Cours 6: Initiation Android

Christophe Morvan

Université de Marne-la-Vallée

19 décembre 2012

Android

De quoi s'agit-il ?

Système d'exploitation pour matériel embarqué

- Téléphone mobile
- Tablette
- Livre électronique
- Télévision
- ...

Plan

1. Vue d'ensemble
   - Historique
   - Logiciel libre
   - Organisation logicielle
2. Architecture, vision système
   - Architecture en détails
   - Sécurité Android
3. Développement Android

Historique (entreprise)

Chronologie

- Oct. 2003 Naissance d'Android
- Août 2005 rachat de la société par Google
- Nov. 2007 fondation de la *Open Handset Alliance*
- Nov. 2007 sortie de la première version beta
- Sept. 2008 première version stable – premier téléphone
- Fév. 2012 : 300 millions de matériels Android (850 000 téléphones activés par jour)

Historique des versions

Versions majeures

- Nov. 2007 sortie de la première version beta
- Sept. 2008 1.0 première version stable – premier téléphone
- Oct. 2009 2.0 puis 2.1 (Eclair)
- Mai 2010 2.2.x (Froyo)
- Dec. 2010 2.3.x (Gingerbread)
- Fev 2011 3.x (Honeycomb) $\rightarrow$ tablettes
- Octobre 2011 4.0.x (Ice cream sandwitch) $\rightarrow$ tablettes + téléphones
- Juin 2012 4.1 puis 4.2 (Jelly Bean)

Android Open Source

Android est *Open source*

Depuis fin 2008, le modèle de développement d'Android est fondé sur l'ouverture du code
La licence est celle de Apache
Cette licence est relativement permissive

Le logiciel libre - GNU

1983 : Début du projet GNU par Richard M. Stallman

GNU = GNU's Not Unix

Objectif: créer un Unix libre

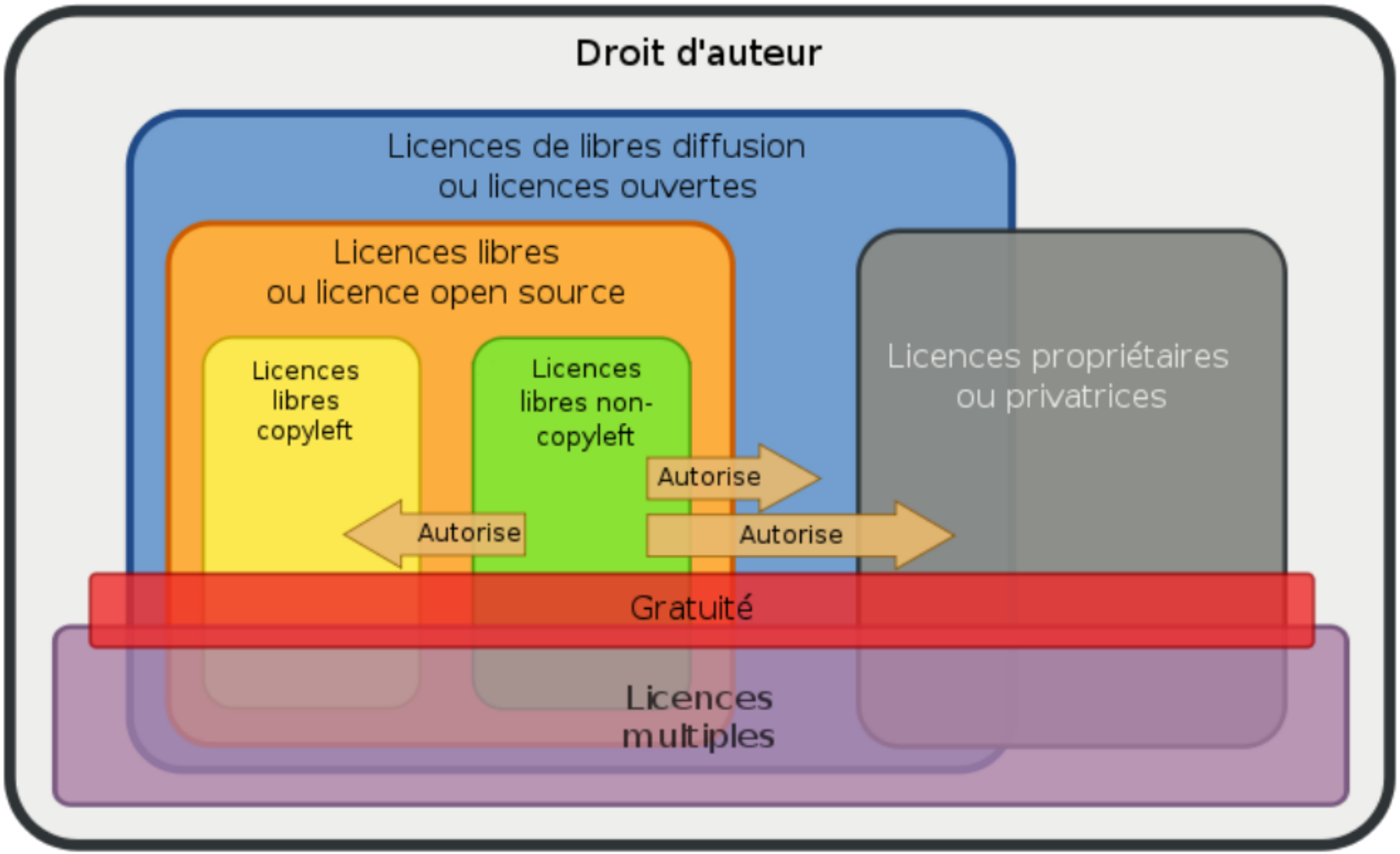
Principes du logiciel libre (licence GPL)

- droit d'utiliser le logiciel
- droit d'étudier les sources
- droit de modifier ces sources
- droit de diffuser ses modifications avec les sources

Conséquences

- Libre $\neq$ non protégé
- Libre $\neq$ gratuit

Les licences



Quelques marques historiques

Le projet Gnu comporte une centaine de logiciels libres
(Langages de programmation, éditeurs de texte, client courriel...)

Hors du projet Gnu

Avr. 1995 Démarrage de Apache

Fev. 1998 Démarrage de Mozilla

Juil. 2000 Démarrage de OpenOffice.org

2001 Démarrage de VideoLAN (vlc)

Mai 2002 OpenOffice.org 1.0

Juin 2002 Mozilla 1.0

2008 Firefox 3.0 – OpenOffice.org 3.0

Jan. 2011 LibreOffice 3.3

2011 Firefox 4.0 ($\rightarrow$ 17.0 aujourd'hui)

Fev. 2012 VLC 2.0

Linux - Chronologie

Android – Linux

Android s'appuie sur le noyau Linux.

Noyau Linux

Linux est un système d'exploitation lancé en octobre 1991 par Linus Torvalds un étudiant Finlandais de 21 ans

Chronologie

Oct 1991 Linux 0.01

Mar 1994 Linux 1.0 ($\rightarrow$ mars 95)

Jui 1996 Linux 2.0 ($\rightarrow$ août 99)

Mai 2000 Linux 2.4

Déc. 2003 Linux 2.6

Juil. 2011 Linux 3.0

Android libre ?

Motif numéro 1

Favoriser l'adoption d'Android par les fabriquants de téléphones.
Ils peuvent modifier la version canonique d'Android pour correspondre à leurs souhaits (ceux de leurs clients)
Ex : Samsung *TouchWiz*, HTC *Sense*

Autres motifs

- Utiliser le noyau linux
- Bénéficier du travail de développeurs tiers

Attention

Le projet Android est libre, pas dirigé par la communauté.

Organisation logicielle

Principes généraux

Noyau Linux

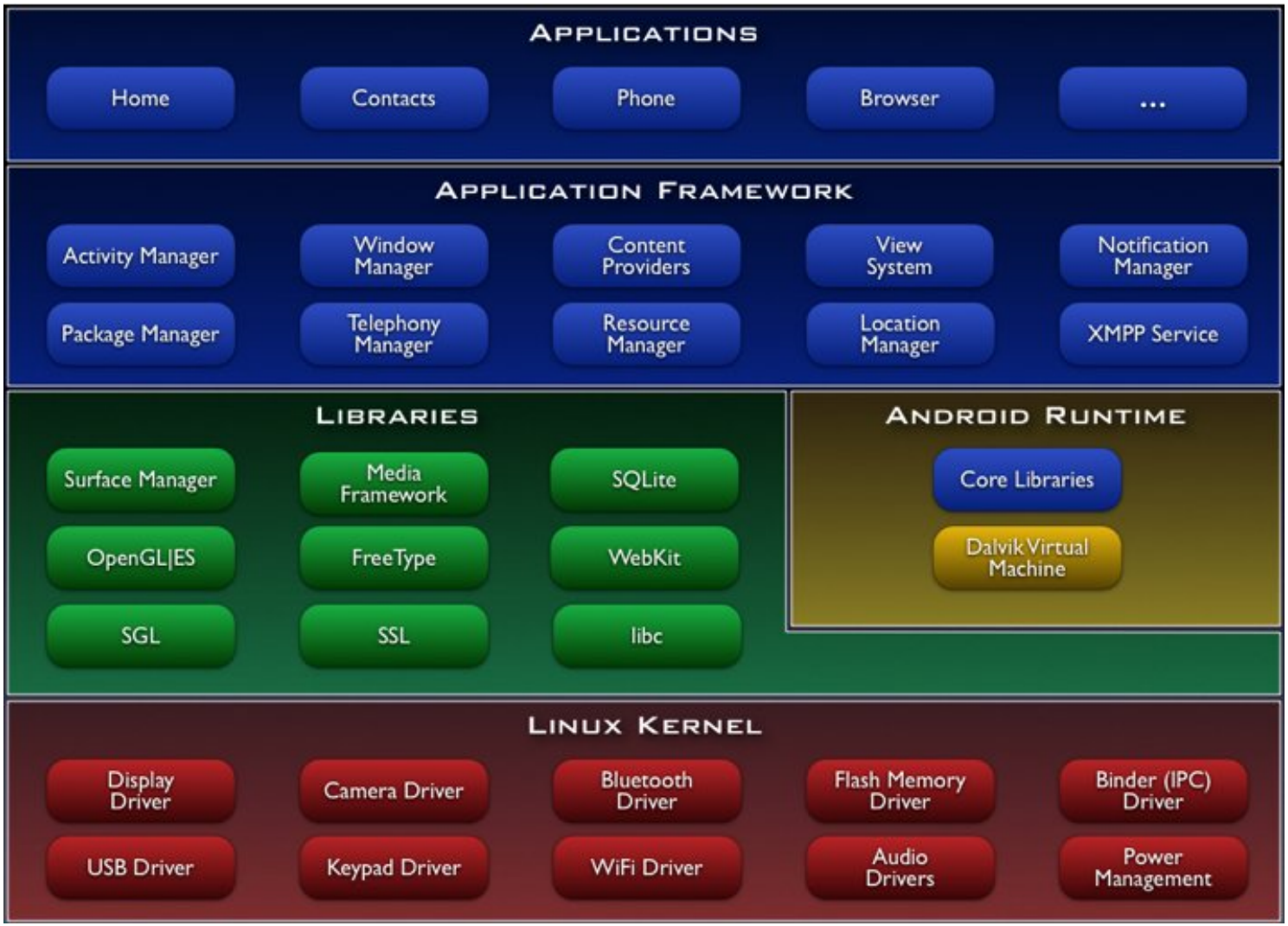
Un unique utilisateur humain qui n'a que quelques privilèges d'administration (au travers une application dédiée)

L'utilisateur peut installer des applications (principalement en utilisant une application dédiée)

En principe ces applications sont écrites en java et sont exécutées par une machine virtuelle spécifique (Dalvik)

Chaque application doit spécifier ses utilisations (réseaux, téléphonie, contacts, ...)

Architecture Android



Matériel

Processeur

Android est conçu pour être déployé en priorité sur des architectures matériels embarquées

Arm x86

Environnement

Multitude de capteurs physiques

- Interface tactile
- GPS, accéléromètres, gyroscopes, magnetomètres, ...

Multitude de connexion réseau

- gsm/edge, umts
- Wifi
- Bluetooth

Quelques applications clés

Téléphonie

Android comprends une application principale pour accéder aux fonctions de téléphonie.

Interface

Une application particulière (*home/Launcher*) donne accès à l'utilisateur aux fonctions et aux applications installées sur l'appareil.

Applications Google (non ouvertes)

En dehors de l'ensemble des applications ouvertes intégrées dans Android, Google fournit plusieurs applications Android.

Activités

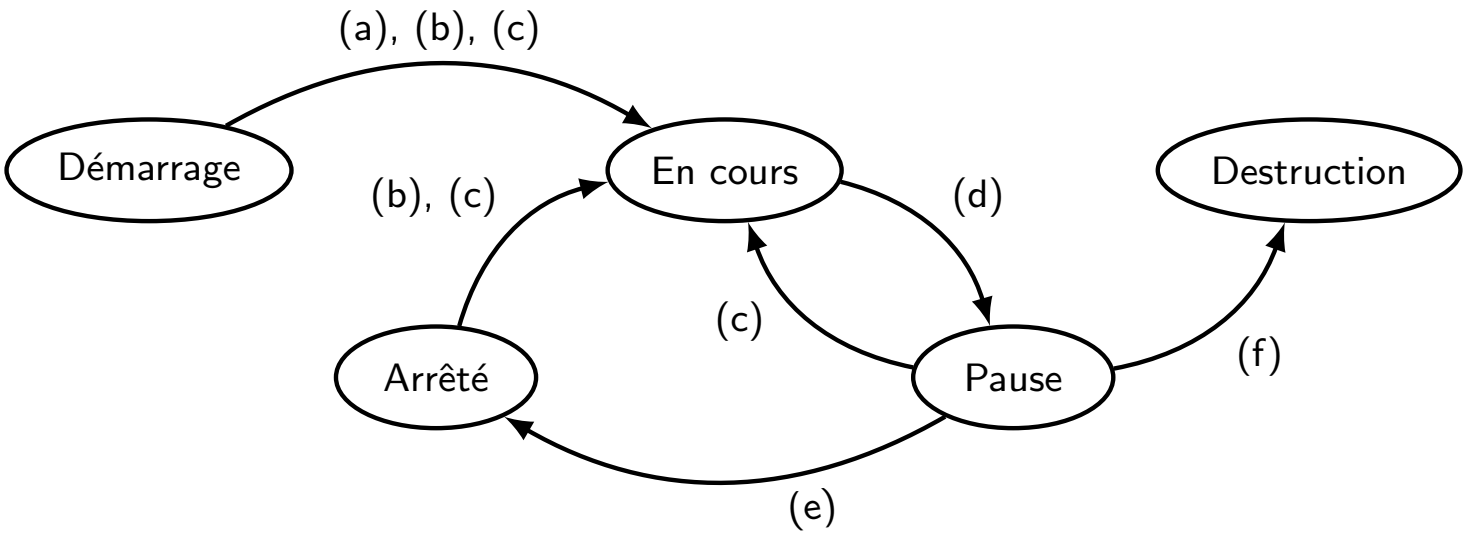
Observation

Le démarrage et l'arrêt d'un processus/programme sont des opérations relativement coûteuses.
Dans un environnement embarqué il peut être judicieux de limiter les démarrages et les arrêts

Activités

Chaque processus est une activité qui s'exécute sur sa propre machine Dalvik
Lorsqu'une application est quittée par l'utilisateur elle est interrompue, mais persiste en mémoire
Moyen de réactivation léger
Le gestionnaire d'activité détermine la fin effective des applications en fonction des besoins

Cycle de l'activité Android (simplifié)



Méthodes spécifiques

Il est possible d'écrire les méthodes suivantes :
(a)`onCreate()`, (b)`onStart()`, (c)`onResume()`,
(d)`onPause()`, (e)`onStop()`, (f)`onDestroy()`

Activités : permissions

Permissions

Accéder à de nombreuses fonctions est soumis à permissions
Elles sont vérifiées par la machine Dalvik
(Les autorisations sont données à l'installation de l'application)

Important

Chaque application détermine un ensemble de permissions
L'utilisateur choisi d'installer l'application ou pas
L'utilisateur ne peut pas installer l'application et restreindre ses permissions (par défaut)

Système de messages : `intent`

Définition des `intent`

Les activités peuvent communiquer entre elles à l'aide de messages, les `intents`

Exemple

Intent : Ouverture de page web → navigateur
Intent : Ouverture d'une vidéo dans une page web → activité vidéo

Une activité peut employer un intent précis (visant une activité particulière)
Ou bien un intent général où toute activité prenant en charge cet intent peut répondre (au choix de l'utilisateur)

Activités de fond : services

Définition

Les services sont des activités sans interfaces
Ils sont plus légers et possèdent moins d'états
En principe le système ne doit pas interrompre un service

Exemple

Systèmes de notification
Clavier logiciel
Dropbox

Activités de contenu : *content providers*

Définition

Les serveurs de contenu sont des applications spécialisées qui servent d'interfaces à des bases de données (présentes sur le terminal)

Méthodes

Ces serveurs peuvent définir les méthodes suivantes :
`insert, delete, update, query`

Exemple

L'activité *contacts* est associée à un serveur de contenu qui lui donne accès aux contacts de l'appareil

Sécurité entre applications

Principes

Chaque application est exécuté dans son propre processus
Chaque application définit un utilisateur unique (au sens Unix)
Sauf décision du développeur les données d'une application ne sont accessible qu'à elle-même

Mécanisme de permissions

Permissions

Tout application doit exprimer précisément les éléments de la plateforme auxquels elle souhaite avoir accès
Ces accès sont réalisés au travers de l'API Dalvik

Exemple

Fonctions Photos
Fonctions GPS
Fonctions Bluetooth
Fonction téléphonie
Fonctions SMS/MMS
Accès réseau

Développement Android (SDK)

Trois volets (Documentation : `http://developer.android.com`)

Communication – Expérimentation

Échange avec la plate-forme (adb)
Émulateur Android

Développement système

Outils de compilation croisée
Sources du système

Développement applicatif

Outil de compilation Dalvik Ant
Plugin Eclipse
NDK : développement natif (C, C++, ...), destiné à être embarqué dans une application classique

Communication – Expérimentation

Communication avec l'appareil : adb

L'outil adb fourni avec le SDK permet de communiquer avec l'appareil Android, par exemple

- placer des données
- installer une application
- exécuter un shell

L'émulateur

L'émulateur permet de faire fonctionner une image Android sur un ordinateur, par exemple, tester :

- une image (rom) modifiée
- une application

Il est possible d'échanger avec l'émulateur avec adb

Développement applicatif (Java)

Quelques principes

Les applications sont guidées par l'interface
Il est possible de spécifier l'interface avec un fichier XML
Les applications définissent un manifeste (comportant entre autre les éléments de l'API dont elles ont besoin)

Systèmes de paquets `apk`

Une fois l'application codée, elle est compilée et placée dans un fichier `apk`. Il contient le code et les données de l'application

Signature (cryptographique)

Pour être installée une application doit être signée
Toute mise à jour nécessite une vérification de la signature

Organisation d'une application

Différents dossiers

Un dossier par application – Contient :

- Le fichier `AndroidManifest.xml`: décrit l'application et les éléments qui la compose
- Dossiers :
  - `bin/` : fichiers générés + l'appli elle-même
  - `libs/` : bibliothèques tierces
  - `res/` : ressources (icônes, fichiers xml importants)
  - `src/` : fichiers sources java

D'autres dossiers peuvent apparaîtrent mais sont moins importants

Le dossier `res/`

- `drawable/` : images/icônes
- `layout/` : fichiers xml
- `menu/` : fichiers xml
- `values/` : fichiers xml

Principe généraux

Principe MVC

Trois blocs indépendants:

- Modèle (moteur)
- Vue (apparence)
- Contrôleur (connexion entre le modèle et la vue)

Souvent la vue et le contrôleur se confondent
Ex. : la visualisation du bouton et le contrôle exercé par le bouton = le bouton

Utilisation des fichiers xml

Les fichiers xml peuvent être utilisés pour définir l'interface (vue/contrôle) d'une application
Eclipse permet de manipuler les fichiers xml de façon simple

Fichiers xml (dans `res/`)

`AndroidManifest.xml`

Fichier principal

Définit l'activité principale – Les autres activités

Définit les permissions utilisées

`layout/`

`activity_main.xml`: disposition de l'activité principale

`menu/`

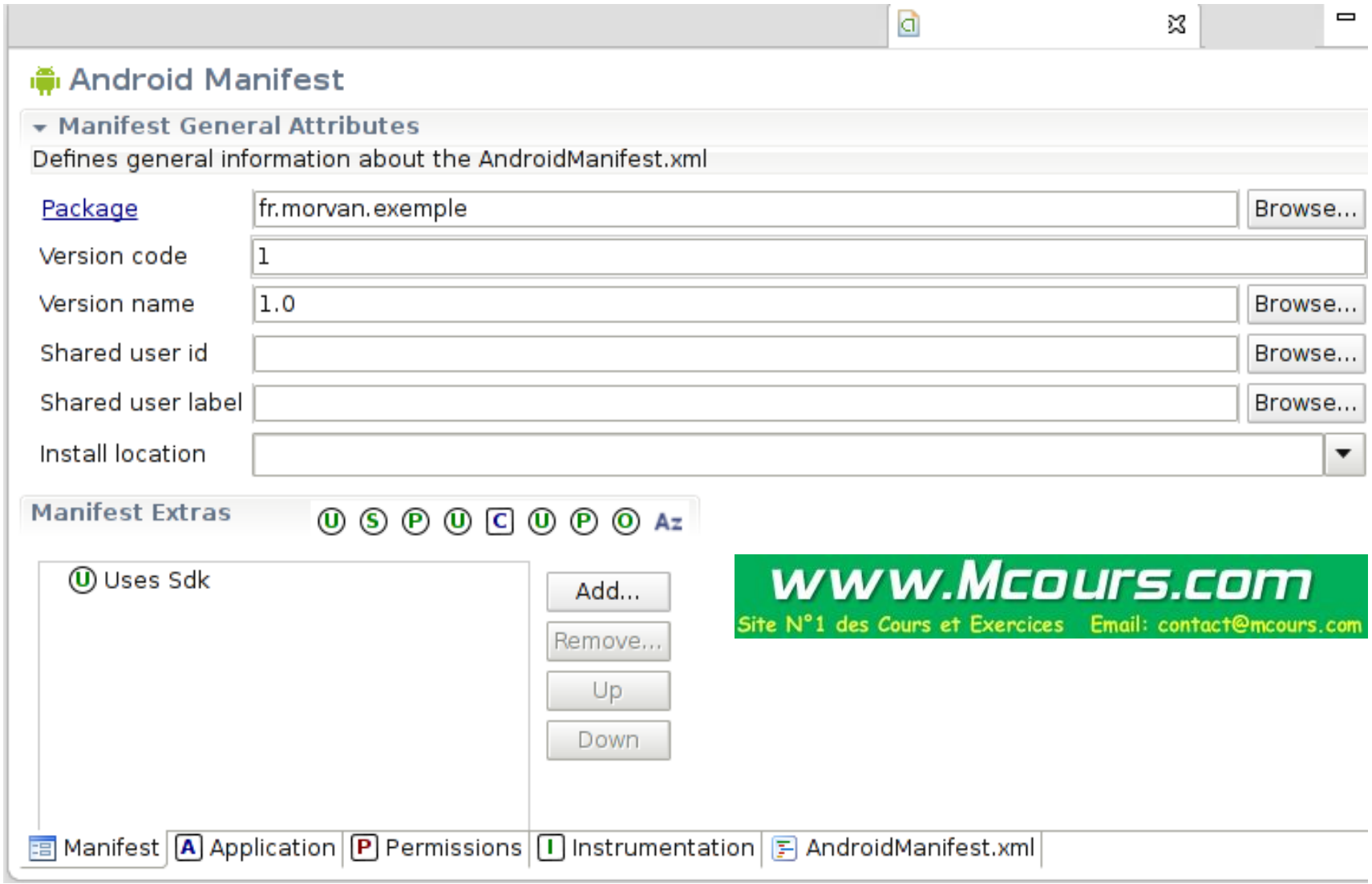
`activity_main.xml`: menu de l'activité principale

`values/`

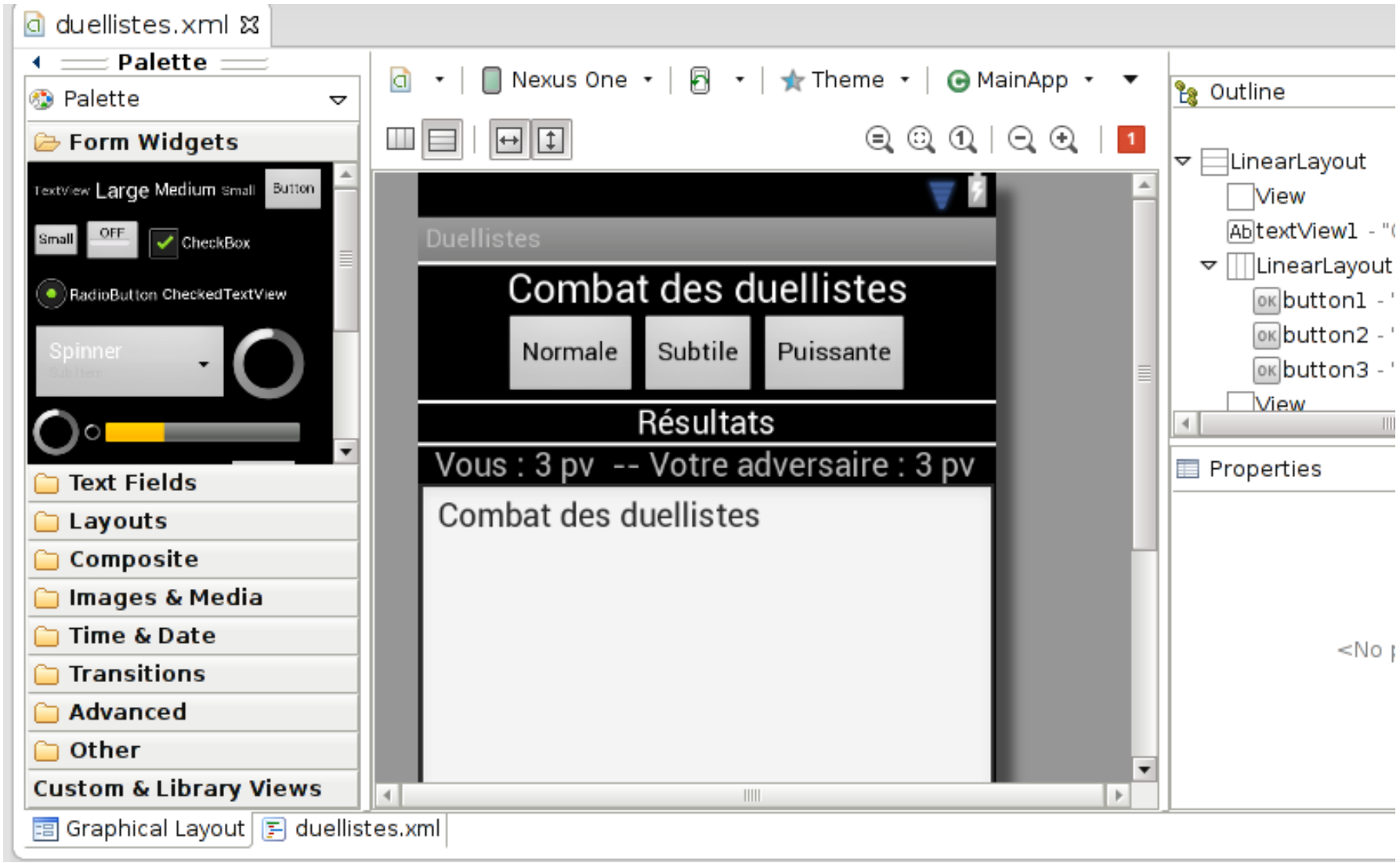
`strings.xml`: chaînes de caractères

Fichiers de traduction

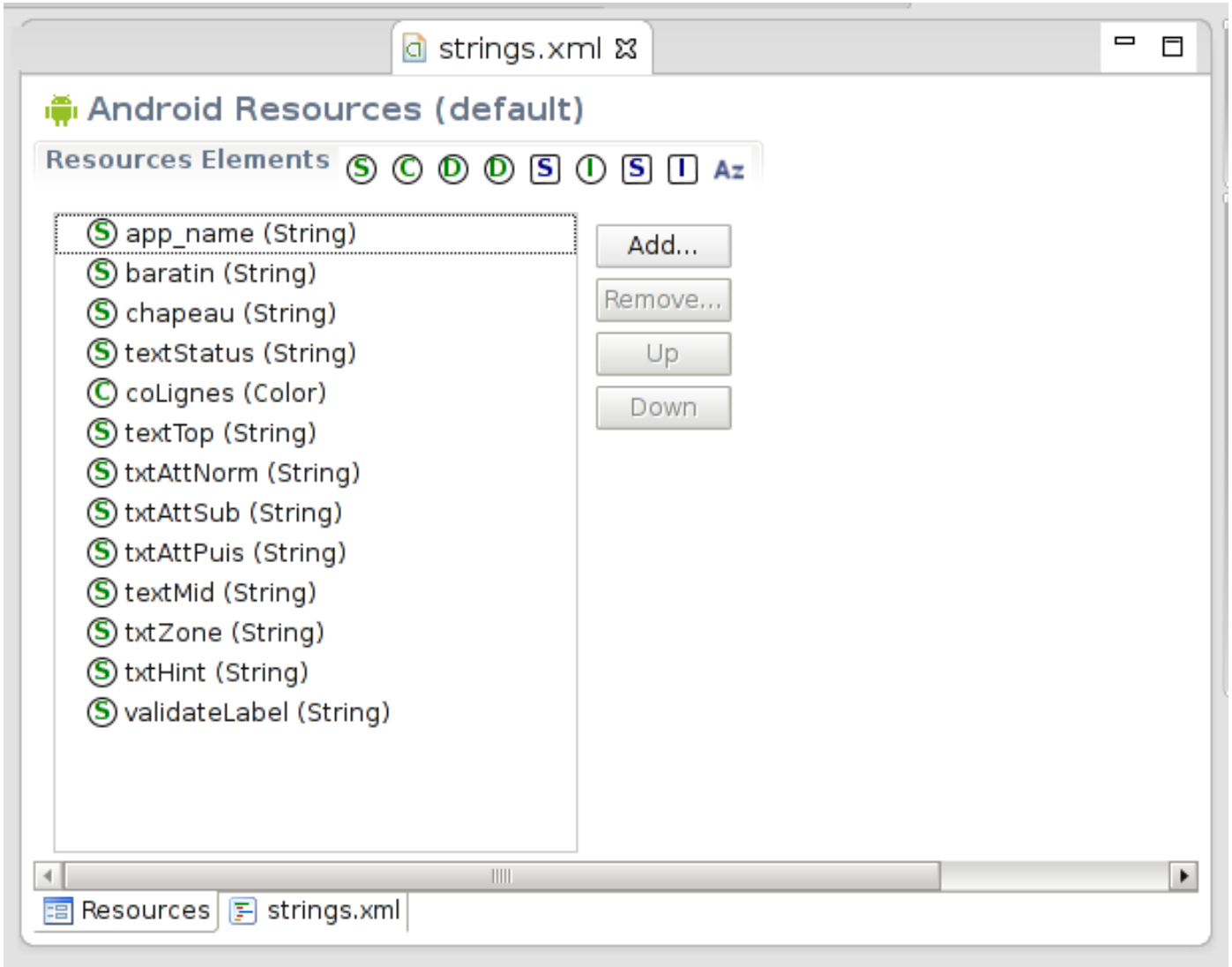
Vue Eclipse : AndroidManifest.xml



Vue Eclipse : layout.xml



Vue Eclipse : strings.xml



Classe principale : l'activité

Principe

Au moins une activité qui hérite de la classe `Activity`
Implémente au moins la méthode `onCreate`

Programme exemple

```
public class MainApp extends Activity {
    /*
     * A la création de l'activité
     */
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // choix de la disposition -> nom du fichier xml
    }
    // Autre code...
}
```

Widgets – Vues

Principe

Les éléments graphiques sont les *widgets* (dans Android *view*)
(Boutons, Menu, Zone de texte, ...)
Leur disposition est décrite dans un fichier xml dans `layout/`

Programme exemple

```
// La méthode findViewById permet d'obtenir un objet décrit
// dans un fichier de disposition (dans layout/)

Button monBouton = (Button) findViewById(R.id.bouton));
TextView txtNom = (TextView) findViewById(R.id.tNom);
EditText zoneTexte= (EditText) findViewById(R.id.zTexte);

zoneNom.setText("François");
zoneTxt.append("Texte ajouté");
```

Les auditeurs – 1

Principe

Pour les interactions avec l'utilisateur on utilise un mécanisme d'auditeurs: *listener*
Il sont attachés à des vues
Ceux-ci permettent faire le lien avec le moteur de l'application

Programme exemple

```
monBouton.setOnClickListener
  (new OnClickListener() {// crée une classe anonyme
       public void onClick(View v) {
           // Nécessaire car abstraite
           // Code exécuté "au clic"
           MonActivite.this.moteur.methode();
       }
   });
```

Les auditeurs – 2

Principe

Il est également possible de manipuler les actions de toucher

Programme exemple

```
texte.setOnTouchListener(new OnTouchListener(){
   @Override
   public boolean onTouch(View v, MotionEvent event) {
      valx=event.getX();
      valy=event.getY();
      String txtUpdate = "X : "+valx+" Y : "+valy;
      texte.setText(txtUpdate);
      return true;
   }});
```

Aller plus loin ?

Documentation officielle

- `http://developer.android.com/` : Documentation pour développeur
- `http://developer.android.com/reference/` : Référence de l'API (y compris doc spécifique API Java)

Bibliographie

- Learning Android (Marko Gargenta) – O'Reilly
- Hackez Google Android (Frédéric Brault, 70 pages) – Eyrolles
- Warescription `http://commonsware.com/warescription`

Vidéos

O'Reilly :Android Open Conference
`http://www.youtube.com/user/OreillyMedia/videos`