INTERFACES MULTIMODALES

Le BoBovigny

PowerPoint, mais en mieux

Mehdi Radgohar, Marsha Rohrer, Hervé Chiquet

19/06/2007

Détrompez-vous, il est facile d'améliorer Powerpoint, un leader mondial des présentations multimédias.

Table des matières

Introduction

Qu'est-ce que le BoBovigny ?

Tout le monde connaît Powerpoint de Microsoft, un leader mondial sur le marché. Tout le monde a déjà remarqué que ce produit est très riche en fonctionnalités et assez simple d'utilisation. Imaginons qu'il soit possible de lui ajouter un aspect plus riche lors de la présentation, qu'il ne soit plus nécessaire de se trouver à côté de notre machine pour gérer nos différents slides. La solution est le BoBovigny project. Celui-ci est une couche additionnelle positionnée au-dessus de Powerpoint permettant la gestion du système à travers différentes modalités comme la voix (utilisation de Sphinx), le mouvement (Wiimote: IR et Sensor bar) et l'utilisation d'une interface tactile (interface tactile de la Wiimote). Désormais l'utilisateur jouit d'une certaine liberté, lui permettant de voyager dans un espace limité tout en contrôlant sa présentation de manière « remote ». Notons que ce projet n'est qu'un prototype.

Pourquoi le nom « BoBovigny » ?

Ce nom a été choisi pour rendre hommage à un étudiant doué et généreux. A travers une discussion intéressante avec cette personne, nous avons pu nous construire une idée assez abstraite de notre système. Un grand merci à cette personne qui se reconnaîtra.

Distribution des tâches

La gestion du projet :

Mehdi Radgohar: Responsable de la mise en place du pont entre Java et office + Détente de l'atmosphère au moment opportun

Marsha Rohrer: Responsable de la reconnaissance vocale avec Sphinx + Une touche féminine

Hervé Chiquet: Gestion de la Wiimote + Ravitaillement

Guide d'utilisateur

Prérequis

Ci-dessous vous trouverez la liste des tous les logiciels et matériels nécessaires pour faire fonctionner notre projet. Nous sommes conscient que cela peut être rébarbatif, mais ce projet est un prototype donc nous avons utilisé plusieurs astuces pour ne pas devoir ré-implémenté des tâches qui existe déjà.

Les liens vers les sites des différents logiciels se trouvent dans la Bibliographie.

- Netbeans 5.5
- Java 1.5 ou plus
- Glovepie 0.29
- J-Integra
- Home Key Logger 1.70, Free edition
- Powerpoint 2007 (Les versions XP ou 2003 devraient aussi fonctionnée)
- Une connexion Bluetooth
- Une sensor bar (disponible avec la Wii)
- La Wiimote
- Un microphone

Lancer le BoBovigny

Cette tâche est, nous le concédons, « un peu » complexe. Cependant si vous suivez les étapes décrites ci-dessous, vous devriez y arriver. Pour justifier cette complexité, nous rappelons que c'est un prototype.

1. Détecter, installer et connecter la Wiimote au PC en utilisant Bluesoleil (ou n'importe qu'elle autre programme)
2. Installer la Wii et la démarrer pour que la Sensor bar soit utilisable. Placer la barre, sous l'écran de projection.
3. Brancher un microphone et l'activer.
4. Démarrer le Key logger. Il est possible que l'anti-virus n'autorise pas ce programme à démarrer. Solution : arrêter l'anti-virus pendant l'utilisation du projet.
5. Ouvrir Glovepie et lancer le script « BoBovignyIR.PIE »

6. Importer le projet Java dans Netbeans et ajouter le paramètre « -Xmx256m » dans VMOptions dans les propriétés d'exécution du projet.
7. Importer les librairies contenu dans le dossier /lib du code source.
8. Importer les librairies « integra.jar » et « powerpoint2007.jar ». Sur le site de J-Integra, on peut trouver les librairies pour les autres versions de Powerpoint.
9. Activer le « NumLock » du clavier et lancer le projet (ManagerSingleton.java).

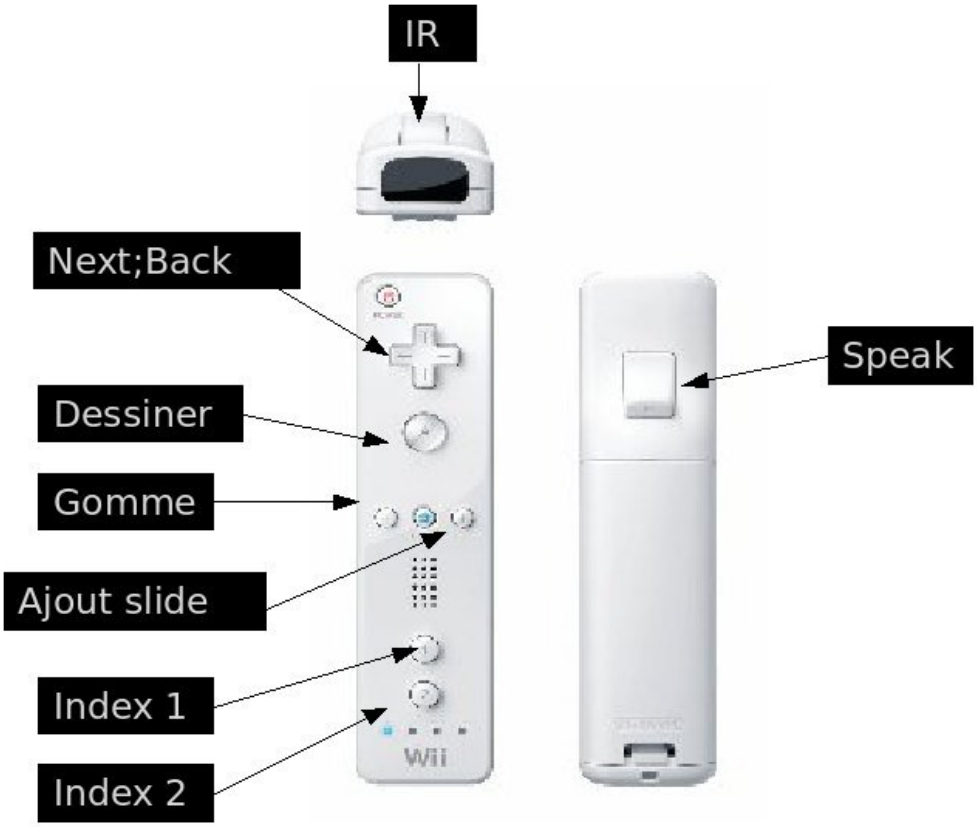
Il est possible que J-Integra produise une erreur « access denied ». Pour corriger, il faut autoriser Powerpoint à être lancé de l'extérieur. Vous trouverez les informations relatives sur cette page : http://j-integra.intrinsyc.com/support/com/doc/remoteaccess.html.

Pour modifier la présentation que vous souhaitez utilisée, vous devez modifier les variables « pathToPPT » et « nameOfPPT » se trouvant dans le fichier « Config.java ».

Les commandes

Durant l'utilisation, l'utilisateur a la possibilité de commander sa présentation soit par la voix, soit avec la Wiimote. Dans les deux images ci-dessous décrivent toutes les opérations qui peuvent être exécutées par l'utilisateur.

Pour utiliser la voix, il faut appuyer sur le bouton « speak » pour parler. Sinon rien ne sera capturé par ce dernier. Cette contrainte permet à l'utilisateur de parler sans que ses mots ne soient faussement interprétés par la reconnaissance vocale.


IR
Next;Back
Dessiner
Gomme
Ajout slide
Index 1
Index 2
Speak
Wii

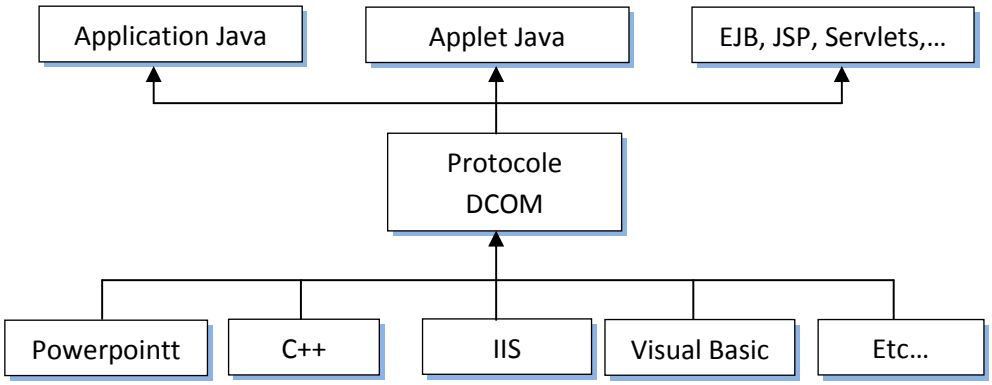

Paint
Back
Add
Next

Architecture

Interconnexion entre PowerPoint et JAVA

Composant ActiveX

Le composant ActiveX, aussi connu sous le nom de *Component Object Model*, est un outil créé par Microsoft pour permettre à d'autre programme d'utiliser ses logiciels. Dans notre cas, nous allons l'utiliser pour pouvoir effectuer toutes les tâches voulues dans PowerPoint depuis une application Java.



J-Integra

A la base nous avons commencé le projet en utilisant JACOB qui est un pont entre Java et Office. Mais durant l'implémentation, nous avons remarqué que ce dernier ne possédait pas la faculté d'être traité en multithreading et que tous les appels aux composants COM devaient venir de la classe qui avait créée l'instance de la classe. Nous nous sommes alors tournés vers une autre librairie : J-Integra. Cette dernière est malheureusement payante, mais une version de démonstration complète pour une période de 30 jours existe. J-Integra permet d'utiliser les appels COM de n'importe quelle application Microsoft depuis une implémentation Java.

Où trouver les commandes ActiveX pour PowerPoint ?

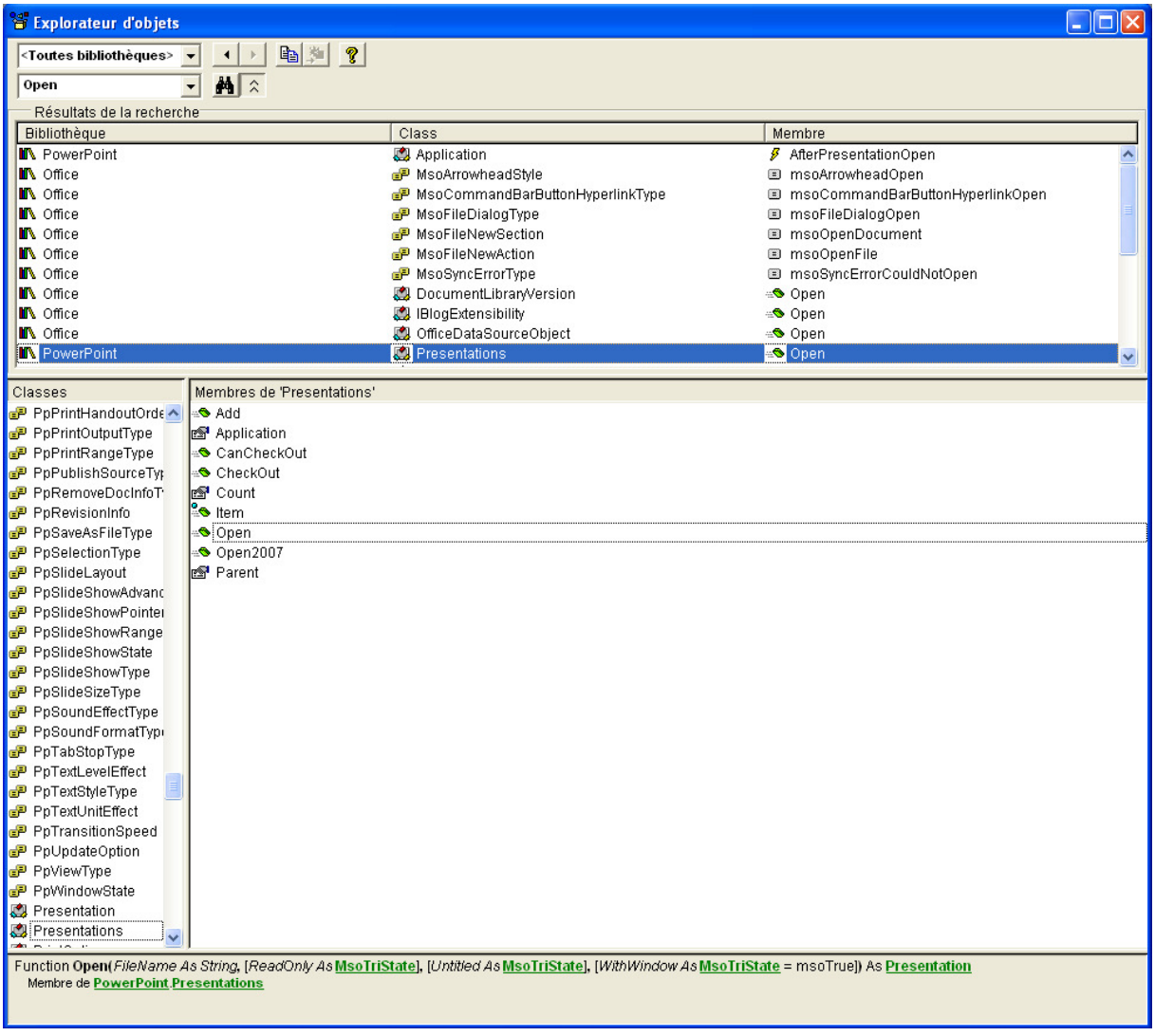
Les commandes ActiveX en Java ne sont pas triviales et, pour l'appel aux fonctionnalités de PowerPoint, il faut connaître leurs emplacements. Voici un petit exemple pour trouver les propriétés d'ouverture d'un fichier et son lancement en « Slide Show ».

```
powerpoint = new Application();

powerpoint.setVisible(-1);

presentations = powerpoint.getPresentations();

presentations.open(pathToFileToOpen, 0, 0, 0);

presentation = presentations.item(new Integer(1));
```

slideShowSettings = presentation.getSlideShowSettings();

runningSlideShow = slideShowSettings.run();

A l'origine, les mots en gras sont inconnus. Pour les trouver, on peut passer par l'éditeur « Microsoft Visual Basic » (VB) dans n'importe quelle application Office. En tapant F2, dans l'éditeur VB, l'explorateur d'objets s'affichera et toutes les opérations possibles seront listées. Pour l'exemple précédent, il suffit de faire une recherche avec « open » et de trouver l'entrée souhaitée comme le montre la figure suivante.



Sphinx

This chapter will shortly describe the speech recognizer that was used for the BoBovigny project. The next paragraphs give a short overview about what sphinx is, how it was used in this project and what were the major problems encountered during elaboration.

What Is Sphinx ?

Apart of being a statue showing a lion with a head of a person for example invented by the Egyptians, Sphinx is a speech recognizer that was entirely written in Java. It is an open source project that can be used to recognize speech. Sphinx is a powerful tool for working with speech. Because it is as powerful as that it is quite difficult to work with it as well. It includes many utilities that are not easy to handle. The next chapter describes how it was inserted in the BoBovigny project.

How Was It Used In The Project ?

For the BoBovigny project Sphinx-4 was used. Knowing that it is a powerful and at the same time confusing tool we tried to include it by adopting a demonstration that is given in the project itself.

The demo that was adopted for this project contained an infinite loop, this means that as soon as the demo is run it tries to recognize everything that one says until the application is stopped. This is not a good property for BoBovigny, since in a presentation one speaks all the time and only some key words have to be recognized. So a way had to be found to stop speech recognition. The solution was to assign a button on the Wiimote to the speech, like this speech is only recognized when the button is pressed.

Difficulties

The first challenge was to find all the libraries and so forth. The next problem that was encountered was to change the path of the grammar. For a certain reason it does not follow the same rules in the BoBovingy project as in the Demo of the Sphinx project.

Once all libraries and paths were in order we got the following error message: “Exception in thread "main" java.lang.OutOfMemoryError: Java heap space”. After some research we found out that we had to add “-Xmx256m” into the VMOptions of the Run properties of the BoBovigny project. This increases the maximal size of the heap from the default value of 128 MB to 256 MB.

In the previous chapter we said that the speech recognition has to be stopped when the associated button of the Wiimote is not pressed. It was quite a challenge to find a way to stop the speech recognition without stopping the application. Stopping the application was not a solution because the start up of the application takes too much time.

Another difficulty was that the recognition of the speech was not as reliable as expected. This is why we had to shrink the grammar that we thought of at the beginning. Doing so we could limit the problems with the recognition but they did not disappear.

Wiimote

Comment la Wiimote est gérée ?

Because Java is not able to detect key strokes a way had to be found how Java can detect that a certain button on the Wiimote was pressed. For BoBovigny the following approach was used:
Every button on the Wiimote is, with the help of Glovepie, assigned to a certain string. This means that every time that this button is pressed the assigned string is printed out. Now these strings have to be detected somehow to make Java react how it is supposed to. How this detection is done is the topic of the following paragraph.

Key logger

Now one knows that every button on the Wiimote is assigned to a certain string. Those strings are captured with the help of a Key logger. A Key logger is a tool that stores every key stroke. The log file of that Key logger is read with the LogReader class in Java. Every time the LogReader encounters a specific string the appropriate action is executed. Then the log file of the Key logger is deleted to avoid repeating one and the same operation several times.

Critiques et améliorations futures

Encore une fois nous mettons un accent sur le fait que ce projet est un prototype. C'est pourquoi beaucoup de choses pourraient y être améliorées. Par exemple le lancement du programme devrait se faire plus simplement avec un seul exécutable.

Actuellement, le moteur de reconnaissance vocale ne fonctionne pas très bien. Souvent le mot détecté est complètement faux, c'est pourquoi nous avons limité la grammaire à quatre actions : paint, add, back et next. Nous avons aussi implémenté ces actions sur les boutons de la Wiimote pour permettre à l'utilisateur plusieurs moyens d'interaction. Si la voix ne lui convient pas, il peut utiliser uniquement la Wiimote ou vice-versa et même combiné les deux. Cependant le problème est la non-fiabilité de Sphinx. Pour que l'application soit commercialisable, il faudrait que cette partie soit plus fiable et multilingue.

Une amélioration pratique pourrait être un outil de configuration des boutons de la Wiimote (un peu du genre de la souris) pour assigner une action spécifique à chacun de ses boutons.

Parmi les actions possibles que ce soit avec la voix ou avec la Wiimote, on pourrait rajouter le changement de couleur et de taille du crayon en mode dessin.

L'indexation des slides se fait actuellement de manière statique à la main et directement dans le code. Dans le futur, la possibilité d'indexer dynamiquement un slide durant la présentation pourrait être introduite et leur nombre pourrait ainsi être plus grand (limité à deux pour le moment). De plus, au lieu de donner des numéros aux slides, il pourrait être intéressant d'utiliser des noms communs et significatifs pour indexer un slide.

Conclusion

Notre projet nous a permis de remarquer que Sphinx n'est pas très fiable pour une application telle que celle-ci. Actuellement, on peut le dire franchement, c'est un prototype, mais on peut déjà constater qu'il apporte plus de richesse à Powerpoint. Nous sommes très contents d'avoir pu construire ce projet, car nous avons dû faire appel à notre imagination pour le mettre sur pied.

Espérons que ce projet amène de nouvelles idées sur le marché pour améliorer les différents programmes de présentations.

Bibliographie

1. The JACOB Project, http://sourceforge.net/projects/jacob-project/
2. J-Integra, http://j-integra.intrinsyc.com/
3. Netbeans, http://www.netbeans.org/
4. Java 1.5 ou plus, http://www.java.sun.com/
5. Glovepie 0.29, http://fr.wiili.org/index.php/GlovePIE
6. Home Key Logger 1.70, http://www.kmint21.com/keylogger/
7. Bluesoleil, http://www.bluesoleil.com/
8. Sphinx4, 2007, http://cmusphinx.sourceforge.net/sphinx4
9. Increase heap size in Java, 2007, http://hausheer.osola.com/docs/5
10. WiiLi, http://fr.wiili.org/index.php/Accueil