Doan Chi-Minh
Lou Ruding
MASTER 2 SDRP

Le 09 Janvier 2006

RESEAUX AVANCES : PROJET DEVELOPPEMENT SUR COMPOSANT MOBILE

Université Bordeaux 1 – Master SDRP – Année 2005/06

1. Présentation du sujet

Dans le cadre du projet de Réseaux Avancés, nous avons eu à développer un gestionnaire de sons sur un composant mobile. L'objectif de ce système est de pouvoir contrôler différents volumes d'une machine distante à partir du mobile jouant le rôle de télécommande. La première étape a consisté à s'informer sur le support d'exécution, à savoir le téléphone portable Sony Ericsson P910i. Ensuite, nous avons défini l'architecture la plus adaptée qui soit simple d'utilisation et surtout assez "légère" pour être gérer par le téléphone.

2. Mise en place du projet

2.1 Spécifications techniques

Nous allons tout d'abord détailler les caractéristiques de notre environnement, puis nous préciserons quels sont les différents outils que nous avons utilisés afin de mettre en oeuvre notre application.

La cible de cet outil est un téléphone mobile avec une connexion réseau en bluetooth et une machine virtuelle JAVA. Pour tester notre application, nous avons eu l'opportunité de manipuler le Sony Ericsson P910i. Celui-ci est tourne sous Symbian OS V7.0. Il supporte des applications développées dans différents langages dont JAVA en utilisant le kit de développement J2ME CLDC/MIDP 2.0 avec JSR-120 et JSR-82.

Pour intégrer ce gestionnaire de volumes, nous nous sommes servis de :

J2ME (Java 2 Platform, Micro Edition) : c'est l'API correspondant au système JAVA tournant sur le P910i. Elle dispose entre autre de classes qui permettent de créer des interfaces graphiques. Il est ainsi possible de dessiner un affichage à l'aide de primitives géométriques simples en utilisant la classe ***javax.microedition.lcdui.Canvas*** ou bien d'utiliser la classe ***javax.microedition.lcdui.Gauge*** fournissant des objets prédifinis tels que des barres graphiques.

CLDC (Connected Limited Device Configuration) : définit les caractéristiques de bases de l'environnement d'exécution. Elle spécifie la machine virtuelle utilisée ainsi que des caractéristiques du composant (par exemple si c'est un téléphone avec une connexion bluetooth).

WTK2.2 : c'est un outil fourni par Sun. Il permet de simuler différents composants mobiles dont le P910i en spécifiant l'environnement suivant : JTWI 1.0 / MIDP 2.0. A noter que les résultats fourni par le simulateur et les tests réels sur le téléphone ne donne pas les mêmes résultats. Ceux-ci diffèrent légèrement mais on n'a pas eu une incidence importante sur le déroulement du projet.

2.2 Développement

Afin de mettre en oeuvre cette application, nous avons défini le cahier des charges suivant :

- le volume d'un élément (volume générale, micro, CD ...) est défini par une valeur entière variant entre 0 et 100,
- il doit être possible de couper le son à l'aide d'une fonction "mute",
- une implémentation assez "légère", c'est-à-dire pas trop gourmande en terme d'espace mémoire,
- un protocole de communication respectant les spécifications définies par le groupe 8 et facilement traitable par le gestionnaire de son côté machine.

Le protocole mis en place afin de communiquer avec la machine distante fonctionne de manière relativement simple. Nous avons recensé les différents évènements dont nous aurons besoin, à savoir :

- synchronisation des différents volumes avec le système distant,
- augmenter le volume,
- diminuer le volume,
- couper le son par l'intermédiaire de la fonction mute.

Dans tous les messages transmis, nous avons décidé qu'il n'y aurait aucune vérification de la transmission des données avec un accusé de réception car nous avons jugé cela inutile. En effet, la vérification peut se faire de manière auditive à chaque fois qu'un événement à lieu. La vérification est immédiate. De plus, un accusé de réception signifie l'envoie d'un message supplémentaire ce qui peut être un inconvénient si d'autres messages pour la gestion de la souris et du clavier par exemple, doivent être envoyés.

Si jamais nous avions opté pour une vérification par accusé de réception, cela aurait donné la possibilité de renvoyer la trame initiale. Mais si il y a un problème au niveau de la machine distante et qu'elle renvoie systématiquement une erreur, on pourrait renvoyer indéfiniment la trame initiale inutilement alors que le problème se situerait du côté serveur.

La synchronisation se fait au démarrage, une fois la connexion bluetooth établie. On envoie d'abord une trame de données vide afin d'informer le serveur distant que le téléphone est prêt à se synchroniser. Ensuite, il doit y avoir autant de réception de trame que de volume à régler.

Par la suite lorsque l'utilisateur souhaite faire varier le volume, il a la possibilité de le faire via le stylet du téléphone. Dans ce cas là, il doit maintenir une pression sur le curseur et le déplacer. Deux possibilités sont envisageables : soit on envoie une trame à chaque fois que le curseur évolue d'une unité, soit on envoie le nouveau volume sonore une fois que la pression du stylet est relâchée.

La première solution implique un envoie relativement massif de messages. Cela peut être contraignant si l'utilisateur s'amuse à faire des aller retour avec le curseur. L'avantage de celle-ci est que le son sera constamment synchronisé avec celui du serveur.

La seconde alternative ne permet pas de maintenir le volume du serveur synchroniser en temps réel. Elle a l'avantage d'être plus légère à exécuter sur le système mobile dont la mémoire est limitée.

En accord avec le groupe 8, il a été défini que les messages concernant le son seraient contenu

dans une trame de taille deux entiers (8 octets). La première partie équivalente à un entier contient le numéro de service qui nous a été attribué pour le son, à savoir le numéro 0. La seconde partie de cette trame doit contenir deux informations principales qui sont l'objet concerné (ça peut être le micro, le son général ...) et quel est le nouveau statut de cet objet (c'est-à-dire le niveau du volume).

Le second entier peut donc être divisé à nouveau en deux octets :

- le premier octet défini le type de son à modifier (cette méthode permet de gérer jusqu'à 127 sons différents) :
  - 0000 0001 correspond au volume général,
  - 0000 0010 correspond au volume d'enregistrement du micro,
  - 0000 0011 correspond au volume du lecteur CD.
- le second octet contient le niveau sonore. On sait qu'un son ne peut être supérieur à 100 et un octet permettant d'aller jusqu'à 127, il sera possible de couvrir toute la largeur du niveau sonore.

Pour dessiner l'interface graphique, nous avons eu le choix entre soit utiliser la classe ***Canvas***, soit ***Gauge***.

La première est surtout utilisée pour écrire des applications nécessitant la gestion d'évènements de bas niveau avec un rafraîchissement immédiat sur l'affichage. D'après l'API, elle est utile pour la création de jeux. Ceci n'étant pas notre cas, nous nous sommes tournés vers la seconde classe.

Celle-ci a l'avantage de nous fournir exactement ce dont nous avons besoin afin de définir une barre de volume. En effet, elle permet d'obtenir un objet graphique telle qu'une barre représentant un entier défini.

De plus, dans un souci de ne pas encombrer la machine virtuelle, la seconde classe, à savoir ***Gauge***, est plus adaptée. En effet, elle permet de dessiner plus rapidement et plus simplement l'interface. L'inconvénient de cette classe est qu'elle organise les différents éléments qu'elle contient sous la forme d'un tableau d'une colonne. Cela nous a posé des contraintes pour dessiner une interface plus conviviale.

Nous avons tenté une implémentation d'une couche de communication par bluetooth en utilisant le code d'un des exemples fourni avec la WTK2.2 (BluetoothDemo). Nous avons supprimer les services qui nous étaient inutiles mais cela n'a pas fonctionné.

2.3 Extensions

Il serait intéressant de pouvoir intégrer cet outil avec le travail des autres groupes afin de faire un gestionnaire complet du système distant. Pour l'instant, il n'a pas été possible de communiquer avec la machine serveur.

Enfin, on pourrait imaginer que lors de la synchronisation, on récupère toutes les caractéristiques sonores de la machine distante. En effet, dans la version actuelle, il est nécessaire de connaître les caractéristiques de sa carte audio et d'ajouter manuellement dans le code une barre de volume. De cette manière, on pourrait créer une barre graphique pour chaque volume en fonction des capacités audio de celle-ci.

3. Conclusion

Ce projet nous a permis d'une part de manipuler du matériel haut de gamme qui n'est pas forcément accessible à toutes les bourses. Nous avons donc pu constater quelques capacités des matériels les plus récents. D'autre part, cela nous a permis de nous insérer dans un projet de faible envergure qui faisait appel à des domaines qui nous étaient inconnus.