

Interfaces Homme - Machine

Cours 6 Applications mobiles et programmation Androïd

1. Introduction aux applications mobiles
2. Principes généraux Androïd
3. Conception d'interfaces simples sous Androïd

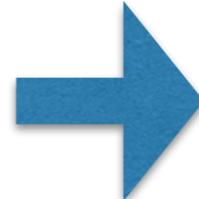
I. Introduction aux applications mobiles

Applications natives / HTML5 ?

Grand débat entre développeurs / entreprises / etc

Dépend de plusieurs facteurs :

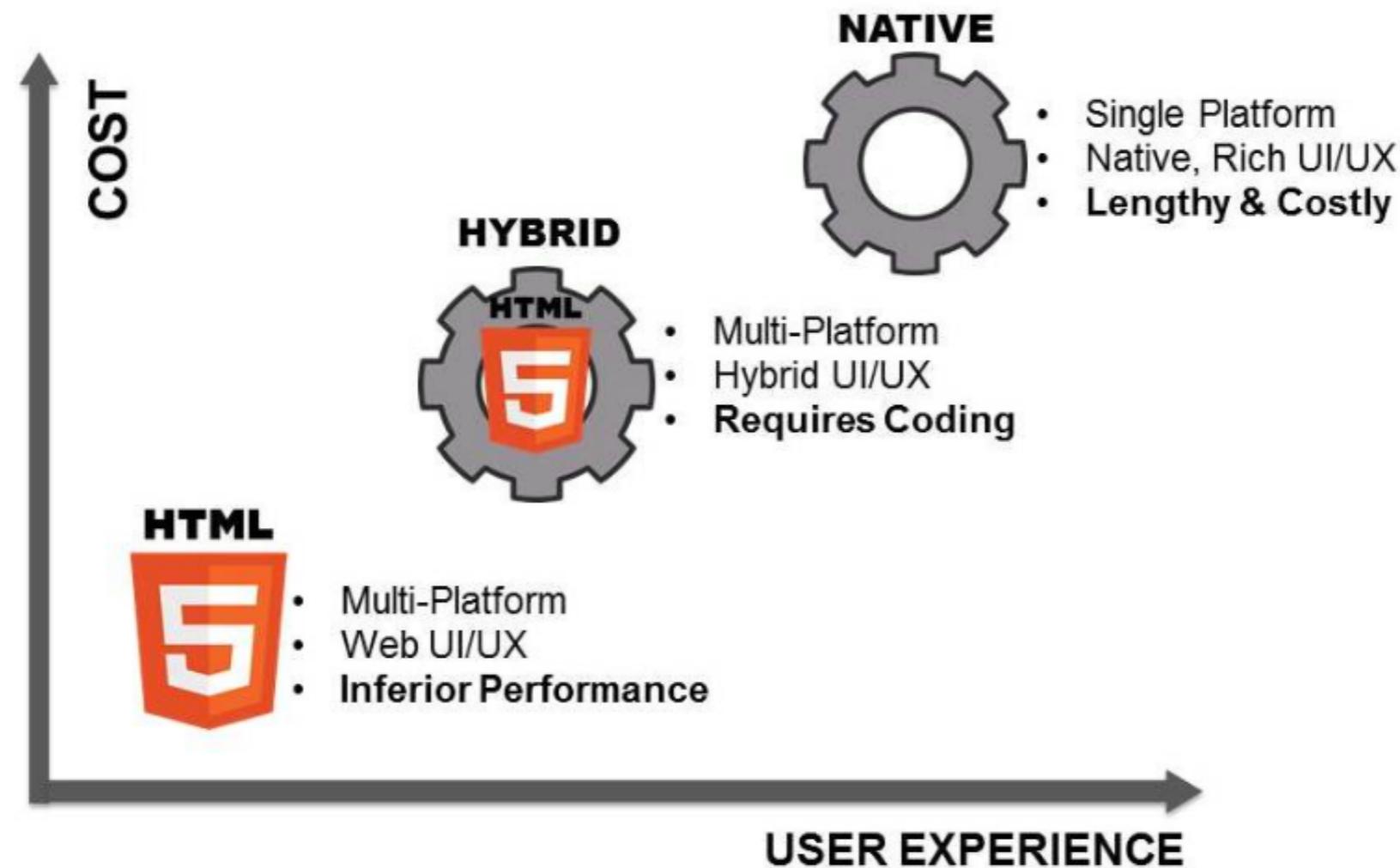
- Facilité de développement
- Durabilité des applications
- Possibilités / limitations de la plateforme
- Compatibilité avec l'OS
- Diffusion
- Stratégie globale de l'entreprise
- Rapport coût/bénéfice
- Ressenti utilisateur
- Etc



Pas uniquement une question technique

I. Introduction aux applications mobiles

Applications natives / HTML5 ?

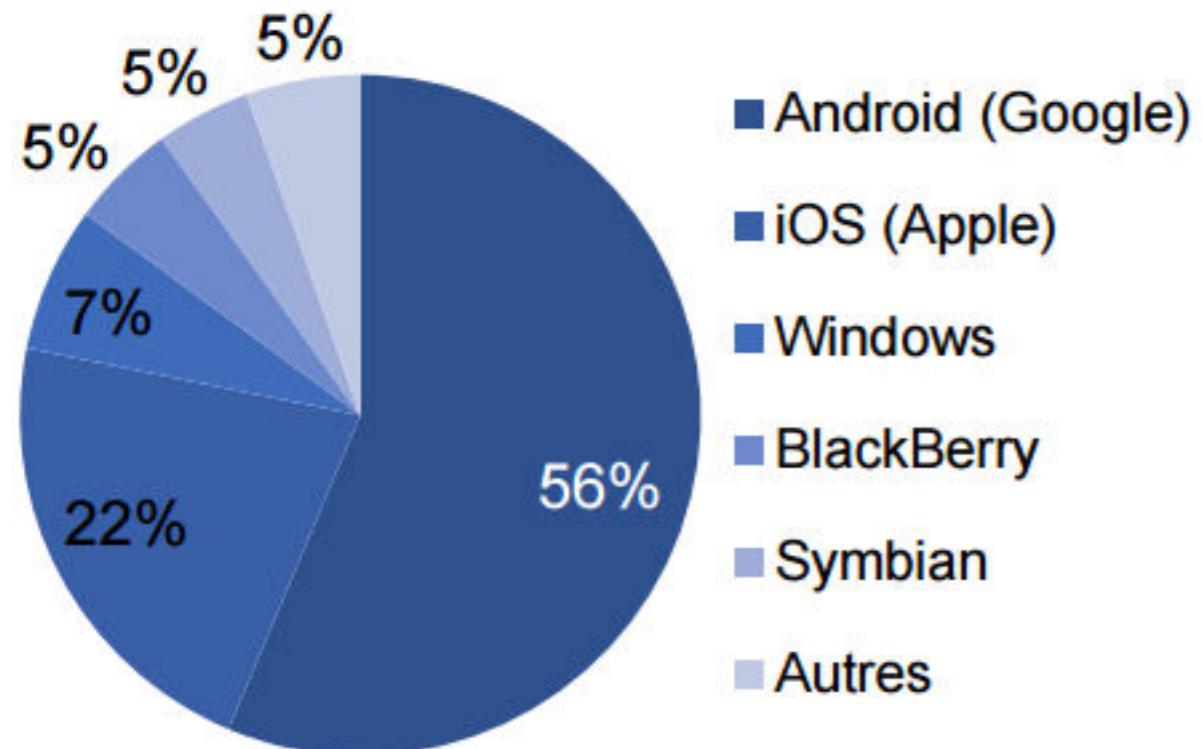


On a choisi de vous faire faire du natif, plus précisément : Android

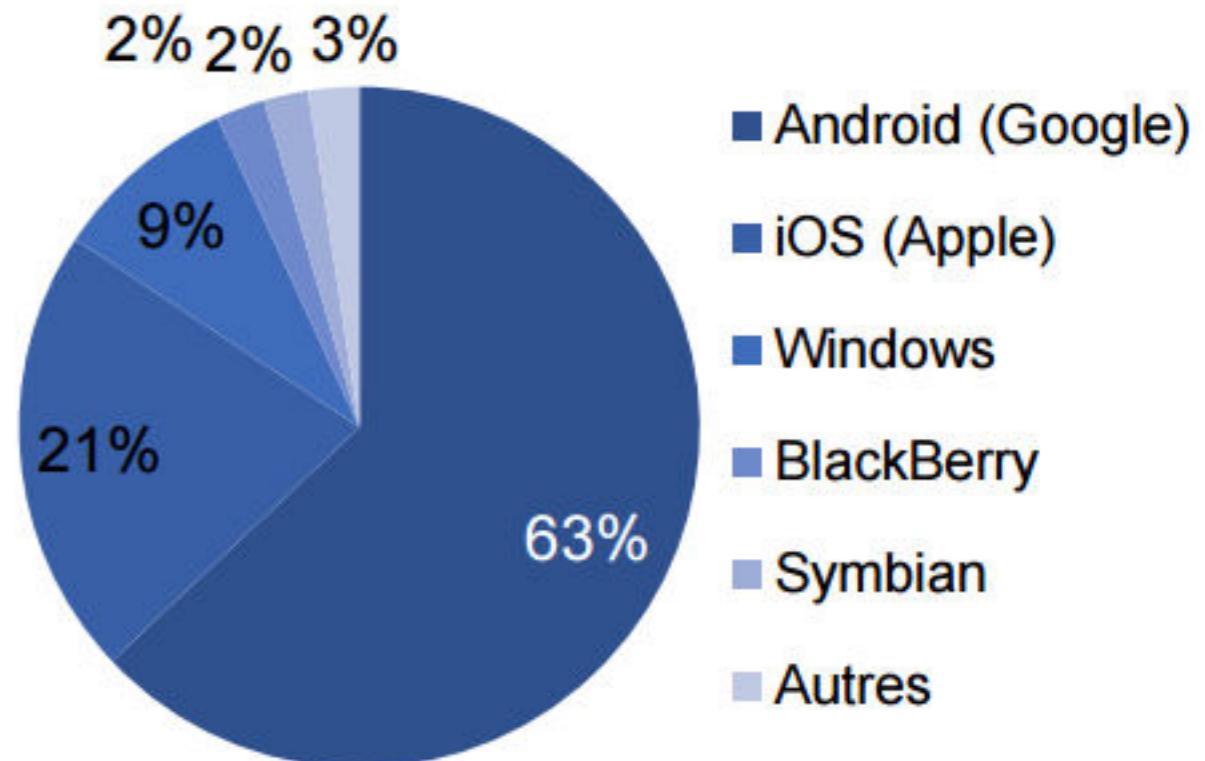
I. Introduction aux applications mobiles

Pourquoi Androïd en 4IF ?

Base installée déc. 2013



Base installée jan. 2015



C'est flexible : tablette, smartphone, smartwatch, tv, etc

C'est « facile » : beaucoup de ressources en ligne

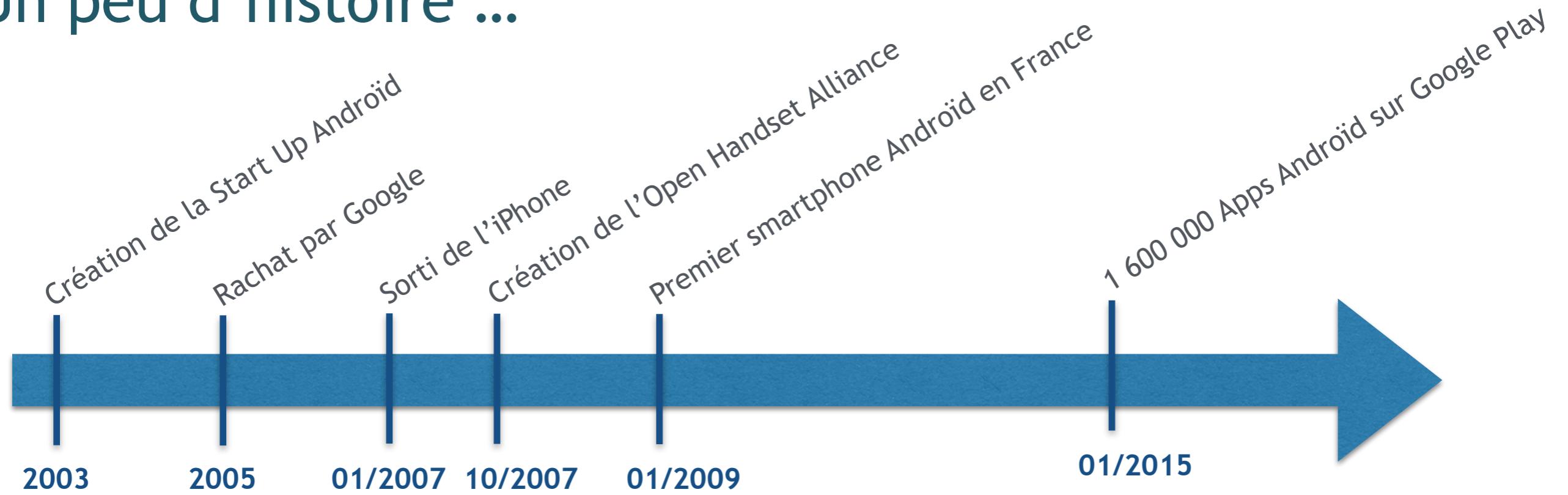
C'est globalement du Java... donc vous êtes supposés maîtriser !

On se dit qu'il faut que vous ayez fait au moins un projet sur une technologie mobile avant de quitter l'INSA

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Un peu d'histoire ...



Open Handset Alliance

- 35 entreprises dans le domaine du mobile, dont Google (+ de 80 aujourd'hui)
- Constructeurs, opérateurs, éditeurs de logiciels, etc
- But : Fournir une plate-forme ouverte et complète pour les appareils mobiles
- Sortie du premier SDK Androïd

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Historique des versions

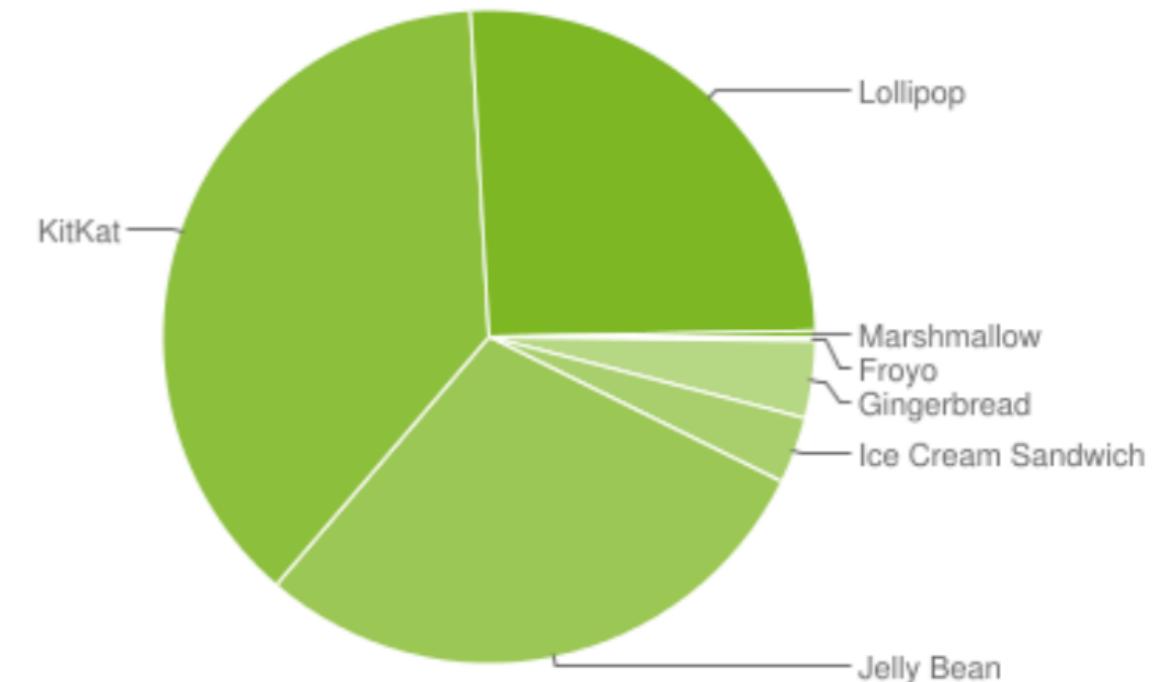
| Nom | Version | API | Date |
|--------------------|---------------|-------|---------|
| Android | 1.0 | 1 | 09/2008 |
| Petit four | 1.1 | 2 | 02/2009 |
| Cupcake | 1.5 | 3 | 04/2009 |
| Donut | 1.6 | 4 | 09/2009 |
| Eclair | 2.0-2.1 | 5-7 | 10/2009 |
| Froyo | 2.2 - 2.2.3 | 8 | 05/2010 |
| Gingerbread | 2.3 - 2.3.7 | 10 | 12/2010 |
| Honeycomb | 3.0 - 3.2.6 | 11-13 | 02/2011 |
| Ice Cream Sandwich | 4.0.1 - 4.0.4 | 15 | 10/2011 |
| Jelly Bean | 4.1 - 4.3.1 | 16-18 | 07/2012 |
| KitKat | 4.4 - 4.4.4 | 19 | 10/2013 |
| Lollipop | 5.0 | 21-22 | 10/2014 |
| Marshmallow | 6.0 | 23 | 05/2015 |

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Répartition des versions en novembre 2015

| Version | Codename | API | Distribution |
|---------|--------------------|-----|--------------|
| 2.2 | Froyo | 8 | 0.2% |
| 2.3.3 - | Gingerbread | 10 | 3.8% |
| 2.3.7 | | | |
| 4.0.3 - | Ice Cream Sandwich | 15 | 3.3% |
| 4.0.4 | | | |
| 4.1.x | Jelly Bean | 16 | 11.0% |
| 4.2.x | | 17 | 13.9% |
| 4.3 | | 18 | 4.1% |
| 4.4 | KitKat | 19 | 37.8% |
| 5.0 | Lollipop | 21 | 15.5% |
| 5.1 | | 22 | 10.1% |
| 6.0 | Marshmallow | 23 | 0.3% |



Évolution des versions très rapide

Veille régulière nécessaire

Globalement, les principes restent

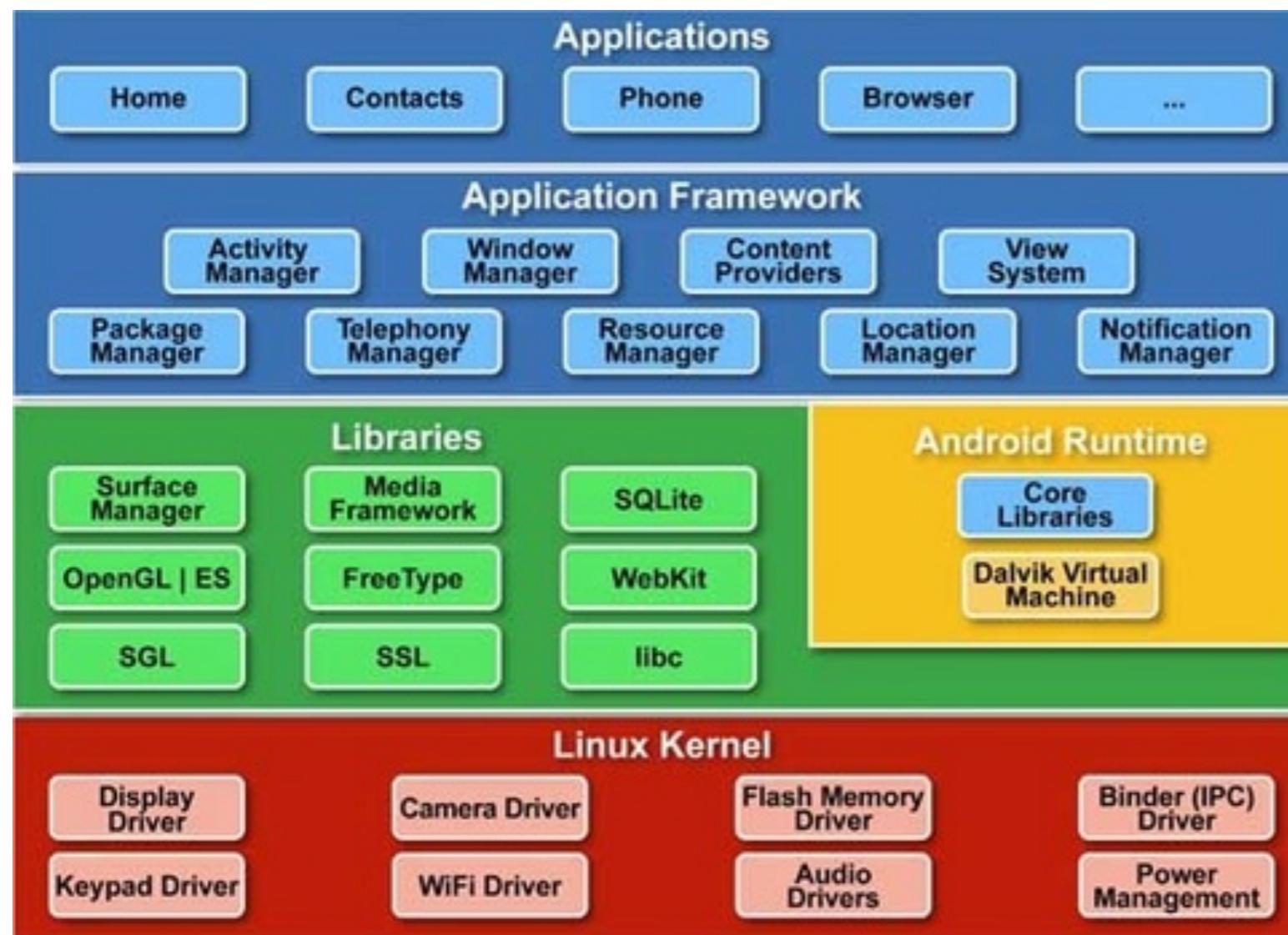
Data collected during a 7-day period ending on November 2, 2015.

Any versions with less than 0.1% distribution are not shown.

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Architecture et environnement Android



- Architecture en 5 couches
- Noyau Linux (3.4.0 / 3.10)
- Bibliothèques / Applications
- Machine Virtuelle Dalvik
 - Java
 - Mais différent de la JVM « standard »
- SDK et API Android
- Un IDE : Android Studio
- Plugin Eclipse n'est plus maintenu

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Création d'un nouveau projet sur Android Studio

New Project
Android Studio

Configure your new project

Application name: My Application

Company Domain: llaporte.example.com

Package name: com.example.llaporte.myapplication

Project location: /Users/llaporte/AndroidStudioProjects/MyApplication

Nom de votre application, qui apparaîtra sur l'appareil et sur Google Play

Domaine de votre entreprise, école, domaine perso, etc
Permet de générer le package name

Edit

Nom du package où se trouve le projet
Sert d'identifiant sur Google Play

Cancel Previous Next Finish

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Création d'un nouveau projet sur Android Studio



Select the form factors your app will run on

Different platforms require separate SDKs

| | | |
|--|-------------|-------------------------------------|
| <input checked="" type="checkbox"/> Phone and Tablet | Minimum SDK | API 10: Android 2.3.3 (Gingerbread) |
| Lower API levels target more devices, but have fewer features available. By targeting API 10 and later, your app will run on approximately 99,5% of the devices that are active on the Google Play Store. Help me choose.. | | |
| <input type="checkbox"/> TV | Minimum SDK | API 21: Android 5.0 (Lollipop) |
| <input type="checkbox"/> Wear | Minimum SDK | API 21: Android 5.0 (Lollipop) |
| <input type="checkbox"/> Glass (Not Installed) | Minimum SDK | |

Cancel

Previous

Next

Finish

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Création d'un nouveau projet sur Android Studio

Add an activity to Mobile

Add No Activity

Blank Activity

Blank Activity with Fragment

Fullscreen Activity

Google AdMob Ads Activity

Google Maps Activity

Google Play Services Activity

Login Activity

Master/Detail Flow

Navigation Drawer Activity

Cancel Previous Next Finish

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Création d'un nouveau projet sur Android Studio

Customize the Activity

Creates a new blank activity with an action bar.

Activity Name: `MainActivity`

Layout Name: `activity_main`

Title: `MainActivity`

Menu Resource Name: `menu_main`

Blank Activity

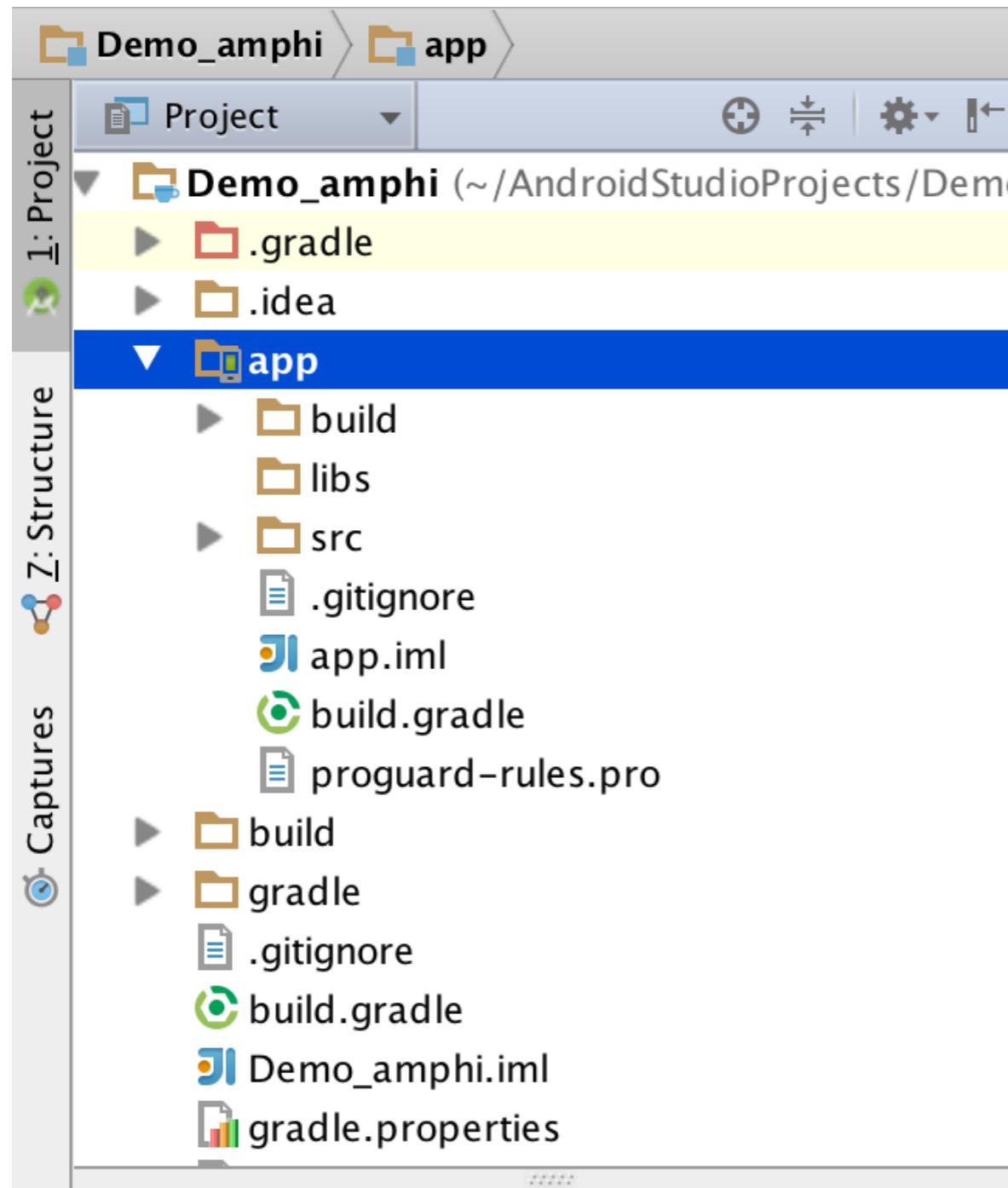
The name of the activity class to create

Cancel Previous Next Finish

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Projet Androïd Studio : Vue Project - Affichage Project



Tous les fichiers / répertoires de votre projet

.idea

- settings intelliJ

app

- fichiers et répertoires des modules de l'application
- dans le paquet .apk

build

- sorties du build pour tous les modules du projet

gradle

- fichiers gradle

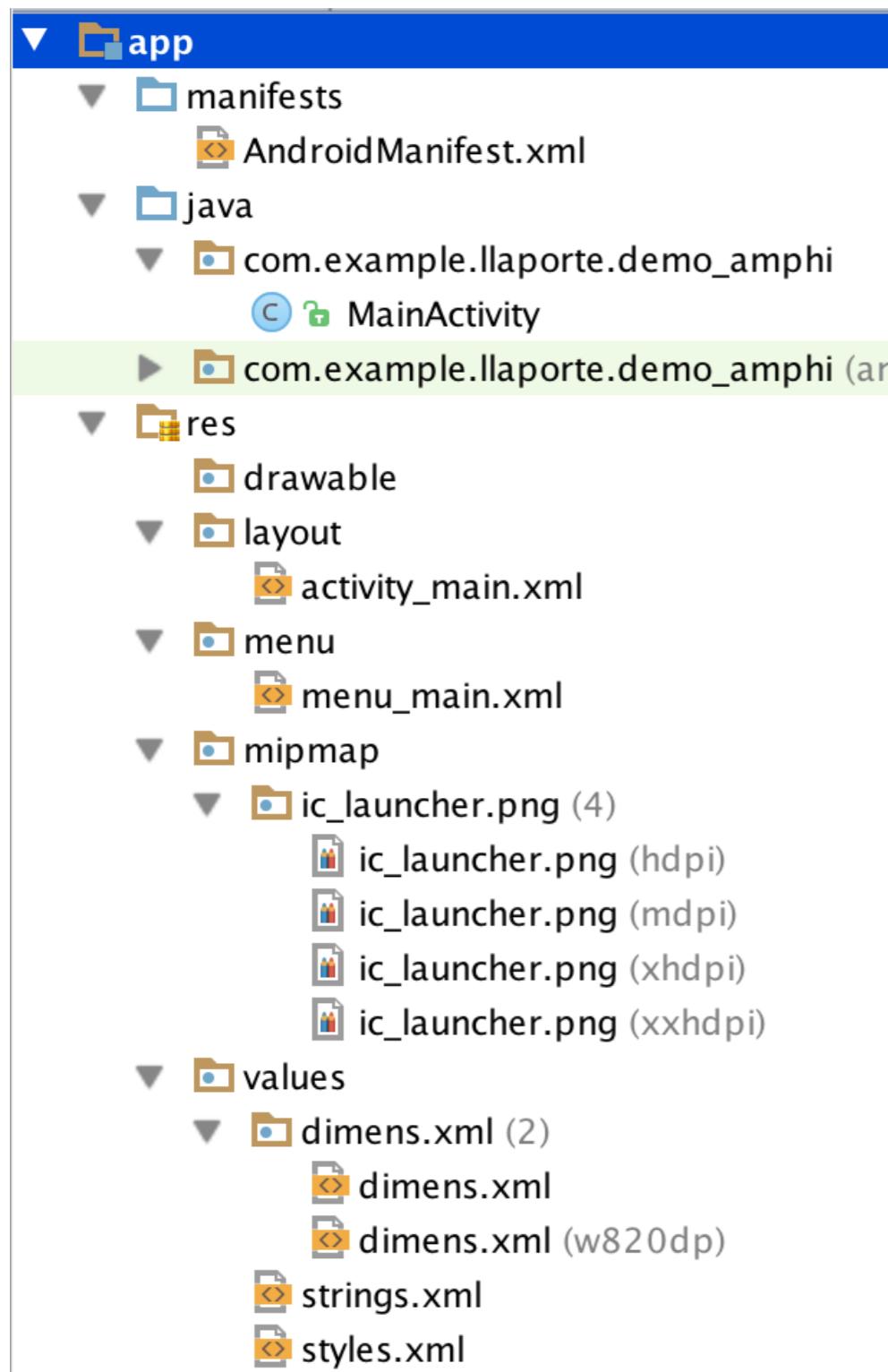
Ancienne vue des projets Android

Contenu du projet tel qu'il est sur le disque dur, avec tous les fichiers

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Projet Androïd Studio : Vue Project - Affichage Androïd



Vue simplifiée

Organisation suivant les parties les plus importantes

manifests

- contient le fichier Manifest.xml qui déclare les éléments de l'application

java

- contient les classes Java de votre application

res

- contient les différentes ressources de l'application
- regroupement dans des dossiers par types de ressource

gradle

- contient les scripts graille
- gradle = outil d'assemblage des dépendances du projet

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Éléments d'une application Android

| Nom | Description | Classe ou package |
|--|---|-----------------------------------|
| Activité | Partie de l'application présentant une vue à l'utilisateur | android.app.Activity |
| Service | Composant en tâche de fond sans vue associée (mise à jour de sources de données/activités, déclenchement de notification) | android.app.Service |
| Fournisseur de contenu (content provider) | Gestion et partage de contenus entre applications | android.content.ContentProvider |
| Gadgets / app widgets | Composés graphiques installés sur le bureau Android | android.appwidget.* |
| Intents | Pour la diffusion de messages demandant la réalisation d'une action à d'autres applications | android.content.Intent |
| Récepteurs d'Intents | Pour permettre que l'applications soit à l'écoute des autres afin de répondre aux Intents qui lui sont destinés | android.content.BroadcastReceiver |
| Notifications | Signalement d'une information à un utilisateur sans interrompre ses actions en cours | android.app.Notification |

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Manifest.xml

Racine du Manifest

Déclare l'ensemble des éléments de l'application

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.llaporte.demo_amphi" >
```

Nom du package de l'application, doit être unique

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Demo_amphi"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="Demo_amphi" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```



Déclarez ici les autres activités, les services, les Intents, etc

```
        <service> ... </service>
        <provide> ... </provider>
    </application>
```

```
</manifest>
```

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Manifest et <uses-sdk>

À la création du projet sur Android Studio :

- Choix de l'API minimale avec laquelle l'application est compatible
- L'application sera compatible avec les API égales ou supérieures

Choix de l'API :

- Fonction des versions majoritaires en cours

Balise <uses-sdk> du fichier Manifest : pour définir la compatibilité, spécifie les API

```
<uses-sdk android:minSdkVersion="integer"
          android:targetSdkVersion="integer"
          android:maxSdkVersion="integer" />
```

- android:minSdkVersion
 - Niveau d'API minimal requis pour faire tourner l'application.
 - Si non renseigné, 1 par défaut : compatibilité avec toutes les versions.
 - Si application non compatible avec tout , alors crash pour les versions non compatibles
- android:targetSdkVersion
 - Version cible de l'API, sur laquelle vous avez testé l'application
- android:maxSdkVersion
 - Version maximale d'API sur laquelle l'application peut tourner
 - developer.android.com : « declaring this attribute is not recommended »
 - Risque que l'application soit retirée des application de l'utilisateur après une mise à jour

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Les ressources

Fichiers contenant des informations qui ne sont ni du JAVA (pas de code) ni dynamique (la ressource est inchangée pendant toute la durée de l'exécution de l'application)

- ⇒ Pas de code
- ⇒ Ressources inchangée pendant toute la durée d'exécution de l'application

Ex : image, texte des messages, etc

Ressources souvent déclarées en XML

Permet notamment l'adaptation de l'interface

- ⇒ Affichage similaire quelque soit le type d'écran (définition de ressources pour chaque résolution ou taille d'écran)
- ⇒ Internationalisation de l'interface (gestion des différentes langues)

Chaque type de ressource est associé à un répertoire particulier

- res/drawable
- res/layout
- ...

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Types de ressources

drawable

- fichiers de types images

layout

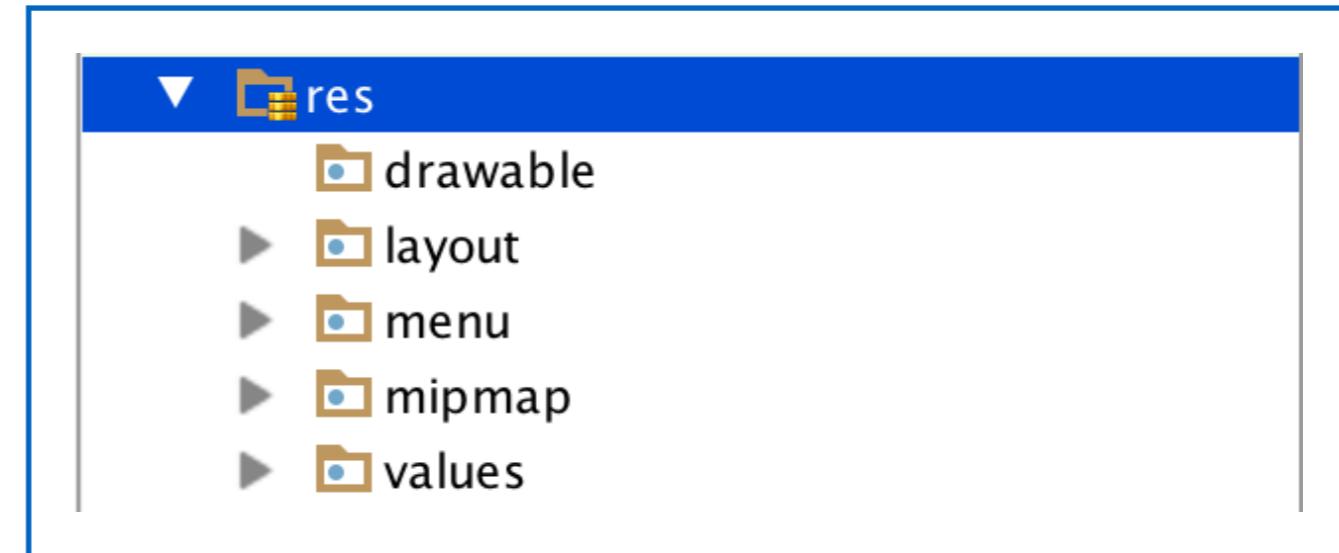
- fichiers xml de layout pour l'interface
- fichiers xml décrivant les gabarits
- affichage et disposition des éléments graphiques
- définition des widgets

menu

- fichiers xml définissant le contenu des menus contextuels
- titre de chaque élément
- icône/image

mipmap

- Icônes de l'application



values

- Éléments statiques de l'application pouvant être réutilisés
- Déclarés et décrits dans des fichiers xml
 - dimens.xml : dimension/taille
 - strings.xml : textes et chaînes de caractères
 - styles.xml : thèmes pour les différents éléments de l'application

D'autres ressources sont disponibles : voir la doc sur developer.android.com

II. Principes généraux Android

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Quantificateur de ressources

Dossiers Android Studio =dossiers génériques pour le cas par défaut

Adaptation de l'interface = adaptation des ressources utilisées

- pour les différentes langues
- pour les différents matériels
- etc

Dossiers et fichiers de ressources distincts pour les différentes adaptations

- ➔ Création de sous-dossiers
- ➔ Distinction par utilisation de quantificateur

Nommage des dossiers avec quantificateurs

- res/<type_de_ressource>-<quantifieur 1>-<quantifieur 2>
- Combinaison de quantificateur possibles, mais il faut respecter un certain ordre
- Exemples : res/drawable-hdpi, res/values-es

Si pas de dossier spécialisé, Android Studio utilise généralement les ressources du dossier par défaut

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Types de quantificateurs

- Langue et région
 - fr pour france
 - fr-rCA pour français du Québec (le -r précise la région)
- Taille de l'écran
 - small, normal, large (tablette), xlarge (tv)
- Orientation écran :
 - port (portrait) ou land (landscape)
- Résolution écran (en dpi-dots per inch)
 - ldpi (120), mdpi (160), hdpi (240), xhdpi (320), xxhdpi (480), xxxhdpi (640)
 - pas de redimensionnement des images matricielles
 - si pas de répertoire correspondant, va chercher dans celui qui a la plus grande résolution en priorité
- Version android (priorité 14)
 - niveau de l'API(v3, v5, v7, v13,...)

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Androïd Studio : Création de ressources

New Resource Directory

Directory name: values-fr-rCA

Resource type: values

Source set: main

Available qual... Chosen qualif... Language: Specific Region Only:

- Country Code
- Network Code
- Layout Direct**
- Smallest Screen
- Screen Width
- Screen Height
- Size
- Ratio
- Orientation
- UI Mode
- Night Mode
- Density
- Touch Screen
- Keyboard
- Text Input

- fr,CA

>> <<

- el: Greek
- en: English
- eo: Esperanto
- es: Spanish
- et: Estonian
- eu: Basque
- fa: Persian
- ff: Fulah
- fi: Finnish
- fj: Fijian
- fo: Faroese
- fr: French**
- fy: Western Frisian
- ga: Irish

- Any Region
- FR: France
- BE: Belgium
- BF: Burkina Faso
- BI: Burundi
- BJ: Benin
- BL: St. Barthélemy
- CA: Canada**
- CD: Congo – Kinshasa
- CF: Central African Republic
- CG: Congo – Brazzaville
- CH: Switzerland
- CI: Côte d'Ivoire
- CM: Cameroon

Tip: Type in list to filter

Show All Regions

Cancel OK

II. Principes généraux Androïd

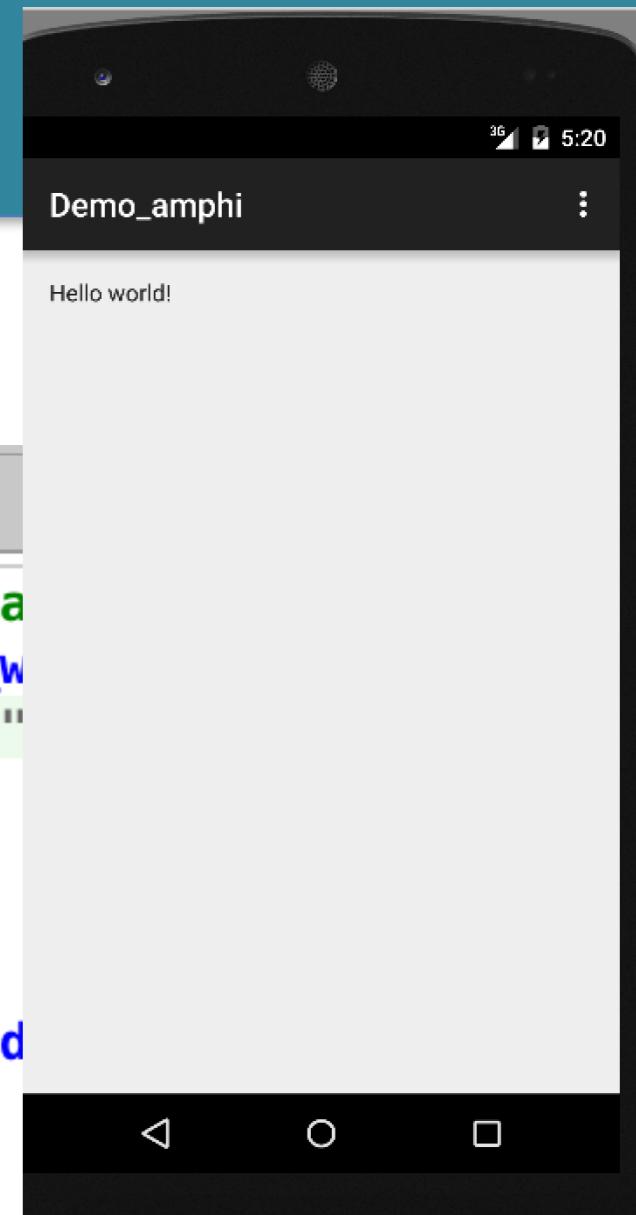
1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Accès aux ressources

- Utilisation des ressources via la classe statique R (R.java)
 - Contient déclarations de classes et constantes
 - Re-générée automatiquement à chaque changement dans le projet
 - Pas de modification manuelle de la classe
- Ressources accessibles via la classe R dès que
 - déclarées dans le fichier XML existant
 - fichier déposé dans la bonne classe
- Ressources accessibles via leur identifiant
 - android.R.type_ressource.nom_ressource
 - type int
- Accès à une ressource de type X et de nom Y (dans un fichier XML)
 - Syntaxe : @X/Y

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités



Accès aux ressources

```
MainActivity.java x activity_main.xml x AndroidManifest.xml x

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity">

    <TextView android:text="@string/hello_world" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

```
MainActivity.java x activity_main.xml x strings.xml x AndroidManifest.xml x
```

Edit translations for all locales in the translations editor.

```
<resources>
    <string name="app_name">Demo_amphi</string>

    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Les activités

Activités et cycle d'une activité = base de la programmation Android.

Une activité

- permet à l'utilisateur de réaliser une tâche
- crée une fenêtre dans laquelle on va placer les éléments de l'interface
 - ➔ utilisation de la méthode : `setContentView(View)`

Généralement présenté à l'utilisateur comme des fenêtres plein écran mais

- possibilité d'utilisation en tant que fenêtre flottante
- possibilité d'inclusion dans une autre activité (`ActivityGroup`)

Gestion des activités via une pile :

- Activité active = activité du dessus de la pile
- Dès qu'une activité arrive sur le dessus de la pile, l'activité qui était en cours est mise en pause
- Elle ne sera réactivée que lorsqu'elle sera de nouveau sur le dessus de la pile

Activité peut être détruite si consomme trop de mémoire

Activité prend différents états au cours de son cycle de vie

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

États d'une activité

Active
(*running*)

L'activité est en haut de la pile
Elle est affichée sur l'écran

En pause
(*paused*)

L'activité a perdu le focus mais est toujours visible
Ex : une activité ne prenant pas tout l'écran est en cours
Reste attachée à la fenêtre
Peut être détruite dans des cas extrême où trop peu de mémoire

Arrêtée
(*stopped*)

L'activité a perdu le focus : une autre activité est en cours
N'est plus visible à l'écran
Peut être tuée si besoin de mémoire

Détruite
(*killed*)

L'activité en pause ou arrêtée peut être tuée/détruite par le système
Conserve les informations, le contexte d'avant sa destruction
Peut être relancée : restauration à l'état d'avant la destruction

II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Cycle de vie des activités

Plus de méthodes que d'états de l'activité

Les méthodes peuvent être surchargées

Cycle de vie complet

Débute avec le `onCreate()`

S'achève avec le `onDestroy()`

`onCreate()`

Initialisation de notre activité

Création de la fenêtre principale de l'IHM et des vues

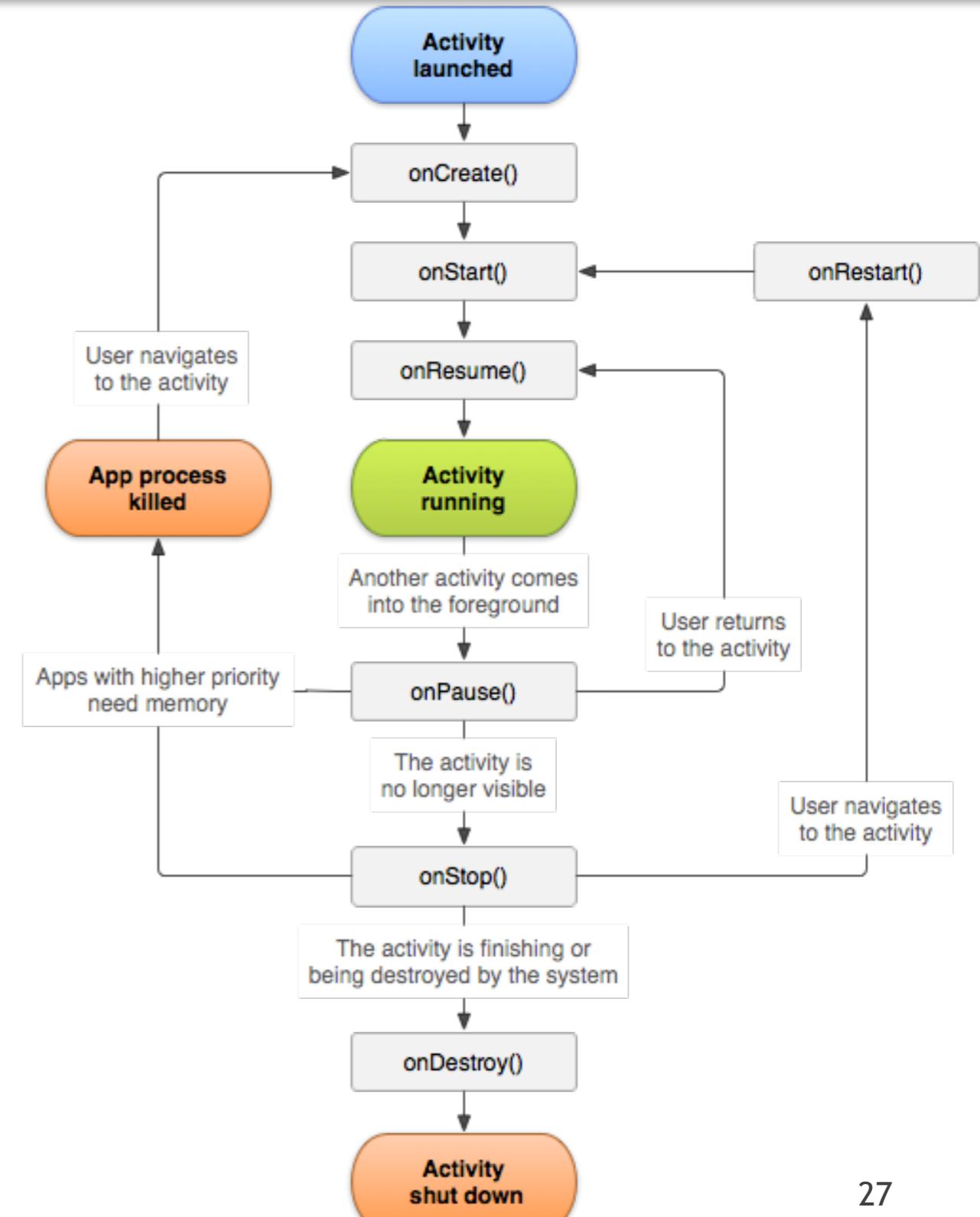
Récupération des widgets

Récupération du contexte (Bundle)

`onDestroy()`

Survient à la demande du programmeur ou si le système détruit l'activité pour gagner de la mémoire

Libération des ressources



II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Cycle de vie des activités

Cycle visible de l'activité

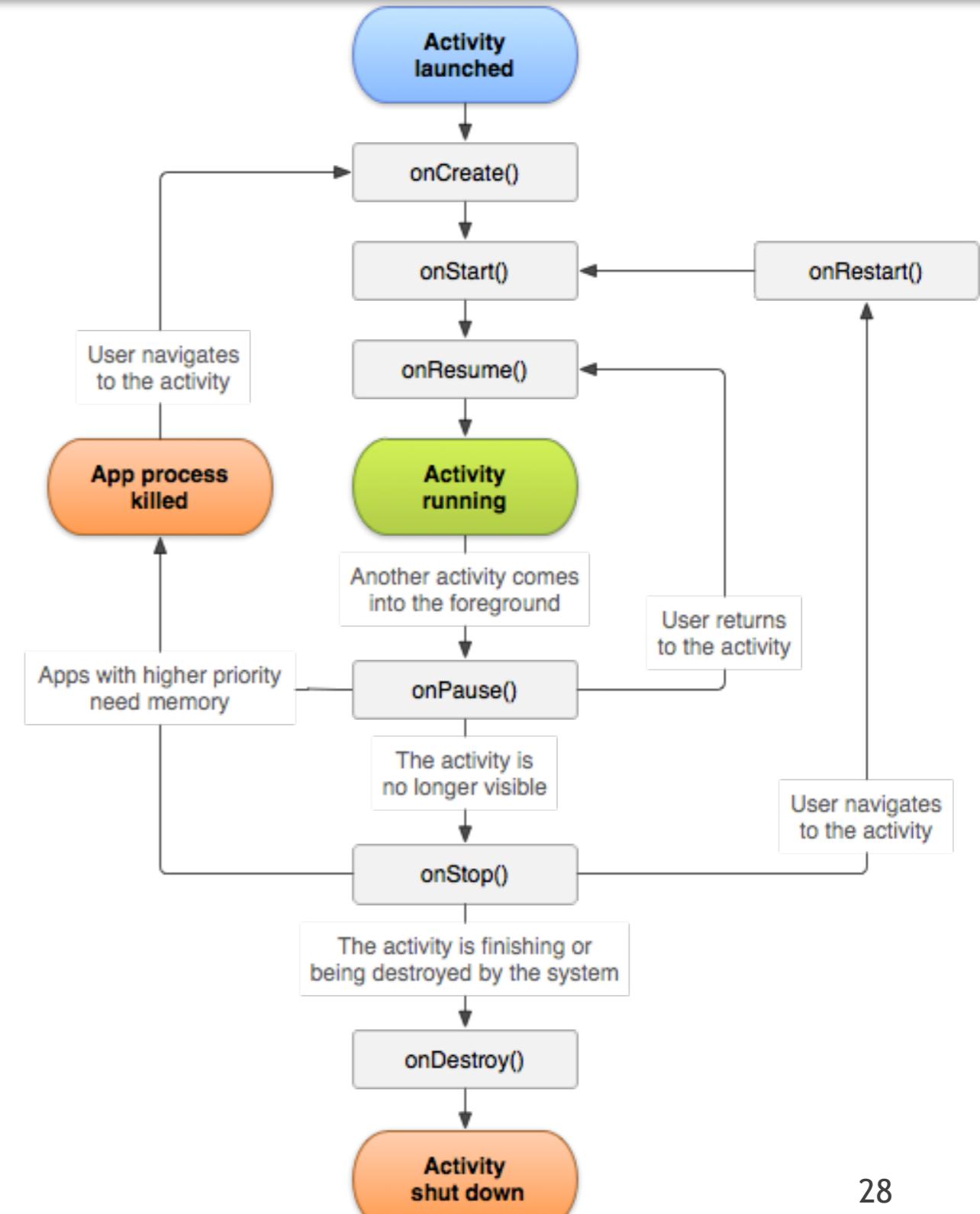
Tout ce qui se passe entre un appel de `onStart()` et l'appel correspondant `onStop()`

`onStart()` et `onStop()` peuvent être appelées plusieurs fois

L'activité est visible sur l'écran mais n'est pas active, pas d'interaction avec l'utilisateur

Maintenance des ressources nécessaires pour montrer l'activité à l'utilisateur

Contient le **cycle de vie actif** de l'activité



II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Cycle de vie des activités

Cycle de vie actif de l'activité

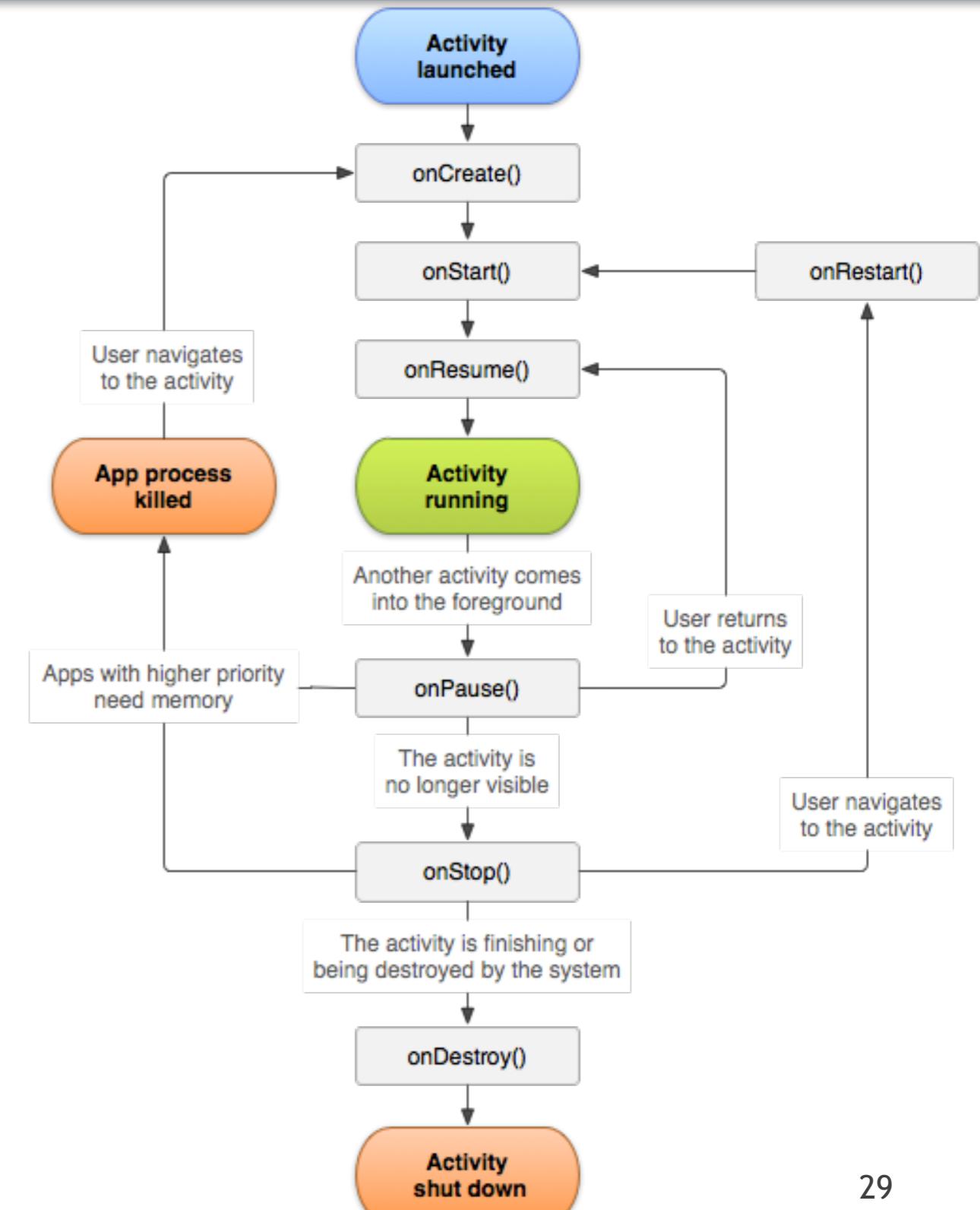
Tout ce qui se passe entre un appel de `onResume()` et l'appel correspondant `onPause()`

`onResume()` et `onPause()` peuvent être appelées plusieurs fois

L'activité a le focus, est visible sur l'écran et interagit avec l'utilisateur

`onResume()` : activité est effectivement sur le haut de la pile

`onPause()` : appelée avant de relancer une activité. Utilisée pour sauvegarder les changements dans les données, arrêter les traitements consommant de la mémoire. Doit être achevée pour pouvoir relancer l'activité



II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Résumé sur les méthodes du cycle de vie

`onCreate()` /
`onDestroy()` Gestion des opérations à effectuer avant l'affichage et lorsque l'activité est complètement détruite de la mémoire. Peu de code dans `onCreate()` pour permettre un lancement rapide

`onStart()` /
`onStop()` Appelées quand l'activité vient visible/invisible

`onResume()` /
`onPause()` Appelées quand l'activité devient active/inactive
l'exécution de `onPause()` doit être rapide

`onRestart()` Appelée quand on relance une activité passée par un `onStop()`
Permet de distinguer un démarrage d'un redémarrage

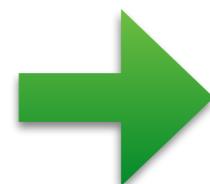
II. Principes généraux Androïd

1. Présentation générale
2. Android SDK
3. Ressources
4. Activités

Sauvegarde et récupération du contexte d'une activité

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

À quoi ça sert ?



Permet de restaurer les valeurs des interfaces d'une activité qui a été déchargée de la mémoire

Exemple :

- Déchargement de la mémoire pour gagner des ressources quand l'activité est stoppée
- Au redémarrage, il faut pouvoir retrouver les valeurs (par exemple, saisie dans des zones de texte, etc)

Si pas d'identifiant (par exemple au niveau des widgets)

- ➔ pas de sauvegarde des valeurs possibles
- ➔ pas de restauration possible via l'objet Bundle

Pas de restauration possible si l'application est totalement détruite

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Rappel : les vues

Unité de base de l'interface graphique

- Viennent se greffer sur les activités
- Fournissent du contenu visuel avec lequel il est possible d'interagir

Possible de disposer les vues à l'aide de conteneur, comme dans JAVA

Les gabarits, les widgets, ... héritent de la classe View (package android.view.View)

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Identification des gabarits et des vues

- Comme pour les ressources, on peut utiliser des identifiants pour les gabarits et les widgets
 - Création des identifiants dans le fichier XML de l'activité
- Pour créer un identifiant,
 - attribut -> android:id
 - valeur -> de la forme @+X/Y
 - le + signifie que l'identifiant n'est pas encore défini
- La classe R est re-générée pour prendre en compte les nouveaux identifiants
- On se servira des identifiants pour récupérer les évènements

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Récupération des gabarits et des widgets

- Identifiants layout et widgets utilisables dans le code Java
- Pour récupérer l'identifiant d'une vue
 - Utilisation de la méthode public View findViewById(int id)
 - Méthode renvoie un objet View
 - il faut la caster dans le type de destination (transtypage)
- Pour indiquer qu'une activité utilise un certain layout
 - utilisation de void setContentView (View view)
 - ex : setContentView(R.layout.activity_main) indique que la vue utilisée est celle définie dans le main_activity
 - Attention il faut accéder à la vue et à ces éléments qu'après le setContentView
 - Si 2 setContentView(), seul le dernier est considéré

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Constitution des interfaces graphiques sous Androïd

Principe

Exploiter toutes les opportunités offertes par les ressources pour que ça fonctionne sur la majorité des terminaux

Deux modes de programmation

- Programmatif : description des écrans dans le code Java
 - Plus flexible
 - Plus difficile de séparer les codes (de l'interface, de l'application)
- Déclaratif : description des écrans dans le code XML
 - Un fichier XML permettant de décrire l'interface pour chaque activité
 - Meilleure séparation au niveau du code

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Constitution des interfaces graphiques sous Androïd

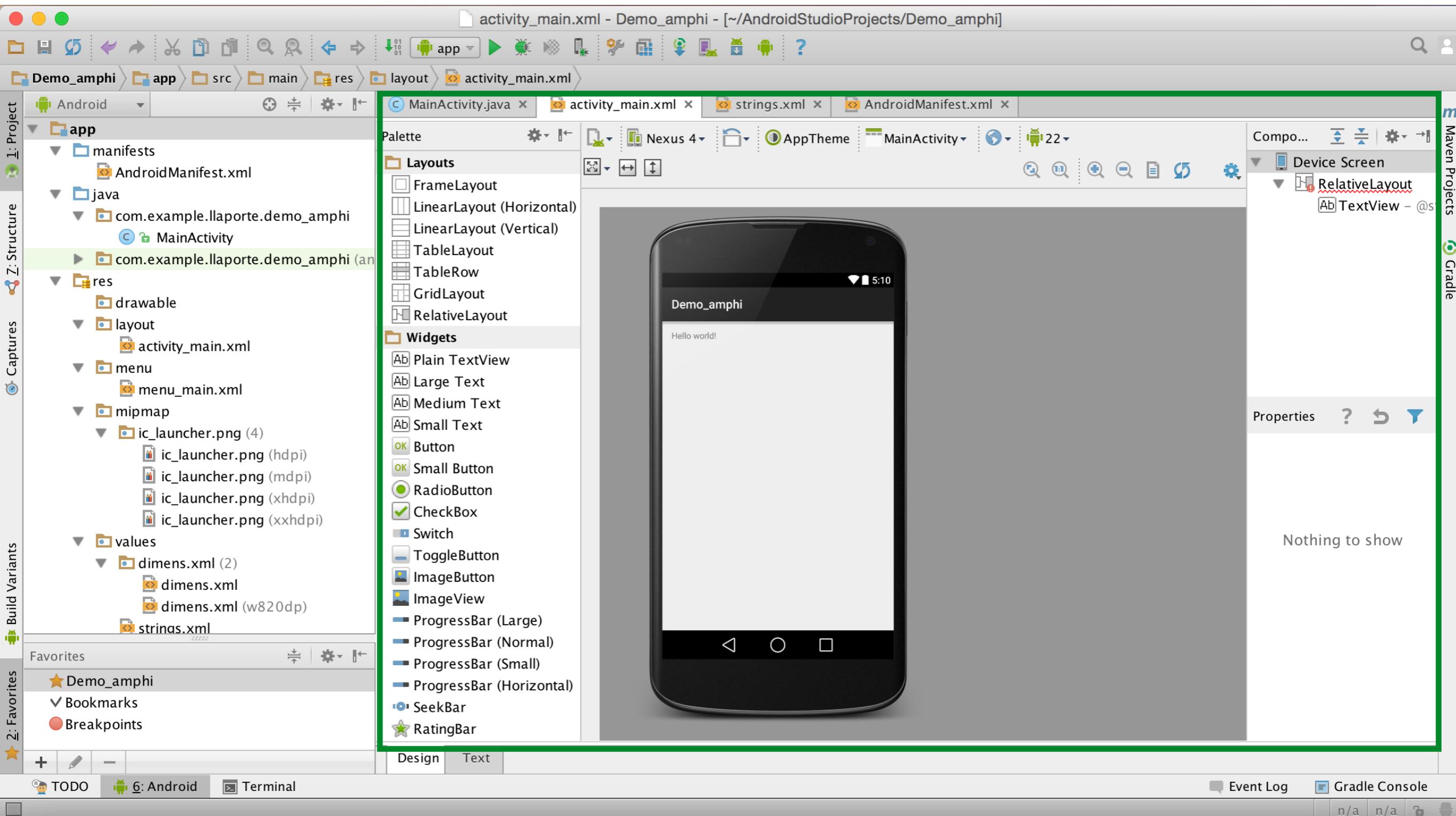
Androïd Studio et générateur d'interfaces

- Permet de construire l'écran en positionnant directement les widgets
- Présente un canvas qui affiche le rendu visuel
- Génération et modification automatique du fichier XML associé
- Peut être buggé
- Pas forcément très pratique / précis
- Ne remplace pas la modification direct du code XML
- Onglet « Design » sur fichier XML

III. Conception d'interfaces simples sous Android

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

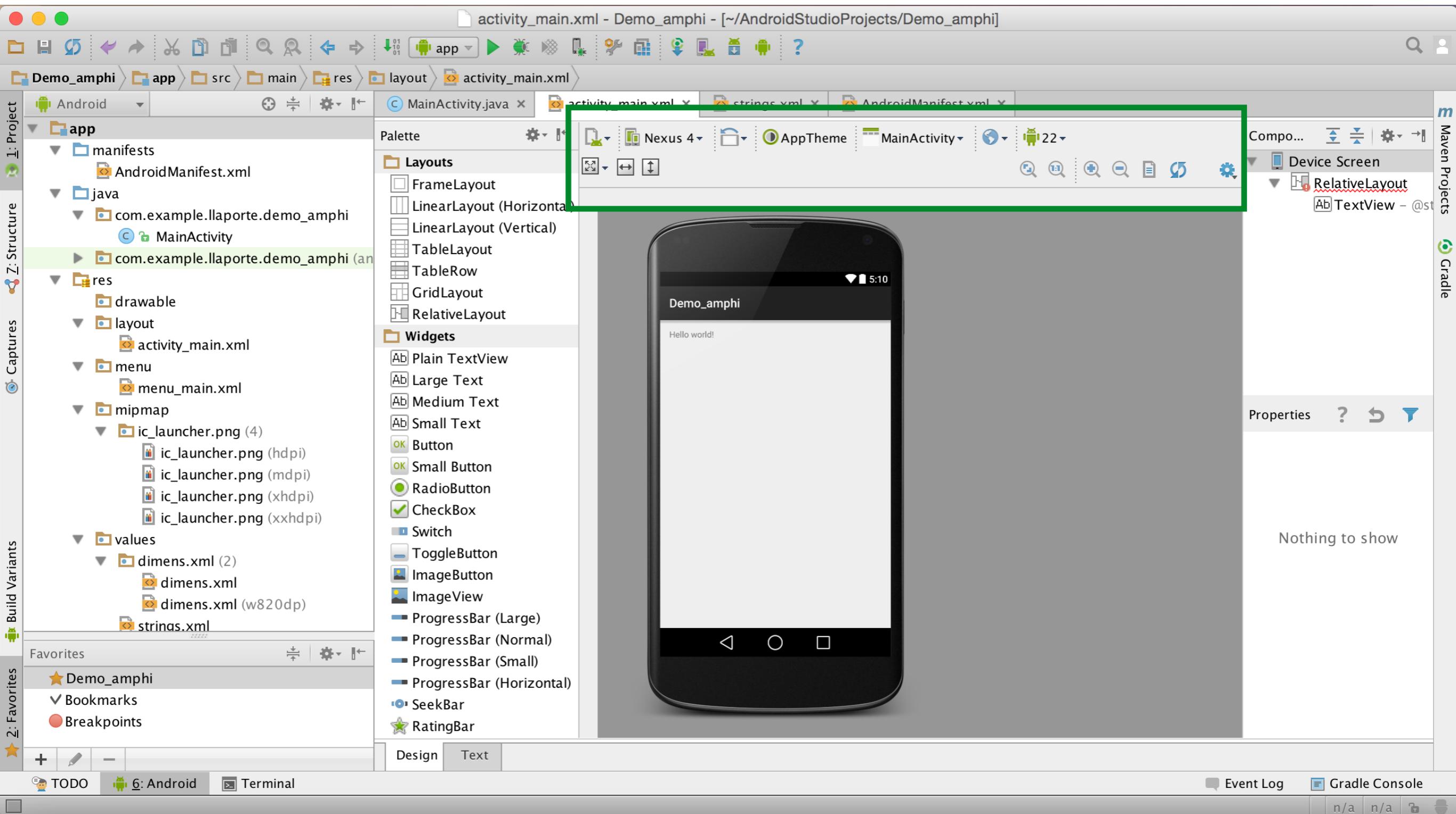
Générateur d'interfaces



III. Conception d'interfaces simples sous Android

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

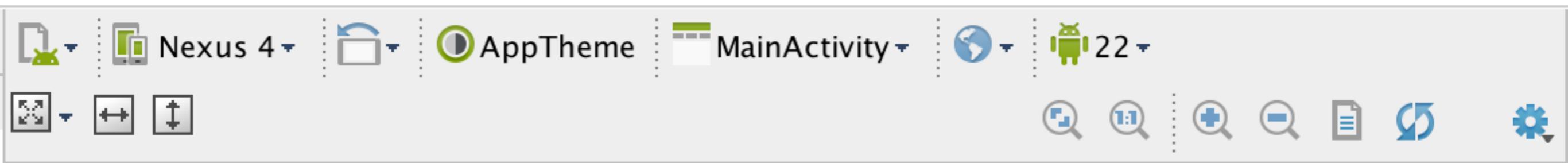
Générateur d'interfaces - Barre de menu



III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Générateur d'interfaces - Barre de menu

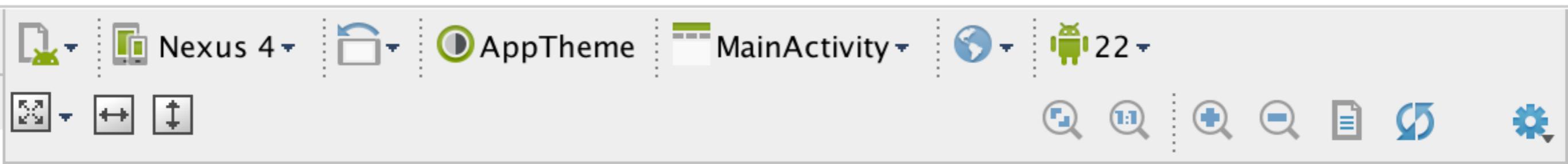


| | |
|--|---|
| | <ul style="list-style-type: none">• Alternance entre les différents écrans/fichiers XML différenciés• Création des fichiers XML différenciés• Affichage simultané des différents aperçus de l'interface |
| | <ul style="list-style-type: none">• Sélection du terminal qui contiendra la visualisation (type du smartphone/de la tablette)• Modification de la résolution pour voir comment l'interface s'adapte.• Utilisation du fichier de ressource avec le quantificateur le plus adapté |
| | <p>Modification de l'aperçu selon trois facteurs :</p> <ul style="list-style-type: none">• orientation : portrait / paysage• Mode jour/nuit• Connexion à un dock, television, watch, etc |

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Générateur d'interfaces - Barre de menu



| | |
|---------------------|---|
| AppTheme | <ul style="list-style-type: none">• Modification du thème (material dark, material light, theme perso, etc) |
| MainActivity | <ul style="list-style-type: none">• Fait le lien avec le fichier Java• Permet d'associer l'écran à une autre activité |
| | <ul style="list-style-type: none">• Choix de la langue de l'interface (si présence de quantificateur permettant l'internationalisation) |
| 22 | <ul style="list-style-type: none">• Vérification du comportement suivant la version de l'API (si quantificateur pour l'API) |

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Générateur d'interfaces - Barre de menu



Modification de certaines propriétés du widget/layout

- Alignement horizontal/vertical: gauche, droit, centré, haut, bas
- Remplissage horizontal
- Remplissage vertical

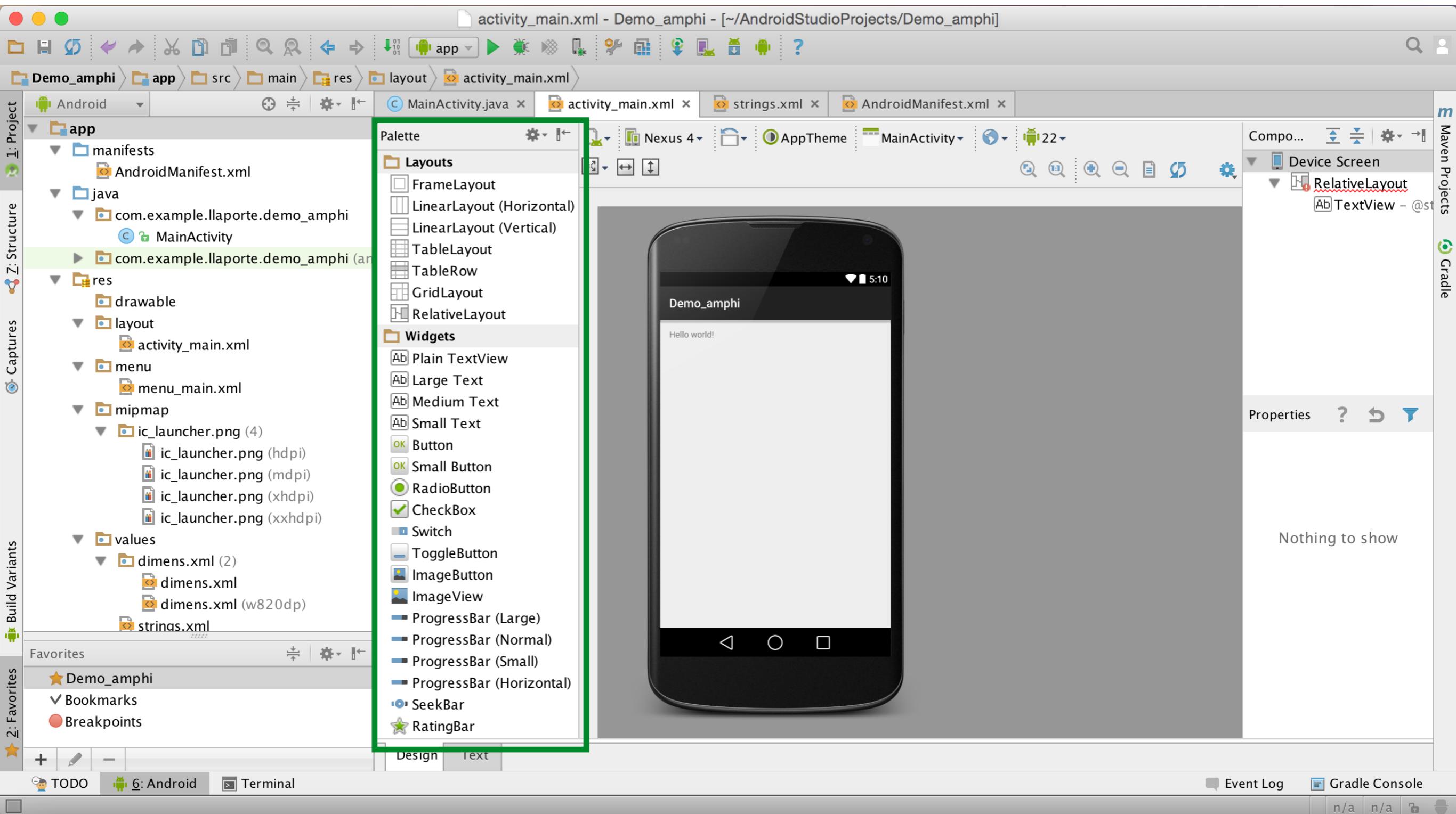
Modification de l'affichage

- Mettre le device à la taille de la fenêtre
- Mettre le device à sa taille réelle
- Agrandir
- Réduire
- Afficher le code source
- Rafraîchir

III. Conception d'interfaces simples sous Android

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Générateur d'interfaces - Palette



III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Exemple de fichier XML pour une activité

Racine du fichier XML, indique le type de gabarit (layout)

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".MainActivity"
    android:id="@+id/"/>

<TextView android:text="@string/hello_world" android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Attributs permettant d'indiquer qu'on a un fichier pour Android. Obligatoire.

Description d'un widget de type TextView

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Qu'est-ce-qu'un layout ?

- Layout = gabarit
 - Vue spéciale pouvant contenir d'autres vues
 - Vue englobante = parent ; vues contenues = enfants
 - N'est pas destiné à fournir du contenu ou des contrôles à l'utilisateur.
 - Dispose les vues d'une certaine façon
- Layout hérite de ViewGroup (classe abstraite), héritant elle même de View.
 - Un layout peut contenir d'autres ViewGroup
 - Un layout peut contenir d'autres layout (imbrication de layout)
- Une vue ne pouvant pas en englober d'autres est un widget (composant)

III. Conception d'interfaces simples sous Androïd

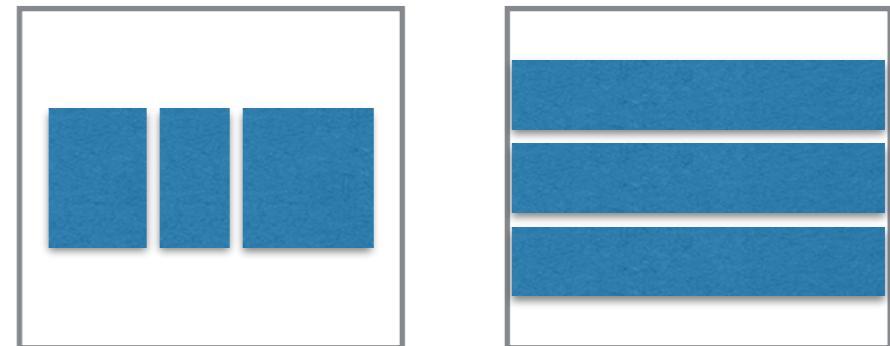
1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Les différents types de layout

LinearLayout

Dispose les éléments

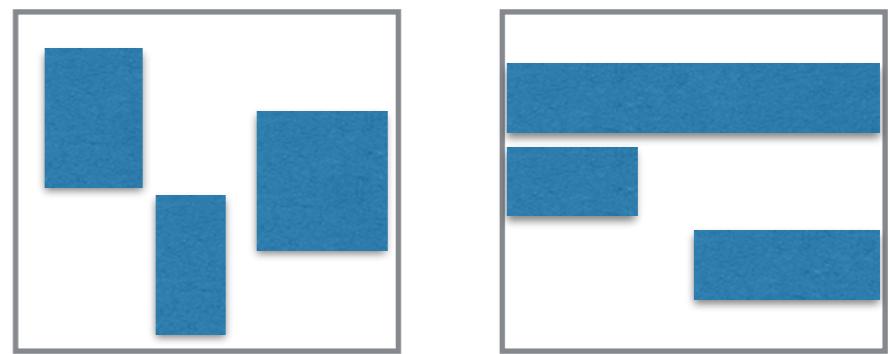
- sur une seule ligne, de gauche à droite ou
- sur une seule colonne de haut en bas



RelativeLayout

Dispose les éléments enfants

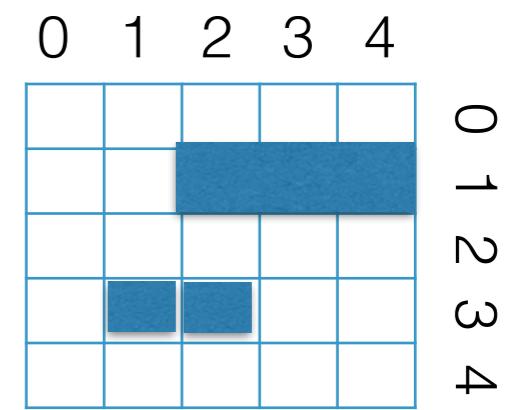
- les uns par rapport aux autres ou
- par rapport à l'élément parent



TableLayout

Dispose les éléments enfants de façon matricielle

- une ligne contient des cellules
- possibilité de merger des lignes/colonnes



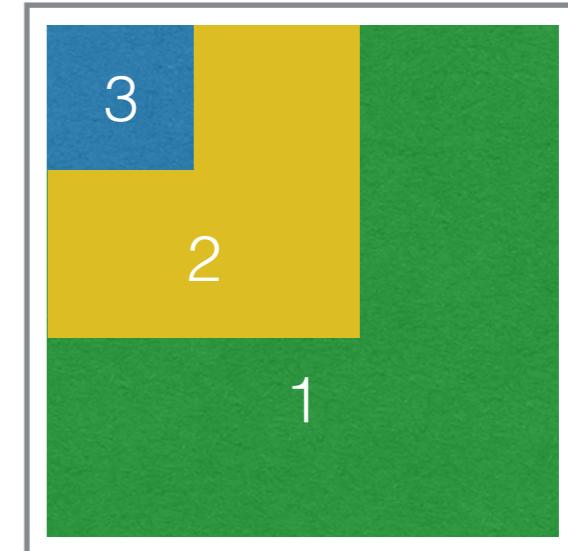
III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Les différents types de layout

FrameLayout

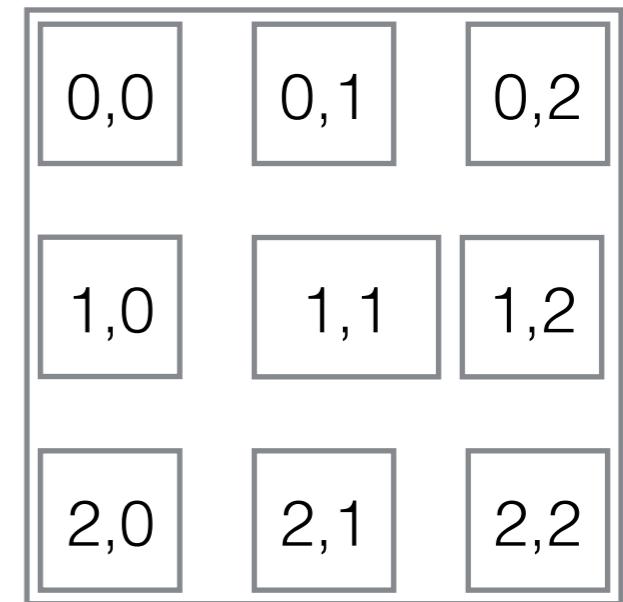
- Dispose les éléments les un au dessus des autres (pile)
- Bloque une zone de l'écran pour un item



GridLayout

Dispose les éléments enfants de façon matriciel

- Nombre de colonnes N fixé
- Nombre de lignes infinies
- Disponible depuis IceCreamSandwich
- Modification de la taille de la cellule, pas de la colonne entière



III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Déclaration XML et attributs

- Les layout sont déclarés en XML, dans le fichier correspondant à l'activité
- Ils peuvent bien entendu être placés via le générateur

Quelques attributs pour les layout

- android:layout_width et android:layout_height
 - = « *match_parent* » : élément enfant remplit l'élément parent
 - = « *wrap_content* » : élément enfant prend juste la taille nécessaire pour le contenu
 - = « *fill_parent* » : deprecated depuis API 8
- android:gravity
 - définit l'alignement (center, left, right, etc)
- android:orientation
 - définit l'orientation de l'empilement (portrait ou paysage)

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Les widgets

- Widget = élément de base permettant d'afficher du contenu ou d'interagir avec l'utilisateur
- Ne gèrent pas la mise en forme
 - effectué par la layout
- Grand nombre de widget avec beaucoup d'attributs
 - TextView
 - EditText
 - Button
 - CheckBox
 - RadioButton / RadioGroup
 - ...

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

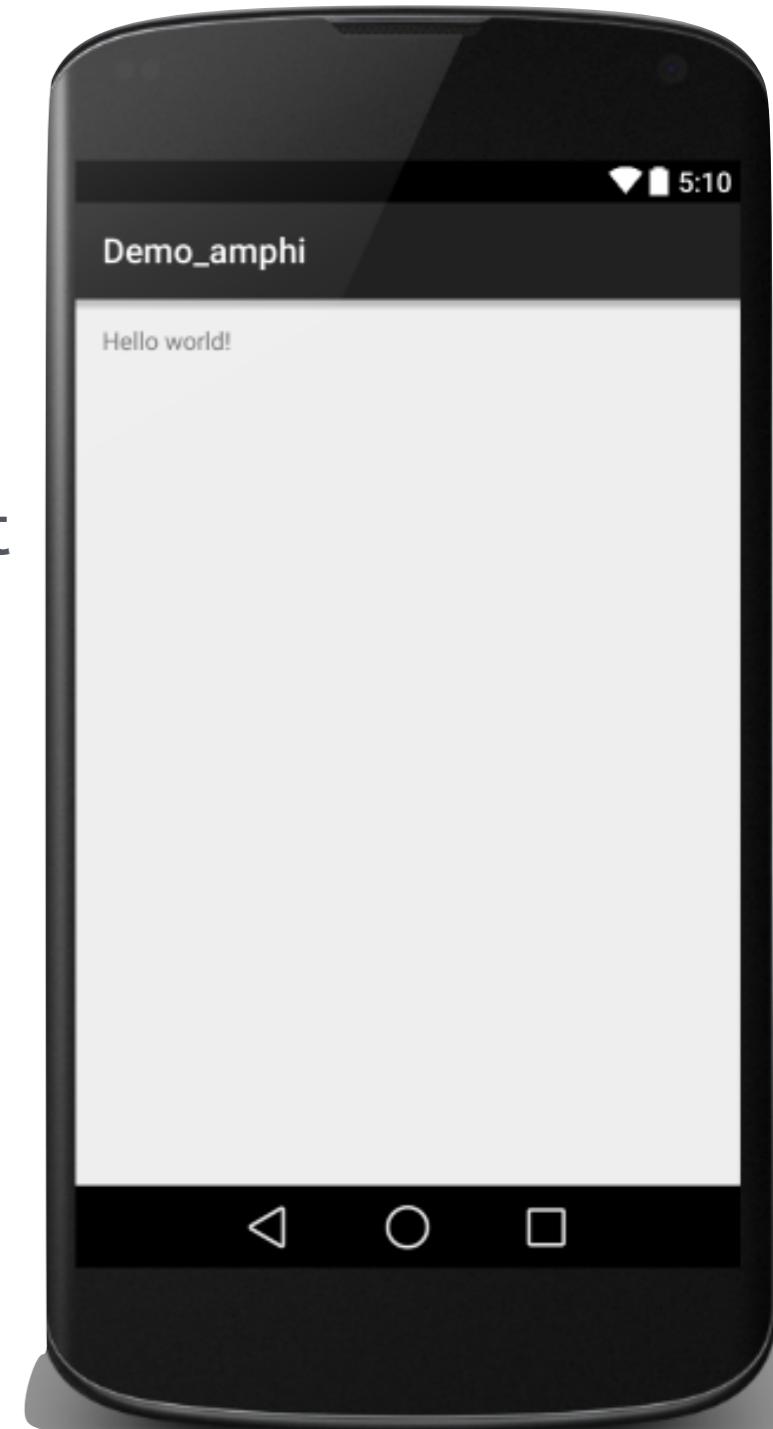
TextView

- Affichage de labels textuels
- Par défaut, le texte affiché n'est pas éditable
- Pour un texte éditable, un champ de saisie : EditText

Création d'un TextView en XML

```
<TextView android:text="@string/hello_world"  
         android:layout_width="wrap_content"  
         android:layout_height="wrap_content" />
```

- Modification possible du style, de la taille, de l'alignement, etc



III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
- 3. Les composants graphiques (widgets)**
4. Gestion des évènements

EditText

- Champs de saisie textuelle
- TextView éditable
 - hérite de TextView
 - peut prendre les mêmes attributs
- Exemple d'attributs :
 - android:text : remplit le champ avec un texte
 - android:hint : affiche un texte d'indication
 - android:inputType : indique le type de texte (multiline, ...)
 - android:lines : indique le nombre de lignes

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. **Les composants graphiques (widgets)**
4. Gestion des évènements

EditText

Création d'un EditView en XML avec android:text

```
<EditText android:text="@string/taille"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:id="@+id/taille" />
```

Création d'un EditView en XML avec android:hint

```
<EditText android:hint="@string/taille"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="numberDecimal"
    android:id="@+id/tailleHint" />
```



III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Button

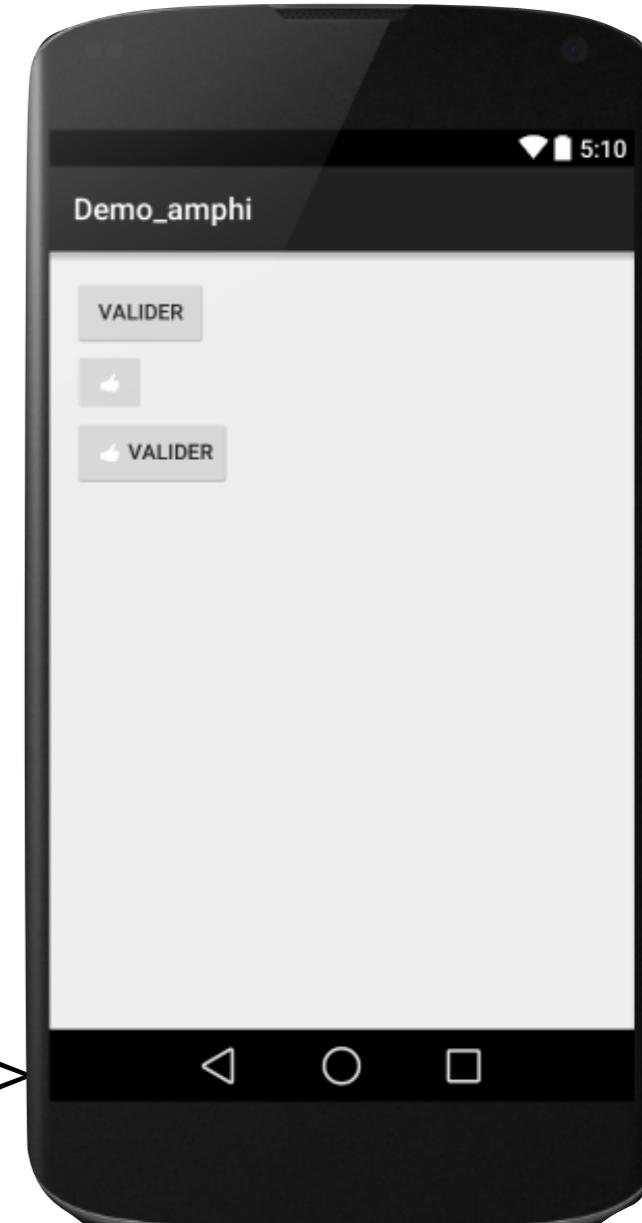
- Affiche un bouton
- TextView cliquable : hérite de TextView
 - Mêmes attributs, mêmes méthodes

Création de boutons avec texte seul, icône seule, texte + icône

```
<Button android:text="Valider"  
        android:id="@+id/Button01"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"/>
```

```
<ImageButton android:text="Valider"  
            android:id="@+id/ButtonImage01"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:src="@drawable/button_icone"/>
```

```
<Button android:text="Valider"  
        android:id="@+id/Button03"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:drawableLeft="@drawable/button_icone"/>
```



III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
- 3. Les composants graphiques (widgets)**
4. Gestion des évènements

CheckBox

- Case pouvant être dans deux états : cochée ou non
- Permet à l'utilisateur de choisir une ou plusieurs options
- Utilisable dans les listes
- Hérite de Button : mêmes attributs, mêmes méthodes
- `android:checked = true` signifie que la case est cochée par défaut

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. **Les composants graphiques (widgets)**
4. Gestion des évènements

CheckBox

Création de CheckBox en XML

```
<CheckBox android:id="@+id/checkbox_english"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/english"
    android:checked="true"/>
<CheckBox android:id="@+id/checkbox_french"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/french"
    android:checked="true"/>
<CheckBox android:id="@+id/checkbox_german"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/german" />
<CheckBox android:id="@+id/checkbox_spanish"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/spanish" />
```



III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. **Les composants graphiques (widgets)**
4. Gestion des évènements

RadioButton

- Bouton radio plutôt que checkBox (un seul bouton pourra être coché)
- A regrouper dans un RadioGroup
- Hérite de Button

RadioGroup

- Layout permettant de regrouper plusieurs boutons radio
- Un seul bouton radio peut être coché à la fois
- Utilisé uniquement avec RadioButton
- Sous-classe de LinearLayout avec une orientation vertical par défaut

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

RadioButton & RadioGroup

Création d'un RadioGroup en XML

```
<RadioGroup xmlns:android=<> http://schemas.android.com/apk/res/  
android >  
    android:id="@+id/group"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:orientation=<> vertical">  
        <RadioButton android:id="@+id/radio_homme"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@string/homme" />  
        <RadioButton android:id="@+id/radio_femme"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:checked="true"  
            android:text="@string/femme" />  
</RadioGroup>
```



III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Récupération des évènements via des listeners

- Représentation des interactions sous forme d'évènements
- Listener : objet qui va détecter l'évènement
- Définition de méthodes de callback qui seront appelé lorsque l'évènement associé se produira
 - Ex : onClick(), etc
- Récupération des widgets pour lesquels on doit surveiller les évènements
 - Se fait dans la classe onCreate()
 - Utilisation de la méthode findViewById(View view)

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Récupération des vues dans le onCreate()

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    //Ici, on récupère les vues dont on a besoin  
    valider = (Button)findViewById(R.id.Button01);  
  
    taille = (EditText)findViewById(R.id.taille);  
  
    english = (CheckBox)findViewById(R.id.english);  
    french = (CheckBox)findViewById(R.id.french);  
  
    group = (RadioGroup)findViewById(R.id.group);  
}
```

III. Conception d'interfaces simples sous Androïd

1. Notions générales
2. Les gabarits (layouts)
3. Les composants graphiques (widgets)
4. Gestion des évènements

Récupération d'évènements

Clic sur un bouton

```
valider.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v){  
        // Actions to do on click  
    }  
});
```

Saisie dans une zone de texte

```
taille.addTextChangedListener(new TextWatcher() {  
    @Override  
    public void onTextChanged(CharSequence s, int start, int before,  
    int count){  
        // Actions to do  
    }  
});
```



Developers

Design

Develop

Distribute

Developer Console



Up and running with material design

Android uses a new design metaphor inspired by paper and ink that provides a reassuring sense of tactility. Visit the [material design](#) site for more resources.

- › Introducing material design
- › Downloads for designers
- › Articles



Training API Guides Reference Tools Google Services Samples

Get Started with Android Studio

Everything you need to build incredible app experiences on phones and tablets, Wear, TV, and Auto.

- › Set up Android Studio
- › Build your first app
- › Learn about Android
- › Sample projects



Google Play

The premier store for distributing Android apps and games, with global reach and tools to help you gain traction in the marketplace.

Overview



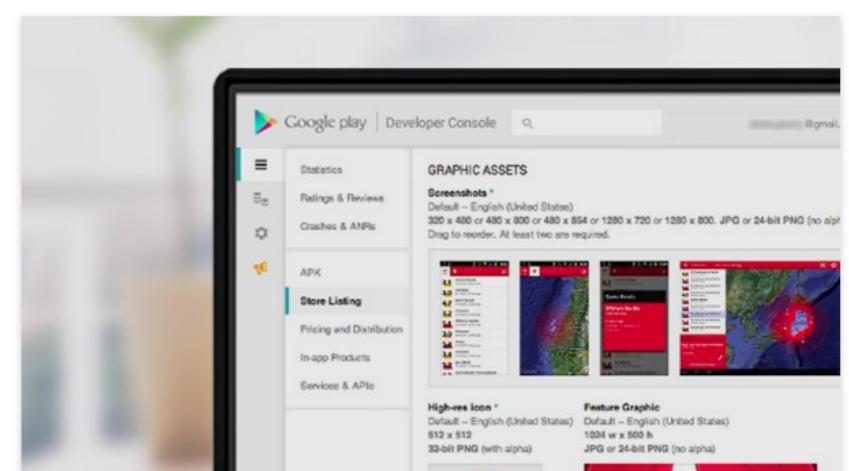
The Google Play Opportunity

Billions of downloads a month and growing.
Get your apps in front of users at Google's scale.



Get Started with Publishing

Start publishing on Google Play in minutes by registering for a developer account.



Developer Console

Learn about the Developer Console, your home for app publishing on Google Play.

Pour s'entraîner

Application de calcul de l'IMC

- Un champ de saisie pour la taille (en mètre)
- Un champ de saisie pour le poids (en kg)
- Un bouton de validation
- Un text qui affiche :
 - un message disant de cliquer sur valider
 - après le clic, la valeur de l'IMC

$$\text{IMC} = \text{poids} / \text{taille}^2$$

Pour s'entraîner

Application de calcul de l'IMC

