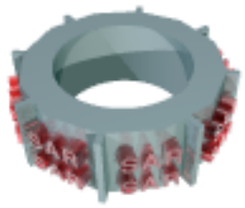
Cours n° 7 :
Programmation sur Plateformes Mobiles: application à iOS et Android

Intervenants :

- **RENAULT Etienne (LRDE/LIP6)**
- **BAKIRAS Harris (LIP6)**

Généralités



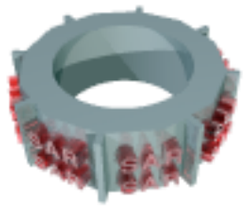
Généralités

Pré-requis

- Langage JAVA
- Pattern MVC / Observer
- XML, Sax, Dom

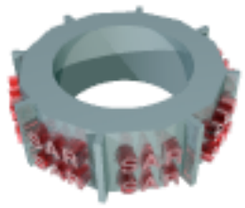
Logiciel

- Eclipse IDE - Plugin ADP - SDK Android
- Telnet pour communiquer avec le simulateur



Développement : IDE

- **Eclipse (ADT) / NetBeans (NBandroid)**
  - **A**ndroid **D**eveloppement **T**ool développé par « Google »
- **Fonctionnalités**
  - Packaging automatique (*.apk équivalent *.ipa)
  - Construction graphique de la GUI
  - Lien direct avec le simulateur / mobile
    - **Packaging et déploiement « on click »**
    - **Mode debug, mode pas à pas ...**
    - **Tests unitaires**
  - Outils de debugage - **D**alvik **D**ebug **M**onitor **S**ystem (DDMS)
  - Gestion des signatures



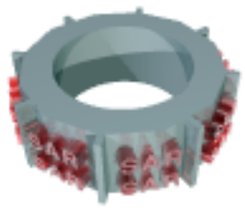
Introduction

Une histoire...

- Août 2005 - Google achète la Startup Android Inc
- Novembre 2007 - Consortium « Open Handset Alliance »
  - **« Standard et Norme » appareil mobile avec Android**
- Décembre 2008 - Android SDK 1.0 sur un T-Mobile G1
- Octobre 2010 - Android devient rentable pour Google
- Mars 2012 - Google Play
  - **Fusion du Android Market et de Google Music**

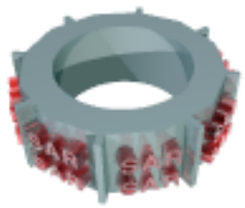
Un logo...

- Bugdroid
  - **Personnage d'un jeu d'Atari des années 1990 «Gauntlet: the third encounter».**



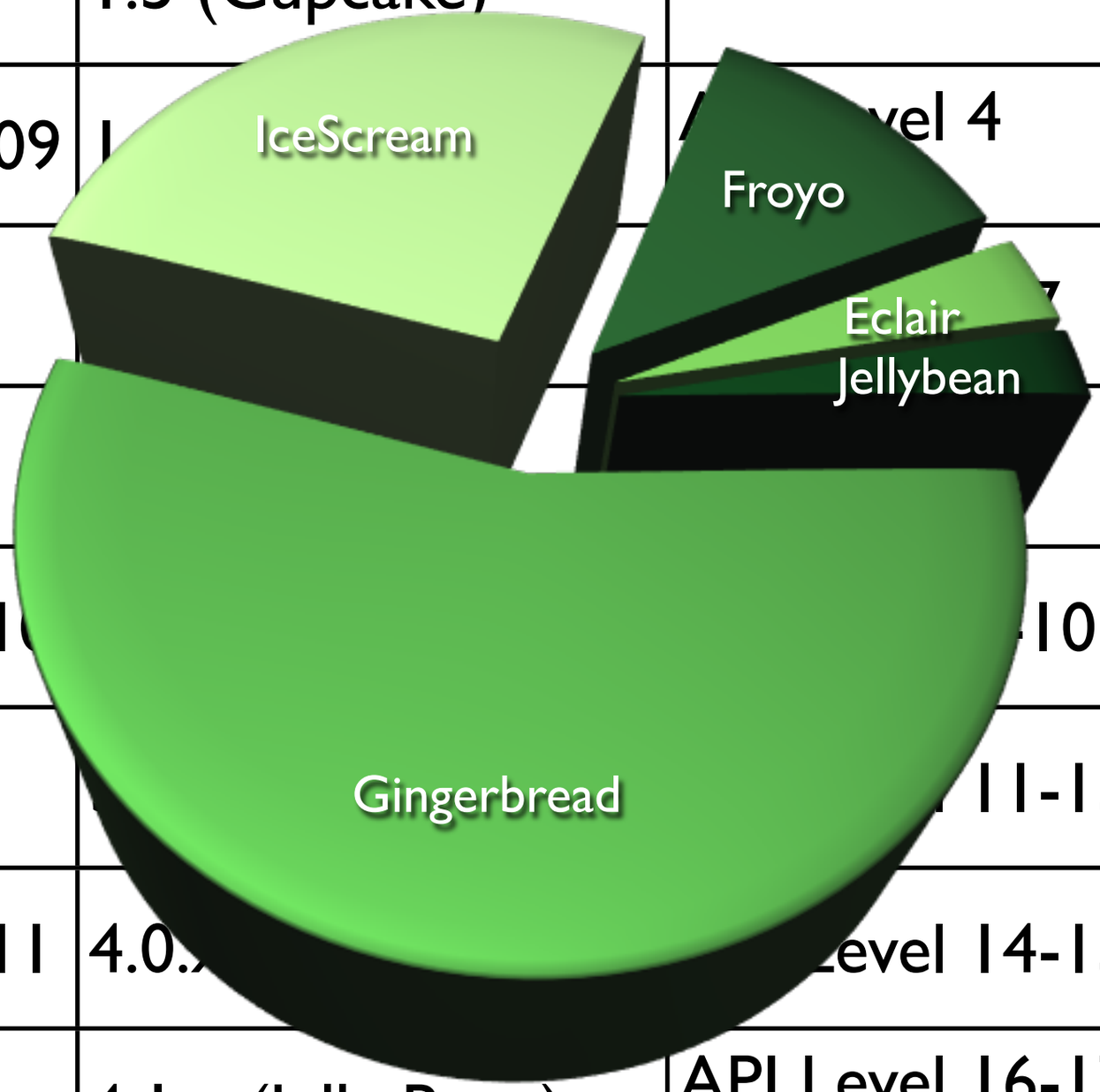
Android : versions

Avril 2009	1.5 (Cupcake)	API Level 3	0.4 %	
Septembre 2009	1.6 (Donut)	API Level 4		
Octobre 2009	2.0/2.1 (Eclair)	API Level 5-7	3.5 %	
Mai 2010	2.2.x (Froyo)	API Level 8	12 %	
Décembre 2010	2.3.x (GingerBread)	API Level 9-10	54.2 %	
Juillet 2011	3.x (Honeycomb)	API Level 11-13	1.8 %	
Décembre 2011	4.0.x (Icescream)	API Level 14-15	25.8 %	
Juillet 2012	4.1.x (Jelly Bean)	API Level 16-17	2.7 %	



Android : versions

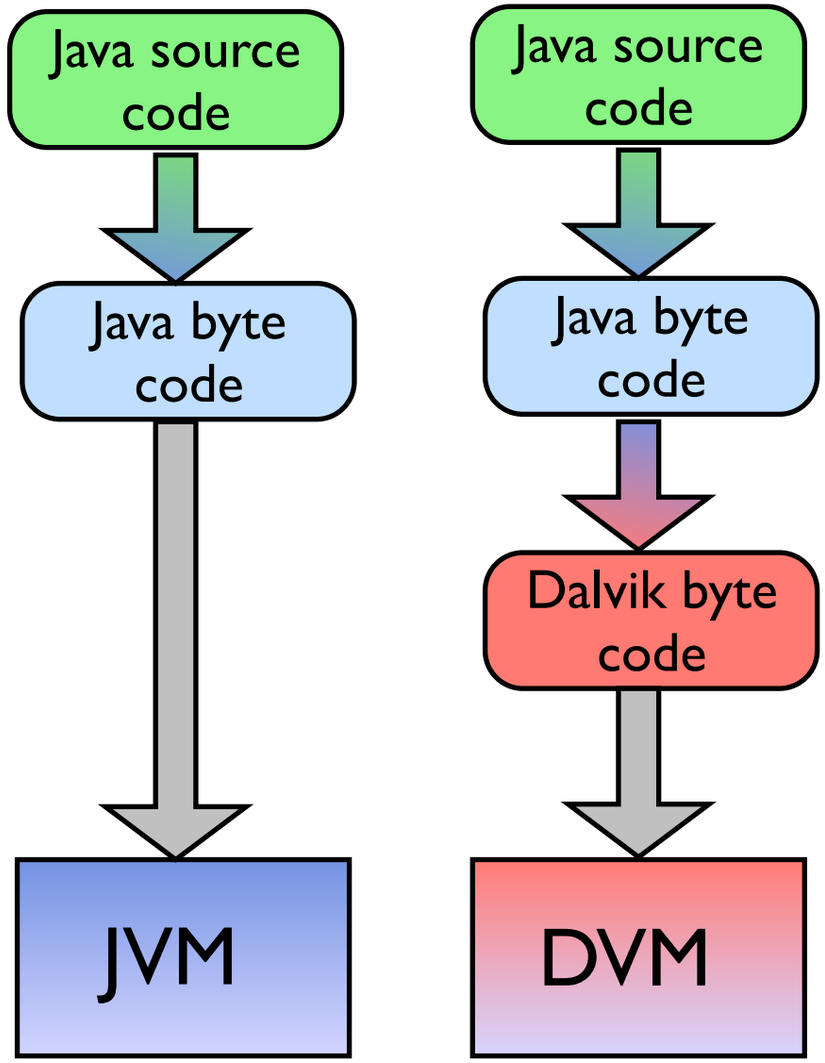
Avril 2009	1.5 (Cupcake)	API Level 3	0.4 %	
Septembre 2009	1	vel 4		
Octobre 2009			3.5 %	
Mai 2010			12 %	
Décembre 201		10	54.2 %	
Juillet 2011		11-13	1.8 %	
Décembre 2011	4.0.x	Level 14-15	25.8 %	
Juillet 2012	4.1.x (Jelly Bean)	API Level 16-17	2.7 %	



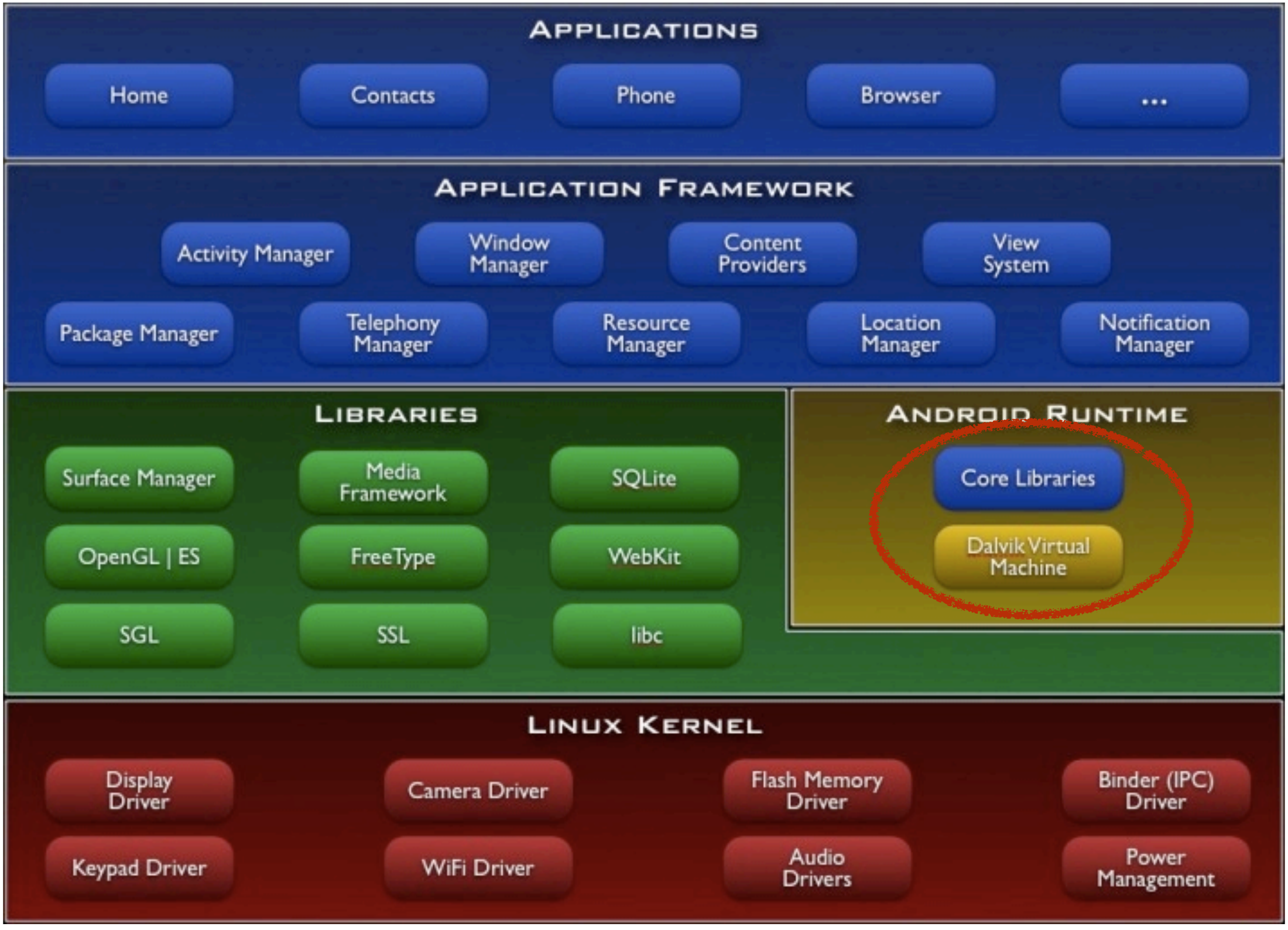
Basé sur Java

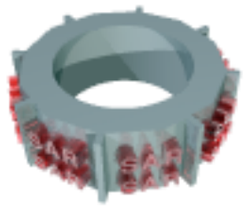
OS basé sur un noyau Linux

- JAVA - SDK Android
  - **SDK Android 4.0 basé sur noyau 3.0.1**
  - **VM : Dalvik Machine Virtuelle**
    - VM à registres illimités (non à pile comme la JVM)
    - Optimisation mémoire
    - Optimisation vitesse d'initialisation
    - Optimisation vitesse d'exécution (2x JVM classique)
    - Chaque processus Android possède sa propre VM
- Xml pour les interfaces



Android : architecture





Packaging : déploiement

◉ Packaging d'une application

- Archive : APK
  - « *.dex » : *.class
  - « res » : ressources de l'application (icônes, images, XML GUI, constantes ....)
  - « Android-manifest.xml » : gestion de l'exécution et des permissions
- L'archive doit être signée
  - Par défaut debug.keystore (Certificat valable pour 365 jours)
  - Générer son certificat avant la distribution
  - Recommandé d'utiliser un certificat pour toutes vos applications
  - Le certificat est utilisé pour la détection des mises à jours

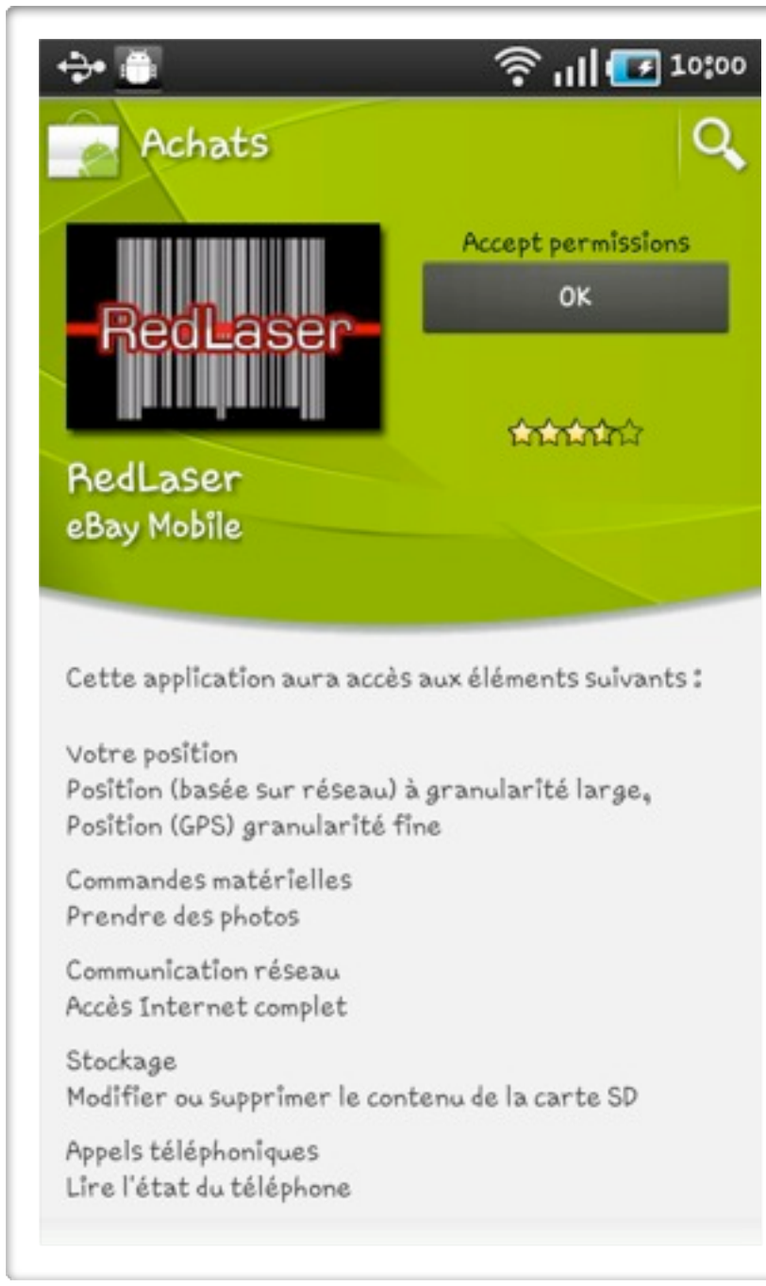
◉ Déploiement d'une application

- « Push and Click for Install »
  - Récupérer l'archive (USB, Bluetooth ....)
  - « Install On Click »

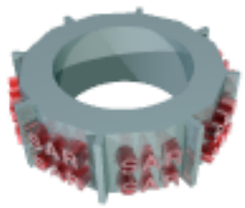
Permissions: Android-Manifest.xml

Spécifier les ressources utilisées

- Informer l'utilisateur
- Sécuriser le droit d'accès aux ressources
- Utiliser ce qui est nécessaire



INTERNET	Accès au réseau
READ_CONTACT/ WRITE_CONTACT	Accès à la BdD des Contacts
ACCESS_FINE_LOCATION	Accès au GPS
BLUETOOTH	Accès au réseau Bluetooth



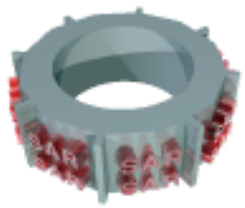
Distribution : sécurité

◉ Mettre à disposition l'archive

- Par un market ....
  - **Google, Orange, autres ...**
  - **Payant (25$ chez google)**
- En téléchargement libre/payant
- Monter votre serveur de distribution « aptoide »

◉ Installation « Copy And Install On Click »

- La sécurité ...
  - **API Google License Verification Library (Google play)**
  - **Sinon ...**



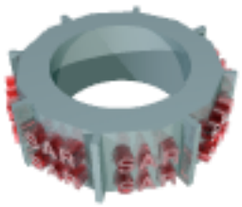
Simulateur / Mobile

Créer une instance VD

- SD Card (512Mb)
- Choix du matériel (Caméra, GPS, Touch ...)
- Skin : choix du mobile à simuler
  - **Galaxy, Nexus, etc.**

Paramètre du mobile

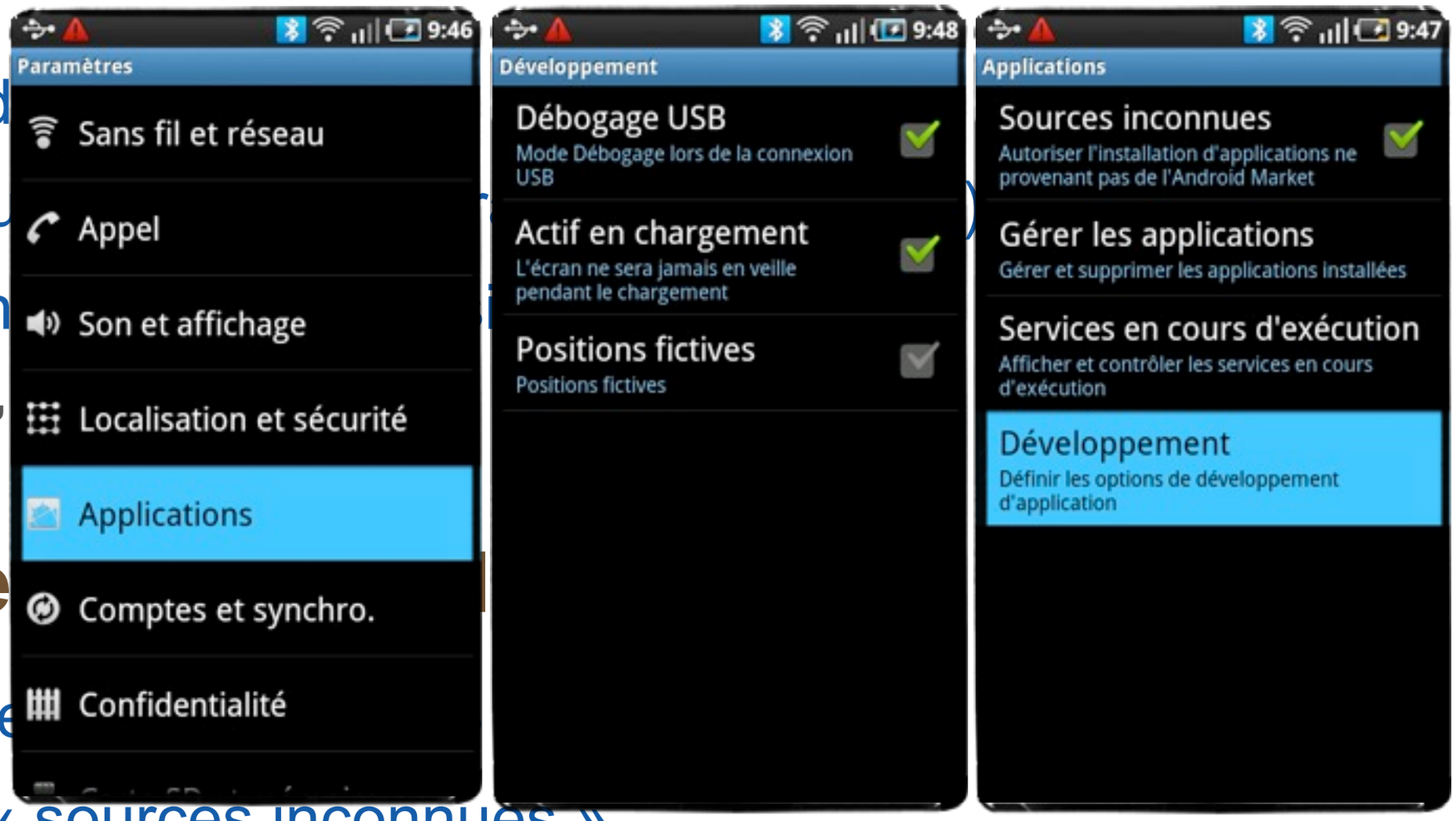
- Activer le Débugage USB
- Activer « sources inconnues »
- Type de connexion USB
- Connecter le mobile



Simulateur / Mobile

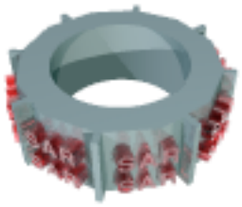
Créer une instance VD

- SD Card
- Choix du
- Skin : ch
  - Galaxy,



Paramè

- Activer le
- Activer « sources inconnues »
- Type de connexion USB
- Connecter le mobile



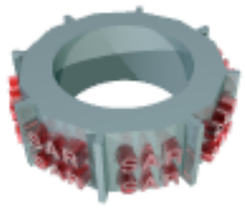
Le mobile type

Différence avec un iPhone

- 4 boutons
  - **HOME : revenir au manager**
  - **MENU : demander le menu d'options (pour l'application en cours)**
  - **ANNULER : fermer l'application en cours**
  - **RECHERCHER : faire une recherche**

- Chaque constructeur peut offrir une personnalisation
  - **Spécificités matérielles**
    - Track-Ball (HTC Desire), caméras avant et arrière, accéléromètre, etc.
  - **Diffusion lente des nouvelles versions**

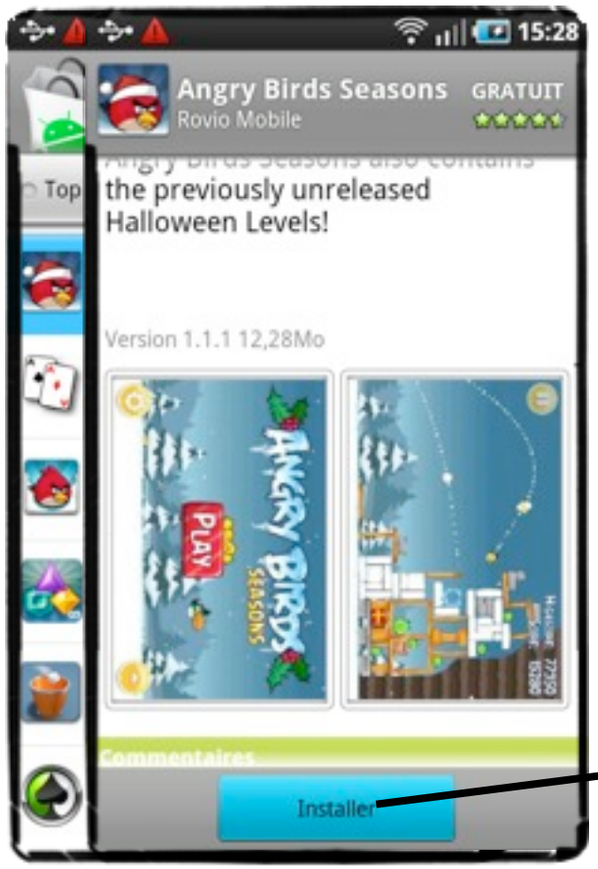
Fondamentaux



Philosophie Android

- **Favoriser la ré-utilisation d'applications**
  - Ne pas ré-inventer la roue ...
  - Ré - utiliser des morceaux d'application ...
- **Trois types de composants :**
  - Activity : Interface compréhensible pour un utilisateur
  - Fragment: Souplesse de manipulation des interfaces
  - Service : Tâche de fond (musique, serveur, synchronisation ...)
- **Faciliter la communication ...**
  - Intent : Bus de communication
  - BroadcastReceiver : Gestion des annonces (batterie, fin de téléchargement ...)

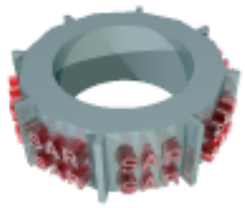
Exemple d'application



Activity

Service
Download

BR
download end



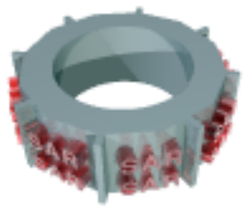
Exemple d'application

Point de vue du développeur

- Activity :
  - 1 Affichage de la liste des applications
  - 1 Affichage du détail d'une application
- BroadcastReceiver
  - Détecter la fin d'un téléchargement d'application
- Service :
  - 1 Service pour télécharger l'application
  - 1 Service pour installer l'application

Activity

- Le cycle de vie
- Les vues
- Les menus et les notifications
- Les évènements
- La gestion des activités et persistence



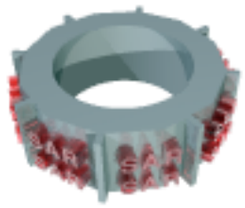
Activity : définition

Elément visuel principal d'une application

- Une entité visuelle complète pour un « utilisateur » : un écran
- Divisée en deux parties distinctes :
  - **La vue : ensemble des éléments graphiques à l'écran**
  - **La gestion de la vue et des événements: partie Contrôleur du MVC**
- Une Application est un ensemble d'activités

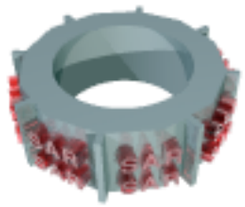
1 Thread UI :

- Gestion du rafraîchissement
- Gestion des événements
- SEUL LE THREAD UI PEUT MODIFIER LES ELEMENTS GRAPHIQUES

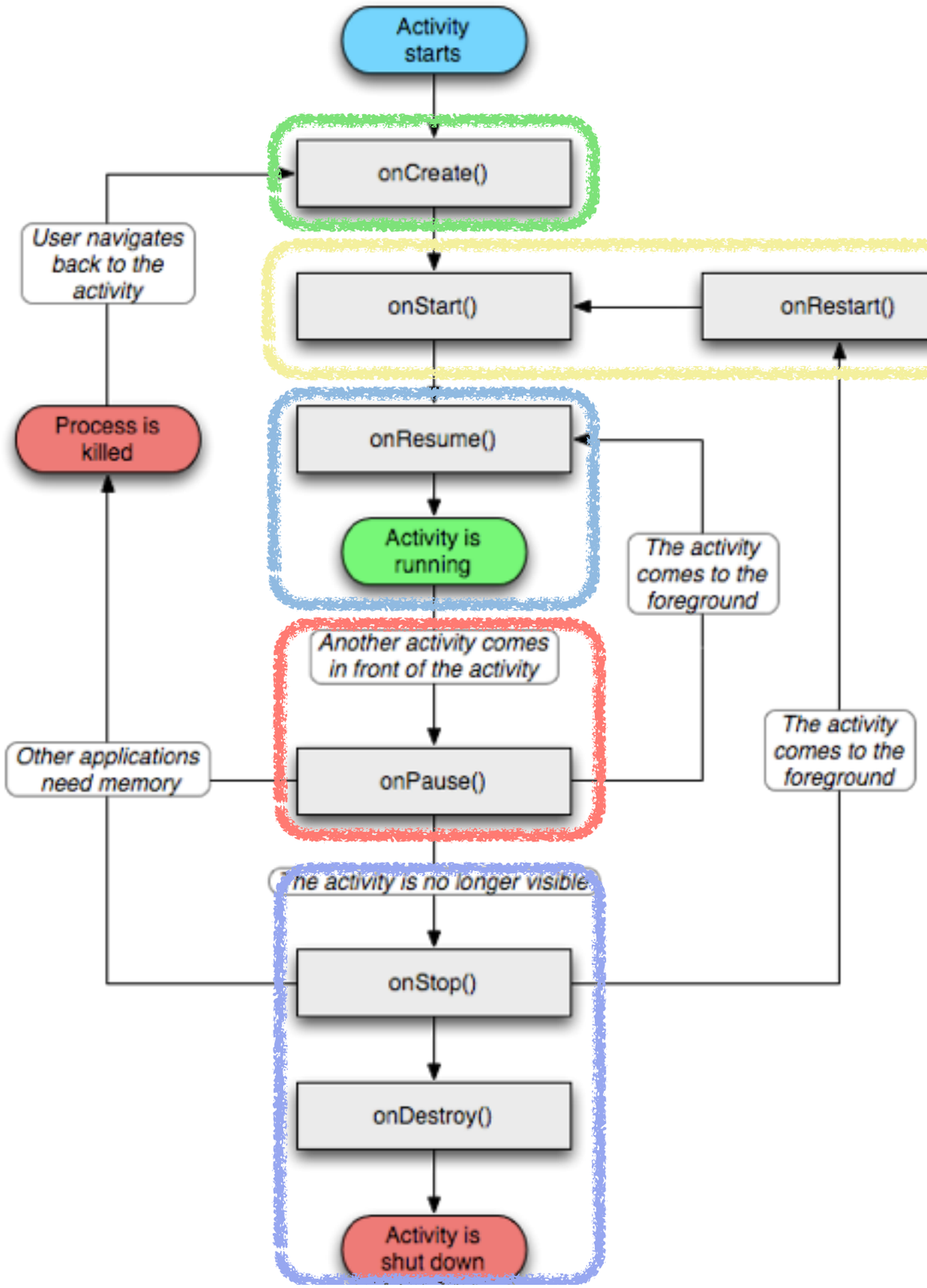


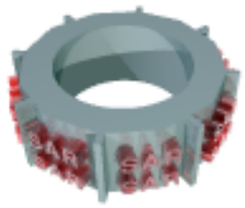
Description du cycle de vie

Opération	Description	Visible	Killable
onCreate	Instanciation + binding evt	no	no
onStart onRestart	Initialisation / Réinitialisation Chargement données persistantes Redémarrer les capteurs	no	no
onResume	Affichage du GUI	yes	no
onPause	Sauvegarde - Persistance Stop animation ... Stop service (GPS , ...)	no	yes
onStop onDestroy	Libération des ressources (RAM) Destruction GUI Libération des ressources	no	yes



Cycle de vie d'une activité





Tracer le cycle de vie : code Java

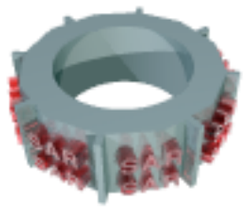
- Hériter de Activity (Surcharger les opérations)

```
public class TPHello extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    protected void onDestroy() {
     Log.i("TPHello","Destroy");
     super.onDestroy();
    }

    protected void onStart() {
     Log.i("TPHello","Restart");
     super.onRestart();
    }
...
}
```

Équivalent du system.out.println
Permet l'utilisation de LogCat

Activity

- Le cycle de vie
- Les vues
- Les menus et les notifications
- Les évènements
- La gestion des activités et persistence



Vue d'une activité

◉ 2 possibilités de conception :

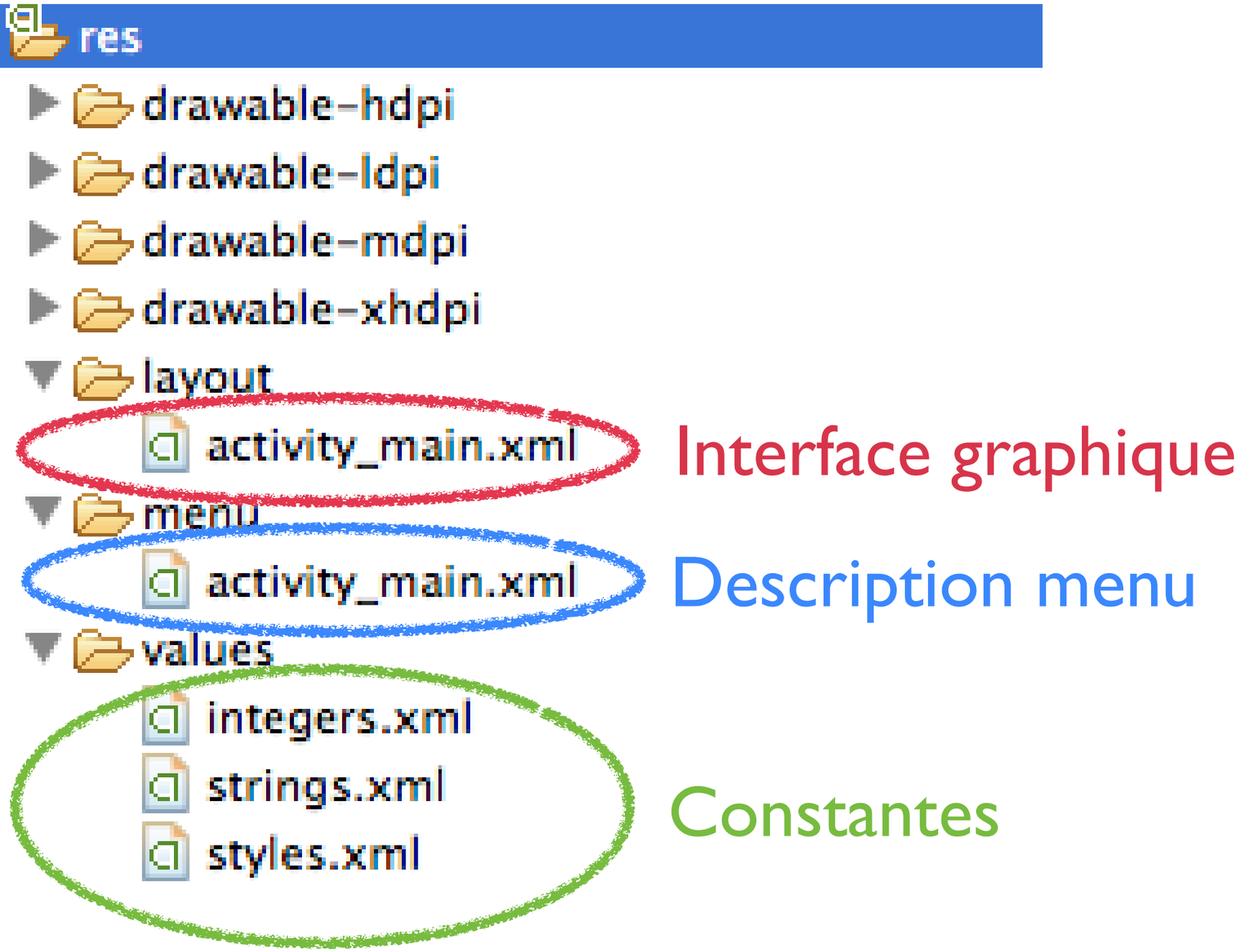
- Pure JAVA
- XML + JAVA : Isolation des vues !!!

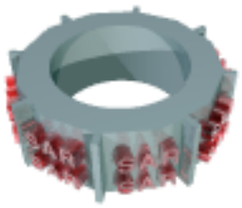
◉ Environnement :

- Répertoire RES
  - **Layout : définition des éléments des vues**
  - **Menu : définition des éléments de menus**
  - **Values: définition des constantes**
  - **etc.**

Vue d'une activité

- 2 pos
  - Pure
  - XML
- Envi
  - Répe
    - Lay
    - Me
    - Val
    - etc.





XML View

Une vue se caractérise par :

- Type (Balise XML) : TextView, Button, ImageView, EditView ...
- Un identifiant de l'objet dans la vue
  - **Défini par la balise XML :** `android:id="@+id/mon_id"`
  - **Généré dans la classe R**
  - **Accès via aux identifiants des objets via R.id**
- Un layout
  - **Accès aux identifiants des GUI XML via R.layout**

Instanciation et Création

- `setContentView(int layout_id)` : dans la méthode `onCreate()`

Lien entre XML et Objet JAVA

- `findViewById(int id)` : récupère l'objet java décrit par un XML

Liens entre les différents éléments

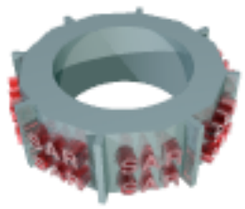
Génération automatique

Class R

Accès

Création / Instanciation

Source

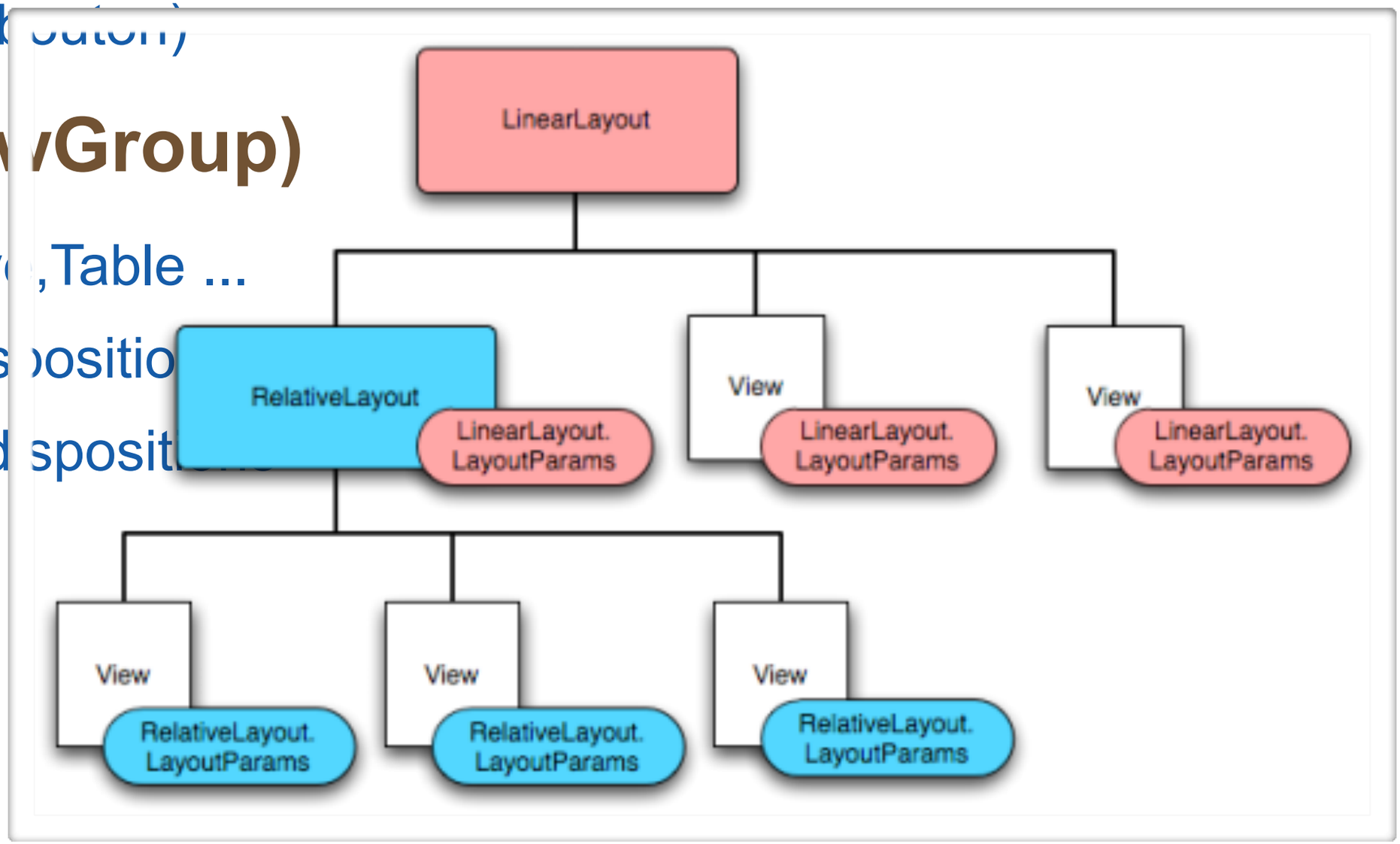


View : définition

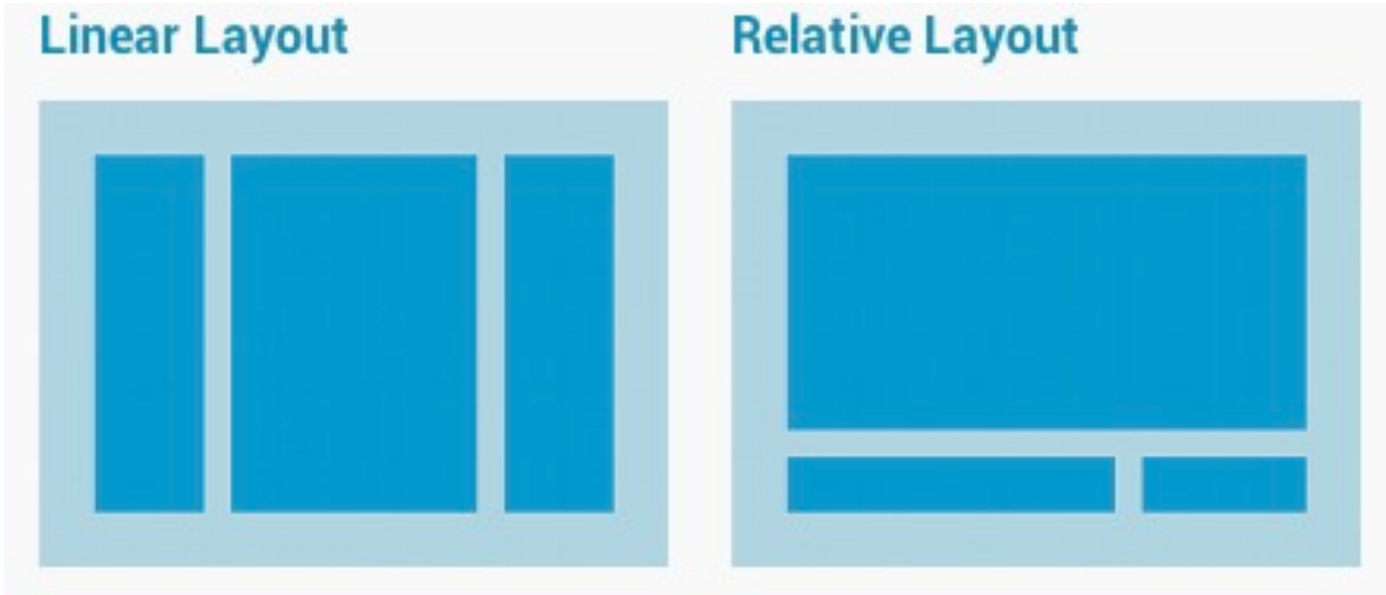
- **Hiérarchie des vues sous la forme d'un arbre**
  - ViewGroup : conteneur de vues (Layout, Grille, Liste)
  - View : feuille (bouton)
- **Layout (ViewGroup)**
  - Linear, Relative,Table ...
  - Spécifier la disposition des vues
  - Héritage des dispositions

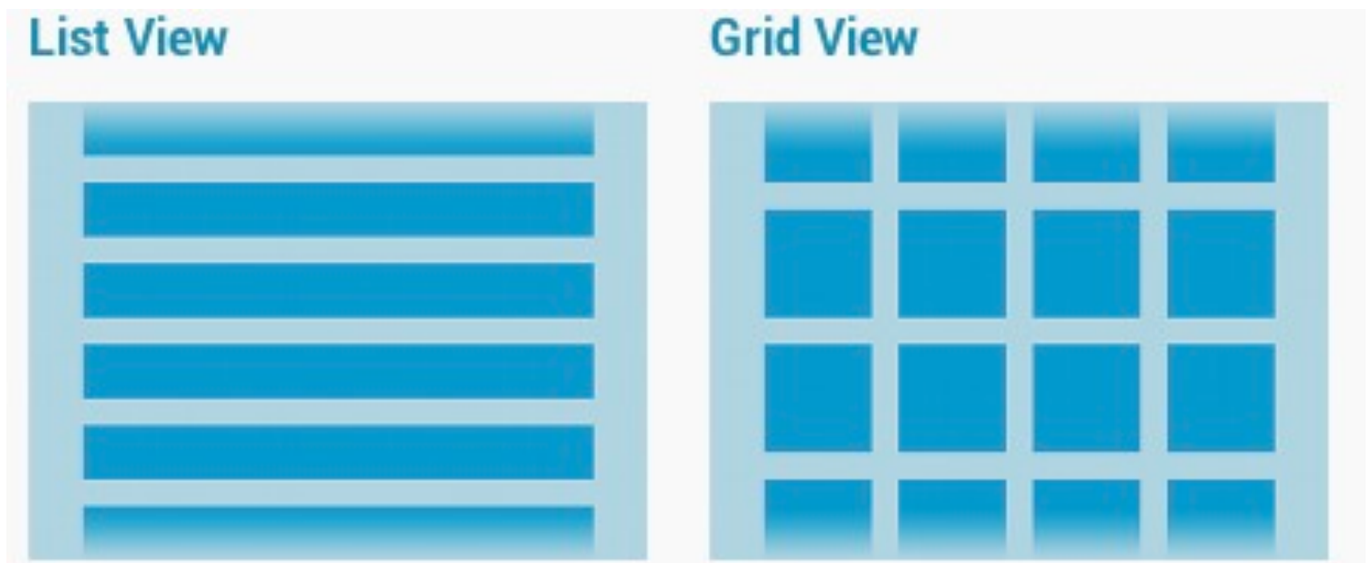
View : définition

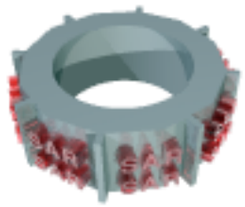
- **Hiérarchie des vues sous la forme d'un arbre**
  - ViewGroup : conteneur de vues (Layout, Grille, Liste)
  - View : feuille (bouton)
- **Layout (ViewGroup)**
  - Linear, Relative, Table ...
  - Spécifier la disposition
  - Héritage des dispositions



Layout : organisation des vues

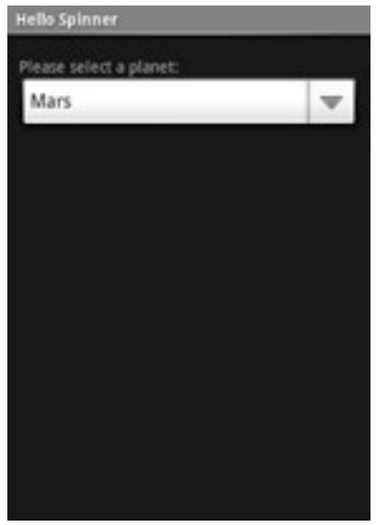


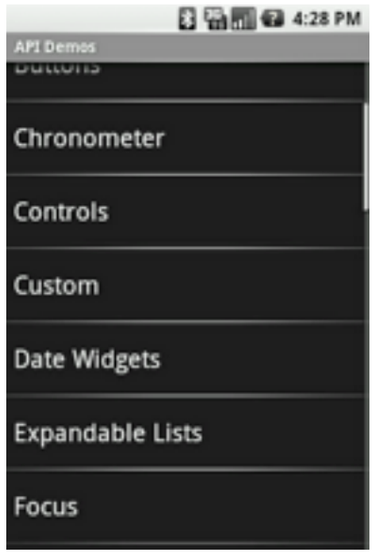


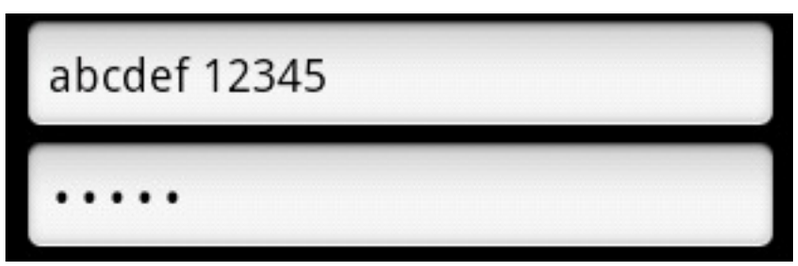


De nombreuses vues...

- AutoCompleteText
- Spinner
- CheckBox
- ListView
- RadioButton
- EditText
- ToggleButton
- ImageView
- ....

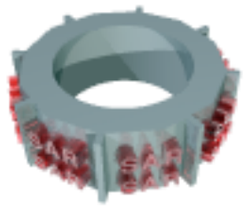






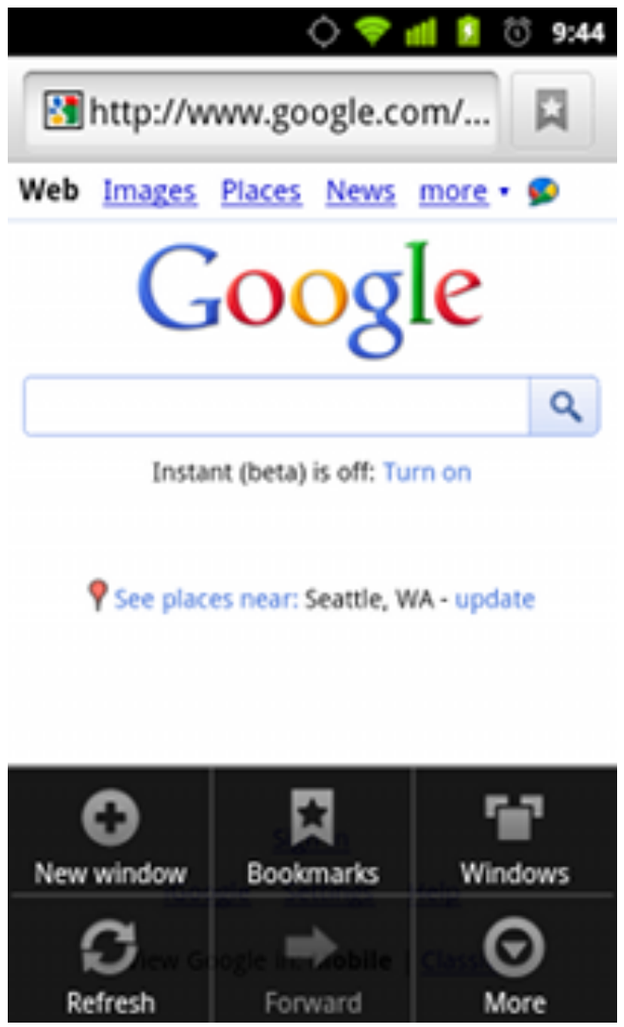
Activity

- Le cycle de vie
- Les vues
- Les menus et les notifications
- Les évènements
- La gestion des activités et persistence



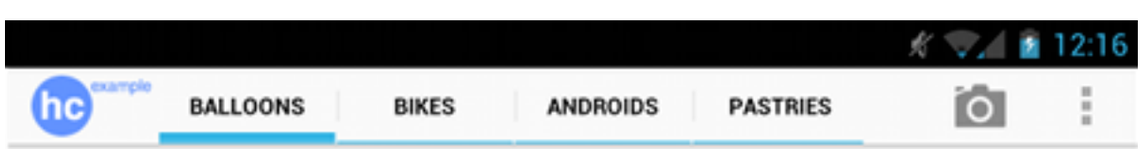
Options Menu

- OptionsMenu :
  - Icône + « MenuItem »
  - Pas de sous menu
  - Activable par pression du «Menu button»
  - 1 Menu / Activity



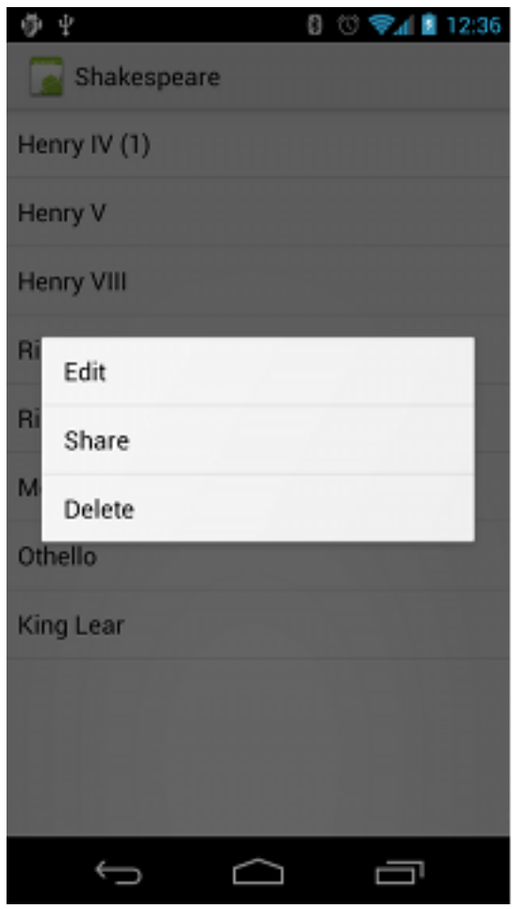
- Jusque Android 2.3.x :
  - Pas plus de 6 items, les autres sont passé dans «more»
  - Apparition dans le bas de la fenêtre

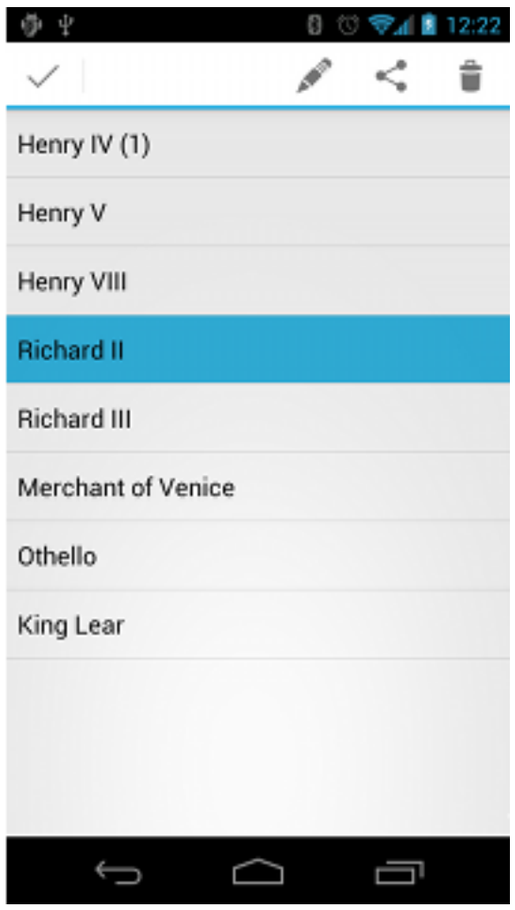
- A partir d'Android 3.x :
  - Utilisation d'un espace dédié «Action Bar»
  - Possibilité de définir son propre layout

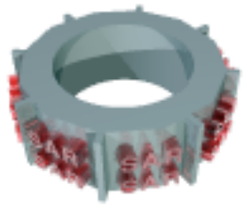


Context Menu

- Context Menu :
  - Menu contextuel
  - Possibilité de sous menus
  - Activable par pression longue sur un élément
  - La vue doit supporter le menu contextuel
    - `registerForContextMenu()`
  - 2 types :
    - floating context menu
    - contextual action mode
  - **Plusieurs Context Menu / Activity**







Menus : cycles de vie

EVT : First Push Menu

onCreateOptionsMenu

onPrepareOptionsMenu

EVT : Push Menu

*

onOptionsMenuClosed/
onOptionsItemSelected

onCreateContextMenu

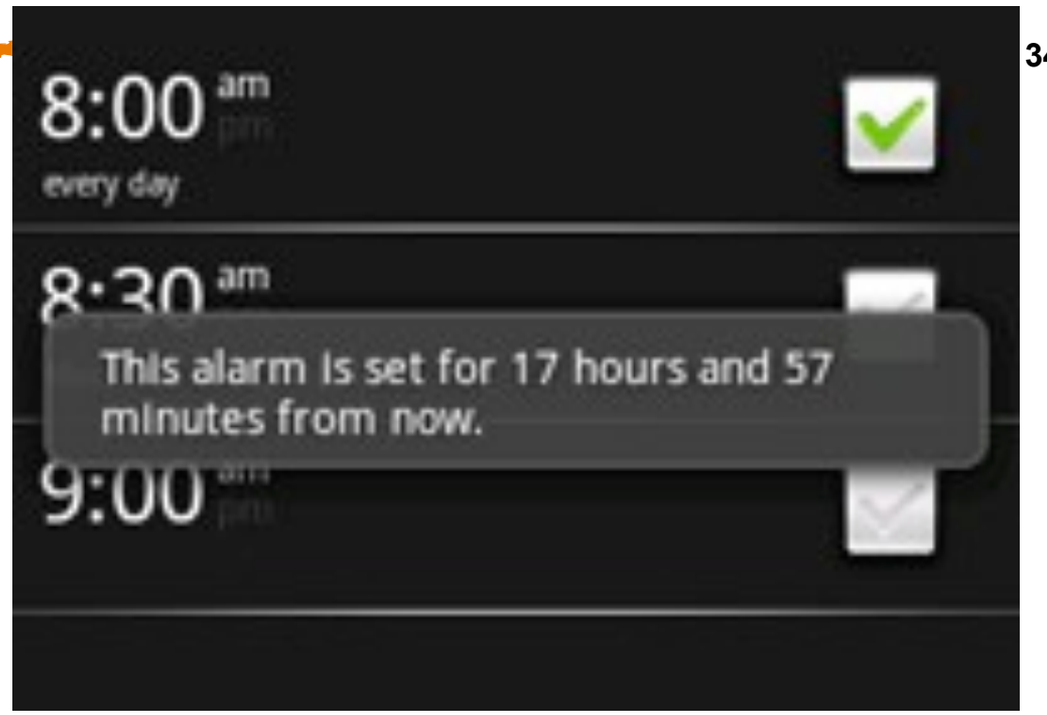
EVT : Push Menu

*

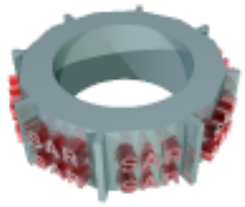
onContextMenuClosed /
OnContextItemSelected

* : Annuler - Sélectionner Item

Notifications

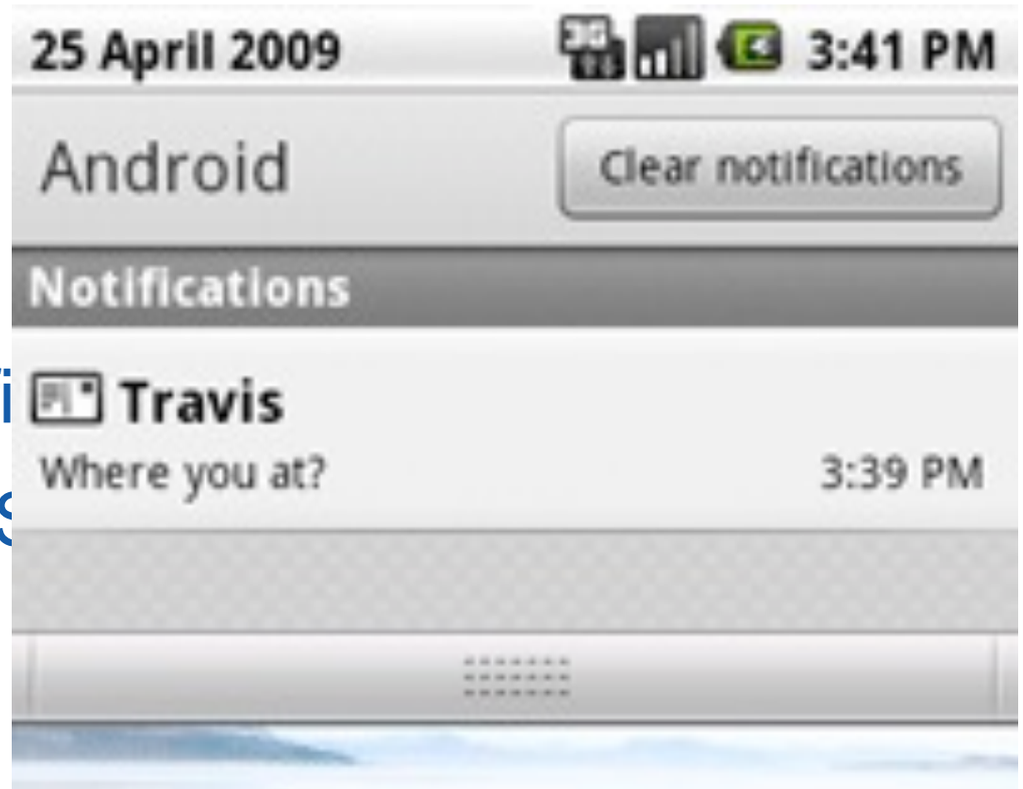


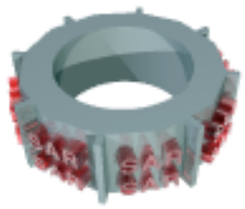
- **Toast Notification**
  - Message immédiat (bulle)
  - Simple d'utilisation
- **Status Bar Notification**
  - Message + icône (barre des notifications)
  - Utiliser dans les tâches de fond (Service)
- **Dialog Notification**
  - Boîte de dialogue ...
  - AlertDialog, ProgressBar ...



Notifications

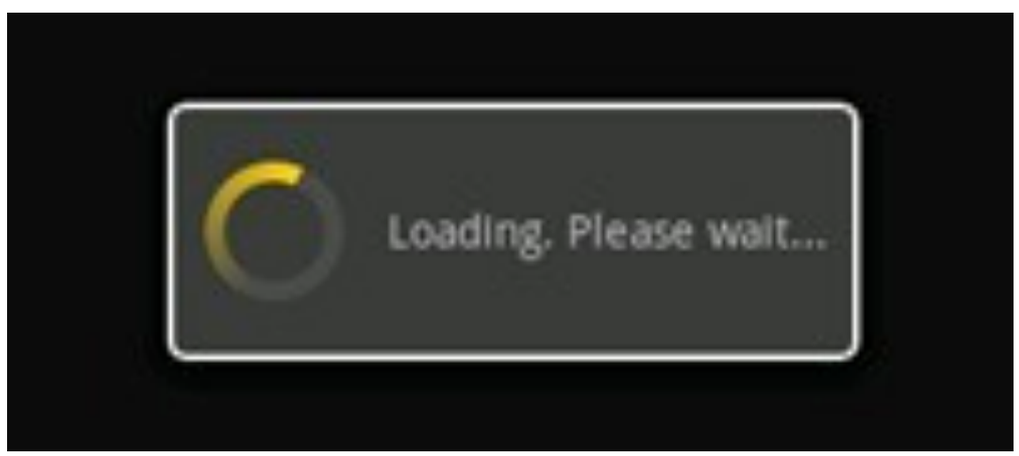
- **Toast Notification**
  - Message immédiat (bulle)
  - Simple d'utilisation
- **Status Bar Notification**
  - Message + icône (barre des notifi
  - Utiliser dans les tâches de fond (S
- **Dialog Notification**
  - Boîte de dialogue ...
  - AlertDialog, ProgressBar ...

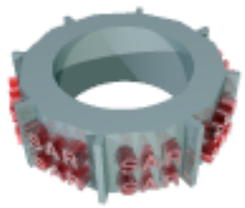




Notifications

- **Toast Notification**
  - Message immédiat (bulle)
  - Simple d'utilisation
- **Status Bar Notification**
  - Message + icône (barre des notifications)
  - Utiliser dans les tâches de fond (Service)
- **Dialog Notification**
  - Boîte de dialogue ...
  - AlertDialog, ProgressBar ...





Notifications : exemples

- Utilisation d'un ProgressDialog :

```
ProgressDialog dialog;

dialog = ProgressDialog.show(getContext()
           , "Working in progress"
           , "Wait for answer from website"
           , true
           , true
           , this);
dialog.show();
dialog.dismiss();
```

- Utilisation d'un Toast

```
Toast.makeText(this, "Hello", Toast.LENGTH_SHORT).show()
```

Activity

- Le cycle de vie
- Les vues
- Les menus et les notifications
- Les évènements
- La gestion des activités et persistence

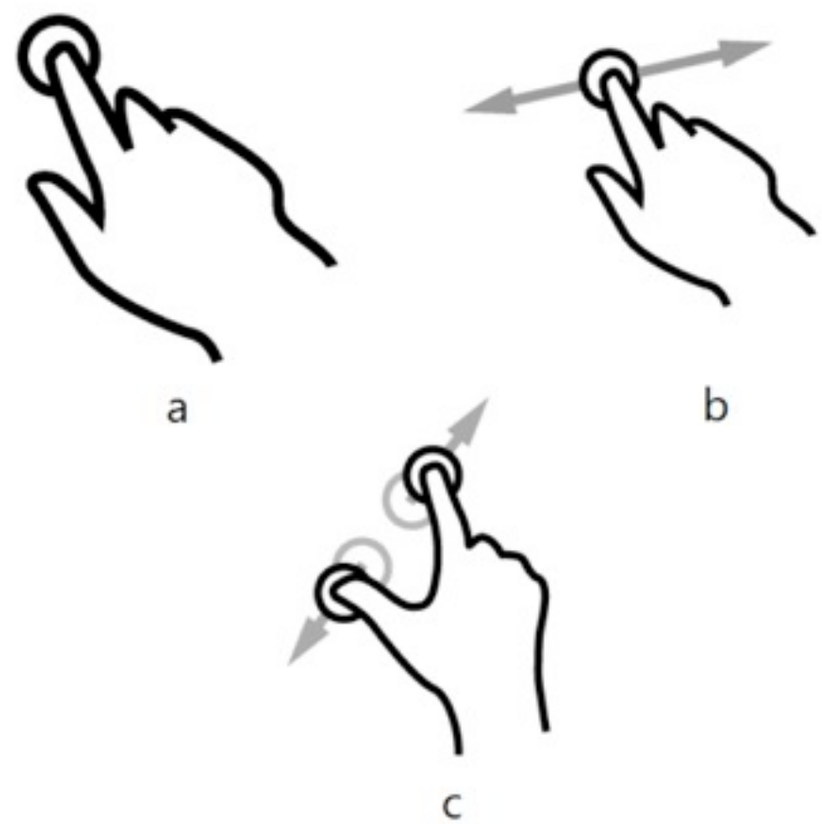
Evénements : Touch

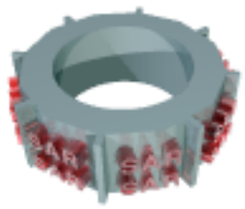
◉ Et le multi-touch ?

- Multi - Touch implémenté depuis Android 2.0 / 2.1 (Eclair)
  - **Ne pouvait gérer que 3 « Touch » simultanément**
  - **Depuis Android 2.2 (Froyo), plus de limite**

- Action d'un « Touch » (down/move/up)

- Gestions avancées ?
  - **(a) Tap (Click, LongClick)**
  - **(b) Drag and Drop (3.x)**
  - **(c) Zoom (Froyo)**
  - **Développer son API ...**





Gestes Multitouch : apperçu rapide

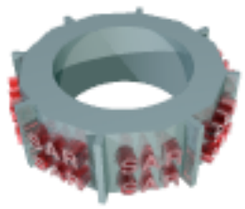
Actions prédéfinies

- `OnGestureListener:` événements prédéfinis
- `ScaleGestureDetector:` gestion du zoom
- `View.OnDragEventListenner:` gestion du drag and drop

Pas de Shake Motion (mais de nombreux exemples d'implémentations sur le net)

Multitouch pour traitements particuliers

- Notion de `action_pointer_down` et `action_pointer_up`
  - **Le multitouch est plus complexe à gérer que sous iOS**

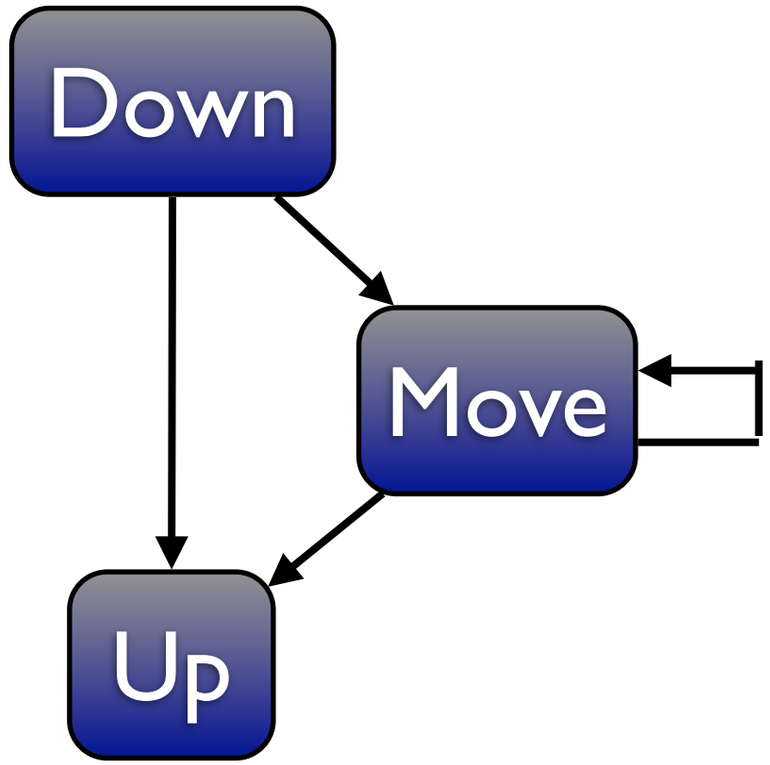


Touch : cycle de vie

- **Utilisation via**

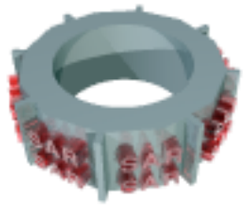
```
public boolean onTouch(View v, MotionEvent event);
```

  - Cycle de vie d'un « Touch »



- **MotionEvent**

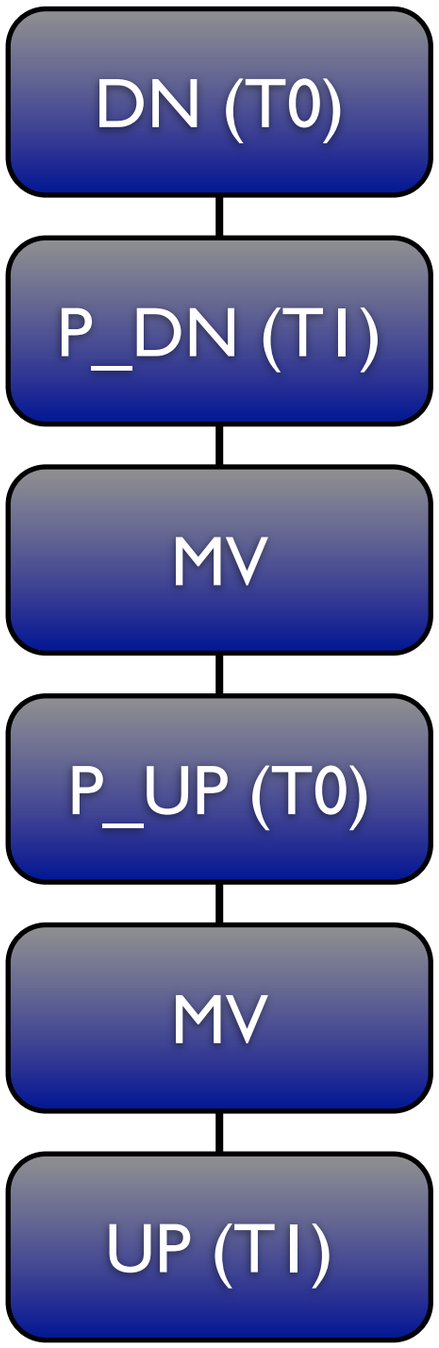
  - Vecteur de « Touch »
    - **Nombre de « Touch »**
    - **Actions**
    - **Position (x,y)**
    - **Temps**
    - **Historiques ...**
  - Les événements sont « recyclables »
    - **Ne pas sauvegarder les instances directement**

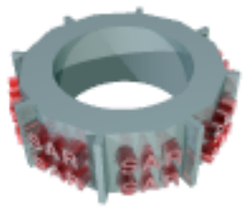


Multitouch

◉ Down / Up != Down Pointer / Up Pointer

- DOWN / UP
  - Dans le cas du MONO-TOUCH
  - SI `event.getPointerCount() <= 1`
- DOWN / UP POINTER
  - Dans le cas du MULTI-TOUCH
  - SI `event.getPointerCount() >= 2`
- Exemple
  - Premier touch Down (T0)
  - Deuxième touch Down (T1)
  - Déplacement de (T1)
  - Touch T0 UP
  - Déplacement de (T1)
  - Touch T1 UP





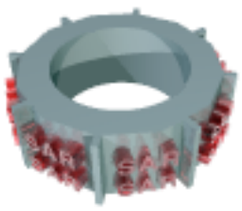
Les autres événements

Événements de « View »

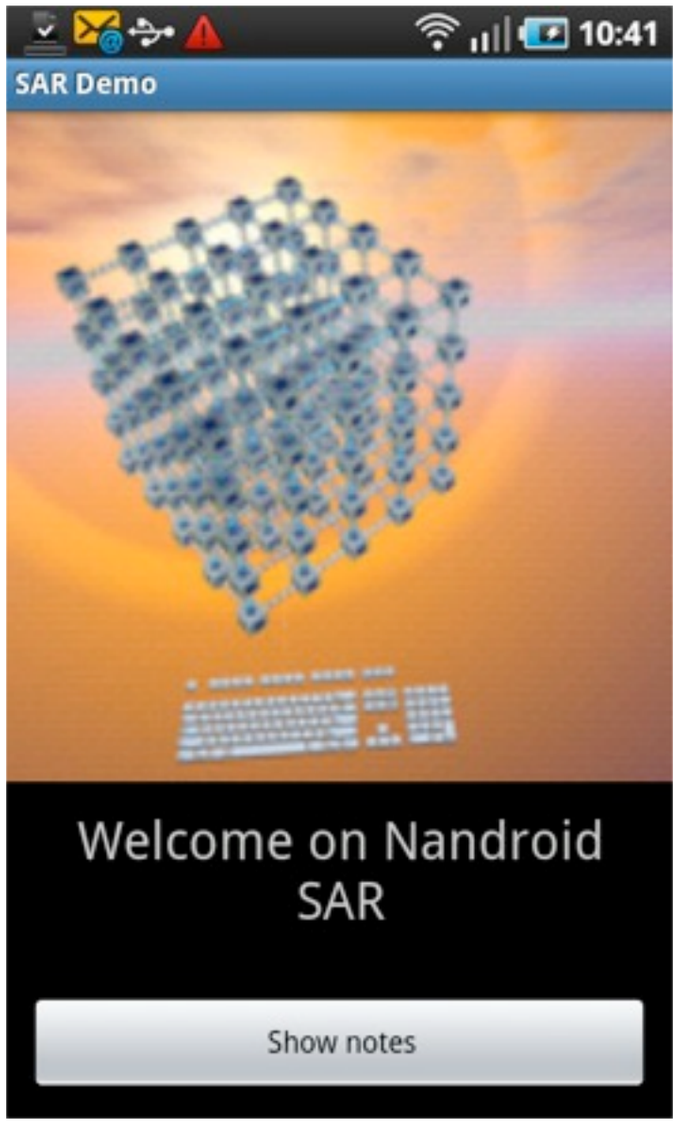
- `onClick:` au clic sur un item (Button ...)
- `onLongClick:` au maintient d'un clic durant un certain temps
- `onCreateContextMenu:` création du contexte menu après un LongClick (ListView)
- `onFocusChange:` lorsque la sélection change (avec le trac-ball)
- `onKey:` au clic sur un bouton du clavier
- `onTouch:` à la détection d'une « gesture » UP/DOWN/MOVE

Recevoir les événements :

- Pattern Observer : implémenter l'interface
- Récupérer la vue
- S'abonner à l'événement



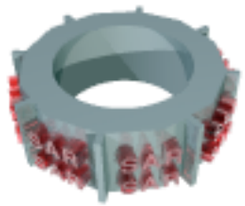
Evénements : exemple




```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    /* Bind the XML view */
    setContentView(R.layout.main);
    /* Link onClick operation */
    Button button =  findViewById(R.id.Button01);
    button.setOnClickListener(this);
}

public void onClick(View v){
  /* Launch new activity */
}
```

Activity

- Le cycle de vie
- Les vues
- Les menus et les notifications
- Les évènements
- La gestion des activités et persistence

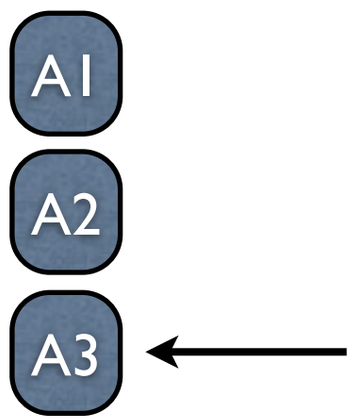


Gestion des Activités : liste d'Activités

- **Le système gère une liste d'Activités**
  - La tête de liste : « en cours d'affichage » (onResume)
  - Les autres : « passives » (onStop)
- **Création**
  - Acitvity pas dans la liste
    - **Création et ajout en tête de liste => onCreate / onStart / onResume**
  - Activity dans la liste
    - **Mise en tête de liste => onRestart / onStart / onResume**
- **Mise en pause**
  - Lancement d'une autre Activité...
    - **onPause / onStop**
- **Terminaison**
  - 1) L'utilisateur appuie sur « Annuler »
  - 2) Cas particuliers (cf « Activity : Killed »)

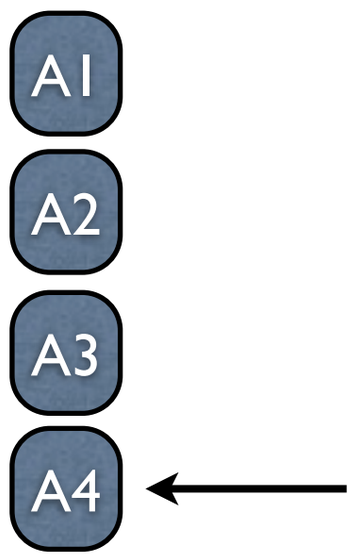
Gestion des Activités : liste d'Activités

- **Le système gère une liste d'Activités**
  - La tête de liste : « en cours d'affichage » (onResume)
  - Les autres : « passives » (onStop)
- **Création**
  - Acitvity pas dans la liste
    - **Création et ajout en tête de liste => onCreate / onStart / onResume**
  - Activity dans la liste
    - **Mise en tête de liste => onRestart / onStart / onResume**
- **Mise en pause**
  - Lancement d'une autre Activité...
    - **onPause / onStop**
- **Terminaison**
  - 1) L'utilisateur appuie sur « Annuler »
  - 2) Cas particuliers (cf « Activity : Killed »)



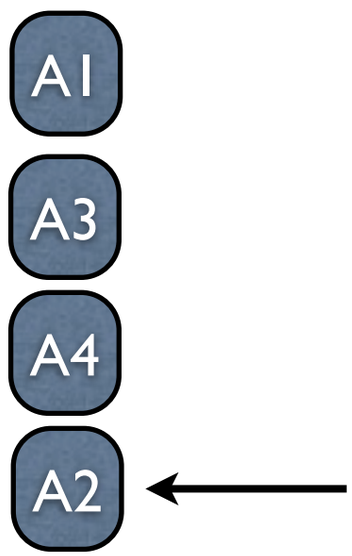
Gestion des Activités : liste d'Activités

- **Le système gère une liste d'Activités**
  - La tête de liste : « en cours d'affichage » (onResume)
  - Les autres : « passives » (onStop)
- **Création**
  - Acitvity pas dans la liste
    - **Création et ajout en tête de liste => onCreate / onStart / onResume**
  - Activity dans la liste
    - **Mise en tête de liste => onRestart / onStart / onResume**
- **Mise en pause**
  - Lancement d'une autre Activité...
    - **onPause / onStop**
- **Terminaison**
  - 1) L'utilisateur appuie sur « Annuler »
  - 2) Cas particuliers (cf « Activity : Killed »)



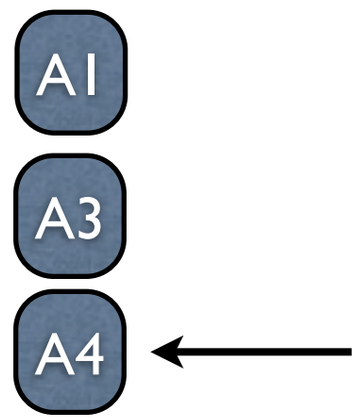
Gestion des Activités : liste d'Activités

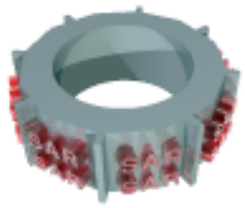
- **Le système gère une liste d'Activités**
  - La tête de liste : « en cours d'affichage » (onResume)
  - Les autres : « passives » (onStop)
- **Création**
  - Acitvity pas dans la liste
    - **Création et ajout en tête de liste => onCreate / onStart / onResume**
  - Activity dans la liste
    - **Mise en tête de liste => onRestart / onStart / onResume**
- **Mise en pause**
  - Lancement d'une autre Activité...
    - **onPause / onStop**
- **Terminaison**
  - 1) L'utilisateur appuie sur « Annuler »
  - 2) Cas particuliers (cf « Activity : Killed »)



Gestion des Activités : liste d'Activités

- **Le système gère une liste d'Activités**
  - La tête de liste : « en cours d'affichage » (onResume)
  - Les autres : « passives » (onStop)
- **Création**
  - Acitvity pas dans la liste
    - **Création et ajout en tête de liste => onCreate / onStart / onResume**
  - Activity dans la liste
    - **Mise en tête de liste => onRestart / onStart / onResume**
- **Mise en pause**
  - Lancement d'une autre Activité...
    - **onPause / onStop**
- **Terminaison**
  - 1) L'utilisateur appuie sur « Annuler »
  - 2) Cas particuliers (cf « Activity : Killed »)





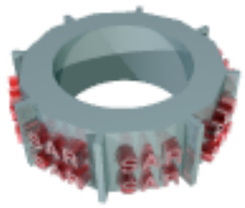
Activity : Killed ...

◉ Terminaison d'une Activity ?

- Cas nominal : create-start-resume-pause-stop-restart
- Autres : « killed » par le système
  - **Mémoire - Changement de l'orientation**
  - **Peut arriver n'importe quand !**
- « killed » ?
  - **Application détruite (onDestroy)**
  - **Mais qui sera notée comme re-exécutable (Toujours dans la liste d'activity)**
  - **Si pas de Persistance ... perte des données**

◉ Solution ?

- Persistance intermédiaire ? durable ?
- Mécanisme offert par Android



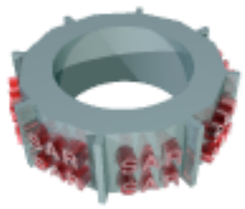
Activity : Killed ...

Mécanisme

- onSaveInstanceState / onRestoreInstanceState
  - **Dans le cycle de vie :**
    - onResume / IDLE / onSave / onPause / onStop /onDestroy
    - onCreate / onStart / onRestore / onResume / IDLE
  - Déclenché uniquement si « killed »

Utilisation

- Eviter la perte de données ?
  - `protected void onSaveInstanceState(Bundle state)`
  - `public void onRestoreInstanceState(Bundle state)`

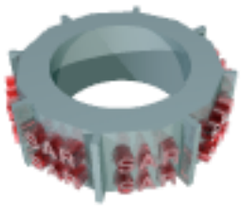


Activity : Killed

Utilisation

- Objet Bundle : via un MAP (clé,valeur) : TYPE SIMPLE
  - `state.putString(String key,String value)`
  - `String state.getString(String key)`
  - **Prise en compte des types simples, Vecteur, etc.**

```
protected void onSaveInstanceState(Bundle outState) {
   super.onSaveInstanceState(outState);
   outState.putString("hello", "message a sauvegarder");
}

protected void onRestoreInstanceState(Bundle savedInstanceState) {
   super.onRestoreInstanceState(savedInstanceState);
   String seq = savedInstanceState.getString("hello");
}
```



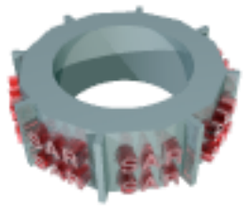
Activity : Killed ...

Pour un type complexe ?

- Implémenter Parcelable

```
public class Student {
   private String nom;
   private String prenom;
}
```

```
public class Student implements Parcelable {
    /* FACTORY : utilisé par Android pour instancier le composant */
    public static final Parcelable.Creator<Student> CREATOR =
      new Parcelable.Creator<Student>() {
        public Student createFromParcel(Parcel in) {
       return new Student(in);
    }
     /* Enregistrer les données */
     public void writeToParcel(Parcel dest, int flag) {
       dest.writeString(nom);
       dest.writeString(prenom);
    }
     /* Charger les données */
    private Student(Parcel in){
       nom = in.readString();
       prenom = in.readString();
...
```



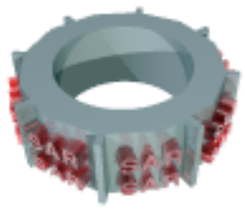
Persistance : préférences

Les préférences

- Equivalent au fichier .INI ou base de registre sous Windows
- Stocker des options, paramètres ...
- Données primaires via MAP (clé,valeur)

```
SharedPreferences settings = getSharedPreferences("preferences_file", MODE_PRIVATE);
SharedPreferences.Editor editor = settings.edit();
editor.putInt("counter", current);
editor.commit();

SharedPreferences settings = getSharedPreferences("preferences_file", MODE_PRIVATE);
current = settings.getInt("counter", -42);
```

- mode :
  - **MODE_PRIVATE : seulement accessible par l'activité**
  - **MODE_WORLD_READABLE : en lecture pour toutes les activités**
  - **MODE_WORLD_WRITEABLE : en écriture pour toutes les activités**

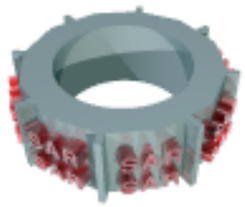


Persistance : file system

Le stockage interne / externe

- Fichier de sauvegarde
- interne, externe ?
  - **interne : la mémoire où s'exécute ANDROID**
  - **externe : média externe (sdcard)**
- Différence :
  - **L'une est toujours présente (interne)**
  - **L'autre peut peut être retirée (externe)**
- Utilisation en interne

```
String toSave = "Hello the world";
FileOutputStream fos = openFileOutput("hello.txt", Context.MODE_PRIVATE);
fos.write(toSave.getBytes());
fos.close();
```



Persistance : autres techniques

Utilisation du système de fichiers :

- vérifier la présence du média

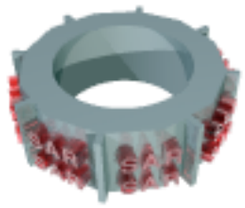
```
String state = Environment.getExternalStorageState();
if (Environment.MEDIA_MOUNTED.equals(state)) {
        // READ - WRITE on file system
} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        // READ only on file system
 } else {
     // No access to MEDIA
 }
```

Utilisation d'une base de données

- BdD SQL-Lite accessible en direct
- Framework d'utilisation : ContentProvider

Exemples d'activités: Les listes

- Listes simples
- Listes personnalisées
- Les adaptateurs



La ListView ...

Fonctionnement d'une ListView

- Mise en place du M - V - C
  - **M = Collection de données**
  - **V = Cellules de la liste (définis par XML)**
  - **C = Binding entre le Modèle et la Vue**

1001 façons d'utiliser une ListView

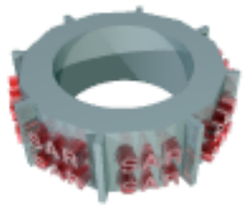
- Gestion simplifiée via un mapping automatique (facile)
  - **Dans le cas de cellule simple ...**
- Gestion personnalisée (plus compliqué, il faut connaitre les concepts associés)
  - **Dans le cas de cellules personnalisées ...**

La ListActivity

◉ Gestion Simplifiée :

- 1) Créer une ListActivity
- 2) Choisir un Layout prédéfini sous forme XML

```
public class MainActivity extends ListActivity {
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        String[] values = new String[] {"Android", "iPhone","WindowsMobile"};

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, values);
        setListAdapter(adapter);
    }
}
```



Le SimpleAdapteur

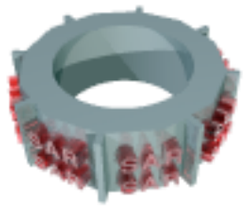
◉ Cellule personnalisée - Binding automatique :

- Le SimpleAdapter ....
- XML personnalisé de la cellule
  - **Plusieurs éléments graphiques (TextView, etc.)**
- Créer une ListActivity
- SimpleAdapter
  - **Spécifier les bindings via des MAP (Clé - Valeur)**
    - Champs des données
    - Identifiant de la vue
- Initialiser la ListView avec l'adapter

SimpleAdapteur : exemple


```
public class MainActivity extends ListActivity {
    ArrayList<HashMap<String,String>> list = new ArrayList<HashMap<String,String>>();
    private SimpleAdapter notes;
    private int itemId = 0;

    public void onCreate(Bundle icicle)
    {
       super.onCreate(icicle);
       for (int i = 0; i < 5; ++i) addItem();

       notes = new SimpleAdapter(
                this, list,
                R.layout.main_item_two_line_row,
                new String[] { "line1","line2" },
                new int[] { R.id.text1, R.id.text2 } );
       setListAdapter( notes );
    }

    private void addItem() {
          ++itemId;
          HashMap<String,String> item = new HashMap<String,String>();
          item.put( "line1","Property" +Integer.toString(itemId ) );
          item.put( "line2","ItemID: "+ Integer.toString( itemId ) );
          list.add( item );
          notes.notifyDataSetChanged();
        }
}
```

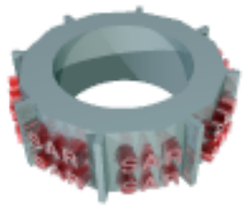
Nommage éléments & XML de cellule



L'Adaptateur

◉ Adaptateur en détail ?

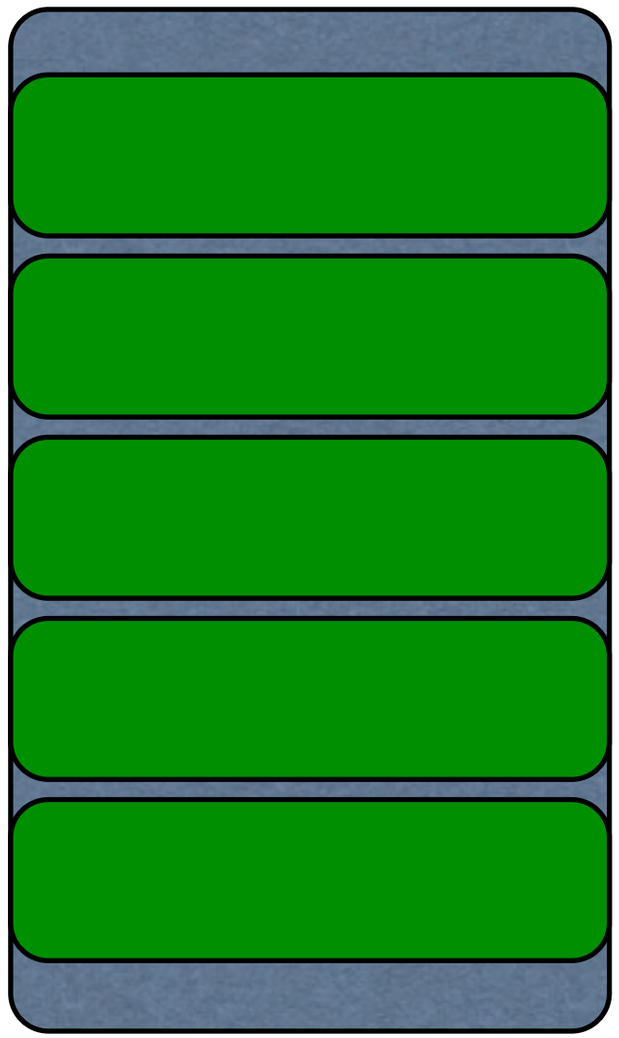
- Surcharger Adapter (ArrayAdapter)
  - **getView(int position,View convertView,View parent)**
    - `Position:` index de l'élément à afficher
    - `convertView:` vue recyclée (peut être NULL)
  - **RECYCLAGE de vues (comme I-Phone)**
    - Eviter des allocations excessives (100 000 cellules ...)
      - Si convertView == NULL : créer la vue
      - Si convertView != NULL : la vue est recyclée
    - En cas de recyclage :
      - ATTENTION A L'INITIALISATION

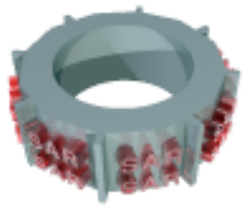


L'Adaptateur

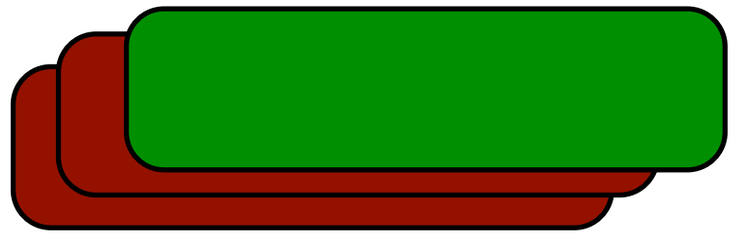
◉ Adaptateur en détail ?

- Surcharger Adapter (ArrayAdapter)
  - **getView(int position,View convertView,View parent)**
    - `Position:` index de l'élément à afficher
    - `convertView:` vue recyclée (peut être NULL)
  - **RECYCLAGE de vues (comme I-Phone)**
    - Eviter des allocations excessives (100 000 cellules ...)
      - Si convertView == NULL : créer la vue
      - Si convertView != NULL : la vue est recyclée
    - En cas de recyclage :
      - ATTENTION A L'INITIALISATION



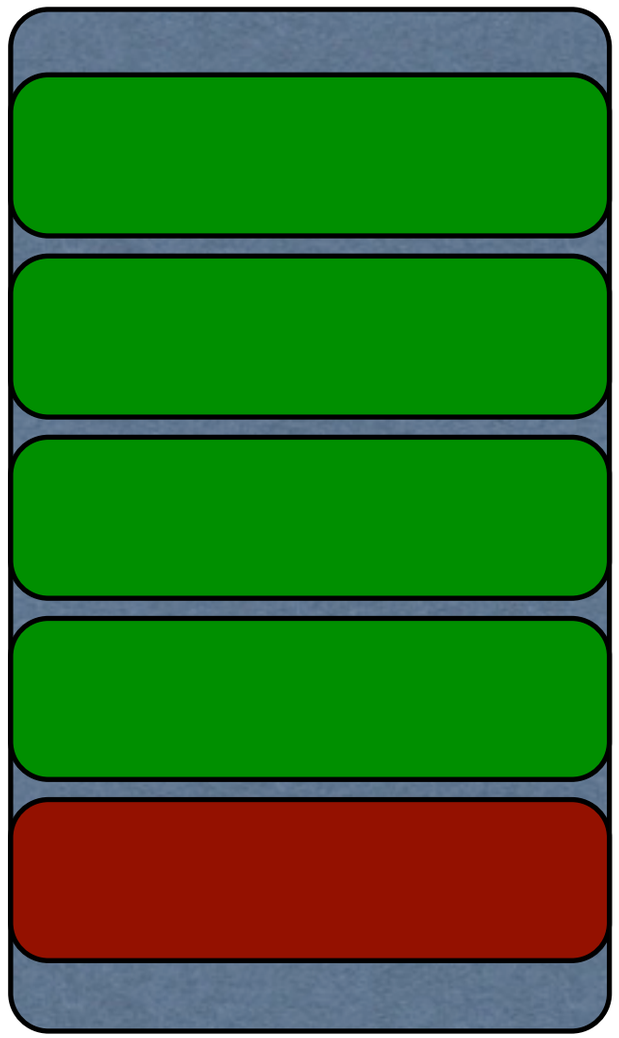


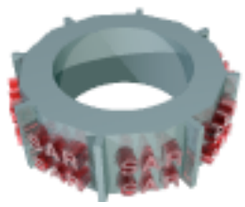
L'Adaptateur



◉ Adaptateur en détail ?

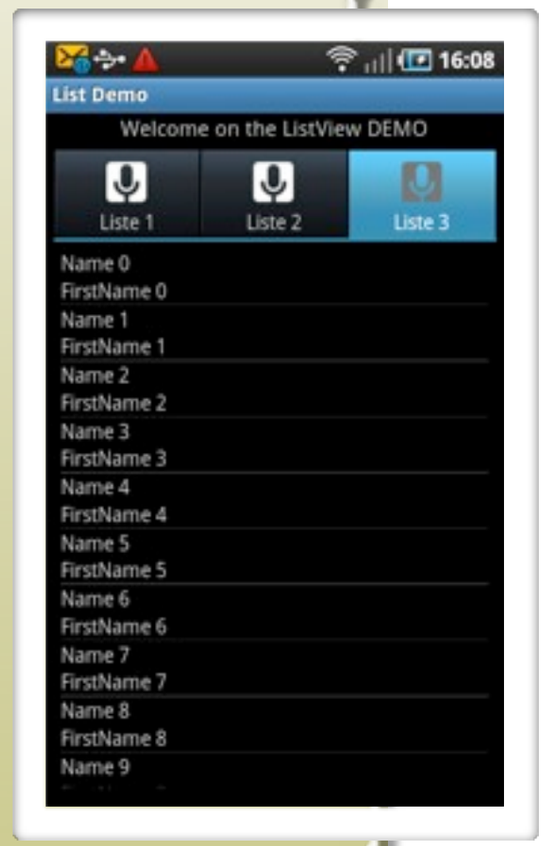
- Surcharger Adapter (ArrayAdapter)
  - **getView(int position,View convertView,View parent)**
    - `Position:` index de l'élément à afficher
    - `convertView:` vue recyclée (peut être NULL)
  - **RECYCLAGE de vues (comme I-Phone)**
    - Eviter des allocations excessives (100 000 cellules ...)
      - Si convertView == NULL : créer la vue
      - Si convertView != NULL : la vue est recyclée
    - En cas de recyclage :
      - ATTENTION A L'INITIALISATION

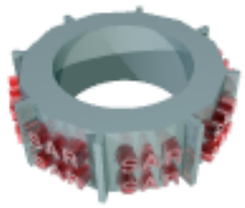




L'Adapteur : exemple

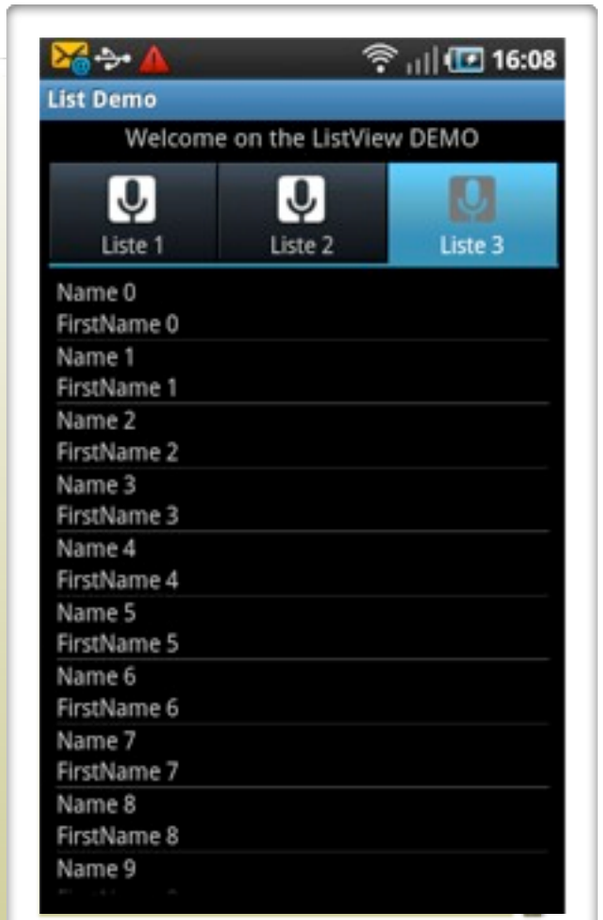

```
public class ContactAdapter extends ArrayAdapter<Contact> {
...
public View getView(int position, View convertView, ViewGroup parent) {
  /* Check if we re-used a new view */
  View view = convertView;
  if (view == null){
  /* Create a new View */
  LayoutInflater li = (LayoutInflater) getContext().
  getSystemService(Context.LAYOUT_INFLATER_SERVICE);
  view = li.inflate(viewId, null);
  }

  /* Initialize the view */
  Contact c = getItem(position);
  if (c != null){
   TextView name = view.findViewById(R.id.TextView01);
   TextView firstname  = view.findViewById(R.id.TextView02);
   name.setText(c.getName());
   firstname.setText(c.getFisrtName());
  }
  return view;
 }
}
```

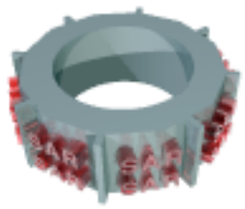




L'Adapteur : exemple

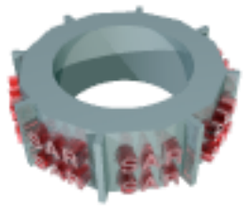

```
private List<Contact> data = new ArrayList<Contact>();

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* Populate list */
        for(int i = 0;i<20;i++){
         Contact c = new Contact();
         c.setName("Name "+i);
         c.setFisrtName("FirstName "+i);
         data.add(c);
        }
        /* Instaciate adapter with XML Layout representing on Item */
        ContactAdapter adapter = new ContactAdapter(
            this,
            R.layout.item_cell_2,
            data);
        getListView().setAdapter(adapter);
    }
```





Les Adaptateurs

- **Il existe différents types d'Adaptateurs ...**
  - `SimpleAdapter`: description statique de la correspondance entre les données et la description XML
  - `ArrayAdapter`: si surchargé permet d'adapter la vue pour chacun des éléments sinon agit comme `SimpleAdapter`
  - `CursorAdapter` et `SimpleCursorAdapter:` permet l'accès aux bases de données
  - `SpinnerAdapter`: permet de faire le lien entre les listes déroulantes et leurs données
- **... il faut savoir les utiliser à bon escient**

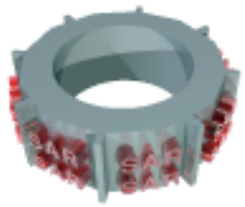


ListView : contextMenu

Evénements

- Déclencher un ContextMenu

```
lv.setOnCreateContextMenuListener(new OnCreateContextMenuListener(){
            public void onCreateContextMenu(ContextMenu menu,
            View v,
            ContextMenuInfo menuInfo) {
                AdapterView.AdapterContextMenuInfo info =
                    (AdapterView.AdapterContextMenuInfo)menuInfo;
                current = info.position;
                /* We must inflat the menu at the creation */
                MenuInflater inflater = getMenuInflater();
                inflater.inflate(R.menu.context_menu_1, menu);
            }
        });
```



ListView : événements

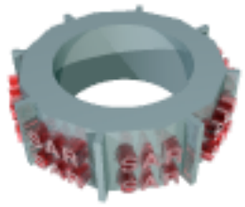
Evénements

- Click (idem pour LongClick)

```
lv.setOnItemClickListener(new OnItemClickListener() {
      public void onItemClick(AdapterView<?> parent
                    ,View view
                    ,int position
                    , long id) {
        Toast.makeText(context,adapter.getItem(position).getNom(),
                        Toast.LENGTH_SHORT).show();
      }
});
```

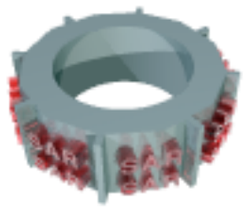
- Les listes sont beaucoup utilisées et il existe beaucoup de façon de les utiliser

Fragments



Fragment : définition

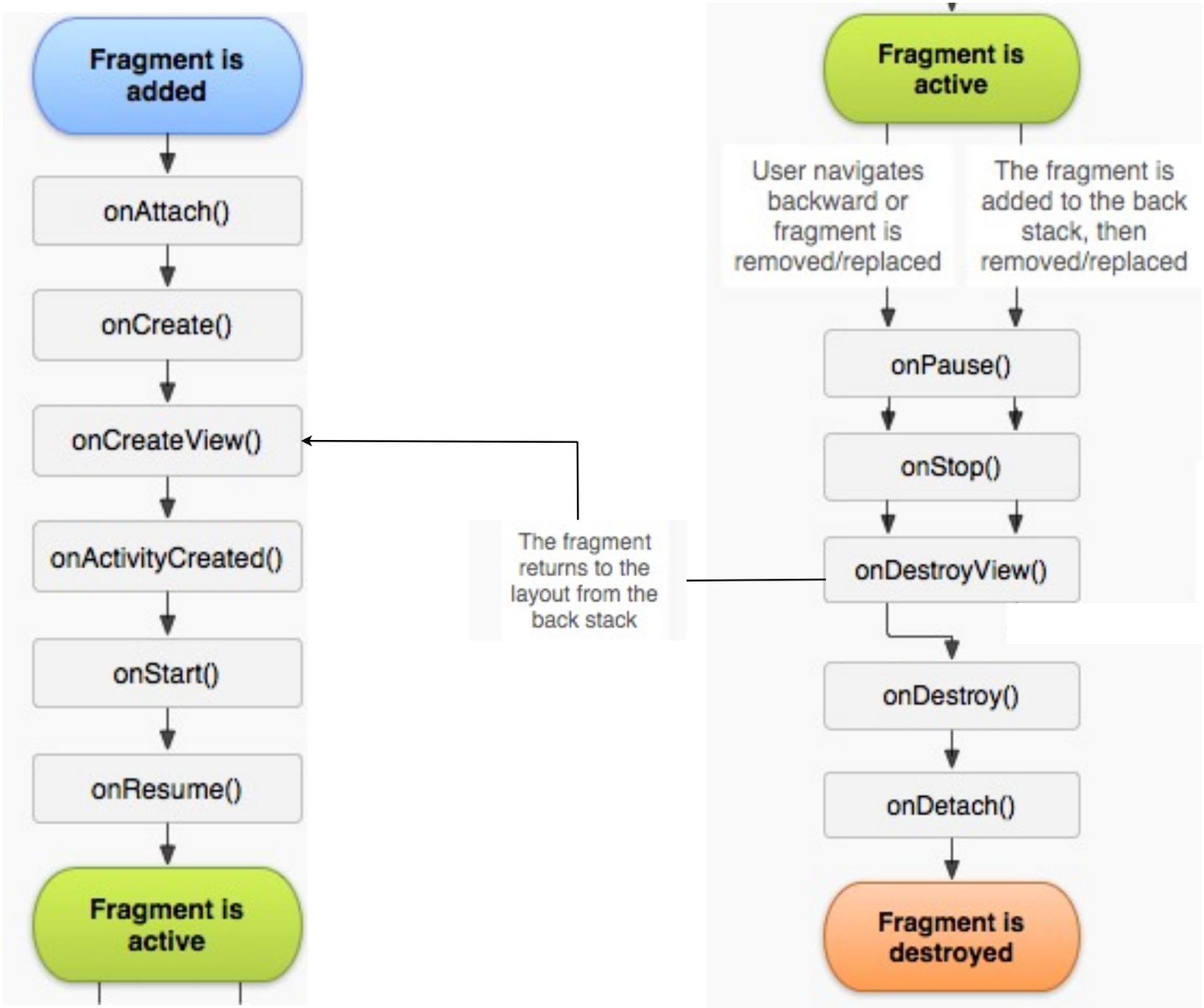
- **Partie modulaire d'une activité**
  - Composant réutilisable
  - Construction de panels de vues
- **Possède son propre cycle de vie**
  - Ses propres événements
  - Manipulation indépendante au niveau de l'activité
  - ... mais suit le cycle de vie de l'activité
- **Ajout ou retirait dynamique d'une activité**
  - Assez proches des vues sous iOS



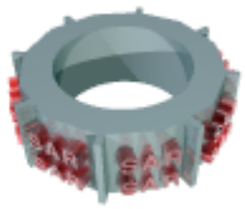
Fragment : remarques

- Depuis Android 3.0 (API level 11.)
- Possède son propre layout
- Possède son propre ViewGroup
- Fonctionne de la même manière que les Activités
- Gestion via le `FragmentManager`
- Pas nécessairement d'UI

Fragment : cycle de vie

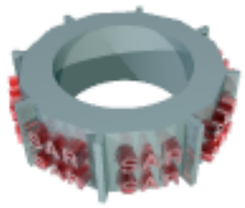


Framework de communication



Les Intents

- **Peut être vu comme une demande de démarrage du composant récepteur**
- **Description abstraite d'une opération à réaliser**
  - `startActivity:` permet de lancer une nouvelle activité
  - `broadcastIntent:` Communication avec les broadcastReceiver (exemple Alarme, SMS, etc.)
  - `startService` or `bindService:` communication avec les Services
- **Sert de «glue» entre les différentes activités**
- **Toutes les activités activable via des Intent doivent être déclarées dans Android-Manifest.xml**



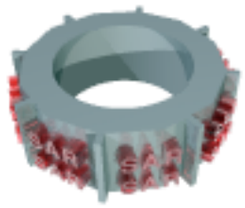
L'IntentFilter

- **Mécanisme permettant de trouver l'activité la plus adéquate pour une action (Intent)**
  - Pour ouvrir un fichier,
  - Pour ouvrir une page web, etc.

- **Se déclarer auprès du système comme recepteurs possible pour le filtre spécifé**

- **Attention ! Un filtre dédié à chaque action**

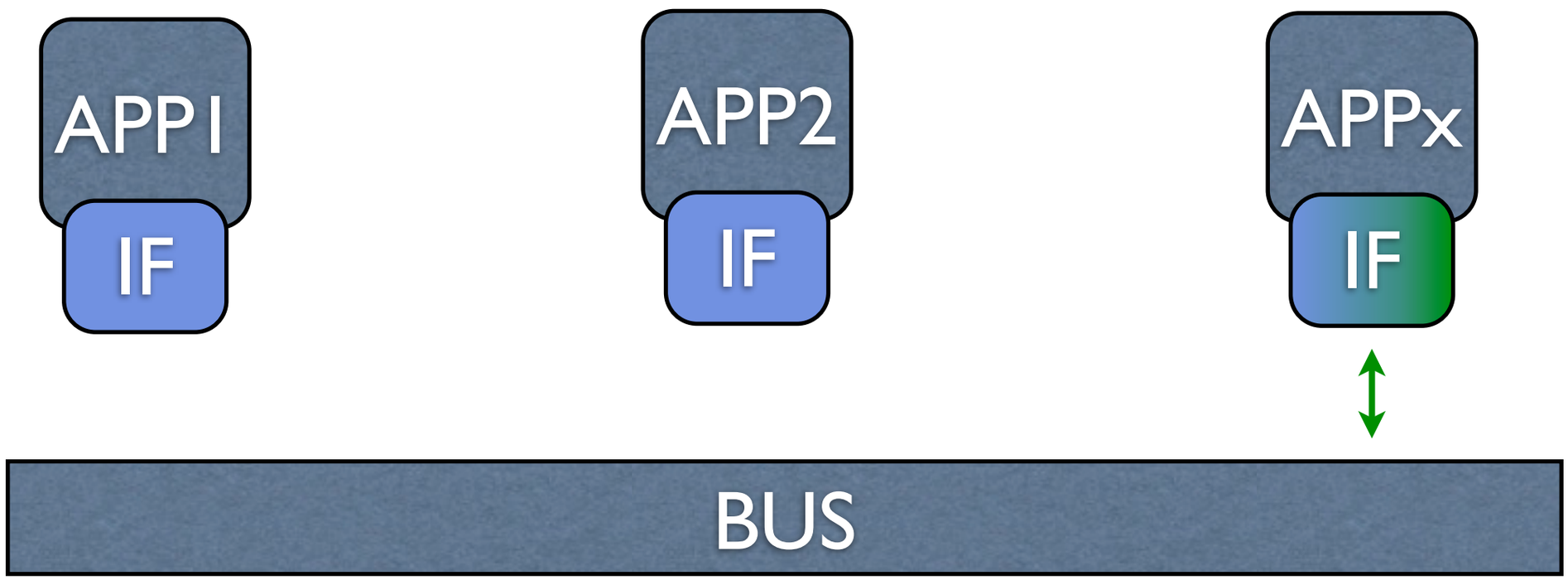
- **Déclaration dans le Android-Manifest.xml**

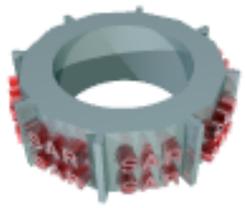


Framework de communication

Bus à objets asynchrone

- Les objets sont des Intents
- Tous les Intents partagent le même bus
- Possibilité de «Broadcast» très utilisé pour les service

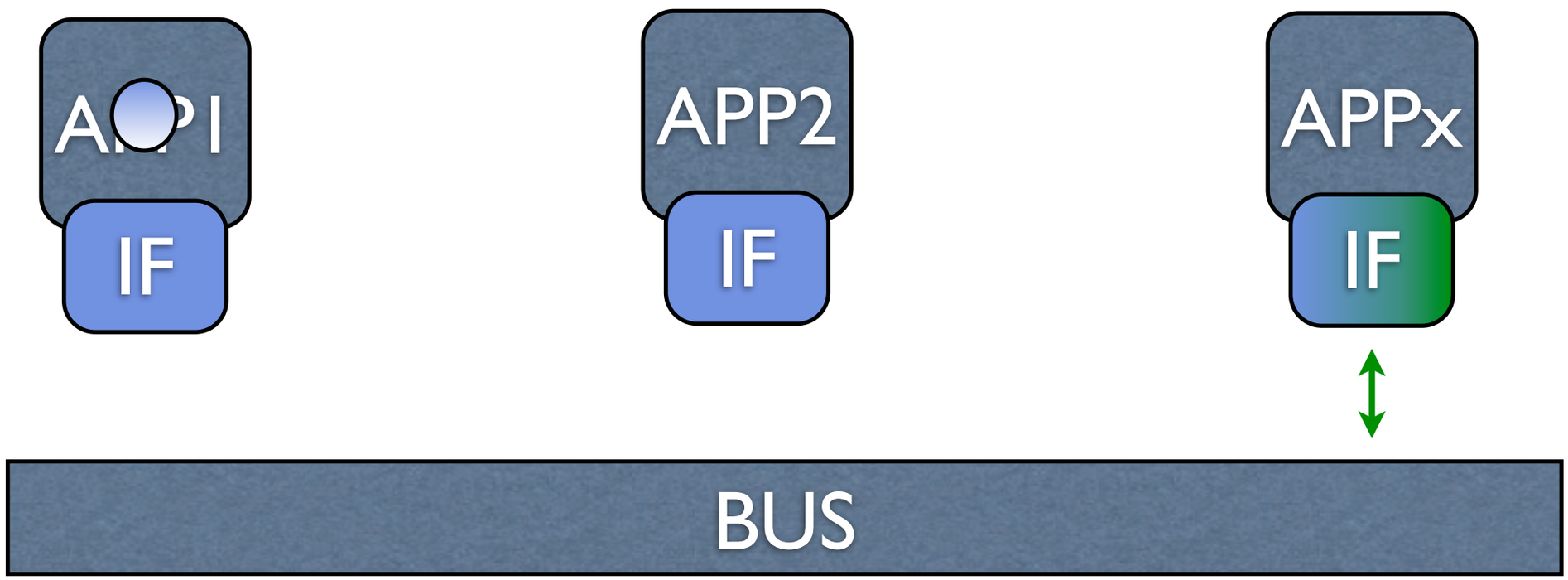


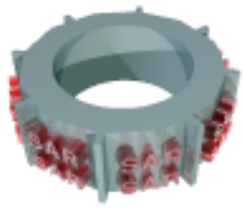


Framework de communication

Bus à objets asynchrone

- Les objets sont des Intents
- Tous les Intents partagent le même bus
- Possibilité de «Broadcast» très utilisé pour les service

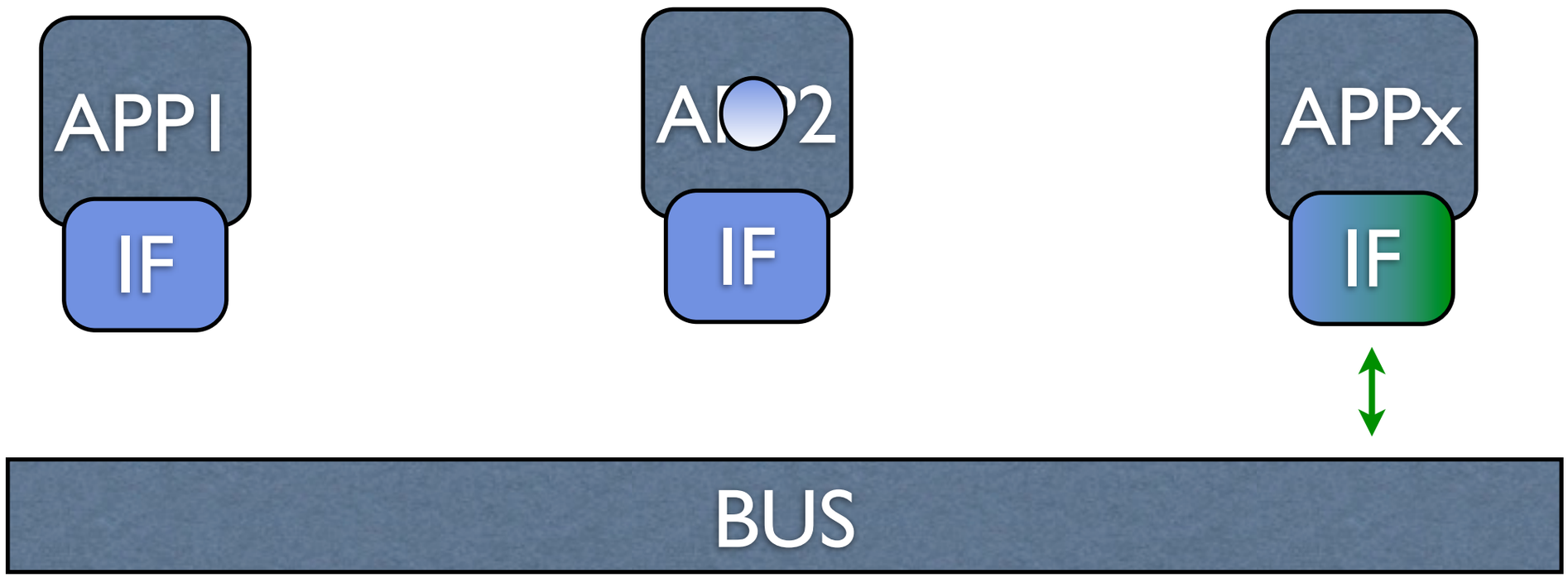




Framework de communication

Bus à objets asynchrone

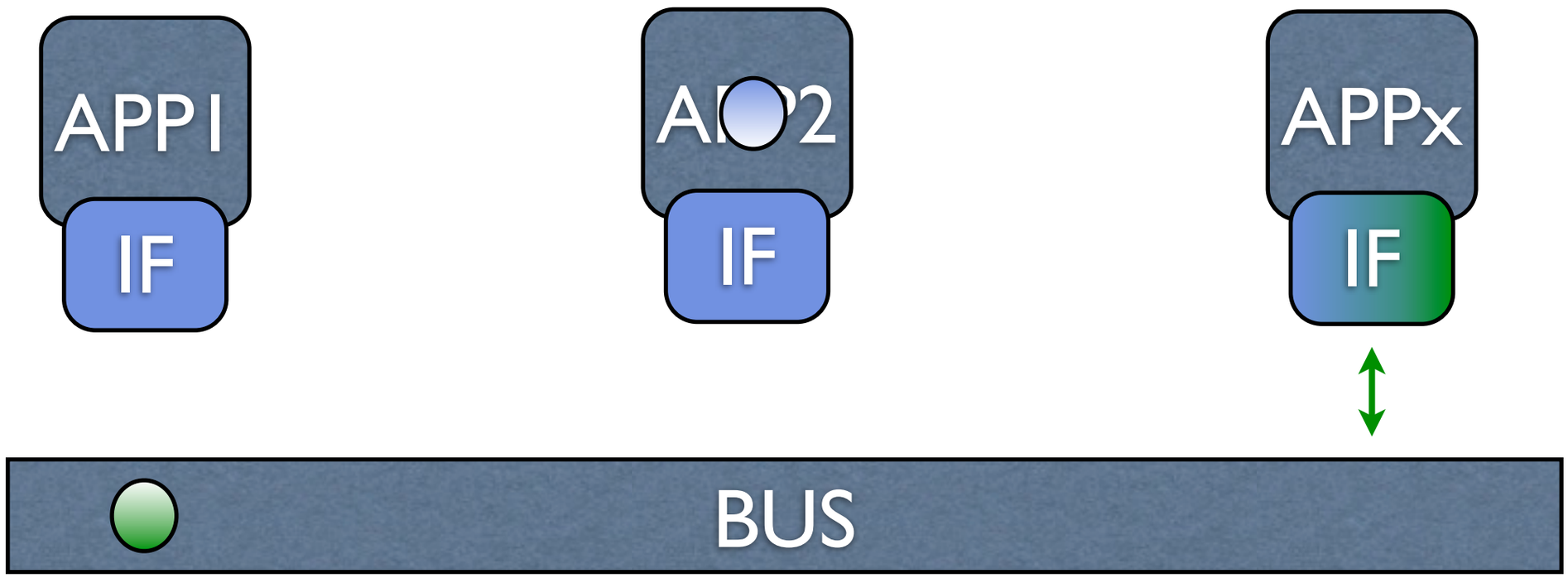
- Les objets sont des Intents
- Tous les Intents partagent le même bus
- Possibilité de «Broadcast» très utilisé pour les service



Framework de communication

Bus à objets asynchrone

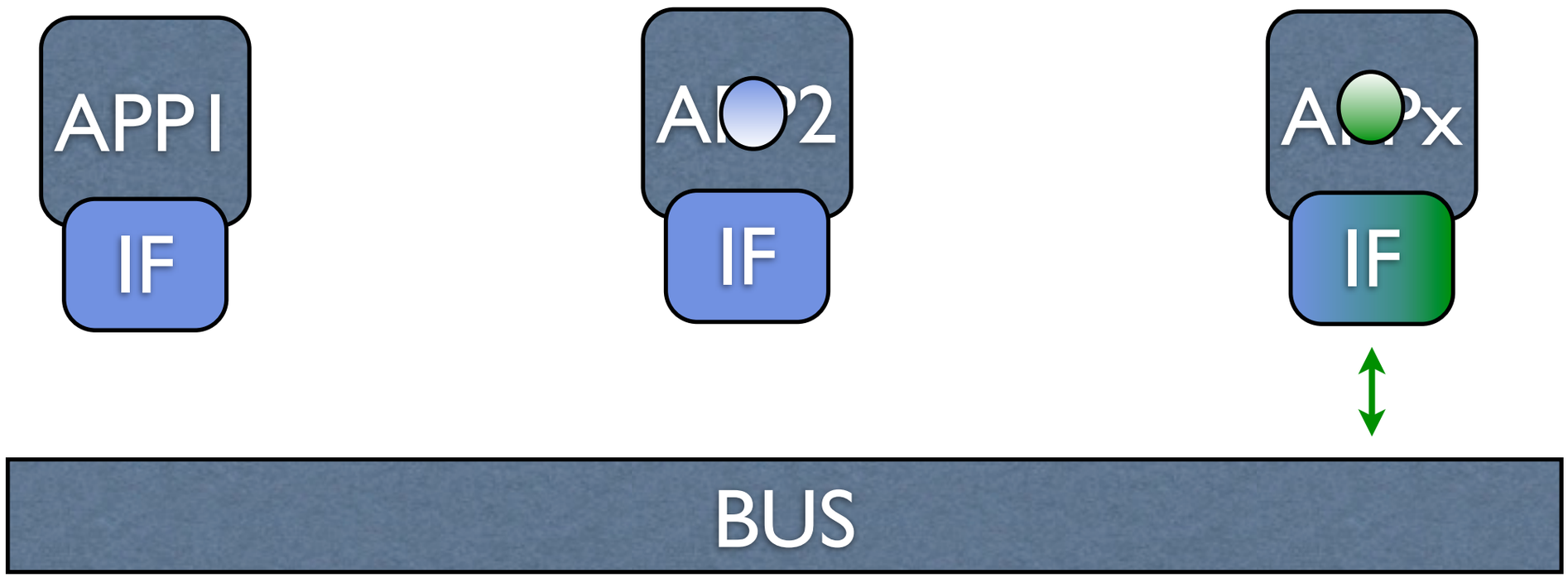
- Les objets sont des Intents
- Tous les Intents partagent le même bus
- Possibilité de «Broadcast» très utilisé pour les service



Framework de communication

Bus à objets asynchrone

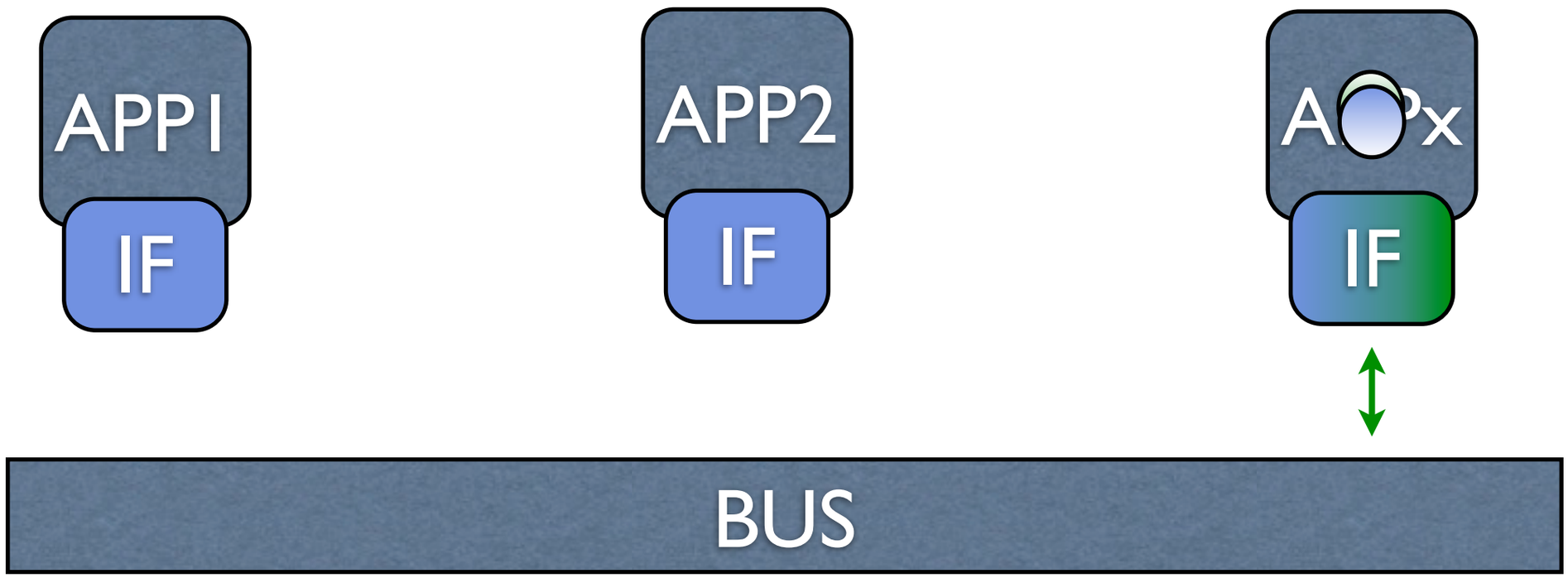
- Les objets sont des Intents
- Tous les Intents partagent le même bus
- Possibilité de «Broadcast» très utilisé pour les service

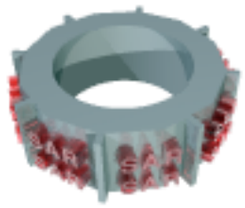


Framework de communication

Bus à objets asynchrone

- Les objets sont des Intents
- Tous les Intents partagent le même bus
- Possibilité de «Broadcast» très utilisé pour les service





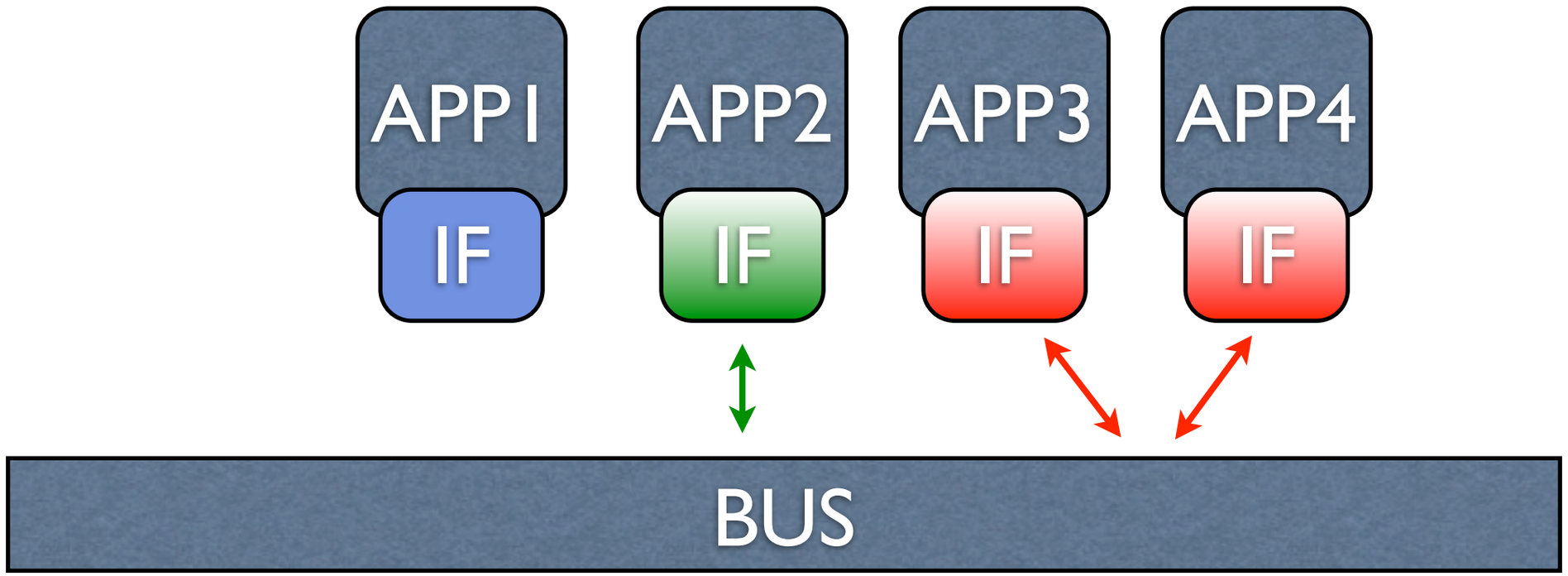
Framework de communication

- **Destinataire du message ?**
  - Par désignation explicite
    - **Destinataire : Contexte + Classe**
    - **Utilisé en interne d'une application**
  - Par désignation implicite
    - **Action + Type de données / URI**
    - **Recherche du destinataire, utilisation des IntentFilter**
- **Action ?**
  - Afficher / Sélectionner / Editer / Appeler ...
- **Type de données ?**
  - Un contact / Une image / Un numéro de téléphone

Framework de communication

Une application doit s'enregistrer

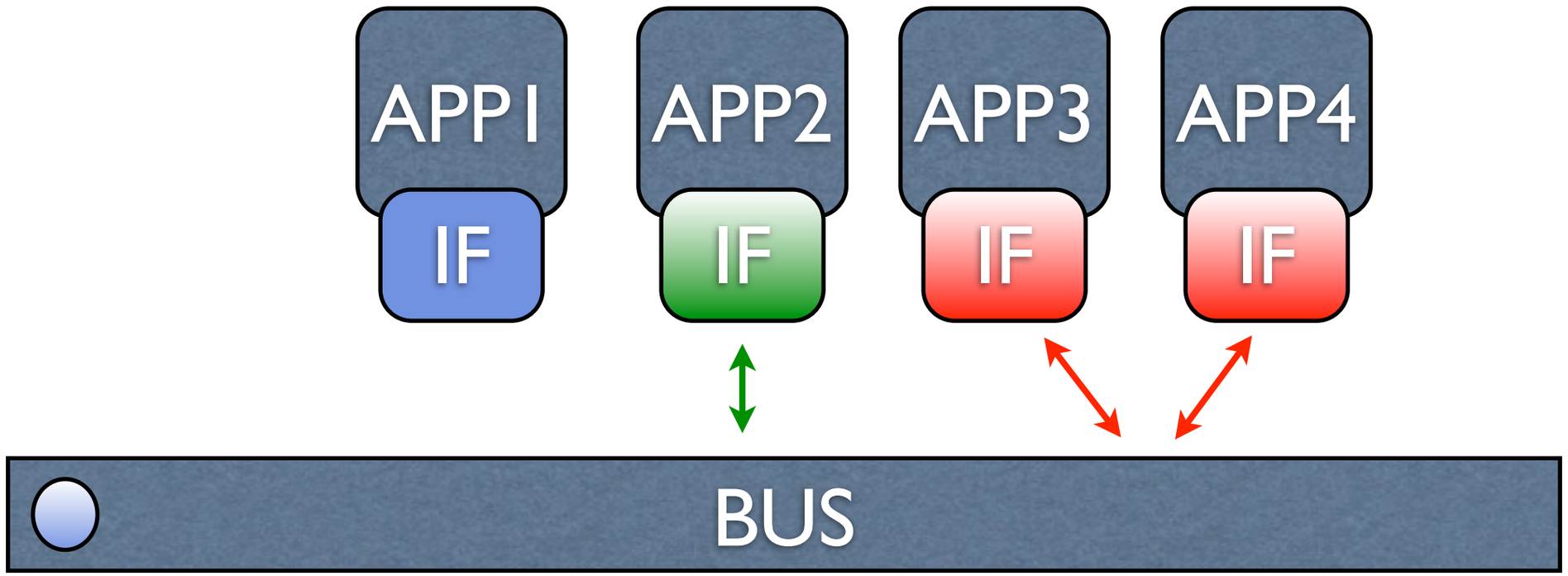
- `IntentFilter:` Action,Type, Catégorie, ..
- Nommage explicite des Intents (doit être déclaré dans le Manifest.xml)
- Nommage implicite(pas besoin d'enregistrer dans Manifest.xml)



Framework de communication

Une application doit s'enregistrer

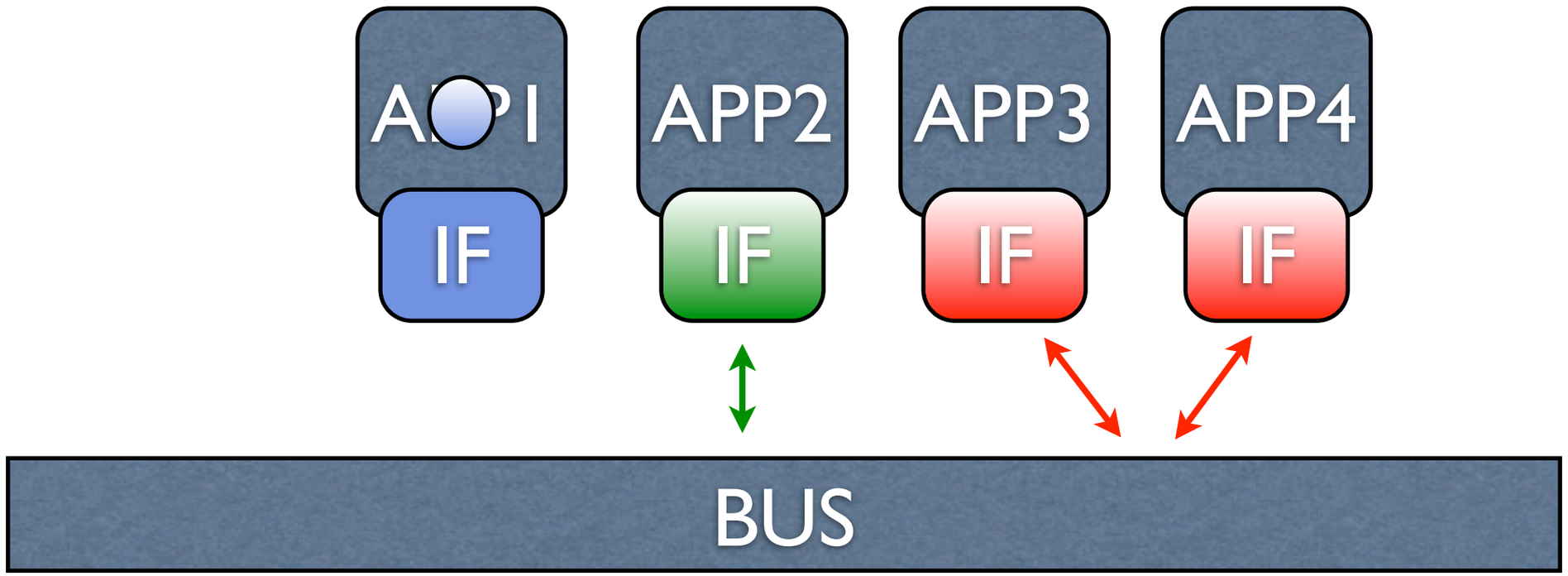
- `IntentFilter:` Action,Type, Catégorie, ..
- Nommage explicite des Intents (doit être déclaré dans le Manifest.xml)
- Nommage implicite(pas besoin d'enregistrer dans Manifest.xml)



Framework de communication

Une application doit s'enregistrer

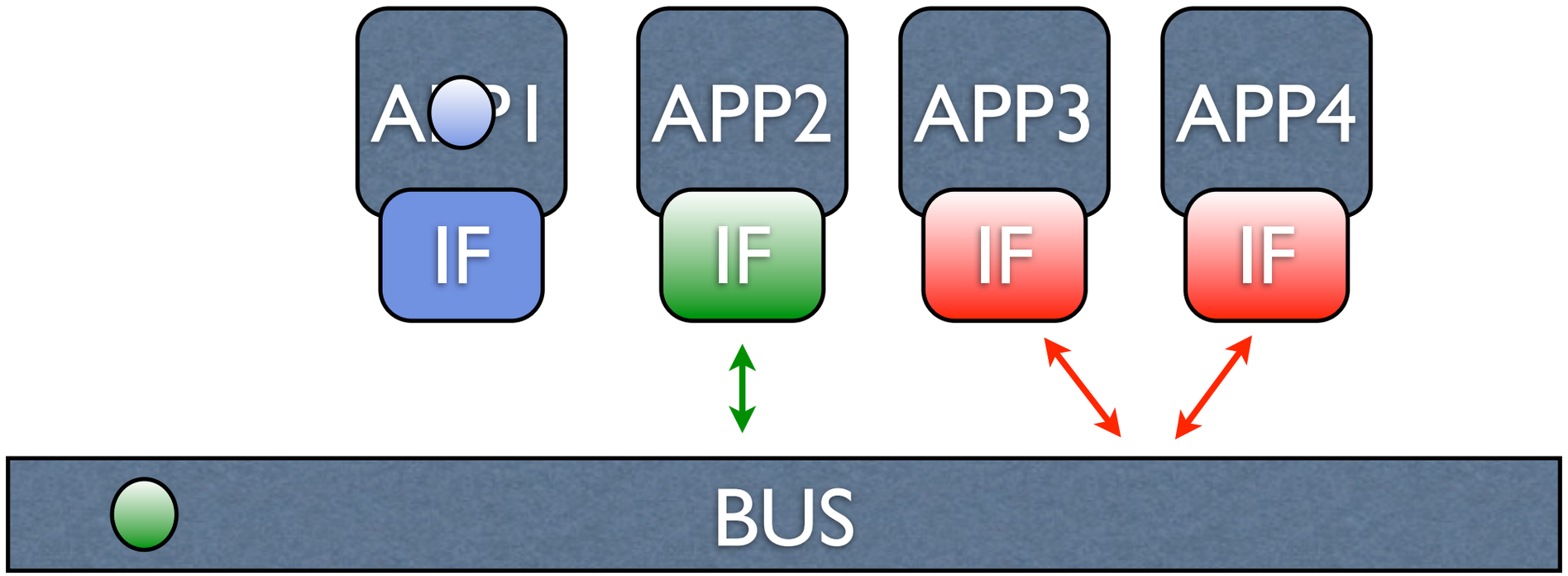
- `IntentFilter:` Action,Type, Catégorie, ..
- Nommage explicite des Intents (doit être déclaré dans le Manifest.xml)
- Nommage implicite(pas besoin d'enregistrer dans Manifest.xml)



Framework de communication

Une application doit s'enregistrer

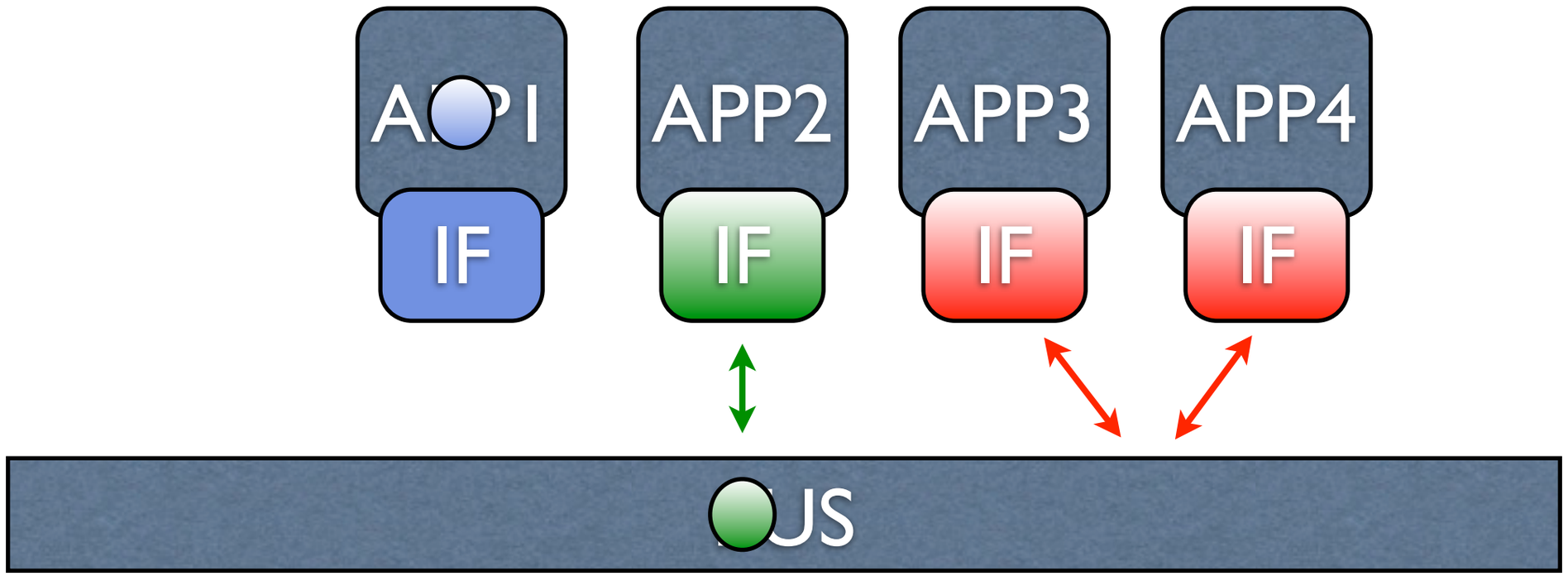
- `IntentFilter:` Action,Type, Catégorie, ..
- Nommage explicite des Intents (doit être déclaré dans le Manifest.xml)
- Nommage implicite(pas besoin d'enregistrer dans Manifest.xml)



Framework de communication

Une application doit s'enregistrer

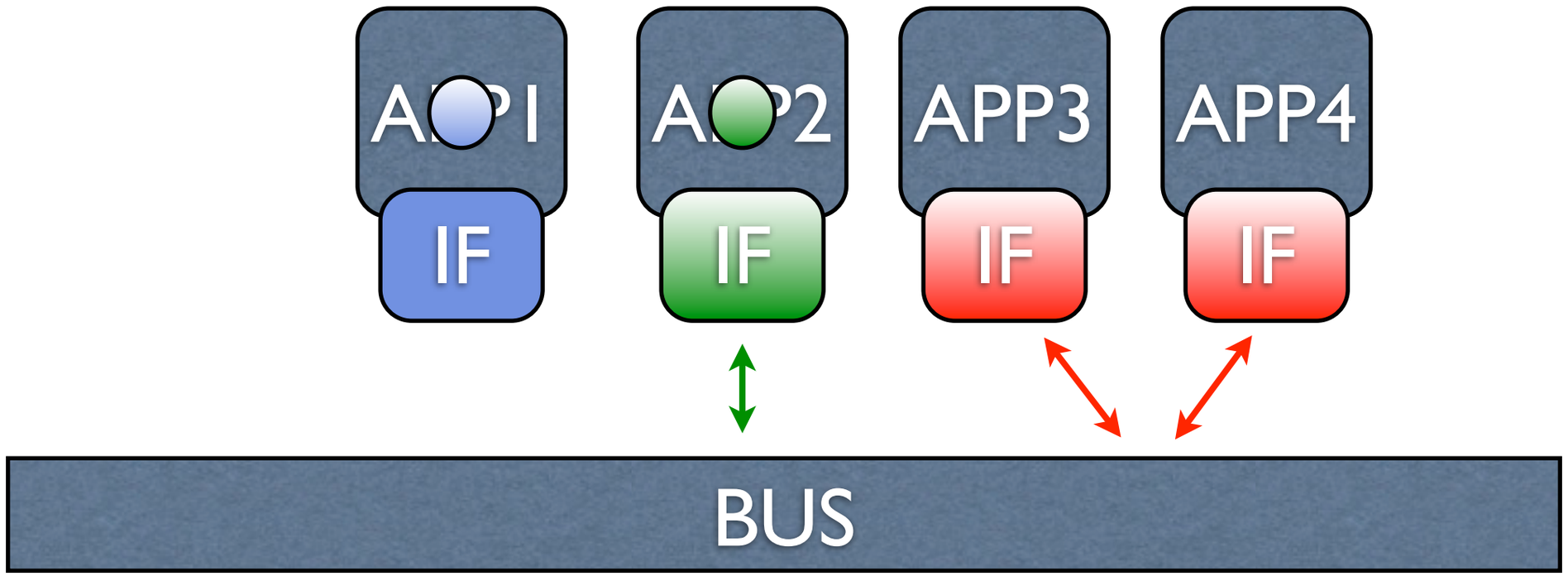
- `IntentFilter:` Action,Type, Catégorie, ..
- Nommage explicite des Intents (doit être déclaré dans le Manifest.xml)
- Nommage implicite(pas besoin d'enregistrer dans Manifest.xml)



Framework de communication

Une application doit s'enregistrer

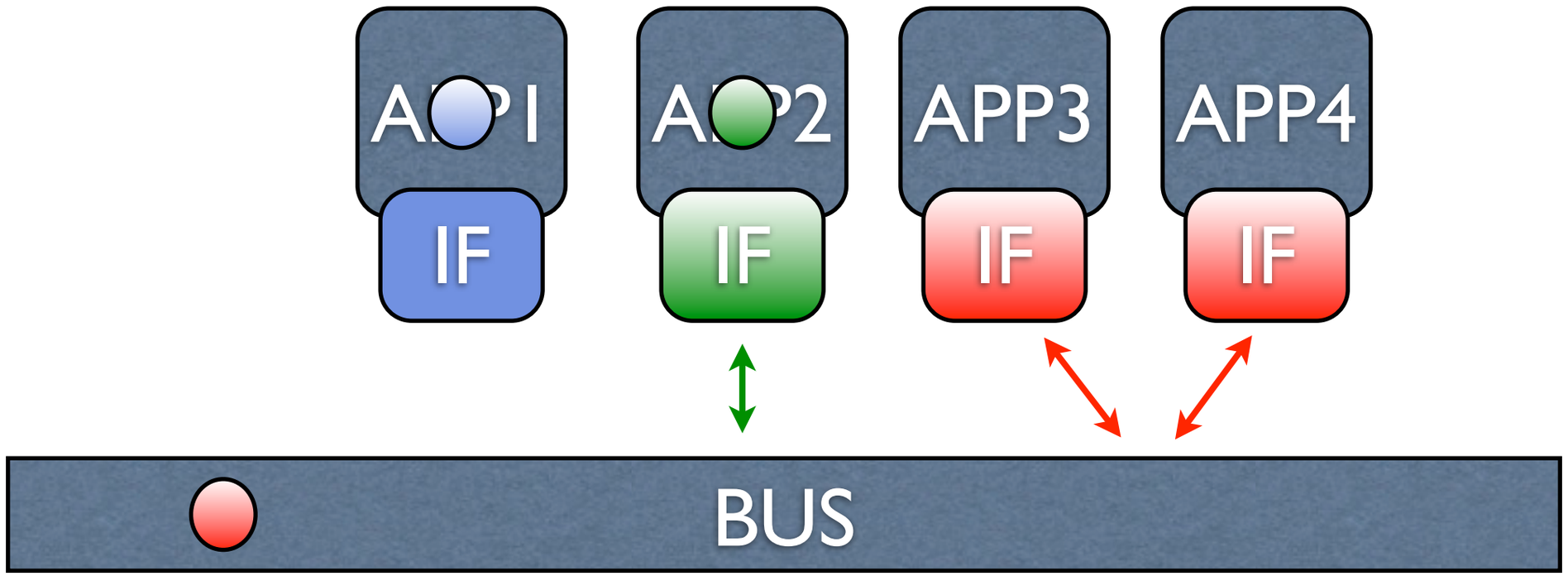
- `IntentFilter:` Action,Type, Catégorie, ..
- Nommage explicite des Intents (doit être déclaré dans le Manifest.xml)
- Nommage implicite(pas besoin d'enregistrer dans Manifest.xml)



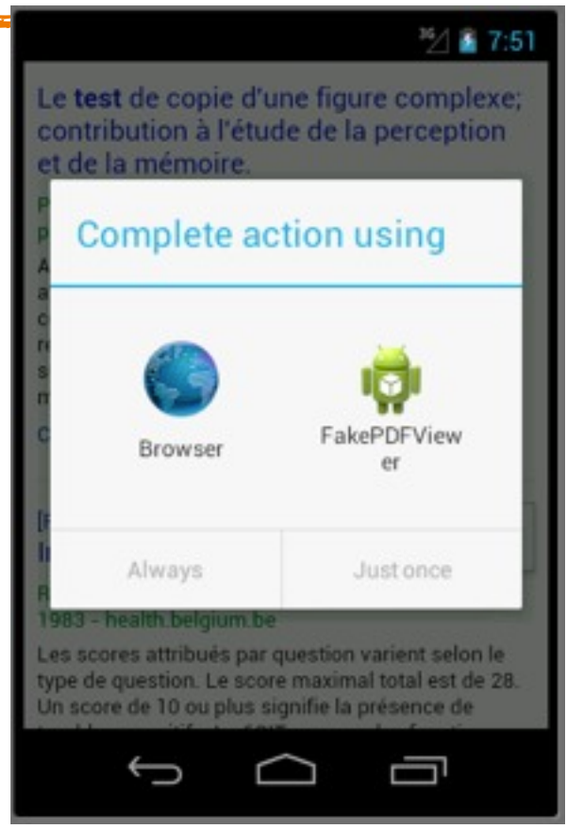
Framework de communication

Une application doit s'enregistrer

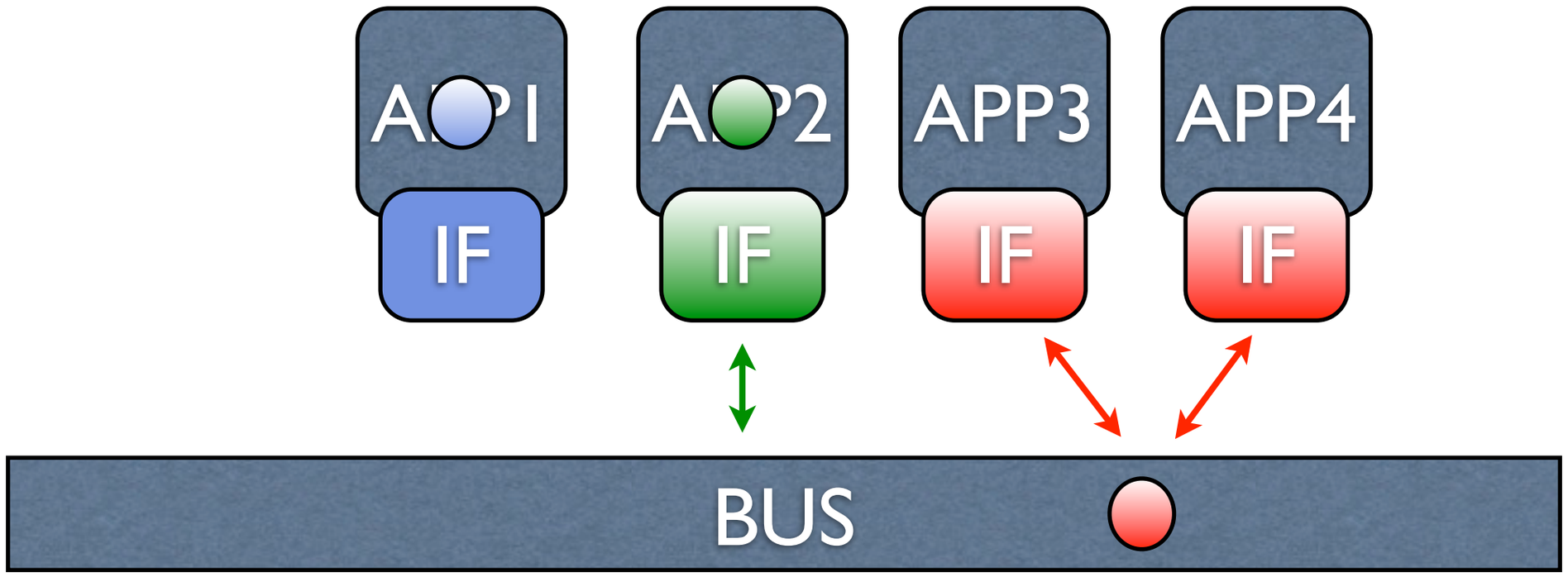
- `IntentFilter:` Action,Type, Catégorie, ..
- Nommage explicite des Intents (doit être déclaré dans le Manifest.xml)
- Nommage implicite(pas besoin d'enregistrer dans Manifest.xml)



Framework de communication



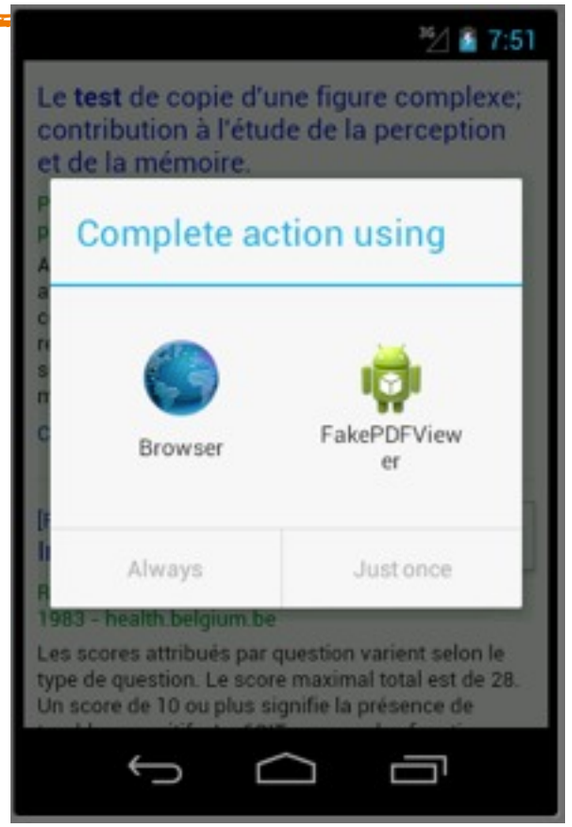
- **Une application doit s'enregistrer**
  - `IntentFilter:` Action,Type, Catégorie, ..
  - Nommage explicite des Intents (doit être déclaré dans le Manifest.xml)
  - Nommage implicite(pas besoin d'enregistrer dans Manifest.xml)

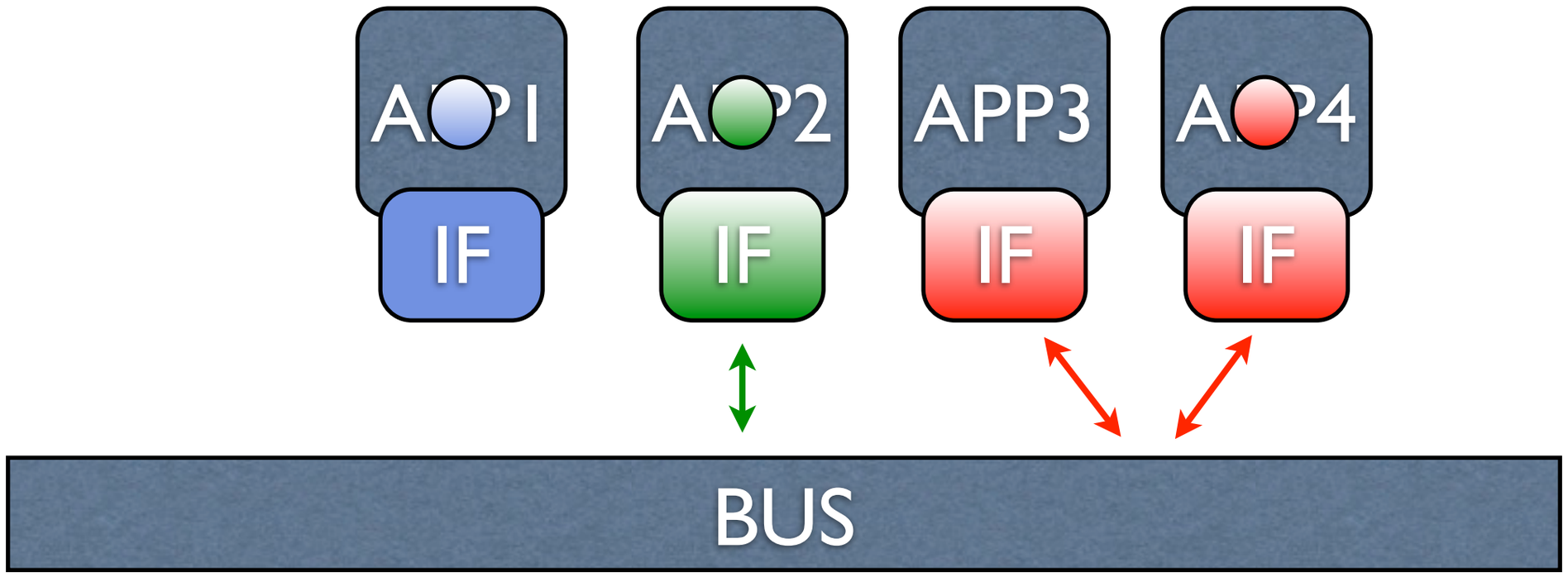


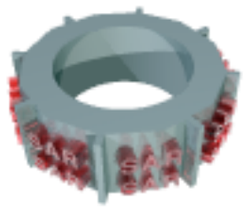
Framework de communication

Une application doit s'enregistrer

- `IntentFilter:` Action,Type, Catégorie, ..
- Nommage explicite des Intents (doit être déclaré dans le Manifest.xml)
- Nommage implicite(pas besoin d'enregistrer dans Manifest.xml)



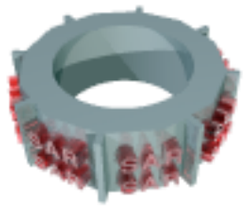




Désignation explicite


```
Intent intent = new Intent(this,MyActivity.class);
startActivityForResult(intent, requestCode);
```

- **Lancer une Activity :**
  - `startActivity(intent)`
  - `startActivityForResult(intent, requestCode)`
- **Lancer un Service :**
  - `startService(intent)`
  - `bindService(intent ...)`



Désignation implicite

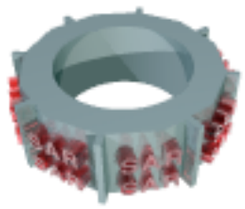
- Toutes les applications fournissant l'IntentFilter spécifié sont potentiellement activables

- Appeler

```
Intent goCall = new Intent(Intent.ACTION_DIAL,
                           Uri.parse("tel:0123456789"));
startActivity(goCall);
```

- Ouvrir une URL

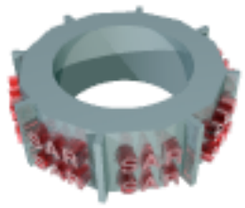
```
Intent goWeb = new Intent(Intent.ACTION_VIEW,
                          Uri.parse("http://..."));
startActivity(goWeb);
```



Framework des Intent

◉ Des Extras : données de l'Intent

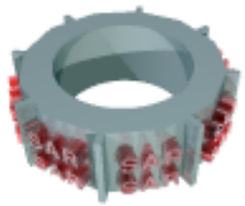
- Mapping (clé / valeur)
  - `Intent.putExtra(String key,TYPE value)`
  - `TYPE Intent.getExtra(String key)`
  - `boolean Intent.hasExtra(String key)`
- URI
  - `Intent.setData(Uri uri)`
  - `Intent.getData(Uri uri)`
- Data « Parceable »
  - Passer des objets à des Intent
  - `Intent.getParcelableExtra(Uri uri)`



Navigation dans les activités

◉ D'une Activity à une autre ?

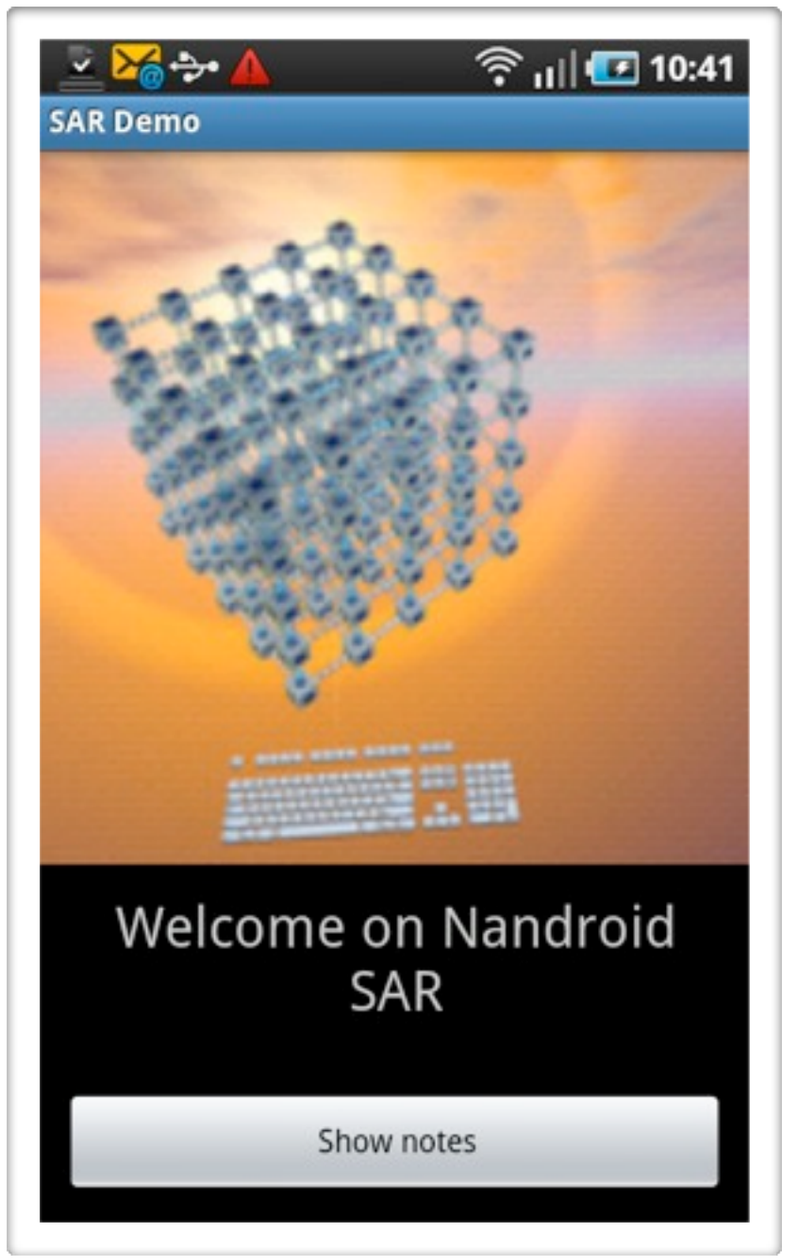
- Activity initiateur
  - **1) Création d'un Intent (ion)**
    - Provenance / Destinataire / données (URI)
    - `Intent intent = new Intent(this,StudentList.class)`
    - `intent.putExtra(String key, String value) ...`
  - **2) Envoyer l'Intent (ion)**
    - `startActivity(Intent intent)`
    - `startActivityForResult(Intent intent,int request)`
  - **3) Fin d'exécution**
    - `protected void onActivityResult(int requestCode, int resultCode, Intent data)`
    - `String data.getExtra(String key) ...`



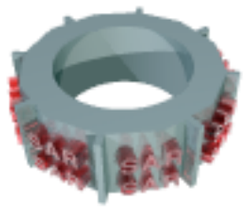
Navigation dans les activités

- Activity récepteur
  - 1) Récupérer l'Intent (ion)
    - Intent `getIntent()`
  - 2) Renvoyer un Intent (ion)
    - `void setIntent(Intent intent)`
    - `void finish(int requestCode)`

Navigation : exemple




```
public void onCreate(Bundle b) {
    super.onCreate();
    /* Bind the XML view */
    setContentView(R.layout.main);

    /* Link onClick operation */
    Button b = findViewById(R.id.Button01);
    button.setOnClickListener(this);
}

public void onClick(View v){
  /* Launch new activity */
  Intent intent = new
     Intent(SARDemo.this,StudentList.class);
  startActivity(intent);
}
```



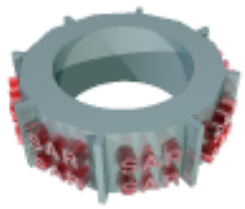
Les BroadcastReceiver

Classe pour récupérer les Intents envoyés

- `onReveive():` permet de récupérer l'Intent
- `abortBroadcast():` arrête la diffusion d'un évènement

Pour s'abonner aux événement systèmes

- Modification des permissions dans le Manifest.xml
  - `<uses-permission android:name="android.permission.RECEIVE_SMS" />`
  - `<uses-permission android:name="android.permission.READ_SMS" />`



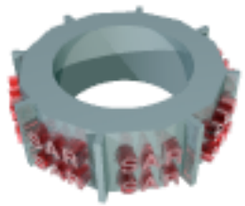
Récupérer SMS

Connection au simulateur

- telnet 127.0.0.1 5554
- sms send +12345 I am saying hello using SMS.

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        registerReceiver( new BroadcastReceiver() {

          @Override
          public void onReceive(Context context, Intent intent) {
              Toast.makeText(context, "SMS RECEIVED!!", Toast.LENGTH_SHORT).show();
          }
        }, new IntentFilter("android.provider.Telephony.SMS_RECEIVED"));
    }
}
```

Service



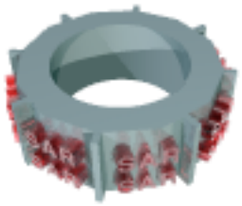
Les Services

Description

- Pas d'interface utilisateur
- Période d'exécution indéfinie
  - Lecteur MP3, synchronisation des flux RSS, etc.
- Déclaration dans le Manifest.xml

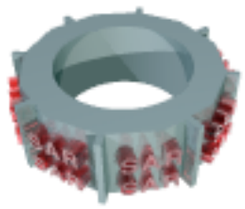
Quelle utilité?

- Demande au système d'exécuter une tâche en arrière plan
- Un moyen d'exposer des fonctionnalités aux autres applications



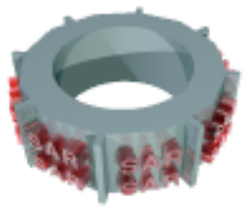
Les Services

- **Tâche de fond**
  - Ne comporte aucune entité visuelle
    - **Peut lancer une activité**
    - **Peut lancer une notification**
  - Cycle de vie simplifié
- **Attention !**
  - Un service **n'est pas** un processus séparé !
  - Un service **n'est pas** un thread !
- **Le système se charge uniquement d'instancier l'objet et d'appeler les callback**
  - **C'est le dévelopeur qui est responsable du comportement : création d'un thread dédié, etc.**

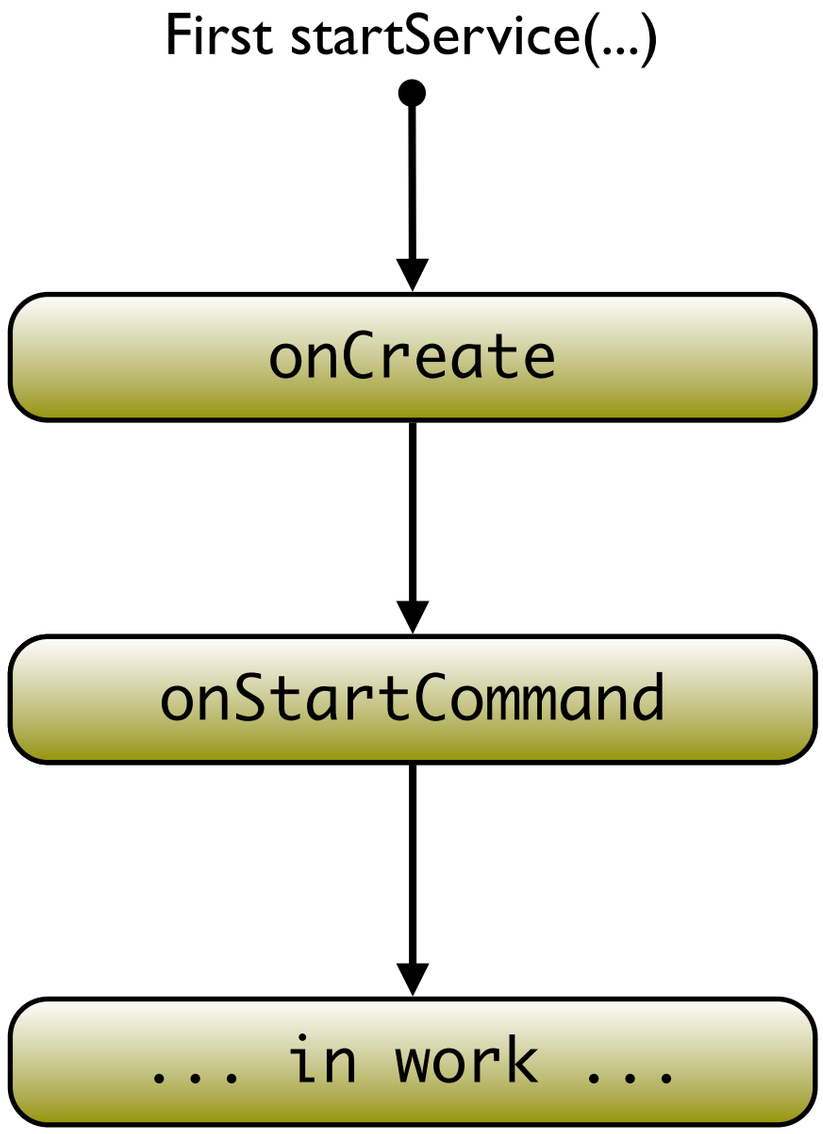


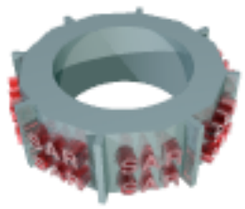
Les Services : modes d'exécutions

- **Mode non borné** `startService:`
  - 1 instance : exécution en continue ...
    - **Communication « one way »**
    - **startService : paramètres dans INTENT**
- **Mode borné via** `bindService:`
  - 1 instance : exécution tant qu'il y a une connexion
  - Exécution d'opération sur le Service
- **Possibilité de faire les 2**

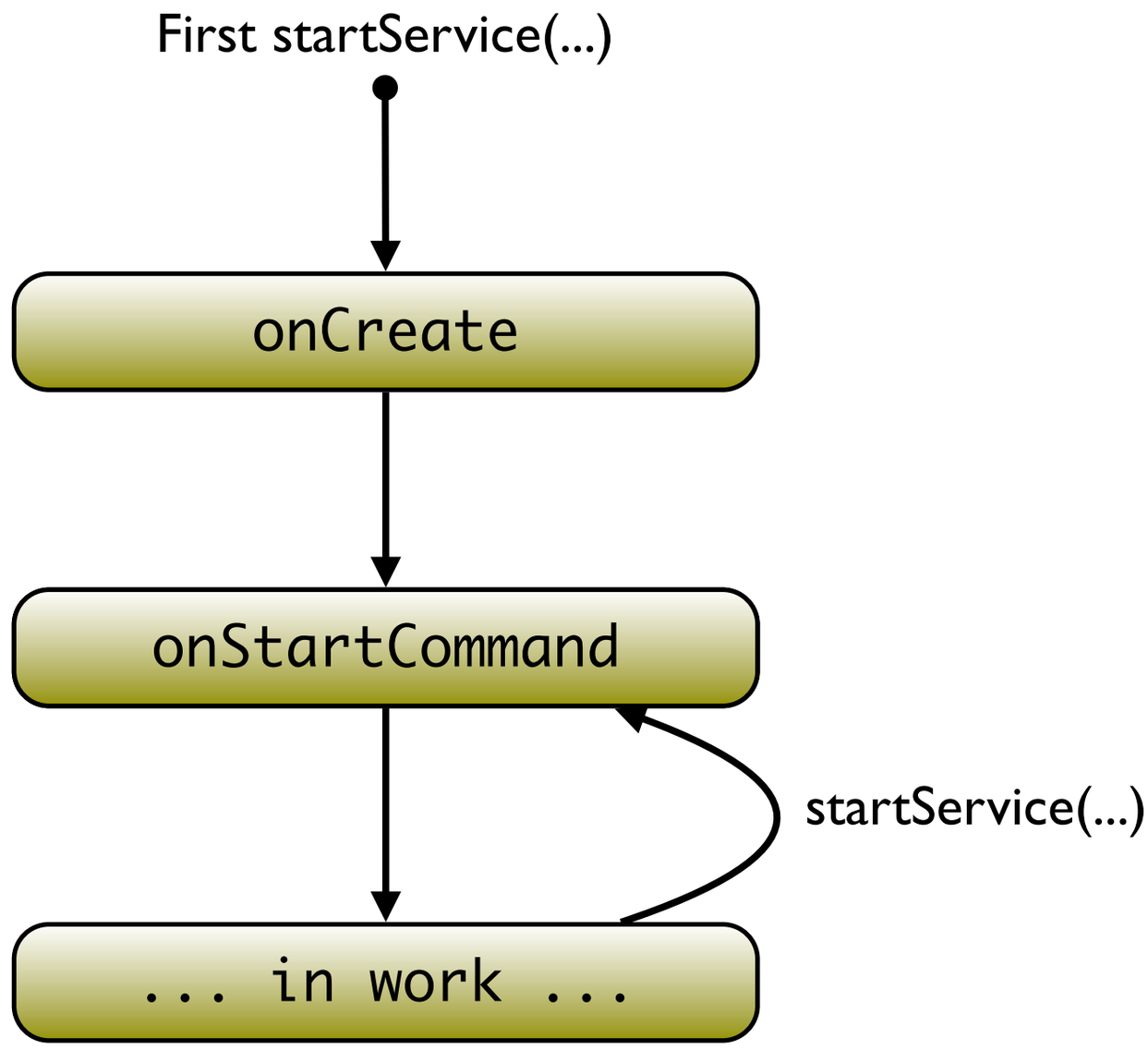


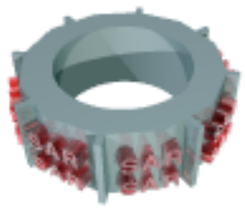
Mode non borné : cycle de vie



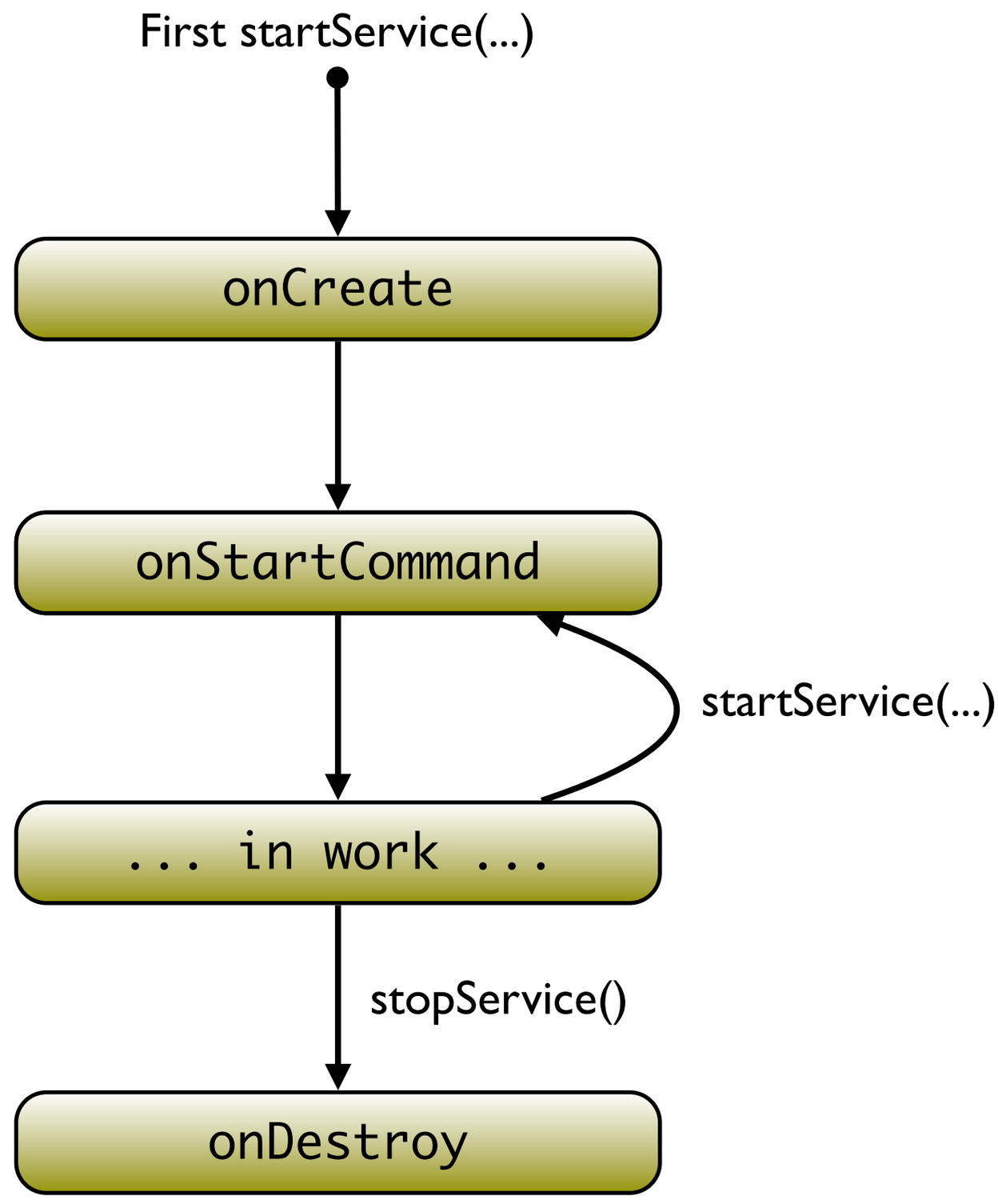


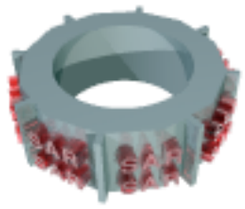
Mode non borné : cycle de vie





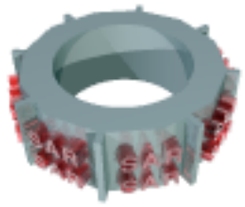
Mode non borné : cycle de vie





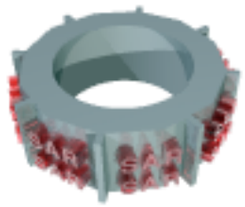
Service non borné : exemple


```
public class MyHelloService extends Service {
    private static final String TAG = "MyHelloService";
    MediaPlayer player;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        Toast.makeText(this, "My Service Created", Toast.LENGTH_LONG).show();
        Log.d(TAG, "onCreate");
        player = MediaPlayer.create(this, R.raw.lilyaaron);
        player.setLooping(false); // Set looping
    }

    @Override
    public void onDestroy() {
        Toast.makeText(this, "My Service Stopped", Toast.LENGTH_LONG).show();
        Log.d(TAG, "onDestroy");
        player.stop();
    }

    @Override
    public void onStart(Intent intent, int startid) {
        Toast.makeText(this, "My Service Started", Toast.LENGTH_LONG).show();
        Log.d(TAG, "onStart");
        player.start();
    }
}
```



Service non borné (suite)

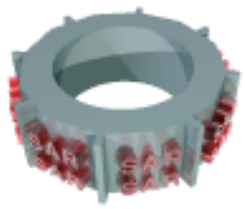

```
public class MainActivity extends Activity implements OnClickListener {
    private static final String TAG = "ServicesDemo";
    Button buttonStart, buttonStop;

    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      buttonStart = (Button) findViewById(R.id.buttonStart);
      buttonStop = (Button) findViewById(R.id.buttonStop);

      buttonStart.setOnClickListener(this);
      buttonStop.setOnClickListener(this);
    }

    public void onClick(View src) {
      switch (src.getId()) {
      case R.id.buttonStart:
        Log.d(TAG, "onClick: starting srvice");
        startService(new Intent(this, MyHelloService.class));
        break;
      case R.id.buttonStop:
        Log.d(TAG, "onClick: stopping srvice");
        stopService(new Intent(this, MyHelloService.class));
        break;
      }
    }
}
```



Service borné

◉ Notion de Client / Serveur

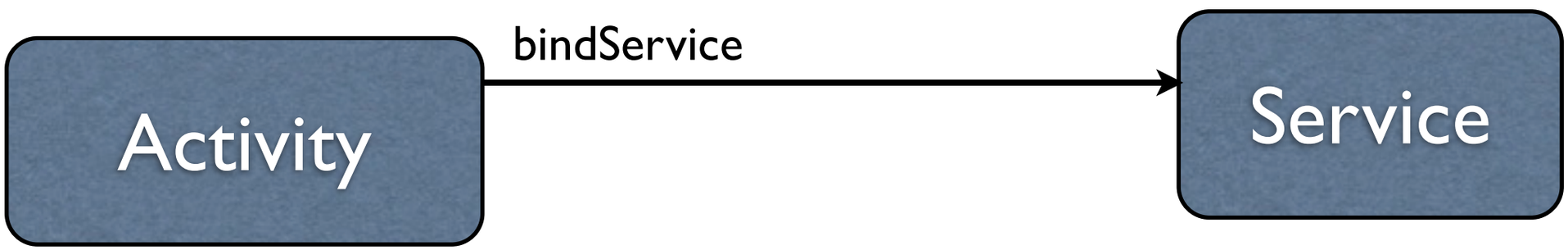
- Communication inter-processus
- Notion de STUB : Client / Serveur (IPC)



Service borné

◉ Notion de Client / Serveur

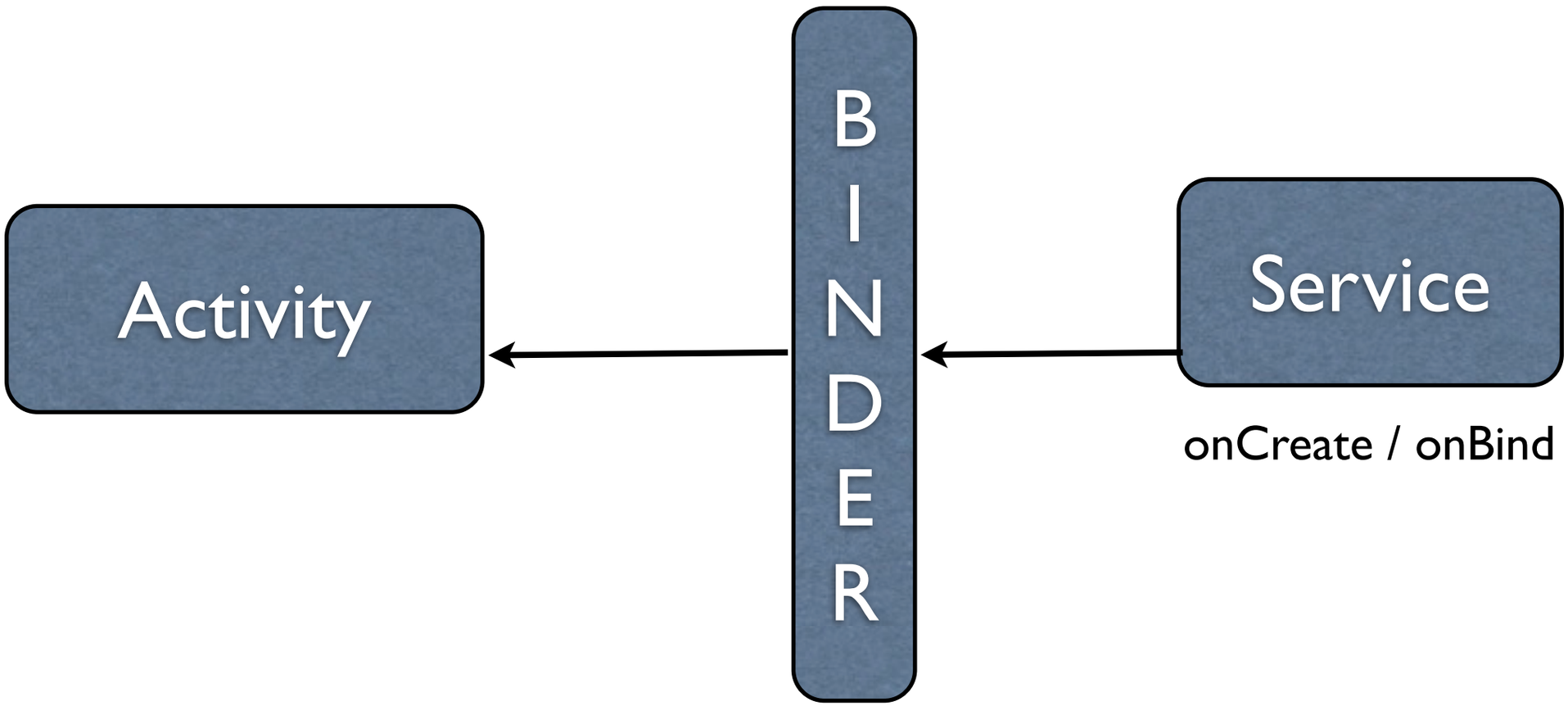
- Communication inter-processus
- Notion de STUB : Client / Serveur (IPC)



Service borné

Notion de Client / Serveur

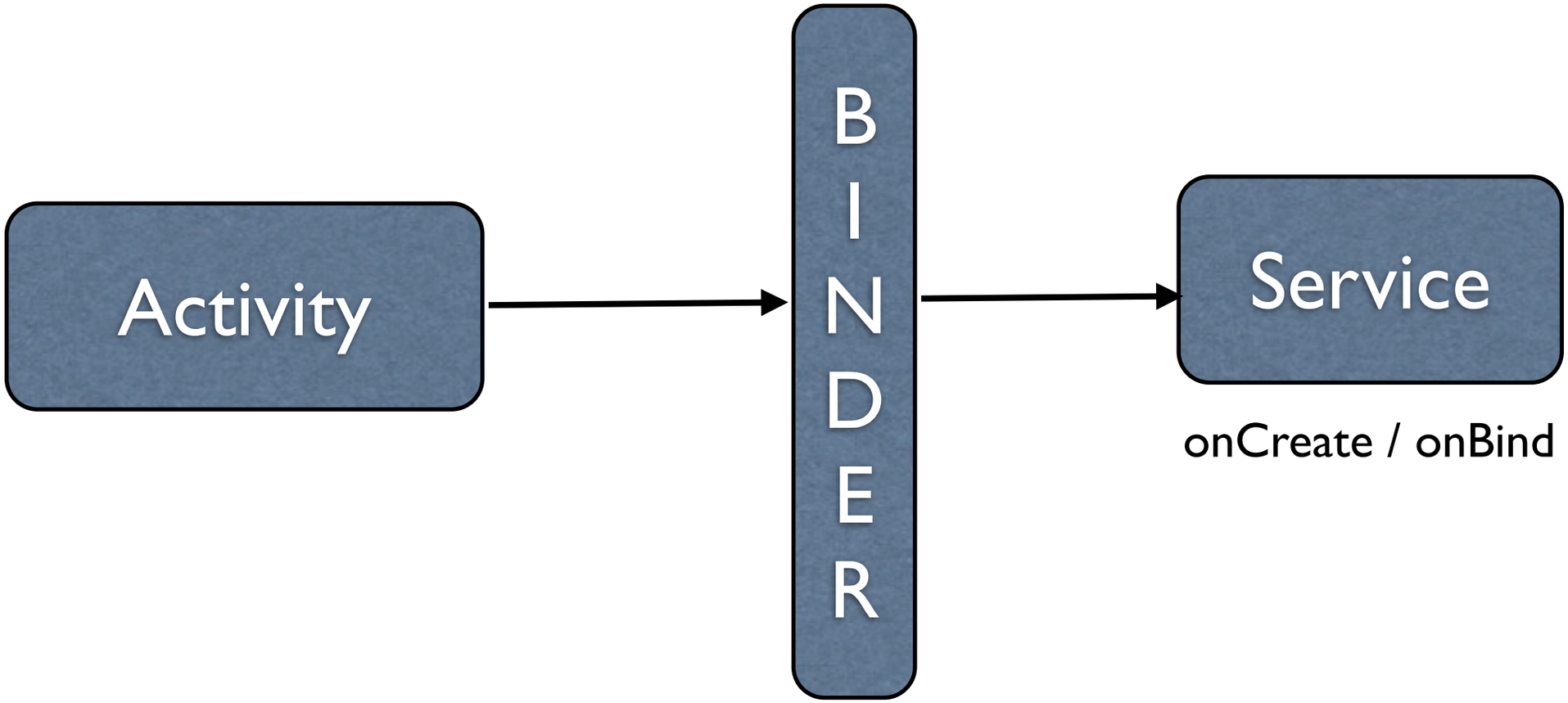
- Communication inter-processus
- Notion de STUB : Client / Serveur (IPC)

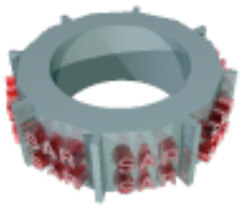


Service borné

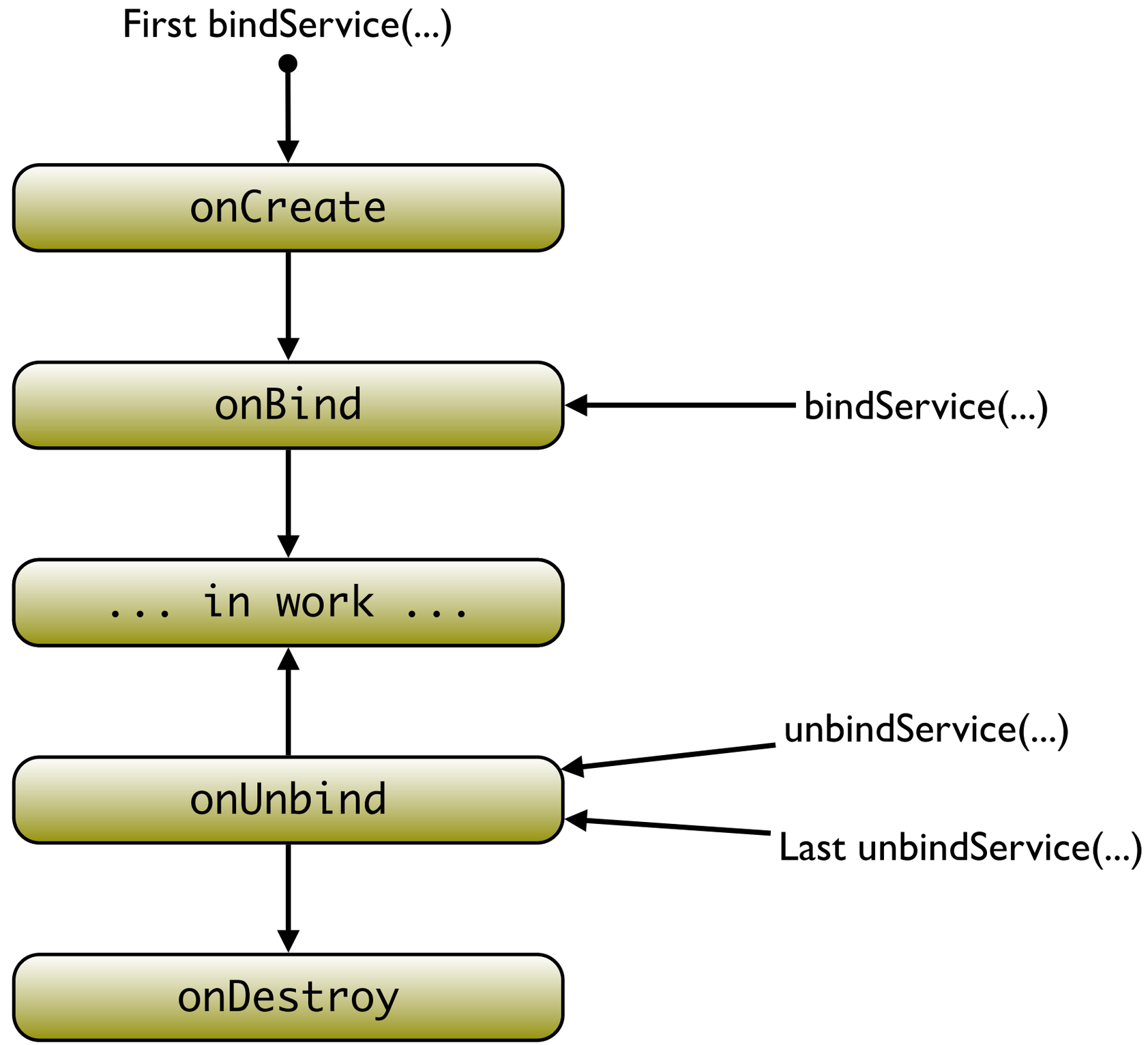
Notion de Client / Serveur

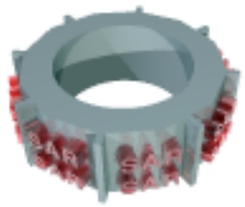
- Communication inter-processus
- Notion de STUB : Client / Serveur (IPC)





Mode borné : cycle de vie

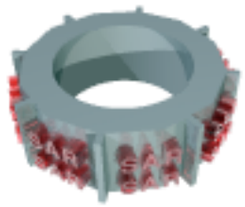




Service borné : exemple


```
public class LocalBinder extends Binder {
    int id = 0;
    public int getIntInstance() {
        return ++id;
    }
}
```

```
public class IntegerBinderService extends Service{
    IBinder mBinder = new LocalBinder();
    private static final String TAG = "BinderPlayingService";

    @Override
    public IBinder onBind(Intent intent) {
        Toast.makeText(this, "My Service bind", Toast.LENGTH_LONG).show();
        return mBinder;
    }

    @Override
    public boolean onUnbind(Intent intent) {
        Toast.makeText(this, "My Service Unbind", Toast.LENGTH_LONG).show();
        return true;
    }

    @Override
    public void onCreate() {
        Toast.makeText(this, "My Service Created", Toast.LENGTH_LONG).show();
    }
}
```



Service borné : exemple

- Creation du mécanisme de connection
  - **onServiceConnected: premier appel au Service**
  - **onServiceDisconnected: cloture de Connexion au Service**

```
ServiceConnection mConnection = new ServiceConnection() {

    public void onServiceDisconnected(ComponentName name) {
        Toast.makeText(MainActivity.this, "Service is disconnected",
                       Toast.LENGTH_SHORT).show();
        mbound = false;
    }

    public void onServiceConnected(ComponentName name, IBinder service) {
        Toast.makeText(MainActivity.this, "Service is connected",
                       Toast.LENGTH_SHORT).show();
        mbound = true;
        myService = (LocalBinder) service;
    }
};
```

Service borné : exemple

```
public void onClick(View src) {
   switch (src.getId()) {

   case R.id.buttonBind:
     Intent mIntent = new Intent(this, IntegerBinderService.class);
     bindService(mIntent, mConnection, BIND_AUTO_CREATE);
     break;

   case R.id.buttonUnbind:
     if (mBound)
        unbindService(mConnection);
     break;

   case R.id.buttonRandom:
       if (mbound)
           Toast.makeText(MainActivity.this, "Random is " + myService.getIntInstance() ,
                          Toast.LENGTH_SHORT).show();
       else
           Toast.makeText(MainActivity.this, "Service not Bind",
                          Toast.LENGTH_SHORT).show();
       break;
}
```

Tâche asynchrone

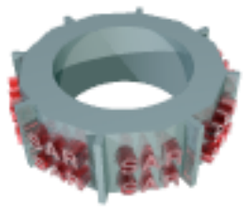
Les tâches asynchrones

Pourquoi ?

- Dans un service pour retourner le plus rapidement possible à l'appelant
- Dans une activité, libération de l'appelant

Le Thread UI doit répondre vite sinon blocage

Pour gérer les traitements long dans les événements



AsyncTask : définition

- **Manipulation facile du Thread UI :**
  - Effectue des opération en arrière plan
  - Diffuse le résultat sur l'interface graphique sans gestion de handler
- **Ne doit pas remplacer les Threads :**
  - Il s'agit d'une facilité d'écriture
  - Pour les taches de quelques secondes
- **Défini par trois paramètres génériques :**
  - `Param:` le type du paramètre envoyé à la tâche pour le calcul
  - `Progress:` le type de l'unité d'avancement utilisée pour la notification
  - `Result:` le résultat du calcul de la tâche

THREAD - AsyncTask

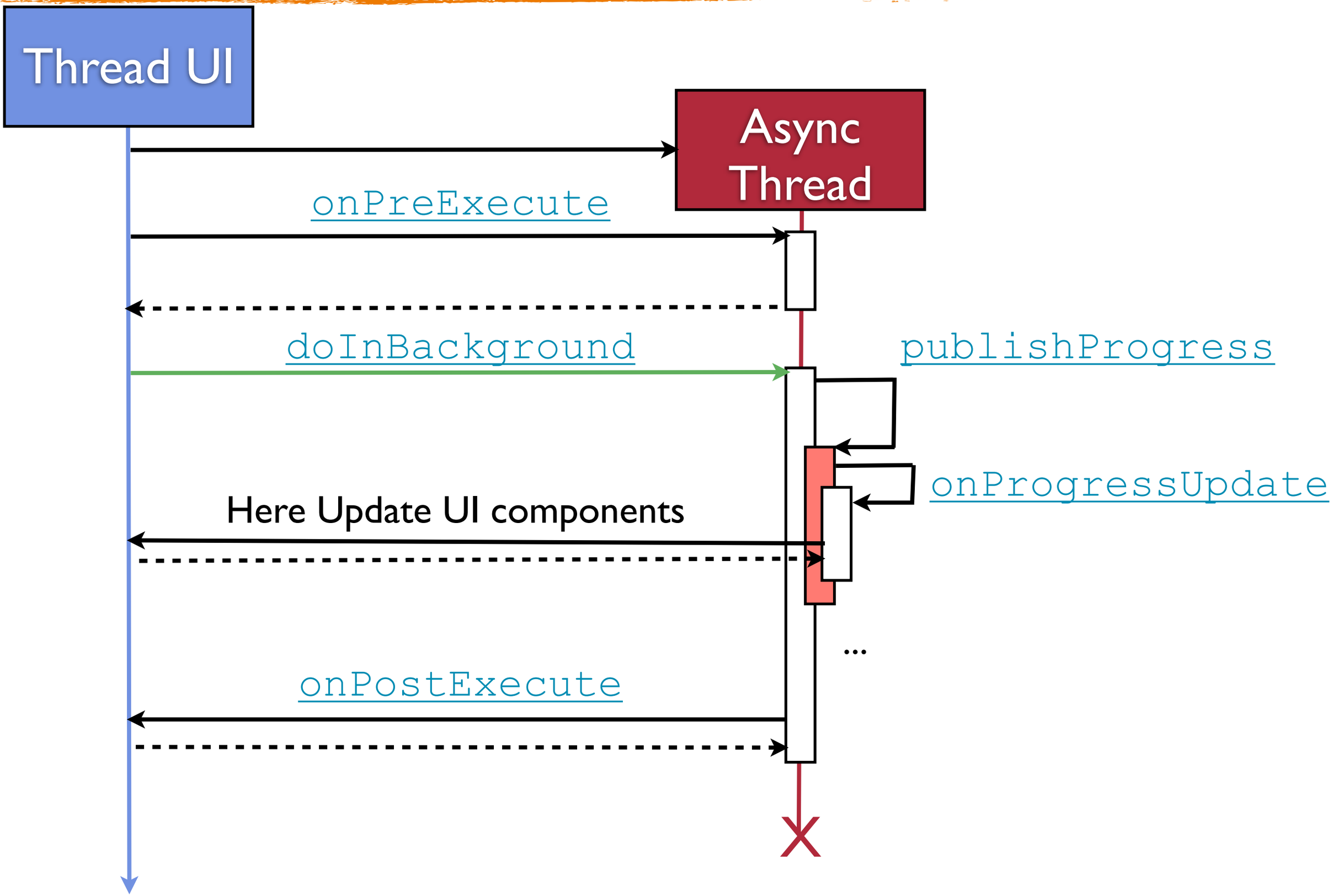
Thread UI

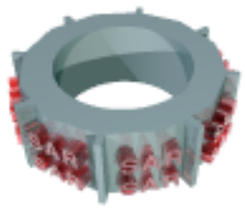
- Dans un événement ==> traitement long
- Répondre vite ... sinon blocage ? (Answer Not Responding)
- « L'application ne répond pas .... »

Solution ?

- Nouveau Thread
  - **Traitement + update**
  - **Attention ... Thread UI doit faire l'update**
- Mécanisme courant
  - **AsyncTask**

AsyncTask : cycle de vie

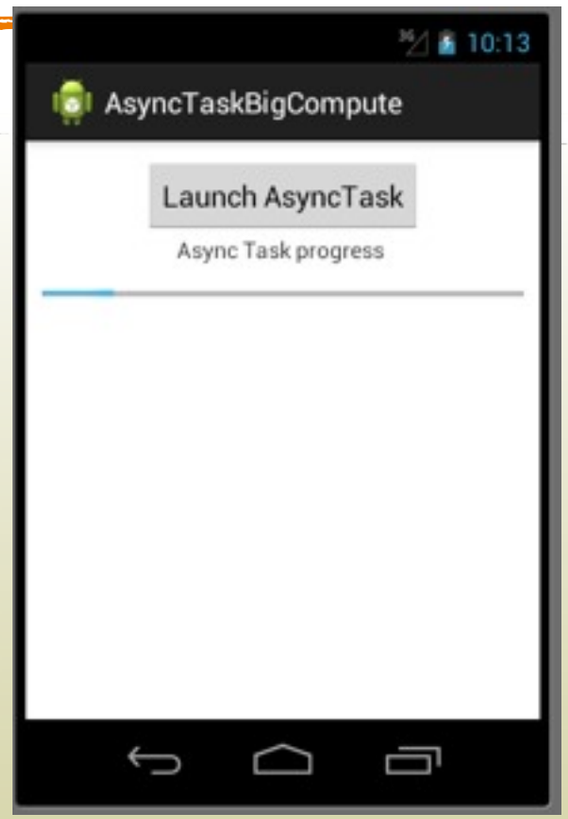




AsynTask : exemple

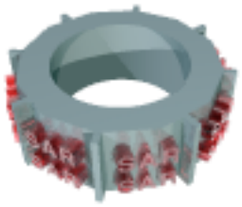

```
private class BigCompute extends AsyncTask<Void, Integer, Void>
    {
        protected void onPreExecute() {
          super.onPreExecute();
          Toast.makeText(getApplicationContext(), "Start Async Comp", Toast.LENGTH_LONG).show();
        }

        protected Void doInBackground(Void... arg0) {
            for (int progress = 0; progress <= 100; ++progress)
            {
                try{ Thread.currentThread().sleep(1000);}
                catch(Exception ie){}
                publishProgress(progress);
            }
            return null;
        }

        protected void onProgressUpdate(Integer... values){
            super.onProgressUpdate(values);
            mProgressBar.setProgress(values[0]);
        }

        protected void onPostExecute(Void result) {
            Toast.makeText(getApplicationContext(), "Async Task End", Toast.LENGTH_LONG).show();
        }
    }
```

AsynTask : exemple (suite)



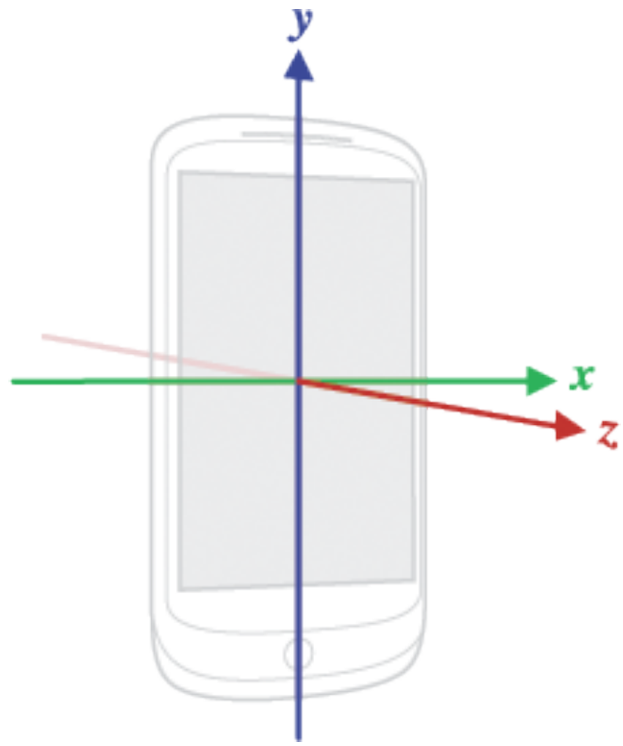

```
public class MainActivity  extends Activity {

    private ProgressBar mProgressBar;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mProgressBar = (ProgressBar) findViewById(R.id.pBAsync);
        Button  mButton = (Button) findViewById(R.id.btnLaunch);

        mButton.setOnClickListener(   new OnClickListener() {
            @Override
            public void onClick(View arg0) {
                BigCompute compute = new BigCompute();
                compute.execute();
            }
        });
    }

    /** The Code of BigCompute... */
```

Les spécificités matérielles

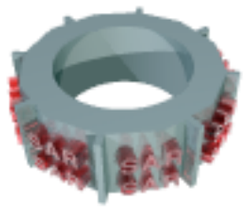


Les capteurs ...

- Définit pour un plan donné (x,y,z)
- Les principaux capteurs ...
  - **GYROSCOPE : Mesurer le vitesse angulaire (x,y,z)**
  - **MAGNETIQUE : Mesurer le champs magnétique (la boussole)**
    - x,y,z : en micro Tesla
  - **ORIENTATION : Mesurer l'inclinaison (x,y,z)**
    - values[0,1,2] Azimuth (x), Pitch(y), Roll(z) : en °
  - **ACCELEROMETRE : Mesurer la vitesse d'accélération (x,y,z)**
    - values[0,1,2] x,y,z : en m/s^2
  - **PROXIMITE : Mesurer la proximité**
    - values[0] : en cm
  - **LUMIERE : Mesurer la quantité de lumière reçue**
    - values[0] : en lux



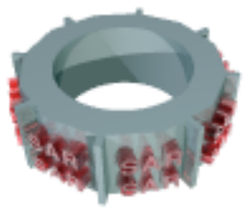
- Les capteurs ne sont pas tous disponibles ...



Les capteurs

Utilisation via le SensorManager

- 1) Récupérer SensorManager
  - `getSystemService(Context.SENSOR_SERVICE)`
- 2) Récupérer le Type de capteur
  - `Sensor.TYPE_TEMPERATURE`
- 3) Vérifier qu'il existe sur le mobile (La liste doit > 0)
- 4) Ajouter un Handler
  - `SensorEventListener`
  - **Démarrage du capteur (dans onResume)**
- 5) Recevoir/Traiter les événements
  - `SensorEvent.values[]`
- 6) Retirer le Handler
  - **NE PAS OUBLIER dans la méthode** `onPause`



Les capteurs ...

- Dans cctivity (onCreate/onStart)

```
SensorManager accM   =  getSystemService(Context.SENSOR_SERVICE);
List<Sensor> sensors = accM.getSensorList(Sensor.TYPE_ACCELEROMETER);
if (sensors.size() > 0){
  Log.d("MTouchGraphic","ACCELEROMETER[" + sensors.get(0).getVendor()+"]");
  accM.registerListener(this, sensors.get(0), SensorManager.SENSOR_DELAY_UI);
}else{
  Log.d("MTouchGraphic","ACCELEROMETER NOK");
}
```

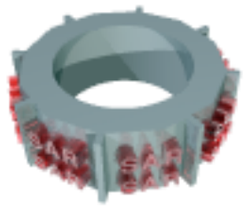
- Recevoir les événements SensorEventListener

```
public void onSensorChanged(SensorEvent event) {
     x = event.values[SensorManager.DATA_X];
     y = event.values[SensorManager.DATA_Y];
     z = event.values[SensorManager.DATA_Z];
}
```

- Arrêter les Sensor (onPause)

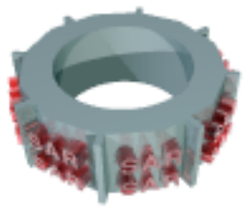
```
public void onPause() {
     accM.unregisterListener(this, sensors.get(0);
}
```

Conclusion



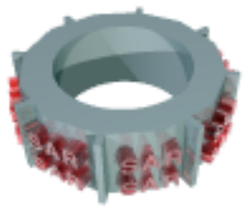
iOS / Android

	Android	iOS
Langage	Java	Objective C
IDE	Eclipse/ADT	XCode/MacOS
Terminaux	Tous	Apple
Market	PlayStore/aptoïde	iTunesStore
Publication	gratuit à 25$	99$
Securité	sur le PlayStore	Tout le temps



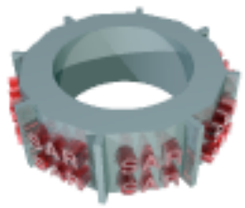
iOS / Android

	Android	iOS
ShakeMotion		`UIResponder`
Detection Gestes	`OnGesture Listener`	`UIResponder`
Exploration FS	`FileStream`	`NSFile Manager`
Rotation	Persistence Légère	Recalcul des Coordonnées



iOS / Android

	Android	iOS
Préférences	`PreferenceManager`	`NSUserDefault`
Notifications	`NotificationManager`	`NSNotification`
Widget	`AppWidget`	
Service	`Service`	`Long Running Background Task`
Tâches Asynchrones	`AsyncTask`	`(Dispatch_queue_t)`

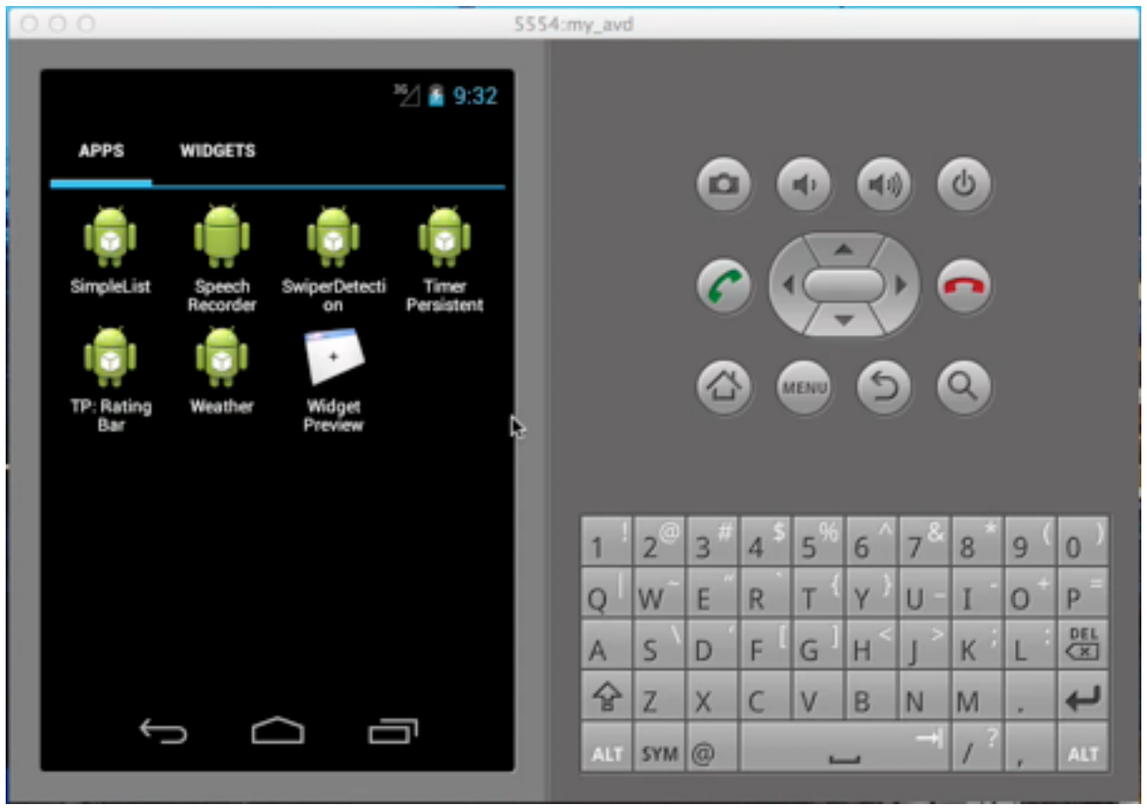


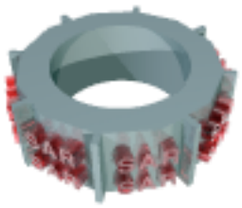
FIN ....

A vos machines !!!

Rating Bar App

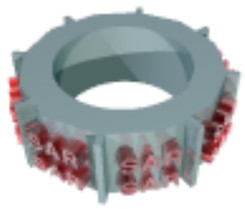
- http://pagesperso-systeme.lip6.fr/Etienne.Renault/teaching/ppm/2012/BundleTPAndroid.zip





Rating Bar App

- http://pagesperso-systeme.lip6.fr/Etienne.Renault/teaching/ppm/2012/BundleTPAndroid.zip



MyLittleMarket : Projet

- http://pagesperso-systeme.lip6.fr/Etienne.Renault/teaching/ppm/2012/projet-android.pdf

◉ Parser du XML

◉ Offrir une interface utilisateur agéable