopté.

Securité

Le code Java est initialement ausculté pour vérifier la structure et la sécurité. Un gestionnaire de sécurité scrute les accès aux périphériques. Tout problème de sécurité est reporté comme une exception lors de l'exécution.

Java dans les projets

Les avantage mentionnés ont des effets secondaires qui rendent Java incompatible avec certains types de projets. Ces propriétés ne sont pas des erreurs ou des faiblesses, mais elles furent consciencieusement pensées, elles appartiennent à la philosophie du langage.
Parmi elles, on trouve :

- accès aux périphériques spécifiques de la plate−forme
- accès direct au matériel
- interventions dans le système d'exploitation

Java Development Kit (JDK)

Le kit de développement Java peut être téléchargé depuis le site de Sun. Il inclut le bagage classiques d'applications, de classes et de documentation en ligne Java. Les applications sont : un compilateur, un débogueur, un visionneur d'applets, ainsi qu'une variété de programmes auxiliaires, nécessaires pour générer et tester des applications et des applets Java. Le kit offre uniquement l'essentiel, le compilateur devant être lancé depuis la ligne de commande. Le paquet contient aussi l'environnement d'exécution Java (Java Runtime Environment − JRE), incluant la machine virtuelle qui est requise pour exécuter le byte−code. Enfin, la documentation décrit l'ensemble de l'API (Application Programmer Interface − Interface de programmation d'applications).

JHelloWorld

Avec l'aide du JDK standard, l'application « hello world » peut être implémentée.
Étape 1 : Génération du code source.

```
sh>vi Helloworld.java
```

```
public class HelloWorld {

  public static void main (String[] args) {


    System.out.println("Hello World!");

  }

}
```

Le nom du fichier et de la classe doivent correpondre.
Étape 2 : Traduction code vers le byte−code.

```
sh>javac Helloworld.java
```

Étape 3 : Lancement de l'application en utilisant la machine virtuelle.

```
sh>java Helloworld
```

JavaScript et Java

On pense souvent que JavaScript et Java ont des similarités. C'est faux. JavaScript fut originellement développé par Netscape comme un langage de script pour être embarqué dans HTML. Ce n'est pas un langage de programmation en lui−même, il dépend du navigateur. Le nom JavaScript est plus à considérer comme un gag marketing.

Essais de standardisation

Jusqu'à maintenant, toutes les tentatives pour rendre Java standard ont échoué. Une des raisons de ceci peut être la volonté de Sun de garder le contrôle exclusif sur les développements futurs et sur les standards de Java.

Décompilation

Un problème de Java est que les applications peuvent être décompilées. Malgré toutes les précautions de sécurité, il est toujours possible à l'heure actuelle de convertir le byte−code en code source. Ceci est possible

car le byte−code est écrit pour un processeur virtuel et contient donc des informations importantes supplémentaires. Ce surplus d'informations rend beaucoup plus aisée la décompilation du byte−code. Vous ne pouvez donc pas cacher une API propriétaire ou une connaissance spéciale dans le code.

Langage miracle ou mode de courte durée

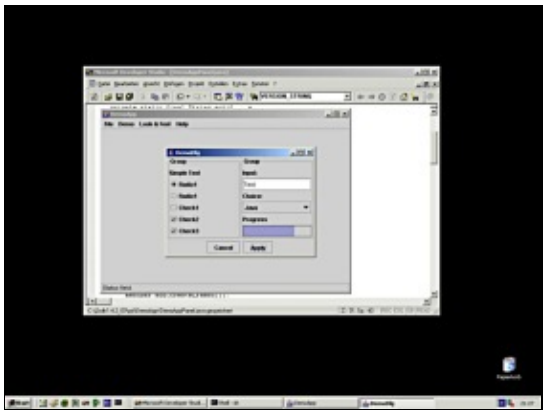
Le concept Java fut d'abord perçu comme l'ultime réponse aux développements de type plate−forme indépendant. Cependant, l'effet de mode fut vite passé. Il existe des conflits de version entre les différentes machines Java et la vitesse d'exécution est un problème. Beaucoup d'entreprises revinrent après quelques essais vers le standard C++. L'augmentation du nombre de téléchargements de wxWidgets rend compte de cela.

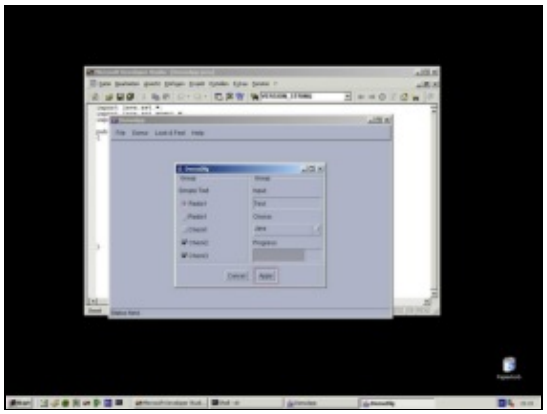
Un site intéressant dans ce contexte est :
http://www.internalmemos.com/memos/memodetails.php?memo_id=1321 où des employés de Sun apportent des arguments contre Java.

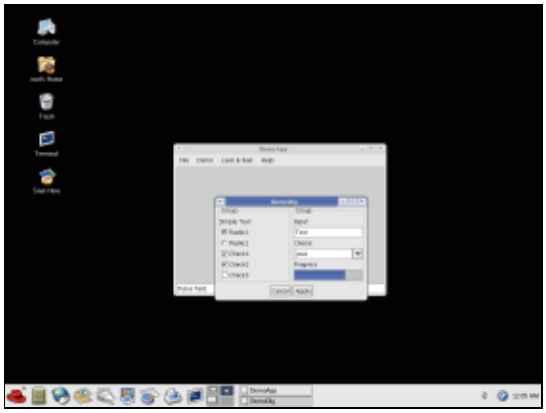
GUI avec Java

Java offre par défaut 2 possibilités de programmer des interfaces graphiques :

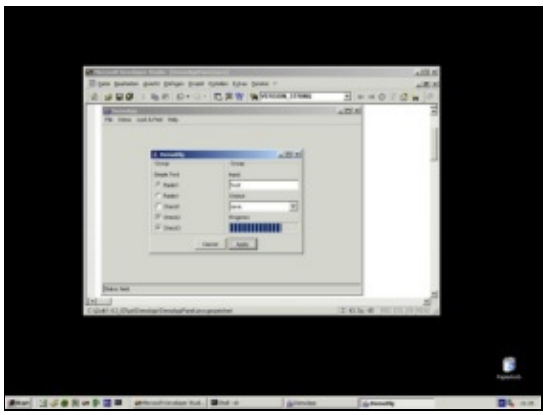
1. Java est fourni avec une riche bibliothèque de classes (JFC, Swing). Aucune fonction de système d'exploitation n'est utilisée ici. Toutes les boites de dialogues sont dessinées avec des instructions Java. Ceci permet de changer le « look&feel » lors de l'exécution. Vous pouvez voir une capture d'écran ci−dessous.
2. Les fonctions de base de AWT. AWT ne propose pas des éléments avancés (des arbres par exemple), ce n'est donc pas le bon choix pour la plupart des applications.

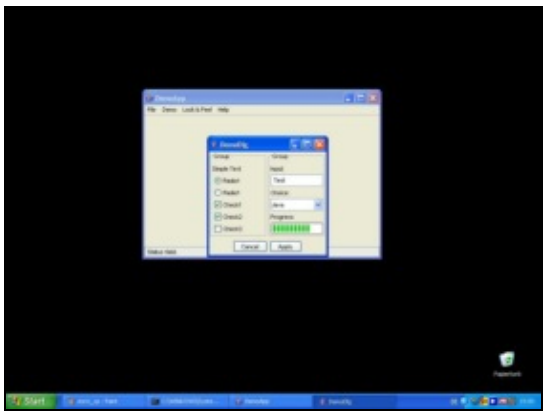






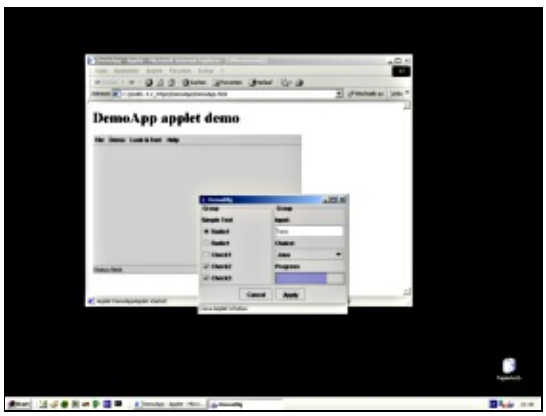
Captures d'écran Java avec l'apparence Metal−, Motif− et GTK+ (Code source ici (java_src.zip))





Captures d'écran Java avec l'apparence Windows sous Windows 2000 et Windows XP (code source identique)

Puisque tous les navigateurs utilisés supportent Java, les applications peuvent donc être écrites pour tourner au sein d'un navigateur. Cette technologie peut alors être utilisée pour des technologies embarquées où le java byte−code est téléchargé depuis un serveur web intégré dans l'application.
Les captures d'écran suivantes montrent la même application mais sous forme d'une applet intégrée dans une page web.



Capture d'écran de l'application sous forme d'applet Java (code source ici (java_applet.zip))

SWT et Eclipse

Bien que Java offre les mêmes éléments GUI que les autres toolkit, les développeurs se sont plaints. Les plus gros problèmes étaient une vitesse d'exécution trop lente et un manque de fonctionnalités. IBM a alors développé une alternative : SWT (pour Standard Widget ToolKit) qui permet l'utilisation d'éléments natifs GUI sous Java. Un des projets référence est l'environnement de développement Eclipse (aussi développé par IBM) qui offre des outils de développement indépendants de la plate−forme. Le toolkit et l'environnement de développement sont tous deux des logiciels libres.

Abréviations utilisées dans le contexte JAVA

JDK (Java Development Kit)	Le paquet Java complet pour générer des applications Java. Contient des applications, des classes Java et la documentation
JRE (Java Runtime Environment)	comprend la machine virtuelle, obligatoire pour faire tourner des applications Java.
J2ME (Java 2 Micro Edition)	Java pour composants à ressources limitées.
J2SE (Java 2 Standard Edition)	Java pour le desktop (Linux, Windows, ...)
J2EE (Java 2 Enterprise Edition)	Java pour la génération d'applications client−serveur multi−couches ainsi que les servlets Java et les JSP (Java Server Pages).
JFC (Java Foundation Class)	Classes pour développer des GUI (−>Swing)

Résumé sur Java

Nom :	JAVA 2 PLATFORM STANDARD EDITION DEVELOPMENT KIT 5.0
Version :	5.0
Systèmes d'exploitation :	• Linux, Windows, AIX, Solaris (SUN), (possibilité de MacOS, OS/2, FreeBSD, Amiga, BeOS) (Jikes −> IBM)
Langage :	JAVA
Licence :	licence propriétaire (SUN)
Avantages :	• Langage robuste (plusieurs sources d'erreurs sont éliminées par le concept du langage).
Inconvénients :	• Langage propriétaire, contrôlé en exclusivité par Sun • machine virtuelle, doit correspondre à l'architecture cible • vitesse d'exécution trop lente

	• programmer en SWT est plus complexe qu'en Swing
Environnement de développement :	e.g. Eclipse
WWW :	http://java.sun.com
Documentation :	manuels, tutoriels général : http://java.sun.com/j2se/1.5.0/docs/, http://www−e.uni−magdeburg.de/mayer/java.html SWT : http://eclipse−wiki.info/SWT, http://www.java−tutor.com/java/swtlinks.html
Projets références :	
Distribution :	très grande distribution

Kylix

Kylix est un environnement de développement multi plate−formes pour Linux et Windows. En s'appuyant sur la bibliothèque CLX (Component Library for Cross−platform) de Borland, on peut coder sous Delphi et C++ des applications capables de tourner sur les deux plate−formes. Selon un rapport de la page de Wikipedia (lien fr.wikipedia.org/wiki/Kylix). Cette bibliothèque est un conteneur de la bibliothèque Qt décrite précédemment. En plus, l'IDE Kylix est évidemment une application Linux non−native, basée sur *wine* (lien fr.wikipedia.org/wiki/WINE) dont l'exécutable doit être lié à *libwine*. Tout bien réfléchi, Kylix peut ne pas être un bon choix pour les développeurs C++ puisque l'utilisation de Qt combiné à un IDE libre est possible.

Résumé sur Kylix

Nom :	Kylix
Version :	3
Systèmes d'exploitation :	Windows, Linux
Langages :	Delphi, C++
Licence :	Logiciel propriétaire
Avantages :	• développement sous Delphi et C++
Inconvénients :	• coûts de licence
Environnement de développement :	Kylix
WWW :	http://www.borland.de/kylix
Documentation :	
Projets références :	
Distribution :	pas très diffusé

Smalltalk

Smalltalk est un classique parmi les langages de programmation. Il fut développé en 1969/70 par Xerox et est toujours aujourd'hui un bon exemple de langage orienté objet. Tout est un objet dans Smalltalk. Il n'y a pas de types de données simples. Smalltalk fonctionne comme Java et .Net (voir plus loin) dans une machine virtuelle. La syntaxe essaie de rester proche de l'anglais parlé mais est totalement différente des autres langages. Smalltalk est utilisé depuis les débuts des environnements graphiques. Smalltalk était en avance de 10 à 15 ans sur son temps. Smalltalk était une réussite jusqu'à ce que Java émerge.

Voici le programme 'Hello world !' sous Smalltalk :

```
Transcript show: 'Hello world !'; cr.
```

Smalltalk est toujours utilisé aujourd'hui. La version la plus largement diffusée est Smalltalk−80 (standardisée en 1980). Squeak est un environnement de développement puissant pour Smalltalk.

Résumé sur Smalltalk

Nom :	Smalltalk (e.g. Squeak)
Version :	3.6
Systèmes d'exploitation :	Windows, Linux, Solaris, MacOSX, Darwin
Langage :	Smalltalk
Licence :	Open Source
Avantages :	Entièrement orienté objet
Inconvénients :	Smalltalk est « poussé sur la touche » par Java et a une communauté d'utilisateurs bien moins active.
Environnement de développement :	Squeak
WWW :	http://www.smalltalk.org
Documentation :	
Projets références :	
Distribution :	pas très diffusé

Mozilla

Mozilla? Un navigateur web ? Comment peut−on programmer avec un navigateur web ? Mozilla n'est pas seulement un navigateur web mais aussi un framework indépendant de la plate−forme qui inclut différents standards tel que XUL (langage basé sur XML). XUL est utilisé pour définir la structure et le contenu d'une application. Tous les fichiers sont en texte clair. Mozillla ne distingue pas les programmes des pages web.

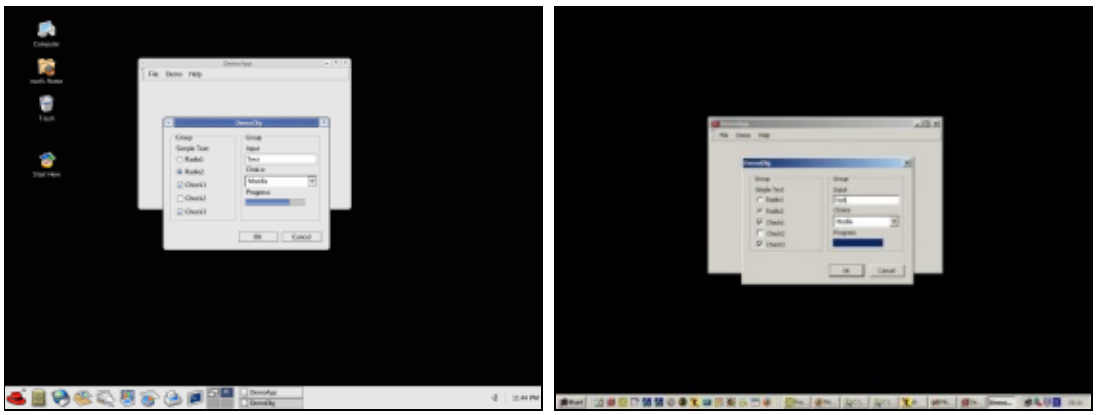
Si vous entrez la chaîne suivante dans la barre d'adresse de mozilla, alors le navigateur lui−même sera affiché :

```
chrome://navigator/content
```

Le code suivant affiche un bouton dans le navigateur Mozilla. Il ouvre une fenêtre contenant le texte « Hello World » lorsque vous cliquez dessus :

```
<?xml version="1.0"?>
<!-- Beispiel XUL Datei -->
<window
xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
<box align="center">
  <button label="Push" onclick="alert('Hello World');" />
</box>
</window>
```

Le développement logiciel avec mozilla est très différent des formes plus classiques de développement. Mozilla dispose de beaucoup d'innovations comme la séparation de l'application et de sa présentation. Ceci permet de changer l'aspect d'une application (à l'aide de « Thèmes »). Des projets connaissant le succès tels que le navigateur web Firefox montrent que c'est un framework robuste.



Captures d'écran sous Linux et Windows 2000 (code source ici (moz_src.tar.gz)).

Résumé sur Mozilla

Nom :	Mozilla
Version :	1.6
Systèmes d'exploitation :	Windows, Linux
Langage :	XUL
Licence:	Mozilla Public License, Netscape Public License
Avantages :	• concepts innovants • support pour de nombreux standards du web

	(JavaScript, Stylesheets – CSS,...) • applications tournant dans le navigateur ou en « indépendant »
Inconvénients :	
Environnement de développement :	
WWW :	http://www.mozilla.org
Documentation :	Manuels, tutoriels, listes de diffusion. E.g www.xulplanet.com
Projets références :	Mozilla Firefox
Distrubution :	très diffusé, mais rarement utilisé comme logiciel de développement.

La réponse de Microsoft

Pendant ce temps, Microsoft n'a bien sûr pas reconnu les outils en place et a introduit sa propre approche. Sous le nom de .Net, une plate–forme fut développée, dont l'un des principaux objectifs était de réduire la migration de développeurs vers la plate–forme concurrente Java. En y regardant de plus près, on s'aperçoit que plusieurs parallèles ont été faits avec les concurrents, même s'ils ont pris des noms différents. L'équivalent du 'byte–code' Java est nommé C# 'Intermediate Language' MSIL.

Qu'est ce que .NET ?

.Net est une technologie propriétaire de Microsoft qui devrait être la base de tous les produits Microsoft à venir. Le support pour la bibliothèque – jusqu'à maintenant favorisée – MFC pour Visual C++ fut abandonné avec l'introduction de .Net. .Net devrait simplifier le développement d'applications liées au réseau et/ou à Internet. Beaucoup d'idées de Java furent adoptées. .Net supporte la programmation oritentée objet et est fourni avec une bibliothèque de classes unique qui peut être utilisée par plusieurs langages de programmation (C#, VB.NET). Cela signifie que le « langage intermédiaire » – qui accède au matériel cible – est généré par le code du programme (comparez : code source Java –> Java byte–code –> machine virtuelle –> matériel physique)...

Les versions futures de Windows devraient être fournies avec le framework .NET.

Qu'est–ce que Visual Studio .NET ?

Visual Studio .NET est un environnement de programmation destiné à simplifier le développement de logiciels .NET mais il n'est pas obligatoire.

Différences entre Visual Basic (VB) et VB .NET

Bien que VB.NET – pour des raisons de compatibilité – supporte beaucoup de fonctions VB et que la syntaxe du langage fut maintenue, c'est un langage de programmation totalement nouveau.

Quel langage de programmation est le plus adapté ?

Puisque le code source de VB.NET et de C# sont transformés en MSIL, le langage de programmation importe peu. Il n'y a, par exemple, pas de différences entre du code C# et VB.NET. Le compilateur C# doit être tout de même plus adapté car il fut développé pour le framework .NET.

.NET et Linux

Malgré l'approche « indépedant de la plate–forme », Microsoft ne devrait sûrement pas développer une version Linux de .NET. C'est pourquoi une équipe de développeurs – proche de Miguel de Icaza (Ximian : Evolution) – s'est engagée dans cette tâche. Le paquet open–source *Mono*, version 1.0, est aujourd'hui disponible.

Résumé sur .NET

Nom :	Microsoft .NET–Framework
Version :	
Systèmes d'exploitation :	Windows, Linux
Langages :	C#, Windows : VB.NET
Licence :	licence propriétaire
Avantages :	• inclus dans les prochains Windows
Inconvénients :	• logiciel propriétaire • pas de version Linux de .NET disponible • API complètement nouvelle
Environnement de développement :	Visual Studio .NET
WWW :	
Documentation :	
Projets références :	
Distribution :	faiblement diffusé pour le moment

Pour conclure

Avant l'évaluation finale, reprenons l'objectif initial : le but est le développement d'une « interface », qui doit communiquer par le réseau avec le matériel connecté. Pour cette raison, le code source doit être traduit pour les plate−formes Linux et Win32. L'application ne doit pas se distinguer des autes logiciels existant sur le système. Avec cette tâche, la vue des paquets testés apparaît biaisée et on ne peut pas émettre un jugement valide globalement.

Le meilleur exemple est le toolkit FLTK. Avec lui, nous disposons d'un système puissant dans un tout petit paquet. Ses forces sont la taille du code source, une bonne interface graphique et une bonne portabilité. Ces propriétés font de ce toolkit un bon outil pour des projets d'applications graphiques ou embarquées. Pour le développement « d'interface », le nombre de classes disponible, la manipulation et l'apparence des applications générées posent problème. C'est pourquoi le toolkit FLTK est moins approprié pour ce type de tâches.

Une grosse déception pour les développeurs doit être le projet GTK+ sous Windows. La communauté Linux pourrait démontrer un peu plus d'engagement. Les avertissements placés sur le site ne sont pas faits pour coder en toute confiance. Ceci est d'autant plus regrettable que le paquet GTK+ semble bien réalisé. Le potentiel est plutôt large ; l'implémentation pour la plate−forme Windows reste en attente.

Utiliser les outsiders Smalltalk et Mozilla relève de choix personnels. Une entreprise, qui fait son chiffre sur du matériel développé en son sein, peut ne pas être portée sur des tentatives plus philosophiques. Bien que Smalltalk soit le meilleur langage de programmation orienté objet et que le langage XUL de Mozilla donne encore plus de signification au navigateur inclus, ces paquets ne sont pas des produits majeurs pour le développement logiciel.

Dans cette étude Kylix, ainsi que GTK+ pour Win32, laissent une impression plutôt négative. On retrouve très peu de la gloire du produit originel Turbo Pascal. Dans les années 80 Borland sortit un IDE puissant avec son produit, qui tournait sur des ordinateurs personnels ainsi que sur les premiers PCs. Il était reconnu pour son prix raisonnable et son exécution rapide. Depuis, bien des choses ont changé. Borland est devenu Inprise puis est redevenu Borland. Turbo Pascal a changé pour Object Pascal, puis Delphi et finalement Kylix (avec bien sûr son lot d'extensions et de changements). Son utilisation n'est plus justifiée à l'heure actuelle − du moins pour de nouveaux projets.

Dans ce contexte, Microsoft démontre qu'il reconnait le besoin actuel. Initialement, l'entreprise essaya de suivre le standard Java avec Visual++. En plus des commandes standard Java, des accès à l'API Win32 et même des accès à la base de registre Windows étaient permis (ce qui est contraire à la philosophie du langage). De plus, les exécutables Win32 étaient générés automatiquement. Après quelques empoignades judiciaires avec Sun, un avertissement devait être affiché indiquant que l'application nouvellement créée peut ne pas fonctionner avec d'autres systèmes d'exploitation. La fin de l'histoire vit Microsoft arrêter son engagement avec Java. Une toute nouvelle stratégie fut développée. Avec .NET et C# un standard tout neuf fut généré. La combinaison de Windows, .NET et C# est certainement une bonne solution, mais c'était aussi le cas avec la combinaison − maintenant obsolète − de Windows avec Visual++ et la bibliothèque de classes MFC. L'inconvénient est que l'on est inconditionnellement à la merci du fournisseur qui veut imposer « son » standard (Windows). Microsoft n'a certainement prévu aucune implémentation de .NET vers d'autres systèmes d'exploitation dans un futur proche. Le projet de conversion Mono doit d'abord prouver son intérêt. Malgré les résultats initiaux, aucune conclusion ne peut être tirée à l'heure actuelle.

Les paquets recommandés sans limitation sont Qt, wxWidget et Java. Le choix final est difficile puisque tous

trois sont capables de générer des interfaces logicielles complexes. Plusieurs facteurs sont à prendre en compte : la qualité du support, les coûts, la maturité du langage, la philosophie de programmation, etc. Les distinctions se trouvent dans les détails ; la philosophie Java, ne permet pas d'accès direct au matériel, mais peut avoir des avantages sous d'autres aspects. D'un point de vue technique, les trois concurrents peuvent convenir sans problèmes.

Reste une conclusion subjective de la part de l'auteur : les partisans de l'Open−Source peuvent se pencher sur wxWidget pour leurs projets. En plus d'un concept agréable et d'un bon support, une documentation complète est disponible.

2005−01−16, generated by lfparser_pdf version 2.51