Jeux en réseaux sur plate-formes Mobiles

Patrice Torguet

IRIT Equipe VORTEX

Jeux sur mobiles

- **Le modèle simple :**
  - **Téléchargement du jeu et/ou de niveaux de jeu**
  - **Jeu en local**
  - **Envoi du score vers un site web spécifique**
- **Jeu à plusieurs**
  - **Tour par tour ou semi temps-réel**
  - **Chat**
  - **Via Bluetooth/WiFi ou GPRS/3G**
- **Vers les MMO sur Mobile :**
  - **Accès à certains aspects d'un MMO sur mobile**
  - **Exemple :**
    - ✓Gestion d'enchères
    - ✓Gestion du portefeuille d'une entreprise virtuelle

Jeux sur mobiles

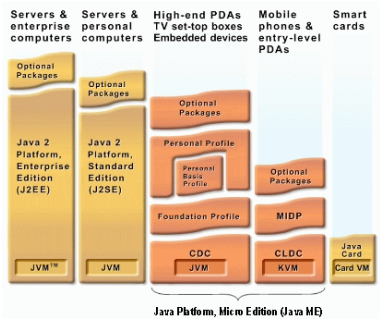
- **Programmation**
  - **Java**
    - ✓Java ME (Micro Edition) de SUN
  - **C/C++**
    - ✓Brew (Binary Runtime Environment for Wireless) de Qualcomm
    - ✓Autres : Symbian OS, Windows Mobile, ExEn, Mophun
  - **Middleware réseau**
    - ✓Neutron de Exit Games

Jeux sur mobiles

- **Performances réseaux**
  - **2,5 G (GPRS, EDGE)**
    - ✓latence entre 3s et 900 ms
    - ✓débit entre 40 et 200 Kbps
    - ✓près d'un milliard de mobiles au monde
  - **3 G (UMTS)**
    - ✓latence ~ 500 ms
    - ✓débit entre 300 Kbps et 400 Kbps
    - ✓environ 100 millions de mobiles dans le monde
  - **3,5 G (HSDPA)**
    - ✓latence ~ 100 ms
    - ✓débit entre 2 Mbps et 15 Mbps
    - ✓moins d'1 million de mobiles dans le monde
  - **WiFi/Bluetooth**
    - ✓latence < 1 ms en connexion directe
    - ✓débit entre 1 Mbps et 74 Mbps
    - ✓moins d'1 million de mobiles dans le monde

Java ME

- **Avantage**
  - **Développement unique pour de nombreuses plate-formes (http://wireless.java.sun.com/device/)**
  - **Emulateur permettant de tester les applications**
- **Inconvénient**
  - **Exécution plus lente**
- **Outils :**
  - **J2SE SDK pour compiler**
  - **Sun Java Wireless Toolkit**
  - **Eclipse ou un autre IDE**



Java ME

- **Les packages disponibles**
  - **java.lang : attention pas de Float et Double (mais réels en virgule fixe disponibles)**
  - **java.util : Hashtable, Vector, Random...**
  - **java.io : a peu près complet mais en lecture seule**
  - **javax.microedition.io : classes réseau**
    - ✓Attention : seule la gestion du protocole HTTP est requise par la norme
  - **javax.microedition.lcdui : IHM pour mobile**
  - **javax.microedition.midlet : équivalent de la classe Applet pour JME**
  - **javax.microedition.rms : base de donnée pour stocker vos infos**
  - **bindings pour OpenGL ES**
- **MIDlet : c'est ce qui correspond à l'application**
  - **.jar : qui contient les .class et les ressources**
  - **.jad : (Java Application Descriptor) décrit votre application (nom, numéro de version, quel profil de Java ME vous demandez...) et généré par un outil spécial**

Java ME

- **MIDlet**
  - **Une classe doit dériver de MIDlet et doit avoir**
    - ✓Un constructeur par défaut (sans paramètres) : c'est celui qui est appelé quand l'utilisateur choisit votre jeu dans la liste des applications java installées
    - ✓startApp() : appelé juste après le constructeur
    - ✓pauseApp() : appelé lors d'une demande de pause de l'environnement (lors de la réception d'un appel typiquement
    - ✓destroyApp(boolean inconditionnel) : appelé lors de la fin de l'application (le booléen indique si l'application à le choix ou non de demander de ne pas s'arrêter - pour demander à l'utilisateur s'il veut vraiment terminer par exemple)
  - **Autre classe importante : l'IHM qui doit dériver de Canvas et doit implémenter 3 méthodes**
    - ✓keyPressed(int keyCode) : indique qu'une touche a été appuyée
    - ✓keyReleased(int keyCode) : indique qu'une touche a été relâchée
    - ✓paint(Graphics g) : méthode qui dessine votre interface graphique

Java ME

- **Utilisation de l'interface HttpConnection**
  - **1) Création de la connexion :**
    - ✓HttpConnection c = (HttpConnection) Connector.open("http://www.monserveur/mapage.php");
  - **2) Choix de la méthode de connexion (exemple POST) :**
    - ✓c.setRequestMethod(HttpConnection.POST);
  - **3) Envoi d'en-têtes HTTP (au minimum le Content-Type)**
    - ✓c.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
  - **4) Envoi de données**
    - ✓OutputStream os = c.openOutputStream();
    - ✓String params = "name=" + name;
    - ✓os.write(params.getBytes());
  - **5) Réception de données**
    - ✓StringBuffer sb = new StringBuffer();
    - ✓InputStream is = c.openDataInputStream();
    - ✓int chr; while ((chr = is.read()) != -1) sb.append((char) chr);
    - ✓System.out.println(sb.toString());

Brew

- **SDK pour développer en C et en C++ (avec des limitations) pour plate-formes mobiles**
- **http://brew.qualcomm.com**
- **Avantages**
  - **Plus rapide et plus puissant que JME - accès natif au hardware**
  - **Meilleure intégration avec OpenGL ES (3D)**
  - **Sockets UDP et TCP disponibles (éventuellement plusieurs)**
  - **Différents modes de commerce (vente unique, location par mois ou location par nombre d'accès)**
  - **Simulateur complet (gestion de SMS, données GPS, pilotable automatiquement)**
- **Inconvénients**
  - **Coût du compilateur et du programme TRUE BREW (certification)**
  - **Moins répandu que JME surtout en Europe**

Brew

- **Bibliothèques disponibles**
  - **OpenGL ES**
  - **Réseau (HTTP, TCP et UDP)**
  - **Gestion de fichiers**
  - **IHM (incluant un viewer HTML)**
  - **Gestion de bases de données génériques ou spécifiques (au mobile)**
  - **Gestion des sons**
  - **Animation 2D (sprites)**
  - **Cryptographie**
  - **Données GPS**
- **Concept d'interface**
  - **Tout fonctionne suivant ce concept : une structure stockant des pointeurs de fonctions (permet la simulation de l'héritage)**
- **Application (Applet)**
  - **Code exécutable**
  - **Ressources**
  - **MIF (fichier d'identification de l'application)**

Brew

- **Applet**
  - **inclusion d'un fichier .bid qui stocke le ClassID (identificateur de l'appli)**
  - **AEEClsCreateInstance : fonction obligatoire qui vérifie le ClassID et crée une instance de structure qui représente l'applet grâce à la méthode AEEApplet_New**
  - **Exemple de structure :**

```
typedef struct_myapp {
  AEEApplet   a; // obligatoire en 1er - représente l'héritage
  // variables spécifiques ici
} myapp;
```

  - **Lors de l'appel à AEEApplet_New on spécifie la taille de la structure applet et des pointeurs de fonctions pour traiter les événements et détruire l'applet.**
  - **Exemple d'utilisation d'une interface :**
    IDISPLAY_Update (pMe->a.m_pIDisplay);
  - **pMe est le pointeur vers l'instance de la structure applet, m_pIDisplay est un pointeur vers une interface IDisplay (fonctions d'affichage) et IDISPLAY_Update est une macro qui invoque la fonction pointée par Update de IDisplay ou d'une interface descendante.**

Brew

- **Exemple de code réseau avec ISockPort (compatible IPv4 et v6)**
  - **Création de socket (pIsockPort - de type ISockPort*)**

```
int result = ISHELL_CreateInstance( pMe->a.m_pIShell, AEECLSID_SOCKPORT,
                                    (void **)&pMe->pISockPort );
```

  - **Précision de l'adresse et du port du serveur (sa de type AEESockAddrStorage)**

```
pMe->sa.wFamily = AEE_AF_INET;
pMe->sa.inet.port = HTONS( 12345 );
INET_PTON( pMe->sa.wFamily, "10.0.1.127", &pMe->sa.inet.addr);
```

  - **On peut choisir le réseau support (si plusieurs disponibles WLAN et UMTS par exemple)**

```
ISOCKPORT_SelectNetwork( pMe->pISockPort, AEE_NETWORK_WLAN );
```

  - **Ouverture du socket (on précise ici qu'on veut ouvrir une connexion TCP)**

```
pMe->wSockType = AEE_SOCKPORT_STREAM;
ISOCKPORT_OpenEx( pMe->pISockPort, pMe->sa.wFamily, pMe->wSockType, 0 );
```

  - **Connexion du socket (éventuellement asynchrone : si le résultat est AEEPORT_WAIT on re-demandera une connexion plus tard)**

```
int result = ISOCKPORT_Connect( pMe->pISockPort, &pMe->sa );
```

  - **Ecriture (ici aussi c'est asynchrone donc on stocke les données et les indices - à écrire et écrit - dans la structure de l'applet)**

```
int result = ISOCKPORT_Write( pMe->pISockPort,
                              pMe->pBuffer + pMe->written,
                              pMe->toWrite - pMe->written );
```

Neutron : un middleware réseau pour mobiles

- **Neutron 4.0**
  - **mise en place de serveurs de jeu aux USA et en Europe**
  - **bibliothèques disponibles pour**
    - ✓Mobiles : JME, Brew, Flash Lite, Windows Mobile, RIM Blackberry
    - ✓PC : Java, .Net, Flash
  - **gestion multi-plate-forme, multi réseaux**
  - **gestion des abonnements et de la facturation**
  - **service après vente développeurs**
- **Ce qu'il gère**
  - **téléchargement de jeux et de données pour jeux**
  - **gestion des scores et des tournois/mise en relation de joueurs**
  - **multi-joueur par tour et semi temps-réel**
  - **chat intégré au jeu**
  - **multi-joueur et multi-plate-forme**

Autres outils/SDK

- **Symbian OS : Système d'exploitation pour mobiles Nokia (la plupart) ainsi que les mobiles 3G de Samsung et Sony Ericsson**
  - **Spécifique à ces mobiles, donc plus rapide et plus standard (pas de portage)**
- **Windows Mobile : Système d'exploitation de Microsoft pour les terminaux mobiles**
  - **Mêmes avantages/inconvénients que Symbian OS mais plutôt pour PDA et Smartphones haut de gamme.**
- **ExEn : Execution Engine de In-Fusio**
  - **Spécifique aux jeux (game engine)**
  - **Basé sur JME**
  - **Permet de ne pas avoir à faire de portage**
- **Mophun de Blaze Global**
  - **Similaire à ExEn (game engine)**
  - **Existe depuis longtemps, très sécurisé (piratage très difficile) mais peu performant**